ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

ОСНОВЫ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Конспект лекций

Самара

ЛИТЕРАТУРА

1. Отраслевой стандарт ОСТ 45.127-99 «Система обеспечения информационной безопасности Взаимоувязанной сети связи Российской Федерации. Термины и определения».
2. Мельников В. П., Клейменов С. А., Петраков А. М. Информационная безопасность и защита информации. Высшее профессиональное образование. М.: Академия, 2009 г.
3. Несторов С.А. Информационная безопасность и защита информации: Учебное пособие, СПб, 2009.
4. Шаньгин В. Ф. Информационная безопасность компьютерных систем и сетей: учебное пособие, М.: ИД «ФОРУМ» 2008.
5. Официальный сайт Российской газеты. Доктрина информационной безопасности Российской Федерации. М., 1998-2013.

6. ОСНОВНЫЕ ПОНЯТИЯ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ (ИБ)

1.1.Постановка задачи сетевой безопасности

В области безопасности информационных систем выделяют две основные группы:

- безопасность компьютера,
- сетевая безопасность.

Если компьютер рассматривается как автономная система, то к первой группе относят все задачи защиты данных, хранящихся и обрабатываемых компьютером. К ним относятся: защита ОС, БД. Ко второй группе относятся вопросы **сетевой безопасности**, т.е. защита данных при их передаче по каналам связи. Сюда же относятся защита от несанкционированного доступа в сеть (НСД). Обе группы задач тесно связаны и трудно отделимы друг от друга. Однако очевидно, что проблемы сетевой безопасности имеют специфические особенности. Автономный компьютер можно эффективно защитить от покушений разными способами, например, запереть клавиатуру на замок, снять винчестер и поместить его в сейф.

Сетевой компьютер принципиально не в состоянии отгородиться от внешнего мира, т.к. он должен общаться с другими компьютерами, даже удаленными на значительные расстояния. Поэтому обеспечение безопасности в сети является более сложной задачей. Вход чужого пользователя (даже законного пользователя сети) в личный компьютер является штатной ситуацией, если компьютер находится в сети. Обеспечение сетевой безопасности заключается в том, чтобы сделать чужое проникновение **контролируемым**, т.е. каждому пользователю сети должны быть четко определены его права по доступу к информации, внешним устройствам и выполнению системных действий на каждом компьютере сети. Другой проблемой сетевой безопасности является перехват и анализ передаваемых сообщений, модификация (искажение) сообщений, а также создание «ложного» трафика. Значительная часть средств обеспечения сетевой ИБ направлена на предотвращение именного этого типа нарушений.

Вопросы сетевой ИБ приобретают особую значимость в настоящее время, когда происходит бурное развитие корпоративных сетей, которые для своего функционирования используют сети общего пользования, в частности Интернет. Провайдеры услуг сетей общего пользования редко обеспечивают защиту пользовательских данных в процессе их транспортировки по своим магистралям, возлагая заботы по ИБ на самих пользователей.

1.2.Основные понятия информационной безопасности

Безопасность сети – это защита ее от случайного или преднамеренного вмешательства в нормальный процесс функционирования, а также от попыток хищения, изменения или разрушения ее компонентов. Безопасность сети достигается принятием мер по обеспечению конфиденциальности и це-

лостности обрабатываемой информации, а также доступности и целостности компонентов и ресурсов сети.

Доступ к информации – это ознакомление с информацией, ее обработка, в частности копирование, модификация или уничтожение информации.

Различают **санкционированный** и **несанкционированный** доступ к информации.

Санкционированный доступ – это доступ, не нарушающий установленные правила разграничения доступа.

Несанкционированный доступ (НСД) – это доступ, характеризующийся нарушением установленных правил разграничения доступа.

Лицо или процесс, осуществляющий НСД, является **нарушителем** правил разграничения доступа.

НСД является наиболее распространенным видом компьютерных нарушений.

Конфиденциальность данных – это статус, предоставляемый данным и определяющий степень их защиты. Конфиденциальность информации – это свойство быть известной только допущенным и прошедшим проверку (авторизированным) субъектам системы (пользователям, процессам, программам). Для остальных субъектов это информация должна быть закрыта.

Субъект – это активный компонент системы, который может стать причиной потока информации от объекта к субъекту или изменения состояния системы.

Объект – это пассивный компонент системы, хранящий, принимающий или передающий информацию. Доступ к объекту означает доступ к содержащейся в нем информации.

Целостность информации обеспечивается в том случае, если данные в системе не отличаются в семантическом отношении от данных в исходных документах, т.е. если не произошло их случайного или преднамеренного искажения или разрушения.

Целостность компонента или **ресурса** системы – это свойство компонента или ресурса быть неизменным в семантическом смысле при функционировании системы в условиях случайных или преднамеренных искажений или разрушающих воздействий.

Доступность компонента или **ресурса системы** – это свойство компонента или ресурса быть доступным для авторизированных законных субъектов системы.

Под **угрозой безопасности** сети понимаются возможные воздействия на сеть, которые прямо или косвенно могут нанести ущерб ее безопасности.

Ущерб безопасности подразумевает нарушение состояния защищенности информации, содержащейся и обрабатывающейся в сети.

Уязвимость сети – это некоторое неудачное свойство системы, которое делает возможным возникновение и реализацию угрозы.

Атака на сеть – это действие, предпринимаемое злоумышленником, которое заключается в поиске и использовании той или иной уязвимости системы. Иными словами, **атака** – это реализация угрозы безопасности.

Целью защиты системы обработки информации является противодействие угрозам безопасности.

Безопасная или защищенная система – это система со средствами защиты, которые успешно и эффективно противостоят угрозам безопасности.

Комплекс средств защиты – это совокупность программных и аппаратных средств, создаваемых и поддерживаемых для обеспечения ИБ. Комплекс создается и поддерживается в соответствии с принятой в данной организации политикой безопасности.

Политика безопасности – это совокупность норм, правил и практических рекомендаций, регламентирующих работу средств защиты сетей от заданного множества угроз безопасности.

1.3. Классификация угроз безопасности корпоративных сетей

Для описания угроз безопасности корпоративных сетей предлагается следующая классификация (рисунок 1.1).

По характеру воздействия

а) пассивное (класс 1.1);

б) активное (класс 1.2).

Пассивным воздействием на корпоративную сеть (КС) называют воздействие, которое не оказывает непосредственного влияния на работу сети, но способно нарушать ее политику безопасности.

Именно отсутствие непосредственного влияния на работу сети приводит к тому, что пассивное удаленное воздействие практически невозможно обнаружить. Примером типового пассивного удаленного воздействия является прослушивание канала связи в сети. При пассивном воздействии, в отличие от активного, не остается никаких следов (от того, что атакующий просмотрит чужое сообщение в системе, ничего не изменится).

Под активным воздействием на КС понимается воздействие, оказывающее непосредственное влияние на работу сети (изменение конфигурации КС, нарушение работоспособности и т. д.) и нарушающее принятую в ней политику безопасности. Практически все типы удаленных атак являются активными воздействиями. Это связано с тем, что в самой природе разрушающего воздействия заложено активное начало. Очевидным отличием активного воздействия от пассивного является принципиальная возможность его обнаружения (естественно, с большими или меньшими усилиями), так как в результате его осуществления в системе происходят определенные изменения.

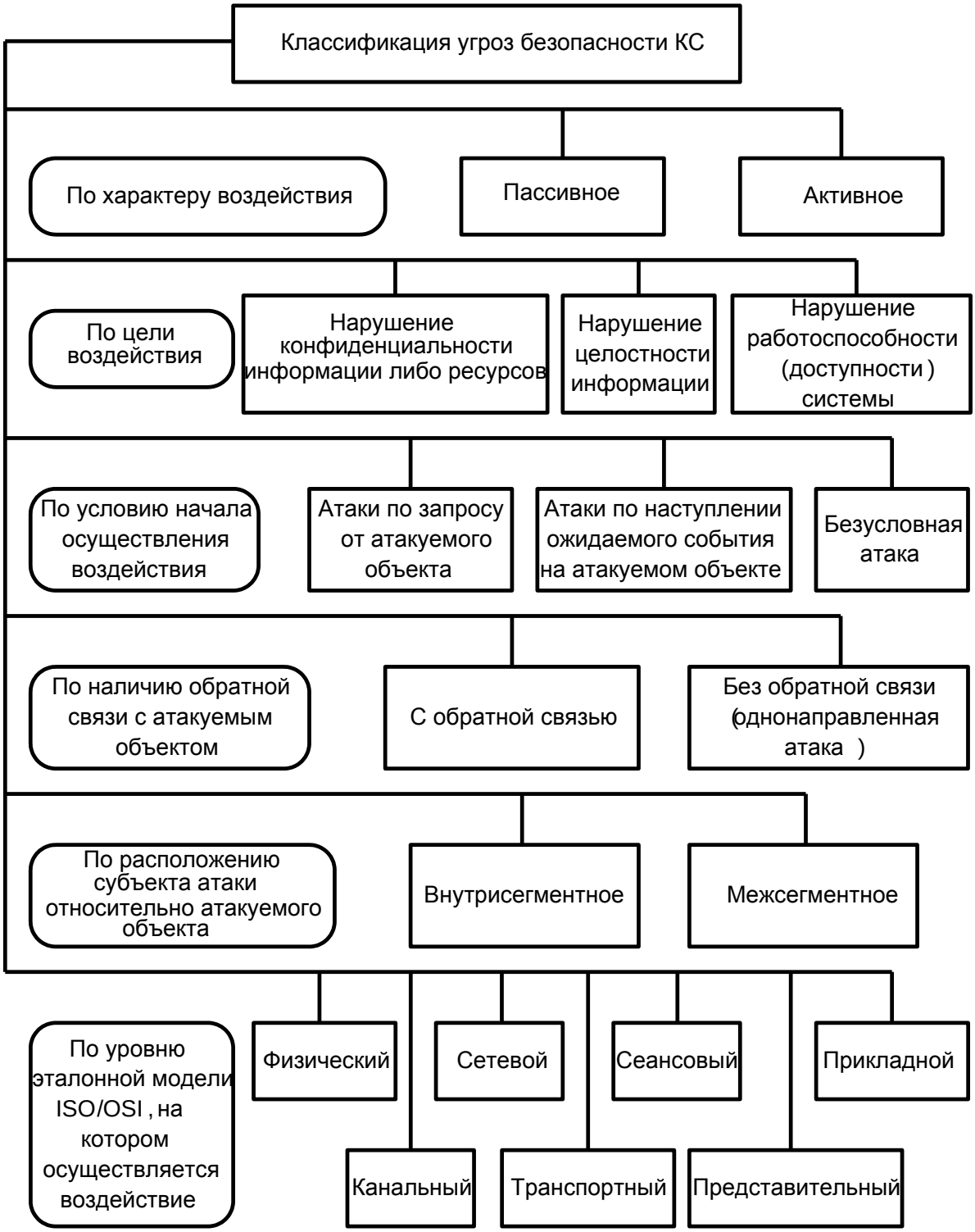


Рисунок 1.1 – Классификация угроз безопасности корпоративных сетей

По цели воздействия:

а) нарушение конфиденциальности информации либо ресурсов сети (класс 2.1);

б) нарушение целостности информации (класс 2.2);

в) нарушение работоспособности (доступности) системы (класс 2.3).

Цель большинства атак - получить несанкционированный доступ к информации. Существуют две принципиальные возможности такого доступа: перехват и искажение. Перехват – это получение информации без возмож-

ности ее искажения. Примером перехвата может служить прослушивание канала в сети. Такая атака

является пассивным воздействием и ведет к нарушению **конфиденциальности информации**.

Искажение информации означает полный контроль над информационным потоком между объектами системы или возможность передачи сообщений от имени другого объекта. Очевидно, что искажение информации ведет к нарушению ее целостности, то есть представляет собой активное воздействие. Примером удаленной атаки, цель которой - нарушение целостности информации, может служить типовая удаленная атака (УА) "ложный объект КС".

Принципиально иной целью атаки является нарушение работоспособности системы. В этом случае основная цель взломщика - добиться, чтобы операционная система на атакованном объекте вышла из строя и, следовательно, для всех остальных объектов системы доступ к ресурсам данного объекта был бы невозможен. Примером удаленной атаки с целью нарушения работоспособности системы, это типовая удаленная атака (УА) "ложный объект КС".

По условию начала осуществления воздействия

Удаленное воздействие, как и любое другое, может начать осуществляться только при определенных условиях. Существуют три вида таких условий:

а) атака после запроса от атакуемого объекта (класс 3.1);

б) атака после наступления ожидаемого события на атакуемом объекте (класс 3.2);

в) безусловная атака (класс 3.3).

В первом случае взломщик ожидает передачи от потенциальной цели атаки запроса определенного типа, который и будет условием начала осуществления воздействия. Примером подобных сообщений в ОС Novell NetWare может служить запрос SAP, а в Internet -запросы DNS и ARP. Такой тип удаленных атак наиболее характерен для КС.

При осуществлении атаки класса 3.2 атакующий осуществляет постоянное наблюдение за состоянием операционной системы объекта атаки и при возникновении определенного события в этой системе начинает воздействие. Как и в предыдущем случае, инициатором начала атаки выступает сам атакуемый объект. Примером такого события может быть прерывание сеанса работы пользователя с сервером в ОС Novell NetWare без выдачи команды LOGOUT.

При безусловной атаке ее начало не зависит от состояния системы атакуемого объекта, то есть воздействие осуществляется немедленно. Следовательно, в этом случае его инициатором является атакующий.

По наличию обратной связи с атакуемым объектом

а) с обратной связью (класс 4.1);

б) без обратной связи, или однонаправленная атака (класс 4.2).

Если взломщику требуется получить ответ на некоторые запросы, переданные на объект воздействия, то есть между атакующим и целью атаки

существует обратная связь, которая позволяет ему адекватно реагировать при изменении ситуации, то такое воздействие можно отнести к классу 4.1.

Инициатор удаленной атаки без обратной связи, напротив, не реагирует ни на какие изменения, происходящие на атакуемом объекте. Воздействие данного вида обычно осуществляется передачей на атакуемый объект одиночных запросов, ответы на которые атакующему не нужны.

По расположению субъекта атаки относительно атакуемого объекта

а) внутрисегментное (класс 5.1);

б) межсегментное (класс 5.2).

Рассмотрим ряд определений.

Субъект атаки, или *источник атаки* - это атакующая программа или оператор, непосредственно осуществляющие воздействие.

Хост (host) – сетевой компьютер.

Маршрутизатор, или *роутер* (router) – устройство, обеспечивающее маршрутизацию пакетов обмена в глобальной сети.

Подсеть (subnetwork в терминологии Internet) – логическое объединение, совокупность хостов, являющихся частью глобальной сети, для которых маршрутизатором выделен одинаковый номер. Хосты внутри одной подсети могут взаимодействовать непосредственно, минуя маршрутизатор.

Сегмент сети (segment) – физическое объединение хостов. Например, сегменты сети образуются совокупностью хостов, подключенных к серверу по схеме "общая шина". При такой схеме подключения каждый хост имеет возможность подвергать анализу любой пакет в своем сегменте.

Для осуществления удаленного воздействия чрезвычайно важно, как по отношению друг к другу располагаются субъект и объект атаки, то есть в одном или в разных сегментах они находятся. В случае внутрисегментной атаки, как следует из названия, субъект и объект атаки находятся в одном сегменте, а при межсегментной – в разных.

Данный классификационный признак позволяет судить о так называемой степени удаленности атаки. Межсегментное воздействие осуществить значительно труднее, чем внутрисегментное, но и опасность оно представляет гораздо большую. В таком случае объект и субъект атаки могут находиться на расстоянии многих тысяч километров друг от друга, что существенно усложняет возможность непосредственного обнаружения атакующего и адекватной реакции на атаку.

По уровню эталонной модели ISO/OSI, на котором осуществляется воздействие

а) физический (класс 6.1);

б) канальный (класс 6.2);

в) сетевой (класс 6.3);

г) транспортный (класс 6.4);

д) сеансовый (класс 6.5);

е) представительный (класс 6.6);

ж) прикладной (класс 6.7).

Несанкционированный доступ (НСД) является наиболее распространенным и многообразным видом компьютерных нарушений. Суть НСД со-

стоит в получении пользователем (нарушителем) доступа к объекту в нарушение правил разграничения доступа, установленных в соответствии с принятой в организации политикой безопасности. НСД использует любую ошибку в системе защиты и возможен при нерациональном выборе средств защиты, их некорректной установке и настройке. НСД может быть осуществлен как штатными средствами, так и специально созданными аппаратными и программными средствами.

Перечислим основные каналы несанкционированного доступа, через которые нарушитель может получить доступ к компонентам сети и осуществить хищение, модификацию и/или разрушение информации:

- все штатные каналы доступа к информации (терминалы пользователей, оператора, администратора системы; средства отображения и документирования информации; каналы связи) при их использовании нарушителями, а также законными пользователями вне пределов их полномочий;
- технологические пульты управления;
- линии связи между аппаратными средствами сети;
- побочные электромагнитные излучения от аппаратуры, линий связи, сетей электропитания и заземления и др.

Рассмотрим некоторые примеры НСД:

- перехват паролей;
- "маскарад";
- незаконное использование привилегий.

Перехват паролей осуществляется специально разработанными программами. При попытке законного пользователя войти в систему программа-перехватчик имитирует на экране дисплея ввод имени и пароля пользователя, которые сразу пересылаются владельцу программы-перехватчика, после чего на экран выводится сообщение об ошибке и управление возвращается операционной системе. Пользователь предполагает, что допустил ошибку при вводе пароля. Он повторяет ввод и получает доступ в систему. Владелец программы-перехватчика, получивший имя и пароль законного пользователя, может теперь использовать их в своих целях. Существуют и другие способы перехвата паролей.

"Маскарад" – это выполнение каких-либо действий одним пользователем от имени другого пользователя, обладающего соответствующими полномочиями. Целью "маскарада" является приписывание каких-либо действий другому пользователю либо присвоение полномочий и привилегий другого пользователя.

Примерами реализации "маскарада" являются:

- вход в систему под именем и паролем другого пользователя (этому "маскараду" предшествует перехват пароля);
- передача сообщений в сети от имени другого пользователя.

"Маскарад" особенно опасен в банковских системах электронных платежей, где неправильная идентификация клиента из-за "маскарада" злоумышленника может привести к большим убыткам законного клиента банка.

Незаконное использование привилегий. Большинство систем защиты устанавливают определенные наборы привилегий для выполнения заданных функций. Каждый пользователь получает свой набор привилегий: обычные пользователи – минимальный, администраторы – максимальный. Несанкционированный захват привилегий, например посредством "маскарада", приводит к возможности выполнения нарушителем определенных действий в обход системы защиты. Следует отметить, что незаконный захват привилегий возможен либо при наличии ошибок в системе защиты, либо из-за халатности администратора при управлении системой и назначении привилегий.

Компьютерные сети характерны тем, что кроме обычных локальных атак, осуществляемых в пределах одной системы, против объектов сетей предпринимают так называемые *удаленные атаки*, что обусловлено распределенностью сетевых ресурсов и информации. Злоумышленник может находиться за тысячи километров от атакуемого объекта, при этом нападению может подвергаться не только конкретный компьютер, но и информация, передающаяся по сетевым каналам связи. Под удаленной атакой понимают информационное разрушающее воздействие на распределенную компьютерную сеть, программно осуществленное по каналам связи [56].

В табл.1.1 показаны основные пути реализации угроз безопасности сети при воздействии на ее компоненты.

Таблица 1.1

Пути реализации угроз безопасности сети

Объекты воздействия	Нарушение конфиденциальности информации	Нарушение целостности информации	Нарушение работоспособности системы
Аппаратные средства	НСД – подключение; ипользование ресурсов; хищение носителей	НСД – подключение; использование ресурсов; модификация, изменение режимов	НСД – изменение режимов; вывод из строя; разрушение
Программное обеспечение	НСД – копирование; хищение; перехват	НСД, внедрение "троянского коня", "вирусов", "червей"	НСД – искажение; удаление; подмена
Данные	НСД – копирование; хищение; перехват	НСД – искажение; модификация	НСД – искажение; удаление; подмена
Персонал	Разглашение; передача сведений о защите; халатность	"Маскарад"; вербовка; подкуп персонала	Уход с рабочего места; физическое устранение

В табл.1.1. приведены специфические названия и термины: "троянский конь", "вирус", "червь", которые употребляются для именования некоторых распространенных угроз безопасности. Хотя эти названия имеют жаргонный оттенок, они уже вошли в общепринятый компьютерный лексикон. Дадим краткую характеристику этих распространенных угроз безопасности.

"Троянский конь" представляет собой программу, которая наряду с действиями, описанными в ее документации, выполняет некоторые другие действия, ведущие к нарушению безопасности системы и деструктивным результатам. Аналогия такой программы с древнегреческим "троянским конем" вполне оправдана, так как в обоих случаях не вызывающая подозрений оболочка таит серьезную угрозу.

Термин "троянский конь" был впервые использован хакером Даном Эдварсом, позднее ставшим сотрудником Агентства Национальной Безопасности США. "Троянский конь" использует, в сущности, обман, чтобы побудить пользователя запустить программу со скрытой внутри угрозой. Обычно для этого утверждается, что такая программа выполняет некоторые весьма полезные функции. В частности, такие программы маскируются под какие-нибудь полезные утилиты.

Опасность "троянского коня" заключается в дополнительном блоке команд, вставленном в исходную безвредную программу, которая затем предоставляется пользователям сети. Этот блок команд может срабатывать при наступлении какого-либо условия (даты, состояния системы) либо по команде извне. Пользователь, запустивший такую программу, подвергает опасности как свои файлы, так и всю сеть в целом. Приведем для примера некоторые деструктивные функции, реализуемые "троянскими конями".

- Уничтожение информации. Выбор объектов и способов уничтожения определяется фантазией автора вредоносной программы.
- Перехват и передача информации. В частности, известна программа, осуществляющая перехват паролей, набираемых на клавиатуре.
- Целенаправленная модификация текста программы, реализующей функции безопасности и защиты системы.

"Троянские кони" наносят ущерб посредством хищения информации и явного разрушения программного обеспечения системы. "Троянский конь" является одной из наиболее опасных угроз безопасности. Радикальный способ защиты от этой угрозы заключается в создании замкнутой среды исполнения программ, которые должны храниться и защищаться от несанкционированного доступа.

Компьютерный "вирус" представляет собой своеобразное явление, возникшее в процессе развития компьютерной и информационной техники. Суть этого явления состоит в том, что программы-вирусы обладают рядом свойств, присущих живым организмам, – они рождаются, размножаются и умирают.

Сетевой "червь" представляет собой разновидность программы-вируса, которая распространяется по глобальной сети и не оставляет своей копии на магнитном носителе. Этот термин используется для именования программ, которые подобно ленточным червям перемещаются по компьютерной сети от одной системы к другой.

Первоначально "черви" были разработаны для поиска в сети других компьютеров со свободными ресурсами, чтобы получить возможность выполнить распределенные вычисления. При правильном использовании технология "червей" может быть весьма полезной. Например, "червь" World

Wide Web Worm формирует индекс поиска участков Web. Однако "червь" легко превращается во вредоносную программу. "Червь" использует механизмы поддержки сети для определения узла, который может быть поражен. Затем с помощью этих же механизмов передает свое тело в этот узел и либо активизируется, либо ждет подходящих условий для активизации.

Сетевые "черви" являются самым опасным видом вредоносных программ, так как объектом их атаки может стать любой из миллионов компьютеров, подключенных к глобальной сети Internet. Для защиты от "червя" необходимо принять меры предосторожности против несанкционированного доступа к внутренней сети. "Троянские кони", компьютерные вирусы и сетевые "черви" относятся к наиболее опасным угрозам. Для защиты от указанных вредоносных программ применяется ряд мер:

- исключение несанкционированного доступа к исполняемым файлам;
- тестирование приобретаемых программных средств;
- контроль целостности исполняемых файлов и системных областей;
- создание замкнутой среды исполнения программ.

1.4. Обеспечение безопасности сетей передачи данных

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

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

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

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

Комплексный подход применяют для защиты сетей крупных организаций или небольших сетей, выполняющих ответственные задачи или обрабатывающих особо важную информацию. Нарушение безопасности информации в сетях крупных организаций может нанести огромный материальный ущерб как самим организациям, так и их клиентам. Поэтому такие организации вынуждены уделять особое внимание гарантиям безопасности и реализовывать комплексную защиту. Комплексного подхода придерживаются

большинство государственных и крупных коммерческих предприятий и учреждений. Этот подход нашел свое отражение в различных стандартах.

1.4.1. Основные виды политики безопасности

Комплексный подход к проблеме обеспечения безопасности основан на разработанной для конкретной сети политике безопасности. **Политика безопасности** представляет собой набор норм, правил и практических рекомендаций, на которых строится управление, защита и распределение информации в сети. Политика безопасности регламентирует эффективную работу средств защиты. Она охватывает все особенности процесса обработки информации, определяя поведение системы в различных ситуациях.

Политика безопасности реализуется посредством административно-организационных мер, физических и программно-технических средств и определяет архитектуру системы защиты. Для конкретной организации **политика безопасности должна носить индивидуальный характер** и зависеть от конкретной технологии обработки информации и используемых программных и технических средств.

Политика безопасности определяется способом управления доступом, определяющим порядок доступа к объектам системы. Различают два основных вида политики безопасности: **избирательную и полномочную**.

1. **Избирательная** политика безопасности основана на избирательном способе управления доступом. *Избирательное (или дискреционное) управление доступом* характеризуется заданным администратором множеством разрешенных отношений доступа (например, в виде троек <объект, субъект, тип доступа>). Обычно для описания свойств избирательного управления доступом применяют математическую модель на основе матрицы доступа.

Матрица доступа представляет собой матрицу, в которой столбец соответствует объекту системы, а строка – субъекту. На пересечении столбца и строки матрицы указывается тип разрешенного доступа субъекта к объекту. Обычно выделяют такие типы доступа субъекта к объекту, как "доступ на чтение", "доступ на запись", "доступ на исполнение" и т.п. Матрица доступа является самым простым подходом к моделированию систем управления доступом. Она является основой для более сложных моделей.

Избирательная политика безопасности широко применяется в коммерческих сетях, так как ее реализация соответствует требованиям коммерческих организаций по разграничению доступа и подотчетности, а также имеет приемлемую стоимость.

2. **Полномочная** политика безопасности основана на полномочном (мандатном) способе управления доступом. *Полномочное (или мандатное) управление доступом* характеризуется совокупностью правил предоставления доступа в зависимости от **метки конфиденциальности информации** и уровня допуска пользователя. Полномочное управление доступом подразумевает, что:

a. все субъекты и объекты системы однозначно идентифицированы;

b. каждому объекту системы присвоена метка конфиденциальности информации, определяющая ценность содержащейся в нем информации;
c. каждому субъекту системы присвоен определенный уровень допуска, определяющий максимальное значение метки конфиденциальности информации объектов, к которым субъект имеет доступ.

Чем важнее объект, тем выше его метка конфиденциальности. Поэтому наиболее защищенными оказываются объекты с наиболее высокими значениями метки конфиденциальности.

Основным назначением полномочной политики безопасности является регулирование доступа субъектов системы к объектам с различными уровнями конфиденциальности, предотвращение утечки информации с верхних уровней должностной иерархии на нижние, а также блокирование возможных проникновений с нижних уровней на верхние.

Помимо управления доступом субъектов к объектам системы проблема защиты информации имеет еще один аспект. Для получения информации о каком-либо объекте системы совсем необязательно искать пути несанкционированного доступа к нему. Необходимую информацию можно получить, наблюдая за обработкой требуемого объекта, т.е. используя каналы утечки информации. В системе всегда существуют информационные потоки. Поэтому администратору необходимо определить, какие информационные потоки в системе являются "легальными", т.е. не ведут к утечке информации, а какие – ведут к утечке. Поэтому возникает необходимость разработки правил, регламентирующих **управление информационными потоками** в системе. Обычно управление информационными потоками применяется в рамках избирательной или полномочной политики, дополняя их и способствуя повышению надежности системы защиты.

Избирательное и полномочное управление доступом, а также управление информационными потоками являются тем фундаментом, на котором строится вся система защиты.

1.4.2. Построение системы защиты сети

Под *системой защиты сети* понимают единую совокупность правовых и морально-этических норм, административно-организационных мер, физических и программно-технических средств, направленных на противодействие угрозам сети с целью сведения к минимуму возможности ущерба.

Процесс построения системы защиты включает следующие этапы:

- анализ возможных угроз;
- планирование системы защиты;
- реализация системы защиты;
- сопровождение системы защиты.

Этап анализа возможных угроз необходим для фиксации состояния сети (конфигурации аппаратных и программных средств, технологии обработки информации) и определения учитываемых воздействий на компоненты системы. Практически невозможно обеспечить защиту от всех воздействий, поскольку невозможно полностью установить все угрозы и способы их реа-

лизаций. Поэтому из всего множества вероятных воздействий выбирают только такие воздействия, которые могут реально произойти и нанести серьезный ущерб.

На *этапе планирования* формулируется система защиты как единая совокупность мер противодействия угрозам различной природы.

По способам осуществления все меры обеспечения безопасности компьютерных систем подразделяют на:

- правовые (законодательные);
- морально-этические;
- административные;
- физические;
- аппаратно-программные.

Перечисленные меры безопасности можно рассматривать как последовательность барьеров или рубежей защиты информации. Для того чтобы добраться до защищаемой информации, нужно последовательно преодолеть несколько рубежей защиты. Рассмотрим их подробнее.

Первый рубеж защиты, встающий на пути человека, пытающегося осуществить НСД к информации, является чисто правовым. Этот аспект защиты информации связан с необходимостью соблюдения юридических норм при передаче и обработке информации. К *правовым мерам* защиты информации относятся действующие в стране законы, указы и другие нормативные акты, регламентирующие правила обращения с информацией ограниченного использования и ответственности за их нарушения. Этим они препятствуют несанкционированному использованию информации и являются сдерживающим фактором для потенциальных нарушителей.

Второй рубеж защиты образуют *морально-этические меры.* Этический момент в соблюдении требований защиты имеет весьма большое значение. Очень важно, чтобы люди, имеющие доступ к компьютерам, работали в здоровом морально-этическом климате.

К морально-этическим мерам противодействия относятся всевозможные нормы поведения, которые традиционно сложились или складываются в обществе по мере распространения компьютеров в стране. Эти нормы большей частью не являются обязательными, как законодательно утвержденные, но их несоблюдение обычно ведет к падению престижа человека, группы лиц или организации. Морально-этические нормы бывают как неписаными (например, общепризнанные нормы честности, патриотизма и т.д.), так и оформленными в некий свод правил или предписаний. Например, "Кодекс профессионального поведения членов Ассоциации пользователей ЭВМ США" рассматривает как неэтичные действия, которые умышленно или неумышленно:

- нарушают нормальную работу компьютерных систем;
- вызывают неоправданные затраты ресурсов (машинного времени, памяти, каналов связи и т.п.);
- нарушают целостность информации (хранимой и обрабатываемой);
- нарушают интересы других законных пользователей и т.п.

Третьим рубежом, препятствующим неправомочному использованию информации, являются административные меры. Администраторы всех рангов с учетом правовых норм и социальных аспектов определяют административные меры защиты информации.

Административные меры защиты относятся к мерам организационного характера. Они регламентируют:

- процессы функционирования сети;
- использование ресурсов сети;
- деятельность ее персонала;
- порядок взаимодействия пользователей с системой, с тем чтобы в наибольшей степени затруднить или исключить возможность реализации угроз безопасности.

Административные меры включают:

- разработку правил обработки информации в сети;
- совокупность действий при проектировании и оборудовании вычислительных центров и других объектов (учет влияния стихии, пожаров, охрана помещений и т.п.);
- совокупность действий при подборе и подготовке персонала (проверка новых сотрудников, ознакомление их с порядком работы с конфиденциальной информацией, с мерами ответственности за нарушение правил ее обработки; создание условий, при которых персоналу было бы невыгодно допускать злоупотребления и т.д.);
- организацию надежного пропускного режима;
- организацию учета, хранения, использования и уничтожения документов и носителей с конфиденциальной информацией;
- распределение реквизитов разграничения доступа (паролей, полномочий и т.п.);
- организацию скрытого контроля за работой пользователей и персонала сети;
- совокупность действий при проектировании, разработке, ремонте и модификации оборудования и программного обеспечения (сертификация используемых технических и программных средств, строгое санкционирование, рассмотрение и утверждение всех изменений, проверка на удовлетворение требованиям защиты, документальная фиксация изменений и т.п.).

Четвертым рубежом являются *физические меры защиты*. К физическим мерам защиты относятся разного рода механические, электро- и электронно-механические устройства или сооружения, специально предназначенные для создания физических препятствий на возможных путях проникновения и доступа потенциальных нарушителей к компонентам системы и защищаемой информации.

Пятым рубежом являются *аппаратно-программные средства защиты*. К ним относятся различные электронные устройства и специальные программы, которые реализуют самостоятельно или в комплексе с другими средствами следующие способы защиты:

- идентификацию (распознавание) и аутентификацию (проверка подлинности) субъектов (пользователей, процессов);
- разграничение доступа к ресурсам сети;
- контроль целостности данных;
- обеспечение конфиденциальности данных;
- регистрацию и анализ событий, происходящих в сети;
- резервирование ресурсов и компонентов сети.

Большинство из перечисленных способов защиты реализуется криптографическими методами защиты информации.

1.5.Базовые технологии безопасности сетей

В разных программных и аппаратных продуктах, предназначенных для защиты данных, часто используются одинаковые подходы, приемы и технические решения. К таким базовым технологиям безопасности относятся аутентификация, авторизация, аудит и технология защищенного канала.

1.5.1. Аутентификация

Аутентификация (authentication) предотвращает доступ к сети нежелательных лиц и разрешает вход для легальных пользователей. Термин «аутентификация» в переводе с латинского означает «установление подлинности». Аутентификацию следует отличать от идентификации. Идентификаторы пользователей используются в системе с теми же целями, что и идентификаторы любых других объектов, файлов, процессов, структур данных, но они не связаны непосредственно с обеспечением безопасности. Идентификация заключается в сообщении пользователем системе своего идентификатора, в то время как аутентификация – это процедура доказательства пользователем того, что он есть тот, за кого себя выдает, в частности, доказательство того, что именно ему принадлежит введенный им идентификатор.

В процедуре аутентификации участвуют две стороны: одна сторона доказывает свою аутентичность, предъявляя некоторые доказательства, а другая сторона – аутентификатор – проверяет эти доказательства и принимает решение. В качестве доказательства аутентичности используются самые разнообразные приемы:

- аутентифицируемый может продемонстрировать знание некоего общего для обеих сторон секрета: слова (пароля) или факта (даты и места события, прозвища человека и т. п.);
- аутентифицируемый может продемонстрировать, что он владеет неким уникальным предметом (физическим ключом), в качестве которого может выступать, например, электронная магнитная карта;
- аутентифицируемый может доказать свою идентичность, используя собственные биохарактеристики: рисунок радужной оболочки глаза или отпечатки пальцев, которые предварительно были занесены в базу данных аутентификатора.

Сетевые службы аутентификации строятся на основе всех этих приемов, по чаще всего для доказательства идентичности пользователя используются пароли. Простота и логическая ясность механизмов аутентификации

на основе паролей в какой-то степени компенсирует известные слабости паролей. Это, во-первых, возможность раскрытия и разгадывания паролей, а во-вторых, возможность «подслушивания» пароля путем анализа сетевого трафика. Для снижения уровня угрозы от раскрытия паролей администраторы сети, как правило, применяют встроенные программные средства для формирования политики назначения и использования паролей: задание максимального и минимального сроков действия пароля, хранение списка уже использованных паролей, управление поведением системы после нескольких неудачных попыток логического входа и т. п. Перехват паролей по сети можно предупредить путем их шифрования перед передачей в сеть.

1.5.2. Авторизация доступа

Средства *авторизации (authorization)* контролируют доступ легальных пользователей к ресурсам системы, предоставляя каждому из них именно те права, которые ему были определены администратором. Кроме предоставления прав доступа пользователям к каталогам, файлам и принтерам система авторизации может контролировать возможность выполнения пользователями различных системных функций, таких как локальный доступ к серверу, установка системного времени, создание резервных копий данных, выключение сервера и т. п.

Система авторизации наделяет пользователя сети правами выполнять определенные действия над определенными ресурсами.

Процедуры авторизации реализуются программными средствами, которые могут быть встроены в операционную систему или в приложение, а также могут поставляться в виде отдельных программных продуктов. При этом программные системы авторизации могут строиться на базе двух схем:

- централизованная схема авторизации, базирующаяся на сервере;
- децентрализованная схема, базирующаяся на рабочих станциях.

В первой схеме сервер управляет процессом предоставления ресурсов пользователю. Главная цель таких систем – реализовать «принцип единого входа». В соответствии с централизованной схемой пользователь один раз логически входит в сеть и получает на все время работы некоторый набор разрешений по доступу к ресурсам сети. Система Kerberos с ее сервером безопасности и архитектурой клиент-сервер является наиболее известной системой этого типа. Системы TACACS и RADIUS, часто применяемые совместно с системами удаленного доступа, также реализуют этот подход.

При втором подходе рабочая станция сама является защищенной – средства защиты работают на каждой машине, и сервер не требуется.

В крупных сетях часто применяется комбинированный подход предоставления пользователю прав доступа к ресурсам сети: сервер удаленного доступа ограничивает доступ пользователя к подсетям или серверам корпоративной сети, то есть к укрупненным элементам сети, а каждый отдельный сервер сети сам по себе ограничивает доступ пользователя к своим внутренним ресурсам: разделяемым каталогам, принтерам или приложениям.

1.5.3. Аудит

Аудит (auditing) – фиксация в системном журнале событий, связанных с доступом к защищаемым системным ресурсам. Подсистема аудита современных ОС позволяет дифференцированно задавать перечень интересующих администратора событий с помощью удобного графического интерфейса. Средства учета и наблюдения обеспечивают возможность обнаружить и зафиксировать важные события, связанные с безопасностью, или любые попытки создать, получить доступ или удалить системные ресурсы. Аудит используется для того, чтобы засекать даже неудачные попытки «взлома» системы.

Учет и наблюдение означает способность системы безопасности «шпионить» за выбранными объектами и их пользователями и выдавать сообщения тревоги, когда кто-нибудь пытается читать или модифицировать системный файл. Если кто-то пытается выполнить действия, определенные системой безопасности для отслеживания, то система аудита пишет сообщение в журнал регистрации, идентифицируя пользователя. Системный менеджер может создавать отчеты о безопасности, которые содержат информацию из журнала регистрации. Для «сверхбезопасных» систем предусматриваются аудио- и видеосигналы тревоги, устанавливаемые на машинах администраторов, отвечающих за безопасность.

Поскольку никакая система безопасности не гарантирует защиту на уровне 100%, то последним рубежом в борьбе с нарушениями оказывается система аудита. Действительно, после того как злоумышленнику удалось провести успешную атаку, пострадавшей стороне не остается ничего другого, как обратиться к службе аудита. Если при настройке службы аудита были правильно заданы события, которые требуется отслеживать, то подробный анализ записей в журнале может дать много полезной информации. Эта информация, возможно, позволит найти злоумышленника или, по крайней мере, предотвратить повторение подобных атак путем устранения уязвимых мест в системе защиты.

1.5.4. Технология защищенного канала

Технология защищенного канала призвана обеспечивать безопасность передачи данных по открытой транспортной сети, например по Интернету. Защищенный канал подразумевает выполнение трех основных функций:

- взаимную аутентификацию абонентов при установлении соединения, которая может быть выполнена, например, путем обмена паролями;
- защиту передаваемых по каналу сообщений от несанкционированного доступа, например, путем шифрования;
- подтверждение целостности поступающих по каналу сообщений, например, путем передачи одновременно с сообщением его дайджеста.

Совокупность защищенных каналов, созданных предприятием в публичной сети для объединения своих филиалов, часто называют *виртуальной частной сетью* (Virtual Private Network, VPN).

Существуют разные реализации технологии защищенного канала, которые, в частности, могут работать на разных уровнях модели OSI. Так, функции популярного протокола SSL соответствуют *представительному* уровню модели OSI. Новая версия *сетевого* протокола IP предусматривает все функции – взаимную аутентификацию, шифрование и обеспечение целостности, – которые по определению свойственны защищенному каналу, а протокол туннелирования PPTP защищает данные на *канальном* уровне.

В зависимости от места расположения программного обеспечения защищенного канала различают две схемы его образования:

- схему с конечными узлами, взаимодействующими через публичную сеть (Рис. 1.2, а);
- схему с оборудованием поставщика услуг публичной сети, расположенным на границе между частной и публичной сетями (Рис. 1.2, б).

В первом случае защищенный канал образуется программными средствами, установленными на двух удаленных компьютерах, принадлежащих двум разным локальным сетям одного предприятия и связанных между собой через публичную сеть. Преимуществом этого подхода является полная защищенность канала вдоль всего пути следования, а также возможность использования любых протоколов создания защищенных каналов, лишь бы на конечных точках канала поддерживался один и тот же протокол. Недостатки заключаются в избыточности и децентрализованности решения. Избыточность состоит в том, что вряд ли стоит создавать защищенный канал на всем пути прохождения данных: уязвимыми для злоумышленников обычно являются сети с коммутацией пакетов, а не каналы телефонной сети или выделенные каналы, через которые локальные сети подключены к территориальной сети. Поэтому защиту каналов доступа к публичной сети можно считать избыточной. Децентрализация заключается в том, что для каждого компьютера, которому требуется предоставить услуги защищенного канала, необходимо отдельно устанавливать, конфигурировать и администрировать программные средства защиты данных. Подключение каждого нового компьютера к защищенному каналу требует выполнения этих трудоемких работ заново.

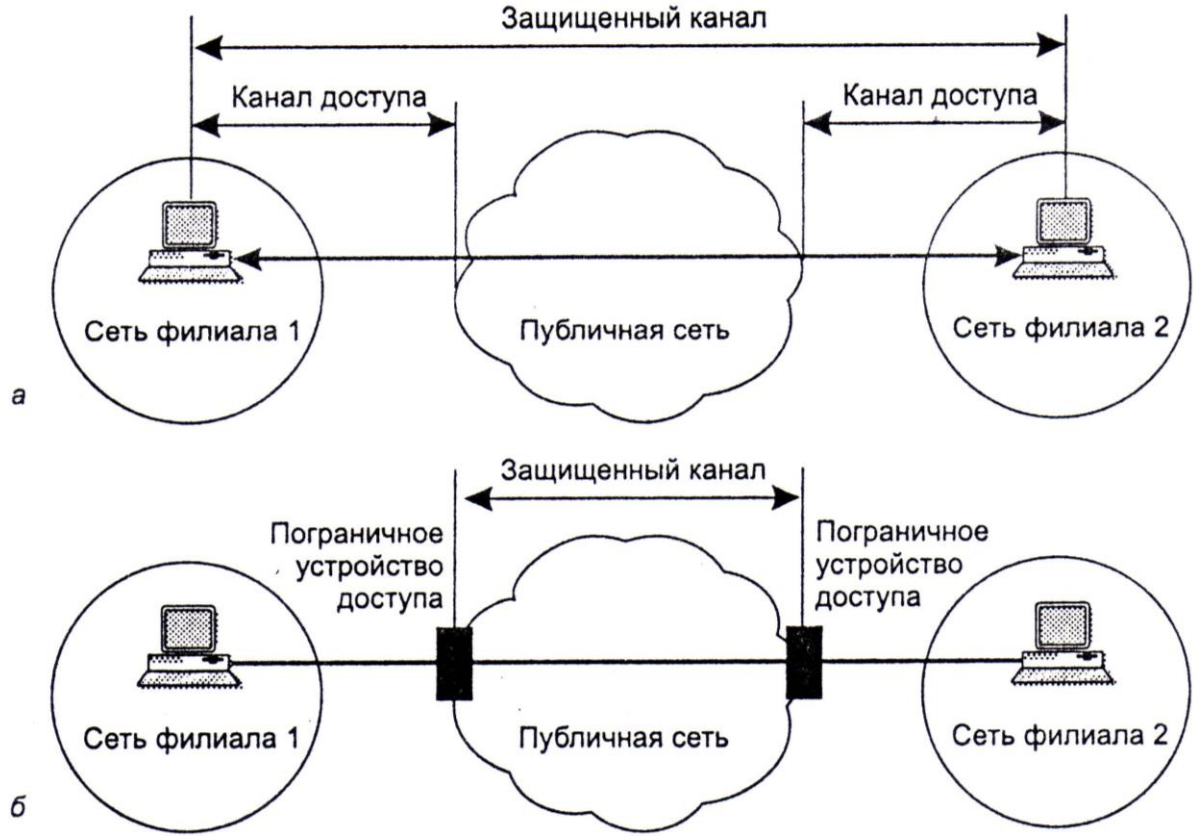


Рисунок 1.2 – Два способа образования защищенного канала

Во втором случае клиенты и серверы не участвуют в создании защищенного канала – он прокладывается только внутри публичной сети с коммутацией пакетов, например, внутри Интернета. Канал может быть проложен, например, между сервером удаленного доступа поставщика услуг публичной сети и пограничным маршрутизатором корпоративной сети. Это хорошо масштабируемое решение, управляемое централизованно как администратором корпоративной сети, так и администратором сети поставщика услуг. Для компьютеров корпоративной сети канал прозрачен – программное обеспечение этих конечных узлов остается без изменений. Такой гибкий подход позволяет легко образовывать новые каналы защищенного взаимодействия между компьютерами независимо от их места расположения. Реализация этого подхода сложнее – нужен стандартный протокол образования защищенного канала, требуется установка у всех поставщиков услуг программного обеспечения, поддерживающего такой протокол, необходима поддержка протокола производителями пограничного коммуникационного оборудования. Однако вариант, когда все заботы по поддержании защищенного канала берет на себя поставщик услуг публичной сети, оставляет сомнения в надежности защиты: во-первых, незащищенными оказываются каналы доступа к публичной сети, во-вторых, потребитель услуг чувствует себя в полной зависимости от надежности поставщика услуг. И, тем не менее, специалисты прогнозируют, что именно вторая схема в ближайшем будущем станет основной в построении защищенных каналов.

7. ПРИНЦИПЫ КРИПТОГРАФИЧЕСКОЙ ЗАЩИТЫ ИНФОРМАЦИИ

Криптография представляет собой совокупность методов преобразования данных, направленных на то, чтобы сделать эти данные недоступными для противника. Такие преобразования позволяют решить две главные проблемы защиты данных: *проблему конфиденциальности* (путем лишения противника возможности извлечь информацию из канала связи) и *проблему целостности* (путем лишения противника возможности изменить сообщение так, чтобы изменился его смысл, или ввести ложную информацию в канал связи).

Проблемы конфиденциальности и целостности информации тесно связаны между собой, поэтому методы решения одной из них часто применимы для решения другой.

2.1.Схема симметричной криптосистемы

Обобщенная схема криптографической системы, обеспечивающей шифрование передаваемой информации, показана на рис.2.1.

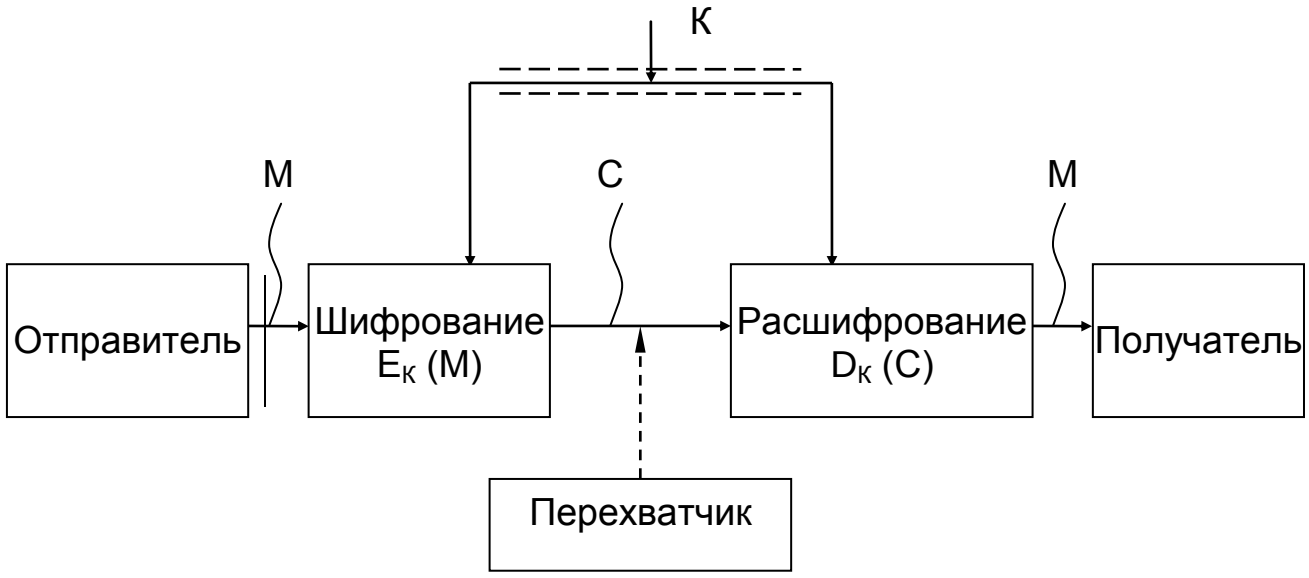


Рисунок 2.1 – Обобщенная схема криптосистемы

Отправитель генерирует *открытый текст* исходного сообщения М, которое должно быть передано законному *получателю* по незащищенному каналу. За каналом следит *перехватчик* с целью перехватить и раскрыть передаваемое сообщение. Для того чтобы перехватчик не смог узнать содержание сообщения М, отправитель шифрует его с помощью обратимого преобразования $E_К$ и получает *шифртекст* (или *криптограмму*) $С = E_К (М)$, который отправляет получателю.

Законный получатель, приняв шифртекст С, расшифровы-вает его с помощью обратного преобразования $D = E_К^{-1}$ и получает исходное сообщение в виде открытого текста М:

$$D_K(C) = E_K^{-1}(E_K(M)) = M.$$

Преобразование E_K выбирается из семейства криптографических преобразований, называемых криптоалгоритмами. Параметр, с помощью которого выбирается отдельное используемое преобразование, называется криптографическим ключом К. Криптосистема имеет разные варианты реализации: набор инструкций, аппаратные средства, комплекс программ компьютера, которые позволяют зашифровать открытый текст и расшифровать шифр-текст различными способами, один из которых выбирается с помощью конкретного ключа К.

Криптографическая система – это однопараметрическое семейство $(E_K)_{K\in\bar{K}}$ обратимых преобразований

$$E_K : \bar{M} \to \bar{C}$$

из пространства $\bar{M}$ сообщений открытого текста в пространство $\bar{C}$ шифрованных текстов. Параметр К (ключ) выбирается из конечного множества $\bar{K}$, называемого *пространством ключей.*

Преобразование шифрования может быть симметричным или асимметричным относительно преобразования расшифрования. Это важное свойство функции преобразования определяет два класса криптосистем:

- симметричные (одноключевые) криптосистемы;
- асимметричные (двухключевые) криптосистемы (с открытым ключом).

Схема симметричной криптосистемы с одним секретным ключом показана на рис.2.1. В ней используются одинаковые секретные ключи в блоке шифрования и блоке расшифрования.

2.2.Схема асимметричной криптосистемы

Обобщенная схема асимметричной криптосистемы с двумя разными ключами $К_1$ и $К_2$ показана на рис. 2.2. В этой криптосистеме один из ключей является открытым, а другой – секретным.

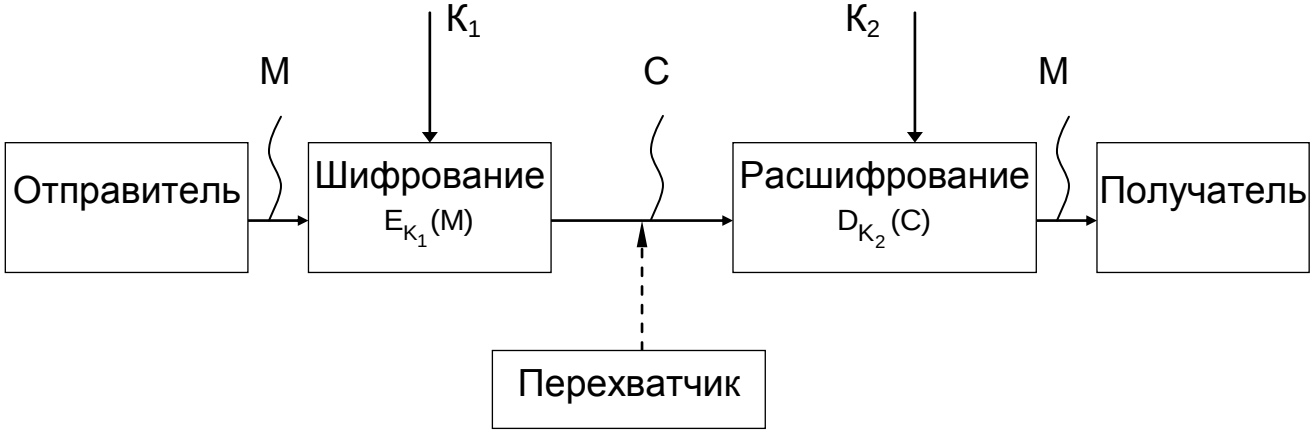


Рисунок 2.2 – Обобщенная схема асимметричной криптосистемы с открытым ключом

В симметричной криптосистеме секретный ключ надо передавать отправителю и получателю по защищенному каналу распространения ключей,

например такому, как курьерская служба. На рис. 2.1 этот канал показан "экранированной" линией. Существуют и другие способы распределения секретных ключей, они будут рассмотрены позднее. В асимметричной криптосистеме передают по незащищенному каналу только открытый ключ, а секретный ключ сохраняют на месте его генерации.

На рис. 2.3 показан поток информации в криптосистеме в случае активных действий перехватчика. Активный перехватчик не только считывает все шифртексты, передаваемые по каналу, но может также пытаться изменять их по своему усмотрению.

Любая попытка со стороны перехватчика расшифровать шифртекст C для получения открытого текста M или зашифровать свой собственный текст M’ для получения правдоподобного шифртекста C’, не имея подлинного ключа, называется *крипто-аналитической атакой.*

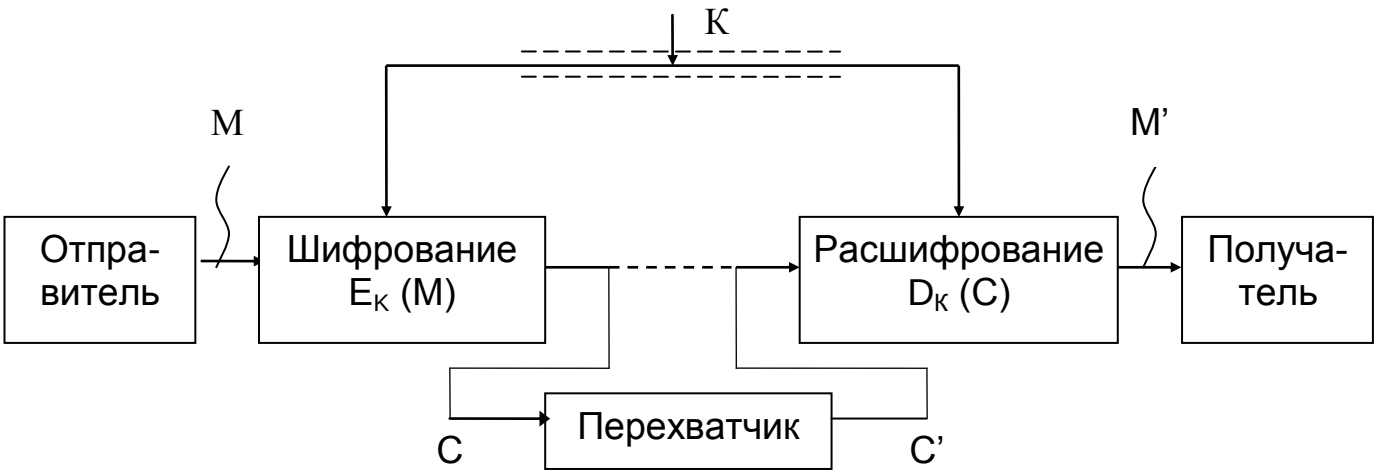


Рисунок 2.3 – Поток информации в криптосистеме при активном перехвате сообщений

Если предпринятые криптоаналитические атаки не достигают поставленной цели и криптоаналитик не может, не имея подлинного ключа, вывести M из C или C’ из M’, то считается, что такая криптосистема является *криптостойкой*.

Криптоанализ – это наука о раскрытии исходного текста зашифрованного сообщения без доступа к ключу. Успешный анализ может раскрыть исходный текст или ключ. Он позволяет также обнаружить слабые места в криптосистеме, что, в конечном счете, ведет к тем же результатам.

Фундаментальное правило криптоанализа, впервые сформулированное голландцем А.Керкхоффом еще в XIX веке заключается в том, что стойкость шифра (криптосистемы) должна определяться только секретностью ключа. Иными словами, правило Керкхоффа состоит в том, что весь алгоритм шифрования, кроме значения секретного ключа, известен криптоаналитику противника. Это обусловлено тем, что криптосистема, реализующая семейство криптографических преобразований, обычно рассматривается как открытая система.

2.3. Аппаратно-программные средства защиты компьютерной информации

Аппаратно-программные средства, обеспечивающие повышенный уровень защиты можно разбить на пять основных групп (Рис. 2.4).

Первую группу образуют *системы идентификации и аутентификации пользователей*. Такие системы применяются для ограничения доступа случайных и незаконных пользователей к ресурсам компьютерной системы. Общий алгоритм работы этих систем заключается в том, чтобы получить от пользователя информацию, удостоверяющую его личность, проверить ее подлинность и затем предоставить (или не предоставить) этому пользователю возможность работы с системой.

При построении подобных систем возникает проблема выбора информации, на основе которой осуществляются процедуры идентификации и аутентификации пользователя. Можно выделить следующие типы:

(1) секретная информация, которой обладает пользователь (пароль, персональный идентификатор, секретный ключ и т.п.); эту информацию пользователь должен запомнить или же могут быть применены специальные средства хранения этой информации);

(2) физиологические параметры человека (отпечатки пальцев, рисунок радужной оболочки глаза и т.п.) или особенности поведения человека (особенности работы на клавиатуре и т.п.).

Системы идентификации, основанные на первом типе информации, принято считать *традиционными*. Системы идентификации, использующие второй тип информации, называются *биометрическими*.

Вторую группу средств, обеспечивающих повышенный уровень защиты, составляют *системы шифрования дисковых данных*. Основная задача, решаемая такими системами, состоит в защите от несанкционированного использования данных, расположенных на магнитных носителях.

Обеспечение конфиденциальности данных, располагаемых на магнитных носителях, осуществляется путем их шифрования с использованием симметричных алгоритмов шифрования. Основным классификационным признаком для комплексов шифрования служит уровень их встраивания в компьютерную систему.

Работа прикладных программ с дисковыми накопителями состоит из двух этапов – “логического” и “физического”.

Логический этап соответствует уровню взаимодействия прикладной программы с операционной системой (например, вызов сервисных функций чтения/записи данных). На этом уровне основным объектом является файл.

Физический этап соответствует уровню взаимодействия операционной системы и аппаратуры. В качестве объектов этого уровня выступают структуры физической организации данных - сектора диска.

В результате, системы шифрования данных могут осуществлять криптографические преобразования данных на уровне файлов (защищаются отдельные файлы) и на уровне дисков (защищаются диски целиком).

Другим классификационным признаком систем шифрования дисковых данных является способ их функционирования.

По способу функционирования системы шифрования дисковых данных делят на два класса:

(1) системы “прозрачного” шифрования;

(2) системы, специально вызываемые для осуществления шифрования.

Рисунок 2.4 – Аппаратно-программные средства защиты компьютерной информации

В системах *прозрачного шифрования* (шифрования “на лету”) криптографические преобразования осуществляются в режиме реального времени, незаметно для пользователя. Например, пользователь записывает подготовленный в текстовом редакторе документ на защищаемый диск, а система защиты в процессе записи выполняет его шифрование.

Системы второго класса обычно представляют собой утилиты, которые необходимо специально вызывать для выполнения шифрования. К ним относятся, например, архиваторы со встроенными средствами парольной защиты.

К третьей группе средств относятся *системы шифрования данных, передаваемых по компьютерным сетям*. Различают два основных способа шифрования: канальное шифрование и оконечное (абонентское) шифрование.

В случае *канального шифрования* защищается вся передаваемая по каналу связи информация, включая служебную. Соответствующие процедуры шифрования реализуются с помощью протокола канального уровня семиуровневой эталонной модели взаимодействия открытых систем OSI.

Этот способ шифрования обладает следующим достоинством - встраивание процедур шифрования на канальный уровень позволяет использовать аппаратные средства, что способствует повышению производительности системы.

Однако, у данного подхода имеются существенные недостатки:

- шифрованию на данном уровне подлежит вся информация, включая служебные данные транспортных протоколов; это осложняет механизм маршрутизации сетевых пакетов и требует расшифрования данных в устройствах промежуточной коммутации (шлюзах, ретрансляторах и т.п.);
- шифрование служебной информации, неизбежное на данном уровне, может привести к появлению статистических закономерностей в шифрованных данных; это влияет на надежность защиты и накладывает ограничения на использование криптографических алгоритмов.

Оконечное (абонентское) шифрование позволяет обеспечить конфиденциальность данных, передаваемых между двумя прикладными объектами (абонентами). Оконечное шифрование реализуется с помощью протокола прикладного или представительного уровня эталонной модели OSI. В этом случае защищенным оказывается только содержание сообщения, вся служебная информация остается открытой. Данный способ позволяет избежать проблем, связанных с шифрованием служебной информации, но при этом возникают другие проблемы. В частности, злоумышленник, имеющий доступ к каналам связи компьютерной сети, получает возможность анализировать информацию о структуре обмена сообщениями, например, об отправителе и получателе, о времени и условиях передачи данных, а также об объеме передаваемых данных.

Четвертую группу средств защиты составляют *системы аутентификации электронных данных*.

При обмене электронными данными по сетям связи возникает проблема аутентификации автора документа и самого документа, т.е. установление подлинности автора и проверка отсутствия изменений в полученном документе.

Для аутентификации электронных данных применяют код аутентификации сообщения (имитовставку) или электронную цифровую подпись. При формировании кода аутентификации сообщения и электронной цифровой подписи используются разные типы систем шифрования.

Код аутентификации сообщения MAC (Message Authentication Code) формируют с помощью симметричных систем шифрования данных. Проверка целостности принятого сообщения осуществляется путем проверки кода MAC получателем сообщения.

В отечественном стандарте симметричного шифрования данных (ГОСТ 28147-89) предусмотрен режим выработки имитовставки, обеспечивающий *имитозащиту*, т.е. защиту системы шифрованной связи от навязывания ложных данных.

Имитовставка вырабатывается из открытых данных посредством специального преобразования шифрования с использованием секретного ключа

и передается по каналу связи в конце зашифрованных данных. Имитовставка проверяется получателем сообщения, владеющим секретным ключом, путем повторения процедуры, выполненной ранее отправителем, над полученными открытыми данными.

Электронная цифровая подпись (ЭЦП) представляет собой относительно небольшое количество дополнительной аутентифицирующей цифровой информации, передаваемой вместе с подписываемым текстом.

Для реализации ЭЦП используются принципы асимметричного шифрования. Система ЭЦП включает процедуру формирования цифровой подписи отправителем с использованием секретного ключа отправителя и процедуру проверки подписи получателем с использованием открытого ключа отправителя.

Пятую группу средств, обеспечивающих повышенный уровень защиты, *образуют средства управления ключевой информацией*. Под ключевой информацией понимается совокупность всех используемых в компьютерной системе или сети криптографических ключей.

Безопасность любого криптографического алгоритма определяется используемыми криптографическими ключами. В случае ненадежного управления ключами злоумышленник может завладеть ключевой информацией и получить полный доступ ко всей информации в компьютерной системе или сети.

Основным классификационным признаком средств управления ключевой информацией является вид функции управления ключами. Различают следующие основные виды функций управления ключами: генерация ключей, хранение ключей и распределение ключей.

Способы генерации ключей различаются для симметричных и асимметричных криптосистем. Для генерации ключей симметричных криптосистем используются аппаратные и программные средства генерации случайных чисел, в частности, схемы с применением блочного симметричного алгоритма шифрования. Генерация ключей для асимметричных криптосистем представляет существенно более сложную задачу в связи с необходимостью получения ключей с определенными математическими свойствами.

Функция хранения ключей предполагает организацию безопасного хранения, учета и удаления ключей. Для обеспечения безопасного хранения и передачи ключей применяют их шифрование с помощью других ключей. Такой подход приводит к *концепции иерархии ключей*. В иерархию ключей обычно входят главный ключ (мастер-ключ), ключ шифрования ключей и ключ шифрования данных. Следует отметить, что генерация и хранение мастер-ключей являются критическими вопросами криптографической защиты.

Распределение ключей является самым ответственным процессом в управлении ключами. Этот процесс должен гарантировать скрытность распределяемых ключей, а также оперативность и точность их распределения. Различают два основных способа распределения ключей между пользователями компьютерной сети:

1) применение одного или нескольких центров распределения ключей;
2) прямой обмен сеансовыми ключами между пользователями.

3. СОВРЕМЕННЫЕ СИММЕТРИЧНЫЕ КРИПТОСИСТЕМЫ

Рассеивание представляет собой распространение влияния одного знака открытого текста на много знаков шифртекста, что позволяет скрыть статистические свойства открытого текста.

Перемешивание предполагает использование таких шифрующих преобразований, которые усложняют восстановление взаимосвязи статистических свойств открытого и шифрованного текстов. Однако шифр должен не только затруднять раскрытие, но и обеспечивать легкость зашифрования и расшифрования при известном пользователю секретном ключе.

Распространенным способом достижения эффектов рассеивания и перемешивания является использование *составного шифра*, т.е. такого шифра, который может быть реализован в виде некоторой последовательности простых шифров, каждый из которых вносит свой вклад в значительное суммарное рассеивание и перемешивание.

В составных шифрах в качестве простых шифров чаще всего используются простые перестановки и подстановки. При *перестановке* просто перемешивают символы открытого текста, причем конкретный вид перемешивания определяется секретным ключом. При **подстановке** каждый символ открытого текста заменяют другим символом из того же алфавита, а конкретный вид подстановки также определяется секретным ключом. Следует заметить, что в современном блочном шифре блоки открытого текста и шифртекста представляют собой двоичные последовательности обычно длиной 64 бита. В принципе каждый блок может принимать 2^{64} значений. Поэтому подстановки выполняются в очень большом алфавите, содержащем до $2^{64} \approx 10^{19}$ "символов".

При многократном чередовании простых перестановок и подстановок, управляемых достаточно длинным секретным ключом, можно получить очень стойкий шифр с хорошим рассеиванием и перемешиванием. Рассмотренные ниже криптоалгоритмы DES, IDEA и отечественный стандарт шифрования данных построены в полном соответствии с указанной методологией.

3.1 Классическая сеть Фейстеля

Сетью Фейстеля называется методика обратимых преобразований текста, при которой значение, вычисленное от одной из частей текста, накладывается на другие части. Сеть Фейстеля представляет собой модификацию метода смешивания текущей части шифруемого блока с результатом некоторой функции, вычисленное от другой независимой части того же блока. Эта методика обеспечивает выполнение важного требования о многократном использовании ключа и материала исходного блока информации. Часто структуру сети выполняют таким образом, чтобы использовать для шифрования и расшифрования один и тот же алгоритм.

Структура классической сети Фейстеля показана на рис. 3.1

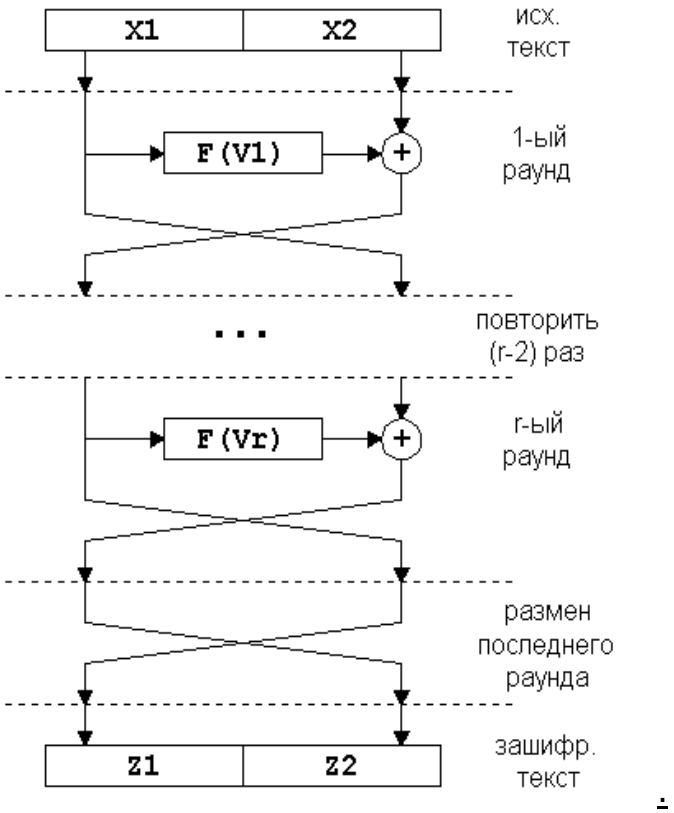


Рисунок 3.1 – Структура классической сети Фейстеля

Независимые потоки информации, порожденные из исходного блока, называются **ветвями сети**. В классической схеме Фейстеля их две. Величины V_i называются **параметрами сети**, обычно это функции от материала ключа. Функция F называется **образующей**. Действие, состоящее из однократного вычисления образующей функциии и последующего наложения ее результата на другую ветвь с обменом местами называется **циклом** или **раундом** сети Фейстеля. Оптимальное число раундов K − от 8 до 32.

Увеличение количества раундов значительно повышает криптостойкость любого симметричного блочного шифра. Благодаря этой особенности сети Фейстеля получили широкое распространение – при обнаружении слабого места в алгоритме почти всегда достаточно не переписывая сам алгоритм увеличить количество раундов на 4-8.

В соответствии с описанной методикой построены блочные симметричные криптоалгоритмы DES, IDEA, ГОСТ 28147-89 и ряд других алгоритмов.

3.2. Американский стандарт шифрования данных DES

Стандарт шифрования данных DES (Data Encryption Standard) опубликован в 1977 г. Национальным бюро стандартов США. Стандарт DES предназначен для защиты от несанкционированного доступа к важной, но несекретной информации в государственных и коммерческих организациях США. Алгоритм, положенный в основу стандарта, распространялся достаточно быстро, и уже в 1980 г. был одобрен Национальным институтом стандартов и технологий США (НИСТ). С этого момента DES превращается в стандарт не только по названию (Data Encryption Standard), но и фактически. Появляются программное обеспечение и специализированные микроЭВМ, предна-

значенные для шифрования и расшифрования информации в сетях передачи данных.

К настоящему времени DES является наиболее распространенным алгоритмом, используемым в системах защиты коммерческой информации. Более того, реализация алгоритма DES в таких системах становится признаком хорошего тона.

Основные достоинства алгоритма DES:

- используется только один ключ длиной 56 бит;
- зашифровав сообщение с помощью одного пакета программ, для расшифровки можно использовать любой другой пакет программ, соответствующий стандарту DES;
- относительная простота алгоритма обеспечивает высокую скорость обработки;
- достаточно высокая стойкость алгоритма.

Алгоритм DES использует комбинацию **подстановок и перестановок**. DES осуществляет шифрование 64-битовых блоков данных с помощью 64-битового ключа, в котором значащими являются 56 бит (остальные 8 бит – проверочные биты для контроля на четность). Дешифрование в DES является операцией, обратной шифрованию, и выполняется путем повторения операций шифрования в обратной последовательности. Обобщенная схема процесса шифрования в алгоритме DES показана на 3.2. Процесс шифрования заключается в начальной перестановке битов 64-битового блока, шестнадцати циклах шифрования и, наконец, в конечной перестановке битов.

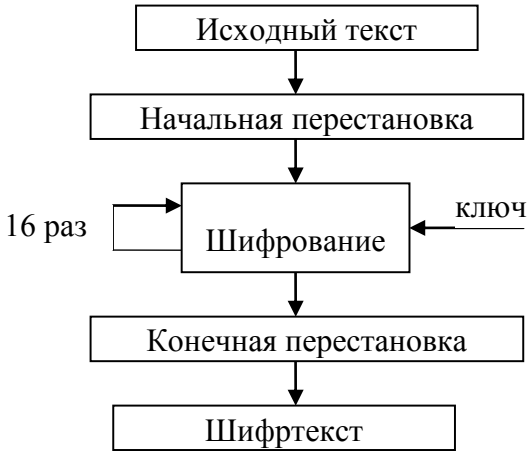


Рисунок 3.2 – Обобщенная схема шифрования в алгоритме DES

Все приводимые таблицы являются стандартными и должны включаться в реализацию алгоритма DES в неизменном виде.

Все перестановки и коды в таблицах подобраны разработчиками таким образом, чтобы максимально затруднить процесс расшифровки путем подбора ключа. При описании алгоритма DES (рис. 3.3) применены следующие обозначения:

L и R – последовательности битов (левая (left) и правая (right));

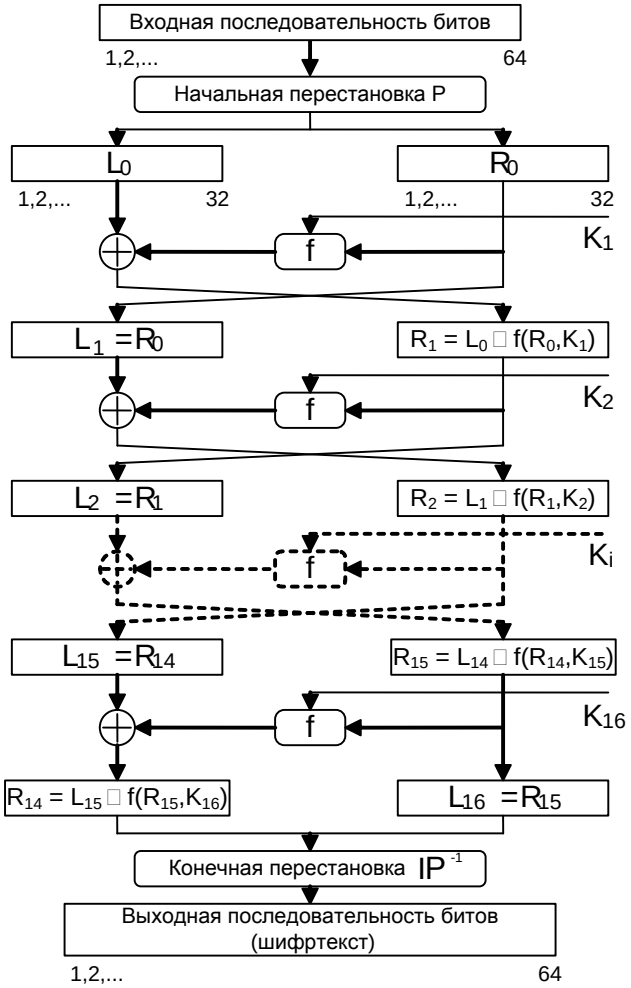


Рисунок 3.3 – Структура алгоритма DES

LR – конкатенация последовательностей L и R, т.е. такая последовательность битов, длина которой равна сумме длин L и R; в последовательности LR биты последовательности R следуют за битами последовательности L;

□ – операция побитового сложения по модулю 2.

Пусть из файла исходного текста считан очередной 64-битовый (8-байтовый) блок T. Этот блок T преобразуется с помощью *матрицы начальной перестановки* IP (табл. 3.1).

Таблица 3.1

Матрица начальной перестановки IP

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Биты входного блока T (64 бита) переставляются в соответствии с матрицей IP: бит 58 входного блока T становится битом 1, бит 50 – битом 2 и т.д. Эту перестановку можно описать выражением $T_0 = IP(T)$. Полученная после-

довательность битов T_0 разделяется на две последовательности: L_0 – левые или старшие биты, R_0 – правые или младшие биты, каждая из которых содер-жит 32 бита.

Затем выполняется итеративный процесс шифрования, состоящий из 16 шагов (циклов). Пусть T_i – результат i-й итерации:

$$T_i = L_i R_i,$$

где $L_i = t_1 t_2 ... t_{32}$ (первые 32 бита); $R_i = t_{33} t_{34} ... t_{64}$ (последние 32 бита). Тогда результат i-й итерации описывается следующими формулами:

$$L_i = R_{i-1}, \quad i = 1, 2, ..., 16;$$

$$R_i = L_{i-1} \square f (R_{i-1}, K_i), \quad i = 1, 2, ..., 16.$$

Функция f называется функцией шифрования. Ее аргументами являются последовательность R_{i-1}, получаемая на предыдущем шаге итерации, и 48-битовый ключ K_i, который является результатом преобразования 64-битового ключа шифра К. (Подробнее функция шифрования f и алгоритм получения ключа K_i описаны ниже.)

На последнем шаге итерации получают последовательности R_{16} и L_{16} (без перестановки местами), которые конкатенируются в 64-битовую последовательность $R_{16} L_{16}$.

По окончании шифрования осуществляется восстановление позиций битов с помощью матрицы обратной перестановки IP^{-1} (табл.3.2).

Таблица 3.2

Матрица обратной перестановки IP^{-1}

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Пример того, как соотносятся элементы первой строки матрицы IP^{-1} с элементами матрицы IP приведен в табл. 3.3.

Таблица 3.3

Связь элементов матриц

Элемент матрицы IP^{-1}	Элемент матрицы IP
40	01
8	02
48	03
16	04
56	05
...	...

Процесс расшифрования данных является инверсным по отношению к процессу шифрования. Все действия должны быть выполнены в обратном порядке. Это означает, что расшифровываемые данные сначала переставляются в соответствии с матрицей IP^{-1}, а затем над последовательностью битов $R_{16}L_{16}$ выполняются те же действия, что и в процессе шифрования, но в обратном порядке.

Итеративный процесс расшифрования может быть описан следующими формулами:

$$R_{i-1} = L_i, \quad i = 1, 2, ..., 16;$$

$L_{i-1} = R_i \square f(L_i, K_i), \quad i = 1, 2, ..., 16.$ Таким образом, для процесса расшифрования с переставленным входным блоком $R_{16}L_{16}$ на первой итерации используется ключ K_{16}, на второй итерации – K_{15} и т.д. На 16-й итерации используется ключ K_1. На последнем шаге итерации будут получены последовательности L_0 и R_0, которые конкатенируются в 64-битовую последовательность L_0R_0. Затем в этой последовательности 64 бита переставляются в соответствии с матрицей IP. Результат такого преобразования – исходная последовательность битов (расшифрованное 64-битовое значение).

Теперь рассмотрим, что скрывается под преобразованием, обозначенным буквой f. Схема вычисления функции шифрования $f(R_{i-1},K_i)$ показана на рис. 3.4.

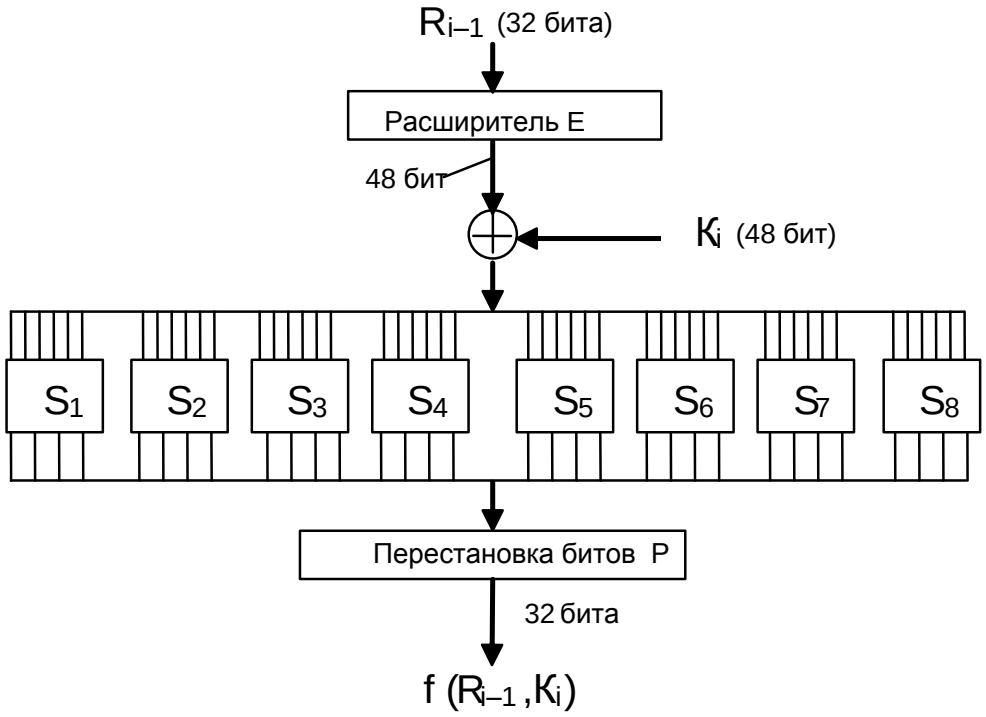


Рисунок 3.4 – Схема вычисления функции шифрования f

Для вычисления значения функции f используются:

- функция E (расширение 32 бит до 48);
- функция S_1, S_2, ..., S_8 (преобразование 6-битового числа в 4-битовое);
- функция P (перестановка битов в 32-битовой последовательности).

Приведем определения этих функций.

Аргументами функции шифрования f являются R_{i-1} (32 бита) и K_i (48 бит). Результат функции E (R_{i-1}) есть 48-битовое число. Функция расширения E, выполняющая расширение 32 бит до 48 (принимает блок из 32 бит и порождает блок из 48 бит), определяется табл. 3.4.

В соответствии с табл. 3.4 первые три бита E (R_{i-1}) – это биты 32, 1 и 2, а последние – 31, 32, 1. Полученный результат (обозначим его E(R_{i-1})) складывается по модулю 2 (операция XOR) с текущим значением

Таблица 3.4

Функция расширения E

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

ключа K_i и затем разбивается на восемь 6-битовых блоков B_1, B_2, ..., B_8:

$$E(R_{i-1}) \oplus K_i = B_1 B_2 \ldots B_8.$$

Далее каждый из этих блоков используется как номер элемента в функциях-матрицах S_1, S_2, ..., S_8, содержащих 4-битовые значения (табл. 3.5).

Таблица 3.5

Функции преобразования S_1, S_2, ..., S_8

Номер строки	Номер столбца 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7	
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8	S_1
2	4	1	4	8	13	6	2	11	15	12	9	7	3	10	5	0	
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13	
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10	
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5	S_2
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15	
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9	
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8	
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1	S_3
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7	
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12	
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15	
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9	S_4
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4	
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14	
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9	
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6	S_5
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14	
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3	
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11	
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8	S_6
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	1	6	
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13	
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1	
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6	S_7
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2	
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12	
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7	
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2	S_8
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8	
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11	

Следует отметить, что выбор элемента в матрице S_j осуществляется достаточно оригинальным образом. Пусть на вход матрицы S_j поступает 6-битовый блок $B_j = b_1\, b_2\, b_3\, b_4\, b_5\, b_6$, тогда двухбитовое число $b_1\, b_6$ указывает номер строки матрицы, а четырехбитовое число $b_2\, b_3\, b_4\, b_5$ – номер столбца. Например, если на вход матрицы S_1 поступает 6-битовый блок $B_1 = b_1\, b_2\, b_3\, b_4\, b_5\, b_6 = 100110$, то 2-битовое число $b_1\, b_6 = 10_{(2)} = 2_{(10)}$ указывает строку с номером 2 матрицы S_1, а 4-битовое число $b_2\, b_3\, b_4\, b_5 = 0011_{(2)} = 3_{(10)}$ указывает столбец с номером 3 матрицы S_1. Это означает, что в матрице S_1 блок $B_1 = 100110$ выбирает элемент на пересечении строки с номером 2 и столбца с номером 3, т.е. элемент $8_{(10)} =$

$=1000_{(2)}$. Совокупность 6-битовых блоков B_1, B_2,..., B_8 обеспечивает выбор четырехбитового элемента в каждой из матриц S_1, S_2, ..., S_8.

В результате получаем $S_1(B_1)$ $S_2(B_2)$ $S_3(B_3)$ ... $S_8(B_8)$, т.е. 32-битовый блок (поскольку матрицы S_j содержат 4-битовые элементы). Этот 32-битовый блок преобразуется с помощью функции перестановки битов P (табл.3.6).

Таким образом, функция шифрования

$$f(R_{i-1}, K_i) = P(S_1(B_1), ..., S_8(B_8)).$$

Как нетрудно заметить, на каждой итерации используется новое значение ключа K_i (длиной 48 бит). Новое значение ключа K_i вычисляется из начального ключа K (рис.3.5). Ключ K представляет собой 64-битовый блок с 8 битами контроля по четности, расположенными в позициях 8, 16, 24, 32, 40, 48, 56, 64. Для удаления контрольных битов и подготовки ключа к работе используется функция G первоначальной подготовки ключа (табл. 3.7).

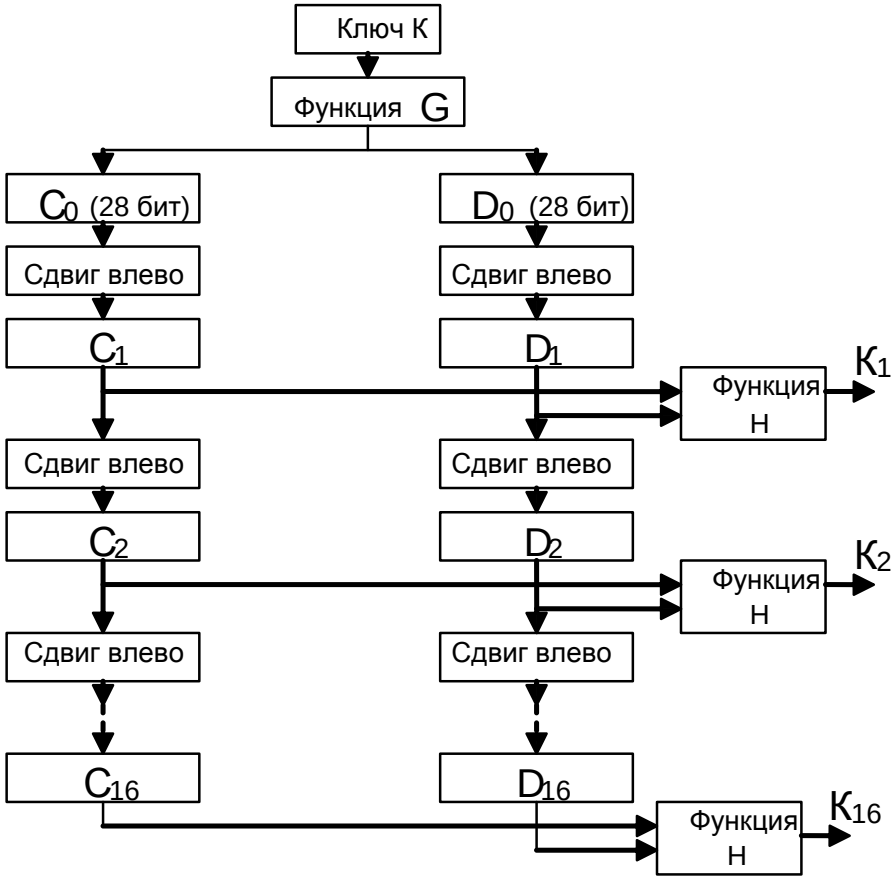


Рисунок 3.5 – Схема алгоритма вычисления ключей K_i Табл. 3.7 разделена на две части. Результат преобразования G(K) разбивается на две половины C_0 и D_0 по 28 бит каждая. Первые четыре строки матрицы G определяют, как выбираются биты последовательности C_0 (первым битом C_0 будет бит 57 ключа шифра, затем бит 49 и т.д., а последними битами – биты 44 и 36 ключа).

Таблица 3.6

Функция P перестановки битов

Таблица 3.7

Функция G первоначальной подготовки ключа (переставленная выборка 1)

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

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Следующие четыре строки матрицы G определяют, как выбираются биты последовательности D_0 (т.е. последовательность D_0 будет состоять из битов 63, 55, 47, ...,12, 4 ключа шифра).

Как видно из табл. 3.7, для генерации последовательностей C_0 и D_0 не используются биты 8, 16, 24, 32, 40, 48, 56 и 64 ключа шифра. Эти биты не влияют на шифрование и могут служить для других целей (например, для контроля по четности). Таким образом, в действительности ключ шифра является 56-битовым.

После определения C_0 и D_0 рекурсивно определяются C_i и D_i, I = 1, 2, ..., 16. Для этого применяются операции циклического сдвига влево на один или два бита в зависимости от номера шага итерации, как показано в табл. 3.8.

Операции сдвига выполняются для последовательностей C_i и D_i независимо. Например, последовательность C_3 получается посредством циклического сдвига влево на две позиции последовательности C_2, а последовательность D_3 – посредством сдвига влево на две позиции последовательности D_2, C_{16} и D_{16} получаются из C_{15} и D_{15} посредством сдвига влево на одну позицию.

Таблица 3.8

Таблица сдвигов s_i для вычисления ключа

Номер итерации	Количество s_i сдвигов влево, бит	Номер итерации	Количество s_i сдвигов влево, бит
1	1	9	1
2	1	10	2
3	2	11	2
4	2	12	2
5	2	13	2
6	2	14	2
7	2	15	2
8	2	16	1

Ключ K_i, определяемый на каждом шаге итерации, есть результат выбора конкретных битов из 56-битовой последовательности C_i D_i и их перестановки. Другими словами, ключ $K_i=H(C_i\ D_i)$, где функция H определяется матрицей, завершающей обработку ключа (табл. 3.9).

Таблица 3.9

Функция H завершающей обработки ключа (переставленная выборка 2)

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Как следует из табл.3.9, первым битом ключа K_i будет 14-й бит последовательности $C_i D_i$, вторым – 17-й бит, 47-м битом ключа K_i будет 29-й бит $C_i D_i$, а 48-м битом – 32-й бит $C_i D_i$.

3.2.1. Основные режимы работы алгоритма DES

Алгоритм DES вполне подходит как для шифрования, так и для аутентификации данных. Он позволяет непосредственно преобразовывать 64-битовый входной открытый текст в 64-битовый выходной шифрованный текст, однако данные редко ограничиваются 64 разрядами.

Чтобы воспользоваться алгоритмом DES для решения разнообразных криптографических задач, разработаны четыре рабочих режима:

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

Режим "Электронная кодовая книга"

Длинный файл разбивают на 64-битовые отрезки (блоки) по 8 байтов. Каждый из этих блоков шифруют независимо с использованием одного и того же ключа шифрования (рис.3.6).

Основное достоинство – простота реализации. Недостаток – относительно слабая устойчивость против квалифицированных криптоаналитиков. Из-за фиксированного характера шифрования при ограниченной длине блока 64 бита возможно проведение криптоанализа "со словарем". Блок такого размера может повториться в сообщении вследствие большой избыточности в тексте на естественном языке.

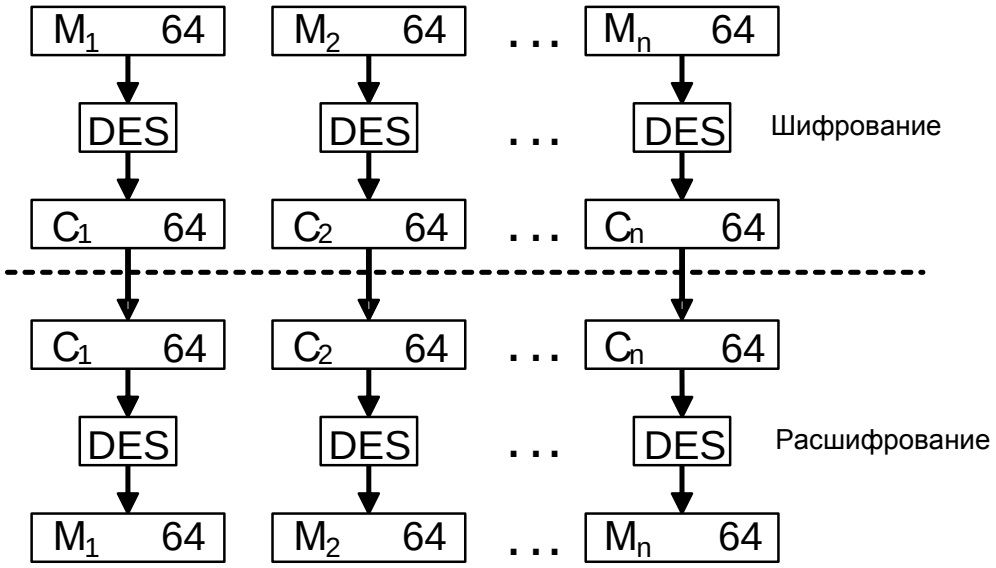


Рисунок 3.6 – Схема алгоритма DES в режиме электронной кодовой книги

Это приводит к тому, что идентичные блоки открытого текста в сообщении будут представлены идентичными блоками шифртекста, что дает криптоаналитику некоторую информацию о содержании сообщения.

Режим "Сцепление блоков шифра"

В этом режиме исходный файл M разбивается на 64-битовые блоки: M = $M_1M_2...M_n$. Первый блок M_1 складывается по модулю 2 с 64-битовым начальным вектором IV, который меняется ежедневно и держится в секрете (рис.3.7). Полученная сумма затем шифруется с использованием ключа DES, известного и отправителю, и получателю информации. Полученный 64-битовый шифр C_1 складывается по модулю 2 со вторым блоком текста, результат шифруется и получается второй 64-битовый шифр C_2, и т.д. Процедура повторяется до тех пор, пока не будут обработаны все блоки текста.

Таким образом, для всех i = 1…n (n – число блоков) результат шифрования C_i определяется следующим образом: C_i = =DES (M_i □ C_{i-1}), где C_0 = IV – начальное значение шифра, равное начальному вектору (вектору инициализации).

Очевидно, что последний 64-битовый блок шифртекста является функцией секретного ключа, начального вектора и каждого бита

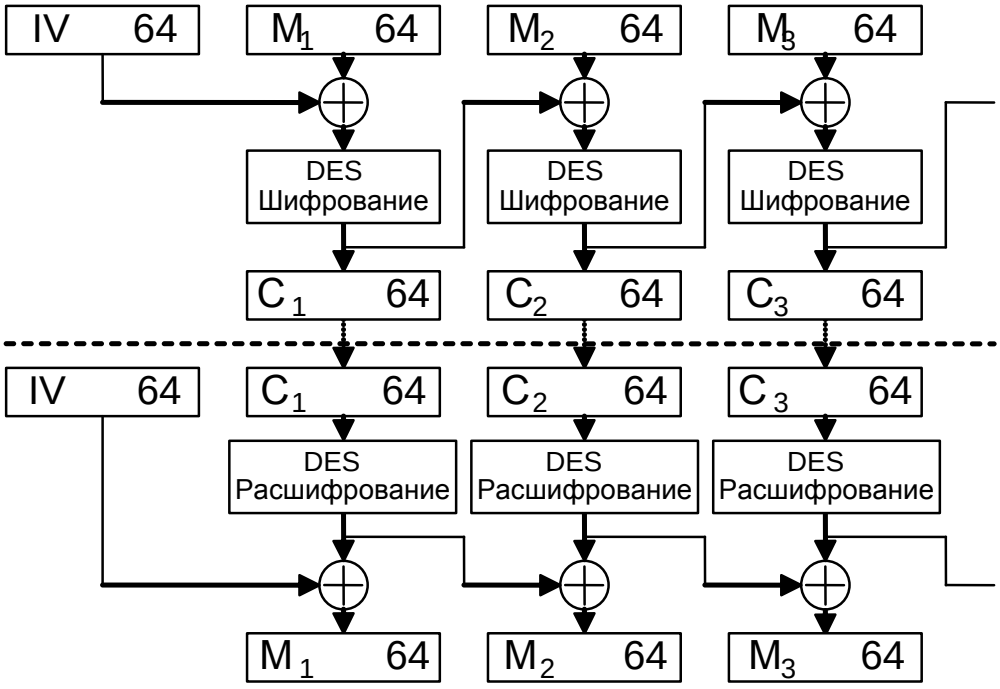


Рисунок 3.7 – Схема алгоритма DES в режиме сцепления блоков шифра

открытого текста независимо от его длины. Этот блок шифртекста называют кодом аутентификации сообщения (КАС).

Код КАС может быть легко проверен получателем, владеющим секретным ключом и начальным вектором, путем повторения процедуры, выполненной отправителем. Посторонний, однако, не может осуществить генерацию КАС, который воспринялся бы получателем как подлинный, чтобы добавить его к ложному сообщению, либо отделить КАС от истинного сообщения для использования его с измененным или ложным сообщением.

Достоинство данного режима в том, что он не позволяет накапливаться ошибкам при передаче.

Блок M_i является функцией только C_{i-1} и C_i. Поэтому ошибка при передаче приведет к потере только двух блоков исходно-го текста.

Режим "Обратная связь по шифру"

В этом режиме размер блока может отличаться от 64 бит (рис.3.8). Файл, подлежащий шифрованию (расшифрованию), считывается последовательными блоками длиной k битов (k =1…64).

Входной блок (64-битовый регистр сдвига) вначале содержит вектор инициализации, выровненный по правому краю.

Предположим, что в результате разбиения на блоки мы получили n блоков длиной k битов каждый (остаток дописывается нулями или пробелами). Тогда для любого i =1…n блок шифр-текста

$$C_i = M_i \square P_{i-1},$$

где P_{i-1} обозначает k старших битов предыдущего зашифрованного блока.

Обновление сдвигового регистра осуществляется путем удаления его старших k битов и записи C_i в регистр. Восстановление зашифрованных данных также выполняется относительно просто: P_{i-1} и C_i вычисляются аналогичным образом и

$$M_i = C_i \oplus P_{i-1}.$$

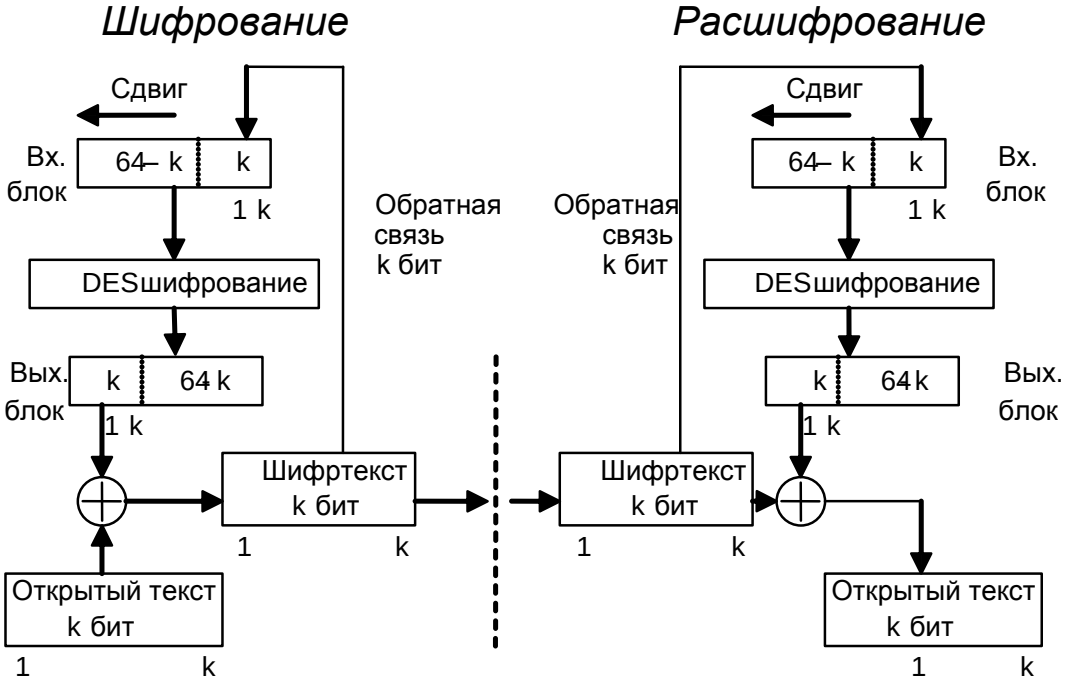


Рисунок 3.8 – Схема алгоритма DES в режиме обратной связи по шифртексту

Режим "Обратная связь по выходу"

Этот режим тоже использует переменный размер блока и сдвиговый регистр, инициализируемый так же, как в режиме CFB, а именно – входной блок вначале содержит вектор инициализации IV, выровненный по правому краю (рис.3.9). При этом для каждого сеанса шифрования данных необходимо использовать новое начальное состояние регистра, которое должно пересылаться по каналу открытым текстом.

Положим

$$M = M_1 M_2 \ldots M_n.$$

Для всех i = 1… n

$$C_i = M_i \oplus P_i,$$

где P_i – старшие k битов операции DES (C_{i-1}).

Отличие от режима обратной связи по шифртексту состоит в методе обновления сдвигового регистра.

Это осуществляется путем отбрасывания старших k битов и дописывания справа P_i.

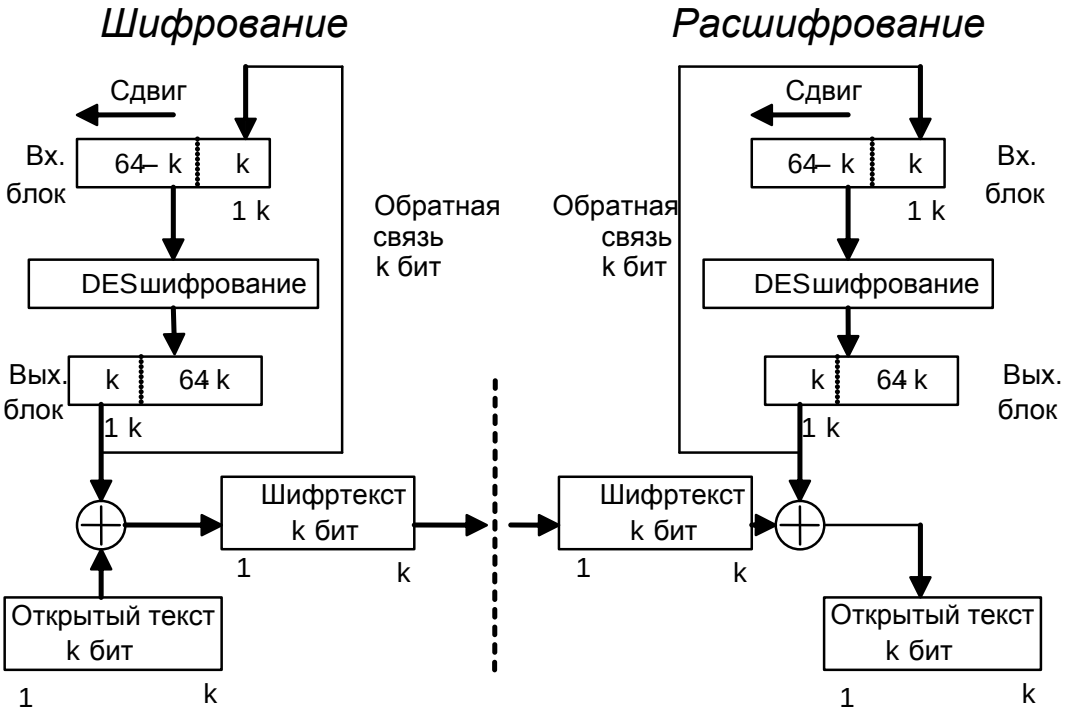


Рисунок 3.9 – Схема алгоритма DES в режиме обратной связи по выходу

3.3. Области применения алгоритма DES

Каждому из рассмотренных режимов (ECB, CBC, CFB, OFB) свойственны свои достоинства и недостатки, что обусловливает области их применения.

Режим ECB хорошо подходит для шифрования ключей: режим CFB, как правило, предназначается для шифрования отдельных символов, а режим OFB нередко применяется для шифрования в спутниковых системах связи.

Режимы CBC и CFB пригодны для аутентификации данных. Эти режимы позволяют использовать алгоритм DES для:

- интерактивного шифрования при обмене данными между терминалом и главной ЭВМ;
- шифрования криптографического ключа в практике автоматизированного распространения ключей;
- шифрования файлов, почтовых отправлений, данных спутников и других практических задач.

Первоначально стандарт DES предназначался для шифрования и расшифрования данных ЭВМ. Однако его применение было обобщено и на аутентификацию.

В системах автоматической обработки данных человек не в состоянии просмотреть данные, чтобы установить, не внесены ли в них какие-либо изменения. При огромных объемах данных, проходящих в современных системах обработки, просмотр занял бы слишком много времени. К тому же избыточность данных может оказаться недостаточной для обнаружения ошибок. Даже в тех случаях, когда просмотр человеком возможен, данные могут быть изменены таким образом, что обнаружить эти изменения человеку очень трудно. Например, "do" может быть заменено на "do not", "$1900" – на

"$9100". Без дополнительной информации человек при просмотре может легко принять измененные данные за подлинные. Такие опасности могут существовать даже при использовании шифрования данных. Поэтому желательно иметь автоматическое средство обнаружения преднамеренных и непреднамеренных изменений данных.

Обыкновенные коды, обнаруживающие ошибки, непригодны, так как если алгоритм образования кода известен, противник может выработать правильный код после внесения изменений в данные. Однако с помощью алгоритма DES можно образовать криптографическую контрольную сумму, которая может защитить как от случайных, так и преднамеренных, но несанкционированных изменений данных.

Этот процесс описывает стандарт для аутентификации данных ЭВМ (FIPS 113). Суть стандарта состоит в том, что данные зашифровываются в режиме обратной связи по шифртексту (режим CFB) или в режиме сцепления блоков шифра (режим CBC), в результате чего получается окончательный блок шифра, представляющий собой функцию всех разрядов открытого текста. После этого сообщение, которое содержит открытый текст, может быть передано с использованием вычисленного окончательного блока шифра, служащего в качестве криптографической контрольной суммы.

Одни и те же данные можно защитить, пользуясь как шифрованием, так и аутентификацией. Данные защищаются от ознакомления шифрованием, а изменения обнаруживаются посредством аутентификации. Алгоритм аутентификации можно применить как к открытому, так и к зашифрованному тексту. При финансовых операциях, когда в большинстве случаев реализуются и шифрование, и аутентификация, последняя применяется и к открытому тексту.

Шифрование и аутентификацию используют для защиты данных, хранящихся в ЭВМ. Во многих ЭВМ пароли зашифровывают необратимым образом и хранят в памяти машины. Когда пользователь обращается к ЭВМ и вводит пароль, последний зашифровывается и сравнивается с хранящимся значением. Если обе зашифрованные величины одинаковы, пользователь получает доступ к машине, в противном случае следует отказ.

Нередко зашифрованный пароль вырабатывают с помощью алгоритма DES, причем ключ полагается равным паролю, а открытый текст – коду идентификации пользователя.

С помощью алгоритма DES можно также зашифровать файлы ЭВМ для их хранения.

Одним из *наиболее важных применений* алгоритма *DES* является *защита сообщений электронной системы платежей (ЭСП) при операциях с широкой клиентурой и между банками*.

Алгоритм DES реализуется в банковских автоматах, терминалах в торговых точках, автоматизированных рабочих местах и главных ЭВМ. Диапазон защищаемых им данных весьма широк – от оплат $50 до переводов на многие миллионы долларов. Гибкость основного алгоритма DES позволяет использовать его в самых разнообразных областях применения электронной системы платежей.

3.4. Комбинирование блочных алгоритмов

В настоящее время блочный алгоритм DES считается относительно безопасным алгоритмом шифрования. Он подвергался тщательному криптоанализу в течение 20 лет, и самым практичным способом его взламывания является метод перебора всех возможных вариантов ключа. Ключ DES имеет длину 56 бит, поэтому существует 2^{56} возможных вариантов такого ключа. Если предположить, что суперкомпьютер может испытать миллион вариантов ключа за секунду, то потребуется 2285 лет для нахождения правильного ключа. Если бы ключ имел длину 128 бит, то потребовалось бы 10^{25} лет (для сравнения: возраст Вселенной около 10^{10} лет).

Нетрудно представить себе, что при постоянном прогрессе возможностей компьютерной техники недалеко то время, когда машины поиска ключа DES методом полного перебора станут экономичными для мощных в финансовом отношении государственных и коммерческих организаций.

Возникает естественный вопрос: нельзя ли использовать DES в качестве строительного блока для создания другого алгоритма с более длинным ключом?

В принципе существует много способов комбинирования блочных алгоритмов для получения новых алгоритмов. Одним из таких способов комбинирования является многократное шифрование, т.е. использование блочного алгоритма несколько раз с разными ключами для шифрования одного и того же блока открытого текста. Двухкратное шифрование блока открытого текста одним и тем же ключом не приводит к положительному результату. При использовании одного и того же алгоритма такое шифрование не влияет на сложность криптоаналитической атаки полного перебора.

Рассмотрим эффективность двухкратного шифрования блока открытого текста с помощью двух разных ключей. Сначала шифруют блок P ключом K_1, а затем получившийся шифртекст $E_{K1}(P)$ шифруют ключом K_2. В результате двухкратного шифрования получают криптограмму

$$C = E_{K2}(E_{K1}(P)).$$

Расшифрование является обратным процессом:

$$P = D_{K1}(D_{K2}(C)).$$

Если блочный алгоритм обладает свойствами группы, то всегда найдется такой ключ K_3, что

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

Если же блочный алгоритм не является группой, то результирующий двухкратно шифрованный блок текста окажется намного сложнее для взламывания методом полного перебора вариантов. Вместо 2^n попыток, где n – длина ключа в битах, потребуется 2^{2n} попыток. В частности, если n=64, то двухкратно зашифрованный блок текста потребует 2^{128} попыток для нахожде-ния ключа.

Однако Р.Меркль и М.Хеллман показали на примере DES, что, используя метод "обмена времени на память" и криптоаналитическую атаку "встреча посредине", можно взломать такую схему двухкратного шифрования за

2^{n+1} попыток [34]. Хотя эта атака потребует очень большого объема памяти (для алгоритма с 56-битовым ключом потребуется 2^{56} 64-битовых блоков или 10^{17} бит памяти).

Более привлекательную идею предложил У.Тачмен [125]. Суть этой идеи состоит в том, чтобы шифровать блок открытого текста P три раза с помощью двух ключей $К_1$ и $К_2$ (рис.3.10). Процедура шифрования:

$$C = E_{к1}(D_{к2}(E_{к1}(P))),$$

т.е. блок открытого текста P сначала шифруется ключом $К_1$, затем расшифровывается ключом $К_2$ и окончательно зашифровывается ключом $К_1$.

Этот режим иногда называют режимом EDE (encrypt-decrypt-encrypt). Введение в данную схему операции расшифрования $D_{к2}$ позволяет обеспечить совместимость этой схемы со схемой однократного использования алгоритма DES. Если в схеме трехкратного использования DES выбрать все ключи одинаковыми, то эта схема превращается в схему однократного использования DES.

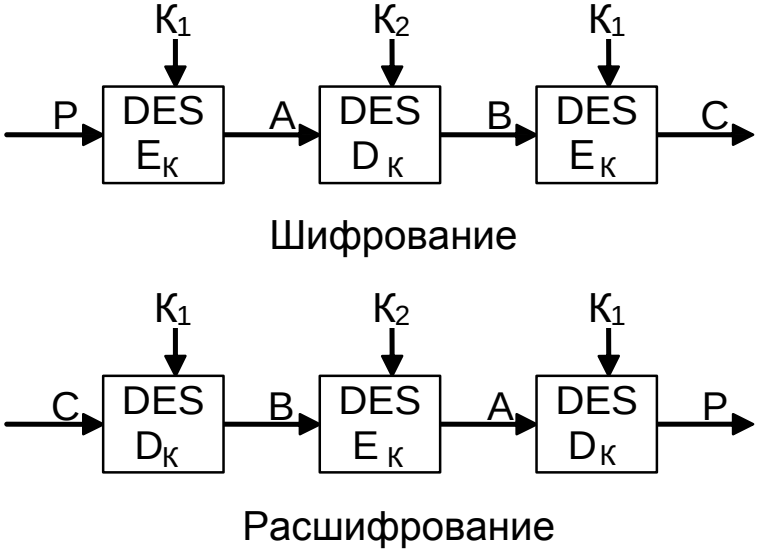


Рисунок 3.10 – Схемы трехкратного применения алгоритма DES с двумя разными ключами

Процедура расшифрования выполняется в обратном порядке:

$$P = D_{к1}(E_{к2}(D_{к1}(C))),$$

т.е. блок шифртекста C сначала расшифровывается ключом $К_1$, затем зашифровывается ключом $К_2$ и окончательно расшифровывается ключом $К_1$.

Если исходный блочный алгоритм имеет n-битовый ключ, то схема трехкратного шифрования имеет 2n-битовый ключ. Чередование ключей $К_1$ и $К_2$ позволяет предотвратить криптоаналитическую атаку "встреча посредине". Данная схема приводится в стандартах X9.17 и ISO 8732 в качестве средства улучшения характеристик алгоритма DES.

При трехкратном шифровании можно применить три различных ключа. При этом возрастает общая длина результирующего ключа. Процедуры шифрования и расшифрования описываются выражениями:

$$C = E_{к3}(D_{к2}(E_{к1}(P))),$$

$$P = D_{к1}(E_{к2}(D_{к3}(C))).$$

Трехключевой вариант имеет еще большую стойкость. Очевидно, что если требуется повысить безопасность большого парка оборудования, использующего DES, то гораздо дешевле переключиться на схемы трехкратных DES, чем переходить на другой тип криптосхем.

3.5. Алгоритм шифрования данных IDEA

Алгоритм IDEA (International Data Encryption Algorithm) является блочным шифром. Он оперирует 64-битовыми блоками открытого текста. Несомненным достоинством алгоритма IDEA является то, что его ключ имеет длину 128 бит. Один и тот же алгоритм используется и для шифрования, и для расшифрования.

Первая версия алгоритма IDEA была предложена в 1990 г., ее авторы – Х.Лей и Дж.Мэсси. Первоначальное название алгоритма PES (Proposed Encryption Standard). Улучшенный вариант этого алгоритма, разработанный в 1991 г., получил название IPES (Improved Proposed Encryption Standard). В 1992 г. IPES изменил свое имя на IDEA. Как и большинство других блочных шифров, алгоритм IDEA использует при шифровании процессы смешивания и рассеивания, причем все процессы легко реализуются аппаратными и программными средствами.

В алгоритме IDEA используются следующие математические операции:

- поразрядное сложение по модулю 2 (операция "исключающее ИЛИ"); операция обозначается как $\oplus$;
- сложение беззнаковых целых по модулю 2^{16} (модуль 65536); операция обозначается как $\boxplus$;
- умножение целых по модулю $(2^{16}+1)$ (модуль 65537), рассматриваемых как беззнаковые целые, за исключением того, что блок из 16 нулей рассматривается как 2^{16}; операция обозначается как $\odot$.

Все операции выполняются над 16-битовыми субблоками.

Эти три операции несовместимы в том смысле, что:

- никакая пара из этих трех операций не удовлетворяет ассоциативному закону,
  например $a \boxplus (b \oplus c) \neq (a \boxplus b) \oplus c$;
- никакая пара из этих трех операций не удовлетворяет дистрибутивному закону,
  например $a \boxplus (b \odot c) \neq (a \boxplus b) \odot (a \boxplus c)$.

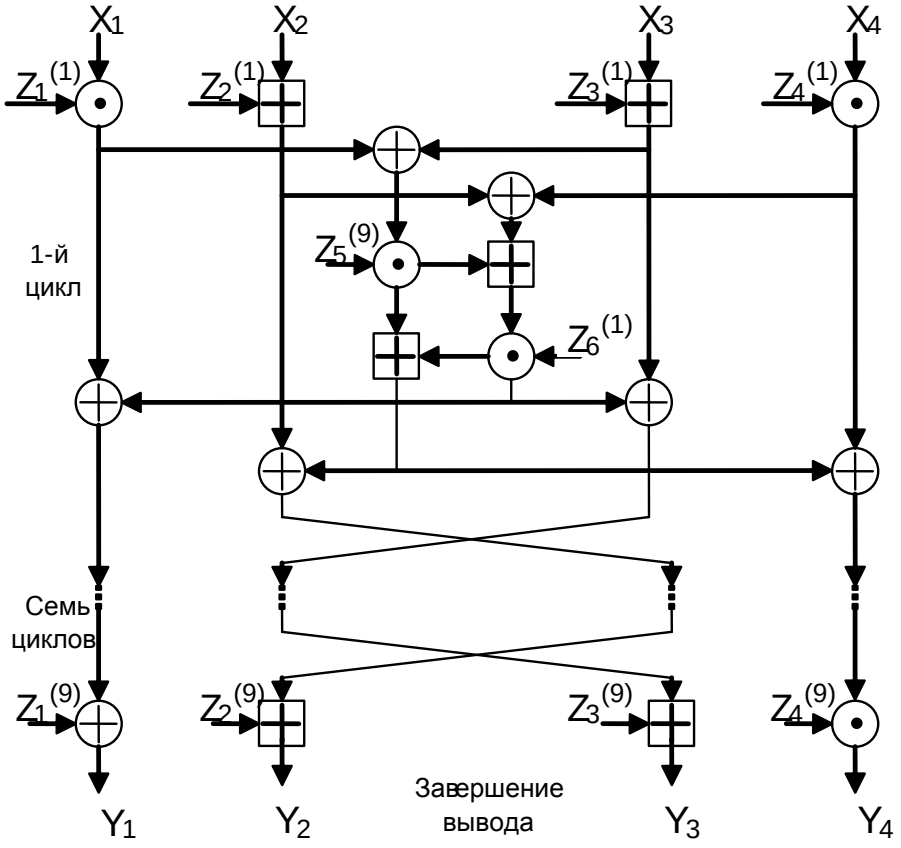
Комбинирование этих трех операций обеспечивает комплексное преобразование входа, существенно затрудняя крипто-анализ IDEA по сравнению с DES, который базируется исключительно на операции "исключающее ИЛИ".

Общая схема алгоритма IDEA приведена на рис.3.11. 64-битовый блок данных делится на четыре 16-битовых субблока. Эти четыре субблока становятся входом в первый цикл алгоритма. Всего выполняется восемь циклов. Между циклами второй и третий субблоки меняются местами. В каждом цикле имеет место следующая последовательность операций:

(1) $\odot$ – умножение субблока X_1 и первого подключа.

(2) ⊞ − сложение субблока X_2 и второго подключа.
(3) ⊞ − сложение субблока X_3 и третьего подключа.
(4) ⊙ − умножение субблока X_4 и четвертого подключа.
(5) □ − сложение результатов шагов (1) и (3).
(6) □ − сложение результатов шагов (2) и (4).
(7) ⊙ − умножение результата шага (5) и пятого подключа.
(8) ⊞ − сложение результатов шагов (6) и (7).
(9) ⊙ − умножение результата шага (8) с шестым подключом.
(10) ⊞ − сложение результатов шагов (7) и (9).
(11) □ − сложение результатов шагов (1) и (9).
(12) □ − сложение результатов шагов (3) и (9).
(13) □ − сложение результатов шагов (2) и (10).
(14) □ − сложение результатов шагов (4) и (10).

Выходом цикла являются четыре субблока, которые получают как результаты выполнения шагов (11), (12), (13) и (14). В завершение цикла переставляют местами два внутренних субблока (за исключением последнего цикла), и в результате формируется вход для следующего цикла.



Обозначения:
X_i – 16-битовый субблок открытого текста, i = 1…4
Y_i – 16-битовый субблок шифртекста, i = 1…4
$Z_j^{(r)}$ – 16-битовый подключ (субблок ключа), j = 1…6, r = 1…8
⊕ – поразрядное суммирование по модулю 2 16-битовых субблоков
⊞ – сложение по модулю 2^{16} 16-битовых целых
⊙ – умножение по модулю 2^{16} 16-битовых целых (с нулевым субблоком, соответствующим 2^{16})

Рисунок 3.11 – Схема алгоритма IDEA (режим шифрования)

После восьмого цикла осуществляют заключительное преобразование выхода:

(1) $\odot$ – умножение субблока X_1 и первого подключа.
(2) $\boxplus$ – сложение субблока X_2 и второго подключа.
(3) $\boxplus$ – сложение субблока X_3 и третьего подключа.
(4) $\odot$ – умножение субблока X_4 и четвертого подключа.

Наконец, эти результирующие четыре субблока $Y_1 \ldots Y_4$ вновь объединяют для получения блока шифртекста.

Создание подключей Z_j также относительно несложно. Алгоритм использует **всего 52 подключа (по шесть для каждого из восьми циклов и еще четыре для преобразования выхода)**. Сначала 128-битовый ключ делят на восемь 16-битовых подключей. Это – первые восемь подключей для алгоритма (шесть подключей – для первого цикла и первые два подключа – для второго цикла). Затем 128-битовый ключ циклически сдвигается влево на 25 бит и снова делится на восемь подключей. Первые четыре из них используют во втором цикле; последние четыре – в третьем цикле. Ключ снова циклически сдвигается влево еще на 25 бит для получения следующих восьми подключей и т.д., пока выполнение алгоритма не завершится.

Расшифрование осуществляют аналогичным образом, за исключением того, что порядок использования подключей становится обратным, причем ряд значений подключей заменяется на обратные значения. Подключи расшифрования являются в основном либо аддитивными, либо мультипликативными обратными величинами подключей шифрования (табл.3.10).

Для реализации алгоритма IDEA было принято предположение, что нулевой субблок равен $2^{16}= -1$; при этом мультипликативная обратная величина от 0 равна 0 [121]. Вычисление значений мультипликативных обратных величин требует некоторых затрат, но это приходится делать только один раз для каждого ключа расшифрования.

Алгоритм IDEA может работать в любом режиме блочного шифра, предусмотренном для алгоритма DES. Алгоритм IDEA обладает рядом преимуществ перед алгоритмом DES. Он значительно безопаснее алгоритма DES, поскольку 128-битовый ключ алгоритма IDEA вдвое больше ключа DES. Внутренняя структура алгоритма IDEA обеспечивает лучшую устойчивость к криптоанализу. Существующие программные реализации алгоритма IDEA примерно вдвое быстрее реализаций алгоритма DES. Алгоритм IDEA шифрует данные

Таблица 3.10

Подключи шифрования и расшифрования алгоритма IDEA

Цикл	Подключи шифрования	Подключи расшифрования

1	$Z_1^{(1)}$	$Z_2^{(1)}$	$Z_3^{(1)}$	$Z_4^{(1)}$	$Z_5^{(1)}$	$Z_6^{(1)}$	$Z_1^{(9)-1}$ $Z_6^{(8)}$	$-Z_2^{(9)}$	$-Z_3^{(9)}$	$Z_4^{(9)-1}$	$Z_5^{(8)}$
2	$Z_1^{(2)}$	$Z_2^{(2)}$	$Z_3^{(2)}$	$Z_4^{(2)}$	$Z_5^{(2)}$	$Z_6^{(2)}$	$Z_1^{(8)-1}$ $Z_6^{(7)}$	$-Z_3^{(8)}$	$-Z_2^{(8)}$	$Z_4^{(8)-1}$	$Z_5^{(7)}$
3	$Z_1^{(3)}$	$Z_2^{(3)}$	$Z_3^{(3)}$	$Z_4^{(3)}$	$Z_5^{(3)}$	$Z_6^{(3)}$	$Z_1^{(7)-1}$ $Z_6^{(6)}$	$-Z_3^{(7)}$	$-Z_2^{(7)}$	$Z_4^{(7)-1}$	$Z_5^{(6)}$
4	$Z_1^{(4)}$	$Z_2^{(4)}$	$Z_3^{(4)}$	$Z_4^{(4)}$	$Z_5^{(4)}$	$Z_6^{(4)}$	$Z_1^{(6)-1}$ $Z_6^{(5)}$	$-Z_3^{(6)}$	$-Z_2^{(6)}$	$Z_4^{(6)-1}$	$Z_5^{(5)}$
5	$Z_1^{(5)}$ $Z_6^{(5)}$	$Z_2^{(5)}$	$Z_3^{(5)}$	$Z_4^{(5)}$	$Z_5^{(5)}$		$Z_1^{(5)-1}$ $Z_6^{(4)}$	$-Z_3^{(5)}$	$-Z_2^{(5)}$	$Z_4^{(5)-1}$	$Z_5^{(4)}$
6	$Z_1^{(6)}$	$Z_2^{(6)}$	$Z_3^{(6)}$	$Z_4^{(6)}$	$Z_5^{(6)}$	$Z_6^{(6)}$	$Z_1^{(4)-1}$ $Z_6^{(3)}$	$-Z_3^{(4)}$	$-Z_2^{(4)}$	$Z_4^{(4)-1}$	$Z_5^{(3)}$
7	$Z_1^{(7)}$	$Z_2^{(7)}$	$Z_3^{(7)}$	$Z_4^{(7)}$	$Z_5^{(7)}$	$Z_6^{(7)}$	$Z_1^{(3)-1}$ $Z_6^{(2)}$	$-Z_3^{(3)}$	$-Z_2^{(3)}$	$Z_4^{(3)-1}$	$Z_5^{(2)}$
8	$Z_1^{(8)}$	$Z_2^{(8)}$	$Z_3^{(8)}$	$Z_4^{(8)}$	$Z_5^{(8)}$	$Z_6^{(8)}$	$Z_1^{(2)-1}$ $Z_6^{(1)}$	$-Z_3^{(2)}$	$-Z_2^{(2)}$	$Z_4^{(2)-1}$	$Z_5^{(1)}$
Преобразование выхода		$Z_1^{(9)}$	$Z_2^{(9)}$	$Z_3^{(9)}$	$Z_4^{(9)}$		$Z_1^{(1)-1}$	$-Z_2^{(1)}$	$-Z_3^{(1)}$	$Z_4^{(1)-1}$	

на IBM PC/486 со скоростью 2,4 Мбит/с. Реализация IDEA на СБИС шифрует данные со скоростью 177 Мбит/с при частоте 25 Мгц. Алгоритм IDEA запатентован в Европе и США.

3.6. Отечественный стандарт шифрования данных

В нашей стране установлен единый алгоритм криптографического преобразования данных для систем обработки информации в сетях ЭВМ, отдельных вычислительных комплексах и ЭВМ, который определяется ГОСТ 28147-89. Стандарт обязателен для организаций, предприятий и учреждений, применяющих криптографическую защиту данных, хранимых и передаваемых в сетях ЭВМ, в отдельных вычислительных комплексах и ЭВМ.

Этот 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.

При описании алгоритма используются следующие обозначения:

L и R – последовательности битов;

LR – конкатенация последовательностей L и R, в которой биты последовательности R следуют за битами последовательности L;

$\oplus$ – операция побитового сложения по модулю 2;

$\boxplus$ – операция сложения по модулю 2^{32} двух 32-разрядных двоичных чисел;

$\boxplus'$ – операция сложения двух 32-разрядных чисел по модулю $2^{32}-1$.

Два целых числа a, b, где $0 \le a, b \le 2^{32}-1$,

$$a= (a_{32}a_{31} \ldots a_2a_1), \quad b = (b_{32}, b_{31}, \ldots, b_2, b_1),$$

представленные в двоичном виде, т.е.

$$a= a_{32}\cdot 2^{31} + a_{31}\cdot 2^{30} +\ldots+ a_2\cdot 2^1 + a_1,$$

$$b = b_{32}\square 2^{31} + b_{31}\square 2^{30} +...+ b_2\square 2^1 + b_1,$$

суммируются по модулю 2^{32} (операция $\boxplus$) по следующему правилу:

$$\mathbf{a \boxplus b = a + b,\ если\ a + b < 2^{32}},$$

$$a \boxplus b = a + b - 2^{32},\ если\ a + b \square 2^{32}.$$

Правила суммирования чисел по модулю $2^{32} - 1$:

$$a \boxplus' b = a + b,\ если\ a + b < 2^{32} - 1,$$

$$a \boxplus' b = a + b - (2^{32} - 1),\ если\ a + b \square 2^{32} - 1.$$

Алгоритм предусматривает четыре режима работы:

- шифрование данных в режиме простой замены;
- шифрование данных в режиме гаммирования;
- шифрование данных в режиме гаммирования с обратной связью;
- выработка имитовставки.

Режим простой замены

Для реализации алгоритма шифрования данных в режиме простой замены используется только часть блоков общей криптосистемы (рис.3.12). Обозначения на схеме:

N_1, N_2 – 32-разрядные накопители;

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

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

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.

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 К в виде восьми 32-разрядных подключей (чисел)* K_i:

$$K=K_7K_6K_5K_4K_3K_2K_1K_0.$$

Последовательность битов блока

$$T_0=(a_1(0), a_2(0), ..., a_{31}(0), a_{32}(0), b_1(0), b_2(0),..., b_{31}(0), b_{32}(0))$$

разбивают на две последовательности по 32 бита: b(0) a(0), где b(0) – левые или старшие биты, a(0) – правые или младшие биты.Эти последовательности вводят в накопители N_1 и N_2 перед началом первого цикла зашифрования. В результате начальное заполнение накопителя N_1

$a(0) = (a_{32}(0), a_{31}(0), ..., a_2(0), a_1(0))$,

32, 31, ... 2, 1 $\square$ номер разряда N_1

начальное заполнение накопителя N_2

$b(0) = (b_{32}(0), b_{31}(0), ..., b_2(0), b_1(0))$.

32, 31, ... 2, 1 □ номер разряда N_2

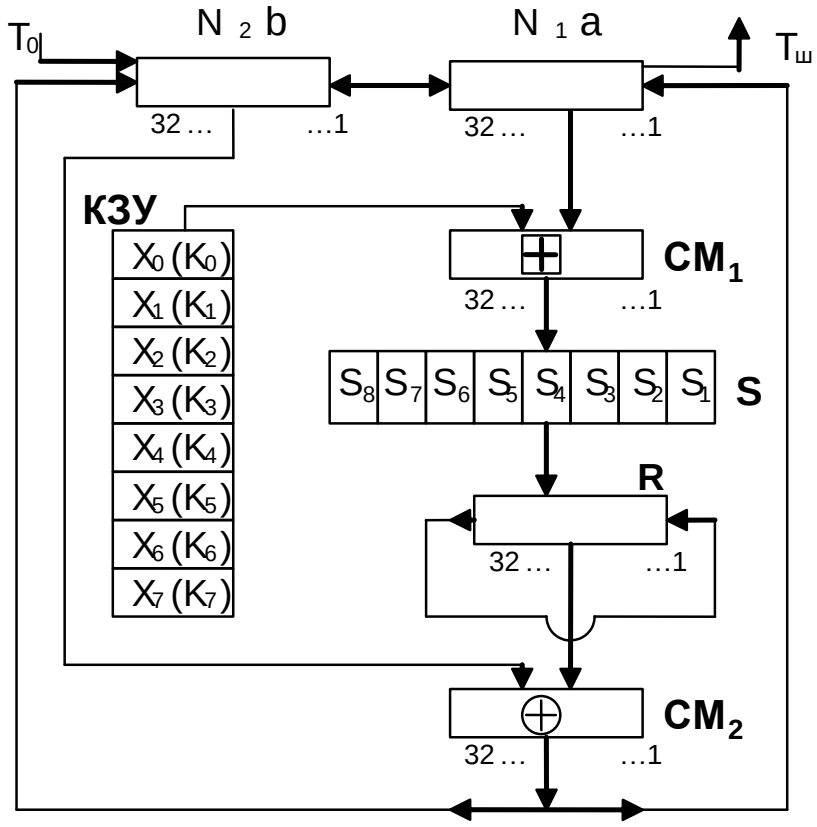


Рисунок 3.12 – Схема реализации режима простой замены

Первый цикл (j=1) процедуры зашифрования 64-разрядного блока открытых данных можно описать уравнениями:

$$\begin{cases} a(1) = f(a(0) \boxplus K_0) \oplus b(0), \\ b(1) = a(0). \end{cases}$$

Здесь a (1) – заполнение N_1 после 1-го цикла зашифрования; b (1) – заполнение N_2 после 1-го цикла зашифрования; f **–** функция шифрования.

Аргументом функции f является сумма по модулю 2^{32} числа a(0) (начального заполнения накопителя N_1) и числа K_0 – подключа, считываемого из накопителя X_0 КЗУ. Каждое из этих чисел равно 32 битам.

Функция f включает две операции над полученной 32-разрядной суммой (a (0) $\boxplus K_0$).

Первая операция называется *подстановкой (заменой)* и выполняется **блоком подстановки** *S*. Блок подстановки S состоит из восьми узлов замены (S-блоков замены) $S_1,S_2,...,S_8$ с памятью 64 бит каждый. Поступающий из $СМ_1$ на блок подстановки S 32-разрядный вектор разбивают на восемь последовательно идущих 4-разрядных векторов, каждый из которых преобразуется в четырехразрядный вектор соответствующим узлом замены. Каждый узел замены можно представить в виде таблицы-перестановки шестнадцати четырехразрядных двоичных чисел в диапазоне 0000…1111. Входной вектор указывает адрес строки в таблице, а число в этой строке является выходным вектором. Затем четырехразрядные выходные векторы последовательно объединяют в 32-разрядный вектор. **Узлы замены (таблицы-перестановки)** представляют собой ключевые элементы, которые являются

общими для сети ЭВМ и редко изменяются. Эти узлы замены должны сохраняться в секрете.

Вторая операция – *циклический сдвиг влево* (на 11 разрядов) 32-разрядного вектора, полученного с выхода блока подстановки S. Циклический сдвиг выполняется регистром сдвига R.

Далее результат работы функции шифрования f суммируют поразрядно по модулю 2 в сумматоре $СМ_2$ с 32-разрядным начальным заполнением b(0) накопителя N_2. Затем полученный на выходе $СМ_2$ результат (значение a(1)) записывают в накопитель N_1, а старое значение N_1 (значение a(0)) переписывают в накопитель N_2 (значение b(1) = a(0)). Первый цикл завершен.

Последующие циклы осуществляются аналогично, при этом во втором цикле из КЗУ считывают заполнение $Х_1$ – подключ $К_1$, в третьем цикле – подключ $К_2$ и т.д., в восьмом цикле – подключ $К_7$. 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: $К_0$, $К_1$, $К_2$,...,$К_6$, $К_7$. В последних восьми циклах с 25-го по 32-й порядок считывания подключей из КЗУ обратный: $К_7$, $К_6$, ..., $К_2$, $К_1$, $К_0$. Таким образом, при зашифровании в 32 циклах осуществляется следующий порядок выборки из КЗУ подключей:

$$К_0, К_1, К_2, К_3, К_4, К_5, К_6, К_7, К_0, К_1, К_2, К_3, К_4, К_5, К_6, К_7,$$
$$К_0, К_1, К_2, К_3, К_4, К_5, К_6, К_7, К_7, К_6, К_5, К_4, К_3, К_2, К_1, К_0.$$

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 $Т_ш$, соответствующим блоку открытых данных $Т_0$.

Уравнения зашифрования в режиме простой замены имеют вид:

$$\begin{cases} a(j) = f(a(j-1) \boxplus K_{j-1(\mathrm{mod}\,8)}) \oplus b(j-1) \\ b(j) = a(j-1) \end{cases} \quad \text{при } j=1\ldots24,$$

$$\begin{cases} a(j) = f(a(j-1) \boxplus K_{32-j}) \oplus b(j-1) \\ b(j) = a(j-1) \end{cases} \quad \text{при } j=25\ldots31,$$

$$\begin{cases} a(32) = a(31) \\ b(32) = f(a(31) \boxplus K_0) \oplus b(31) \end{cases} \quad \text{при } j=32$$

где a (j) = (a_{32}(j), a_{31}(j), ..., a_1(j)) – заполнение N_1 после j-го цикла зашифрования;

b (j) = (b_{32}(j), b_{31}(j), ..., b_1(j)) – заполнение N_2 после j-го цикла зашифрования, j=1…32.

Блок зашифрованных данных $Т_ш$ (64 разряда) выводится из накопителей N_1, N_2 в следующем порядке: из разрядов 1…32 накопителя N_1, затем из разрядов 1…32 накопителя N_2, т.е. начиная с младших разрядов:

$$Т_ш = (a_1(32), a_2(32), ..., a_{32}(32), b_1(32), b_2(32), ..., b_{32}(32)).$$

Остальные блоки открытых данных зашифровываются в режиме простой замены аналогично.

Расшифрование в режиме простой замены. Криптосхема, реализующая алгоритм расшифрования в режиме простой замены, имеет тот же вид, что и при зашифровании (см. рис. 3.12).

В КЗУ вводят 256 бит ключа, на котором осуществлялось зашифрование. Зашифрованные данные, подлежащие расшифрованию, разбиты на блоки $T_ш$ по 64 бита в каждом. Ввод любо-го блока

$$T_ш = (a_1(32), a_2(32), ..., a_{32}(32), b_1(32), b_2(32), ..., b_{32}(32))$$

в накопители N_1 и N_2 производят так, чтобы начальное значение накопителя N_1 имело вид

$(a_{32}(32), a_{31}(32), ..., a_2(32), a_1(32))$,

32, 31, ..., 2, 1 □ номер разряда N_1

а начальное заполнение накопителя N_2 – вид

$(b_{32}(32), b_{31}(32), ..., b_2(32), b_1(32))$.

32, 31, ..., 2, 1 □ номер разряда N_2

Расшифрование осуществляется по тому же алгоритму, что и зашифрование, с тем изменением, что заполнения накопителей $X_0, X_1, ..., X_7$ считываются из КЗУ в циклах расшифрования в следующем порядке:

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

Уравнения расшифрования имеют вид:

$$\begin{cases} a(32-j) = f(a(32-j+1) \boxplus K_{j-1}) \oplus b(32-j+1) \\ b(32-j) = a(32-j+1) \end{cases} \quad \text{при } j=1\ldots8;$$

$$\begin{cases} a(32-j) = f(a(32-j+1) \boxplus K_{32-j(\mathrm{mod}\,8)}) \oplus b(32-j+1) \\ b(32-j) = a(32-j+1) \end{cases} \quad \text{при } j=9\ldots31;$$

$$\begin{cases} a(0) = a(1) \\ b(0) = f(a(1) \boxplus K_0) \oplus b(1) \end{cases} \quad \text{при } j=32.$$

Полученные после 32 циклов работы заполнения накопителей N_1 и N_2 образуют блок открытых данных

$$T_0 = (a_1(0), a_2(0), ..., a_{32}(0), b_1(0), b_2(0),..., b_{32}(0)),$$

соответствующий блоку зашифрованных данных $T_ш$. При этом состояние накопителя N_1

$(a_{32}(0), a_{31}(0), ..., a_2(0), a_1(0))$,

32, 31, ..., 2, 1 □ номер разряда N_1

состояние накопителя N_2

$(b_{32}(0), b_{31}(0), ..., b_2(0), b_1(0))$.

32, 31, ..., 2, 1 □ номер разряда N_2

Аналогично расшифровываются остальные блоки зашифрованных данных.

Если алгоритм зашифрования в режиме простой замены 64-битового блока T_0 обозначить через A, то

$$A(T_0) = A(a(0), b(0)) = (a(32), b(32)) = T_ш.$$

Следует иметь в виду, что 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.

Режим гаммирования

Зашифрование открытых данных в режиме гаммирования. Криптосхема, реализующая алгоритм зашифрования в режиме гаммирования, показана на рис.3.13. Открытые данные разбивают на 64-разрядные блоки

$$T_0^{(1)}, T_0^{(2)}, ..., T_0^{(i)}, ..., T_0^{(m)},$$

где $T_0^{(i)}$ – i-й 64-разрядный блок открытых данных, i = 1…m, 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, т.е.

$$Г_ш = (Г_ш^{(1)}, Г_ш^{(2)}, ..., Г_ш^{(i)}, Г_ш^{(m)}),$$

где $Г_ш^{(i)}$ – i-й 64-разрядный блок, i = 1…m.

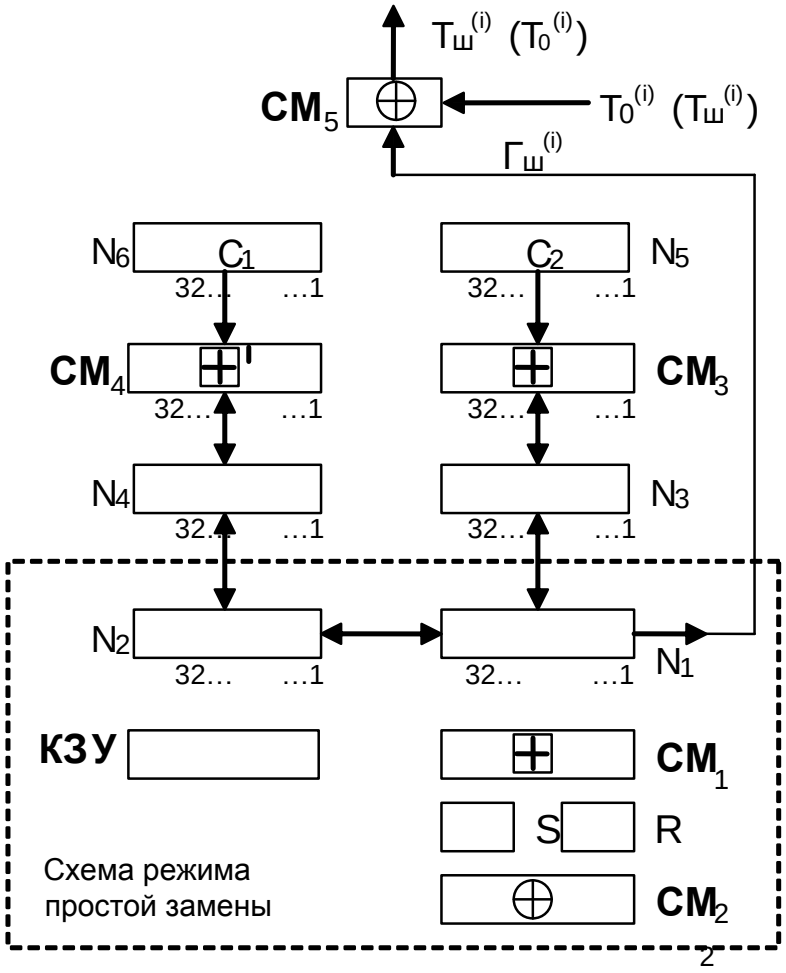


Рисунок 3.13 – Схема реализации режима гаммирования

Число двоичных разрядов в блоке $T_0^{(m)}$ может быть меньше 64, при этом неиспользованная для зашифрования часть гаммы шифра из блока $Г_ш^{(m)}$ отбрасывается.

Уравнение зашифрования данных в режиме гаммирования имеет вид

$$T_ш^{(i)} = T_0^{(i)} \square Г_ш^{(i)},$$

где $Г_ш^{(i)}=A(Y_{i-1} \boxplus C_2, Z_{i-1} \boxplus C_1)$, i=1…m; $T_ш^{(i)}$ – i-й блок 64-разрядного блока зашифрованного текста; $A(\cdot)$ – функция зашифрования в режиме простой замены; C_1, C_2 – 32-разрядные двоичные константы; Y_i, Z_i – 32-разрядные двоичные последовательности.

Величины Y_i, Z_i определяются итерационно по мере формирования гаммы $Г_ш$ следующим образом:

$$(Y_0, Z_0) = A(\tilde{S}),$$

где $\tilde{S}$ – синхропосылка (64-разрядная двоичная последова-тельность),

$$(Y_i, Z_i) = (Y_{i-1} \boxplus C_2,\ Z_{i-1} \boxplus C_1),\ i = 1…m.$$

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

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

В КЗУ вводится 256 бит ключа; в накопители N_1 и N_2 – 64-разрядная двоичная последовательность (синхропосылка)

$$\tilde{S} = (S_1, S_2, ..., S_{64}).$$

Синхропосылка $\tilde{S}$ является исходным заполнением накопителей N_1 и N_2 для последовательной выработки m блоков гаммы шифра.

Исходное заполнение накопителя N_1:

$(S_{32}, S_{31}, ...,S_2, S_1)$;

32, 31, ..., 2, 1 □ номер разряда N_1

исходное заполнение накопителя N_2:

$(S_{64}, S_{63}, ..., S_{34}, S_{33})$.

64, 63, ..., 34, 33 □ номер разряда N_2

Исходное заполнение N_1 и N_2 (синхропосылка $\tilde{S}$) зашифровывается в режиме простой замены. Результат зашифрования

$$A(\tilde{S}) = (Y_0, Z_0)$$

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 суммируют 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 переписывают 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 $Г_ш^{(1)}=(□_1^{(1)}, □_2^{(1)}, ..., □_{63}^{(1)}, □_{64}^{(1)})$, который суммируют поразрядно по модулю 2 в сумматоре $СМ_5$ с первым 64-разрядным блоком открытых данных

$$T_0^{(1)} = (t_1^{(1)}, t_2^{(1)}, ..., t_{63}^{(1)}, t_{64}^{(1)}).$$

В результате суммирования по модулю 2 значений $Г_ш^{(1)}$ и $T_0^{(1)}$ получают первый 64-разрядный блок зашифрованнных данных:

$$T_ш^{(1)} = Г_ш^{(1)} □ T_0^{(1)} = (□_1^{(1)}, □_2^{(1)}, ..., □_{63}^{(1)}, □_{64}^{(1)}),$$

где $□_i^{(1)} = t_i^{(1)} □ □_i^{(1)}$, i = 1…64.

Для получения следующего 64-разрядного блока гаммы шифра $Г_ш^{(2)}$ заполнение N_4 суммируется по модулю $(2^{32} –1)$ в сумматоре $СМ_4$ с константой C_1 из N_6. Результат записывается в N_4. Заполнение N_3 суммируется по модулю 2^{32} в сумматоре $СМ_3$ с константой C_2 из N_5. Результат записывается в N_3. Новое заполнение N_3 переписывают в N_1, а новое заполнение N_4 – в N_2 , при этом заполнения N_3 и N_4 сохраняют. Заполнения N_1, N_2 зашифровывают в режиме простой замены.

Полученное в результате зашифрования заполнение накопителей N_1 и N_2 образует второй 64-разрядный блок гаммы шифра $Г_ш^{(2)}$, который суммируется поразрядно по модулю 2 в сумматоре $СМ_5$ со вторым блоком открытых данных $T_0^{(2)}$:

$$Т_ш^{(2)} = Г_ш^{(2)} \square Т_0^{(2)}.$$

Аналогично вырабатываются блоки гаммы шифра $Г_ш^{(3)}$, $Г_ш^{(4)}$, ..., $Г_ш^{(m)}$ и зашифровываются блоки открытых данных $Т_0^{(3)}$, $Т_0^{(4)}$, ..., $Т_0^{(m)}$.

В канал связи или память ЭВМ передаются синхропосылка $\tilde{S}$ и блоки зашифрованных данных

$$Т_ш^{(1)}, Т_ш^{(2)}, ..., Т_ш^{(m)}.$$

Расшифрование в режиме гаммирования. При расшифровании криптосхема имеет тот же вид, что и при зашифровании (см. рис.3.13).

Уравнение расшифрования:

$$Т_0^{(i)} = Т_ш^{(i)} \square Г_ш^{(i)} = Т_ш^{(i)} \square A (Y_{i-1} \boxplus C_2, Z_{i-1} \boxplus C_1), \ i = 1…m.$$

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

Рассмотрим реализацию процедуры расшифрования. В КЗУ вводят 256 бит ключа, с помощью которого осуществляется зашифрование данных $Т_0^{(1)}$, $Т_0^{(2)}$, ..., $Т_0^{(m)}$. 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 $Т_ш^{(1)}$, $Т_ш^{(2)}$, ..., $Т_ш^{(m)}$ суммируются поразрядно по модулю 2 в сумматоре $СМ_5$ с блоками гаммы шифра $Г_ш^{(1)}$, ..., $Г_ш^{(m)}$. В результате получаются блоки открытых данных

$$Т_0^{(1)}, Т_0^{(2)}, ..., Т_0^{(m)};$$

при этом $Т_0^{(m)}$ может содержать меньше 64 разрядов.

Режим гаммирования с обратной связью

Зашифрование открытых данных в режиме гаммирования с обратной связью. Криптосхема, реализующая алгоритм зашифрования в режиме гаммирования с обратной связью, имеет вид, показанный на рис. 3.14.

Открытые данные, разбитые на 64-разрядные блоки $Т_0^{(1)}$, $Т_0^{(2)}$, ..., $Т_0^{(m)}$, зашифровываются в режиме гаммирования с обратной связью путем поразрядного сложения по модулю 2 с гаммой шифра $Г_ш$, которая вырабатывается блоками по 64 бита:

$$Г_ш = (Г_ш^{(1)}, Г_ш^{(2)}, ..., Г_ш^{(m)}).$$

Число двоичных разрядов в блоке $Т_0^{(m)}$ может быть меньше 64, при этом неиспользованная для шифрования часть гаммы шифра из блока $Г_ш^{(m)}$ отбрасывается.

Уравнения зашифрования в режиме гаммирования с обратной связью имеют вид:

$$Т_ш^{(1)} = A (\tilde{S}) \square Т_0^{(1)} = Г_ш^{(1)} \square Т_0^{(1)},$$

$$Т_ш^{(i)} = A (Т_ш^{(i-1)}) \square Т_0^{(i)} = Г_ш^{(i)} \square Т_0^{(i)}, \ i = 2…m.$$

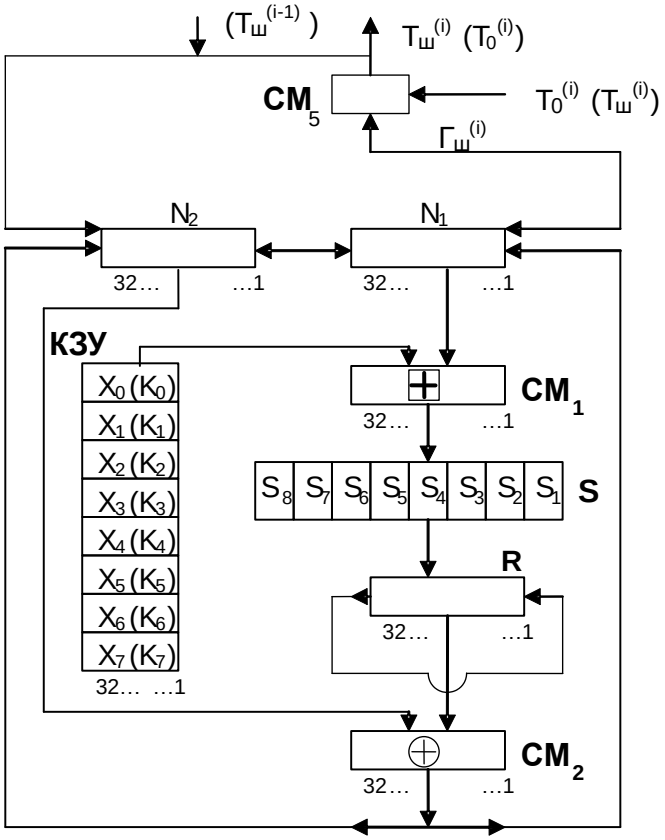


Рисунок 3.14 – Схема реализации режима гаммирования с обратной связью

Здесь $Т_ш^{(i)}$ – i-й 64-разрядный блок зашифрованного текста; A(·) – функция зашифрования в режиме простой замены; m – определяется объемом открытых данных.

Аргументом функции A (·) на первом шаге итеративного алгоритма является 64-разрядная синхропосылка $\tilde{S}$, а на всех последующих шагах – предыдущий блок зашифрованных дан-ных $Т_ш^{(i–1)}$.

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 синхро-посылка $\tilde{S} = (S_1, S_2, ..., S_{64})$ 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 $Г_ш^{(1)}=A(\tilde{S})$, который суммируется поразрядно по модулю 2 в сумматоре $СМ_5$ с первым 64-разрядным блоком открытых данных

$$Т_0^{(1)} = (t_1^{(1)}, t_2^{(1)},..., t_{64}^{(1)}).$$

В результате получают первый 64-разрядный блок зашифрованных данных

$$Т_ш^{(1)} = Г_ш^{(1)} \square Т_0^{(1)},$$

где $Т_ш^{(1)} = (\square_1^{(1)}, \square_2^{(1)}, ..., \square_{64}^{(1)})$.

Блок зашифрованных данных $Т_ш^{(1)}$ одновременно является также исходным состоянием накопителей N_1, N_2 для выработки второго блока гаммы шифра $Г_ш^{(2)}$, и поэтому по обратной связи $Т_ш^{(1)}$ записывается в указанные накопители N_1 и N_2.

Заполнение накопителя N_1

($\square_{32}^{(1)}$, $\square_{31}^{(1)}$, …, $\square_{2}^{(1)}$, $\square_{1}^{(1)}$).

32, 31, ..., 2, 1 □ номер разряда N_1

Заполнение накопителя N_2

($\square_{64}^{(1)}$, $\square_{63}^{(1)}$, …, $\square_{34}^{(1)}$, $\square_{33}^{(1)}$).

32, 31, ..., 2, 1 □ номер разряда N_2

Заполнение накопителей N_1 и N_2 зашифровывается в режиме простой замены. Полученное в результате зашифрования заполнение накопителей N_1 и N_2 образует второй 64-разрядный блок гаммы шифра $Г_ш^{(2)}$, который суммируется поразрядно по модулю 2 в сумматоре $СМ_5$ со вторым блоком открытых данных $Т_0^{(2)}$:

$$Г_ш^{(2)} \square Т_0^{(2)} = Т_ш^{(2)}.$$

Выработка последующих блоков гаммы шифра $Г_ш^{(i)}$ и зашифрование соответствующих блоков открытых данных $Т_0^{(i)}$ (i=3…m) производится аналогично.

Если длина последнего m-го блока открытых данных $Т_0^{(m)}$ меньше 64 разрядов, то из $Г_ш^{(m)}$ используется только соответствующее число разрядов гаммы шифра, остальные разряды отбрасываются.

В канал связи или память ЭВМ передаются синхропосылка $\tilde{S}$ и блоки зашифрованных данных $Т_ш^{(1)}$, $Т_ш^{(2)}$, ..., $Т_ш^{(m)}$.

Расшифрование в режиме гаммирования с обратной связью. При расшифровании криптосхема имеет тот же вид, что и при зашифровании (см. рис.3.14).

Уравнения расшифрования:

$$Т_0^{(1)} = A(\tilde{S}) \square Т_ш^{(1)} = Г_ш^{(1)} \square Т_ш^{(1)},$$

$$Т_0^{(i)} = Г_ш^{(i)} \square Т_ш^{(i)} = A\,(Т_ш^{(i-1)}) \square Т_ш^{(i)},\ i = 2…m.$$

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 $Т_0^{(1)}$, $Т_0^{(2)}$, ..., $Т_0^{(m)}$. В накопители N_1 и N_2 вводится синхропосылка $\tilde{S}$. Исходное заполнение накопителей N_1 и N_2 (синхропосылка $\tilde{S}$) зашифровывается в режиме простой замены. Полученное в результате зашифрования заполнение N_1 и N_2 образует первый блок гаммы шифра

$$Г_ш^{(1)} = A(\tilde{S}),$$

который суммируется поразрядно по модулю 2 в сумматоре $СМ_5$ с блоком зашифрованных данных $Т_ш^{(1)}$.

В результате получается первый блок открытых данных

$$Т_0^{(1)} = Г_ш^{(1)} \square Т_ш^{(1)}.$$

Блок зашифрованных данных $Т_ш^{(1)}$ является исходным заполнением накопителей N_1 и N_2 для выработки второго блока гаммы шифра $Г_ш^{(2)}$: $Г_ш^{(2)} = A(Т_ш^{(1)})$. 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 $Т_ш^{(2)}$. В результате получают второй блок от-

крытых данных. Аналогично в N_1, N_2 последовательно записывают блоки зашифрованных данных $Т_Ш^{(2)}$, $Т_Ш^{(3)}$, ..., $Т_Ш^{(m)}$, 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 $Т_Ш^{(3)}$,$Т_Ш^{(4)}$, ..., $Т_Ш^{(m)}$.

В результате получают блоки открытых данных

$$Т_0^{(3)}, Т_0^{(4)}, ..., Т_0^{(m)},$$

при этом последний блок открытых данных $Т_0^{(m)}$ может содержать меньше 64 разрядов.

Режим выработки имитовставки

Имитовставка – это блок из P бит, который вырабатывают по определенному правилу из открытых данных с использованием ключа и затем добавляют к зашифрованным данным для обеспечения их имитозащиты.

Имитозащита – это защита системы шифрованной связи от навязывания ложных данных.

В стандарте ГОСТ 28147-89 определяется процесс выработки имитовставки, который единообразен для любого из режимов шифрования данных. Имитовставка $И_р$ вырабатывается из блоков открытых данных либо перед шифрованием всего сообщения, либо параллельно с шифрованием по блокам. Первые блоки открытых данных, которые участвуют в выработке имитовставки, могут содержать служебную информацию (например, адресную часть, время, синхропосылку) и не зашифровываются.

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 $Т_0^{(i)}$, i = 1…m.

Первый блок открытых данных $Т_0^{(1)}$ подвергают преобразованию $\tilde{A}(\cdot)$, 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 $\tilde{A}(Т_0^{(1)})$ суммируют по модулю 2 со вторым блоком открытых данных $Т_0^{(2)}$. Результат суммирования $(\tilde{A}(Т_0^{(1)}) \oplus Т_0^{(2)})$ снова подвергают преобразованию $\tilde{A}(\cdot)$.

Полученное 64-разрядное число $\tilde{A}(\tilde{A}(Т_0^{(1)}) \oplus Т_0^{(2)})$ суммируют по модулю 2 с третьим блоком $Т_0^{(3)}$ и снова подвергают преобразованию $\tilde{A}(\cdot)$, получая 64-разрядное число

$$\tilde{A}(\tilde{A}(\tilde{A}(Т_0^{(1)}) \oplus Т_0^{(2)}) \oplus Т_0^{(3)}), \quad \text{и т.д.}$$

Последний блок $Т_0^{(m)}$ (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 $\tilde{A}(\cdot)$.

Из полученного 64-разрядного числа выбирают отрезок $И_p$ (имитовставку) длиной P бит:

$$И_p = [a^{(m)}_{32-p+1}(16), a^{(m)}_{32-p+2}(16), ..., a^{(m)}_{32}(16)],$$

где $a_i^{(m)}$ – i-й бит 64-разрядного числа, полученного после 16-го цикла последнего преобразования $\tilde{A}(\cdot)$, $32 - p + 1 \le i \le 32$.

Имитовставка $И_p$ передается по каналу связи или в память ЭВМ в конце зашифрованных данных, т.е.

$$Т_ш^{(1)}, Т_ш^{(2)}, ..., Т_ш^{(m)}, И_p.$$

Поступившие к получателю зашифрованные данные

$$Т_ш^{(1)}, Т_ш^{(2)}, ..., Т_ш^{(m)}$$

расшифровываются, и из полученных блоков открытых данных $Т_0^{(1)}, Т_0^{(2)}, ..., Т_0^{(m)}$ аналогичным образом вырабатывается имитовставка $И_p'$. Эта имитовставка $И_p'$ сравнивается с имитовставкой $И_p$, полученной вместе с зашифрованными данными из канала связи или из памяти ЭВМ. В случае несовпадения имитовставок полученные при расшифровании блоки открытых данных $Т_0^{(1)}, Т_0^{(2)}, ..., Т_0^{(m)}$ считают ложными.

3.7. Блочные и поточные шифры

Проектирование алгоритмов шифрования данных основано на рациональном выборе функций, преобразующих исходные (незашифрованные) сообщения в шифртекст. Идея непосредствен-ного применения такой функции ко всему сообщению реализуется очень редко. Практически все применяемые криптографические методы связаны с разбиением сообщения на большое число фраг-ментов (или знаков) фиксированного размера, каждый из которых шифруется отдельно. Такой подход существенно упрощает задачу шифрования, так как сообщения обычно имеют различную длину.

Различают три основных способа шифрования: поточные шифры, блочные шифры и блочные шифры с обратной связью. Для классификации методов шифрования данных следует выбрать некоторое количество характерных признаков, которые можно применить для установления различий между этими методами. Будем полагать, что каждая часть или каждый знак сообщения шифруется отдельно в заданном порядке.

Можно выделить следующие характерные признаки методов шифрования данных:

1) Выполнение операций с отдельными битами или блоками. Известно, что для некоторых методов шифрования знаком сообщения, над которым производят операции шифрования, является отдельный бит, тогда как другие методы оперируют конечным множеством битов, обычно называемым блоком.
2) Зависимость или независимость функции шифрования от результатов шифрования предыдущих частей сообщения.
3) Зависимость или независимость шифрования отдельных знаков от их положения в тексте. В некоторых методах знаки шифруются с использова-

нием одной и той же функции независимо от их положения в сообщении, а в других методах, например при поточном шифровании, различные знаки сообщения шифруются с учетом их положения в сообщении. Это свойство называют **позиционной зависимостью** или независимостью шифра.

4) Симметрия или асимметрия функции шифрования. Эта важная характеристика определяет существенное различие между обычными симметричными (одноключевыми) криптосистемами и асимметричными (двухключевыми) криптосистемами с открытым ключом. Основное различие между ними состоит в том, что в асимметричной криптосистеме знания ключа шифрования (или расшифрования) недостаточно для раскрытия соответствующего ключа расшифрования (или шифрования).

В табл. 3.11 приведены типы криптосистем и их основные характеристики.

Таблица 3.11

Основные характеристики криптосистем

Тип криптосистемы	Операции с битами или блоками	Зависимость от предыдущих знаков	Позиционная зависимость	Наличие симметрии функции шифрования
Поточного шифрования	Биты	Не зависит	Зависит	Симметричная
Блочного шифрования	Блоки	Не зависит	Не зависит	Симметричная или несимметричная
С обратной связью от шифртекста	Биты или блоки	Зависит	Не зависит	Симметричная

Поточное шифрование состоит в том, что биты открытого текста складываются по модулю 2 с битами псевдослучайной последовательности. К достоинствам поточных шифров относятся высокая скорость шифрования, относительная простота реализации и отсутствие размножения ошибок. Недостатком является необходимость передачи информации синхронизации перед заголовком сообщения, которая должна быть принята до расшифрования любого сообщения. Это обусловлено тем, что если два различных сообщения шифруются на одном и том же ключе, то для расшифрования этих сообщений требуется одна и та же псевдослучайная последовательность. Такое положение может создать угрозу криптостойкости системы. Поэтому часто используют дополнительный, случайно выбираемый ключ сообщения, который передается в начале сообщения и применяется для модификации ключа шифрования. В результате разные сообщения будут шифроваться с помощью различных последовательностей.

Поточные шифры широко применяются для шифрования преобразованных в цифровую форму речевых сигналов и цифровых данных, требующих оперативной доставки потребителю информации. До недавнего времени такие применения были преобладающими для данного метода шифрования. Это обусловлено, в частности, относительной простотой проектирования и

реализации генераторов хороших шифрующих последовательностей. Но самым важным фактором, конечно, является отсутствие размножения ошибок в поточном шифре. Стандартным методом генерирования последовательностей для поточного шифрования является метод, применяемый в стандарте шифрования DES в режиме обратной связи по выходу (режим OFB).

При **блочном шифровании** открытый текст сначала разбивается на равные по длине блоки, затем применяется зависящая от ключа функция шифрования для преобразования блока открытого текста длиной m бит в блок шифртекста такой же длины. Достоинством блочного шифрования является то, что каждый бит блока шифртекста зависит от значений всех битов соответствующего блока открытого текста, и никакие два блока открытого текста не могут быть представлены одним и тем же блоком шифртекста. Алгоритм блочного шифрования может использоваться в различных режимах. Четыре режима шифрования алгоритма DES фактически применимы к любому блочному шифру: режим прямого шифрования или шифрования с использованием электронной книги кодов ECB (Electronic code Book), шифрование со сцеплением блоков шифртекста CBC (Cipher block chaining), шифрование с обратной связью по шифртексту CFB (Cipher feedback) и шифрование с обратной связью по выходу OFB (Output feedback).

Основным достоинством прямого блочного шифрования ECB является то, что в хорошо спроектированной системе блочного шифрования небольшие изменения в шифртексте вызывают большие и непредсказуемые изменения в соответствующем открытом тексте, и наоборот. Вместе с тем применение блочного шифра в данном режиме имеет серьезные недостатки. Первый из них заключается в том, что вследствие детерминированного характера шифрования при фиксированной длине блока 64 бита можно осуществить криптоанализ шифртекста "со словарем" в ограниченной форме. Это обусловлено тем, что идентичные блоки открытого текста длиной 64 бита в исходном сообщении представляются идентичными блоками шифртекста, что позволяет криптоаналитику сделать определенные выводы о содержании сообщения. Другой потенциальный недостаток этого шифра связан с размножением ошибок. Результатом изменения только одного бита в принятом блоке шифртекста будет неправильное расшифрование всего блока. Это, в свою очередь, приведет к появлению искаженных битов (от 1 до 64) в восстановленном блоке исходно-го текста.

Из-за отмеченных недостатков блочные шифры редко применяются в указанном режиме для шифрования длинных сообщений. Однако в финансовых учреждениях, где сообщения часто состоят из одного или двух блоков, блочные шифры широко используют в режиме прямого шифрования. Такое применение обычно связано с возможностью частой смены ключа шифрования, поэтому вероятность шифрования двух идентичных блоков открытого текста на одном и том же ключе очень мала.

Криптосистема с открытым ключом также является системой блочного шифрования и должна оперировать блоками довольно большой длины. Это обусловлено тем, что криптоаналитик знает открытый ключ шифрования и мог бы заранее вычислить и составить таблицу соответствия блоков откры-

того текста и шифртекста. Если длина блоков мала, например 30 бит, то число возможных блоков не слишком большое (при длине 30 бит это $2^{30} = \approx 10^9$), и может быть составлена полная таблица, позволяющая моментально расшифровать любое сообщение с использованием известного открытого ключа. Асимметричные криптосистемы с открытым ключом подробно разбираются в следующей главе.

Наиболее часто блочные шифры применяются в **системах шифрования с обратной связью**. Системы шифрования с обратной связью встречаются в различных практических вариантах. Как и при блочном шифровании, сообщения разбивают на ряд блоков, состоящих из m бит. Для преобразования этих блоков в блоки шифртекста, которые также состоят из m бит, используются специальные функции шифрования. Однако если в блочном шифре такая функция зависит только от ключа, то в блочных шифрах с обратной связью она зависит как от ключа, так и от одного или более предшествующих блоков шифртекста.

Практически важным шифром с обратной связью является шифр со сцеплением блоков шифртекста CBC. В этом случае m бит предыдущего шифртекста суммируются по модулю 2 со следующими m битами открытого текста, а затем применяется алгоритм блочного шифрования под управлением ключа для получения следующего блока шифртекста. Еще один вариант шифра с обратной связью получается из стандартного режима CFB алгоритма DES, т.е. режима с обратной связью по шифртексту.

Достоинством криптосистем блочного шифрования с обратной связью является возможность применения их для обнаружения манипуляций сообщениями, производимых активными перехватчиками. При этом используется факт размножения ошибок в таких шифрах, а также способность этих систем легко генерировать код аутентификации сообщений. Поэтому системы шифрования с обратной связью используют не только для шифрования сообщений, но и для их аутентификации. Криптосистемам блочного шифрования с обратной связью свойственны некоторые недостатки. Основным из них является размножение ошибок, так как один ошибочный бит при передаче может вызвать ряд ошибок в расшифрованном тексте. Другой недостаток связан с тем, что разработка и реализация систем шифрования с обратной связью часто оказываются более трудными, чем систем поточного шифрования.

На практике для шифрования длинных сообщений применяют поточные шифры или шифры с обратной связью. Выбор конкретного типа шифра зависит от назначения системы и предъявляемых к ней требований.

4. АСИММЕТРИЧНЫЕ КРИПТОСИСТЕМЫ

4.1. Концепция криптосистемы с открытым ключом

Эффективными системами криптографической защиты данных являются асимметричные криптосистемы, называемые также криптосистемами с открытым ключом. В таких системах **для зашифрования данных используется один ключ, а для расшифрования – другой ключ** (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 расшифрования данных получатель зашифрованной информации использует второй ключ, который является *секретным*. Разумеется, ключ расшифрования не может быть определен из ключа зашифрования.

Обобщенная схема асимметричной криптосистемы с открытым ключом показана на рис. 4.1. В этой криптосистеме применяют два различных ключа: $К_в$ – открытый ключ отправителя А; $k_в$ – секретный ключ получателя В. Генератор ключей целесообразно располагать на стороне получателя В (чтобы не пересылать секретный ключ $k_в$ по незащищенному каналу). Значения ключей $К_в$ и $k_в$ зависят от начального состояния генератора ключей.

Раскрытие секретного ключа $k_в$ по известному открытому ключу $К_в$ должно быть вычислительно неразрешимой задачей.

Характерные особенности асимметричных криптосистем:

1. Открытый ключ $К_в$ и криптограмма С могут быть отправлены по незащищенным каналам, т.е. противнику известны $К_в$ и С.

2. Алгоритмы шифрования и расшифрования

$$E_в : M \to C,$$
$$D_в : C \to M$$

являются открытыми.

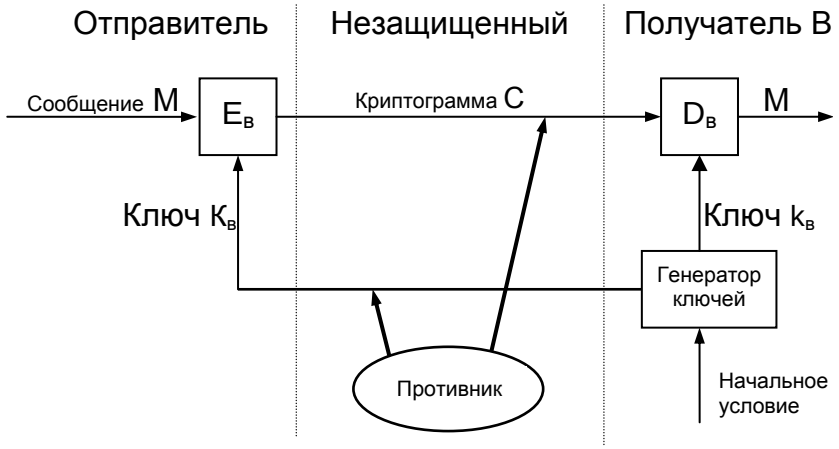


Рисунок 4.1 – Обобщенная схема асимметричной криптосистемы с открытым ключом

Защита информации в асимметричной криптосистеме основана на секретности ключа $k_в$.

У.диффи и м.хеллман сформулировали требования, выполнение которых обеспечивает безопасность асимметричной криптосистемы:

1. Вычисление пары ключей ($К_в$, $k_в$) получателем В на основе начального условия должно быть простым.

2. Отправитель А, зная открытый ключ $К_в$ и сообщение М, может легко вычислить криптограмму

$$C = E_{K_в}(M) = E_в(M). \qquad (4.1)$$

3. Получатель В, используя секретный ключ $k_в$ и криптограмму С, может легко восстановить исходное сообщение

$$M = D_{k_в}(C) = D_в(C) = D_в[E_в(M)]. \qquad (4.2)$$

4. Противник, зная открытый ключ $К_в$, при попытке вычислить секретный ключ $k_в$ наталкивается на непреодолимую вычислительную проблему.

5. Противник, зная пару ($К_в$, С), при попытке вычислить исходное сообщение М наталкивается на непреодолимую вычислительную проблему [28].

4.2. Однонаправленные функции

Концепция асимметричных криптографических систем с открытым ключом основана на применении однонаправленных функций. Неформально однонаправленную функцию можно определить следующим образом. Пусть X и Y – некоторые произвольные множества. Функция

$$f : X \to Y$$

является однонаправленной, если для всех $x \in X$ можно легко вычислить функцию

$$y = f(x), \text{ где } y \in Y.$$

И в то же время для большинства $y \in Y$ достаточно сложно получить значение $x \in X$, такое, что $f(x)=y$ (при этом полагают, что существует по крайней мере одно такое значение x).

Основным критерием отнесения функции f к классу однонаправленных функций является отсутствие эффективных алгоритмов обратного преобразования $Y \to X$.

В качестве первого примера однонаправленной функции рассмотрим целочисленное умножение. Прямая задача – вычисление произведения двух очень больших целых чисел P и Q, т.е. нахождение значения

$$N = P * Q, \qquad (4.3)$$

является относительно несложной задачей для ЭВМ.

Обратная задача – разложение на множители большого целого числа, т.е. нахождение делителей P и Q большого целого числа $N = P * Q$, является практически неразрешимой задачей при достаточно больших значениях N. По современным оценкам теории чисел при целом $N \approx 2^{664}$ и $P \approx Q$ для разложения числа N потребуется около 10^{23} операций, т.е. задача практически неразрешима на современных ЭВМ.

Следующий характерный пример однонаправленной функции – это модульная экспонента с фиксированными основанием и модулем. Пусть A и N – целые числа, такие, что $1 \le A < N$. Определим множество Z_N:

$$Z_N = \{0, 1, 2, ..., N-1\}.$$

Тогда модульная экспонента с основанием A по модулю N представляет собой функцию

$$f_{A,N} : Z_N \to Z_N,$$

$$f_{A,N}(x) = A^x \pmod N, \qquad (4.4)$$

где X – целое число, $1 \le x \le N-1$.

Существуют эффективные алгоритмы, позволяющие достаточно быстро вычислить значения функции $f_{A,N}(x)$.

Если $y = A^x$, то естественно записать $x = \log_A(y)$.

Поэтому задачу обращения функции $f_{A,N}(x)$ называют задачей нахождения дискретного логарифма или задачей дискретного логарифмирования.

Задача дискретного логарифмирования формулируется следующим образом. Для известных целых A, N, Y найти целое число X, такое, что

$$A^x \bmod N = y.$$

Алгоритм вычисления дискретного логарифма за приемлемое время пока не найден. Поэтому модульная экспонента считается однонаправленной функцией.

По современным оценкам теории чисел при целых числах $A \approx 2^{664}$ и $N \approx 2^{664}$ решение задачи дискретного логарифмирования (нахождение показателя степени x для известного y) потребует около 10^{26} операций, т.е. эта задача имеет в 10^3 раз большую вычислительную сложность, чем задача разложения на множители. При увеличении длины чисел разница в оценках сложности задач возрастает.

Следует отметить, что пока не удалось доказать, что не существует эффективного алгоритма вычисления дискретного логарифма за приемлемое время. Исходя из этого, модульная экспонента отнесена к однонаправленным функциям условно, что, однако, не мешает с успехом применять ее на практике.

Вторым важным классом функций, используемых при построении криптосистем с открытым ключом, являются так называемые однонаправленные функции с "потайным ходом" (с лазейкой). Дадим неформальное определение такой функции. Функция

$$f : X \to Y$$

относится к классу однонаправленных функций с "потайным ходом" в том случае, если она является однонаправленной и, кроме того, возможно эффективное вычисление обратной функции, если известен "потайной ход" (секретное число, строка или другая информация, ассоциирующаяся с данной функцией).

В качестве примера однонаправленной функции с "потайным ходом" можно указать используемую в криптосистеме RSA модульную экспоненту с

фиксированными модулем и показателем степени. Переменное основание модульной экспоненты используется для указания числового значения сообщения M либо криптограммы C

4.3. Криптосистема шифрования данных RSA

Алгоритм RSA предложили в 1978 г. три автора: P.Райвест (Rivest), A.Шамир (Shamir) и A.Адлеман (Adleman). Алгоритм получил свое название по первым буквам фамилий его авторов. Алгоритм RSA стал первым полноценным алгоритмом с открытым ключом, который может работать как в режиме шифрования данных, так и в режиме электронной цифровой подписи.

Надежность алгоритма основывается на трудности факторизации больших чисел и трудности вычисления дискретных логарифмов.

В криптосистеме RSA открытый ключ $K_в$, секретный ключ $k_в$, сообщение M и криптограмма C принадлежат множеству це-лых чисел

$$Z_N = \{0, 1, 2, ..., N-1\}, \qquad (4.5)$$

где N – модуль:

$$N = P*Q. \qquad (4.6)$$

Здесь P и Q – случайные большие простые числа. Для обеспечения максимальной безопасности выбирают P и Q равной длины и хранят в секрете.

Множество 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:

$$1< K_в \le \varphi(N), \quad НОД(K_в, \varphi(N)) = 1, \qquad (4.7)$$

$$\varphi(N)=(P-1)(Q-1), \qquad (4.8)$$

где $\varphi(N)$ – функция Эйлера.

Функция Эйлера $\varphi(N)$ указывает количество положительных целых чисел в интервале от 1 до N, которые взаимно просты с N.

Второе из указанных выше условий означает, что открытый ключ $K_в$ и функция Эйлера $\varphi(N)$ должны быть взаимно простыми.

Далее, используя расширенный алгоритм Евклида, вычисляют секретный ключ $k_в$, такой, что

$$k_в * K_в \equiv 1 \pmod{\varphi(N)} \qquad (4.9)$$

или

$$k_в = K_в^{-1} \pmod{(P-1)(Q-1)}.$$

Это можно осуществить, так как получатель B знает пару простых чисел (P,Q) и может легко найти $\varphi(N)$. Заметим, что $k_в$ и N должны быть взаимно простыми.

Открытый ключ $K_в$ используют для шифрования данных, а секретный ключ $k_в$ – для расшифрования.

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

$$C = E_{K_в}(M) = E_B(M) = M^{K_в} \pmod{N}. \qquad (4.10)$$

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 $C = M^{K_в} (\text{mod } N)$, т.е. определение значения M по известным значениям C, $K_в$ и N, практически не осуществимо при $N \approx 2^{512}$.

Однако обратную задачу, т.е. задачу расшифрования криптограммы C, можно решить, используя пару (секретный ключ $k_в$, криптограмма C) по следующей формуле:

$$M = D_{k_в}(C) = D_B(C) = C^{k_в} (\text{mod } N). \qquad (4.11)$$

Процесс расшифрования можно записать так:

$$D_B(E_B(M)) = M. \qquad (4.12)$$

Подставляя в (4.12) значения (4.10) и (4.11), получаем:

$$(M^{K_в})^{k_в} = M \ (\text{mod } N)$$

или

$$M^{K_в k_в} = M \ (\text{mod } N). \qquad (4.13)$$

Величина φ (N) играет важную роль в теореме Эйлера, которая утверждает, что если НОД (x, N) =1, то

$$x^{\varphi(N)} \equiv 1 \ (\text{mod } N),$$

или в несколько более общей форме

$$x^{n \cdot \varphi(N)+1} \equiv x \ (\text{mod } N). \qquad (4.14)$$

Сопоставляя выражения (4.13) и (4.14), получаем

$$K_в * k_в = n * \varphi(N) + 1$$

или, что то же самое,

$$K_в * k_в \equiv 1 \ (\text{mod } \varphi(N)).$$

Именно поэтому для вычисления секретного ключа $k_в$ используют соотношение (4.9).

Таким образом, если криптограмму

$$C = M^{K_в} (\text{mod } N)$$

возвести в степень $k_в$, то в результате восстанавливается исходный открытый текст M, так как

$$(M^{K_в})^{k_в} = M^{K_в k_в} = M^{n \cdot \varphi(N)+1} \equiv M \ (\text{mod } N).$$

Таким образом, получатель B, который создает криптосистему, защищает два параметра: 1) секретный ключ $k_в$ и 2) пару чисел (P,Q), произведение которых дает значение модуля N. С другой стороны, получатель B открывает значение модуля N и открытый ключ $K_в$.

Противнику известны лишь значения $K_в$ и N. Если бы он смог разложить число N на множители P и Q, то он 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 не осуществимо (при условии, что длины выбранных P и Q составляют не менее 100 десятичных знаков).

4.3.1. Процедуры шифрования 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 получатель 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 $N = P * Q$.

3. Пользователь B вычисляет функцию Эйлера

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

и выбирает случайным образом значение открытого ключа $K_в$ с учетом выполнения условий:

$$1 < K_в \le \varphi(N), \quad НОД(K_в, \varphi(N)) = 1.$$

4. Пользователь B вычисляет значение секретного ключа $k_в$, используя расширенный алгоритм Евклида при решении сравнения

$$k_в \equiv K_в^{-1} \pmod{\varphi(N)}.$$

5. Пользователь B пересылает пользователю A пару чисел $(N, K_в)$ по незащищенному каналу.

Если пользователь a хочет передать пользователю в сообщение м, он выполняет следующие шаги.

6. Пользователь A разбивает исходный открытый текст M на блоки, каждый из которых может быть представлен в ви-де числа

$$M_i = 0, 1, 2, ..., N-1.$$

7. Пользователь A шифрует текст, представленный в виде последовательности чисел M_i по формуле

$$C_i = M_i^{K_в} \pmod{N}$$

и отправляет криптограмму

$$C_1, C_2, C_3, ..., C_i, ...$$

пользователю B.

8. Пользователь B расшифровывает принятую криптограмму

$$C_1, C_2, C_3, ..., C_i, ...,$$

используя секретный ключ $k_в$, по формуле

$$M_i = C_i^{k_в} \pmod{N}.$$

В результате будет получена последовательность чисел M_i, которые представляют собой исходное сообщение M. Чтобы алгоритм RSA имел практическую ценность, необходимо иметь возможность без существенных

затрат генерировать большие простые числа, уметь оперативно вычислять значения ключей $K_в$ и $k_в$.

4.3.2. Безопасность и быстродействие криптосистемы RSA

Безопасность алгоритма RSA базируется на трудности решения задачи факторизации больших чисел, являющихся произведениями двух больших простых чисел. Действительно, криптостойкость алгоритма RSA определяется тем, что после формирования секретного ключа $k_в$ и открытого ключа $K_в$ "стираются" значения простых чисел P и Q, и тогда исключительно трудно определить секретный ключ $k_в$ по открытому ключу $K_в$, поскольку для этого необходимо решить задачу нахождения делителей P и Q модуля N.

Разложение величины N на простые множители P и Q позволяет вычислить функцию φ (N) = (P –1)(Q –1) и затем определить секретное значение $k_в$, используя уравнение

$$K_в * k_в \equiv 1 \pmod{\varphi(N)}.$$

Другим возможным способом криптоанализа алгоритма RSA является непосредственное вычисление или подбор значения функции φ (N) = (P –1)(Q –1). Если установлено значение φ (n), то сомножители P и Q вычисляются достаточно просто. В самом деле, пусть

$$x = P + Q = N + 1 - \varphi(N),$$

$$y = (P - Q)^2 = (P + Q)^2 - 4*N.$$

Зная φ (N), можно определить x и затем y; зная x и y, можно определить числа P и Q из следующих соотношений:

$$P = 1/2\ (x + \sqrt{y}), \quad Q = 1/2\ (x - \sqrt{y}).$$

Однако эта атака не проще задачи факторизации модуля N [28].

Задача факторизации является трудно разрешимой задачей для больших значений модуля N.

Сначала авторы алгоритма RSA предлагали для вычисления модуля N выбирать простые числа P и Q случайным образом, по 50 десятичных разрядов каждое. Считалось, что такие большие числа N очень трудно разложить на простые множители. Один из авторов алгоритма RSA, P. Райвест, полагал, что разложение на простые множители числа из почти 130 десятичных цифр, приведенного в их публикации, потребует более 40 квадриллионов лет машинного времени. Однако этот прогноз не оправдался из-за сравнительно быстрого прогресса компьютеров и их вычислительной мощности, а также улучшения алгоритмов факторизации.

Ряд алгоритмов факторизации приведен в [45]. Один из наиболее быстрых алгоритмов, известных в настоящее время, алгоритм NFS (Number Field Sieve) может выполнить факторизацию большого числа N (с числом десятичных разрядов больше 120) за число шагов, оцениваемых величиной

$$e^{2(\ln n)^{1/3}(\ln(\ln n))^{2/3}}.$$

В 1994 г. было факторизовано число со 129 десятичными цифрами. Это удалось осуществить математикам А.Ленстра и М.Манасси посредством

организации распределенных вычислений на 1600 компьютерах, объединенных сетью, в течение восьми месяцев. По мнению А.Ленстра и М.Манасси, их работа компрометирует криптосистемы RSA и создает большую угрозу их дальнейшим применениям. Теперь разработчикам криптоалгоритмов с открытым ключом на базе RSA приходится избегать применения чисел длиной менее 200 десятичных разрядов. Самые последние публикации предлагают применять для этого числа длиной не менее 250 – 300 десятичных разрядов.

Была сделана попытка расчета оценок безопасных длин ключей асимметричных криптосистем на ближайшие 20 лет исходя из прогноза развития компьютеров и их вычислительной мощности, а также возможного совершенствования алгоритмов факторизации. Эти оценки (табл.4.1.) даны для трех групп пользователей (индивидуальных пользователей, корпораций и государственных организаций), в соответствии с различием требований к их информационной безопасности. Конечно, данные оценки следует рассматривать как сугубо приблизительные, как возможную тенденцию изменений безопасных длин ключей асимметричных криптосистем со временем.

Таблица 4.1

Оценки длин ключей для асимметричных криптосистем, бит

Год	Отдельные пользователи	Корпорации	Государственные организации
1995	768	1280	1536
2000	1024	1280	1536
2005	1280	1536	2048
2010	1280	1536	2048
2015	1536	2048	2048

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. И все же аппаратная реализация RSA примерно в 1000 раз медленнее аппаратной реализации симметричного криптоалгоритма DES.

Одна из самых быстрых аппаратных реализаций RSA с модулем 512 бит на сверхбольшой интегральной схеме имеет быстродействие 64 Кбит/с. Лучшими из серийно выпускаемых СБИС являются процессоры фирмы CYLINK, выполняющие 1024-битовое шифрование RSA.

Программная реализация RSA примерно в 100 раз медленнее программной реализации DES. С развитием технологии эти оценки могут несколько изменяться, но асимметричная криптосистема RSA никогда не достигнет быстродействия симметричных криптосистем.

Следует отметить, что малое быстродействие криптосистем RSA ограничивает область их применения, но не перечеркивает их ценность.

4.4. Схема шифрования Полига – Хеллмана

Схема шифрования Полига – Хеллмана [121] сходна со схемой шифрования RSA. Она представляет собой несимметричный алгоритм, поскольку используются различные ключи для шифрования и расшифрования. В то же время эту схему нельзя отнести к классу криптосистем с открытым ключом, так как ключи шифрования и расшифрования легко выводятся один из другого. Оба ключа (шифрования и расшифрования) нужно держать в секрете.

Аналогично схеме RSA криптограмма C и открытый текст P определяются из соотношений:

$$C = P^e \bmod n,$$

$$P = C^d \bmod n,$$

где $e*d \equiv 1$ (по модулю некоторого составного числа).

В отличие от алгоритма RSA в этой схеме число n не определяется через два больших простых числа; число n должно оставаться частью секретного ключа. Если кто-либо узнает значения e и n, он сможет вычислить значение d.

не зная значений e или d, противник будет вынужден вычислять значение

$$e = \log_P C \ (\text{mod } n).$$

Известно, что это является трудной задачей.

Схема шифрования Полига – Хеллмана запатентована в США и Канаде.

4.5. Схема шифрования Эль Гамаля

Схема Эль Гамаля, предложенная в 1985 г., может быть использована как для шифрования, так и для цифровых подписей. Безопасность схемы эль гамаля 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 должно храниться в секрете.

Далее вычисляют $Y = G^X \bmod P$. Число Y является открытым ключом.

Для того чтобы зашифровать сообщение M, выбирают случайное целое число k, $1< k< p-1$, такое, что числа к 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 шифртекстом. Заметим, что длина шифртекста вдвое больше длины исходного открытого текста M.

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

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

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 (*) справедливо.

В реальных схемах шифрования необходимо использовать в качестве модуля P большое целое простое число, имеющее в двоичном представлении длину 512…1024 бит.

При программной реализации схемы Эль Гамаля [123] скорость ее работы (на SPARC-II) в режимах шифрования и расшифрования при 160-битовом показателе степени для различных длин модуля P определяется значениями, приведенными в табл.4.2.

Таблица 4.2

Скорости работы схемы Эль Гамаля

Режим работы	Длина модуля, бит		
	512	768	1024
Шифрование	0,33 с	0,80 с	1,09 с
Расшифрование	0,24 с	0,58 с	0,77 с

4.6. 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. А 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 а хочет передать зашифрованное комбинированным методом сообщение м пользователю в, то порядок его действий будет таков.

1. Создать (например, сгенерировать случайным образом) **симметричный ключ**, называемый в этом методе **сеансовым ключом** $К_S$.
2. Зашифровать сообщение М на сеансовом ключе $К_S$.
3. Зашифровать сеансовый ключ $К_S$ на открытом ключе $К_B$ пользователя В.
4. Передать по открытому каналу связи в адрес пользователя В зашифрованное сообщение вместе с зашифрованным сеансовым ключом.

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

5. Расшифровать на своем секретном ключе k_B сеансовый ключ $К_S$.
6. С помощью полученного сеансового ключа $К_S$ расшифровать и прочитать сообщение М.

При использовании комбинированного метода шифрования можно быть уверенным в том, что только пользователь В сможет правильно расшифровать ключ $К_S$ и прочитать сообщение М.

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 табл. 4.3. приведены распространенные длины ключей симметричных и асимметричных криптосистем, для которых трудность атаки полного перебора примерно равна трудности факто-ризации соответствующих модулей асимметричных криптосистем [121].

Таблица 4.3

***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

Комбинированный метод допускает возможность выполнения процедуры аутентификации, т.е. проверки подлинности передаваемого сообщения. Для этого пользователь А на основе функции хэширования сообщения и своего секретного ключа k_A с помощью известного алгоритма электронной цифровой подписи (ЭЦП) генерирует свою подпись и записывает ее, например, в конец передаваемого файла.

Пользователь В, прочитав принятое сообщение, может убедиться в подлинности цифровой подписи абонента А. Используя тот же алгоритм ЭЦП и результат хэширования принятого сообщения, пользователь В проверяет полученную подпись (см.гл.6). Комбинированный метод шифрования является наиболее рациональным, объединяя в себе высокое быстродействие симметричного шифрования и высокую криптостойкость, гарантируемую системами с открытым ключом.

5. ИДЕНТИФИКАЦИЯ И ПРОВЕРКА ПОДЛИННОСТИ

5.1. Основные понятия и концепции

С каждым объектом компьютерной системы (КС) связана некоторая информация, однозначно идентифицирующая его. Это может быть *число, строка символов, алгоритм,* определяющий данный объект. Эту информацию называют *идентификатором объекта.* Если объект имеет некоторый идентификатор, зарегистрированный в сети, он называется законным (легальным) объектом; остальные объекты относятся к незаконным (нелегальным).

Идентификация объекта – одна из функций подсистемы защиты. Эта функция выполняется в первую очередь, когда объект делает попытку войти в сеть. Если процедура идентификации завершается успешно, данный объект считается законным для данной сети.

Следующий шаг – аутентификация объекта (проверка подлинности объекта). Эта процедура устанавливает, является ли данный объект именно таким, каким он себя объявляет.

После того как объект идентифицирован и подтверждена его подлинность, можно установить сферу его действия и доступные ему ресурсы КС. Такую процедуру называют *предоставлением полномочий (авторизацией)*.

Перечисленные три процедуры инициализации являются процедурами защиты и относятся к одному объекту КС [55].

При защите каналов передачи данных *подтверждение подлинности* (аутентификация) *объектов* означает взаимное установление подлинности объектов, связывающихся между собой по линиям связи. Процедура подтверждения подлинности выполняется обычно в начале сеанса в процессе установления соединения абонентов. (Термин "соединение" указывает на логическую связь (потенциально двустороннюю) между двумя объектами сети. Цель данной процедуры – обеспечить уверенность, что соединение установлено с законным объектом и вся информация дойдет до места назначения.

После того как соединение установлено, необходимо обеспечить выполнение требований защиты при обмене сообщениями:

(а) получатель должен быть уверен в подлинности источника данных;

(б) получатель должен быть уверен в подлинности передаваемых данных;

(в) отправитель должен быть уверен в доставке данных получателю;

(г) отправитель должен быть уверен в подлинности доставленных данных.

Для выполнения требований (а) и (б) средством защиты является *цифровая подпись*. Для выполнения требований (в) и (г) отправитель должен получить *уведомление о вручении* с помощью удостоверяющей почты (certified mail). Средством защиты в такой процедуре является цифровая подпись подтверждающего ответного сообщения, которое в свою очередь является доказательством пересылки исходного сообщения.

Если эти четыре требования реализованы в КС, то гарантируется защита данных при их передаче по каналу связи и обеспечивается функция защиты, называемая функцией подтверждения (неоспоримости) передачи. В этом случае отправитель не может отрицать ни факта посылки сообщения, ни его содержания, а получатель не может отрицать ни факта получения сообщения, ни подлинности его содержания.

5.2. Идентификация и аутентификация пользователя

Прежде чем получить доступ к ресурсам компьютерной системы, пользователь должен пройти процесс представления компьютерной системе, который включает две стадии:

- идентификацию - пользователь сообщает системе по ее запросу свое имя (идентификатор);
- аутентификацию - пользователь подтверждает идентификацию вводя в систему уникальную, не известную другим пользователям информацию о себе (например, пароль).

Для проведения процедур идентификации и аутентификации пользователя необходимы:

- во-первых, наличие соответствующего *субъекта (модуля) аутентификации*;
- во-вторых, наличие *аутентифицирующего объекта*, хранящего уникальную информацию для аутентификации пользователя.

Различают две формы представления объектов, аутентифицирующих пользователя:

1) внешний аутентифицирующий объект, не принадлежащий системе;
2) внутренний объект, принадлежащий системе, в который переносится информация из внешнего объекта.

Внешние объекты могут быть технически реализованы на различных носителях информации - магнитных дисках, пластиковых картах и т.п. Естественно, что внешняя и внутренняя формы представления аутентифицирующего объекта должны быть семантически тождественны.

5.2.1 Типовые схемы идентификации и аутентификации пользователя.

Рассмотрим структуры данных и протоколы идентификации и аутентификации пользователя [73]. Допустим, что в компьютерной системе зарегистрировано n пользователей. Пусть i-й аутентифицирующий объект i-го пользователя содержит два информационных поля:

ID_i - неизменный идентификатор i-го пользователя, который является аналогом имени и используется для идентификации пользователя;

K_i - аутентифицирующая информация пользователя, которая может изменяться и служит для аутентификации (например, пароль $P_i = K_i$).

Описанная структура соответствует практически любому ключевому носителю информации, используемому для опознания пользователя. Например, для носителей типа пластиковых карт выделяется неизменяемая

информация ID_i первичной персонализации пользователя и объект в файловой структуре карты, содержащий K_i.

Совокупную информацию в ключевом носителе можно назвать первичной аутентифицирующей информацией i-го пользователя. Очевидно, что внутренний аутентифицирующий объект не должен существовать в системе длительное время (больше времени работы конкретного пользователя). Для длительного хранения следует использовать данные в защищенной форме.

Рассмотрим две типовые схемы идентификации и аутентификации.

Схема 1. В компьютерной системе выделяется объект-эталон для идентификации и аутентификации пользователей. Структура объекта-эталона для схемы 1 показана в таблице 5.1.

Таблица 5.1.

Номер пользователя	Информация для идентификации	Информация для аутентификации
1	ID_1	E_1
2	ID_2	E_2
...	...	...
N	ID_n	E_n

Здесь $E_i = F(ID_i, K_i)$,

где F - функция, которая обладает свойством “невосстановимости” значения K_i по E_i и ID_i. “Невосстановимость” K_i оценивается некоторой пороговой трудоемкостью T_o решения задачи восстановления аутентифицирующей информации K_i по E_i и ID_i. Кроме того, для пары K_i и K_j возможно совпадение соответствующих значений E. В связи с этим вероятность ложной аутентификации пользователя не должна быть больше некоторого порогового значения P_o. На практике задают $T_o = 10^{20} ... 10^{30}$, $P_o = 10^{-7} ... 10^{-9}$ [73].

Протокол идентификации и аутентификации (для схемы 1).

1. Пользователь предъявляет свой идентификатор ID.
2. Если ID не совпадает ни с одним ID_i , зарегистрированным в компьютерной системе, то идентификация отвергается - пользователь не допускается к работе, иначе (существует ID_i = ID) устанавливается, что пользователь, назвавшийся пользователем i, прошел идентификацию.
3. Субъект аутентификации запрашивает у пользователя его аутентификатор K.
4. Субъект аутентификации вычисляет значение

$$Y = F(ID_i, K).$$

5. Субъект аутентификации производит сравнение значений Y и E_i. При совпадении этих значений устанавливается, что данный пользователь успешно аутентифицирован в системе. Информация об этом пользователе передается в программные модули, использующие ключи пользователей (т.е. в систему шифрования, разграничения доступа и т. д.). В противном случае аутентификация отвергается - пользователь не допускается к работе.

Данная схема идентификации и аутентификации пользователя может быть модифицирована. Модифицированная схема 2 обладает лучшими характеристиками по сравнению со схемой 1.

Схема 2. В компьютерной системе выделяется модифицированный объект-эталон, структура которого показана в таблице 5.2.

Таблица 5.2.

Номер пользователя	Информация для идентификации	Информация для аутентификации
1	ID_1, S_1	E_1
2	ID_2, S_2.	E_2
...	...	...
N	ID_n, S_n	E_n

В отличие от схемы 1, в схеме 2 значение $E_i = F(S_i, K_i)$,
где S_i - **случайный вектор, задаваемый при создании идентификатора пользователя**, т.е. при создании строки, необходимой для идентификации и аутентификации пользователя;
F - функция, которая обладает свойством “невосстановимости” значения K_i по E_i и S_i.

Протокол идентификации и аутентификации (для схемы 2).

1. Пользователь предъявляет свой идентификатор ID.
2. Если ID не совпадает ни с одним ID_i, зарегистрированным в компьютерной системе, то идентификация отвергается - пользователь не допускается к работе, иначе (существует $ID_i = ID$) устанавливается, что пользователь, называвшийся пользователем i, прошел идентификацию.
3. По идентификатору ID_i выделяется вектор S_i.
4. Субъект аутентификации запрашивает у пользователя аутентификатор К.
5. Субъект аутентификации вычисляет значение
$$Y = F(S_i, К).$$
6. Субъект аутентификации производит сравнение значений Y и E_i. При совпадении этих значений устанавливается, что данный пользователь успешно аутентифицирован в системе. В противном случае аутентификация отвергается - пользователь не допускается к работе.

Вторая схема аутентификации применяется в ОС UNIX. В качестве идентификатора ID используется имя пользователя (запрошенное по Loqin), в качестве аутентификатора K_i - пароль пользователя (запрошенный по Password), функция F представляет собой алгоритм шифрования DES. Эталоны для идентификации и аутентификации содержатся в файле Etc/passwd.

Следует отметить, что необходимым требованием устойчивости схем аутентификации к восстановлению информации $К_i$ является случайный равновероятный выбор $К_i$ из множества возможных значений.

Системы парольной аутентификации имеют пониженную стойкость, поскольку в них выбор аутентифицирующей информации происходит из относительно небольшого множества осмысленных слов. Мощность этого множества определяется энтропией соответствующего языка.

Особенности применения пароля для аутентификации пользователя.

Традиционно каждый законный пользователь компьютерной системы получает идентификатор и/или пароль. В начале сеанса работы пользователь предъявляет свой идентификатор системе, которая затем запрашивает у пользователя пароль.

Простейший метод подтверждения подлинности с использованием пароля основан на сравнении представляемого пользователем пароля P_A с исходным значением P_A' , хранящимся в компьютерном центре (рис. 5.1). Поскольку пароль должен храниться в тайне, он должен шифроваться перед пересылкой по незащищенному каналу. Если значения P_A и P_A' совпадают, то пароль P_A считается подлинным, а пользователь - законным [123].

Если кто-нибудь, не имеющий полномочий для входа в систему, узнает каким-либо образом пароль и идентификационный номер законного пользователя, он получает доступ в систему.

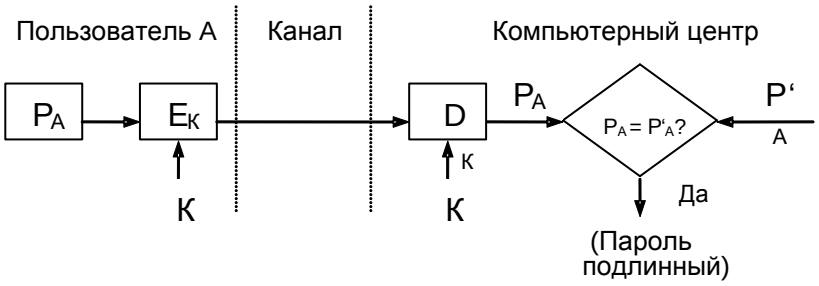


Рисунок 5.1 – Схема простой аутентификации с помощью пароля

Иногда получатель не должен раскрывать исходную открытую форму пароля. В этом случае отправитель должен пересылать вместо открытой формы пароля отображение пароля, получаемое с использованием односторонней функции $\alpha(\cdot)$ пароля. Это преобразование должно гарантировать невозможность раскрытия противником пароля по его отображению, так как противник наталкивается на неразрешимую числовую задачу.

Например, функция $\alpha(\cdot)$ может быть определена следующим образом:

$$\alpha(P) = E_P(ID),$$

где P - пароль отправителя,

ID - идентификатор отправителя,

E_P - процедура шифрования, выполняемая с использованием пароля P в качестве ключа.

Такие функции особенно удобны, если длина пароля и ключа одинаковы. В этом случае подтверждение подлинности с помощью пароля состоит из пересылки получателю отображения $\alpha(P)$ и сравнения его с предварительно вычисленным и хранимым эквивалентом $\alpha'(P)$.

На практике пароли состоят только из нескольких букв, чтобы дать возможность пользователям запомнить их. Короткие пароли уязвимы к атаке

полного перебора всех вариантов. Для того, чтобы предотвратить такую атаку, функцию $\alpha(P)$ определяют иначе, а именно:

$$\alpha(P) = E_{P \oplus K}(ID),$$

где K и ID - соответственно ключ и идентификатор отправителя.

Очевидно, значение $\alpha(P)$ вычисляется заранее и хранится в виде $\alpha'(P)$ в идентификационной таблице у получателя (рис. 5.2). Подтверждение подлинности состоит из сравнения двух отображений пароля $\alpha(P_A)$ и $\alpha'(P_A)$ и признания пароля P_A, если эти отображения равны. Конечно, любой, кто получит доступ к идентификационной таблице, может незаконно изменить ее содержимое, не опасаясь, что эти действия будут обнаружены.

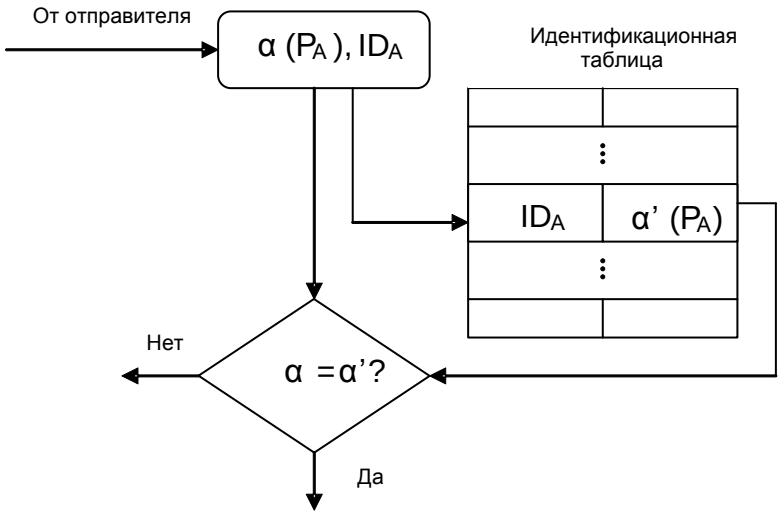


Рисунок 5.2 – Аутентификация с помощью пароля с использованием идентификационной таблицы

Применение для целей идентификации и аутентификации персонального идентификационного номера PIN рассматривается в главе 9.

5.2.2. Биометрическая идентификация и аутентификация пользователя

Процедуры идентификации и аутентификации пользователя могут базироваться не только на секретной информации которой обладает пользователь (пароль, секретный ключ, персональный идентификатор и т.п.).

В последнее время все большее распространение получает биометрическая идентификация и аутентификация пользователя, позволяющая уверенно идентифицировать потенциального пользователя путем измерения физиологических параметров и характеристик человека, особенностей его поведения.

Отметим основные достоинства биометрических методов идентификации и аутентификации пользователя по сравнению с традиционными [73]:

- высокая степень достоверности идентификации по биометрическим признакам из-за их уникальности;
- неотделимость биометрических признаков от дееспособной личности;
- трудность фальсификации биометрических признаков.

В качестве биометрических признаков, которые могут быть использованы при идентификации потенциального пользователя, можно выделить следующие:

- узор радужной оболочки и сетчатки глаз;
- отпечатки пальцев;
- геометрическая форма руки;
- форма и размеры лица;
- особенности голоса;
- биомеханические характеристики рукописной подписи;
- биомеханические характеристики “клавиатурного почерка”.

При регистрации пользователь должен продемонстрировать один или несколько раз свои характерные биометрические признаки. Эти признаки (известные как подлинные) регистрируются системой как контрольный “образ” законного пользователя. Этот образ пользователя хранится в электронной форме и используется для проверки идентичности каждого, кто выдает себя за соответствующего законного пользователя. В зависимости от совпадения или несовпадения совокупности предъявленных признаков с зарегистрированными в контрольном образе их предъявивший признается законным пользователем (при совпадении) или нет (при несовпадении).

Системы идентификации по узору радужной оболочки и сетчатки глаз могут быть разделены на два класса:

- использующие рисунок радужной оболочки глаза;
- использующие рисунок кровеносных сосудов сетчатки глаза.

Поскольку вероятность повторения данных параметров равна 10^{-78}, такие системы являются наиболее надежными среди всех биометрических систем. Такие средства идентификации применяются там, где требуется высокий уровень безопасности (например, в США в зонах военных и оборонных объектов).

Системы идентификации по отпечаткам пальцев являются самыми распространенными. Одной из основных причин широкого распространения таких систем является наличие больших банков данных по отпечаткам пальцев. Основными пользователями подобных систем во всем мире являются полиция, различные государственные и некоторые банковские организации.

Системы идентификации по геометрической форме руки используют сканеры формы руки, обычно устанавливаемые на стенах. Следует отметить, что подавляющее большинство пользователей предпочитают системы именно этого типа, а не описанные выше.

Системы идентификации по лицу и голосу являются наиболее доступными из-за их дешевизны, поскольку большинство современных компьютеров имеют видео- и аудиосредства. Системы данного класса широко применяются при удаленной идентификации субъекта доступа в телекоммуникационных сетях.

Системы идентификации личностей по динамике рукописной подписи учитывают интенсивность каждого усилия подписывающего, частотные характеристики написания каждого элемента подписи и начертание подписи в целом.

Системы идентификации по биомеханическим характеристикам "клавиатурного почерка" основываются на том, что моменты нажатия и отпускания клавиш при наборе текста на клавиатуре существенно отличаются у различных пользователей. Этот динамический ритм набора ("клавиатурный почерк") позволяет построить достаточно надежные средства идентификации. В случае обнаружения изменения клавиатурного почерка пользователя ему автоматически запрещается работа на ЭВМ.

Следует отметить, что применение биометрических параметров при идентификации субъектов доступа автоматизированных систем пока не получило надлежащего нормативно-правового обеспечения, в частности, в виде стандартов. Поэтому применение систем биометрической идентификации допускается только в автоматизированных системах, обрабатывающих и хранящих персональные данные, составляющие коммерческую и служебную тайну [73].

5.3. Взаимная проверка подлинности пользователей

Обычно стороны, вступающие в информационный обмен, нуждаются во взаимной проверке подлинности (аутентификации) друг друга. Этот **процесс взаимной аутентификации выполняют в начале сеанса связи**.

Для проверки подлинности применяют следующие способы:

- механизм запроса-ответа;
- механизм отметки времени ("временной штемпель").

Механизм запроса-ответа состоит в следующем. Если пользователь A хочет быть уверенным, что сообщения, получаемые им от пользователя B, не являются ложными, он включает в посылаемое для B сообщение непредсказуемый элемент – запрос X (например, некоторое случайное число). При ответе пользователь B должен выполнить над этим элементом некоторую операцию (например, вычислить некоторую функцию f (X)). Это невозможно осуществить заранее, так как пользователю B неизвестно, какое случайное число X придет в запросе. Получив ответ с результатом действий B, пользователь A может быть уверен, что B – подлинный. Недостаток этого метода – возможность установления закономерности между запросом и ответом.

Механизм отметки времени подразумевает регистрацию времени для каждого сообщения. В этом случае каждый пользователь сети может определить, насколько "устарело" пришедшее сообщение, и решить не принимать его, поскольку оно может быть ложным.

В обоих случаях для защиты механизма контроля следует применять шифрование, чтобы быть уверенным, что ответ послан не злоумышленником.

При использовании отметок времени возникает проблема *допустимого временного интервала задержки* для подтверждения подлинности сеанса. Ведь сообщение с "временным штемпелем" в принципе не может быть передано мгновенно. Кроме того, компьютерные часы получателя и отправителя не могут быть абсолютно синхронизированы. Какое запаздывание "штемпеля" является подозрительным?

Для взаимной проверки подлинности обычно используют *процедуру "рукопожатия"* [55, 123]. Эта процедура базируется на указанных выше механизмах контроля и заключается во взаимной проверке ключей, используемых сторонами. Иначе говоря, стороны признают друг друга законными партнерами, если докажут друг другу, что обладают правильными ключами. Процедуру рукопожатия обычно применяют в компьютерных сетях при организации сеанса связи между пользователями, пользователем и хост-компьютером, между хост-компьютерами и т.д.

Рассмотрим в качестве примера процедуру рукопожатия для двух пользователей A и B. (Это допущение не влияет на общность рассмотрения. Такая же процедура используется, когда вступающие в связь стороны не являются пользователями). Пусть применяется симметричная криптосистема. Пользователи A и B разделяют один и тот же секретный ключ K_{AB}. Вся процедура показана на рис. 5.3.

- Пусть пользователь A инициирует процедуру рукопожатия, отправляя пользователю B свой идентификатор ID_A в открытой форме.
- Пользователь B, получив идентификатор ID_A, находит в базе данных секретный *ключ* K_{AB} и вводит его в свою криптосистему.
- Тем временем *пользователь A* генерирует случайную *последовательность S* с помощью псевдослучайного генератора PG и отправляет ее *пользователю B* в виде криптограммы
$$E_{K_{AB}}(S).$$
- Пользователь B расшифровывает эту криптограмму и раскрывает исходный вид последовательности S.
- Затем оба пользователя A и B преобразуют последователь-ность S, используя открытую одностороннюю функцию $\alpha(\cdot)$.
- Пользователь B шифрует сообщение $\alpha(S)$ и отправляет эту криптограмму *пользователю A*.
- Наконец, пользователь A расшифровывает эту криптограмму и сравнивает полученное сообщение $\alpha'(S)$ с исходным $\alpha(S)$. Если эти сообщения равны, пользователь A признает подлинность пользователя B.

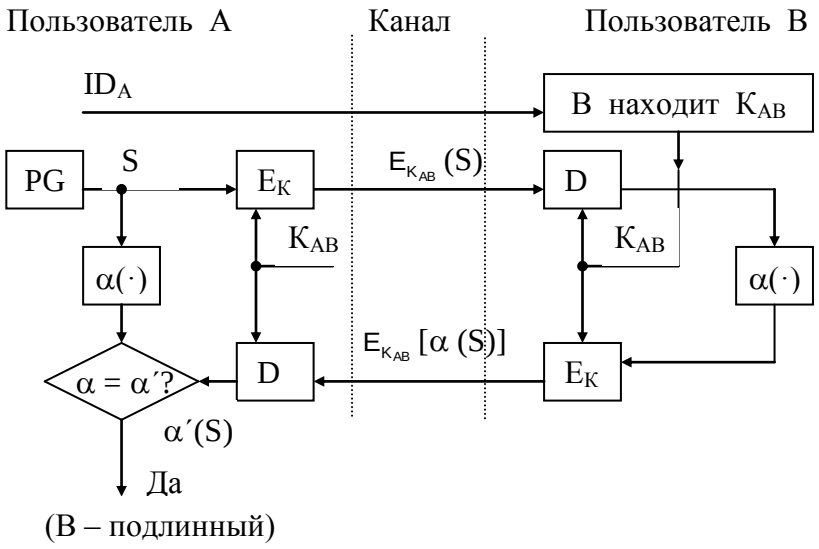


Рисунок 5.3 – Схема процедуры рукопожатия (пользователь A проверяет подлинность пользователя B)

Очевидно, пользователь B проверяет подлинность пользователя A таким же способом. Обе эти процедуры образуют процедуру рукопожатия, которая обычно выполняется в самом начале любого сеанса связи между любыми двумя сторонами в компьютерных сетях.

Достоинством модели рукопожатия является то, что ни один из участников сеанса связи не получает никакой секретной информации во время процедуры подтверждения подлинности.

Иногда пользователи хотят иметь непрерывную проверку подлинности отправителей в течение всего сеанса связи. Один из простейших способов непрерывной проверки подлинности показан на рис. 5.4 [123]. Передаваемая криптограмма имеет вид

$$E_K (ID_A, M),$$

где ID_A – идентификатор отправителя A; M – сообщение.

Получатель B, принявший эту криптограмму, расшифровывает ее и раскрывает пару (ID_A, M). Если принятый идентификатор ID_A совпадает с хранимым значением ID_A', получатель B признает эту криптограмму.

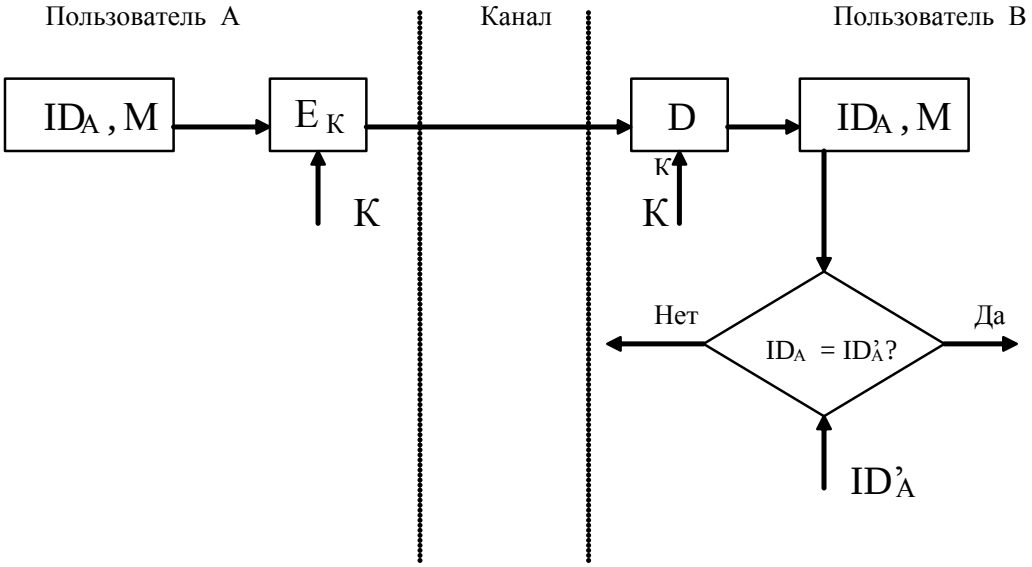


Рисунок 5.4 – Схема непрерывной проверки подлинности отправителя

Другой вариант непрерывной проверки подлинности использует вместо идентификатора отправителя его секретный пароль. Заранее подготовленные пароли известны обеим сторонам. Пусть P_A и P_B – пароли пользователей A и B соответственно. Тогда пользователь A создает криптограмму

$$C = E_K (P_A, M).$$

Получатель криптограммы расшифровывает ее и сравнивает пароль, извлеченный из этой криптограммы, с исходным значением. Если они равны, получатель признает эту криптограмму.

Процедура рукопожатия была рассмотрена в предположении, что пользователи A и B уже имеют общий *секретный сеансовый ключ*. Реальные процедуры предназначены для распределения ключей между подлинными партнерами и включает как этап распределения ключей, так и этап собственно подтверждения подлинности партнеров по информационному обмену.

5.4. Протоколы идентификации с нулевой передачей знаний

Широкое распространение интеллектуальных карт (смарт-карт) для разнообразных коммерческих, гражданских и военных применений (кредитные карты, карты социального страхования, карты доступа в охраняемое помещение, компьютерные пароли и ключи, и т.п.) потребовало обеспечения безопасной идентификации таких карт и их владельцев. Во многих приложениях главная проблема заключается в том, чтобы при предъявлении интеллектуальной карты оперативно обнаружить обман и отказать обманщику в допуске, ответе или обслуживании.

Для безопасного использования интеллектуальных карт разработаны протоколы идентификации с нулевой передачей знаний [121]. Секретный ключ владельца карты становится неотъемлемым признаком его личности. Доказательство знания этого секретного ключа с нулевой передачей этого знания служит доказательством подлинности личности владельца карты.

5.4.1. Упрощенная схема идентификации с нулевой передачей знаний

Схему идентификации с нулевой передачей знаний предложили в 1986 г. У.Фейге, А.Фиат и А.Шамир. Она является наиболее известным доказательством идентичности с нулевой передачей конфиденциальной информации.

Рассмотрим сначала упрощенный вариант схемы идентификации с нулевой передачей знаний для более четкого выявления ее основной концепции. Прежде всего выбирают случайное значение модуля n, который является произведением двух больших простых чисел. Модуль n должен иметь длину 512…1024 бит. Это значение n может быть представлено

группе пользователей, которым придется доказывать свою подлинность. В процессе идентификации участвуют две стороны:

- сторона A, доказывающая свою подлинность,
- сторона B, проверяющая представляемое стороной A доказательство.

Для того чтобы сгенерировать открытый и секретный ключи для стороны A, доверенный арбитр (Центр) выбирает некоторое число V, которое является квадратичным вычетом по модулю n. Иначе говоря, выбирается такое число V, что сравнение

$$x^2 \equiv V \pmod{n}$$

имеет решение и существует целое число

$$V^{-1} \bmod n.$$

Выбранное значение V является *открытым ключом* для A. Затем вычисляют наименьшее значение S, для которого

$$S \equiv \text{sqrt}\,(V^{-1}) \pmod{n}.$$

Это значение S является *секретным ключом* для A.

Теперь можно приступить к выполнению протокола идентификации.

1. Сторона A выбирает некоторое случайное число r, r < n. Затем она вычисляет

$$x = r^2 \bmod n$$

и отправляет x стороне B.

2. Сторона B посылает A случайный бит b.

3. Если b=0, тогда A отправляет r стороне B. Если b=1, то A отправляет стороне B

$$y = r * S \bmod n.$$

4. Если b = 0, сторона B проверяет, что

$$x = r^2 \bmod n,$$

чтобы убедиться, что A знает sqrt (x). Если b=1, сторона B проверяет, что

$$x = y^2 * V \bmod n,$$

чтобы быть уверенной, что A знает $\text{sqrt}\,(V^{-1})$.

Эти шаги образуют один цикл протокола, называемый *аккредитацией*. Стороны A и B повторяют этот цикл t раз при разных случайных значениях r и b до тех пор, пока B не убедится, что A знает значение S.

Если сторона A не знает значения S, она может выбрать такое значение r, которое позволит ей обмануть сторону B, если B отправит ей b=0, либо A может выбрать такое r, которое позволит обмануть B, если B отправит ей b=1. Но этого невозможно сделать в обоих случаях. Вероятность того, что A обманет B в одном цикле, составляет 1/2. Вероятность обмануть B в t циклах равна $(1/2)^t$.

Для того чтобы этот протокол работал, сторона A никогда не должна повторно использовать значение r. Если A поступила бы таким образом, а сторона B отправила бы стороне A на шаге 2 другой случайный бит b, то

B имела бы оба ответа A. После этого B может вычислить значение S, и для A все закончено.

Параллельная схема идентификации с нулевой передачей знаний

Параллельная схема идентификации позволяет увеличить число аккредитаций, выполняемых за один цикл, и тем самым уменьшить длительность процесса идентификации.

Как и в предыдущем случае, сначала генерируется число n как произведение двух больших чисел. Для того, чтобы сгенерировать открытый и секретный ключи для стороны A, сначала выбирают K различных чисел V_1, V_2, ..., V_K, где каждое V_i является квадратичным вычетом по модулю n. Иначе говоря, выбирают значение V_i таким, что сравнение

$$x^2 \equiv V_i \bmod n$$

имеет решение и существует $V_i^{-1} \bmod n$. Полученная строка $V_1, V_2, ..., V_K$ является *открытым ключом*.

Затем вычисляют такие наименьшие значения S_i, что

$$S_i = \text{sqrt}\,(V_i^{-1}) \bmod n.$$

Эта строка $S_1, S_2, ..., S_K$ является *секретным ключом* стороны A.

Протокол процесса идентификации имеет следую-щий вид:

1. Сторона A выбирает некоторое случайное число r, r<n. Затем она вычисляет $x=r^2 \bmod n$ и посылает x стороне B.

2. Сторона B отправляет стороне A некоторую случайную двоичную строку из K бит: $b_1, b_2, ..., b_K$.

3. Сторона A вычисляет

$$y = r * (S_1^{b1} * S_2^{b2} * ... * S_K^{bK}) \bmod n.$$

Перемножаются только те значения S_i, для которых $b_i=1$. Например, если $b_1=1$, то сомножитель S_1 входит в произведение, если же $b_1=0$, то S_1 не входит в произведение, и т.д. Вычисленное значение y отправляется стороне B.

4. Сторона B проверяет, что

$$x = y^2 * (V_1^{b1} * V_2^{b2} * ... * V_K^{bK}) \bmod n.$$

Фактически сторона B перемножает только те значения V_i, для которых $b_i=1$. Стороны A и B повторяют этот протокол t раз, пока B не убедится, что A знает $S_1, S_2, ..., S_K$.

Вероятность того, что A может обмануть B, равна $(1/2)^{Kt}$. Авторы рекомендуют в качестве контрольного значения брать вероятность обмана B равной $(1/2)^{20}$ при K=5 и t=4.

СХЕМА ИДЕНТИФИКАЦИИ ГИЛЛОУ – КУИСКУОТЕРА

Алгоритм идентификации с нулевой передачей знания, разработанный л.гиллоу и Ж.Куискуотером, имеет несколько лучшие характеристики, чем

предыдущая схема идентификации. В этом алгоритме обмены между сторонами а и в и аккредитации в каждом обмене доведены до абсолютного минимума – для каждого доказательства требуется только один обмен с одной аккредитацией. Однако объем требуемых вычислений для этого алгоритма больше, чем для схемы Фейге–Фиата–Шамира.

Пусть сторона A – интеллектуальная карточка, которая должна доказать свою подлинность проверяющей стороне B. Идентификационная информация стороны A представляет собой битовую строку I, которая включает имя владельца карточки, срок действия, номер банковского счета и др. Фактически идентификационные данные могут занимать достаточно длинную строку, и тогда их хэшируют к значению I.

Строка I является аналогом открытого ключа. Другой открытой информацией, которую используют все карты, участвующие в данном приложении, являются модуль n и показатель степени V. Модуль n является произведением двух секретных простых чисел.

Секретным ключом стороны A является величина G, выбираемая таким образом, чтобы выполнялось соотношение

$$I * G^V \equiv 1 \pmod{n}.$$

Сторона A отправляет стороне B свои идентификационные данные I. Далее ей нужно доказать стороне B, что эти идентификационные данные принадлежат именно ей. Чтобы добиться этого, сторона A должна убедить сторону B, что ей известно значение G.

Вот протокол доказательства подлинности A без передачи стороне B значения G:

1. Сторона A выбирает случайное целое r, такое, что $1 < r \le n - 1$. Она вычисляет

$$T = r^V \bmod n$$

и отправляет это значение стороне B.

2. Сторона B выбирает случайное целое d, такое, что $1 < d \le n - 1$, и отправляет это значение d стороне A.

3. Сторона A вычисляет

$$D = r * G^d \bmod n$$

и отправляет это значение стороне B.

4. Сторона B вычисляет значение

$$T' = D^V I^d \bmod n.$$

Если $T \equiv T' \pmod{n}$,

то проверка подлинности успешно завершена.

Математические выкладки, использованные в этом протоколе, не очень сложны:

$$T' = D^V I^d = (r G^d)^V I^d = r^V G^{dV} I^d = r^V (I G^V)^d = r^V \equiv T \pmod{n},$$

поскольку G вычислялось таким образом, чтобы выполнялось соотношение

$$IG^V \equiv 1 \pmod{n}.$$

6. ЭЛЕКТРОННАЯ ЦИФРОВАЯ ПОДПИСЬ

6.1. Проблема аутентификации данных и электронная цифровая подпись

При обмене электронными документами по сети связи существенно снижаются затраты на обработку и хранение документов, убыстряется их поиск. Но при этом возникает проблема аутентификации автора документа и самого документа, т.е. установления подлинности автора и отсутствия изменений в полученном документе. В обычной (бумажной) информатике эти проблемы решаются за счет того, что информация в документе и рукописная подпись автора жестко связаны с физическим носителем (бумагой). В электронных документах на машинных носителях такой связи нет.

Целью аутентификации электронных документов является их защита от возможных видов злоумышленных действий, к которым относятся:

- активный перехват – нарушитель, подключившийся к сети, перехватывает документы (файлы) и изменяет их;
- маскарад – абонент C посылает документ абоненту B от имени абонента A;
- ренегатство – абонент A заявляет, что не посылал сообщения абоненту B, хотя на самом деле послал;
- подмена – абонент B изменяет или формирует новый документ и заявляет, что получил его от абонента A;
- повтор – абонент C повторяет ранее переданный документ, который абонент A посылал абоненту B.

Эти виды злоумышленных действий могут нанести сущест-венный ущерб банковским и коммерческим структурам, государственным предприятиям и организациям, частным лицам, применяющим в своей деятельности компьютерные информационные технологии.

При обработке документов в электронной форме совершенно непригодны традиционные способы установления подлинности по рукописной подписи и оттиску печати на бумажном документе. Принципиально новым решением является электронная цифровая подпись (ЭЦП).

Электронная цифровая подпись используется для аутенти-фикации текстов, передаваемых по телекоммуникационным каналам. Функционально она аналогична обычной рукописной подписи и обладает ее основными достоинствами:

- удостоверяет, что подписанный текст исходит от лица, поставившего подпись;
- не дает самому этому лицу возможности отказаться от обязательств, связанных с подписанным текстом;
- гарантирует целостность подписанного текста.

Цифровая подпись представляет собой относительно небольшое количество дополнительной цифровой информации, передаваемой вместе с подписываемым текстом.

Система ЭЦП включает две процедуры: 1) процедуру постановки подписи; 2) процедуру проверки подписи. В процедуре постановки подписи ис-

пользуется секретный ключ отправителя сообщения, в процедуре проверки подписи – открытый ключ отправителя.

При формировании ЭЦП отправитель прежде всего вычисляет хэш-функцию h(M) подписываемого текста M. Вычисленное значение хэш-функции h(M) представляет собой один короткий блок информации m, характеризующий весь текст M в целом. Затем число m шифруется секретным ключом отправителя. Получаемая при этом пара чисел представляет собой ЭЦП для данного текста M.

При проверке ЭЦП получатель сообщения снова вычисляет хэш-функцию m = h(M) принятого по каналу текста M, после чего при помощи открытого ключа отправителя проверяет, соответствует ли полученная подпись вычисленному значению m хэш-функции.

принципиальным моментом в системе эцп является невозможность подделки эцп пользователя без знания его секретного ключа подписывания.

В качестве подписываемого документа может быть использован любой файл. Подписанный файл создается из неподписанного путем добавления в него одной или более электронных подписей.

Каждая подпись содержит следующую информацию:

- дату подписи;
- срок окончания действия ключа данной подписи;
- информацию о лице, подписавшем файл (Ф.И.О., должность, краткое наименование фирмы);
- идентификатор подписавшего (имя открытого ключа);
- собственно цифровую подпись.

6.2. Однонаправленные хэш-функции

Хэш-функция предназначена для сжатия подписываемого документа M до нескольких десятков или сотен бит. Хэш-функция h(·) принимает в качестве аргумента сообщение (документ) M произвольной длины и возвращает хэш-значение h(M)=H фиксированной длины. Обычно хэшированная информация является сжатым двоичным представлением основного сообщения произвольной длины. Следует отметить, что значение хэш-функции h(M) сложным образом зависит от документа M и не позволяет восстановить сам документ M.

Хэш-функция должна удовлетворять целому ряду условий:

- хэш-функция должна быть чувствительна к всевозможным изменениям в тексте M, таким как вставки, выбросы, перестановки и т.п.;
- хэш-функция должна обладать свойством необратимости, то есть задача подбора документа M', который обладал бы требуемым значением хэш-функции, должна быть вычислительно неразрешима;
- вероятность того, что значения хэш-функций двух различных документов (вне зависимости от их длин) совпадут, должна быть ничтожно мала [123].

Большинство хэш-функций строится на основе однонаправленной функции f(·), которая образует выходное значение длиной n при задании двух входных значений длиной n. Этими входами являются блок исходного текста M_i и хэш-значение H_{i-1} предыдущего блока текста (рис.6.1):

$$H_i = f(M_i, H_{i-1}).$$

Хэш-значение, вычисляемое при вводе последнего блока текста, становится хэш-значением всего сообщения M.

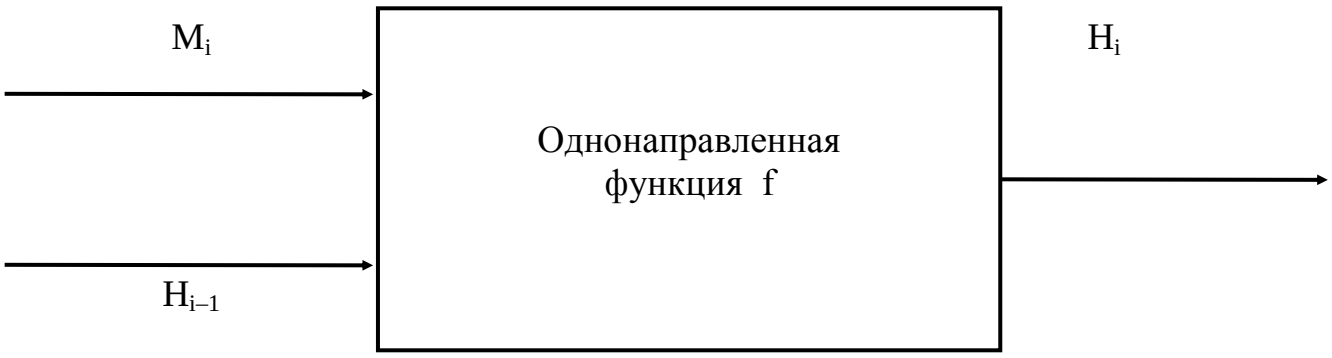


Рисунок 6.1 – Построение однонаправленной хэш-функции

В результате однонаправленная хэш-функция всегда формирует выход фиксированной длины n (независимо от длины входного текста).

6.2.1. Однонаправленные хэш-функции на основе симметричных блочных алгоритмов

Однонаправленную хэш-функцию можно построить, используя симметричный блочный алгоритм. Наиболее очевидный подход состоит в том, чтобы шифровать сообщение m посредством блочного алгоритма в режиме CBC или CFB с помощью фиксированного ключа и некоторого вектора инициализации IV. Последний блок шифртекста можно рассматривать в качестве хэш-значения сообщения M. При таком подходе не всегда возможно построить безопасную однонаправленную хэш-функцию, но всегда можно получить код аутентификации сообщения MAC (Message Authentication Code).

Более безопасный вариант хэш-функции можно получить, используя блок сообщения в качестве ключа, предыдущее хэш-значение – в качестве входа, а текущее хэш-значение – в качестве выхода. Реальные хэш-функции проектируются еще более сложными. Длина блока обычно определяется длиной ключа, а длина хэш-значения совпадает с длиной блока.

Поскольку большинство блочных алгоритмов являются 64-битовыми, некоторые схемы хэширования проектируют так, чтобы хэш-значение имело длину, равную двойной длине блока.

Если принять, что получаемая хэш-функция корректна, безопасность схемы хэширования базируется на безопасности лежащего в ее основе блочного алгоритма. Схема хэширования, у которой длина хэш-значения равна длине блока, показана на рис. 6.2. Ее работа описывается выражениями:

$$H_0 = I_H,$$
$$H_i = E_A(B) \oplus C,$$

где I_H – некоторое случайное начальное значение; A, B и C могут принимать значения M_i, H_{i-1}, $(M_i \oplus H_{i-1})$ или быть константами.

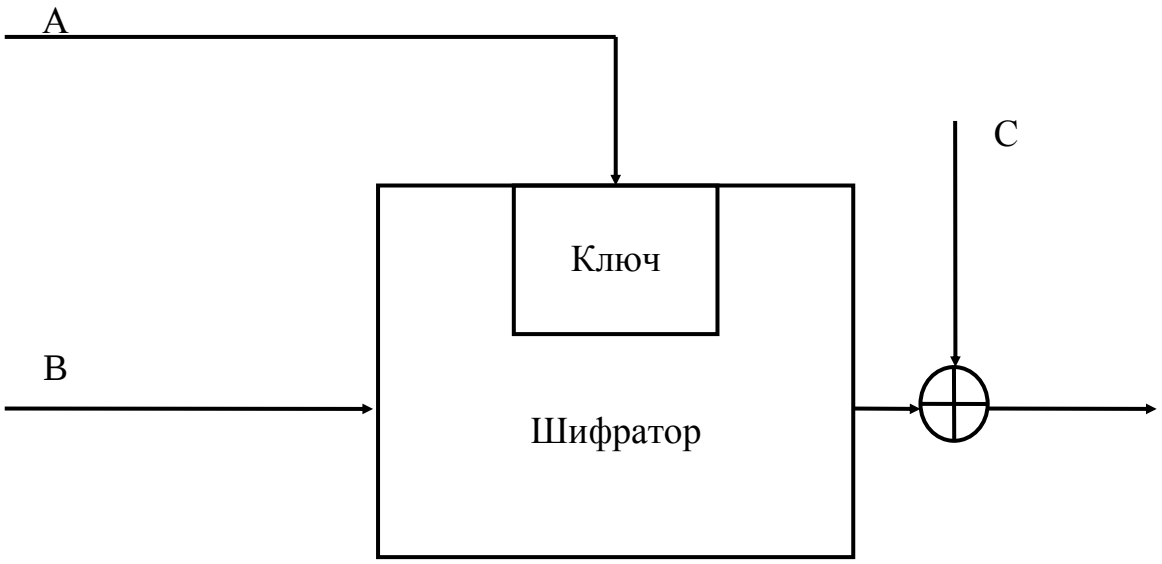


Рисунок 6.2 – Обобщенная схема формирования хэш-функции

Сообщение M разбивается на блоки M_i принятой длины, которые обрабатываются поочередно.

Три различные переменные A, B и C могут принимать одно из четырех возможных значений, поэтому в принципе можно получить 64 варианта общей схемы этого типа. Из них 52 варианта являются либо тривиально слабыми, либо небезопасными. Остальные 12 безопасных схем хэширования перечислены в табл. 6.1 [121].

Таблица 6.1

Схемы безопасного хэширования, у которых длина хэш-значения равна длине блока

Номер схемы	Функция хэширования
1	$H_i = E_{H_{i-1}}(M_i) \oplus M_i$
2	$H_i = E_{H_{i-1}}(M_i \oplus H_{i-1}) \oplus M_i \oplus H_{i-1}$
3	$H_i = E_{H_{i-1}}(M_i) \oplus H_{i-1} \oplus M_i$
4	$H_i = E_{H_{i-1}}(M_i \oplus H_{i-1}) \oplus M_i$
5	$H_i = E_{M_i}(H_{i-1}) \oplus H_{i-1}$
6	$H_i = E_{M_i}(M_i \oplus H_{i-1})\ M_i \oplus H_{i-1}$
7	$H_i = E_{M_i}(H_{i-1})\ M_i \oplus H_{i-1}$
8	$H_i = E_{M_i}(M_i \oplus H_{i-1}) \oplus H_{i-1}$
9	$H_i = E_{M_i \oplus H_{i-1}}(M_i) \oplus M_i$
10	$H_i = E_{M_i \oplus H_{i-1}}(H_{i-1}) \oplus H_{i-1}$

11	$H_i = E_{M_i \oplus H_{i-1}}(M_i) \oplus H_{i-1}$
12	$H_i = E_{M_i \oplus H_{i-1}}(H_{i-1}) \oplus M_i$

Первые четыре схемы хэширования, являющиеся безопасными при всех атаках, приведены на рис.6.3.

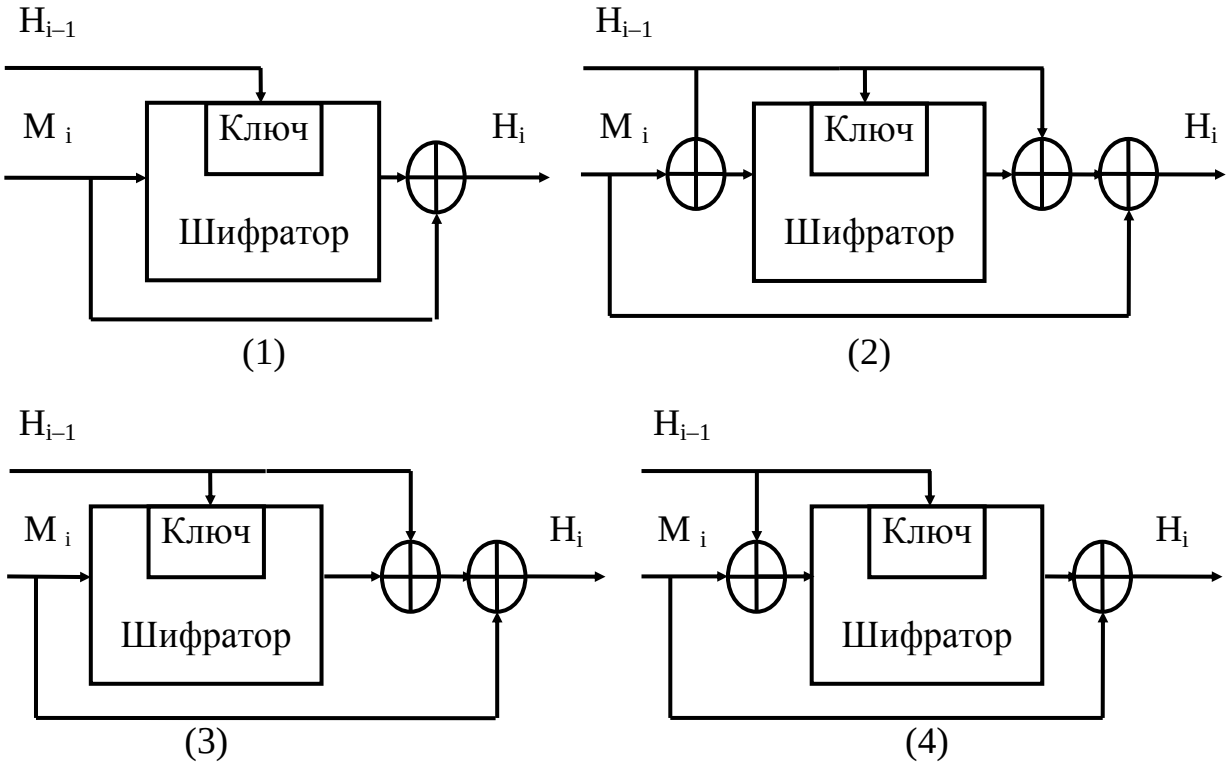


Рисунок 6.3 – Четыре схемы безопасного хэширования

6.2.2. Отечественный стандарт хэш-функции

Российский стандарт гост р 34.11-94 определяет алгоритм и процедуру вычисления хэш-функции для любых последовательностей двоичных символов, применяемых в криптографических методах обработки и защиты информации. Этот стандарт базируется на блочном алгоритме шифрования ГОСТ 28147-89, хотя в принципе можно было бы использовать и другой блочный алгоритм шифрования с 64-битовым блоком и 256-битовым ключом.

Данная хэш-функция формирует 256-битовое хэш-значение.

Функция сжатия $H_i = f(M_i, H_{i-1})$ (оба операнда M_i и H_{i-1} являются 256-битовыми величинами) определяется следующим образом:

1. Генерируются 4 ключа шифрования K_j, j = 1…4, путем линейного смешивания M_i, H_{i-1} и некоторых констант C_j.

2. Каждый ключ K_j, используют для шифрования 64-битовых подслов h_i слова H_{i-1} в режиме простой замены: $S_j = E_{K_j}(h_j)$. Результирующая последовательность S_4, S_3, S_2, S_1 длиной 256 бит запоминается во временной переменной S.

3. Значение H_i является сложной, хотя и линейной функцией смешивания S, M_i и H_{i-1}.

При вычислении окончательного хэш-значения сообщения M учитываются значения трех связанных между собой переменных:

H_n – хэш-значение последнего блока сообщения;

Z – значение контрольной суммы, получаемой при сложении по модулю 2 всех блоков сообщения;

L – длина сообщения.

Эти три переменные и дополненный последний блок M´ сообщения объединяются в окончательное хэш-значение следующим образом:

$$H = f (Z \oplus M', f (L, f (M', H_n))).$$

Данная хэш-функция определена стандартом ГОСТ Р 34.11-94 для использования совместно с российским стандартом электронной цифровой подписи [40, 41].

6.3. Алгоритмы электронной цифровой подписи

Технология применения системы ЭЦП предполагает наличие сети абонентов, посылающих друг другу подписанные электронные документы. Для каждого абонента генерируется пара ключей: секретный и открытый. Секретный ключ хранится абонентом в тайне и используется им для формирования ЭЦП. Открытый ключ известен всем другим пользователям и предназначен для проверки ЭЦП получателем подписанного электронного документа. Иначе говоря, открытый ключ является необходимым инструментом, позволяющим проверить подлинность электронного документа и автора подписи. Открытый ключ не позволяет вычислить секретный ключ.

Для генерации пары ключей (секретного и открытого) в алгоритмах ЭЦП, как и в асимметричных системах шифрования, используются разные математические схемы, основанные на применении однонаправленных функций. Эти схемы разделяются на две группы. В основе такого разделения лежат известные сложные вычислительные задачи:

- задача факторизации (разложения на множители) больших целых чисел;
- задача дискретного логарифмирования.

6.3.1. Алгоритм цифровой подписи RSA

Первой и наиболее известной во всем мире конкретной системой ЭЦП стала система RSA, математическая схема которой была разработана в 1977 г. в Массачуссетском технологическом институте США.

Сначала необходимо вычислить пару ключей (секретный ключ и открытый ключ). Для этого отправитель (автор) электронных документов вычисляет два больших простых числа P и Q, затем находит их произведение

$$N = P * Q$$

и значение функции

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

Далее отправитель вычисляет число E из условий:

$$E \le \varphi (N), \quad НОД (E, \varphi (N)) = 1$$

и число D из условий:

$$D < N, \quad E * D \equiv 1 \pmod{\varphi (N)}.$$

Пара чисел (E,N) является открытым ключом. Эту пару чисел автор передает партнерам по переписке для проверки его цифровых подписей. Число D сохраняется автором как секретный ключ для подписывания.

Обобщенная схема формирования и проверки цифровой подписи RSA показана на рис.6.4.

Допустим, что отправитель хочет подписать сообщение M перед его отправкой. Сначала сообщение M (блок информации, файл, таблица) сжимают с помощью хэш-функции h(·) в целое число m:

$$m = h(M).$$

Затем вычисляют цифровую подпись S под электронным документом M, используя хэш-значение m и секретный ключ D:

$$S = m^D \pmod N.$$

Пара (M,S) передается партнеру-получателю как электрон-ный документ m, подписанный цифровой подписью S, причем подпись S сформирована обладателем секретного ключа D.

После приема пары (M,S) получатель вычисляет хэш-значение сообщения M двумя разными способами. Прежде всего он восстанавливает хэш-значение m´, применяя криптографическое преобразование подписи S с использованием открытого ключа E:

$$m' = S^E \pmod N.$$

Кроме того, он находит результат хэширования принятого сообщения M с помощью такой же хэш-функции h(·):

$$m = h(M).$$

Если соблюдается равенство вычисленных значений, т.е.

$$S^E \pmod N = h(M),$$

то получатель признает пару (M,S) подлинной. Доказано, что только обладатель секретного ключа D может сформировать цифровую подпись S по документу M, а определить секретное число D по открытому числу E не легче, чем разложить модуль N на множители.

Кроме того, можно строго математически доказать, что результат проверки цифровой подписи S будет положительным только в том случае, если при вычислении S был использован секретный ключ D, соответствующий открытому ключу E. Поэтому открытый ключ E иногда называют "идентификатором" подписавшего.

Недостатки алгоритма цифровой подписи RSA.

1. При вычислении модуля N, ключей E и D для системы цифровой подписи RSA необходимо проверять большое количество дополнительных условий, что сделать практически трудно. Невыполнение любого из этих условий делает **возможным фальсификацию** цифровой подписи со стороны того, кто обнаружит такое невыполнение. При подписании важных документов нельзя допускать такую возможность даже теоретически.

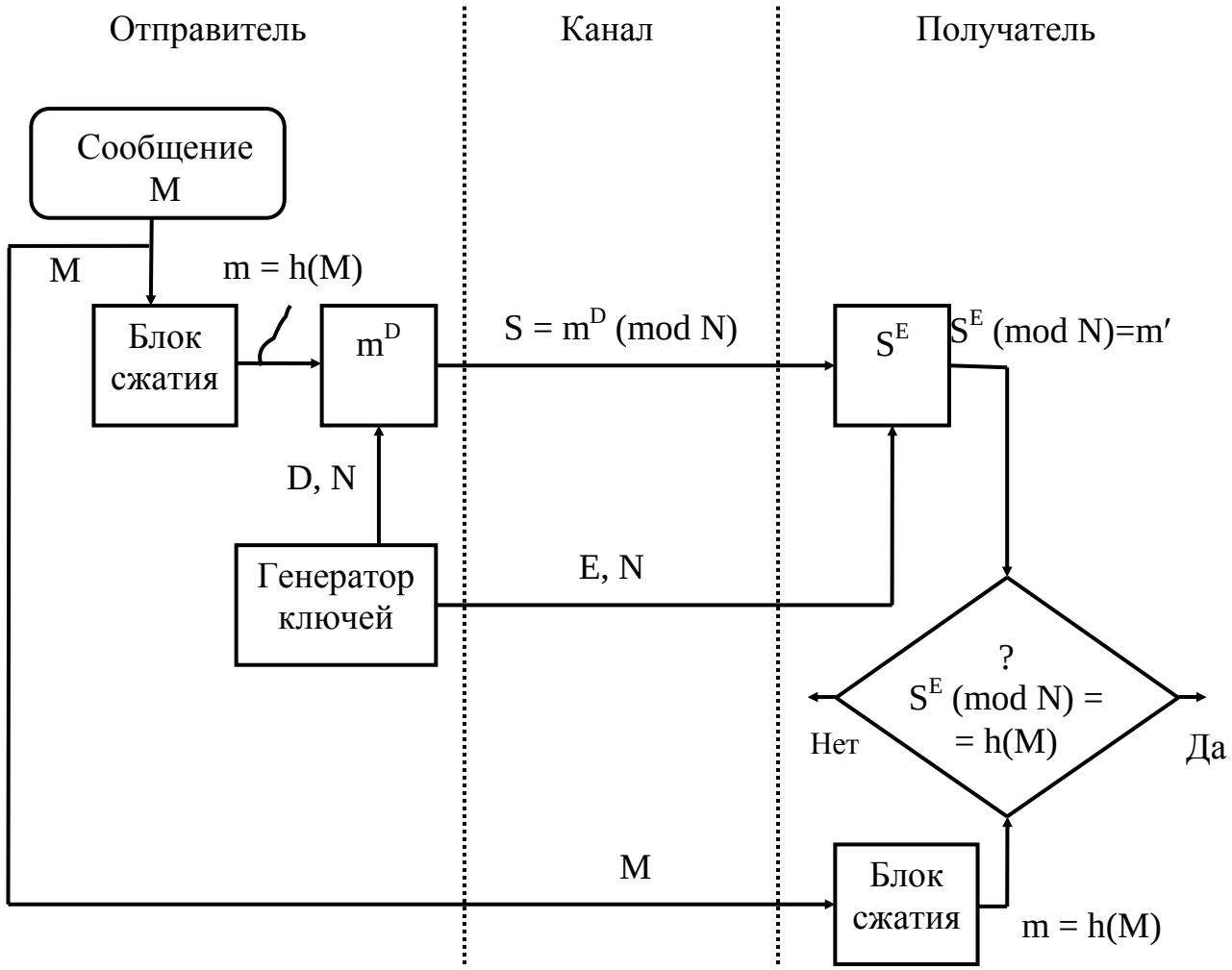


Рисунок 6.4 – Обобщенная схема цифровой подписи RSA

2. Для обеспечения криптостойкости цифровой подписи RSA по отношению к попыткам фальсификации на уровне, например, национального стандарта США на шифрование информации (алгоритм DES), т.е. 10^{18}, необходимо использовать при вычислениях N, D и E целые числа не менее 2^{512} (или около 10^{154}) каждое, что требует больших вычислительных затрат, превышающих на 20…30% вычислительные затраты других алгоритмов циф-ровой подписи при сохранении того же уровня криптостойкости.

3. Цифровая подпись RSA уязвима к так называемой мультипликативной атаке. Иначе говоря, алгоритм цифровой подписи RSA позволяет злоумышленнику без знания секретного ключа D сформировать подписи под теми документами, у которых результат хэширования можно вычислить как произведение результатов хэширования уже подписанных документов.

Более надежный и удобный для реализации на персональных компьютерах алгоритм цифровой подписи был разработан в 1984 г. американцем арабского происхождения Тахером Эль Гамалем. В 1991 г. НИСТ США обосновал перед комиссией Конгресса США выбор алгоритма цифровой подписи Эль Гамаля в качестве основы для национального стандарта.

6.3.2. Алгоритм цифровой подписи Эль Гамаля (EGSA)

Название EGSA происходит от слов El Gamal Signature Algorithm (алгоритм цифровой подписи Эль Гамаля). Идея EGSA основана на том, что для обоснования практической невозможности фальсификации цифровой подписи может быть использована более сложная вычислительная задача, чем разложение на множители большого целого числа,– задача дискретного логарифмирования. Кроме того, Эль Гамалю удалось избежать явной слабости алгоритма цифровой подписи RSA, связанной с возможностью подделки цифровой подписи под некоторыми сообщениями без определения секретного ключа.

Рассмотрим подробнее алгоритм цифровой подписи Эль Гамаля. Для того чтобы генерировать пару ключей (открытый ключ – секретный ключ), сначала выбирают некоторое большое простое целое число P и большое целое число G, причем $G < P$. Отправитель и получатель подписанного документа используют при вычислениях одинаковые большие целые числа P ($\sim 10^{308}$ или $\sim 2^{1024}$) и G ($\sim 10^{154}$ или $\sim 2^{512}$), которые не являются секретными.

Отправитель выбирает случайное целое число X, $1 < X \le (P-1)$, и вычисляет

$$Y = G^X \bmod P.$$

Число Y является открытым ключом, используемым для проверки подписи отправителя. Число Y открыто передается всем потенциальным получателям документов.

Число X является секретным ключом отправителя для подписывания документов и должно храниться в секрете.

Для того чтобы подписать сообщение M, сначала отправитель хэширует его с помощью хэш-функции $h(\cdot)$ в целое число m:

$$m = h(M), \quad 1 < m < (P-1),$$

и генерирует случайное целое число K, $1 < K < (P-1)$, такое, что K и $(P-1)$ являются взаимно простыми. Затем отправитель вычисляет целое число a:

$$a = G^K \bmod P$$

и, применяя расширенный алгоритм Евклида, вычисляет с помощью секретного ключа X целое число b из уравнения

$$m = X * a + K * b \pmod{(P-1)}.$$

Пара чисел (a,b) образует цифровую подпись S:

$$S = (a,b),$$

проставляемую под документом M.

Тройка чисел (M,a,b) передается получателю, в то время как пара чисел (X,K) держится в секрете.

После приема подписанного сообщения (M,a,b) получатель должен проверить, соответствует ли подпись S = (a,b) сообщению M. Для этого получатель сначала вычисляет по принятому сообщению M число

$$m = h(M),$$

т.е. хэширует принятое сообщение M.

Затем получатель вычисляет значение

$$A = Y^a a^b \pmod P$$

и признает сообщение M подлинным, если, и только если

$$A = G^m \pmod P.$$

Иначе говоря, получатель проверяет справедливость соотношения

$$Y^a a^b \pmod P = G^m \pmod P.$$

Можно строго математически доказать, что последнее равенство будет выполняться тогда, и только тогда, когда подпись S=(a,b) под документом M получена с помощью именно того секретного ключа X, из которого был получен открытый ключ Y. Таким образом, можно надежно удостовериться, что отправителем сообщения M был обладатель именно данного секретного ключа X, не раскрывая при этом сам ключ, и что отправитель подписал именно этот конкретный документ M.

Следует отметить, что выполнение каждой подписи по методу Эль Гамаля требует нового значения k, причем это значение должно выбираться случайным образом. Если нарушитель раскроет когда-либо значение k, повторно используемое отправителем, то он сможет раскрыть секретный ключ x отправителя.

Схема Эль Гамаля является характерным примером подхода, который допускает пересылку сообщения M в открытой форме вместе с присоединенным аутентификатором (a,b). В таких случаях процедура установления подлинности принятого сообщения состоит в проверке соответствия аутентификатора сообщению.

Схема цифровой подписи Эль Гамаля имеет ряд преимуществ по сравнению со схемой цифровой подписи RSA:

1. При заданном уровне стойкости алгоритма цифровой подписи целые числа, участвующие в вычислениях, имеют запись на 25% короче, что уменьшает сложность вычислений почти в два раза и позволяет заметно сократить объем используемой памяти.

2. При выборе модуля P достаточно проверить, что это число является простым и что у числа (P –1) имеется большой простой множитель (т.е. всего два достаточно просто проверяемых условия).

3. Процедура формирования подписи по схеме Эль Гамаля не позволяет вычислять цифровые подписи под новыми сообщениями без знания секретного ключа (как в RSA).

Однако алгоритм цифровой подписи Эль Гамаля имеет и некоторые недостатки по сравнению со схемой подписи RSA. В частности, длина цифровой подписи получается в 1,5 раза больше, что, в свою очередь, увеличивает время ее вычисления.

6.3.3. Алгоритм цифровой подписи DSA

Алгоритм цифровой подписи DSA (Digital Signature Algorithm) предложен в 1991 г. в НИСТ США для использования в стандарте цифровой подписи DSS (Digital Signature Standard). Алгоритм DSA является развитием алгоритмов цифровой подписи Эль Гамаля и К.Шнорра [121].

Отправитель и получатель электронного документа используют при вычислении большие целые числа: G и P – простые числа, l бит каждое ($512 \le l \le 1024$); Q – простое число длиной 160 бит (делитель числа (P –1)). Числа G, P, Q являются открытыми и могут быть общими для всех пользователей сети.

отправитель выбирает случайное целое число x, 1< x< q. Число x является секретным ключом **отправителя для формирования электронной цифровой подписи.**

Затем отправитель вычисляет значение

$$Y = G^X \bmod P.$$

Число **Y является открытым ключом** для проверки подписи отправителя. Число Y передается всем получателям документов.

Этот алгоритм также предусматривает использование односторонней функции хэширования $h(\cdot)$. В стандарте DSS определен алгоритм безопасного хэширования SHA (Secure Hash Algorithm).

Для того чтобы подписать документ M, отправитель хэширует его в целое хэш-значение m:

$$m = h(M), \quad 1< m < q,$$

затем генерирует случайное целое число K, 1< K< q, и вычисляет число r:

$$r = (G^K \bmod P) \bmod q.$$

Затем отправитель вычисляет с помощью секретного ключа X целое число s:

$$s = \frac{m + r * X}{K} \bmod q.$$

Пара чисел r и s образует цифровую подпись

$$S = (r, s)$$

под документом M.

Таким образом, подписанное сообщение представляет собой тройку чисел [M, r, s].

Получатель подписанного сообщения [M, r, s] проверяет выполнение условий

$$0 < r < q, \quad 0 < s < q$$

и отвергает подпись, если хотя бы одно из этих условий не выполнено.

Затем получатель вычисляет значение

$$w = \frac{1}{s} \bmod q,$$

хэш-значение

$$m = h(M)$$

и числа

$$u_1 = (m * w) \bmod q,$$

$$u_2 = (r * w) \bmod q.$$

Далее получатель с помощью открытого ключа Y вычисляет значение

$$v = ((G^{u_1} * Y^{u_2}) \bmod P) \bmod q$$

и проверяет выполнение условия

$$v = r.$$

Если условие v = r выполняется, тогда подпись S = (r,s) под документом M признается получателем подлинной.

Можно строго математически доказать, что последнее равенство будет выполняться тогда, и только тогда, когда подпись S = (r,s) под документом M получена с помощью именно того секретного ключа X, из которого был получен открытый ключ Y. Таким образом, можно надежно удостовериться, что отправитель сообщения владеет именно данным секретным ключом X (не раскрывая при этом значения ключа X) и что отправитель подписал именно данный документ M.

По сравнению с алгоритмом цифровой подписи Эль Гамаля алгоритм DSA имеет следующие основные преимущества:

1. При любом допустимом уровне стойкости, т.е. при любой паре чисел G и P (от 512 до 1024 бит), числа q, X, r, s имеют длину по 160 бит, сокращая длину подписи до 320 бит.

2. Большинство операций с числами K, r, s, X при вычислении подписи производится по модулю числа q длиной 160 бит, что сокращает время вычисления подписи.

3. При проверке подписи большинство операций с числами u_1, u_2, v, w также производится по модулю числа q длиной 160 бит, что сокращает объем памяти и время вычисления.

Недостатком алгоритма DSA является то, что при подписывании и при проверке подписи приходится выполнять сложные операции деления по модулю q:

$$s = \frac{m + rX}{K} \pmod q, \quad w = \frac{1}{s} \pmod q,$$

что не позволяет получать максимальное быстродействие.

Следует отметить, что реальное исполнение алгоритма DSA может быть ускорено с помощью выполнения предварительных вычислений. Заметим, что значение r не зависит от сообщения M и его хэш-значения m. Можно заранее создать строку случайных значений K и затем для каждого из этих значений вычислить значения r. Можно также заранее вычислить обратные значения K^{-1} для каждого из значений K. Затем, при поступлении сообщения M, можно вычислить значение s для данных значений r и K^{-1}. Эти предварительные вычисления значительно ускоряют работу алгоритма DSA.

6.3.4. Отечественный стандарт цифровой подписи

Отечественный стандарт цифровой подписи обозначается как ГОСТ Р 34.10-94 [40]. Алгоритм цифровой подписи, определяемый этим стандартом, концептуально близок к алгоритму DSA. В нем используются следующие параметры:

p – большое простое число длиной от 509 до 512 бит либо от 1020 до 1024 бит;

q – простой сомножитель числа (p –1), имеющий длину 254…256 бит.

a – любое число, меньшее (p –1), причем такое, что $a^q \bmod p = 1$;

x – некоторое число, меньшее q;

$y = a^x \bmod p$.

Кроме того, этот алгоритм использует однонаправленную хэш-функцию H(x). Стандарт ГОСТ Р 34.11-94 определяет хэш-функцию, основанную на использовании стандартного симметричного алгоритма ГОСТ 28147-89.

Первые три параметра p, q и a являются открытыми и могут быть общими для всех пользователей сети. Число x является секретным ключом. Число y является открытым ключом.

Чтобы подписать некоторое сообщение m, а затем проверить подпись, выполняются следующие шаги.

1. Пользователь A генерирует случайное число k, причем $k < q$.

2. Пользователь A вычисляет значения

$$r = (a^k \bmod p) \bmod q,$$

$$s = (x * r + k\,(H(m))) \bmod q.$$

Если $H(m) \bmod q = 0$, то значение $H(m) \bmod q$ принимают равным единице. Если r=0, то выбирают другое значение k и начина-ют снова.

Цифровая подпись представляет собой два числа:

$$r \bmod 2^{256} \quad \text{и} \quad s \bmod 2^{256}.$$

Пользователь A отправляет эти числа пользователю B.

3. Пользователь B проверяет полученную подпись, вычисляя

$$v = H(m)^{q-2} \bmod q,$$

$$z_1 = (s * v) \bmod q,$$

$$z_2 = ((q - r) * v) \bmod q,$$

$$u = ((a^{z_1} * y^{z_2}) \bmod p) \bmod q.$$

Если u = r, то подпись считается верной.

Различие между этим алгоритмом и алгоритмом DSA заключается в том, что в DSA

$$s = (k^{-1}\,(x * r + (H(m)))) \bmod q,$$

что приводит к другому уравнению верификации.

Следует также отметить, что в отечественном стандарте ЭЦП параметр q имеет длину 256 бит. Западных криптографов вполне устраивает q длиной примерно 160 бит. Различие в значениях параметра q является отражением стремления разработчиков отечественного стандарта к получению более безопасной подписи.

Этот стандарт вступил в действие с начала 1995 г.

6.4. Цифровые подписи с дополнительными функциональными свойствами.

Рассматриваемые в этом разделе цифровые подписи обладают дополнительными функциональными возможностями, помимо обычных свойств аутентификации сообщения и невозможности отказа подписавшего лица от обязательств, связанных с подписанным текстом. В большинстве случаев они объединяют базовую схему цифровой подписи, например, на основе алгоритма RSA, со специальным протоколом, обеспечивающим достижение

тех дополнительных свойств, которыми базовая схема цифровой подписи не обладает.

К схемам цифровой подписи с дополнительными функциональными свойствами относятся:

- схемы слепой (blind) подписи,
- схемы неоспоримой (undeniable) подписи.

6.4.1. Схемы слепой подписи

В отличие от обычных схем цифровой подписи, описанных в разделе 6.3, *схемы слепой подписи* (иногда называемые схемами подписи вслепую) являются двусторонними протоколами между отправителем A и стороной B, подписывающей документ.

Основная идея этих схем заключается в следующем. Отправитель A посылает порцию информации стороне B, которую B подписывает и возвращает A. Используя полученную подпись, сторона A может вычислить подпись стороны B на более важном для себя сообщении m. По завершении этого протокола сторона B ничего не знает ни о сообщении m, ни о подписи под этим сообщением.

Цель слепой подписи состоит в том, чтобы воспрепятствовать подписывающему лицу B ознакомиться с сообщением стороны A, которое он подписывает, и с соответствующей подписью под этим сообщением. Поэтому в дальнейшем подписанное сообщение невозможно связать со стороной A.

Приведем пример применения слепой подписи. Схема слепой подписи может найти применение в тех случаях, когда отправитель A (клиент банка) не хочет, чтобы подписывающая сторона B (банк) имела возможность в дальнейшем связать сообщение m и подпись $s_B(m)$ с определенным шагом выполненного ранее протокола.

В частности, это может быть важно при организации анонимных безналичных расчетов, когда сообщение m могло бы представлять денежную сумму, которую A хочет потратить. Когда сообщение m с подписью $s_B(m)$ предъявляется банку B для оплаты, банк B не сможет проследить, кто именно из клиентов предъявляет подписанный документ. Это позволяет пользователю A остаться анонимным. Принципы организации системы анонимных безналичных расчетов с использованием так называемой “электронной наличности” (“цифровых денег”) на базе протоколов слепой подписи рассмотрены в.

Для построения протокола слепой подписи необходимы следующие компоненты:

1. Механизм обычной цифровой подписи для подписывающей стороны B. Пусть $s_B(X)$ обозначает подпись стороны B на документе X.
2. Функции f (·) и g (·) (известные только отправителю) такие, что

$$g (s_B (f (m))) = s_m (m),$$

при этом f (·) - маскирующая (blinding) функция,

g (·) - демаскирующая (unblinding) функция,

f (m) - замаскированное (blinded) сообщение m.

При выборе s_B, f и g существует ряд ограничений.

Выберем в качестве алгоритма подписи s_B для стороны B схему цифровой подписи RSA (см. п.6.3) с открытым ключом (N, E) и секретным ключом D, причем

$N = P * Q$ - произведение двух больших случайных простых чисел.

Пусть k - некоторое фиксированное целое число, взаимно простое с N, т.е. НОД (N, k) = 1.

Маскирующая функция $f: Z_n \to Z_n$

определяется как $f(m) = m * k^E \bmod N$,

а демаскирующая функция $g: Z_n \to Z_n$

определяется как $g(m) = k^{-1} m \bmod N$. При таком выборе f, g и s получаем

$$g(s_B(f(m))) = g(s_B(m\,k^E \bmod N)) = g(m^D k \bmod N) = m^D \bmod N = s_B(m),$$

что соответствует требованию 2.

Согласно протоколу слепой подписи, который предложил Д.Чом [121], отправитель A сначала получает подпись стороны B на замаскированном сообщении m^*. Используя эту подпись, сторона A вычисляет подпись B на заранее выбранном сообщении m, где $0 \le m \le N-1$. При этом стороне B ничего неизвестно ни о значении m, ни о подписи, связанной с m.

Пусть сторона B имеет для подписи по схеме RSA открытый ключ (N, E) и секретный ключ D. Пусть k - случайное секретное целое число, выбранное стороной A и удовлетворяющее условиям $0 \le k \le N-1$ и НОД (N, k) = 0.

Протокол слепой подписи Д.Чома включает следующие шаги:

(1) Отправитель A вычисляет замаскированное сообщение $m^* = m\,k^E \bmod N$ и посылает его стороне B.

(2) Подписывающая сторона B вычисляет подпись $s^* = (m^*)^D \bmod N$ и отправляет эту подпись стороне A.

Сторона A вычисляет подпись $s = k^{-1} s^* \bmod N$, которая является подписью B на сообщение m.

Нетрудно видеть, что

$$(m^*)^D \equiv (mk^E)^D \equiv m^D k \pmod N,$$

поэтому

$$k^{-1} s^* \equiv m^D k\, k^{-1} \equiv m^D \pmod N.$$

Д.Чом разработал несколько алгоритмов слепой подписи для создания системы анонимных безналичных электронных расчетов eCash [49, 108].

6.4.2. Схемы неоспоримой подписи

Неоспоримая подпись, как и обычная цифровая подпись, зависит от подписанного документа и секретного ключа. Однако в отличие от обычных цифровых подписей, неоспоримая подпись не может быть верифицирована без участия лица поставившего эту подпись. Возможно, более подходящим названием для этих подписей было бы “подписи, не допускающие подлога”.

Рассмотрим два возможных сценария применения неоспоримой подписи [107,117].

Сценарий 1. Сторона A (клиент) хочет получить доступ в защищенную зону, контролируемую стороной B (банком). Этой защищенной зоной может быть, например, депозитарий (хранилище ценностей клиентов). Сторона B требует от A поставить на заявке о допуске в защищенную зону подпись, время и дату до предоставления ему доступа. Если A применит неоспори-

мую подпись, тогда сторона B не сможет впоследствии доказать кому-либо, что A получил допуск, без непосредственного участия A в процессе верификации подписи.

Сценарий 2. Предположим, что известная корпорация A разработала пакет программного обеспечения. Чтобы гарантировать подлинность пакета и отсутствие в нем вирусов, сторона A подписывает этот пакет неоспоримой подписью и продает его стороне B. Сторона B решает сделать копии этого пакета программного обеспечения и перепродать его третьей стороне C. При использовании стороной A неоспоримой подписи сторона C не сможет убедиться в подлинности этого пакета программного обеспечения и отсутствии в нем вирусов без участия стороны A.

Конечно, этот сценарий не препятствует стороне B поставить на пакете свою подпись, но тогда для стороны B будут утрачены все маркетинговые преимущества, связанные с использованием торговой марки корпорации A. Кроме того, будет легче раскрыть мошенническую деятельность стороны B.

Рассмотрим алгоритм неоспоримой цифровой подписи, разработанный Д.Чомом [107]. Сначала опишем алгоритм генерации ключей, с помощью которого каждая сторона A, подписывающая документ, выбирает секретный ключ и соответствующий открытый ключ.

Каждая сторона A должна выполнить следующее:

1. Выбрать случайное простое число $p = 2q + 1$, где q - также простое число.
2. Выбрать генераторное число α для подгруппы порядка q в циклической группе Z_p^*:
   2.1. Выбрать случайный элемент $\beta \in Z_p^*$ и вычислить $\alpha = \beta^{(p-1)/q} \bmod p$.
   2.2. Если $\alpha = 1$, тогда возвратиться к шагу 2.1.
3. Выбрать случайное целое $x \in \{1, 2, ..., q-1\}$ и вычислить $y = \alpha^x \bmod p$.
4. Для стороны A открытый ключ равен (p, α, y), секретный ключ равен x.

Согласно алгоритму неоспоримой подписи Д.Чома, сторона A подписывает сообщение m, принадлежащее подгруппе порядка q в Z_p^*. Любая сторона B может проверить эту подпись при участии A.

В работе алгоритма неоспоримой подписи можно выделить два этапа:

1. генерация подписи,
2. верификация подписи.

На этапе генерации подписи сторона A вычисляет

$$s = m^x \bmod p,$$

где s - подпись стороны A на сообщении m.

Сообщение m с подписью s отсылается стороне B.

Этап верификации подписи выполняется стороной B с участием стороны A и включает следующие шаги:

(1) B получает подлинный открытый ключ (p, α, y) стороны A.

(2) B выбирает два случайных секретных целых числа $a, b \in \{1, 2, ..., q-1\}$.

(3) B вычисляет $z = s^a y^b \bmod p$ и отправляет значение z стороне A.

(4) A вычисляет $w = (z)^{1/x} \bmod p$, где $xx^{-1} \equiv 1 \pmod q$, и отправляет значение w стороне B.

(5) B вычисляет $w' = m^a \alpha^b \bmod p$ и признает подпись s подлинной, если и только если $w = w'$.

Убедимся, что проверка подписи s работает:

$w \equiv (z)^{1/x} \equiv (s^a y^b)^{1/x} \equiv (m^{xa}\alpha^{xb})^{1/x} \equiv m^a\alpha^b \equiv w' \bmod p.$

Можно показать, что с высокой степенью вероятности злоумышленник не сможет заставить B принять фальшивую подпись. Предположим, что s представляет собой подделку подписи стороны A на сообщении m, т.е. $s \neq m^x \bmod p$. Тогда вероятность принятия стороной этой подписи в данном алгоритме составляет только 1/q, причем эта вероятность не зависит от вычислительных ресурсов злоумышленника.

Подписавшая сторона A при некоторых обстоятельствах могла бы попытаться отказаться от своей подлинной подписи одним из трех способов:

(а) отказаться от участия в протоколе верификации;

(б) некорректно выполнить протокол верификации;

(в) объявить подпись фальшивой, даже если протокол верификации оказался успешным.

Отречение от подписи способом (а) рассматривалось бы как очевидная попытка неправомерного отказа.

Против способов (б) и (в) бороться труднее, здесь требуется специальный протокол дезавуирования. Этот протокол определяет, пытается ли подписавшая сторона A дезавуировать правильную подпись s или эта подпись является фальшивой. В этом протоколе по существу дважды применяется протокол верификации и затем производится проверка с целью убедиться, что сторона A выполняет этот протокол корректно.

Протокол дезавуирования для схемы неоспоримой подписи Д.Чома включает следующие шаги:

(1) B принимает от стороны A сообщение m с подписью s и получает подлинный открытый ключ (p, α, y) стороны A.

(2) B выбирает случайные секретные целые числа $a, b \in \{1, 2, ..., q-1\}$, вычисляет $z = s^a y^b \bmod p$ и отправляет значение z стороне A.

(3) A вычисляет $w = (z)^{1/x} \bmod p$, где $xx^{-1} \equiv 1 (\bmod q)$, и отправляет значение w стороне B.

(4) Если $w = m^a \alpha^b \bmod p$, тогда B признает подпись s подлинной и протокол прекращается.

(5) B выбирает случайные секретные целые числа $a', b' \in \{1, 2, ..., q-1\}$, вычисляет $z' = s^{a'} y^{b'} \bmod p$ и отправляет значение z' стороне A.

(6) A вычисляет $w' = (z')^{1/x} \bmod p$ и отправляет значение w' стороне B.

(7) Если $w' = m^{a'} \alpha^{b'} \bmod p$, тогда B принимает подпись s и протокол останавливается.

(8) B вычисляет $c = (w\alpha^{-b})^{a'} \bmod p$, $c' = (w'\alpha^{-b'})^a \bmod p$. Если c = c', тогда B заключает, что подпись s фальшивая; в противном случае, B делает вывод, что подпись s подлинная, а сторона A пытается дезавуировать подпись s.

Нетрудно убедиться в том, что этот протокол достигает поставленной цели. Пусть m - сообщение и предположим, что s - подпись стороны A под сообщением m.

Если подпись s фальшивая, т.е. $s \neq m^x \bmod p$ и если стороны A и B следуют протоколу должным образом, тогда w = w' (и поэтому справедливо заключение B, что подпись s фальшивая).

Пусть s на самом деле является подписью стороны A под сообщением m, т.е. $s = m^x \bmod p$. Предположим, что B точно следует протоколу, а A не следует. Тогда вероятность того, что w = w' (и A преуспевает в дезавуировании подписи), составляет только 1/q.

Следует отметить, что третья сторона C никогда не должна принимать в качестве доказательства подлинности подписи s запись стороной B протокола верификации, поскольку сторона B может выдумать успешную запись шага 2 и последующих шагов протокола верификации без участия подписывающей стороны A.

Неоспоримая подпись может быть верифицирована только путем непосредственного взаимодействия с подписывающей стороной A.

7. УПРАВЛЕНИЕ КРИПТОГРАФИЧЕСКИМИ КЛЮЧАМИ

Любая криптографическая система основана на использовании криптографических ключей. В симметричной криптосистеме отправитель и получатель сообщения используют один и тот же секретный ключ. Этот ключ должен быть неизвестен всем остальным и должен периодически обновляться одновременно у отправителя и получателя. Процесс распределения (рассылки) секретных ключей между участниками информационного обмена в симметричных криптосистемах имеет весьма сложный характер.

Асимметричная криптосистема предполагает использование двух ключей – открытого и личного (секретного). Открытый ключ можно разглашать, а личный надо хранить в тайне. При обмене сообщениями необходимо пересылать только открытый ключ. Важным требованием является обеспечение подлинности отправителя сообщения. Это достигается путем взаимной аутентификации участников информационного обмена.

Под *ключевой информацией* понимают совокупность всех действующих в сети ключей. Если не обеспечено достаточно надежное управление ключевой информацией, то, завладев ею, злоумышленник получает неограниченный доступ ко всей информации.

Управление ключами – информационный процесс, включающий реализацию следующих основных функций:

- генерация ключей;
- хранение ключей;
- распределение ключей.

7.1. Генерация ключей

Безопасность любого криптографического алгоритма определяется используемым криптографическим ключом. Добротные криптографические ключи должны иметь достаточную длину и случайные значения битов. В табл. 4.3 приведены длины ключей симметричной и асимметричной криптосистем, обеспечивающие одинаковую стойкость к атаке полного перебора (атаке "грубой силы") [123].

Для получения ключей используются аппаратные и программные средства генерации случайных значений ключей. Как правило, применяют датчики псевдослучайных чисел (ПСЧ). Однако степень случайности генерации чисел должна быть достаточно высокой. Идеальными генераторами являются устройства на основе "натуральных" случайных процессов, например на основе *белого радиошума*.

В сети со средними требованиями защищенности вполне приемлемы программные генераторы ключей, которые вычисляют ПСЧ как сложную функцию от текущего времени и (или) числа, введенного пользователем.

Один из методов генерации сеансового ключа для симметричных криптосистем описан в стандарте ANSI X9.17. Он предполагает использование криптографического алгоритма DES (хотя можно применить и другие симметричные алгоритмы шифрования).

Обозначения:

$E_K(X)$ – результат шифрования алгоритмом DES значения X;

К – ключ, зарезервированный для генерации секретных ключей;
V_0 – секретное 64-битовое начальное число;
Т – временная отметка.

Схема генерации случайного сеансового ключа R_i в соответствии со стандартом ANSI X 9.17 показана на рис. 7.1. Случайный ключ R_i генерируют, вычисляя значение

$$R_i = E_K (E_K (T_i) \oplus V_i).$$

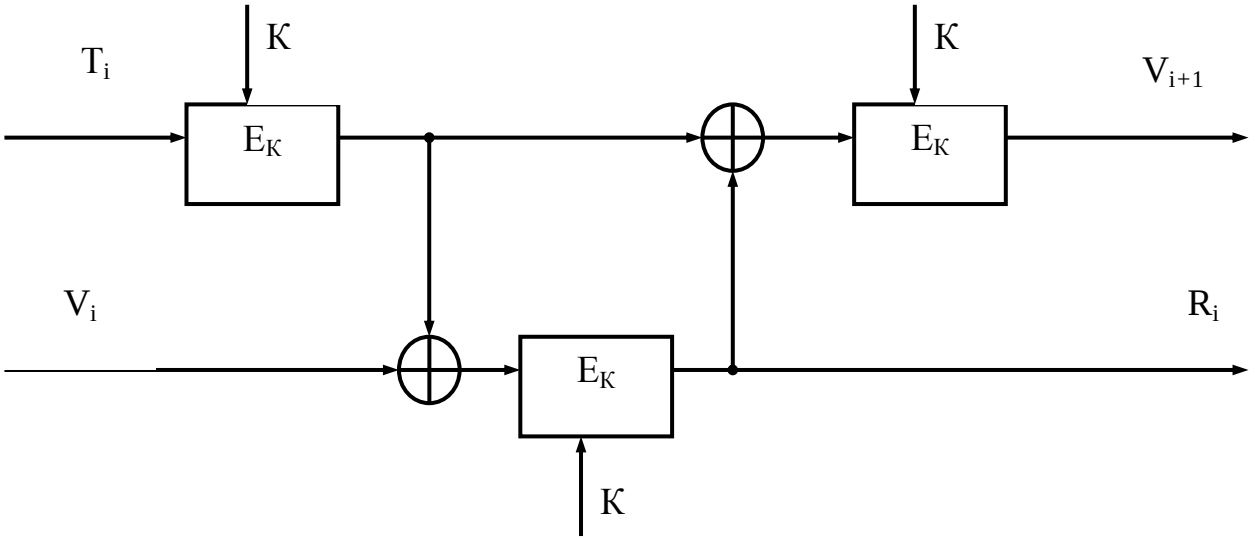


Рисунок 7.1 – Схема генерации случайного ключа R_i в соответствии со стандартом ANSI X9.17

Следующее значение V_{i+1} вычисляют так:

$$V_{i+1} = E_K (E_K (T_i) \oplus R_i).$$

Если необходим 128-битовый случайный ключ, генерируют пару ключей R_i, R_{i+1} и объединяют их вместе.

Если ключ не меняется регулярно, это может привести к его раскрытию и утечке информации. Регулярную замену ключа можно осуществить, используя процедуру модификации ключа.

Модификация ключа – это генерирование нового ключа из предыдущего значения ключа с помощью односторонней (однонаправленной) функции. Участники информационного обмена разделяют один и тот же ключ и одновременно вводят его значение в качестве аргумента в одностороннюю функцию, получая один и тот же результат. Затем они берут определенные биты из этих результатов, чтобы создать новое значение ключа.

Процедура модификации ключа работоспособна, но надо помнить, что новый ключ безопасен в той же мере, в какой был безопасен прежний ключ. Если злоумышленник сможет добыть прежний ключ, то он сможет выполнить процедуру модификации ключа.

Генерация ключей для асимметричных криптосистем с открытыми ключами много сложнее, потому что эти ключи должны обладать определенными математическими свойствами (они должны быть очень большими и простыми и т.д.).

7.2. Хранение ключей

Под *функцией хранения ключей* понимают организацию их безопасного хранения, учета и удаления. Ключ является самым привлекательным для злоумышленника объектом, открывающим ему путь к конфиденциальной информации. Поэтому вопросам безопасного хранения ключей следует уделять особое внимание. Секретные ключи никогда не должны записываться в явном виде на носителе, который может быть считан или скопирован.

7.2.1. Носители ключевой информации

Ключевой носитель может быть технически реализован различным образом на разных носителях информации – магнитных дисках, устройствах хранения ключей типа TOUCH MEMORY, пластиковых картах и т.д.

Магнитные диски представляют собой распространенный тип носителя ключевой информации. Применение магнитного диска (МД) в качестве носителя ключа позволяет реализовать необходимое свойство отчуждаемости носителя ключа от защищенной компьютерной системы, т.е. осуществлять временное изъятие МД из состава технических средств компьютерной системы. Особенно целесообразно использование в качестве ключевых носителей съемных накопителей – гибких магнитных дисков, съемных магнитооптических носителей и т.д. [73].

Основное преимущество МД по сравнению с другими носителями ключевой информации заключается в том, что оборудование для взаимодействия с МД (дисковод) входит в состав штатных средств компьютера.

Другая важная особенность, определяющая широкое распространение МД, - стандартный формат хранения информации на дисках и стандартные программные средства доступа к дискам. Кроме того, из всех средств хранения ключевой информации гибкие магнитные диски имеют самую низкую стоимость..

Для обеспечения надежного хранения ключевой информации на МД применяют как минимум двукратное резервирование объектов хранения. Это позволяет защитить ключевую информацию от ошибок при считывании с МД и от сбоев программной и аппаратной части.

Для предотвращения возможности перехвата ключевой информации в процессе ее чтения с МД применяют хранение ключевой информации на МД в зашифрованном виде.

Устройство хранения ключей типа TOUCH MEMORY является относительно новым носителем ключевой информации, предложенным американской компанией Dallas Semiconductor. Носитель информации TOUCH MEMORY (TM) представляет собой энергонезависимую память, размещенную в металлическом корпусе, с одним сигнальным контактом и одним контактом земли. Корпус TM имеет диаметр 16,25 мм и толщину 3,1 или 5,89 мм (в зависимости от модификации прибора).

В структуру TM входят следующие основные блоки [73].

1. Постоянное запоминающее устройство (ПЗУ) хранит 64-разрядный код, состоящий из байтового кода типа прибора, 48-битового уникального серий-

ного номера и 8-битовой контрольной суммы. Содержимое ПЗУ уникально и не может быть изменено в течение всего срока службы прибора.
2. Оперативное запоминающее устройство (ОЗУ) емкостью от 128 до 8192 байт содержат практически все модификации ТМ. В одной из модификаций оперативная память аппаратно защищена от несанкционированного доступа.
3. Встроенная миниатюрная литиевая батарейка со сроком службы не менее 10 лет обеспечивает питанием все блоки устройства.

Особенностью технологии хранения и обмена ключевой информации между носителем ТМ и внешними устройствами является сравнительно низкая скорость (обусловленная последовательной передачей данных) и высокая вероятность сбоя в тракте чтения-записи, обусловленная тем, что контакт устройства ТМ с устройством чтения осуществляется пользователем вручную без дополнительной фиксации (простое касание, что и определило название прибора ТМ). В связи с этим особое значение приобретают вопросы надежного обмена между программами обработки ключевой информации пользователей и носителем ТМ.

В устройстве ТМ конструктивно отработаны вопросы надежности функционирования и вопросы интерфейса со считывающим устройством на основе одного сигнального контакта. Для обеспечения достоверного чтения применяются корректирующие коды, для обеспечения достоверной записи в приборе предусмотрена технология буферизации. При проведении операции записи первоначально вектор передаваемой в ТМ информации помещается в буфер, далее выполняется операция чтения из буфера, затем прочтенная из буфера информация сравнивается с записываемой и в случае совпадения подается сигнал на перенос информации из буфера в память долговременного хранения.

Таким образом, носитель ТМ является микроконтроллерным устройством без собственной вычислительной мощности и с ограниченным объемом хранения, но с достаточно высокими надежностными характеристиками. Поэтому применение ТМ вполне обосновано в случае повышенных требований к надежности носителя ключа и небольшого объема ключевой информации, хранимой в ТМ.

Электронные пластиковые карты становятся в настоящее время наиболее распространенным и универсальным носителем конфиденциальной информации, который позволяет идентифицировать и аутентифицировать пользователей, хранить криптографические ключи, пароли и коды.

Интеллектуальные карты (смарт-карты), обладающие наибольшими возможностями, эффективно применяются не только для хранения ключевой информации, но и широко используются в электронных платежных системах, в комплексных решениях для медицины, транспорта, связи, образования и т.п. Более подробные сведения об электронных пластиковых картах приводятся в разделе 9.4.

7.2.2. Концепция иерархии ключей

Любая информация об используемых ключах должна быть защищена, в частности храниться в зашифрованном виде.

Необходимость в хранении и передаче ключей, зашифрованных с помощью других ключей, приводит к концепции *иерархии ключей*. В стандарте ISO 8532 (Banking-Key Management) подробно изложен **метод главных/сеансовых ключей** (master/session keys). Суть метода состоит в том, что вводится иерархия ключей: **главный ключ** (ГК), **ключ шифрования ключей** (КК), **ключ шифрования данных** (КД).

Иерархия ключей может быть:

- двухуровневой (КК/КД),
- трехуровневой (ГК/КК/КД).

Самым нижним уровнем являются *рабочие или сеансовые КД*, которые используются для шифрования данных, персональных идентификационных номеров (PIN) и аутентификации сообщений. Когда эти ключи надо зашифровать с целью защиты при передаче или хранении, используют ключи следующего уровня – *ключи шифрования ключей*. Ключи шифрования ключей никогда не должны использоваться как сеансовые (рабочие) КД, и наоборот.

Такое разделение функций необходимо для обеспечения максимальной безопасности. Фактически стандарт устанавливает, что различные типы рабочих ключей (например, для шифрования данных, для аутентификации и т.д.) должны всегда шифроваться с помощью различных версий ключей шифрования ключей.

В частности, ключи шифрования ключей, используемые для пересылки ключей между двумя узлами сети, известны также как *ключи обмена между узлами сети* (cross domain keys). Обычно в канале используются два ключа для обмена между узлами сети, по одному в каждом направлении. Поэтому каждый узел сети будет иметь *ключ отправления* для обмена с узлами сети и *ключ получения* для каждого канала, поддерживаемого другим узлом сети.

На верхнем уровне иерархии ключей располагается *главный ключ, мастер-ключ*. Этот ключ применяют для шифрования КК, когда требуется сохранить их на диске. Обычно в каждом компьютере используется только один мастер-ключ.

Мастер-ключ распространяется между участниками обмена неэлектронным способом – при личном контакте, чтобы исключить его перехват и/или компрометацию. Раскрытие противником значения мастер-ключа полностью уничтожает защиту компьютера.

Значение мастер-ключа фиксируется на длительное время (до нескольких недель или месяцев). Поэтому генерация и хранение мастер-ключей являются критическими вопросами криптографической защиты. На практике мастер-ключ компьютера создается истинно случайным выбором из всех возможных значений ключей. Мастер-ключ помещают в защищенный по считыванию и записи и от механических воздействий блок криптографической системы таким образом, чтобы раскрыть значение этого ключа было невозможно. Однако все же должен существовать способ проверки, является ли значение ключа правильным.

Проблема аутентификации мастер-ключа может быть решена различными путями. Один из способов аутентификации показан на рис. 7.2.

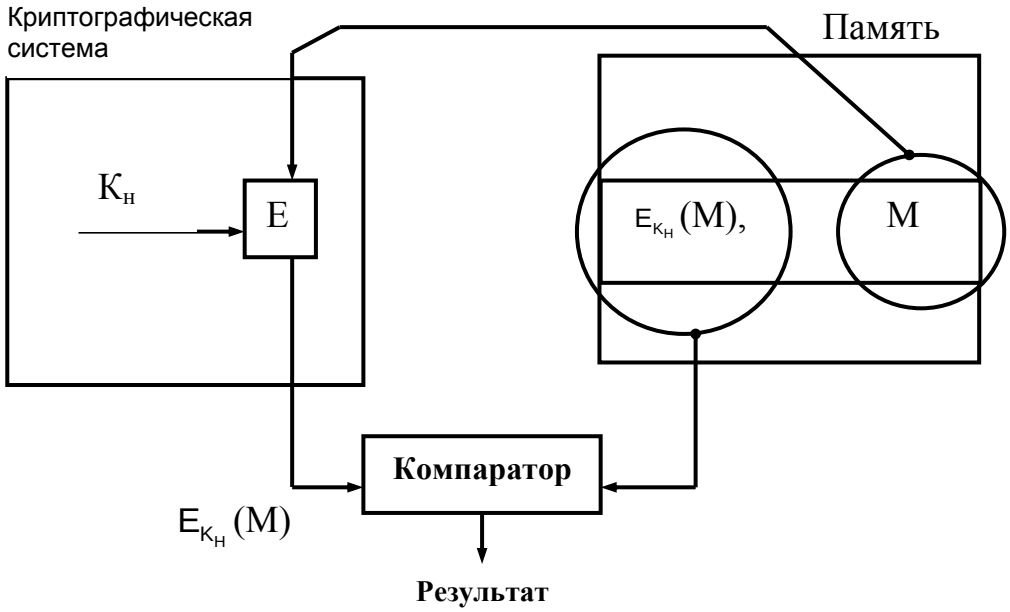


Рисунок 7.2 – Схема аутентификации мастер-ключа хост-компьютера

Администратор, получив новое значение мастер-ключа $К_Н$ хост-компьютера, шифрует некоторое сообщение М ключом $К_Н$. Пара (криптограмма $Ек_Н$ (М), сообщение М) помещается в память компьютера. Всякий раз, когда требуется аутентификация мастер-ключа хост-компьютера, берется сообщение М из памяти и подается в криптографическую систему. Получаемая криптограмма сравнивается с криптограммой, хранящейся в памяти. Если они совпадают, считается, что данный ключ является правильным.

Рабочие ключи (например, сеансовый) обычно создаются с помощью псевдослучайного генератора и могут храниться в незащищенном месте. Это возможно, поскольку такие ключи генерируются в форме соответствующих криптограмм, т.е. генератор ПСЧ выдает вместо ключа $К_S$ его криптограмму $E_{К_Н}(К_S)$, получаемую с помощью мастер-ключа хост-компьютера. Расшифровывание такой криптограммы выполняется только перед использованием ключа $К_S$.

Схема защиты рабочего (сеансового) ключа показана на рис. 7.3. Чтобы зашифровать сообщение М ключом $К_S$, на соответствующие входы криптографической системы подается криптограмма $E_{К_Н}(К_S)$ и сообщение М. Криптографическая система сначала восстанавливает ключ $К_S$, а затем шифрует сообщение М, используя открытую форму сеансового ключа $К_S$.

Таким образом, безопасность сеансовых ключей зависит от безопасности криптографической системы. Криптографический блок может быть спроектирован как единая СБИС и помещен в физически защищенное место.

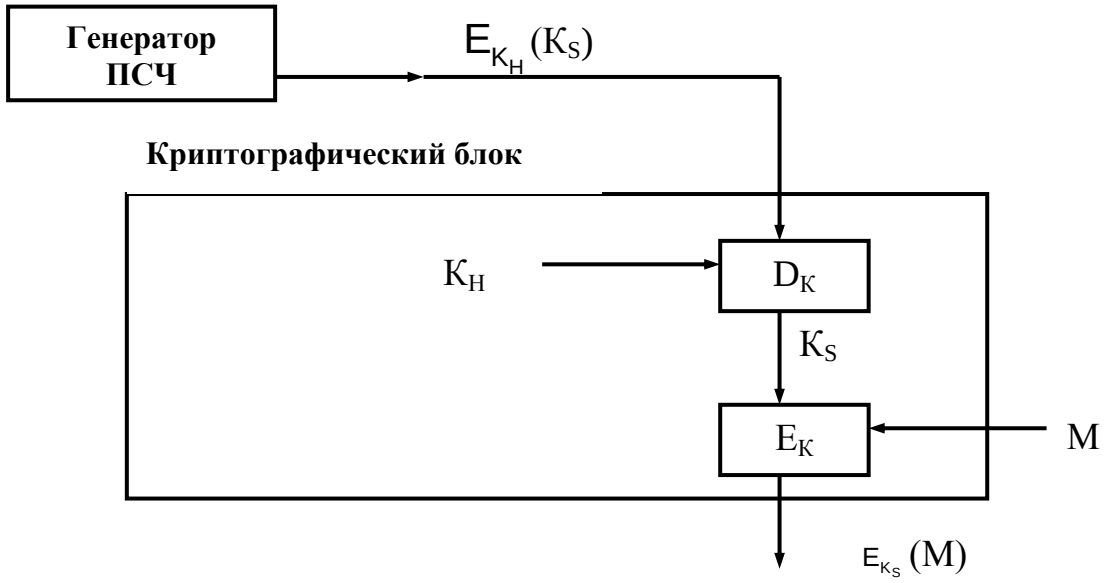


Рисунок 7.3 – Схема защиты сеансового ключа K_S

Очень важным условием безопасности информации является периодическое обновление ключевой информации в сети. При этом должны переназначаться как рабочие ключи, так и мастер-ключи. В особо ответственных сетях обновление ключевой информации (сеансовых ключей) желательно делать ежедневно. Вопрос обновления ключевой информации тесно связан с третьим элементом управления ключами – распределением ключей.

7.3. Распределение ключей

Распределение ключей – самый ответственный процесс в управлении ключами. К нему предъявляются следующие требования:

1) оперативность и точность распределения;
2) скрытность распределяемых ключей.

Распределение ключей между пользователями компьютерной сети реализуется двумя способами [55]:

1) использованием одного или нескольких центров распределения ключей;

2) прямым обменом сеансовыми ключами между пользователями сети.

Недостаток первого подхода состоит в том, что центру распределения ключей известно, кому и какие ключи распределены, и это позволяет читать все сообщения, передаваемые по сети. Возможные злоупотребления существенно влияют на защиту. При втором подходе проблема состоит в том, чтобы надежно удостоверить подлинность субъектов сети.

В обоих случаях должна быть обеспечена подлинность сеанса связи. Это можно осуществить, используя механизм запроса-ответа или механизм отметки времени.

Механизм запроса-ответа заключается в следующем. Пользователь A включает в посылаемое сообщение (запрос) для пользователя B непредсказуемый элемент (например, случайное число). При ответе пользователь B должен выполнить некоторую операцию с этим элементом (например, добавить единицу), что невозможно осуществить заранее, поскольку неизвестно, какое случайное число придет в запросе. После получения ре-

зультата действий пользователя B (ответ) пользователь A может быть уверен, что сеанс является подлинным.

Механизм отметки времени предполагает фиксацию времени для каждого сообщения. Это позволяет каждому субъекту сети определить, насколько старо пришедшее сообщение, и отвергнуть его, если появится сомнение в его подлинности. При использовании отметок времени необходимо установить допустимый временной интервал задержки.

В обоих случаях для защиты элемента контроля используют шифрование, чтобы быть уверенным, что ответ отправлен не злоумышленником и не изменен штемпель отметки времени.

Задача распределения ключей сводится к построению протокола распределения ключей, обеспечивающего:

- взаимное подтверждение подлинности участников сеанса;
- подтверждение достоверности сеанса механизмом запроса-ответа или отметки времени;
- использование минимального числа сообщений при обмене ключами;
- возможность исключения злоупотреблений со стороны центра распределения ключей (вплоть до отказа от него).

В основу решения задачи распределения ключей целесообразно положить принцип отделения процедуры подтверждения подлинности партнеров от процедуры собственно распределения ключей. Цель такого подхода состоит в создании метода, при котором после установления подлинности участники сами формируют сеансовый ключ без участия центра распределения ключей с тем, чтобы распределитель ключей не имел возможности выявить содержание сообщений.

7.3.1. Распределение ключей с участием центра распределения ключей

При распределении ключей между участниками предстоящего информационного обмена должна быть гарантирована подлинность сеанса связи. Для взаимной проверки подлинности партнеров приемлема *модель рукопожатия*. В этом случае ни один из участников не будет получать никакой секретной информации во время процедуры установления подлинности [55].

Взаимное установление подлинности гарантирует вызов нужного субъекта с высокой степенью уверенности, что связь установлена с требуемым адресатом и никаких попыток подмены не было. Реальная процедура организации соединения между участниками информационного обмена включает как этап распределения, так и этап подтверждения подлинности партнеров.

При включении в процесс распределения ключей центра распределения ключей (ЦРК) осуществляется его взаимодействие с одним или обоими участниками сеанса с целью распределения секретных или открытых ключей, предназначенных для использования в последующих сеансах связи [125].

Следующий этап – подтверждение подлинности участников – содержит обмен удостоверяющими сообщениями, чтобы иметь возможность выявить любую подмену или повтор одного из предыдущих вызовов.

Рассмотрим протоколы для симметричных криптосистем с секретными ключами и для асимметричных криптосистем с открытыми ключами. Вызывающий (исходный объект) обозначается через A, а вызываемый (объект назначения) – через B. Участники сеанса A и B имеют уникальные идентификаторы Id_A и Id_B соответственно.

7.3.2. Протокол аутентификации и распределения ключей для симметричных криптосистем

Рассмотрим в качестве примера протокол аутентификации и распределения ключей Kerberos (по-русски – Цербер). Первоначально протокол Kerberos был разработан в Массачусетском Технологическом Институте (США) для проекта Athena. Протокол Kerberos спроектирован для работы в сетях TCP/IP и предполагает участие в аутентификации и распределении ключей **третьей доверенной стороны**. Kerberos обеспечивает надежную аутентификацию в сети, разрешая законному пользователю доступ к различным машинам в сети. Протокол Kerberos **основывается на симметричной криптографии** (реализован алгоритм DES, хотя возможно применение и других симметричных криптоалгоритмов). Kerberos разделяет отдельный секретный ключ с каждым субъектом сети, и знание такого секретного ключа равносильно доказательству подлинности субъекта сети [117, 125].

Основной протокол Kerberos является вариантом протокола аутентификации и распределения ключей Нидхема-Шредера [117]. В основном **протоколе Kerberos (версия 5)** участвуют две взаимодействующие стороны A и B и доверенный сервер KS (Kerberos Server). Стороны A и B, каждая по отдельности, разделяют свой секретный ключ с сервером KS. Доверенный сервер KS выполняет роль **центра распределения ключей ЦРК**.

Пусть сторона A хочет получить сеансовый ключ для информационного обмена со стороной B.

Сторона A инициирует фазу распределения ключей, посылая по сети серверу KS идентификаторы Id_A и Id_B:

(1) $A \rightarrow KS: Id_A, Id_B.$

Сервер KS генерирует сообщение с временной отметкой T, сроком действия L, случайным сеансовым ключом K и идентификатором Id_A. Он шифрует это сообщение секретным ключом, который разделяет со стороной B.

Затем сервер KS берет временную отметку T, срок действия L, сеансовый ключ K, идентификатор Id_B стороны B и шифрует все это секретным ключом, который разделяет со стороной A. Оба эти зашифрованные сообщения он отправляет стороне A:

(2) $KS \rightarrow A: E_A(T, L, K, Id_B), E_B(T, L, K, Id_A).$

Сторона A расшифровывает первое сообщение своим секретным ключом, проверяет отметку времени T, чтобы убедиться, что это сообщение не является повторением предыдущей процедуры распределения ключей.

Затем сторона A генерирует сообщение со своим идентификатором Id_A и отметкой времени T, шифрует его сеансовым ключом K и отправляет сто-

роне B. Кроме того, A отправляет для B сообщение от KS, зашифрованное ключом стороны B:

(3) $A \rightarrow B$: $E_K (Id_A, T)$, $E_B (T, L, K, Id_A)$.

Только сторона B может расшифровать сообщения (3). Сторона B получает отметку времени T, срок действия L, сеансовый ключ K и идентификатор Id_A. Затем сторона B расшифровывает сеансовым ключом K вторую часть сообщения (3). Совпадение значений T и Id_A в двух частях сообщения подтверждают подлинность A по отношению к B.

Для взаимного подтверждения подлинности сторона B создает сообщение, состоящее из отметки времени T плюс 1, шифрует его ключом K и отправляет стороне A:

(4) $B \rightarrow A$: $E_K (T+1)$.

Если после расшифрования сообщения (4) сторона A получает ожидаемый результат, она знает, что на другом конце линии связи находится действительно B.

Этот протокол успешно работает при условии, что часы каждого участника синхронизированы с часами сервера KS. Следует отметить, что в этом протоколе необходим обмен с KS для получения сеансового ключа каждый раз, когда A желает установить связь с B. Протокол обеспечивает надежное соединение объектов A и B при условии, что ни один из ключей не скомпрометирован и сервер KS защищен.

Система Kerberos обеспечивает защиту сети от несанкционированного доступа, базируясь исключительно на программных решениях, и предполагает многократное шифрование передаваемой по сети управляющей информации.

Система Kerberos имеет структуру типа клиент-сервер и состоит из клиентских частей C, установленных на все машины сети (рабочие станции пользователей и серверы), и Kerberos-сервера KS, располагающегося на каком-либо (не обязательно выделенном) компьютере.

Kerberos-сервер, в свою очередь, можно разделить на две части: сервер идентификации AS (Authentication Server) и сервер выдачи разрешений TGS (Ticket Granting Server). Информационными ресурсами, необходимыми клиентам C, управляет сервер информационных ресурсов RS (рис.7.4).

Область действия системы Kerberos распространяется на тот участок сети, все пользователи которого зарегистрированы под своими именами и паролями в базе данных Kerberos-сервера.

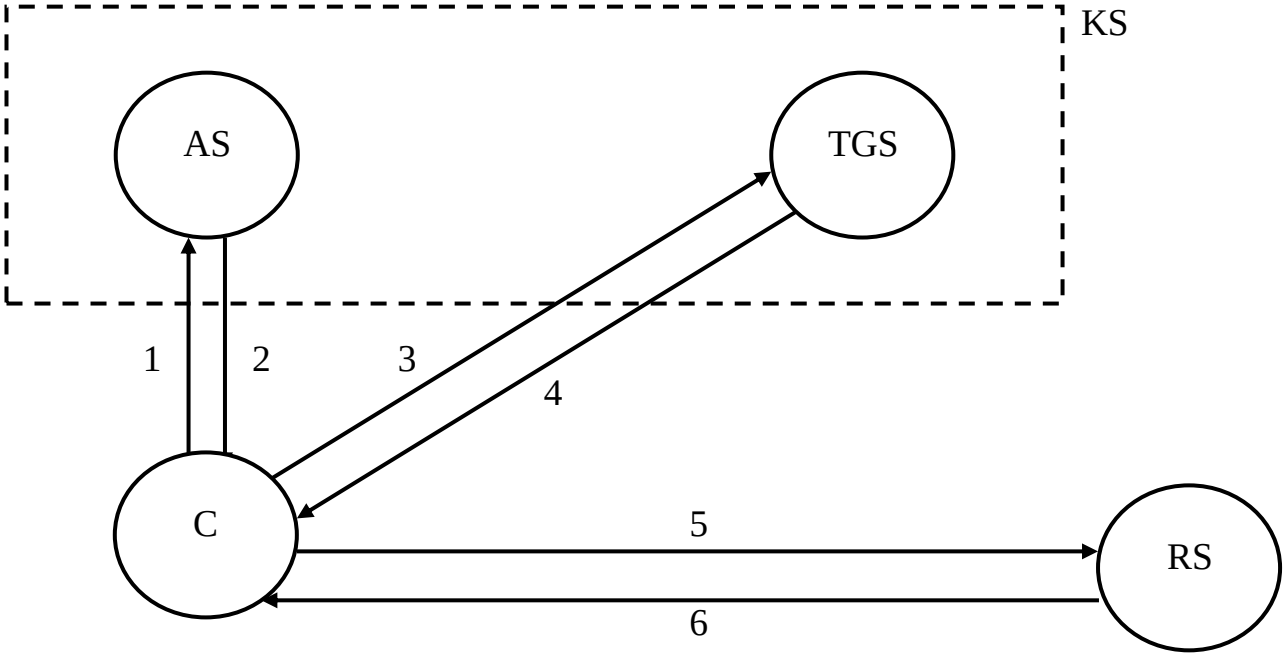


Рисунок 7.4 – Схема и шаги протокола Kerberos

Обозначения:

KS - сервер системы Kerberos;
AS - сервер идентификации;
TGS - сервер выдачи разрешений;
RS - сервер информационных ресурсов;
C - клиент системы Kerberos;
1 : C → AS : - запрос разрешить обратиться к TGS;
2 : AS → C : - разрешение обратиться к TGS;
3 : C → TGS : - запрос на допуск к RS;
4 : TGS → C : - разрешение на допуск к RS;
5 : C → RS : - запрос на получение информационного ресурса от RS;
6 : RS → C : - подтверждение подлинности сервера RS и предоставление информационного ресурса.

Укрупненно процесс идентификации и аутентификации пользователя в системе Kerberos можно списать следующим образом. Пользователь (клиент) C, желая получить доступ к ресурсу сети, направляет запрос серверу идентификации AS. Последний идентифицирует пользователя с помощью его имени и пароля и выдает разрешение на доступ к серверу выдачи разрешений TGS, который в свою очередь, по запросу клиента C разрешает использование необходимых ресурсов сети с помощью целевого сервера информационных ресурсов RS.

Данная модель взаимодействия клиента с серверами может функционировать только при условии обеспечения конфиденциальности и целостности передаваемой управляющей информации. Без строгого обеспечения информационной безопасности клиент не может отправлять серверам AS, TGS и RS свои запросы и получать разрешения на доступ к обслуживанию в сети. Чтобы избежать возможности перехвата и несанкционированного использования информации, Kerberos применяет при передаче любой управляющей информации в сети сложную систему многократного шифрования с

использованием комплекса секретных ключей (секретный ключ клиента, секретный ключ сервера, секретные сеансовые ключи, клиент-сервер) [125].

7.3.3. Протокол для асимметричных криптосистем с использованием сертификатов открытых ключей

В этом протоколе используется идея сертификатов открытых ключей.

Сертификатом открытого ключа С называется сообщение ЦРК, удостоверяющее целостность некоторого открытого ключа объекта. Например, сертификат открытого ключа для пользователя А, обозначаемый C_A, содержит отметку времени Т, идентификатор Id_A и открытый ключ $К_A$, зашифрованные секретным ключом ЦРК $k_{ЦРК}$, т.е.

$$C_A = E_{k_{ЦРК}} (T, Id_A, К_A).$$

Отметка времени Т используется для подтверждения актуальности сертификата и тем самым предотвращает повторы прежних сертификатов, которые содержат открытые ключи и для которых соответствующие секретные ключи несостоятельны.

Секретный ключ $k_{ЦРК}$ известен только менеджеру ЦРК. Открытый ключ $К_{ЦРК}$ известен участникам А и В. ЦРК поддерживает таблицу открытых ключей всех объектов сети, которые он обслуживает.

Вызывающий объект А инициирует стадию установления ключа, запрашивая у ЦРК сертификат своего открытого ключа и открытого ключа участника В:

(1) $A \rightarrow$ ЦРК : Id_A, Id_B, ´Вышлите сертификаты ключей А и В´. Здесь Id_A и Id_B – уникальные идентификаторы соответственно участников А и В.

Менеджер ЦРК отвечает сообщением

(2) ЦРК $\rightarrow$ А : $E_{k_{ЦРК}} (T, Id_A, К_A)$, $E_{k_{ЦРК}} (T, Id_B, К_B)$.

Участник А, используя открытый ключ ЦРК $К_{ЦРК}$, расшифровывает ответ ЦРК, проверяет оба сертификата. Идентификатор Id_B убеждает А, что личность вызываемого участника правильно зафиксирована в ЦРК и $К_B$ – действительно открытый ключ участника В, поскольку оба зашифрованы ключом $k_{ЦРК}$.

Хотя открытые ключи предполагаются известными всем, посредничество ЦРК позволяет подтвердить их целостность. Без такого посредничества злоумышленник может снабдить А своим открытым ключом, который А будет считать ключом участника В. Затем злоумышленник может подменить собой В и установить связь с А, и его никто не сможет выявить.

Следующий шаг протокола включает установление свя-зи А с В:

(3) $A \rightarrow B : C_A, E_{k_A} (T), E_{К_B} (r_1)$.

Здесь C_A – сертификат открытого ключа пользователя А;

$E_{k_A} (T)$ – отметка времени, зашифрованная секретным ключом участника А и являющаяся подписью участника А, поскольку никто другой не может создать такую подпись;

r_1 – случайное число, генерируемое А и используемое для обмена с В в ходе процедуры подлинности.

Если сертификат C_A и подпись A верны, то участник B уверен, что сообщение пришло от A. Часть сообщения $E_{K_B}(r_1)$ может расшифровать только B, поскольку никто другой не знает секретного ключа k_B, соответствующего открытому ключу K_B. Участник B расшифровывает значение числа r_1 и, чтобы подтвердить свою подлинность, посылает участнику A сообщение

(4) $B \rightarrow A : E_{K_A}(r_1)$.

Участник A восстанавливает значение r_1, расшифровывая это сообщение с использованием своего секретного ключа k_A. Если это ожидаемое значение r_1, то A получает подтверждение, что вызываемый участник действительно B.

Протокол, основанный на симметричном шифровании, функционирует быстрее, чем протокол, основанный на криптосистемах с открытыми ключами. Однако способность систем с открытыми ключами генерировать цифровые подписи, обеспечивающие различные функции защиты, компенсирует избыточность требуемых вычислений.

7.3.4. Прямой обмен ключами между пользователями

При использовании для информационного обмена криптосистемы с симметричным секретным ключом два пользователя, желающие обменяться криптографически защищенной информацией, должны обладать общим секретным ключом. Пользователи должны обменяться общим ключом по каналу связи безопасным образом. Если пользователи меняют ключ достаточно часто, то доставка ключа превращается в серьезную проблему.

Для решения этой проблемы можно применить два способа:

1) использование криптосистемы с открытым ключом для шифрования и передачи секретного ключа симметричной криптосистемы;

2) использование системы открытого распределения ключей Диффи–Хеллмана.

Первый способ был подробно изложен в § 4.6. Второй способ основан на применении системы открытого распределения ключей. Эта система позволяет пользователям обмениваться ключами по незащищенным каналам связи. Интересно отметить, что система открытого распределения ключей базируется на тех же принципах, что и система шифрования с открытыми клю-чами [24].

Алгоритм открытого распределения ключей Диффи–Хеллмана. Алгоритм Диффи–Хеллмана был первым алгоритмом с открытыми ключами (предложен в 1976 г.). Его безопасность обусловлена трудностью вычисления дискретных логарифмов в конечном поле, в отличие от легкости дискретного возведения в степень в том же конечном поле.

Предположим, что два пользователя A и B хотят организовать защищенный коммуникационный канал.

1. Обе стороны заранее уславливаются о модуле N (N должен быть простым числом) и примитивном элементе $g \in Z_N$,
($1 \le g \le N-1$), который образует все ненулевые элементы множества Z_N, т.е.

$$\{g, g^2, ..., g^{N-1} = 1\} = Z_N - \{0\}.$$

Эти два целых числа N и g могут не храниться в секрете. Как правило, эти значения являются общими для всех пользователей системы.

2. Затем пользователи A и B независимо друг от друга выбирают собственные секретные ключи k_A и k_B (k_A и k_B – случайные большие целые числа, которые хранятся пользователями A и B в секрете).

3. Далее пользователь A вычисляет открытый ключ

$$y_A = g^{k_A} \pmod N,$$

а пользователь B – открытый ключ

$$y_B = g^{k_B} \pmod N.$$

4. Затем стороны A и B обмениваются вычисленными значениями открытых ключей y_A и y_B по незащищенному каналу. (Мы считаем, что все данные, передаваемые по незащищенному каналу связи, могут быть перехвачены злоумышленником.)

5. Далее пользователи A и B вычисляют общий секретный ключ, используя следующие сравнения:

пользователь A: $K = (y_B)^{k_A} = (g^{k_B})^{k_A} \pmod N$;

пользователь B: $K' = (y_A)^{k_B} = (g^{k_A})^{k_B} \pmod N$.

При этом $K = K'$, так как $(g^{k_B})^{k_A} = (g^{k_A})^{k_B} \pmod N$.

Схема реализации алгоритма Диффи–Хеллмана показана на рис. 7.5.

Ключ K может использоваться в качестве общего секретного ключа (ключа шифрования ключей) в симметричной криптосистеме.

Кроме того, обе стороны A и B могут шифровать сообщения, используя следующее преобразование шифрования (типа RSA): $C = E_K(M) = M^K \pmod N$.

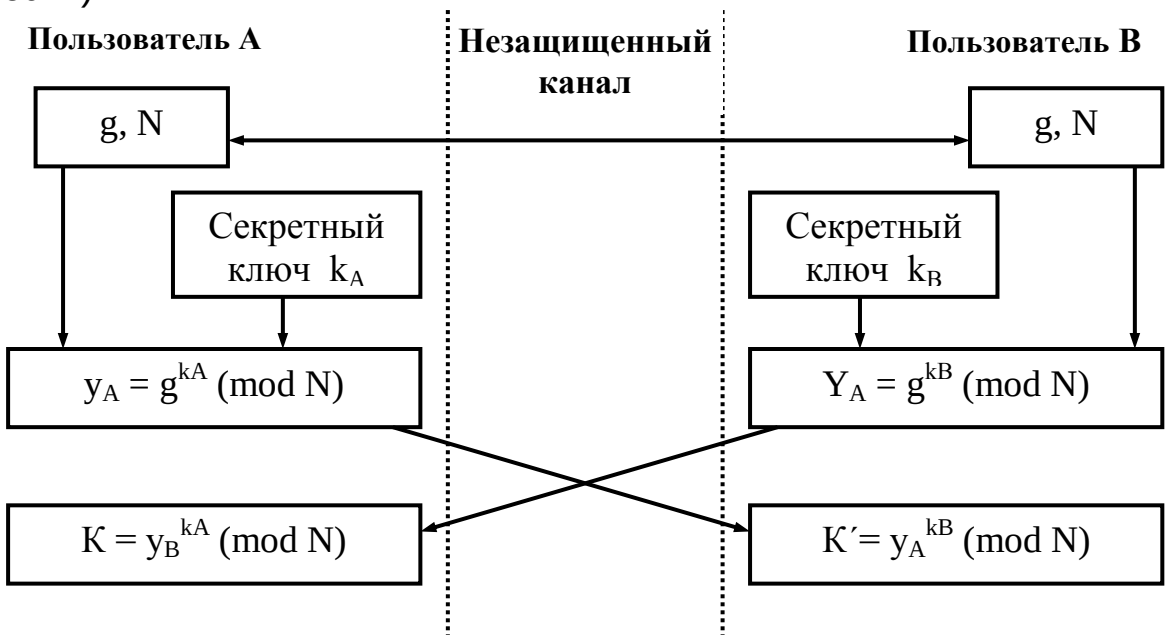


Рисунок 7.5 – Схема реализации алгоритма Диффи-Хеллмана

Для выполнения расшифрования получатель сначала находит ключ расшифрования K* с помощью сравнения

$$K * K^* \equiv 1 \pmod{N-1},$$

а затем восстанавливает сообщение

$$M = D_K(C) = C^{K^*} (\text{mod } N).$$

Злоумышленник, перехватив значения N, g, y_A и y_B, тоже хотел бы определить значение ключа K. Очевидный путь для решения этой задачи состоит в вычислении такого значения k_A по N, g, y_A, что $g^{k_A} \bmod N = y_A$ (поскольку в этом случае, вычислив k_A, можно найти $K = (y_B)^{k_A} \bmod N$). Однако нахождение k_A по N, g и y_A – задача нахождения дискретного логарифма в конечном поле, которая считается неразрешимой.

Выбор значений N и g может иметь существенное влияние на безопасность этой системы. Модуль N должен быть большим и простым числом. Число (N –1)/2 также должно быть простым числом. Число g желательно выбирать таким, чтобы оно было примитивным элементом множества Z_N. (В принципе достаточно, чтобы число g генерировало большую подгруппу мультипликативной группы по mod N.)

Алгоритм открытого распределения ключей Диффи– Хеллмана позволяет обойтись без защищенного канала для передачи ключей. Однако, работая с этим алгоритмом, необходимо иметь гарантию того, что пользователь A получил открытый ключ именно от пользователя B, и наоборот. Эта проблема решается с помощью электронной подписи, которой подписываются сообщения об открытом ключе.

Метод Диффи–Хеллмана дает возможность шифровать данные при каждом сеансе связи на новых ключах. Это позволяет не хранить секреты на дискетах или других носителях. Не следует забывать, что любое хранение секретов повышает вероятность попадания их в руки конкурентов или противника.

Преимущество метода Диффи–Хеллмана по сравнению с методом RSA заключается в том, что формирование общего секретного ключа происходит в сотни раз быстрее. В системе RSA генерация новых секретных и открытых ключей основана на генерации новых простых чисел, что занимает много времени.

Протокол SKIP управления криптоключами. Протокол SKIP (Simple Key management for Internet Protocol) может использоваться в качестве интегрирующей среды и системы управления криптоключами.

Протокол SKIP базируется на криптографии открытых ключей Диффи–Хеллмана и обладает рядом достоинств:

- обеспечивает высокую степень защиты информации;
- обеспечивает быструю смену ключей;
- поддерживает групповые рассылки защищенных сообщений;
- допускает модульную замену систем шифрования;
- вносит минимальную избыточность.

Концепция SKIP-протокола основана на организации множества двухточечных обменов (по алгоритму Диффи–Хеллмана) в компьютерной сети.

- Узел I имеет секретный ключ i ($i = k_I$) и сертифицированный открытый ключ $g^i \bmod N$.

- Подпись сертификата открытого ключа производится при помощи надежного алгоритма (ГОСТ, DSA и др.). Открытые ключи свободно распространяются центром распределения ключей из общей базы данных.
- Для каждой пары узлов I, J вычисляется совместно используемый секрет (типичная длина 1024 бита): $g^{ij} \bmod N$.
- Разделяемый ключ K_{ij} вычисляется из этого секрета путем уменьшения его до согласованной в рамках протокола длины 64…128 бит.
- Узел вычисляет ключ K_{ij} (используемый как ключ шифрования ключей) для относительно длительного применения и размещает его в защищенной памяти.

Следует отметить, что если сеть содержит n узлов, то в каждом узле должно храниться (n –1) ключей, используемых исключительно для организации связи с соответствующими узлами. Поэтому всего в сети с n узлами должно храниться $1/2 * n * (n - 1)$ различных ключей и они должны меняться время от времени (срок службы таких ключей составляет недели). Например, при n = 6 число обменов ключей составляет $1/2 * 6 * (6-1)=15$; при n=1000 число обменов ключей составит $1/2 * 1000 * (1000 - 1) \approx 500000$. Поэтому существует практическое ограничение на значение n.