ВЫСШЕЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАНИЕ

А. В. ЧЕРЕМУШКИН

КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ

ОСНОВНЫЕ СВОЙСТВА И УЯЗВИМОСТИ

Допущено
Учебно-методическим объединением по образованию
в области информационной безопасности
в качестве учебного пособия для студентов высших учебных заведений,
обучающихся по специальности «Компьютерная безопасность»

Москва
Издательский центр «Академия»
2009

УДК 003.26(075.8)
ББК 81.2-8я73
Ч-465

Рецензенты:
зав. кафедрой защиты информации и криптографии Томского государственного университета, профессор, д-р техн. наук *Г. П. Агибалов*;
профессор Московского государственного института радиотехники, электроники и автоматики (технического университета), канд. физ.-мат. наук *В. П. Зязин*;
зав. кафедрой информационной безопасности Московского государственного института электроники и математики (технического университета), канд. техн. наук *А. Б. Лось*

Черемушкин А. В.

Ч-465 Криптографические протоколы. Основные свойства и уязвимости : учеб. пособие для студ. учреждений высш. проф. образования / А. В. Черемушкин. — М. : Издательский центр «Академия», 2009. — 272 с.

ISBN 978-5-7695-5748-4

Изложены основные принципы построения криптографических протоколов, подробно описаны свойства протоколов. Рассмотрено большое число примеров, демонстрирующих типовые уязвимости и их влияние на свойства протоколов.

Для студентов учреждений высшего профессионального образования, обучающихся по специальности «Компьютерная безопасность».

УДК 003.26(075.8)
ББК 81.2-8я73

Учебное издание

Черемушкин Александр Васильевич

Криптографические протоколы. Основные свойства и уязвимости

Учебное пособие

Редактор *И. Б. Ковалева*. Технический редактор *Н. И. Горбачева*
Компьютерная верстка: *Т. А. Клименко*. Корректоры *Л. В. Гаврилина*, *А. П. Сизова*

Изд. № 101114709. Подписано в печать 14.05.2009. Формат 60 × 90/16. Гарнитура «Таймс». Бумага офсетная № 1. Печать офсетная. Усл. печ. л. 17,0. Тираж 2 000 экз. Заказ № 8819

Издательский центр «Академия». www.academia-moscow.ru
Санитарно-эпидемиологическое заключение № 77.99.02.953.Д.007496.07.04 от 20.07.2004.
117342, Москва, ул. Бутлерова, 17-Б, к. 360. Тел./факс: (495) 334-8337, 330-1092.

Отпечатано с электронных носителей издательства.
ОАО "Тверской полиграфический комбинат" 170024, г. Тверь, пр-т Ленина, 5.
Телефон: (4822) 44-52-03, 44-50-34, Телефон/факс: (4822)44-42-15
Home page - www.tverpk.ru Электронная почта (E-mail) - sales@tverpk.ru

ISBN 978-5-7695-5748-4

ПРЕДИСЛОВИЕ

Широкое развитие технологий сетевого обмена информацией, и прежде всего в сети Интернет, поставило перед многими пользователями ряд вопросов:

- действительно ли сообщение пришло от конкретного отправителя;
- не было ли оно подменено или сфабриковано кем-либо другим;
- было ли сообщение создано данным отправителем, или он просто переслал чье-либо сообщение;
- не является ли сообщение (например, об оплате какого-либо товара) повторно отправленным и т. п.

Все эти и подобные вопросы относятся к области исследований, получившей название *протоколы обеспечения безопасности*. Проблема построения протоколов обеспечения безопасности при удаленном сетевом взаимодействии привела в свою очередь к созданию новой интенсивно развивающейся дисциплины — *криптографические протоколы*. Первоначально они представляли собой надстройку над коммуникационнными протоколами, вводимую для обеспечения тех или иных аспектов безопасности. Но постепенно они интегрировались не только с коммуникационными протоколами, но и с другими прикладными протоколами и стали их непосредственной частью.

Сейчас уже создано огромное множество криптографических протоколов, предназначенных для решения самых разнообразных прикладных задач и для различных условий применения.

Проблема построения криптографических протоколов неразрывно связана с исследованиями уровня безопасности, который можно обеспечить при их использовании. Параллельно с поиском слабых мест и доработкой протоколов происходила формализация самого понятия безопасности, обеспечиваемой данным протоколом. Для

многих протоколов, первоначально казавшихся надежными, были найдены атаки, показывающие их уязвимость. В результате более чем двадцатилетних исследований появилось много теоретических методов анализа протоколов, на основе которых разработано значительное число автоматизированных средств анализа безопасности протоколов.

В настоящем пособии изложены основные понятия, связанные с криптографическими протоколами, рассмотрены основные свойства, характеризующие безопасность, и типовые уязвимости на примере известных протоколов. Приведены современные определения различных свойств, характеризующих безопасность протоколов, а также примеры атак, многие из которых найдены с помощью формальных систем для анализа безопасности протоколов. Автор не ставит задачу изложения основ построения таких формальных систем, требующего привлечения большого числа математических понятий, и ограничивается рассмотрением многочисленных примеров, иллюстрирующих влияние конструктивных особенностей протоколов на их свойства.

Содержание книги можно условно разбить на две части. Сначала рассматриваются общие понятия и основные способы обеспечения аутентификации, т. е. проверки и подтверждения подлинности, сообщений (гл. 3), сторон (гл. 4), источника (гл. 5), затем обсуждается их использование на примере различных протоколов передачи и распределения ключей (гл. 7 — 12), в которых эти свойства играют решающую роль.

Книга предназначена для первичного ознакомления с данной тематикой и может рассматриваться как введение в данную область.

Заметим также, что в пособии обсуждаются только общие особенности архитектуры и применения криптографических конструкций при построении протоколов обеспечения безопасности. В ней не рассматриваются проблемы построения конкретных прикладных протоколов. Исключением является гл. 11, в которой дается общая характеристика протокола IPsec и на его примере демонстрируется применение основных приемов построения криптографических протоколов. С вопросами применения криптографических протоколов в банковской и

коммерческой деятельности подробнее можно ознакомиться, например, в книгах [3, 4, 8] и др. Данное издание во многом основано на учебном пособии [9] и может рассматриваться как его продолжение.

Заметим, что в англоязычной литературе появилось уже много фундаментальных изданий по данной тематике (см., например, [64]). В то же время, несмотря на большое число изданий на русском языке, наблюдается значительное расхождение в трактовке основных понятий. Поэтому при формулировке определений активно использовался словарь криптографических терминов [11]. Поскольку многие термины первоначально появились в англоязычной литературе, во всех основных случаях дан их английский вариант.

СПИСОК ОБОЗНАЧЕНИЙ

$A, B, \ldots$ — участники протокола;

$\lfloor a \rfloor$ — целая часть числа a;

$\lceil a \rceil$ — минимальное целое, не меньшее числа a;

$a \bmod n$ — остаток от деления натурального числа a на натуральное число n;

cert_A — сертификат открытого ключа pk_A участника A;

$D_A(M)$ — сокращенная запись от $D_{\mathrm{sk}_A}(M)$; означает результат расшифрования сообщения M на секретном ключе sk_A участника A;

$\mathrm{diag}(a, b, \ldots, c)$ — диагональная матрица с элементами на главной диагонали $a, b, \ldots, c$ и остальными равными нулю;

$E_A(M)$ — сокращенная запись от $E_{\mathrm{pk}_A}(M)$; означает результат зашифрования сообщения M на открытом ключе pk_A участника A;

$E_k(M)$ — результат зашифрования сообщения M на ключе k для симметричной системы шифрования;

$E_k(x, y, \ldots, z) = E_k(x\|y\| \quad \|z)$ — результат зашифрования сообщения, являющегося конкатенацией $x\|y\| \quad \|z$ строк $x, y, \ldots, z$ (аналогично для E_A, D_A);

$h(M)$ — значение криптографической хеш-функции h для сообщения M;

$h_k(M)$ — значение криптографической хеш-функции, задаваемой ключом k, для сообщения M;

$h(x, y, \ldots, z) = h(x\|y\| \ldots \|z)$ — значение хеш-функции от конкатенации $x\|y\| \quad \|z$ строк $x, y, \ldots, z$ (если из контекста понятно, что речь идет именно о функции, зависящей от одной переменной);

$h_k(x, y, \ldots, z) = h_k(x\|y\| \ldots \|z)$ — значение хеш-функции, задаваемой ключом k, от конкатенации $x\|y\| \ldots \|z$ строк $x, y, \ldots, z$;

$\mathrm{KDP}(n, q)$ — схема распределения ключей на основе пересечений множеств;

k_{AB} — общий ключ участников A, B для симметричной системы шифрования;

$m\text{-}\mathrm{KDP}(n, q)$ — схема распределения ключей на основе пересечений множеств, устойчивая к компрометации m ключей;

$\binom{n}{k}$ — число сочетаний из n по k;

$(n)_k$ — число размещений из n по k;

$OA(n, m, \lambda)$ — ортогональный массив;

$(\mathrm{pk}_A, \mathrm{sk}_A)$ — пара открытый ключ/секретный ключ участника A;

$\operatorname{rang} A$ — ранг матрицы A;

$\mathrm{Sig}_k(M)$ — результат применения алгоритма формирования цифровой подписи к сообщению M при ключе k;

$\mathrm{Sig}_A(M)$ — результат применения алгоритма формирования цифровой подписи к сообщению M при секретном ключе sk_A участника A;

$\mathrm{Sig}_A(x, y, \ldots, z) = \mathrm{Sig}_A(x\|y\| \ldots \|z)$ — значение цифровой подписи под сообщением, являющимся конкатенацией $x\|y\| \ldots \|z$ строк $x, y, \ldots, z$;

$\mathrm{scert}_A = E_{k_T}(k_{AT}, A, t)$ — сертификат секретного ключа k_{AT}, общего для участника A и центра T;

$\mathrm{Ver}_k(M, s)$ — результат применения алгоритма проверки цифровой подписи s под сообщением M при ключе k;

$x\|y\| \ldots \|z$ — конкатенация строк $x, y, \ldots, z$, т. е. строка, составленная в результате последовательного приписывания их одна к другой;

$\mathbf{Z}_n$ — кольцо вычетов по модулю n;

$\overline{1, n}$ — сокращенная запись для $1, 2, \ldots, n$.

Понятие криптографического протокола

1.1. Основные определения

Понятие протокола. *Протокол* (protocol) — описание распределенного алгоритма, в процессе выполнения которого два (или более) участника последовательно выполняют определенные действия и обмениваются сообщениями. Последовательность шагов протокола группируется в циклы (раунды). В качестве участников (иначе — субъектов, сторон) протокола могут выступать не только пользователи или абоненты, но и процессы, выполняющие какую-либо функциональную роль, например клиентские и серверные приложения. Предполагается, что все участники выполняют в нем какую-либо активную роль, а пассивные наблюдатели не являются участниками протокола.

Цикл (раунд) протокола (round, pass of cryptographic protocol) — в криптографических протоколах с двумя участниками — временной интервал, в котором активен только один из участников. Другое название — *проход* (pass) *протокола.* Цикл (раунд) завершается формированием и отправлением сообщения с последующим переходом активного участника в состояние ожидания и передачей активности другому участнику. В протоколах с тремя и более участниками в синхронном случае цикл — период времени между двумя точками синхронизации. К очередной точке синхронизации каждый участник должен отправить все сообщения, которые ему предписано передать другим участникам в текущем цикле.

Шаг протокола (step of a protocol, protocol action) — конкретное законченное действие, выполняемое участником протокола во время одного цикла (раунда) протокола; например, вычисление значения некоторой функции, проверка правильности сертификата ключа, генерация случайного числа, отправка сообщения и т. п.

Коммуникационный протокол устанавливает последовательность действий участников при передаче информации или ин-

формационном обмене. Обычный коммуникационный протокол обеспечивает установку соединения/сеанса, выбор маршрута, обнаружение искажений, восстановление передаваемой информации и т. п.

Безопасность протокола выражается в обеспечении гарантий выполнения таких свойств, характеризующих безопасность, как доступность, конфиденциальность, целостность и др. Для обеспечения функций безопасности, отвечающих этим свойствам, в протоколах применяют специальные конструкции. Протокол, обеспечивающий поддержку хотя бы одной из функций безопасности, называют *защищенным*, или, точнее, *протоколом обеспечения безопасности* (security protocol).

Функции — сервисы безопасности. Для уточнения понятия безопасности протокола нам потребуется ввести несколько общих терминов. Если подняться на системный уровень, то понятие безопасности распределенных информационных систем в настоящее время конкретизируется как степень надежности реализации разнообразных функций (сервисов) безопасности. Впервые в наиболее полном виде концепция функций — сервисов безопасности изложена в международном стандарте «Базовая эталонная модель взаимодействия открытых систем. Часть 2. Архитектура безопасности», утвержденном в 1989 г. В 1991 г. этот стандарт был повторен в «Рекомендациях X.800: Архитектура безопасности взаимодействия открытых систем, для применений МККТТ*». Он содержит описание основных (базовых) функций — сервисов безопасности для случая взаимодействия двух систем, а также основных механизмов, обеспечивающих эти услуги, включая криптографические средства. Указано также их желательное расположение в эталонной семиуровневой модели взаимодействия открытых систем.

Функция — сервис безопасности (security services) — защитная функция, выполняемая подсистемой безопасности и определяемая ее целевым назначением. В соответствии со стандартом ISO 7498.2 выделено пять классов таких функций для архитектуры безопасности эталонной модели взаимодействия: аутентификация сторон и аутентификация источника данных, разграничение доступа, конфиденциальность, целостность данных и невозможность отказа от факта отправления или получения

*МККТТ — Международный консультативный комитет по телеграфии и телефонии.

Таблица 1.1

Функции — сервисы безопасности

Функция — сервис безопасности	Назначение функции
Аутентификация источника данных (data origin authentication service)	Обеспечивает возможность проверки того, что полученные данные действительно созданы конкретным источником. Данная функция не обеспечивает защиты от повторного навязывания или модификации данных
Аутентификация сторон (peer entity authentication service)	Обеспечивает возможность проверки того, что одна из сторон информационного взаимодействия действительно является той, за которую себя выдает. Применяется в целях защиты от атаки типа имитация и от атаки на протокол с повторной передачей
Конфиденциальность данных (data confidentiality service)	Обеспечивает невозможность несанкционированного получения доступа к данным или раскрытия данных
Невозможность отказа (non-repudiation service)	Обеспечивает невозможность отказа одной из сторон от факта участия в информационном обмене (полностью или в какой-либо его части)
Невозможность отказа с доказательством получения (non-repudiation service with proof of delivery)	Обеспечивает невозможность отказа получателя от факта получения сообщения
Невозможность отказа с доказательством источника (non-repudiation service with proof of origin)	Обеспечивает невозможность отказа одной из сторон от факта отправления сообщения
Целостность данных (data integrity service)	Обеспечивает возможность проверки того, что защищаемая информация не подверглась несанкционированной модификации или разрушению

Окончание табл. 1.1

Функция — сервис безопасности	Назначение функции
Обеспечение целостности соединения без восстановления (connection integrity service without recovery)	Обеспечивает возможность проверки того, что все данные, передаваемые при установленном соединении, не подверглись модификации без восстановления этих данных
Обеспечение целостности соединения с восстановлением (connection integrity service with recovery)	Обеспечивает возможность проверки того, что все данные, передаваемые при установленном соединении, не подверглись модификации с восстановлением этих данных
Разграничение доступа (access control service)	Обеспечивает невозможность несанкционированного использования ресурсов системы. Данный термин понимается в самом широком смысле. На практике решение о предоставлении доступа основывается на аутентификации сторон

сообщения, которые могут конкретизироваться для конкретных условий применения (табл. 1.1).

Для построения защищенных распределенных систем современные стандарты определяют и ряд других функций — сервисов безопасности, например туннелирование, межсетевое экранирование, сокрытие трафика и др.

Обычно используют следующее расположение сервисов для создания «эшелонированной обороны» в общей архитектуре информационной системы:

- средства выявления злоумышленной активности и контроля защищенности;
- межсетевое экранирование для защиты внешних соединений и поддержка виртуальных частных сетей (периметр безопасности);
- активный аудит и управление (для обнаружения атак);
- идентификация/аутентификация и управление доступом;
- конфиденциальность;
- пассивный аудит (для оценки последствий, поиска виновного, выяснения причин).

Понятие криптографического протокола. Возвращаясь к протоколам, заметим, что фактически большинство основных свойств безопасности реально обеспечивается только криптографическими методами и за их выполнение отвечает криптографическая подсистема. Поэтому защищенные протоколы часто отождествляют с криптографическими протоколами.

Криптографический протокол (cryptographic protocol) — протокол, предназначенный для выполнения функций криптографической системы; в процессе его выполнения участники используют криптографические алгоритмы. В данном случае под *криптографической системой* понимают систему обеспечения безопасности информации криптографическими методами. В настоящее время основными функциями криптографической системы являются обеспечение конфиденциальности, целостности, аутентификации, невозможности отказа и неотслеживаемости. В качестве подсистем она может включать системы шифрования, системы идентификации, системы имитозащиты, системы цифровой подписи и некоторые другие, а также ключевую систему, обеспечивающую работу остальных систем.

В основе выбора и построения криптографических систем лежит условие обеспечения криптографической стойкости. Под *стойкостью* криптографических систем понимают их способность противостоять атакам противника и (или) нарушителя, как правило, имеющим целью нейтрализацию одной или нескольких функций безопасности и, прежде всего, получение секретного ключа. Под *нарушителем* в данном случае понимается внутренний нарушитель, т. е. участник протокола, нарушающий предписанные протоколом действия, а под *противником* — внешний субъект (или коалиция субъектов), наблюдающий за передаваемыми сообщениями и, возможно, вмешивающийся в работу участников путем перехвата, искажения (модификации), вставки (создания новых), повтора и перенаправления сообщений, блокирования передачи в целях нарушения одной или нескольких функций — сервисов безопасности. Часто допускается, что противник может образовывать коалицию с нарушителем. Заметим, что во многих формальных методах анализа протоколов противник отождествляется с сетью (точнее — с совокупностью каналов связи), по которой проводится обмен сообщениями.

Большие подборки различных криптографических протоколов вместе с описанием их свойств и атак на них можно найти, например, в публикациях [20, 39, 54, 64].

1.2. Свойства, характеризующие безопасность протоколов

Поскольку криптографическая система может обеспечивать различные функции безопасности, для реализации которых применяют разнообразные криптографические протоколы, то и свойств, характеризующих безопасность криптографического протокола, также достаточно много. Обычно свойства протоколов, характеризующие их стойкость к различным атакам, формулируют как цели (goals) или требования к протоколам. Трактовка этих целей со временем меняется и уточняется. Наиболее полное и современное толкование этих целей дается в документах международной организации Internet Engineering Task Force (IETF).

В настоящее время в документах IETF фигурируют двадцать свойств (целей, требований) безопасности, распределенных по десяти группам [20]:

1) аутентификация (нешироковещательная):
   G1 — аутентификация субъекта;
   G2 — аутентификация сообщения;
   G3 — защита от повтора;
2) аутентификация при рассылке по многим адресам или при подключении к службе подписки/уведомления:
   G4 — неявная (скрытая) аутентификация получателя;
   G5 — аутентификация источника;
3) G6 — авторизация (доверенной третьей стороной);
4) свойства совместной генерации ключа:
   G7 — аутентификация ключа;
   G8 — подтверждение правильности ключа;
   G9 — защищенность от чтения назад;
   G10 — формирование новых ключей;
   G11 — защищенная возможность договориться о параметрах безопасности;
5) G12 — конфиденциальность;
6) анонимность:
   G13 — защита идентификаторов от прослушивания (несвязываемость);
   G14 — защита идентификаторов от других участников;
7) G15 — ограниченная защищенность от атак типа «отказ в обслуживании»;
8) G16 — инвариантность отправителя;
9) невозможность отказа от ранее совершенных действий:

G17 — подотчетность;
G18 — доказательство источника;
G19 — доказательство получателя;
10) G20 — безопасное временно́е свойство.

Приведем определения свойств безопасности согласно [20].

Аутентификация (нешироковещательная)

Аутентификация (*нешироковещательная*) (authentication unicast) — это проверка идентичности (не путать с идентификатором), заявленной участником или субъектом системы, в качестве которого может выступать одна из сторон коммуникации или источник некоторых данных. Проверяемая идентичность может быть хорошо известна (реальное имя, телефонный номер, почтовый адрес, номер страхового полиса, адрес протокола IP или адрес электронной почты); это может быть также несвязываемый идентификатор (подобный псевдониму). Верификация достигается представлением аутентификационной информации (документов), которая подтверждает связь субъекта и идентификатора. Аутентификация обычно подразделяется на аутентификацию сторон и аутентификацию сообщений (или данных). Основное отличие между этими двумя типами заключается в том, что аутентификация сообщений не обеспечивает гарантии своевременности (аутентифицированное сообщение может быть старым), в то время как аутентификация субъекта подразумевает передачу сообщений с одновременной проверкой в текущем сеансе выполнения протокола. Аутентификация обычно является *односторонней* (участник A аутентифицирует участника B). *Взаимная аутентификация* осуществляется в обоих направлениях.

G1 — аутентификация субъекта (аутентификация сторон) (peer entity authentication). Это проверка с подтверждением подлинности одной из сторон наличия или полномочий (посредством представленных доказательств и (или) документов) идентичности второй стороны, участвующей в выполнении протокола, а также того, что она действительно принимает участие в выполнении текущего сеанса протокола. Обычно такая проверка осуществляется посредством набора данных, который мог быть сгенерирован только вторым участником (например, как отклик на запрос). Таким образом, обычно аутентификация субъекта предразумевает, что некоторые данные могут быть безошибочно возвращены некоторому субъекту, что предполага-

ет аутентификацию источника данных (data origin authentication).

G2 — аутентификация сообщения (message authentication). Заключается в обеспечении аутентификации источника данных и целостности передаваемого сообщения. Аутентификация источника данных (data origin authentication) означает, что протокол должен обеспечивать средства гарантии того, что полученное сообщение или часть данных были созданы некоторым участником в некоторый (как правило, неопределенный) момент времени, предшествующий получению сообщения, и что эти данные не были искажены или подделаны, но без предоставления гарантий однозначности и своевременности. Поскольку уверенность в том, что данные были созданы некоторым участником без гарантии того, что они не были модифицированы, не представляет практического интереса, обычно полагают, что требование аутентификации сообщения влечет требование его целостности. Только немногие протоколы Internet обеспечивают аутентификацию источника данных без обеспечения аутентификации субъекта.

G3 — защита от повтора (replay protection). Это гарантирование одним из участников того, что аутентифицированное сообщение не является старым. В зависимости от контекста может иметь разный смысл: сообщение было сгенерировано в данном сеансе протокола, либо сообщение было сгенерировано в течение известного промежутка времени, либо сообщение не было принято ранее.

Аутентификация при рассылке по многим адресам или при подключении к службе подписки/уведомления

Аутентификация при рассылке по многим адресам или при подключении к службе подписки/уведомления (authentication in multicast or via a subscribe/notify service) — это требование аутентификации для групп участников с одним источником и большим числом потенциальных получателей (широковещательная передача) либо источник и служба, которые отправляют информацию подключенным (и авторизованным) пользователям.

Перечислим основные требования для таких систем.

G4 — неявная (скрытая) аутентификация получателя (implicit destination authentication). Протокол должен

предоставлять средства для гарантии того, что отправленное сообщение будет прочитано только теми сторонами, которым оно направлялось, т. е. только законные авторизованные участники получат доступ к текущей информации, широковещательному сообщению или групповой коммуникации (передаче). Это относится и к группам с очень динамичным составом участников.

G5 — аутентификация источника (source authentication). Законные группы участников должны быть способны аутентифицировать источник и содержание информации или групповой коммуникации. Это относится к случаям, когда группы участников не доверяют друг другу.

Авторизация (доверенной третьей стороной)

Свойство G6 — *авторизация* (authorization) (*доверенной третьей стороной*) заключается в том, что в некоторых протоколах доверенная третья сторона (ТЗР) представляет одного субъекта B другому субъекту A. В результате этого субъект A получает гарантии того, что субъект B удостоверен (trusted) с помощью ТЗР и авторизован (наделен правами) в требуемом для протокола смысле. Когда протокол выполняется тремя участниками A, B, ТЗР, то субъект A, вероятно, не может иметь доступ к контрольному списку или другим механизмам для авторизации B (потому что имя B не известно субъекту A либо может являться псевдонимом), но субъект A получает гарантии того, что субъект B авторизован ТЗР.

Свойства совместной генерации ключа

Перечислим *основные свойства совместной генерации ключа* (key agreement properties).

G7 — аутентификация ключа (key authentication). Это свойство предполагает, что один из участников получает подтверждение того, что никакой другой участник кроме заранее определенного второго участника (и, возможно, других доверенных участников) не может получить доступа ни к одному секретному ключу. (Заметим, что в такой формулировке это свойство одностороннее, т. е. касается только одного из участников.)

G8 — подтверждение правильности ключа (key confirmation, key proof of possession). Один из участников получает подтверждение того, что второй участник (возможно, неопределенный) действительно обладает конкретным секретным клю-

чом (либо имеет доступ ко всем ключевым материалам, необходимым для его вычисления).

G9 — защищенность от чтения назад/совершенная секретность в будущем (perfect forward secrecy — PFS). Протокол обладает этим свойством, если компрометация долговременных ключей не приводит к компрометации старых сеансовых ключей.

G10 — формирование новых ключей (fresh key derivation). Протокол использует динамическое распределение ключей в целях получения новых (fresh) ключей. (Ранее использовался термин «back traffic protection».)

G11 — защищенная возможность договориться о параметрах безопасности (secure capabilities negotiation, resistance against downgrading and negotiation attacks). Если протокол открытого распределения ключей дает возможность сторонам договариваться о параметрах безопасности (таких как идентификаторы защищенной ассоциации, длина ключа и наборы алгоритмов шифрования), то это свойство важно для подтверждения того, что заявленные свойства и параметры, о которых договорились участники протокола, не были подменены противником.

Конфиденциальность

Свойство G12 — *конфиденциальность* (confidentiality, secrecy) состоит в том, что специфический набор данных (обычно посылаемый или получаемый как часть «защищенного» сообщения, а также сформированный на основе данных, полученных в результате обмена) не станет доступным или раскрытым для неавторизованных субъектов или процессов, а останется неизвестным противнику. Мы принимаем соглашение, что секретность сеансового ключа, сгенерированного в результате процедуры открытого распределения ключей, рассматривается не здесь, а выше при описании свойства G7. Заметим, что секретность долговременного ключа, используемого в протоколе, не рассматривается как целевое свойство безопасности протокола, а относится к исходным предположениям.

Анонимность

Многие протоколы не обеспечивают свойства *анонимности* (anonymity), поскольку, как правило, сторона хочет знать, с кем

она взаимодействует при формировании ключа. Однако некоторые протоколы позволяют скрывать идентификаторы.

G13 — защита идентификаторов от прослушивания (несвязываемость) (identity protection against eavesdroppers). Атакующий, осуществляющий перехват сообщений, не должен иметь возможность связать сообщения одного из участников с самим участником.

G14 — защита идентификаторов от других участников (identity protection against peer). В процессе взаимодействия другие участники не должны иметь возможности связать сообщения конкретного участника с самим участником, а только с несвязанным с ним псевдонимом или частным идентификатором.

Ограниченная защищенность от атак типа «отказ в обслуживании»

Свойство G15 — *ограниченная защищенность от атак типа «отказ в обслуживании»* (limited denial of service (DoS) resistance) состоит в том, что трудно обеспечить защищенность от DoS-атак.

Одна из причин этого заключается в том, что протокол может быть объектом DoS-атак по различным причинам; наиболее распространенная состоит в том, что протокол потребляет слишком много ресурсов (памяти, вычислительной мощности), перед тем как стороны аутентифицируют друг друга. Однако есть много других причин, среди них та, что протокол может быть уязвимым:

- к DoS-атакам в выделении памяти;
- DoS-атакам в вычислительной мощности;
- bombing attacks от третьих сторон (когда один или несколько хостов посылают жертве огромное число пакетов).

Инвариантность отправителя

Свойство G16 — *инвариантность отправителя* (sender invariance) состоит в том, что участник получает гарантии того, что источник сообщений не изменялся с начала выполнения сеанса, хотя само установление идентичности этого источника для получателя не важно.

Невозможность отказа от ранее совершенных действий

Невозможность отказа от ранее совершенных действий (non-repudiation) предполагает предотвращение того, что участник откажется от реально совершенного им действия.

G17 — подотчетность (accountability). Это свойство системы (включая все системные ресурсы), состоящее в том, что предоставляются гарантии того, что действия системных субъектов могут быть однозначно прослежены теми субъектами, кто отвечает за эти действия.

G18 — доказательство источника (proof of origin). Это бесспорное доказательство (очевидность) того, что сообщение было отправлено.

G19 — доказательство получения (proof of delivery). Это бесспорное доказательство (очевидность) того, что сообщение было получено.

Безопасное временно́е свойство

Свойство G20 — *безопасное временно́е свойство* (safety temporal property) состоит в том, что используя два оператора временно́й логики (линейной временно́й логики), а именно операторы «Всегда» и «Когда-то в прошлом», можно формализовать свойства вида: «Для любого достижимого состояния, имеющего свойство p, ранее было состояние со свойством q».

Свойствами этого типа, которые непосредственно не видны из стандартных свойств безопасности, являются:

- если пользователь должен платить за ресурс, то он должен иметь доступ к ресурсу;
- если хост получает большое число пакетов в режиме потока, то он должен их ранее запросить.

Заметим также, что все свойства безопасности, рассмотренные выше, могут быть выражены в форме безопасного временно́го свойства G20.

Новые свойства безопасности

В последнее время в некоторых документах IETF drafts (в частности, в протоколе Extensible Authentication Protocol (EAP) Working Group) обсуждаются новые свойства безопасности.

Формирование сеанса (session formation). Протокол обеспечивает формирование сеанса, если он связывает каждый конкретный сеанс протокола с уникальным значением, обобщенным идентификатором сеанса, который обычно представляет собой набор случайных данных и идентификаторов. Этот обобщенный идентификатор сеанса позволяет отличить каждый конкретный сеанс протокола от других сеансов этого протокола. После выполнения начальной части протокола все сообщения содержат обобщенный идентификатор сеанса (не обязательно в открытом виде).

Последовательное представление (consistent view). После выполнения какой-либо части протокола все участники данного сеанса протокола имеют одинаковое представление обо всех участниках этого сеанса и выполняемых ими ролях, а также о состоянии выполнения протокола.

Именование ключей (key naming). Для того чтобы иметь гарантии от неправильного использования ключевых материалов в каждой защищенной ассоциации протокола обмена, криптогрпфический протокол должен использовать явные имена ключей и соответствующий контекст, позволяющий информировать проверяющего о порядке использования этого ключевого материала.

Если протокол обеспечивает доказательство владения ключами, то протокол должен применять явные имена ключей, используемые в течение доказательства владения, чтобы предотвратить ситуацию, когда используется более одного набора материалов для выполнения протокола обмена.

В настоящее время предпринимаются попытки формализовать эти свойства для последующего использования их в автоматизированных системах анализа протоколов, но пока до конца неясно, как это можно сделать.

Имеются и другие свойства, обсуждаемые в документах IETF, например:

- криптографическое разделение ключей (cryptographic separation of keys);
- возможность договориться о выборе алгоритмов шифрования (cipher suite negotiation);
- устойчивость к атаке со словарем (dictionary attack resistance);
- криптографическое связывание (cryptographic binding);
- поддержка быстрого восстановления соединения (support for fast reconnect);

Таблица 1.2

Свойства безопасности, характеризующие основные протоколы

Протокол	Свойство G														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
EAP-IKEv2	×	×	×			×	×			×					×
EKE	×	×										×			
IKE	×	×	×				×		×	×	×		×	×	×
IKEv2	×	×	×				×		×	×	×				×
DHCP-IPSec-tunnel	×	×										×			
Kerberos	×	×	×			×	×			×					
SSH	×	×	×				×			×	×				
TLS	×	×	×				×			×	×		×		
TLS-v1.1	×	×	×				×			×	×		×		
TLS-SRP	×	×	×				×			×	×		×		
TLS-sharedkeys	×	×	×				×			×	×		×		
SET	×	×	×										×		

- подтверждение успешного завершения и индикация ошибок (acknowledged success and failure indications);
- независимость сеансов (session independence);
- защищенность от атаки «противник в середине» (man-in-the-middle attack resistance);
- взаимная живучесть (peer liveness).

Два последних свойства в действительности не являются новыми свойствами безопасности, они непосредственно связаны со свойствами аутентификации, рассмотренными выше.

Приведем в качестве примера свойства безопасности, которыми обладают некоторые известные криптографические протоколы (табл. 1.2).

1.3. Виды криптографических протоколов

Криптографические протоколы можно классифицировать различными способами, например:

по числу участников

- двусторонний;
- трехсторонний;
- многосторонний;

по числу передававаемых сообщений:

- интерактивный (есть взаимный обмен сообщениями);
- неинтерактивный (только однократная передача); неинтерактивные протоколы часто называют *схемами*.

Наиболее содержательным является подход, при котором в основе классификации лежит функциональное (целевое) назначение протокола. Если предположить, что протокол выполняет одну функцию, то получаем следующие типы протоколов:

- протокол обеспечения целостности сообщений:
  с аутентификацией источника;
  без аутентификации источника;
- протокол (схема) цифровой подписи:
  протокол индивидуальной/групповой цифровой подписи;
  с восстановлением/без восстановления сообщения;
  протокол цифровой подписи вслепую;
  протокол конфиденциальной цифровой подписи;
  протокол цифровой подписи с доказуемостью подделки;
- протокол идентификации (аутентификации участников):
  односторонней аутентификации;
  двусторонней (взаимной) аутентификации;
- конфиденциальная передача:
  обычный обмен сообщениями;
  широковещательная/циркулярная передача;
  честный обмен секретами;
  забывающая передача;
  протокол привязки к биту (строке);
- протокол распределения ключей:
  протокол (схема) предварительного распределения ключей;
  протокол передачи ключа (обмена ключами);
  протокол совместной выработки ключа (открытого распределения ключей);
  протокол парный/групповой;
  протокол (схема) разделения секрета;
  протокол (распределения ключей) телеконференции.

Свойства большинства из перечисленных выше протоколов будут рассмотрены далее, а пока поясним некоторые термины.

Протокол аутентификации сообщений (message authentication protocol) — криптографический протокол, предназначенный для обеспечения целостности сообщений, передаваемых от одного участника к другому; под целостностью понимается гарантируемая получателю возможность удостовериться, что сообщение поступило от заявленного отправителя и в неискаженном виде.

Протокол (или схема) идентификации (identification protocol) — протокол аутентификации сторон, участвующих во взаимодействии и не доверяющих друг другу.

Схема цифровой подписи в простейшем случае состоит из двух алгоритмов: формирования и проверки подписи. Если требуется скрыть содержание документа от подписывающего участника (подпись вслепую) либо если без участия подписывающего лица процедура проверки подписи невозможна (конфиденциальная цифровая подпись), требуются специальные протоколы формирования и проверки подписи.

Задача распределения ключей (key distribution protocol), необходимых для функционирования криптографической системы, является одной из центральных в криптографии, поскольку в большинстве случаев стойкость криптографической системы основана на недоступности ключей. Поэтому *протоколы распределения ключей* аккумулируют большинство свойств, предъявляемых к криптографическим протоколам. Они должны обеспечивать не только конфиденциальность, но и аутентификацию всех аспектов взаимодействия: ключа, источника, отправителя и получателя, времени сеанса, чтобы гарантировать, что никакой другой участник, кроме заранее определенного второго участника (и, возможно, других доверенных участников), не мог получить доступа ни к одному секретному ключу.

Протокол обмена секретами (secret exchange protocol) — криптографический протокол с двумя участниками, в котором обмен секретами организован таким образом, чтобы в случае его прерывания (по любой причине) знания участников о секретах друг друга были приблизительно одинаковыми.

Протокол привязки к биту (bit commitment protocol) — криптографический протокол с двумя участниками (отправителем и получателем), посредством которого отправитель передает получателю бит информации (битовое обязательство) таким образом, что выполняются два условия: 1) после передачи бита получателю (так называемого этапа привязки) отправитель уже не может изменить его значение; 2) получатель не может само-

стоятельно определить значение бита и узнает его только после выполнения отправителем так называемого этапа раскрытия.

Протокол подбрасывания (по телефону) монеты (coin flipping (by telephone) protocol) — криптографический протокол, позволяющий двум не доверяющим друг другу участникам сгенерировать общий случайный бит. Главное свойство таких протоколов состоит в том, что если хотя бы один из участников честный, то сгенерированный бит будет случайным независимо от действий другого участника. Имеются обобщения на случай конечных битовых строк, а также на случай произвольного числа участников.

Интересный вид протоколов распределения ключей составляют так называемые *групповые протоколы* (group-oriented protocol), в которых ключи распределяются между группами участников. Если все группы, имеющие на это право, формируют одинаковые ключи, то такой протокол называют *протоколом разделения секрета* (secret sharing protocol). Если же у различных групп должны быть разные ключи, то это — протокол телеконференции.

В *протоколах групповой подписи* (group signature protocol) предполагается одновременное участие заранее определенной группы участников, причем в случае отсутствия хотя бы одного участника группы формирование подписи невозможно.

Приведенную выше классификацию можно уточнить, если при этом отдельно рассматривать примитивные и прикладные криптографические протоколы.

Примитивный криптографический протокол (primitive cryptographic protocol) — это криптографический протокол, который не имеет самостоятельного прикладного значения, но используется как базовый компонент при построении прикладных криптографических протоколов. Как правило, он решает какую-либо одну абстрактную задачу. Примеры — протокол обмена секретами, протокол привязки к биту, протокол подбрасывания монеты (по телефону).

Прикладной криптографический протокол (application cryptographic protocol) предназначен для решения практических задач обеспечения функций — сервисов безопасности с помощью криптографических систем. Следует заметить, что прикладные протоколы, как правило, обеспечивают не одну, а сразу несколько функций безопасности. Более того, такие протоколы, как IPsec, на самом деле, являются большими семействами различ-

ных протоколов, включающими много разных вариантов для различных ситуаций и условий применения.

Классификацию криптографических протоколов можно проводить и по другим признакам:

- по типу используемых криптографических систем:
  - протокол на основе симметричных криптографических систем;
  - протокол на основе асимметричных криптографических систем;
  - смешанный протокол;
- по способу функционирования:
  - интерактивный/неинтерактивный протокол;
  - одно-, двух-, трех-, четырехшаговый протокол;
  - протокол с арбитром (протокол с посредником);
  - двусторонний/с доверенной третьей стороной (с центром доверия) протокол;
  - работающий в реальном времени (on-line)/в отложенном режиме (off-line) и т. п.

Протокол с арбитром, или протокол с посредником (arbitrated protocol) — криптографический протокол, в котором для разрешения споров между участниками требуется арбитраж.

Если обмен сообщениями осуществляется с участием специально выделенного участника, обладающего доверием других участников, то говорят *о протоколах с доверенной третьей стороной*, или о *протоколах с центром доверия*. Если участники вынуждены обращаться к центру постоянно в каждом сеансе протокола, то говорят о работе в реальном времени (on-line). Если же достаточно заранее перед началом сеанса один раз обратиться к центру, то говорят о работе в отложенном режиме (off-line). Иногда выделяют еще режим in-line, когда весь обмен сообщениями идет через центр доверия.

Особый класс протоколов составляют *протоколы доказательства*:

- интерактивное/неинтерактивное доказательство;
- протокол доказательства знания (какого-либо факта);
- протокол доказательства умения решать какую-то задачу.

Доказательство интерактивное (interactive proof) осуществляется путем выполнения протокола двумя участниками — доказывающим и проверяющим; в процессе работы участники обмениваются сообщениями, обычно зависящими от случайных чисел, которые могут содержаться в секрете. Цель доказывающего — убедить проверяющего в истинности некоторого

утверждения. Проверяющий либо принимает, либо отвергает доказательство. В отличие от обычного математического понятия «доказательство» в данном случае доказательство носит не абсолютный, а вероятностный характер и характеризуется двумя вероятностями. Если доказываемое утверждение верно, то доказательство должно быть верным с вероятностью, стремящейся к единице при увеличении числа повторений протокола. Если же доказываемое утверждение ложно, то при увеличении числа повторений протокола вероятность правильности доказательства должна стремиться к нулю.

Доказательство знания (proof of knowledge) — доказательство интерактивное, при котором доказывающий убеждает проверяющего в том, что он владеет секретной информацией, не раскрывая ее. К категории доказательства знания относятся протоколы идентификации. Центральным свойством таких протоколов является *разглашение нулевое* (zero-knowledge property) — свойство протокола доказательства знания, обеспечивающее такое его выполнение, что никакая информация о доказываемом утверждении, кроме факта его истинности, не может быть получена нечестным проверяющим из переданных сообщений за время, полиномиально зависящее от суммарной длины этих сообщений.

Если число примитивных криптографических протоколов сравнительно невелико, то прикладных протоколов имеется трудноообозримое множество. Одни из них уже стали стандартами и широко применяются, другие — имеют очень ограниченное распространение. Процесс создания, доработки и обновления протоколов идет постоянно. Он обусловлен не только появлением новых областей применения, многообразием возможных практических ситуаций и выработкой новых требований к протоколам, но и постоянными усилиями по анализу их безопасности, в результате чего обнаруживаются все новые и новые уязвимости. Поскольку описание прикладных протоколов не является задачей данной книги, ограничимся их перечислением, показывающим пример возможной классификации криптографических протоколов по области применения:

- система электронного обмена данными, включающая протоколы электронного документооборота;
- система электронных платежей:

  протоколы систем с виртуальными деньгами, регулирующие, например, удаленный платеж по электронным чекам;

протоколы систем с электронными деньгами:
протокол удаленного платежа по кредитным картам;
протокол электронного денежного перевода;
протокол электронного дебетового поручения;
протоколы систем с цифровыми деньгами:
протокол снятия со счета цифровой наличности;
платежный протокол с цифровыми деньгами;
протокол депозита для сдачи цифровых денег в банк;
протокол с идентификацией повторной траты монеты;

- система электронной коммерции:
  протокол подписания контракта;
  протокол сертифицированной электронной почты;
  протокол электронного аукциона;
- поддержка правовых отношений, включающая протоколы голосования (электронные выборы);
- игровые протоколы:
  протокол бросания жребия (по телефону);
  протокол игры в покер (по телефону) и т. д.

Сделаем некоторые пояснения.

Система электронных платежей (electronic cash system, e-cash system) — система осуществления транзакций и взаиморасчетов между сторонами в электронной (безбумажной) форме. Различают: системы перевода денежных средств электронные с использованием электронных систем обмена данными (межбанковские переводы) и системы, в которых используются дематериализованные денежные аналоги. Последние различаются по степени дематериализации: деньги электронные, деньги виртуальные и деньги цифровые. Система электронных платежей состоит из набора протоколов, из которых основными являются протоколы, реализующие транзакцию снятия со счета, транзакцию платежа и транзакцию депозита.

Протокол подписания контракта (contract signing protocol) позволяет двум участникам путем обмена сообщениями по каналам связи подписать контракт, существующий только в электронной форме. Основное требование к криптографической стойкости таково: при любом прерывании выполнения протокола шансы каждого из участников получить контракт, подписанный другим, и при этом не подписаться самому, ничтожно малы.

Протокол сертифицированной электронной почты предполагает обязательное получение участниками уведомлений о факте отправки и получения сообщения (не зависящих от его содержания), которое в дальнейшем позволит доказать отправите-

лю факт получения сообщения, а получателю — что оно не было отправлено отправителем в случае, если он попытается утверждать обратное.

Протокол голосования (election scheme, voting scheme, voting protocol) позволяет проводить процедуру голосования, в которой избирательные бюллетени существуют только в электронной форме. Является криптографическим протоколом, так как обеспечивает тайный характер голосования. Основное свойство — универсальная проверяемость, т. е. предоставление возможности всякому желающему, включая сторонних наблюдателей, в любой момент времени проверить правильность подсчета голосов.

1.4. Основные атаки на безопасность протоколов

Под *атакой на протокол* понимают попытку проведения анализа сообщений протокола и (или) выполнения не предусмотренных протоколом действий в целях нарушения работы протокола и (или) получения информации, составляющей секрет его участников.

Атака считается успешной, если нарушено хотя бы одно из заявленных свойств, характеризующих безопасность протокола. Например, если в результате вмешательства в ход выполнения протокола противнику не удалось извлечь какой-либо ценной информации, но он смог ввести в заблуждение одного из участников так, что тот остался в полной уверенности, что он взаимодействовал с законным участником, а не с противником, то тем самым произошло нарушение свойства аутентификации сторон.

Заметим, что стойкость протокола к атакам зависит от многих факторов: от надежности криптографических механизмов, правильности реализации, точности выполнения порядка предписанных действий участниками протокола и др. В данном пособии основное внимание будет сконцентрировано на слабостях протоколов, обусловленных способом формирования и порядком отправки сообщений участниками, т. е. ошибках, допущенных при синтезе самих протоколов, представляющих собой предписания, определяющие порядок действий всех участников. При этом обычно будем исходить из предположения, что выбранные криптографические механизмы являются надежными (perfect encryption hypothesis), реализации — правильными, а са-

ми участники, если не оговорено противное, честно выполняют все предписания.

При классификации атак на безопасность протоколов можно применять различные подходы. Например, аналогично классификации самих протоколов, проведенной в подразд. 1.3, можно опираться на следующие признаки:

- способ воздействия (активные, пассивные атаки);
- цели (на какое свойство, характеризующее безопасность протокола, осуществляется атака);
- область применения (универсальные атаки; атаки, зависящие от способа реализации);
- класс протоколов (атаки на протоколы идентификации, передачи или распределения ключей, цифровой подписи и т. п.);
- применяемый математический аппарат (атаки без дополнительных средств, например используя только повтор, отражение или задержку сообщений; статистические атаки — с анализом статистики прерванных и доведенных до завершения сеансов; алгебраические атаки — на основе анализа алгебраических закономерностей и др.);
- способ применения (число необходимых сеансов; можно ли ограничиться старыми или нужны вновь открытые параллельные сеансы);
- требуемые ресурсы (ограниченные или неограниченные вычислительные возможности);
- наличие и характер дополнительной информации (число сеансов с известными сообщениями; сеансовые или долговременные ключи и т. п.);

Не углубляясь в детали, приведем перечень наиболее широко известных атак на криптографические протоколы и перечислим способы защиты от них.

1. *Подмена* (impersonation) — попытка подменить одного пользователя другим. Нарушитель, выступая от имени одной из сторон и полностью имитируя ее действия, получает в ответ сообщения определенного формата, необходимые для подделки отдельных шагов протокола. Успех атаки определяется тем, насколько протокол устойчив к подобным подменам. Методы противодействия хранятся в тайне от противника информации, определяющей алгоритм идентификации. Помимо этого можно использовать различные форматы сообщений, передаваемых на разных шагах протокола, а также вставлять в них специальные идентификационные метки и номера сообщений.

В протоколах с использованием третьей стороны возможны атаки, основанные на подмене доверенного сервера. Например, одна из сторон, имеющая доверительные отношения с сервером, выступает от его имени, подменяет его трафик обмена с другими сторонами и в результате получает возможность раскрывать значения генерируемых центром ключей. Эта атака может быть успешной для протоколов, в которых аутентификация при доступе к серверу основана только на идентификаторах сторон и случайных числах, генерируемых при каждом взаимодействии. Для защиты от таких атак применяют средства привязки ключей не к одной, а к обеим взаимодействующим сторонам путем передачи обоих идентификаторов в зашифрованном виде.

2. *Повторное навязывание сообщения* (replay attack) — повторное использование ранее переданного в текущем или предыдущем сеансе сообщения или какой-либо его части в текущем сеансе протокола. Например, повторная передача информации ранее проведенного протокола идентификации может привести к повторной успешной идентификации того же самого или другого пользователя. В протоколах передачи ключей данная атака часто применяется для повторного навязывания уже использованного ранее сеансового ключа. Другое название подобной атаки на протокол передачи ключей — *атака на основе новизны* (freshness attack).

Методы противодействия состоят в обеспечении целостности сеанса и невозможности вставки в него лишних сообщений. Для этого используют технику типа «запрос — ответ», вставку в передаваемые сообщения временны́х меток, случайных чисел или возрастающих последовательностей чисел.

3. Еще один тип подобных атак, связанный с обратной передачей отправителю ранее переданных им сообщений, получил название *атака отражением* (reflection attack). Часто атаки данного типа относят к классу атак с повторным навязыванием сообщения.

Для защиты от таких атак протоколы специально делают несимметричными, включая в зашифрованные сообщения идентификаторы сторон либо изменяя процедуры так, чтобы стороны должны были выполнять разные действия, вводят в протокол идентификационную информацию, используют различные ключи для приема и передачи сообщений.

4. *Задержка пе́редачи сообщения* (forced delay) — перехват противником сообщения и навязывание его в более поздний мо-

мент времени. Это также разновидность атаки с повторным навязыванием сообщения.

Методы противодействия включают использование случайных чисел совместно с ограничением временно́го промежутка для ответа, использование временны́х меток.

5. *Комбинированная атака* (interleaving attack) — подмена или другой метод обмана, использующий комбинацию данных из ранее выполненных протоколов, в том числе протоколов, ранее навязанных противником.

Метод противодействия состоит в обеспечении целостности сеансов протоколов и отдельных сообщений.

6. Частный случай предыдущей атаки, в котором противник специально открывает одновременно несколько параллельных сеансов в целях использования сообщений из одного сеанса в другом, получил название *атака с параллельными сеансами* (parallel-session attack).

7. *Атака с использованием специально подобранных текстов* — атака на протоколы типа «запрос — ответ», при которой противник по определенному правилу выбирает запросы в целях получить информацию о долговременном ключе доказывающего. Эта атака может включать специально подобранные открытые тексты, если доказывающий должен подписать или зашифровать запрос, и специально подобранные шифрованные тексты, если доказывающий должен расшифровать запрос.

Методы противодействия этой атаке состоят во включении случайных чисел в запросы или ответы, а также в использовании протоколов с нулевым разглашением.

8. *Использование противником своих средств в качестве части телекоммуникационной структуры* — атака, при которой в протоколе идентификации между участниками A и B противник C входит в телекоммуникационный канал и становится его частью при реализации протокола между участниками A и B. При этом противник может подменить информацию, передаваемую между участниками A и B. Эта атака особенно опасна в случае формирования участниками A и B общего ключа по протоколу Диффи — Хеллмана. Она известна как *противник в середине* и заключается в полной подмене всех сообщений между сторонами.

Для защиты от данной атаки необходимо дополнить протокол средствами взаимной аутентификации сторон. Это могут быть либо дополнительные, либо встроенные процедуры взаимной аутентификации. Использование дополнительных процедур

в протоколе не всегда удобно, так как злоумышленник может подменять не все сообщения, а только относящиеся к выработке ключа. Более предпочтительно, чтобы аутентификация была заложена в сами процедуры выработки ключа, и в случае активного вмешательства протокол заведомо давал бы различные значения ключа сторонам и злоумышленнику. Кроме того, необходимо использовать защищенный канал для установления общего ключа между участниками A и B.

9. Во многих случаях проведение атаки облегчает дополнительная информация. Например, *атака с известным сеансовым ключом* (known-key attack) заключается в попытке получения информации о долговременном ключе или любой другой ключевой информации, позволяющей восстанавливать сеансовые ключи для других сеансов протокола.

Для защиты от такой атаки обеспечивают независимость между различными применяемыми ключами, которая достигается путем применения протоколов совместной выработки ключа, позволяющих гарантировать свойство новизны ключа (freshness) и не позволяющих ни одному из участников заранее предсказать значение ключа.

10. *Атака с неизвестным общим ключом* (unknown key-share attack) — атака, при которой нарушитель C открывает два сеанса с участниками A и B, выступая в первом случае от имени B, хотя последний может ничего не знать об этом. В результате будет сформирован общий ключ между участниками A и B, причем участник A будет уверен, что сформировал общий ключ с участником B, а участник B будет уверен, что сформировал общий ключ с участником C. Сам ключ может быть неизвестен участнику C. Примерный сценарий может выглядеть так: пусть B — филиал банка, A — держатель счета. Сертификат выдается центральным банком и содержит информацию о держателе счета. Пусть протокол электронного депозита вкладов должен выработать ключ для филиала. После аутентификации B как отправителя зашифрованный вклад помещают на счет, указанный в сертификате. Если не проводится никакой последующей аутентификации в зашифрованном сообщении депозита, то при успешно проведенной атаке вклад будет положен на счет C вместо счета A [42].

11. Наконец, имеется большое число типов атак, которые зависят от конкретной реализации протокола. Например, для криптографических протоколов на основе симметричных систем шифрования можно использовать особенности работы самих си-

Таблица 1.3

Основные виды атак на протоколы

Обозначение атаки	Название атаки	Определение
MITM	Противник в середине (man-in-the-middle attack)	Атака на протокол, при которой противник, подменяя сообщения в канале, имитирует действия каждой из сторон
Replay	Атака с повторной передачей (replay attack)	Атака на протокол с повторной передачей, при которой отдельные сообщения протокола запоминаются, а затем используются нарушителем в других сеансах протокола
TF	Подмена типа атаки (type flaw attack)	Атака на протокол, при которой поле сообщения протокола, имеющее заранее определенный тип, интерпретируется в нужный момент как поле другого типа
STS	Атака с известным разовым ключом (short-term secret)	Разновидность атаки на протокол с повторной передачей, основанная на знании одноразового секрета
PS	Атака с параллельными сеансами (parallel-session attack)	Атака на протокол, при которой нарушитель использует сообщения из нескольких одновременно работающих сеансов протокола
KN	Атака с известным сеансовым ключом (known-key attack)	Атака на протокол, при которой противнику известен сеансовый ключ
UKS	Атака с неизвестным общим ключом (unknown key-share attack)	Атака, при которой нарушитель C формирует неизвестный ему общий ключ между участниками A и B, связываясь с A от имени B, хотя последний может ничего не знать об этом, а с B — от своего имени

Таблица 1.4

Слабости, обнаруженные в известных протоколах

Протокол	Атака на протокол
ISO с симметричными ключами 1-проходный	Replay
ISO с симметричными ключами 2-проходный	Replay
Andrew Secure RPC	TF, Replay
Needham — Shroeder с симметричными ключами	STS
Denning — Sacco с симметричными ключами	TF
Otway–Rees	TF
Wide Mouthed Frog	PS
Yahalom	TF
Woo — Lam с симметричными ключами	Replay, PS
Woo — Lam с открытыми ключами	PS
ISO с открытыми ключами 1-проходный	Replay
ISO с открытыми ключами 2-проходный	Replay
NSPK (Needham–Shroeder Public Key)	MITM
NSPK с ключевым сервером	MITM
NSL (NSPK, модифицированный Lowe)	MITM
Denning — Sacco Key Distribution (с открытыми ключами)	MITM
Shamir — Rivest — Adleman 3-проходный	Replay, PS
CCITT X.509	TF
EKE (Encrypted Key Exchange)	PS

стем шифрования и, в частности, реализованных способов и режимов шифрования, синхронизации и т. п.

Чтобы защититься от подобных атак, необходимо провести анализ архитектуры протокола и структуры передаваемых сообщений в целях определения возможных уязвимостей, позволяющих, например, осуществить навязывание сообщений с известными или одинаковыми значениями определенных полей либо с помощью подмены типа некоторых полей.

12. Для криптографических протоколов, построенных на основе асимметричных систем шифрования, основной уязвимостью является возможность подмены открытого ключа одного из участников на другой открытый ключ с известной противнику секретной половиной этого ключа. В частности, это позволяет противнику узнавать содержание зашифрованных сообщений, отправляемых данному участнику. В данном случае нару-

шается свойство связанности открытого ключа и идентификатора участника. Поэтому атаку данного типа называют *атакой на основе связывания* (binding attack).

Для защиты от подобных атак вместо открытых ключей используют их сертификаты, создавая специальную инфраструктуру для выдачи, отзыва и проведения проверки их правильности.

В табл. 1.3 перечислены наиболее часто используемые типы атак. Заметим, что указанные в табл. 1.3 атаки могут рассматриваться как основные приемы проведения атак. На практике, как правило, применяют их комбинации. Данные атаки в их применении к известным протоколам будут подробно рассмотрены далее, а пока перечислим в табл. 1.4 известные примеры успешных атак на описываемые в данной книге протоколы.

1.5. Формальные методы анализа протоколов обеспечения безопасности

Исследование различных атак на криптографические протоколы и выяснение причин уязвимостей, позволяющих осуществлять эти атаки, является хорошим подспорьем для синтеза защищенных протоколов. Благодаря этому во многих случаях удается не повторять прошлых ошибок и строить достаточно надежные протоколы. Вместе с тем данный подход носит чисто эвристический характер и не позволяет замечать имеющиеся слабости в протоколе.

Дело в том, что несмотря на кажущуюся простоту — в протоколе формируется всего несколько коротких сообщений — проблема оценки безопасности протокола является очень сложной и многогранной. Так, для многих протоколов, считавшихся долгое время достаточно надежными, даже спустя десятки лет удавалось найти атаки, показывающие их уязвимость при определенных условиях. Это происходило благодаря попыткам формального обоснования свойств протокола и более четкой формулировке исходных предположений, на основе которых проводился анализ безопасности протокола.

Успех в применении формальных методов анализа, позволявших находить новые атаки, а также острая необходимость в эффективных средствах анализа большого числа прикладных промышленных протоколов, для которых проблема гарантирования свойств безопасности являлась чрезвычайно актуальной, поскольку применение ненадежных протоколов было связано с

большими потенциальными потерями, привели к появлению целого ряда различных средств, позволяющих осуществлять анализ протоколов в автоматизированном или полуавтоматизированном режиме.

В основе указанных средств лежат различные подходы, изложение которых выходит за рамки данной книги. Поэтому ограничимся здесь только кратким обзором.

Далее будут рассмотрены основные формальные системы анализа.

Формальные системы анализа на основе логик

Формальные системы анализа на основе логик используют дедуктивный подход, основанный на логической проверке корректности рассматриваемого протокола. Проверяемые свойства формулируются как утверждения в рамках некоторой логики, определяющей синтаксис и семантику, и для их проверки ищут логический вывод этих утверждений. Данный подход применяется, как правило, для доказательства корректности системы, для доказательства того, что система (протокол) действительно удовлетворяет определенным требованиям. Целью анализа является вывод утверждения, которое будет представлять корректность протокола. Невозможность вывода такого утверждения означает, что протокол, вероятно, некорректен.

Важным преимуществом такого подхода является возможность использования техники автоматического доказательства теорем.

При анализе протоколов обычно используют следующие модальные логики:

- *дохастическую логику*, или *логику доверия* (doxastic logic, logic of belief), — систему выводов, которая применяет правила о том, как вывести новое доверие на основании имеющегося доверия;
- *эпистемическую логику*, основанную на знаниях (epistemic logic, logic of knowledge); напоминает дохастическую с тем отличием, что она основана на знании;
- *темпоральную* (или *временну́ю*) *логику* (temporal logic), описывающую последовательности состояний и анализирующую правила вывода формул двух типов: формулы состояния (обращаются в истину в некотором состоянии) и формулы пути (обращаются в истину на протяжении некоторого пути).

Наибольшее распространение в силу их большей простоты получили логики доверия.

Логики доверия. В 1990 г. опубликована работа [37], ставшая основой для создания большого числа различных методов анализа, получивших название логик доверия. Авторы [37] предложили логику (называемую BAN-логикой) для описания тех утверждений о безопасности протокола, которым доверяют участники на каждом шаге его выполнения. В процессе выполнения шагов протокола проводится корректировка общей совокупности этих утверждений при некоторых типовых предположениях о возможном поведении сторон. Авторы предложили формальные методы записи таких высказываний, правила вывода, позволяющие строить формальные доказательства некоторых утверждений, определяющих конечные цели обоснования безопасности протокола.

Данная логика занимает особое место, так как она представляет первую попытку построения формального языка для описания исходных предположений, правил вывода и конечных целей анализа безопасности.

В дальнейшем появилось много других примеров и расширений: логика AUTLOG (V. Kessler, G. Wedel), логика объяснения (accountability logic) (R. Kaylar), логики RV (D. Kindred), GNY (L. Gong, R. Needham, R. Yahalom), BGNY/HOL, SvO (P. Syverson) и др.

Успешное решение задачи анализа протокола с помощью логик доверия объясняется, в частности, их очевидной простотой, они позволяют легко проводить доказательства вручную.

Хотя поиск доказательств не требует большой изобретательности, так как все дополнительные предположения заранее установлены, в ручных доказательствах иногда встречаются неявные пропуски важных деталей, исходных предположений и правил вывода. С другой стороны, при использовании быстрых автоматических доказательств могут быть получены такие выводы, которые окажутся непригодными для практики либо невыполнимыми.

Наиболее эффективно процесс автоматизации вывода утверждений для логики доверия был реализован для RV-логики. Ее автор Д. Киндред (D. Kindred) [55] предложил алгоритм, позволяющий строить все утверждения, выводимые из исходных предположений с помощью правил вывода. Основным преимуществом данного алгоритма является то, что в результате получается обозримое конечное множество утверждений, так называемое «представление логики доверия». Это было достигнуто путем введения отношения эквивалентности на множестве од-

нотипных высказываний и построения индуктивного (по рангу формулы) алгоритма генерации теории. Сравнивая такие представления для двух различных протоколов, можно выделять утверждения, справедливые для одного из протоколов и не справедливые для другого, и тем самым выяснять их индивидуальные особенности и уязвимости.

В работе [55] упоминается система верификации протоколов REVERE, в которой реализованы алгоритмы генерации протоколов в различных логиках доверия и приведены результаты ее применения. Эта система написана на языке программирования Standard ML (SML), который является функциональным, строго типизированным, поддерживающим модульную систему и абстрактные типы данных, определенных пользователем, с частично изменяемым синтаксисом, поэтому он удобен для работы с формулами и выражениями. Кроме того, формальная семантика языка SML опубликована, что делает его удобным для построения систем верификации. Как отмечается в работе [55], время генерации представления для большинства вариантов теории для различных протоколов в рамках логик BAN, AUTLOG, Accountability Logic и RV не превышает 1 мин.

Заметим, что логики доверия не позволяют анализировать все аспекты, связанные с безопасностью. С их помощью анализируют только те обстоятельства, которым доверяют честные участники протокола, правильно выполняющие все свои действия. Однако логики доверия имеют ряд недостатков:

- абстрактный характер не позволяет анализировать уязвимости, связанные с конкретной реализацией протокола;
- учитывается только внешний нарушитель (противник), но не рассматривается нечестный участник (внутренний нарушитель);
- не учитываются слабости криптографических схем;
- логика оперирует только с утверждениями о доверии, но не позволяет анализировать свойство конфиденциальности, так как не учитывается возможность неавторизованного использования секретов участниками.

Тем не менее применение логик доверия фактически впервые показало важность четкого формулирования исходных предположений и конечных целей анализа.

Автоматическое доказательство теорем (theorem proving). Автоматическое доказательство теорем достаточно широко применяется для верификации криптографических протоколов. Наиболее известны два подхода: NRL Protocol Analyzer

(C. Meadows [63]) и универсальный доказыватель Isabelle/HOL (L. Paulson [73, 74]), использующий теорию High Order Logic (HOL). Они позволили получить детальную верификацию для многих известных протоколов, в частности позволили анализировать атаки на основе новизны (freshness).

В данном случае проблема заключается в том, что доказательства могут уводить в сторону от исходной задачи. Основной трудностью остается получение нужной леммы в нужное время для получения нужного доказательства. Поэтому автоматизация этого процесса остается частичной. Более того, доказательства получаются длинными, сложными и для своего проведения требуют большого практического опыта.

Для анализа криптографических протоколов разработаны также специализированные алгоритмы и инструменты, основанные на методике доказательства теорем: это система TAPS (E. Cohen) [40] и система Хуимы (A. Huima) [53]. Специализированные системы требуют значительно меньшего взаимодействия с пользователем, чем универсальные. Однако в отличие от метода проверки на модели, они не позволяют находить контрпримеры для случаев, когда доказательство свойств безопасности приводит к ложным выводам.

Формальные системы анализа на основе верификационной техники

Верификация на основе автоматов. Конечные автоматы применяют не только для спецификации, но и для проведения анализа криптографических протоколов. В этом случае используют методику, известную под названием *методика анализа достижимости* [91]. Эта методика предполагает описание системы в следующем виде: для каждого перехода строят глобальное состояние системы, которое выражают через состояния всех участников системы и состояния коммуникационных каналов между ними. Каждое глобальное состояние затем анализируют и определяют его свойства. Затем вводят понятие небезопасного состояния и проверяют его достижимость.

Методика анализа достижимости эффективна для определения корректности протокола по отношению к его спецификациям, однако она не гарантирует безопасности от активного противника.

Вместе с тем моделирование протоколов, как правило, приводит к автоматам с бесконечным числом состояний.

Описан автоматический метод для верифицирования свойств конфиденциальности криптографических протоколов [51], в котором используется расширение древовидных автоматов, позволяющих совмещать теоретико-автоматные методики с особенностями дедуктивных методик. Древовидные автоматы представляют собой эффективный базовый механизм дедукции, пришедший из области доказательства теорем, позволяющий манипулировать бесконечными объектами. Преимуществом данного метода является обеспечение фактических гарантий выполнения свойств, характеризующих безопасность. Кроме того, имеется возможность анализа протоколов при наличии нескольких одновременно работающих параллельных сеансов протокола.

В работах [47, 48, 70] предложен метод TA4SP (Tree Automata-based Protocol Analyser), обладающий той особенностью, что он сочетает преимущества систем автоматического доказательства теорем с формальной абстрактной интерпретацией, называемой аппроксимацией (оценкой сверху). Аппроксимация упрощает доказательство таким образом, что оно затем может быть выполнено автоматически. Благодаря ей в рассматриваемом подходе нет требования о завершении процесса переписывания, но вместо этого обеспечивается эффективное построение аппроксимирующего автомата, позволяющего сразу ответить на вопрос о достижимости. Для абстрактной интерпретации протоколов используется комбинация моделей древоидных автоматов и систем переходов: древоидные автоматы и дополнение применяются для абстрактной структуры, а системы переходов состояний — для представления понятия времени.

Верификация на основе проверки на модели (model checking). Метод проверки на модели, впервые предложенный в начале 1980-х гг., в последнее время широко используется в индустриальной практике, в частности в практических приложениях, особенно для проверки аппаратуры и коммуникационных протоколов. Метод заключается в специфицировании свойств системы в виде формул темпоральной логики, построении модели в виде конечного автомата, автоматической проверке того, что эта модель удовлетворяет спецификации, причем при отрицательном выводе вырабатывается контрпример.

Главная идея применения этого метода к анализу протоколов заключается в том, что по протоколу строят модель (систему переходов), состояния которой представляют собой определенные множества высказываний. Эту модель затем проверяют на удовлетворение некоторому свойству в каждом состоянии, которого

можно достичь выходя из определенного множества начальных состояний. Поскольку число состояний, как правило, бесконечно, методы такого рода эффективны в том и только в том случае, если удается осуществить эффективное сведение к конечной модели. При таком сведении следует помнить, что результаты верификации нужно относить не к протоколу, а к его модели.

Метод проверки на модели состоит в переборе всех возможных переходов автомата из одного состояния в другое из некоего начального состояния системы, чтобы убедиться в том, что все они являются безопасными, или доказать обратное. Основное достоинство этого подхода заключается в том, что если свойство не выполняется, то программа-верификатор предоставляет контрпример, состоящий из траектории, ведущей к опасному состоянию системы, называемому состоянием атаки, тем самым явно указывается атака на протокол.

Метод проверки на модели имеет недостаток, широко известный под названием «взрыв числа состояний». При моделировании времени как непрерывной сущности даже самая простая модель имеет бесконечное число состояний. Наиболее известным подходом к решению этой проблемы является метод символической проверки модели, в котором проблема упрощается с помощью формирования классов эквивалентности, и, в частности, метод «проверки на лету». Последний метод реализован в форме очень эффективного анализатора протоколов OFMC (On-the-Fly Model-Checker) [24, 25]. Он представляет собой комбинацию двух методов. Первым методом является использование инертных (lazy) типов данных, позволяющих редуцировать бесконечное пространство состояний до конечного. Второй метод состоит в применении символического подхода для моделирования противника, чьи действия воспроизводятся оперативно в стиле управления по запросу (demand-driven way). Анализатор OFMC строит бесконечное дерево по выбранной проблеме анализа протокола по принципу управления по запросу, т. е. «на лету» (on-the-fly), отсюда и происходит название. Авторы формализовали применение символического метода для того, чтобы значительно сократить пространство состояний без исключения из рассмотрения каких-либо атак. Подход, названный авторами *методом инертного противника*, использует символическое представление с целью избежать явного перечисления возможных сообщений, которые противник может создавать. Сообщения, известные противнику, представляют выражениями с переменными, значения которых не фиксируют.

Средство проверки на модели, основанное на задаче выполнимости SATMC (SAT-based Model-Checker) [13, 14], строит пропозициональную формулу, кодирующую отношение перехода в системе переходов, моделирующей работу протокола, специфицированное с помощью темпоральной логики, начальное состояние и множество состояний, представляющих различные варианты свойств безопасности. В результате компиляции осуществляется преобразование комбинации указанных проблем безопасности в задачи планирования, для решения которых используется современная техника решения задач выполнимости, разработанная для проблем планирования. Пропозициональная формула подается на вход алгоритма проверки выполнимости (SAT-солвера), причем любое найденное решение преобразуется обратно в атаку.

Средство Athena [80] является анализатором моделей, специально разработанным для анализа протоколов. Оно интегрирует подходы на основе проверки на модели и автоматического доказательства теорем на основе параметрического пространства нитей. Как и подход NRL Protocol Analyzer [63], средство Athena работает в обратном направлении, стартуя из запрещенного состояния, и пытается выявить начальные условия, необходимые для достижения этого состояния. По мере продвижения поиска состояния конкретизируются в том смысле, что все больше переменных становятся связанными. Средство Athena использует в качестве модели вычислений расширение модели пространства нитей (strand space) [87]. Благодаря простоте и интуитивности этой модели Athena обладает высокой эффективностью.

AVISPA. В 2005 г. появился новый программный продукт — AVISPA (Automated Validation of Internet Security Protocols and Applications) [18], разработанный в рамках международного проекта, в котором участвовали LORIA — INRIA (Франция), ETH (Цюрих, Швейцария), Университет Генуи (Италия), Siemens AG (Германия). Судя по заявлениям его разработчиков, продукт AVISPA должен стать прорывом в области анализа криптопротоколов. Разработка данного средства рассматривается как единый европейский проект, реализуемый с участием многих ведущих институтов и организаций европейских стран. Он интегрирует различные современные подходы к анализу протоколов, такие как проверка на модели (model-checking), древовидные автоматы, временна́я логика. При этом используются разработки, созданные после 2000 г. Специально для него были разработаны версии языков HLPSL (High-Level Protocols

Specification Language) [16] и IF (Intermediate Format) [17], позволившие существенно расширить класс изучаемых протоколов, а также интегрировать в единую платформу сразу несколько различных методов.

В отличие от других средств исполняемый программный код этого средства доступен через Интернет. Поэтому изучение этого средства представляет большой интерес как с точки зрения исследования возможностей реализованного в нем подхода, так и с точки зрения применения его на практике. Полная информация о разработке продукта AVISPA и публикациях, лежащих в его основе, доступна на интернет-сайте http://www.avispa-project.org.

Структура средства AVISPA показана на рис. 1.1. Принцип его работы состоит в следующем: сначала составляют спецификацию исследуемого протокола на языке HLPSL и записывают в файл с расширением hlpsl. Это язык высокого уровня, основанный на ролях: независимые базовые роли для каждого участника и композиционные роли для представления сценариев базовых ролей.

Спецификация на языке HLPSL транслируется в промежуточный формат IF с использованием транслятора HLPSL2IF. Промежуточный формат IF представляет собой язык более низкого уровня, чем язык HLPSL, который компилируется непосредственно выходными модулями средства AVISPA. Это преобразование прозрачно для пользователя, так как транслятор ра-

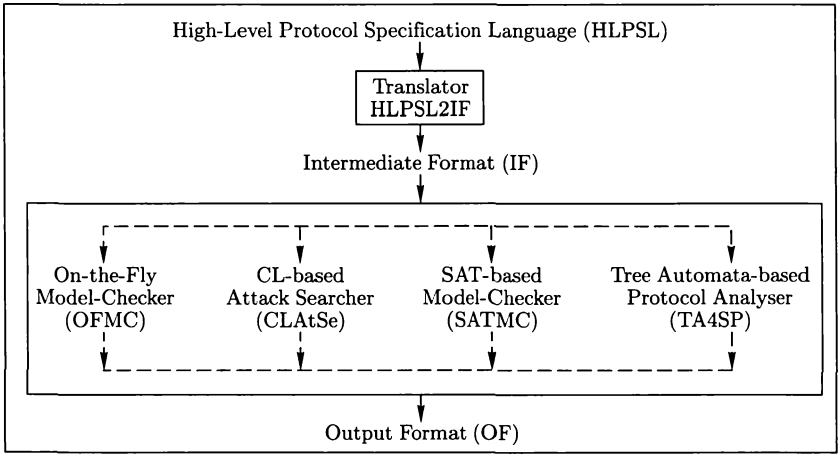

Рис. 1.1. Архитектура AVISPA

ботает автоматически. Более подробно о промежуточном формате можно прочитать в публикациях [17, 19].

После поступления на вход средства AVISPA спецификации протокола на языке IF он проверяется на предмет выполнения или нарушения указанных целей безопасности. В настоящее время средство AVISPA включает четыре выходных модуля:

1) OFMC (On-the-Fly Model-Checker);
2) CLAtSe (CL-based Attack Searcher) [88];
3) SATMC (SAT-based Model-Checker);
4) TA4SP (Tree Automata-based Protocol Analyser).

Этот список может быть в дальнейшем расширен. Данные модули частично дублируют и взаимно дополняют друг друга.

В настоящее время осуществляется доработка и тестирование средства AVISPA. В качестве основы для тестирования выбрана библиотека протоколов, встречающихся в документах IETF. Результаты и проблемные ситуации, возникающие при анализе протоколов, отражаются в документах [20, 21].

Другие средства. Помимо AVISPA в первое десятилетие XXI в. появились и другие автоматизированные инструментальные средства анализа протоколов, реализованные в рамках различных национальных проектов. Здесь следует упомянуть такие средства, как Proverif (INRIA, Франция), HERMES (проект EVA, Франция), Scyther (ETH, Цюрих) и др. Они представляют собой мощные интегрированные, основанные на последних достижениях верификационной техники и автоматического доказательства средства анализа протоколов, которые позволяют анализировать протокол как для конечного, так и для бесконечного числа сеансов.

Контрольные вопросы

1. Охарактеризуйте понятие «протокол обеспечения безопасности».
2. Приведите пример некриптографического протокола обеспечения безопасности.
3. Перечислите виды аутентификации.
4. Приведите примеры защищенных протоколов, в которых не требуется обеспечение свойства конфиденциальности.
5. Дайте определение понятия «интерактивное доказательство».
6. Перечислите возможные подходы к классификации криптографических протоколов.
7. Перечислите наиболее распространенные атаки на криптографические протоколы.
8. Приведите примеры способов защиты от атак на криптографические протоколы.

Криптографические хеш-функции

2.1. Функции хеширования и целостность данных

Определение хеш-функции. Пусть X — множество, элементы которого будем называть *сообщениями*. Обычно полагают $X \subseteq A^*$, где A^* — множество конечных последовательностей символов из некоторого конечного алфавита A (как правило, $A = \{0,1\}$). Пусть $Y = V_n$ — множество двоичных векторов фиксированной длины n. *Хеш-функцией* называют всякую легко вычисляемую функцию

$$h \quad X \to Y$$

Хеш-функции применяют:

- при проведении статистических экспериментов: наблюдаемые случайные величины с бесконечным числом исходов удобнее «заменять» на случайные величины с конечным числом исходов;
- тестировании логических устройств: для тестирования микросхем на их вход подают очень длинную тестовую последовательность, а затем значение хеш-функции от выходной последовательности сравнивают с эталонным;
- построении алгоритмов быстрого поиска в текстовых базах данных: для осуществления поиска нужного сообщения в большом списке сообщений различной длины удобнее сравнивать друг с другом не сами сообщения, а короткие значения их сверток, играющих одновременно роль контрольных сумм;
- проверке целостности записей в базах данных: при извлечении каждой записи вычисляют значение свертки и сравнивают с хранимым истинным значением контрольной суммы, а также во многих других случаях.

Основным требованием, предъявляемым к таким хеш-функциям, является *равномерность* распределения их значений при случайном выборе значений аргументов. Для конечного множества X при случайном и равновероятном выборе сообщений это условие эквивалентно наличию одинакового числа прообразов,

т. е. сообщений с заданным значением хеш-функции, для каждого значения свертки. Обычно число возможных сообщений значительно превосходит число возможных значений сверток, в силу чего для каждого значения свертки имеется большое число прообразов. При этом пару сообщений с одинаковыми значениями хеш-функции называют *коллизией*.

Целостность данных и аутентификация сообщений. В криптографических приложениях хеш-функции применяют, прежде всего, для установления целостности и аутентификации источника данных. Рассмотрим эти свойства более подробно.

Термин *аутентификация* означает установление подлинности. Он может относиться ко всем аспектам взаимодействия: сеансу связи, взаимодействующим сторонам, передаваемым сообщениям и т. д. Применительно к самой информации аутентификация означает проверку того, что данные, передаваемые по каналу связи, являются подлинными по своему источнику и содержанию, по времени создания, времени пересылки и т. д.

Целостность данных — свойство, позволяющее убедиться в том, что данные не изменялись неавторизованным способом с тех пор, как они были созданы, переданы или сохранены авторизованным источником. Под изменениями обычно понимают пропуски, вставки, замены и перестановки фрагментов сообщения.

Аутентификация источника данных — получение подтверждения того, что рассматриваемый документ был создан именно указанным источником информации. Подчеркнем, что при этом не требуется проверка времени создания и единственности документа, важно только то, что он был создан в некоторый (обычно неопределенный) момент времени в прошлом. Нарушение «единственности документа», в частности, подразумевает его повторную передачу или повторное использование. Если источник сообщений фиксирован, то вместо термина «аутентификация источника данных» используют термин «аутентификация сообщений».

Целостность данных и аутентификация источника данных тесно связаны друг с другом. Действительно, если данные подверглись модификации, то у них автоматически изменился источник. Если же не установлен источник, то без ссылки на него нельзя разрешить проблему целостности. В связи с этим обычно полагают, по определению, что аутентификация источника данных включает проверку целостности данных.

Особо остановимся на дополнительном гарантировании единственности и своевременности передачи сообщений. В этом слу-

чае используют термин «аутентификация транзакции», означающий аутентификацию сообщения с подтверждением единственности и своевременности передачи данных. Такой тип аутентификации предоставляет возможность защиты от повторного использования ранее переданных сообщений, что является необходимым в тех случаях, когда подобная угроза может привести к нежелательным последствиям. Примером таких приложений являются электронные банковские платежи или системы автоматизированного управления подвижными объектами.

Для обеспечения единственности и своевременности передачи сообщений обычно используют дополнительные параметры, которые добавляют к передаваемым сообщениям. Это могут быть метки времени или некоторые последовательности чисел. Если метки времени позволяют установить время создания или передачи документа, то последовательность чисел гарантирует правильность порядка получения сообщений. Помимо этого для аутентификации последующих сообщений могут использоваться случайные числа, передаваемые в предыдущих сообщениях. Такой способ позволяет организовать «жесткое сцепление» идущих друг за другом сообщений.

Криптографические хеш-функции. В криптографии особо выделяют два важных типа криптографических хеш-функций — задаваемые ключом и не зависящие от ключа (или бесключевые), часто называемые просто хеш-функциями. Рассмотрим их применение для решения задач контроля целостности данных и аутентификации источника данных.

Хеш-функции, задаваемые ключом, применяют в системах с симметричными ключами и доверяющими друг другу сторонами. Если передающая и проверяющая стороны доверяют друг другу, то они могут иметь общий секретный ключ.

При решении задачи контроля целостности для каждого набора данных вычисляют значение хеш-функции, называемое *кодом аутентичности сообщения*, которое передают или хранят вместе с самими данными. При получении данных пользователь вычисляет значение свертки и сравнивает его с имеющимся контрольным значением. Несовпадение говорит о том, что данные были изменены. Хеш-функция, служащая для выработки кода аутентичности сообщения, должна позволять (в отличие от обычной контрольной суммы) обнаруживать не только случайные ошибки в наборах данных, возникающие при их хранении и передаче, но и сигнализировать об активных атаках злоумышленника, пытающегося осуществить навязывание ложной

информации. Для того чтобы злоумышленник не мог самостоятельно вычислить контрольное значение свертки и тем самым осуществить успешную имитацию или подмену данных, хеш-функция должна существенно зависеть от секретного, не известного злоумышленнику ключа пользователя. Применение зависящих от ключа хеш-функций не позволяет противнику создавать поддельные, сфабрикованные сообщения (fabrication) при атаках типа *имитация* (impersonation) и модифицировать передаваемые сообщения (modification) при атаках типа *подмена* (substitution). Аутентификация источника при этом выполняется автоматически, так как никто кроме двух сторон, обладающих общим секретным ключом, не может вычислить правильное значение хеш-функции.

Хеш-функции, задаваемые ключом, называют *кодами аутентификации сообщений* (message authentication code — MAC). Они дают возможность без дополнительных средств гарантировать как подлинность источника данных, так и целостность данных в системах с доверяющими друг другу пользователями.

В случае с не доверяющими друг другу сторонами ключ должен быть известен только одной передающей стороне, а проверяющая сторона должна пользоваться только открытой информацией. В связи с этим использование симметричных систем, при котором обе стороны обладают одним и тем же секретным ключом, уже не представляется возможным. В такой ситуации применяют схемы цифровой подписи, позволяющие осуществлять контроль целостности и аутентификацию источника данных. Как правило, при этом сообщение, прежде чем быть удостоверенным личной подписью, зависящей от секретного ключа пользователя, «сжимается» с помощью хеш-функции, выполняющей функцию кода обнаружения ошибок. В данном случае хеш-функция не зависит от секретного ключа и может быть известна всем. Основными требованиями к ней являются однонаправленность, дающая гарантии невозможности подделки, т. е. подбора сообщения с правильным значением подписи, а также трудность нахождения коллизий, гарантирующая невозможность подмены подписанного документа.

Хеш-функции, не зависящие от ключа, называют *кодами обнаружения ошибок* (modification detection code — MDC; manipulation detection code, message integrity code — MIC). Для гарантирования с их помощью целостности данных необходимо использовать дополнительные средства (например, защищенный канал, шифрование или схему цифровой подписи). Для аутенти-

фикации источника данных в системах с не доверяющими друг другу сторонами используют схему цифровой подписи, при этом бесключевые хеш-функции обычно применяют для предварительного сжатия подписываемых данных.

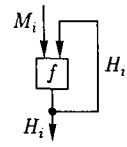

Рис. 2.1. Применение одношаговой сжимающей функции

Одношаговые сжимающие функции. Как правило, хеш-функции строят на основе так называемых *одношаговых сжимающих функций* $y = f(x_1, x_2)$, зависящих от двух переменных. Здесь x_i $(i = 1, 2)$, y — двоичные векторы длиной m и n соответственно, причем n — длина свертки. Для получения значения $h(M)$ сообщение M сначала разбивают на блоки длиной m (при этом если длина сообщения не кратна m, то последний блок неким специальным образом дополняют до полного), а затем к полученным блокам $M_1, M_2, \ldots, M_N$ применяют следующую последовательную процедуру вычисления свертки (рис. 2.1):

$$\begin{gathered} H_0 = v, \\ H_i = f(M_i, H_{i-1}), \quad i = 1, \ldots, N, \\ h(M) = H_N. \end{gathered} \tag{2.1}$$

Здесь v — некоторый фиксированный начальный вектор. Если функция f задается ключом, то этот вектор можно принять равным нулевому вектору. Если же функция f не зависит от ключа, то для исключения возможности перебора коротких сообщений (при попытках обращения хеш-функции) этот вектор можно составить из фрагментов, указывающих дату, время, номер сообщения и т. п.

При таком подходе свойства хеш-функции h полностью определяются свойствами одношаговой сжимающей функции f

2.2. Хеш-функции, задаваемые ключом

Свойства хеш-функций, задаваемых ключом. Обычные атаки на такие хеш-функции заключаются в имитации, т. е. в передаче сфабрикованных сообщений в пустом канале, а также в подмене передаваемых сообщений в целях навязывания приемной стороне ложных сообщений. Поэтому в криптографических приложениях к функциям хеширования, зависящим от ключа, предъявляют следующие основные требования:

1) высокая сложность подбора сообщения с правильным значением свертки (что обеспечивает невозможность создания поддельных ложных сообщений при атаках типа «имитация»);

2) высокая сложность подбора для заданного сообщения с известным значением свертки другого сообщения с правильным значением свертки (что обеспечивает невозможность модификации передаваемых сообщений при атаках типа «подмена»); заметим, здесь не требуется, чтобы найденная пара сообщений образовывала коллизию.

Иногда эти свойства объединяют в одно более сильное свойство вычислительной устойчивости. Это требование означает высокую сложность подбора для каждого множества сообщений $\{x_1, \quad , x_t\}$, $t \geq 0$ с известными значениями сверток еще одного сообщения x, $x \neq x_i$ $(i = 1, \ldots, t)$ с правильным значением свертки (возможен случай $h(x) = h(x_i)$, $i \in \{1, \ldots, t\}$).

Заметим, что здесь и всюду далее слова «высокая сложность» означают такую вычислительную сложность задачи, при которой ее решение с использованием имеющейся вычислительной техники за реальное время невозможно.

Ключевые функции применяют в ситуациях, когда стороны доверяют друг другу и могут иметь общий секретный ключ. Обычно в этих условиях не требуется, чтобы система обеспечивала защиту в случае отказа получателя от факта получения сообщения или его подмены. Поэтому от хеш-функций, задаваемых ключом, не требуется устойчивости к коллизиям.

Заметим, что из свойства вычислительной устойчивости вытекает невозможность определения ключа, так как знание ключа позволяет вычислять значение свертки для любого набора данных.

В то же время обратное утверждение неверно, так как подбор значения функции хеширования возможен в некоторых случаях без предварительного определения ключа.

Пример хеш-функции на основе блочного шифра. В качестве примера рассмотрим широко распространенную хеш-функцию, построенную на основе одношаговой сжимающей функции вида

$$f_k(x, H) = E_k(x \oplus H),$$

где E_k — алгоритм блочного шифрования (рис. 2.2, *а*).

Для вычисления значения $h(M)$ сообщение M представляют в виде определенной последовательности n-битовых блоков $M_1, M_2, \quad , M_N$.

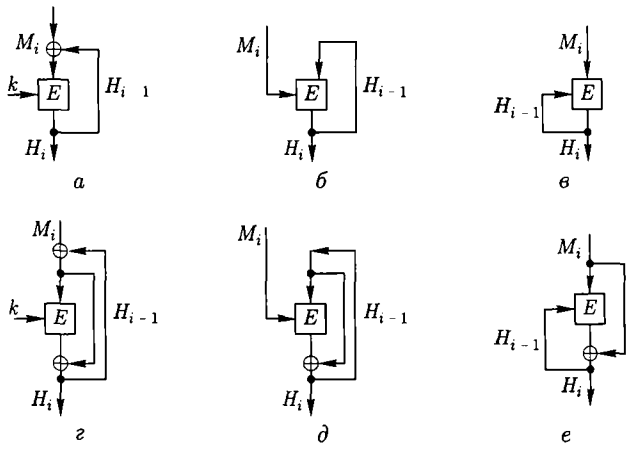

Рис. 2.2. Варианты ($а$ — $е$) построения одношаговых хеш-функций

Алгоритм вычисления свертки имеет следующий вид:

$$H_0 = 0,$$
$$H_i = E_k(M_i \oplus H_{i-1}), \quad i = 1, \quad ., N, \tag{2.2}$$
$$h(M) = H_N.$$

Данный алгоритм фактически совпадает с режимом шифрования со сцеплением блоков, с той лишь разницей, что в качестве результата берется не весь шифрованный текст $H_1, H_2, \quad , H_N$, а только его последний блок. Такой режим в ГОСТ 28147 — 89 называют *режимом выработки имитовставки.*

Построение хеш-функций, задаваемых ключом, на основе бесключевых. Еще одной основой для построения хеш-функций, задаваемых ключом, могут служить бесключевые хеш-функции. При этом для вычисления значения свертки ключ приписывают исходному сообщению.

Заметим, что если ключ просто дописывать в начало или в конец исходного сообщения, то это может привести к потенциальным слабостям, позволяющим в некоторых случаях осуществлять модификацию сообщений.

Пусть, например, ключ k добавляют к началу сообщения согласно формуле $h_k(M) = h(k||M)$, где символом $||$ обозначена операция конкатенации (приписывания) сообщений. В дальнейшем для удобства в тех случаях, когда это не вызывает недоразумений (хеш-функция всегда зависит от одной переменной), будем вместо символа конкатенации использовать запятую. Если функция h построена на основе одношаговой сжимающей

функции по формуле (2.1), то по известным значениям M и $H = h(k, M)$ можно вычислить значения этой функции для любых сообщений вида (M, M') с дописанным произвольным окончанием M' Это объясняется итеративностью процедуры вычисления функции, в силу которой для нахождения значения $H' = h(k, M, M')$ не требуется знание ключа k; достаточно воспользоваться уже вычисленным «промежуточным» значением H. Поэтому такая функция не устойчива к модификации.

В случае если ключ добавляют в конец сообщения согласно формуле $H = h_k(M) = h(M, k)$, знание коллизии для функции h, т. е. пары x_1, x_2 $(x_1 \neq x_2)$, такой, что $h(x_1) = h(x_2)$, позволяет вычислять значения $h(x_1, k) = h(x_2, k)$ для любого ключа k. Поэтому трудоемкость модификации сообщения $M = x_1$ оценивается не величиной $O(2^n)$, а сравнима с трудоемкостью поиска коллизии, оцениваемой величиной $O(2^{n/2})$, так как в данном случае применима атака, основанная на *парадоксе дней рождений* (см. далее).

В связи с этим более предпочтительными являются способы введения ключа, при которых ключ приписывают к сообщению не в одном, а, по крайней мере, в двух местах. Авторы [64] указывают два таких способа:

$$H = h(k, y, M, k),$$

$$H = h(k, y_1, h(k, y_2, M)),$$

где y, y_1, y_2 — дополнения ключа k до размера, кратного длине блока n.

Для определенных бесключевых хеш-функций h такой подход позволяет строить эффективно вычисляемые и устойчивые к атакам ключевые хеш-функции. Недостатком такого метода является слишком большая длина n свертки. Дело в том, что для целей проверки целостности обычно выбирают длину свертки n в пределах $32 - 64$, а для аутентификации необходимо выполнение условия $n \geq 128$.

Еще один известный пример построения хеш-функции, задаваемой ключом, на основе бесключевой хеш-функции дает алгоритм HMAC, согласно которому

$$\mathrm{HMAC}_k(M) = h(k^* \oplus a, h(k^* \oplus b, M)),$$

где a, b — некоторые константы, длина которых совпадает с размером блока; k^* получают дополнением ключа k нулями в начале до получения полного блока [26, 56].

Например, при использовании в качестве хеш-функции h алгоритмов MD5 и SHA-1 получаем алгоритмы HMAC-MD5 и HMAC-SHA-1 соответственно.

Использование хеш-функции, задаваемой ключом, и симметричного шифрования. Возможно совместное использование хеш-функции, задаваемой ключом, и симметричного шифрования в соответствии с одной из следующих схем:

$$E_{k_1}(M, h_{k_2}(M)),$$
$$(E_{k_1}(M), h_{k_2}(M)),$$
$$(E_{k_1}(M), h_{k_2}(E_{k_1}(M))),$$

При таком подходе не только ключи шифрования k_1 и хеш-функции k_2 должны быть независимыми, но и сами алгоритмы шифрования и вычисления значения хеш-функции также должны иметь существенные различия. В противном случае возникают дополнительные соотношения, которые можно использовать для отбраковки ключей.

Использование бесключевых хеш-функций с последующим симметричным шифрованием. Аутентификация источника возможна также при совместном использовании бесключевой хеш-функции и симметричного шифрования. Для этого достаточно воспользоваться одной из следующих форм передаваемого сообщения: $E_k(M, h(M))$ или $(M, E_k(h(M)))$. Следует отметить, что алгоритм шифрования E_k в данном случае должен быть стойким к атакам на основе известного открытого текста.

Специально построенные хеш-функции, задаваемые ключом. Существуют хеш-функции, задаваемые ключом, не использующие какую-либо основу, например блочное шифрование или бесключевую хеш-функцию, а разработанные независимо с учетом эффективной реализации на современных ЭВМ.

В качестве примера рассмотрим хеш-функцию MAA (Message Authenticator Algorithm), утвержденную стандартом ISO 8731-2 в 1983 г. Она вычисляется определенным образом.

1. Расширение ключа (не зависит от открытого текста): 64-битовый ключ K по определенному алгоритму расширяют до шести 32-битовых векторов X, Y, V, W, S, T (X, Y — начальные значения; V, W — основные переменные цикла; S, T добавляют к концу сообщения).

2. Установка начальных значений: представляют сообщение M в виде следующей последовательности 32-битовых векторов: $M = M_1 \dots M_t$, в которой $M_{t-1} = S$, $M_t = T$, $1 \le t \le 10^6$

Полагают $v = V$ $H_1 = X$, $H_2 = Y$ Определяют константы A, B, C, D.

3. Основной цикл:

$$
\begin{aligned}
v &= v \ll 1 \quad \text{(сдвиг на 1 шаг влево)},\\
U &= v \oplus W,\\
t_1 &= (H_1 \oplus M_i) \times_1 ((((H_2 \oplus M_i) + U) \vee A) \wedge C),\\
t_2 &= (H_1 \oplus M_i) \times_2 ((((H_2 \oplus M_i) + U) \vee B) \wedge D),\\
H_1 &= t_1, \quad H_2 = t_2,\\
i &= i + 1.
\end{aligned}
$$

Здесь $\times_1$, $\times_2$ — операции умножения по $\bmod(2^{32} - 1)$ и $\bmod(2^{32} - 2)$ соответственно.

4. Результат: $h_K(M) = H_1 \oplus H_2$.

2.3. Хеш-функции, не зависящие от ключа

Свойства хеш-функций, не зависящих от ключа. Обычно требуется, чтобы хеш-функции, не зависящие от ключа, обладали следующими свойствами:

1) однонаправленность;

2) устойчивость к коллизиям;

3) устойчивость к нахождению второго прообраза.

Эти свойства означают, соответственно высокую сложность нахождения сообщения с заданным значением свертки, пары сообщений с одинаковыми значениями свертки, второго сообщения с тем же значением свертки для заданного сообщения с известным значением свертки.

Например, хеш-функция CRC-32, представляющая контрольную сумму, является линейным отображением и поэтому не удовлетворяет ни одному из этих трех свойств.

Использование в качестве бесключевой хеш-функции рассмотренной выше функции (2.2), построенной на основе алгоритма блочного шифрования в режиме выработки имитовставки (см. рис. 2.2, *а*), также нецелесообразно, так как обратимость блочного шифрования позволяет подбирать входное сообщение для любого значения свертки при фиксированном и общеизвестном ключе.

Справедливо утверждение 1.

Утверждение 1. Если функция хеширования h построена на основе одношаговой сжимающей функции f по пра-

вилу (2.1), то из устойчивости к коллизиям функции f следует устойчивость к коллизиям функции h.

Действительно, если функция h имеет коллизию, то на некотором шаге i должна существовать коллизия функции f (при определении коллизий функцию $f(x_1, x_2)$ следует рассматривать как функцию от одной переменной, полученной конкатенацией переменных x_1, x_2 в один входной вектор).

Укажем взаимозависимость между первым и вторым свойствами.

Утверждение 2. Если хеш-функция устойчива к коллизиям, то она устойчива к нахождению второго прообраза.

Действительно, если для заданной пары сообщение — свертка можно подобрать второй прообраз, то полученная пара сообщений будет составлять коллизию.

Утверждение 3. Устойчивая к коллизиям хеш-функция не обязательно является однонаправленной.

В качестве примера несжимающей функции приведем функцию $h(x) = x$, которая, очевидно, является устойчивой к коллизиям и к нахождению второго прообраза, но не является однонаправленной.

В качестве примера сжимающей хеш-функции рассмотрим функцию h, определенную следующими условиями:

$h(x) = (1, x)$, если $x = n$,

$h(x) = (0, g(x))$, если $x \neq n$,

где x — битовая длина; $g(x)$ — сжимающая n-битовая функция, устойчивая к коллизиям.

Функция h также является устойчивой к коллизиям и к нахождению второго прообраза, но, очевидно, не является однонаправленной.

Утверждение 4. Пусть $h \quad X \to Y$ — хеш-функция, $|Y| < \delta|X|$, $0 < \delta < 1$. Тогда если существует эффективный алгоритм обращения функции h, то существует вероятностный алгоритм нахождения коллизии функции h с вероятностью успеха, большей $1 - \delta$.

Доказательство. Будем случайно и равновероятно выбирать сообщение x, вычислять $y = h(x)$, $x' = h^{-1}(y)$ и сравнивать x с x' Покажем, что данный алгоритм имеет вероятность успеха $p > 1 - \delta$. Под успехом мы понимаем построение x', отличного от x.

Пусть $X = X_1 \cup \quad \cup X_m$ — разбиение X на классы, состоящие из сообщений с одинаковыми значениями хеш-функции. Ясно, что $m \le |Y|$. Легко заметить, что выполняются следующие соотношения:

$$p = \frac{1}{|X|} \sum_{i=1}^{m} \sum_{x \in X_i} \frac{|X_i| - 1}{|X_i|} = \frac{1}{|X|} \sum_{i=1}^{m} (|X_i| - 1) =$$

$$= \frac{|X| - m}{|X|} \ge \frac{|X| - |Y|}{|X|} > 1 - \delta.$$

Утверждение доказано.

Заметим, что трудоемкость подбора прообраза для однонаправленной функции или трудоемкость поиска второго прообраза оценивают величиной $O(2^n)$. В то же время трудоемкость поиска коллизии оценивают величиной $O(2^{n/2})$, так как в данной ситуации применима атака, основанная на парадоксе дней рождений.

Парадокс дней рождений заключается в том, что вероятность p наличия коллизии (т. е. совпадения) в выборке из m элементов объема $O(\sqrt{m})$ принимает достаточно большое значение.

Далее нам потребуется *неравенство Йенсена*: если функция f непрерывна и выпукла вверх на отрезке, то для произвольных точек $x_1, \ldots, x_t$ этого отрезка и при любых $a_i \ge 0$ $(i = \overline{1,t})$, $\sum_{i=1}^{t} a_i = 1$, выполняется неравенство

$$\sum_{i=1}^{t} a_i f(x_i) \le f(\sum_{i=1}^{t} a_i x_i).$$

Лемма 1. Пусть $p_1 + p_2 + \quad + p_m = 1$ $(0 \le p_i \le 1$, $1 \le i \le m)$, тогда

$$\sum p_{i_1} p_{i_2} \ldots p_{i_k} \le \prod_{i=1}^{k-1} \left(1 - \frac{i}{m}\right),$$

где сумма берется по всем наборам различных индексов $i_1, i_2, \ldots, i_k$; $1 \le i_j \le m$; $1 \le j \le k$.

Доказательство. Воспользуемся индукцией по k. При $k = 1$ утверждение очевидно. Предположим, что для $k - 1$ оно верно и докажем для k.

Обозначим символом S исходную сумму и разложим ее по первому сомножителю

$$S = \sum p_{i_1} p_{i_2} \dots p_{i_k} = \sum_{j=1}^{m} p_j (1-p_j)^{k-1} \sum p_{i_2}^{(j)} \dots p_{i_k}^{(j)},$$

где $p_i^{(j)} = p_i (1-p_j)^{-1}$ $(i \neq j)$; последняя сумма берется по всем различным наборам индексов, не содержащих j.

По индукции в силу равенств

$$p_1^{(j)} + \quad + p_{j-1}^{(j)} + p_{j+1}^{(j)} + \dots + p_m^{(j)} = 1, \quad 1 \leq j \leq m,$$

получаем

$$\sum p_{i_2}^{(j)} \dots p_{i_k}^{(j)} \leq \prod_{i=1}^{k-2} \left(1 - \frac{i}{m-1}\right)$$

В то же время в силу неравенства Йенсена, примененного для выпуклых вверх на отрезке $[0,1]$ функций $(1-x)^{(k-1)}$ и $-x^2$, можно записать

$$\sum_{j=1}^{m} p_j (1-p_j)^{k-1} \leq \left(\sum_{j=1}^{m} p_j (1-p_j) \right)^{k-1} =$$

$$= \left(1 - \sum_{j=1}^{m} p_j^2\right)^{k-1} \leq \left(1 - \frac{1}{m}\right)^{k-1}$$

Окончательно получаем

$$S \leq \left(1 - \frac{1}{m}\right)^{k-1} \prod_{i=1}^{k-2} \left(1 - \frac{i}{m-1}\right) = \prod_{i=1}^{k-1} \left(1 - \frac{i}{m}\right)$$

Утверждение 5. Для вероятности p существования коллизии в выборке объемом k из m элементов, полученной по схеме независимых испытаний с возвращением, справедлива нижняя оценка

$$p > 1 - \mathrm{e}^{-\frac{(k-1)^2}{2m}}$$

где e — основание натурального логарифма.

Доказательство. Рассмотрим сначала случай, когда на множестве из m элементов задано равномерное распределение. Вероятность отсутствия совпадений в выборке объемом k при независимых испытаниях равна

$$1-p=\frac{m(m-1)\dots(m-k+1)}{m^k}=\prod_{i=0}^{k-1}\left(1-\frac{i}{m}\right),$$

откуда в силу неравенства $\ln(1-x)<-x$ при $0<x<1$ получаем

$$\ln(1-p)=\sum_{i=0}^{k-1}\ln\left(1-\frac{i}{m}\right)<-\sum_{i=1}^{k-1}\frac{i}{m}=$$

$$=-\frac{k(k-1)}{2m}<-\frac{(k-1)^2}{2m}.$$

Теперь заметим, что при неравномерном распределении в силу леммы 1 имеет место оценка

$$1-p=\sum p_{i_1}p_{i_2}\dots p_{i_k}\le\prod_{i=1}^{k-1}\left(1-\frac{i}{m}\right),$$

где сумма берется по всем наборам различных индексов $i_1,i_2,\dots,i_k$; $1\le i_j\le m$; $1\le j\le k$. Поэтому вероятность отсутствия коллизии в выборке при неравномерном распределении не выше, чем при равномерном, и оценка существования коллизии также остается справедливой.

Как следствие из этого утверждения получаем нижнюю оценку для выборки объемом k из m элементов при заданном значении вероятности p:

$$k>\sqrt{2m\ln\frac{1}{1-p}}+1\approx c_p\sqrt{m},$$

где

$$c_p=\sqrt{2\ln\frac{1}{1-p}}.$$

Таким образом, для поиска коллизий с вероятностью успеха не менее p достаточно перебрать $c_p\sqrt{m}$ случайных элементов исходного множества и сравнить значения их сверток. Например, при $p=1/2$ имеем $c_p\approx 1{,}177$; при $c_p=1$ получаем $p\approx 0{,}39$.

Применяя это утверждение к хеш-функциям, получаем, что для поиска коллизии функции $h \quad X \to Y$ при $|Y| = m = 2^n$ при вероятности успеха p потребуется просмотреть, по крайней мере, $k > c_p\sqrt{2^n}$ элементов множества X.

Другой алгоритм поиска коллизий состоит в следующем: пусть имеется некоторое множество W из r_2 сообщений с известными значениями сверток. Осуществляем независимую генерацию по схеме выборки с возвращением некоторого числа r_1 сообщений, вычисляем значения их сверток и сравниваем получившиеся значения с известными значениями сверток для сообщений множества W. Алгоритм заканчивает работу при получении первого совпадения. В этом случае имеем

$$1 - p = \left(1 - \frac{r_2}{m}\right)^{r_1}$$

откуда

$$\ln(1-p) = r_1 \ln\left(1 - \frac{r_2}{m}\right) < -\frac{r_1 r_2}{m}.$$

Поэтому вероятность успеха p можно оценить неравенством

$$p > 1 - e^{-\frac{r_1 r_2}{m}}$$

Наибольшей эта вероятность становится при $r_1 = r_2 = \sqrt{m}$. В этом случае ее значение приблизительно равно 0,63.

Рассмотрим примеры построения хеш-функций, не зависящих от ключа.

Бесключевые хеш-функции на основе блочных шифров. Пусть E_k — алгоритм блочного шифрования, n — размер блока. Рассмотрим следующие одношаговые сжимающие функции, построенные по правилу (2.1) на основе алгоритма E_k (см. рис. 2.2):

1) $f(x, H) = E_k(x \oplus H)$;
2) $f(x, H) = E_x(H)$;
3) $f(x, H) = E_H(x)$.

Все эти функции не удовлетворяют условию однонаправленности. Например, первая, очевидно, обратима, поскольку ключ k должен быть общеизвестен. Для второй и третьей можно построить вероятностный алгоритм обращения на основе парадокса дней рождений со сложностью $O(2^{n/2})$.

Для построения хеш-функции, удовлетворяющей свойству однонаправленности, рассмотрим функцию, заданную формулой $E'_k(x) = E_k(x) \oplus x$, где E_k — алгоритм блочного шифрования.

Такая функция является однонаправленной по обоим аргументам. Поэтому если вместо E_k взять функцию $E'_k(x)$, то на ее основе также можно построить хеш-функцию по правилу (2.1), определив одношаговую сжимающую функцию одной из следующих формул (см. рис. 2.2, *г* — *е*):

1) $f(x, H) = E'_k(x \oplus H) = E_k(x \oplus H) \oplus x \oplus H$;

2) $f(x, H) = E'_x(H) = E_x(H) \oplus H$ (Davies — Meyer);

3) $f(x, H) = E'_H(x) = E_H(x) \oplus x$.

Заметим, что вторая функция лежит в основе американского стандарта SHA-1 (Secure Hash Algorithm), третья — в основе российского стандарта функции хеширования ГОСТ Р 34.11 — 94.

Если размеры блока и длина ключа не совпадают, то можно рассматривать такие конструкции. Пусть, как и выше, E_k — алгоритм блочного шифрования, для которого n — размер блока, l — размер ключа. Пусть G — некоторое отображение, ставящее в соответствие вектору длиной n вектор длиной l. Тогда вместо функции, изображенной на рис. 2.2, *в*, можно рассмотреть, например, следующие одношаговые сжимающие функции, построенные на основе алгоритма E_k:

4) $f(x, H) = E_{G(H)}(x) \oplus x$ (Matyas — Meyer — Oseas);

5) $f(x, H) = E_{G(H)}(x) \oplus x \oplus H$ (Miyaguchi — Preneel).

Значением любой из хеш-функций, построенных по правилу (2.1) из приведенных одношаговых сжимающих функций, является вектор длиной n, равной размеру блока. В случае если эта величина оказывается недостаточной, ее можно увеличить, заменив одношаговую функцию f на функцию f' с удвоенной размерностью значений. Это можно сделать, например, путем двукратного применения функции f с последующим перемешиванием полублоков согласно формуле

$$f'(x, H_1, H_2) = \pi(f(x, H_1), f(x, H_2)),$$

в которой π переставляет полублоки a, b, c, d по правилу

$$\pi((a, b), (c, d)) = (a, d, c, b).$$

Такой подход, использующий схему, изображенную на рис. 2.2, *б*, реализован в конструкции одношаговой функции в алгоритме MDC-2.

Другие варианты построения хеш-функций на основе схем блочного шифрования подробно рассмотрены в [12, 75].

Семейство алгоритмов MD4. Другие примеры бесключевых хеш-функций дают известные алгоритмы MD4 (Message Digest), MD5, RIPEMD-160 (предложен организацией RIPE)

Таблица 2.1

Алгоритмы семейства MD4

Алгоритм	Число циклов	Число шагов в цикле	Длина блока n
MD4	3	16	128
MD5	4	16	128
RIPEMD-160	5	16	160
SHA-1	4	20	160
SHA-256	64	1	256
SHA-384	80	1	384
SHA-512	80	1	512

и SHA-1 (предложен Агентством национальной безопасности США и введен в действие Национальным институтом стандартов (NIST) США в качестве стандарта SHS (Secure Hash Standard) в 1995 г.). Они оперируют с блоками длиной n, совпадающей с длиной результирующего значения свертки, причем $n = 128$ для алгоритмов MD4, MD5 и $n = 160$ для алгоритмов RIPEMD-160 и SHA-1. Позднее в 2002 г. NIST утвердил еще три хеш-функции с большей длиной блока (табл. 2.1).

Указанные алгоритмы спроектированы специально с учетом эффективной реализации на современных ЭВМ. При их использовании исходное сообщение M разбивают на блоки M_i длиной $m = 512$ бит. В результате получают последовательность блоков $M_1, M_2, \ldots, M_N$. Последний блок формируют путем дописывания к концу сообщения комбинации $10\ldots0$ до получения блока размером 448 бит, к которому затем добавляют комбинацию из 64 бит, представляющую битовую длину сообщения. Затем вычисляют значение свертки согласно процедуре (2.1) с использованием одношаговой сжимающей функции $f(x, H) = E_x(H) \oplus H$ по правилу

$$H_i = f(M_i, H_{i-1}), \quad 1 \le i \le N,$$

где M_i — блок сообщения длиной m бит; H_i — блоки из n бит; E_x — некоторое преобразование множества блоков; значение начального вектора H_0 определяется в описании алгоритма E_x.

Сами преобразования E_x имеют структуру, аналогичную структуре блочного шифра на основе регистра сдвига. Например, в алгоритмах MD4, MD5 использована схема, изображен-

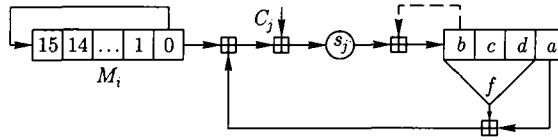

Рис. 2.3. Преобразование E алгоритмов MD4, MD5

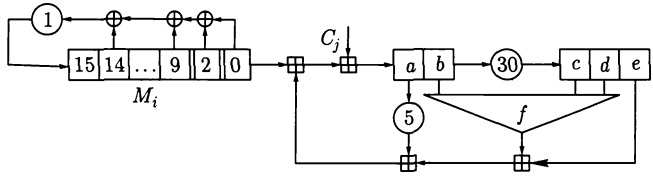

Рис. 2.4. Преобразование E алгоритма SHA-1

ная на рис. 2.3, а в алгоритме SHA-1 — схема, показанная на рис. 2.4. Блок открытого текста длиной 512 бит записывается в накопитель, который работает как циклический регистр сдвига длиной 14 с размером ячейки 32 бита. В каждом такте с него считывается 32 бита и подается на вход нелинейного регистра (длиной 4 для MD и 5 для SHA). Начальный вектор $H_0 = (a, b, c, d)$ записывается во второй регистр в указанном на рис. 2.3 порядке. Заключительное состояние, в которое перейдет второй регистр после выполнения всех циклов, будет определять искомое значение свертки $H_N = (a', b', c', d')$. Для усложнения на каждом шаге j происходит суммирование с некоторыми заранее заданными константами C_j, а также выполняются определенные циклические сдвиги s_j. Функции $f_j(b, c, d)$ меняются определенным образом.

В алгоритме MD4 выполняется 3 цикла по 16 шагов, в которых применяются соответственно функции:

1) $(b \wedge c) \vee (\neg b \wedge d)$;
2) $(b \wedge c) \vee (b \wedge d) \vee (c \wedge d)$;
3) $b \oplus c \oplus d$,

где символами $\vee, \wedge, \neg$ обозначены логические функции покоординатных соответственно дизъюнкции, конъюнкции и отрицания двоичных векторов.

В алгоритме MD5 выполняется 4 цикла по 16 шагов, в которых применяются соответственно функции:

1) $(b \wedge c) \vee (\neg b \wedge d)$;
2) $(b \wedge c) \vee (b \wedge d) \vee (c \wedge d)$;
3) $b \oplus c \oplus d$;
4) $b \oplus (c \vee \neg d)$.

Кроме того, в алгоритме MD5 введена дополнительная обратная связь, показанная на рис. 2.3 пунктиром.

В алгоритме SHA-1 выполняется 4 цикла по 20 шагов, в которых применяются соответственно функции:

1) $(b \wedge d) \vee (c \wedge \neg d)$;
2) $b \oplus c \oplus d$;
3) $(b \wedge c) \vee (b \wedge d) \vee (c \wedge d)$;
4) $b \oplus c \oplus d$.

При этом по сравнению с алгоритмом MD4 изменен как первый регистр, в котором введена линейная обратная связь, так и второй, длина которого увеличена до 5 ячеек (см. рис. 2.4). Значения циклических сдвигов в этом алгоритме фиксированы.

ГОСТ Р 34.11—94. В стандарте хеш-функции ГОСТ Р 34.11—94 приняты значения $n = m = 512$. Одношаговая сжимающая функция $f(x, H)$, используемая для вычисления последовательности значений $H_i = f(M_i, H_{i-1})$, построена на базе 4 параллельно работающих схем блочного шифрования (ГОСТ 28147—89), каждая из которых имеет 256-битовый ключ и оперирует с блоками размером 64 бита. Каждый из ключей вычисляют в соответствии с некоторой линейной функцией от блока исходного сообщения x_i и значения H_{i-1}. Значение H_i является линейной функцией результата шифрования, блока исходного сообщения M_i и значения H_{i-1}. После вычисления значения H_N для последовательности блоков $M_1, M_2, \ldots, M_N$ применяют еще два шага вычисления согласно формуле

$$H = h(M) = f(\Sigma, f(L, H_N)),$$

где Σ — сумма по модулю 2 всех блоков сообщения; L — длина сообщения.

2.4. Хеш-функции на основе дискретного логарифмирования

Приведем один результат, который позволяет строить хеш-функции, удовлетворяющие условию сложности подбора коллизий в предположении сложности задачи дискретного логарифмирования.

Пусть имеется большое простое число p с условием $p = 2q + 1$, где число q также простое. Выбираем два примитивных элемента α и β в кольце Z_p, для которых значение логарифма $\log_\alpha \beta$ является секретным. Определим функцию $h \quad Z_q \times Z_q \to Z_p \setminus \{0\}$ равенством $h(x_1, x_2) = \alpha^{x_1}\beta^{x_2} \bmod p$.

Теорема 1. Для данной хеш-функции выполняется условие сложности подбора коллизий в предположении сложности нахождения дискретного логарифма $\log_\alpha \beta$.

Доказательство. Предположим, что найдена коллизия:

$$h(x_1, x_2) = h(x_2', x_2'), \quad (x_1, x_2) \neq (x_2', x_2'),$$

тогда

$$\alpha^{x_1 - x_1'} = \beta^{x_2' - x_2} \bmod p.$$

Пусть $d = \text{НОД}(x_2' - x_2, p-1) \in \{1, 2, q, p-1\}$; НОД — наибольший общий делитель. Рассмотрим возможные варианты.

Если $d = 1$, то с помощью расширенного алгоритма Евклида можно найти обратный элемент $(x_2' - x_2)^{-1} \bmod p$ и вычислить $\log_\alpha \beta = (x_1 - x_1')(x_2' - x_2)^{-1} \bmod p$.

Если $d = 2$, то $\text{НОД}(x_2' - x_2, q) = 1$. Полагаем $y = (x_2' - x_2)^{-1} \bmod q$, тогда $(x_2' - x_2)y = kq + 1$ при некотором $k \in \{0, 1\}$, откуда

$$\beta^{(x_2' - x_2)y} = \beta^{kq+1} = (-1)^k \beta \bmod p$$

и значит

$$\log_\alpha \beta = \begin{cases} (x_2' - x_2)y \bmod p - 1, \\ (x_2' - x_2)y + q \bmod p - 1. \end{cases}$$

Случаи $d = q$, $d = p - 1$ невозможны в силу ограничений $0 \leq x_2, x_2' \leq q - 1$, так как в этом случае

$$-(q-1) \leq x_2' - x_2 \leq q - 1.$$

Теорема доказана.

2.5. Возможные атаки на функции хеширования

Выше указывалось, что во многих случаях хеш-функции строятся на основе одношаговых сжимающих функций. Поэтому имеется тесная связь атак на хеш-функцию с атаками на соответствующую одношаговую сжимающую функцию. В частности, последняя должна обладать практически всеми теми же свойствами, которыми обладает и сама хеш-функция.

Итеративный способ построения хеш-функций позволяет иногда при ее обращении или построении коллизий использовать метод «встречи посередине». Для защиты от этой опасности в конце сообщения обычно дописывают блоки с контрольной суммой и длиной сообщения.

Возможны атаки, использующие слабости тех схем, на базе которых построены хеш-функции. Например, для построения коллизий хеш-функций, основанных на алгоритмах блочного шифрования, можно использовать наличие слабых ключей или свойство дополнения (как это имеет место для алгоритма DES), наличие неподвижных точек (для которых $E_k(x) = x$), коллизии ключей (т. е. пар различных ключей, для которых выполняется равенство $E_k(x) = E_{k'}(x)$) и т. п.

Серьезный прорыв в развитии методов анализа хеш-функций был сделан в 2004 г., когда на конференции «EUROCRYPT 2004» китайские математики привели примеры коллизий для алгоритмов MD4, MD5, HAVAL-128, RIPEMD [89]. Позднее суть метода авторы изложили на конференции «EUROCRYPT 2005» на примере алгоритма MD5 [90]. Метод является развитием разностного (дифференциального) метода анализа схем блочного шифрования и выполняется в четыре этапа. Сначала осуществляют выбор разностей между блоками исходного сообщения, которые позволяют с большой вероятностью получить нужную разность между промежуточными заполнениями регистра. При этом рассматривают два типа разностей: по модулю 2 и по модулю 2^{32}. Затем строят разностную схему соотношений для промежуточных заполнений регистра, позволяющих получить нулевую разность между итоговыми заполнениями. Третий этап посвящен формированию набора соотношений на значения ячеек регистра, выполнение которых позволяет повысить вероятность успешного поиска. В заключение осуществляют поиск пары подходящих сообщений, которые приводят к построению коллизии. Для этого осуществляют последовательный случайный подбор пар 512-битовых блоков с заданными значениями разностей между ними.

Различные варианты метода были реализованы и другими авторами, причем вычисления можно проводить на обычном персональном компьютере в течение нескольких минут (см., например, [81]).

Кроме того, удалось модифицировать метод так, чтобы в коллизии участвовали не случайные наборы данных с фиксированной разностью, а осмысленные тексты. Так, в работе [58] пока-

зано, как изготовить два одинаково подписанных сертификата, содержащих разную информацию.

Контрольные задания

1. В каких целях применяют хеш-функции?

2. Перечислите основные требования, предъявляемые к хеш-функциям.

3. Почему нельзя использовать в качестве криптографических хеш-функций линейные отображения?

4. Сравните требования, предъявляемые к хеш-функциям, задаваемым ключом, и бесключевым хеш-функциям.

5. Можно ли использовать в качестве бесключевой хеш-функцию, задаваемую фиксированным общеизвестным ключом?

6. Покажите, что хеш-функция, изображенная на рис. 2.2, *а*, не удовлетворяет условию однонаправленности.

7. Для хеш-функций, изображенных на рис. 2.2, *б*, *в*, предложите вероятностный алгоритм обращения на основе парадокса дней рождений со сложностью $O(2^{n/2})$.

8. Можно ли использовать в качестве хеш-функции, зависящей от ключа, функцию вида $h_k(M) = h(h(M), k)$?

9. Предложите способ построения одношаговой хеш-функции из хеш-функции на основе дискретного логарифма, описанной в подразд 2.4.

10. Пусть при случайном выборе $x \in X$ значение функции $h \quad X \to Y$, $|Y| = m$, имеет равномерное распределение [23]. Покажите, что при $r \geq 2$:

1) вероятность r-кратного совпадения значений функции h при попарно различных значениях аргумента равна

$$P(H(x_1) = \quad = h(x_r)/x_i \neq x_j, 1 \leq i < j \leq r) = \frac{1}{m^{r-1}};$$

2) среднее число r-кратных совпадений значений функции h в выборке с возвращением объема $N_r = c(m^{r-1})^{1/r}$, $c \geq 1$ равно

$$\binom{N_r}{r} \frac{1}{m^{r-1}} \approx \frac{c^r}{r!}.$$

11. Пусть при случайном выборе x значение $h(x)$ функции $h \quad \{0,1\}^N \to \{0,1\}^n$ $(3 \leq n \leq N)$ имеет равномерное распределение [86]. Покажите, что в выборке $x_1, x_2, \ldots, x_k$ объемом $k = 2^{\lceil \frac{n+1}{2} \rceil}$ при независимых испытаниях вероятность p существования коллизии будет больше $1/2$.

Указание: воспользуйтесь цепочкой неравенств

$$1 - p = \prod_{i=1}^{k-1} \left(1 - \frac{i}{2^n}\right) = \exp\left(-\frac{k(k-1)}{2^{n+1}}\right) < \mathrm{e}^{-3/4} < \frac{1}{2}.$$

12. Используя лемму 1, покажите, что в задании 11 можно отказаться от условия равномерности распределения значений хеш-функции.

13. Пусть при случайном выборе x значение $h(x)$ функции $h \quad \{0,1\}^N \to \{0,1\}^n$ $(3 \le n \le N)$ имеет равномерное распределение [86]. Покажите, что в выборке $x_1, x_2, \ldots, x_k$ объемом $k = 2^{\lfloor \frac{n-t}{2} \rfloor}$ при независимых испытаниях вероятность p существования коллизии будет не больше $1/2^{t+1}$.

Указание: воспользуйтесь цепочкой неравенств

$$p \le \sum_{i=2}^{k-1} \frac{i-1}{2^n} = \frac{k(k-1)}{2^{n+1}} < \frac{k^2}{2^{n+1}} \le \frac{2^{n-t}}{2^{n+1}} = \frac{1}{2^{t+1}}.$$

14. Докажите неравенство Йенсена.

Глава 3

Коды аутентификации

3.1. Определения и свойства

Рассмотрим более подробно свойства хеш-функций, задаваемых ключом. Как правило, такие функции рассматривают в терминах теории кодирования как обобщение кодов, обнаруживающих ошибки, и называют *кодами аутентификации*. На самом деле, понятие кода аутентификации является более широким и не сводится к понятию хеш-функции. Оно охватывает различные случаи кодирования сообщений: не только те, в которых сообщение передается в открытом виде и к нему дописывается код аутентичности сообщения, но и те, в которых обеспечивается закрытие самого сообщения для выполнения свойства конфиденциальности. Более подробно теория кодов аутентификации изложена в книге [5]. Мы ограничимся рассмотрением только понятия кода аутентификации без сокрытия, которое фактически сводится к понятию хеш-функции, задаваемой ключом.

Определение 1. *Кодом аутентификации* (*без сокрытия*) называется четверка (X, A, K, h): X — множество сообщений; A — множество значений кода аутентичности сообщения; K — множество ключей; $h \quad K \times X \to A$ — ключевая хеш-функция.

Протокол аутентификации, решающий задачи обеспечения целостности сообщения при передаче/хранении, а также аутентификации источника данных для случая доверяющих друг другу сторон, основанный на использовании кода аутентификации (называемый также *схемой имитозащиты*), предполагает наличие общего секретного ключа и заключается в передаче одного сообщения (x, a), где $a = h_k(x)$:

$$A \to B \quad (x, a).$$

Протокол аутентификации состоит в следующем: участник A для передачи сообщения x вычисляет проверочное значение — код аутентичности сообщения $a = h_k(x)$, дописывает его к x и

отправляет полученную пару (x, a) участнику B. Получив такое сообщение, участник B проверяет равенство $a = h_k(x)$: если оно выполнено, то принимает его; если — нет, то отвергает.

Напомним, что основными требованиями к хеш-функциям, задаваемым ключом, являются:

- высокая сложность подбора сообщения с правильным значением свертки; это обеспечивает невозможность создания поддельных ложных сообщений при атаках типа «имитация»;
- высокая сложность подбора для заданного сообщения с известным значением свертки другого сообщения с правильным значением свертки; это обеспечивает невозможность модификации передаваемых сообщений при атаках «подмена».

Первое требование означает защищенность от атаки типа «имитация» (impersonation), т. е. от передачи сфабрикованных сообщений в пустом канале в целях навязывания принимающей стороне ложных сообщений, второе — защищенность от атаки типа «подмена» (substitution) передаваемых сообщений.

Вероятности навязывания. Для характеристики неустойчивости протокола аутентификации к перечисленным выше атакам вводят понятие вероятности успешного навязывания. Будем предполагать, что на множестве ключей задано некоторое вероятностное распределение $(p(k); k \in K)$. Обозначим символом $P(x \to a)$ вероятность того, что сообщению x соответствует код аутентичности сообщения a:

$$P(x \to a) = P\{k \quad h_k(x) = a\} = \sum_{\{k \in K : h_k(x) = a\}} p(k).$$

Аналогично символом $P(x \to a, x' \to a')$ обозначим вероятность того, что паре сообщений x, x' соответствует пара кодов аутентичности сообщения a, a'

Символом $P(x' \to a' / x \to a)$ обозначим вероятность того, что сообщению x' соответствует код аутентичности сообщения a', при условии, что сообщению x отвечает код аутентичности сообщения a:

$$P\left(x' \to a' / x \to a\right) = \frac{P(x \to a, x' \to a')}{P(x \to a)}.$$

Теперь определим следующие величины:

$$p_0 = \max_{x,a} P(x \to a),$$

$$p_1 = \max_{x, x' \neq x, a, a'} P(x' \to a' / x \to a).$$

Если на множестве X задать некоторое распределение вероятностей, то можно определить еще одну величину, характеризующую среднее значение максимальной вероятности подмены сообщения (x, a):

$$p_2 = \sum_{x,a} P(x,a) \max_{x' \neq x, a'} P(x' \to a' / x \to a),$$

где $P(x,a) = P(x)P(x \to a)$ — вероятность появления сообщения (x, a) в канале связи; $x \in X$, $a \in A$.

Заметим, что всегда $p_1 \geq p_2$, так как

$$p_2 = \sum_{x,a} P(x,a) \max_{x' \neq x, a'} P(x' \to a' / x \to a) \leq p_1 \sum_{x,a} P(x,a) = p_1.$$

Величина p_0 характеризует максимальную вероятность успешного угадывания противником значения кода аутентичности a для сообщения x.

Величина p_1 характеризует максимальную вероятность успешной подмены противником пары (x, a) на пару (x', a'); $x' \neq x$.

Величина p_2 характеризует среднее значение максимальной вероятности успешной подмены противником сообщения (x, a) на сообщение (x', a'), $x' \neq x$ по всем возможным парам (x, a).

Теорема 2. Для любого кода аутентификации (X, A, K, h) с $|A| = n$ и любого распределения вероятностей на множестве K выполняются свойства:

1) $p_0 \geq 1/n$, $p_1 \geq 1/n$, $p_2 \geq 1/n$;
2) $p_0 = 1/n \Leftrightarrow \forall x, a \; P(x \to a) = 1/n$;
3) $p_1 = 1/n \Leftrightarrow p_2 = 1/n \Leftrightarrow \forall x, x' \neq x, a, a'$
$$P(x' \to a' / x \to a) = 1/n;$$
4) $p_0 = p_1 = 1/n \Leftrightarrow \forall x, x' \neq x, a, a'$
$$P(x \to a, x' \to a') = 1/n^2$$

Доказательство. Нижние оценки (свойство 1) вытекают соответственно из неравенств:

$$1 = \sum_a P(x \to a) \leq n p_0,$$

$$1 = \sum_{a'} P(x' \to a' / x \to a) \leq n p_1,$$

$$p_2 = \sum_{x,a} P(x,a) \max_{x' \neq x, a'} P(x' \to a' / x \to a) \geq$$

$$\geq \sum_{x,a} P(x,a) \;\; 1/n = 1/n.$$

Свойство 2 вытекает из того, что среднее значение совпадает с максимальным только для одинаковых величин. Аналогично для свойства 3 получаем, что $p_1 = 1/n$ в том и только в том случае, если вероятности $P(x' \to o'/x \to a)$ одинаковы.

Наконец, условие $p_2 > 1/n$ равносильно тому, что найдутся $x, x' \neq x, a, a'$ такие, что $P(x' \to a'/x \to a) > 1/n$.

Докажем свойство 4. Если $p_0 = p_1 = 1/n$, то в силу свойств 2, 3 при всех $x, x' \neq x, a, a'$ выполнены равенства

$$P(x, a) = P(x' \to a'/x \to a) = 1/n.$$

Отсюда имеем

$$P(x \to a, x' \to a') = P(x, a)P(x' \to a'/x \to a) = 1/n^2$$

Обратное очевидно. Теорема доказана.

Таким образом, кодами аутентификации с наилучшими свойствами будут те, для которых выполнено свойство 4 теоремы 2. Назовем такие коды аутентификации *оптимальными*.

3.2. Ортогональные массивы

Для характеристики и построения оптимальных кодов аутентификации нам потребуется понятие ортогонального массива.

Определение 2. Пусть $n \geq 2$, $m \geq 2$, $\lambda \geq 1$. *Ортогональным массивом* $OA(n, m, \lambda)$ называют таблицу, имеющую λn^2 строк и m столбцов, в клетки которой вписаны элементы множества A, $|A| = n$, такую, что для любых двух столбцов каждая из пар (a, a'), $a, a' \in A$ встречается ровно в λ строках.

Лемма 2. Справедливы свойства:

1) если приписать один над другим два ортогональных массива $OA(n, m, \lambda_1)$ и $OA(n, m, \lambda_2)$, то получится ортогональный массив $OA(n, m, \lambda_1 + \lambda_2)$;

2) если удалить из ортогонального массива $OA(n, m, \lambda)$ t столбцов $(1 \leq t \leq m - 2)$, то получится ортогональный массив $OA(n, m - t, \lambda)$;

3) если переставить строки (столбцы) ортогонального массива, то в результате получится ортогональный массив с теми же параметрами;

4) если в некоторой колонке ортогонального массива осуществить замену элементов по некоторой подстановке, то в

результате получится ортогональный массив с теми же параметрами.

Доказательство леммы 2 следует непосредственно из определения 2.

Теорема 3. Если существует ортогональный массив $OA(n, m, 1)$, то $m \leq n + 1$.

Доказательство. Пусть $A = \{0, 1, \quad , n\}$. В силу свойства 4 леммы 2 можно считать, что в ортогональном массиве первая строка состоит из одних нулей. Поскольку $\lambda = 1$, в остальных строках содержится не более одного нуля. Тогда число строк, содержащих нулевой элемент, составит $1 + m(n-1) \leq n^2$, откуда и вытекает требуемое неравенство.

Пример 3.1. При простом p ортогональный массив $OA(p, p, 1)$ можно построить следующим образом: на пересечении строки с номером (i, j), $0 \leq i, j \leq p-1$ и столбца с номером x, $0 \leq x \leq p-1$ ставим элемент $a = ix + j \bmod p$. Действительно, система

$$\begin{cases} a = ix + j' \\ a' = ix' + j \end{cases}$$

разрешима относительно (i, j), $0 \leq i, j \leq p-1$ при любых $0 \leq a, a' \leq \leq p-1$.

Теорема 4. Если существует ортогональный массив $OA(n, m, \lambda)$, то

$$\lambda \geq \frac{m(n-1)+1}{n^2}.$$

Доказательство. Как и в случае теоремы 3, можно считать, что $A = \{0, 1, \quad , n\}$ и в ортогональном массиве первая строка состоит из одних нулей. Подсчитаем число N пар $(0, 0)$, содержащихся в оставшихся строках. С одной стороны, поскольку для любых двух столбцов каждая пара элементов встречается ровно в λ строках, получаем $N = (\lambda - 1)m(m-1)$. С другой стороны, если обозначить через x_i число нулей в i-й строке, то

$$N = \sum_{i=2}^{\lambda n^2} x_i(x_i - 1) = \sum_{i=2}^{\lambda n^2} x_i^2 - \sum_{r=2}^{\lambda n^2} x_i = \sum_{i=2}^{\lambda n^2} x_i^2 - m(\lambda n - 1),$$

так как в любом столбце каждый элемент встречается ровно λn раз.

Для оценки первого слагаемого воспользуемся неравенством Йенсена. Заметим, что функция $f(x) = -x^2$ непрерывна и выпукла вверх, откуда следует, что при $t = \lambda n^2 - 1$ выполняется неравенство

$$\frac{1}{t}\sum_{i=1}^{t} x_i^2 \geq \left(\frac{1}{t}\sum_{i=1}^{t} x_i\right)^2$$

поэтому

$$(\lambda - 1)m(m-1) + m(\lambda n - 1) \geq \frac{(m(\lambda n - 1))^2}{\lambda n^2 - 1},$$

или

$$[(\lambda - 1)(m-1) + \lambda n - 1](\lambda n^2 - 1) \geq m(\lambda n - 1)^2$$

Раскрывая скобки и приводя подобные члены, получаем

$$\lambda^2 n^2 (n-1) \geq \lambda[m(n-1)^2 + n - 1],$$

откуда

$$\lambda n^2 \geq m(n-1) + 1.$$

Теорема доказана.

Пример 3.2. При простом p и $d \geq 2$ ортогональный массив $OA\left(p, \frac{p^d - 1}{p-1}, p^{d-2}\right)$ можно построить следующим образом: пусть $\mathbf{Z}_p^d$ — векторное пространство размерности d над полем $\mathbf{Z}_p$. На пересечении строки номером $k = (k_1, \ldots, k_d) \in \mathbf{Z}_p^d$ и столбца номером $x = (0, \ldots, 0, 1, x_{j+1}, \ldots, x_d)$, $0 \leq j \leq d-1$ ставим элемент

$$a = (k, x) = \sum_{i=1}^{d} k_i x_i.$$

Поскольку построенное таким способом множество столбцов помечено попарно линейно независимыми векторами пространства $\mathbf{Z}_p^d$, то система

$$\begin{cases} (k, x) = a \\ (k, x') = a' \end{cases}$$

разрешима относительно k при любых $x \neq x', a, a'$ Число строк будет равно p^d, столбцов —

$$m = 1 + p + p^2 + \quad + p^{d-1} = \frac{p^d - 1}{p-1}.$$

Поэтому $n = p$ и $\lambda = p^{d-2}$. (При $n = p = 2$ имеем $m = 2^d - 1$.)

3.3. Характеристика оптимальных кодов аутентификации

Из приведенных выше определений вытекает, что ортогональный массив $OA(n, m, \lambda)$ задает код аутентификации с равномерным распределением на множестве ключей, имеющий вероятности $p_0 = p_1 = p_2 = 1/n$.

При $\lambda = 1$ верно и обратное утверждение.

Теорема 5. Пусть (X, A, K, h) — код аутентификации; $|A| = n$, $|X| = m$, $p_0 = p_1 = 1/n$, тогда: 1) $|K| \geq n^2$; 2) $|K| = n^2$ — в том и только в том случае, если табличное задание функции h представляет собой ортогональный массив $OA(n, m, 1)$, а распределение на множестве ключей — равномерное (D. P. Stinson, 1990).

Доказательство. Согласно свойству 4 теоремы 2 при всех $x, x' \neq x, a, a'$ имеем

$$P(x \to a, x' \to a') = 1/n^2 > 0.$$

Поэтому каждое множество $K_{x,x',a,a'}$, состоящее из ключей k с условием $h_k(x) = a$ и $h_k(x') = a'$, не пусто, а значит

$$|K| = \sum_{a,a'} |K_{x,x',a,a'}| \geq n^2$$

Если $|K| = n^2$, то каждое множество $K_{x,x',a,a'}$ содержит ровно один ключ, откуда следует, что распределение вероятностей на множестве ключей равномерно.

Теорема доказана.

При $\lambda \geq 1$ справедлива теорема 6 (D. R. Stinson, 1992).

Теорема 6. Пусть (X, A, K, h) — код аутентификации; $|A| = n$, $|X| = m$, $p_0 = p_1 = 1/n$, тогда: 1) $|K| \geq m(n-1)+1$; 2) $|K| = m(n-1)+1$ — в том и только в том случае, если табличное задание функции h представляет собой ортогональный массив $OA(n, m, \lambda)$ при

$$\lambda = \frac{m(n-1)+1}{n^2},$$

а распределение на множестве ключей — равномерное:

$$p(k) = \frac{1}{m(n-1)+1}, \quad k \in K.$$

Доказательство. Докажем* свойство 1. Пусть $|K| = l$. Определим матрицы $B = \left(b_{(x,a),k}\right)_{mn\times l}$ и $B' = \left(b'_{(x,a),k}\right)_{mn\times l}$ следующим образом:

$$b_{(x,a),k} = \begin{cases} p(k), & h_k(x) = a, \\ 0, & h_k(x) \neq a, \end{cases}$$

$$b'_{(x,a),k} = \begin{cases} 1, h_k(x) = a, \\ 0, h_k(x) \neq a. \end{cases}$$

Тогда $C = B \quad B'^T$ — квадратная матрица размера $mn \times$ $\times\, mn$, причем

$$l = |K| \geq \operatorname{rang} B' \geq \operatorname{rang} C.$$

Покажем, что ранг матрицы C равен $m(n-1)+1$. Имеем

$$c_{(x,a),(x',a')} = \sum_k b_{(x,a),k} b'_{(x,a),k} = \sum_{k:h_k(x)=a, h_k(x')=a'} p(k) =$$

$$= \begin{cases} 1/n^2, & x \neq x', \\ 1/n, & x = x', a = a', \\ 0, & x = x', a \neq a' \end{cases}$$

Поэтому матрица C имеет блочный вид. Рассматривая матрицу C размера $mn \times mn$ как матрицу размера $m \times m$ над кольцом матриц размера $n \times n$ и используя элементарные преобразования со строками с учетом равенства $G^2 = G$, можно привести ее к блочному верхнетреугольному виду:

$$C = \frac{1}{n} \begin{pmatrix} E & G & & G \\ G & E & & G \\ & & & \\ G & G & & E \end{pmatrix} \sim$$

$$\sim \frac{1}{n} \begin{pmatrix} E & G & G & G \\ 0 & E-G & 0 & 0 \\ & & & \\ 0 & 0 & 0 & E-G \end{pmatrix}$$

где $E_{n\times n}$ — единичная матрица; $G = (1/n)_{n\times n}$.

*Способ доказательства предложен А. А. Нечаевым в 1986 г.

Заметим, что $\operatorname{rang} G = 1$, $G^2 = G$. Поэтому минимальный многочлен матрицы G равен $x(x-1)$ и $G \sim \operatorname{diag}(1, 0, \quad, 0)$.

Следовательно, $(E-G) \sim \operatorname{diag}(0, 1, \ldots, 1)$, $\operatorname{rang}(E-G) = n-1$, откуда

$$\operatorname{rang} C = \operatorname{rang} E + (m-1)\operatorname{rang}(E-G) =$$

$$= n + (m-1)(n-1) = m(n-1) + 1.$$

Докажем свойство 2. Сначала покажем необходимость. Если $l = m(n-1)+1$, то $\operatorname{rang} B' = \operatorname{rang} B = l$. В частности, столбцы матрицы $B' = (B_1^{\downarrow}, \ldots, B_l^{\downarrow})$ линейно независимы. Зафиксируем некоторый ключ $k_0 \in K$ и рассмотрим линейную комбинацию столбцов вида

$$U^{\downarrow} = \sum_{(x,a):h_{k_0}(x)=a} \left(\sum_{k:h_k(x)=a} p(k) B_k^{\downarrow} \right)$$

С использованием теоремы 2 получаем, что элемент этого столбца в строке с номером (x', a') равен

$$u_{(x',a')} = \sum_{(x,a):h_{k_0}(x)=a} \left(\sum_{k:h_k(x)=a, h_k(x')=a'} p(k) \right) =$$

$$= \sum_{(x,a):h_{k_0}(x)=a} P(x \to a, x' \to a') = \begin{cases} \dfrac{1}{n} + \dfrac{m-1}{n^2}, & h_{k_0}(x') = a', \\ \dfrac{m-1}{n^2}, & h_{k_0}(x') \neq a' \end{cases}$$

Отсюда следует, что выполнено матричное равенство

$$U^{\downarrow} = \frac{1}{n} B_{k_0}^{\downarrow} + \frac{m-1}{n^2} J^{\downarrow},$$

где $J^{\downarrow}$ — столбец из единиц.

В то же время

$$\sum_k p(k) B_k^{\downarrow} = \frac{1}{n} J^{\downarrow},$$

откуда следует, что для векторов-столбцов выполнено равенство

$$U^{\downarrow} = \sum_{(x,a):h_{k_0}(x)=a} \left(\sum_{k:h_k(x)=a} B_k^{\downarrow} \right) = \frac{1}{n} B_{k_0}^{\downarrow} + \frac{m-1}{n} \sum_k p(k) B_k^{\downarrow}.$$

Поскольку столбцы линейно независимы, коэффициенты при одинаковых столбцах в левой и правой частях должны совпадать.

В частности, для коэффициентов при столбце $B_{k_0}^{\downarrow}$ получаем равенство

$$m\,p(k_0) = \frac{1}{n} + \frac{m-1}{n} p(k_0),$$

откуда

$$p(k_0) = \frac{1}{m(n-1)+1}.$$

Ключ k_0 выбираем как произвольный, поэтому получаем, что распределение на множестве ключей должно быть равномерным.

Из условия $p_0 = p_1 = 1/n$ теперь вытекает (в силу теоремы 2), что для множеств $K_{x,x',a,a'}$, состоящих из ключей k с условием $h_k(x) = a$ и $h_k(x') = a'$, выполнено равенство $|K_{x,x',a,a'}| = l/n^2 = \lambda$ при всех $x \neq x', a, a'$, а значит, табличное задание функции h является ортогональным массивом $OA(n, m, \lambda)$.

Покажем достаточность. Если табличное задание функции h является ортогональным массивом $OA(n, m, \lambda)$, то при равномерном распределении на множестве строк оно должно задавать код аутентификации, для которого при всех $x \neq x', a, a'$ будут выполняться следующие равенства:

$$P(x \to a) = 1/n, \quad P(x' \to a' / x \to a) = 1/n.$$

Теорема доказана.

Контрольные задания

1. Приведите пример оптимального кода аутентификации с неравномерным распределением на множестве ключей.
2. Докажите лемму 2.
3. Приведите пример ортогонального массива $OA(n, m, \lambda)$ при $\lambda > 1$.
4. Покажите, что наборы из n^2 троек $(i, j, L(i, j))$, $1 \leq i, j \leq n$, построенные по таблице латинского квадрата L порядка n, задают ортогональный массив $OA(n, 3, 1)$.

5. Два латинских квадрата L_1 и L_2 называют ортогональными, если все пары $(L_1(i,j), L_2(i,j))$ при различных $1 \leq i,j \leq n$ разные. Покажите, что по набору из m попарно ортогональных латинских квадратов порядка n можно построить ортогональный массив $OA(n, m+2, 1)$.

6. С использованием теоремы 3 объясните, почему не может существовать более $n-1$ ортогональных латинских квадратов порядка n.

7. Покажите, что каждый ортогональный массив $OA(n, n, 1)$ можно дополнить до $OA(n, n+1, 1)$.

Указание. Воспользуйтесь доказательством теоремы 3.

8. Покажите, как построить ортогональный массив $OA(p^m, p^m + \\ + 1, 1)$ на основе поля из p^m элементов.

Схемы цифровых подписей

4.1. Общие положения

Цифровая подпись для сообщения M представляет собой число (точнее — цифровую последовательность фиксированной длины), зависящее от самого сообщения и некоторого ключа, известного только автору подписи. При этом предполагается, что она должна быть легко проверяемой, причем для проверки подписи не требуется знание ключа. При возникновении спорной ситуации, связанной с отказом подписывающего от факта подписи им некоторого сообщения либо с попыткой подделки подписи, третья сторона (арбитр) должна иметь возможность разрешить спор.

Для вычисления и проверки цифровой подписи необходимы два алгоритма, которые образуют схему цифровой подписи.

Определение 3. Пусть $X \subseteq A^*$ — множество исходных сообщений, $S = V_n$ — множество значений цифровой подписи, K — множество ключей. *Схемой цифровой подписи* будем называть набор $(X, S, K, \mathrm{Sig}, \mathrm{Ver})$, в котором:

- $\mathrm{Sig} \quad X \times K \to S$ — алгоритм формирования (вычисления) цифровой подписи;
- $\mathrm{Ver} \quad X \times S \times K \to \{0, 1\}$ — алгоритм проверки цифровой подписи.

При этом должно выполняться условие: если $M \in X$, $s \in S$, $k \in K$, то $\mathrm{Ver}_k(M, s) = 1$ в том и только в том случае, если $\mathrm{Sig}_k(M) = s$.

Заметим, что алгоритм Sig_k должен быть секретным, чтобы исключить возможность вычисления правильного значения подписи каждого пользователя без использования его секретного ключа, а алгоритм Ver_k — открытым для гарантирования возможности проверки подписи любым из участников без знания какой-либо секретной информации. Поэтому для реализации схемы цифровой подписи, как правило, либо используют си-

стемы с открытыми ключами, что приводит к необходимости создания инфраструктуры (сертификатов) открытых ключей, либо прибегают к услугам третьей стороны — доверенного центра, выступающего посредником в процедурах подписания и проверки подписи.

Свойства схемы цифровой подписи. Цифровая подпись предназначена для обеспечения аутентификации источника и установления целостности сообщения.

Напомним, что термин *аутентификация источника данных* означает получение подтверждения того, что рассматриваемый документ был создан именно указанным источником информации (без установления времени создания и единственности документа). Как отмечалось выше, обеспечение целостности данных, т. е. невозможности их модификации после создания, можно рассматривать как часть задачи аутентификации источника данных.

Для решения задачи аутентификации источника данных нужно обеспечить невозможность отказа от авторства, т. е. отказа пользователя от факта подписи конкретного сообщения, а также невозможность приписывания авторства, т. е. создания сообщения с правильным значением подписи от имени другого пользователя.

Первую проблему обычно сводят ко второй исходя из следующего предположения: если для схемы цифровой подписи исключена возможность приписывания авторства, то никто кроме автора не мог создать данную пару «сообщение + подпись».

При решении второй проблемы в основном используют сложностной подход, признавая схему надежной в том случае, если задача создания сообщения с правильным значением подписи от имени другого пользователя крайне трудоемка и за реальное время с использованием имеющихся ресурсов ее решить невозможно.

Более детально надежность схемы цифровой подписи оценивается сложностью решения следующих задач:

- *подделка подписи* — нахождение значения подписи под заданным документом лицом, не являющимся владельцем секретного ключа;
- *подделка документа* — модификация подписанного сообщения без знания секретного ключа;
- *подмена сообщения* — подбор двух различных сообщений с одинаковыми значениями подписи без знания секретного ключа;

• *генерация подписанного сообщения* — нахождение хотя бы одного сообщения с правильным значением подписи без знания секретного ключа.

Сравнение цифровой и собственноручной подписей. Использование термина «подпись» в данном контексте оправдано тем, что цифровая подпись имеет много общего с обычной собственноручной подписью на бумажном документе. Собственноручная подпись также решает перечисленные выше задачи, однако между обычной и цифровой подписями имеются существенные различия (табл. 4.1).

Инфраструктура (сертификатов) открытых ключей. Принципиальной сложностью, возникающей при использовании цифровой подписи на практике, является необходимость создания *инфраструктуры* (*сертификатов*) *открытых ключей.* Дело в том, что для алгоритма проверки подписи необходима дополнительная открытая информация, связанная с обеспечением возможности открытой проверки подписи и зависящая от секретного ключа автора подписи. Эту информацию обычно назы-

Таблица 4.1

Основные различия между собственноручной и цифровой подписями

Собственноручная подпись	Цифровая подпись
Не зависит от подписываемого текста; всегда одинакова	Зависит от подписываемого текста; практически всегда разная
Неразрывно связана с подписывающим лицом, однозначно определяется его психофизическими свойствами; не может быть утеряна	Определяется секретным ключом, принадлежащим подписывающему лицу; может быть утеряна владельцем
Неотделима от носителя (бумаги), поэтому отдельно подписывается каждый экземпляр документа	Легко отделима от документа, поэтому верна для всех его копий
Не требует для реализации дополнительных механизмов	Требует дополнительных механизмов, реализующих алгоритмы ее вычисления и проверки
Не требует создания поддерживающей инфраструктуры	Требует создания доверенной инфраструктуры сертификатов открытых ключей

вают *открытым ключом цифровой подписи.* Для исключения возможности подделки этой информации (открытого ключа) лицами, которые хотят выступить от лица законного владельца подписи (секретного ключа), создается инфраструктура, состоящая из центров доверия, называемых *центрами сертификации* (certification authority), обеспечивающих аутентичность открытых ключей путем придания им сертификатов ключей, заверенных цифровой подписью и подтверждающих достоверность соответствия данной открытой информации заявленному владельцу, в целях обнаружения подлога.

Иногда для придания большей гибкости дополнительно создают *центры регистрации* (registration authority) — центры доверия, которые работают вместе с центром сертификации, выполняя роль местного автономного центра хранения реестра сертификатов ключей. Основные функции центра регистрации — регистрация пользователей в системе и присвоение им уникальных идентификаторов, оптимизация управления реестром сертификатов при большом числе запросов, позволяющая масштабировать систему управления сертификатами для большого числа пользователей на большой территории, одновременно сдвигая процесс подтверждения ближе к пользователям.

Электронная цифровая подпись. В соответствии с Федеральным законом от 10 января 2002 г. № 1-ФЗ «Об электронной цифровой подписи» введено понятие *электронной цифровой подписи* и определены условия, при которых такая подпись в электронном документе юридически равнозначна собственноручной подписи в документе на бумажном носителе: «Электронная цифровая подпись в электронном документе равнозначна собственноручной подписи в документе на бумажном носителе при одновременном соблюдении следующих условий: сертификат ключа подписи, относящийся к этой электронной цифровой подписи, не утратил силу (действует) на момент проверки или на момент подписания электронного документа при наличии доказательств, определяющих момент подписания; подтверждена подлинность электронной цифровой подписи в электронном документе; электронная цифровая подпись используется в соответствии со сведениями, указанными в сертификате ключа подписи».

В законе также определено понятие *удостоверяющий центр* как юридическое лицо, выполняющее следующие функции:

- изготовление сертификатов открытых ключей электронных цифровых подписей;

- создание ключей электронных цифровых подписей по обращению участников сети с гарантией сохранения в тайне секретного ключа подписи;
- приостановление и возобновление действия сертификатов открытых ключей электронных цифровых подписей, а также их аннулирование;
- ведение реестра сертификатов ключей электронных цифровых подписей, обеспечение его актуальности и возможности свободного доступа к нему участников информационных систем;
- проверка уникальности открытых ключей электронных цифровых подписей в реестре сертификатов открытых ключей электронных цифровых подписей и архиве удостоверяющего центра;
- выдача сертификатов открытых ключей электронных цифровых подписей в форме документов на бумажных носителях и (или) в форме электронных документов с информацией об их действии;
- осуществление по обращениям пользователей сертификатов открытых ключей электронных цифровых подписей подтверждения подлинности электронной цифровой подписи в электронном документе в отношении выданных им сертификатов открытых ключей электронных цифровых подписей.

Создание программно-аппаратных комплексов, выполняющих функции удостоверяющих центров, с технической точки зрения не представляет большой сложности. Они строятся во многом аналогично центрам сертификации, которые используются в криптографических системах с открытыми ключами. Однако с юридической точки зрения здесь имеется множество проблем. Дело в том, что в случае возникновения споров, связанных с отказом от авторства или с подделкой электронной цифровой подписи, такие центры должны нести юридическую ответственность за достоверность выдаваемых сертификатов. В частности, они должны возмещать понесенные убытки в случае конфликтных ситуаций, когда алгоритм проверки подписи подтверждает ее правильность. Кроме того, такие центры неявно выступают в качестве третьей стороны в двусторонних соглашениях, поэтому они должны нести обязательства и по своевременности проверки и отзыва сертификатов. Еще одной трудностью является то, что юридическое «равноправие» удостоверяющих центров приводит к тому, что инфраструктура удостоверяющих центров должна быть двухуровневой: наверху федеральный уполномоченный орган исполнительной власти, осуществляющий заве-

рение подписей уполномоченных лиц удостоверяющих центров, ниже — все остальные удостоверяющие центры. В то же время при большой территориальной удаленности и разнообразии областей применения более удобной является многоуровневая архитектура.

В связи с этим сложилась практика заключения договоров между участниками информационного взаимодействия с применением электронных цифровых подписей. В таком договоре должно быть четко указано:

- кто должен нести ответственность в случае, если подписанные сделки не состоятся;
- кто должен нести ответственность в случае, если система окажется ненадежной и будет взломана, т. е. будет выявлен факт подделки секретного ключа;
- какова ответственность уполномоченного по выдаче сертификатов органа в случае, если открытый ключ будет сфальсифицирован;
- какова ответственность владельца секретного ключа в случае его утраты;
- кто несет ответственность за плохую реализацию системы в случае повреждения или разглашения секретного ключа;
- каков порядок и каковы сроки проверки правильности значений подписи и сертификатов;
- каков порядок разрешения споров и т. п.

Поскольку данные проблемы носят юридический, а не технический характер, для их разрешения нужен юридически правильно заключенный договор, оформленный стандартным образом на бумаге, который обязаны заключить все пользователи системы электронной цифровой подписи.

Электронная подпись. Заметим также, что в международных документах и правовых актах часто употребляется термин *электронная подпись*, причем он означает электронную идентификационную информацию, добавляемую в качестве атрибута к электронному документу, позволяющую идентифицировать физических лиц путем сличения их подписей. Этот термин охватывает различные технологии, в том числе использующие биометрические характеристики, временны́е и физические характеристики процесса собственноручной подписи, цифровые подписи, электронные ключи, пластиковые карты и др. Основное назначение — идентификация пользователя (в информационной системе), подтверждение целостности подписываемого документа, а также обеспечение невозможности отказа от факта подписи.

В настоящее время всем трем целям удовлетворяет только технология цифровой подписи.

Подходы к построению схем цифровой подписи. В настоящее время предложено несколько принципиально различных подходов к созданию схем цифровой подписи; их можно разделить на три группы:

1) схемы на основе систем шифрования с открытыми ключами;

2) схемы со специально разработанными алгоритмами вычисления и проверки подписи;

3) схемы на основе симметричных систем шифрования.

Далее эти схемы будут подробно рассмотрены.

4.2. Цифровые подписи на основе систем шифрования с открытыми ключами

Возможность использования систем шифрования с открытыми ключами для построения систем цифровой подписи фактически заложена в самой постановке задачи. Действительно, пусть имеется пара преобразований (E, D), первое из которых зависит от открытого ключа, второе — от секретного. Для того чтобы вычислить цифровую подпись s для сообщения, владелец секретного ключа может применить к сообщению M второе преобразование D: $s = D(M)$. В таком случае вычислить подпись может только владелец секретного ключа, в то время как проверить равенство $E(s) = M$ может каждый. Основными требованиями к преобразованиям E и D являются:

1) выполнение равенства $M = E(D(M))$ для всех сообщений M;

2) невозможность вычисления значения $D(M)$ для заданного сообщения M без знания секретного ключа.

Отличительной особенностью предложенного способа построения цифровой подписи является возможность отказа от передачи самого подписываемого сообщения M, так как его можно восстановить по значению подписи. В связи с этим подобные системы называют *схемами цифровой подписи с восстановлением текста.*

Заметим, что если при передаче сообщение дополнительно шифруется с помощью асимметричного шифра, то пара преобразований (E, D), используемая в схеме цифровой подписи, должна отличаться от той, которая используется для шифро-

вания сообщений. В противном случае появляется возможность передачи в качестве шифрованных ранее подписанных сообщений. При этом целесообразнее шифровать подписанные данные, чем, наоборот, — подписывать шифрованные данные, поскольку в первом случае противник получит только шифрованный текст, а во втором — и открытый, и шифрованный тексты.

Очевидно, что рассмотренная схема цифровой подписи на основе пары преобразований (E, D) удовлетворяет требованию невозможности подделки, в то время как требование невозможности создания подписанного сообщения не выполнено: для любого значения s каждый может вычислить значение $M = E(s)$ и тем самым получить подписанное сообщение. Требование невозможности подмены сообщения заведомо выполняется, так как преобразование взаимно однозначно.

Для защиты от создания злоумышленником подписанного сообщения можно применить некоторое взаимно однозначное отображение $R \quad M \mapsto \tilde{M}$, вносящее избыточность в представление исходного сообщения, например, путем увеличения его длины, а затем уже вычислять подпись $s = D(\tilde{M})$. В этом случае злоумышленник, подбирая подпись s и вычисляя значения $\tilde{M} = E(s)$, будет сталкиваться с проблемой отыскания таких значений $\tilde{M}$, для которых существует прообраз M. Если отображение R выбрано таким, что число возможных образов $\tilde{M}$ значительно меньше числа всех возможных последовательностей той же длины, то задача создания подписанного сообщения будет сложной.

Другой подход к построению схем цифровых подписей на основе систем шифрования с открытым ключом состоит в использовании бесключевых хеш-функций. Для заданного сообщения M сначала вычисляют значение хеш-функции $h(M)$, а затем уже значение подписи $s = D(h(M))$. Ясно, что в таком случае по значению подписи уже нельзя восстановить сообщение. Поэтому подписи необходимо передавать вместе с сообщениями. Такие подписи получили название *цифровых подписей с дополнением*. Заметим, что системы подписи, построенные с использованием бесключевых хеш-функций, заведомо удовлетворяют всем требованиям, предъявляемым к цифровым подписям. Например, невозможно создание сообщения с известным значением подписи, поскольку бесключевая хеш-функция должна быть однонаправленной.

В качестве системы шифрования с открытыми ключами можно использовать, например, систему RSA.

4.3. Цифровые подписи на основе специально разработанных алгоритмов

Цифровая подпись Фиата — Шамира. Рассмотрим подход к построению схемы цифровой подписи, основанный на сложности задачи факторизации больших чисел. Сначала заметим, что задачи факторизации и извлечения квадратного корня в кольце вычетов по составному модулю полиномиально сводимы одна к другой.

Рассмотрим случай двух простых сомножителей $n = pq$. Если мы знаем факторизацию числа n, то с учетом равносильности сравнения $x^2 \equiv y(\mathrm{mod}\, n)$ системе сравнений

$$\begin{cases} x^2 \equiv y(\mathrm{mod} p), \\ x^2 \equiv y(\mathrm{mod} q) \end{cases}$$

можно решить каждое из этих уравнений отдельно, а затем вычислить значение x, пользуясь китайской теоремой об остатках. Для решения сравнения $x^2 \equiv y(\mathrm{mod} p)$ можно воспользоваться следующим подходом. Находим факторизацию числа $p - 1 = 2^k h + 1$, $(2, h) = 1$. Пусть α — примитивный элемент $\mathbf{Z}_p^*$. Тогда элемент $\beta = \alpha^h$ имеет порядок 2^k Найдем s, t такие, что $2s + ht = 1$, и элемент $r \bmod 2^k$, удовлетворяющий сравнению $y^{ht} \equiv (\beta^2)^r \bmod p$. Поскольку $x^2 = y\,(\mathrm{mod}\, p)$, то y^{ht} и β^r лежат в подгруппе порядка 2^{k-1} и такое r существует. (Его можно находить последовательно начиная с младшего разряда в разложении $r = r_0 + 2r_1 + 2^2 r_2 + \ldots + 2^{k-2} r_{k-2}$, возводя в соответствующую степень левую и правую части исходного сравнения.) Теперь решением будет число $a = y^s \beta^r \bmod p$.

Наоборот, если известны решения сравнения, то, выбирая среди них два таких, что

$$\begin{cases} a^2 \equiv b^2(\mathrm{mod} n), \\ (a \pm b) \bmod n \neq 0, \end{cases}$$

получаем, что произведение $(a + b)(a - b)$ делится на число n, но ни один из сомножителей $(a + b)$ и $(a - b)$ на n не делится. Тем самым, вычисляя наибольшие общие делители $(a + b, n)$ и $(a - b, n)$, можно факторизовать число n.

Идея построения схемы цифровой подписи принадлежит А. Фиату и А. Шамиру (A. Fiat, A. Shamir). Приведем одну из модификаций схемы, предложенную ими совместно с

В. Фейджем (V. Feige), в которой реализуется цифровая подпись с дополнением.

Пусть h — некоторая хеш-функция, преобразующая исходное сообщение в битовую строку длиной m. Выберем различные простые числа p и q и положим $n = pq$. В качестве секретного ключа каждый участник должен сгенерировать m различных случайных чисел $a_1, a_2, \ldots, a_m \in \mathbf{Z}_n$; $(a_i, n) = 1$; $i = 1, \ldots, m$. Открытым ключом объявляется набор чисел $b_1, b_2, \ldots, b_m \in \mathbf{Z}_n$, где $b_i = (a_i^{-1})^2 \bmod n$; $i = 1, \quad , m$.

Алгоритм вычисления цифровой подписи для сообщения M состоит из следующих действий:

1) выбрать случайное число r, $1 \leq r \leq n-1$;

2) вычислить $u = r^2 \bmod n$;

3) вычислить $h(M, u) = s = (s_1, s_2, \ldots, s_m)$;

4) вычислить $t = r \prod_{i=1}^{m} a_i^{s_i} \bmod n$;

5) подписью для сообщения M считать пару (s, t).

Алгоритм проверки цифровой подписи состоит из следующих действий:

1) по открытому ключу $b_1, b_2, \ldots, b_m$ и значению t вычислить

$$w = t^2 \prod_{i=1}^{m} b_i^{s_i} \bmod n;$$

2) вычислить $h(M, w) = s'$;

3) проверить равенство $s = s'$

Заметим, что алгоритм Sig носит вероятностный характер, так как при разных r получают разные значения подписи.

Достоинствами описанной схемы являются возможность выработки цифровых подписей для нескольких различных сообщений с использованием одного секретного ключа, а также сравнительная простота алгоритмов вычисления и проверки подписи. Например, для схемы цифровой подписи, основанной на алгоритме RSA, соответствующие алгоритмы требуют выполнения значительно большего числа умножений. Попытка компрометации этой схемы сталкивается с необходимостью решения сложной задачи нахождения квадратных корней по модулю n.

Недостатком схемы является большая длина ключа, которая определяется числом m. Если двоичная запись числа n содержит l знаков, то длина подписи составляет $m + l$ бит, длина секретного ключа — ml бит, а открытого ключа — $(m+1)l$ бит. При

этом необходимо учитывать, что для обеспечения достаточной стойкости данной схемы цифровой подписи числа l и m должны иметь в своей двоичной записи несколько сотен бит.

Цифровая подпись Эль-Гамаля. Рассматриваемая цифровая подпись Эль-Гамаля (T. ElGamal) основана на сложности другой задачи — вычисления значения логарифма в конечном поле.

Пусть p — простое число, α — примитивный элемент поля $\mathbf{Z}_p$. Выберем случайное число a в интервале $1 \le a \le p-2$ и вычислим значение $\beta = \alpha^a \bmod p$. Число a является *секретным ключом*, набор (p, α, β) — *открытым ключом*.

Алгоритм вычисления цифровой подписи для сообщения M состоит из следующих действий:

1) выбрать случайное целое число r, $1 \le r \le p-2$;

2) вычислить $\gamma = \alpha^r \bmod p$;

3) для $x = M$ вычислить $\delta = (x - a\gamma)r^{-1} \bmod (p-1)$;

4) подписью для сообщения M считать пару (γ, δ).

Алгоритм проверки цифровой подписи заключается в проверке сравнения $\beta^\gamma \gamma^\delta \equiv \alpha^x (\mathrm{mod} p)$. Если оно верно, то подпись принимается; если — нет, то отвергается.

Основным достоинством такой схемы цифровой подписи является возможность выработки цифровых подписей для большого числа сообщений с использованием одного секретного ключа. При этом попытка компрометации схемы сталкивается с необходимостью решения сложной математической задачи, связанной с нахождением решений показательных уравнений, в частности с нахождением значения логарифма в поле $\mathbf{Z}_p$.

Сделаем два замечания. Первое касается выбора числа r. Оно должно уничтожаться сразу после вычисления подписи. Действительно, зная число r и значение подписи, легко найти секретный ключ a:

$$a = (x - r\delta)\gamma^{-1} \bmod (p-1).$$

В результате подпись может быть полностью скомпрометирована. Кроме того, число r должно быть действительно случайным и не должно повторяться для различных подписей, полученных при одном значении секретного ключа, так как в случае повторения также можно найти секретный ключ a. Факт повторения легко обнаружить по совпадению значений γ в цифровой подписи.

Второе замечание связано с тем, что реально при вычислении подписи на шаге 3 алгоритма в качестве x целесообразнее

использовать свертку $x = h(M)$, а не само сообщение M. Это защищает схему подписи от возможности подбора сообщений с известным значением подписи. Существует несколько способов такого подбора. Например, если выбрать случайно числа i, j, удовлетворяющие условиям $0 < i < p-1$, $0 < j < p-1$, $(j, p-1) = 1$, и положить

$$\gamma = \alpha^i \beta^j \bmod p,$$
$$\delta = -\gamma j^{-1} \bmod (p-1),$$
$$x = -\gamma ij^{-1} \bmod (p-1),$$

то легко убедиться в том, что пара (γ, δ) является верной цифровой подписью для сообщения $M = x$.

Цифровые подписи семейства Эль-Гамаля. Схема цифровой подписи Эль-Гамаля послужила образцом для построения во многом сходных по своим свойствам схем подписей большого семейства. В их основе лежит проверка сравнения вида

$$\alpha^A \beta^B \equiv \gamma^C (\mathrm{mod} p), \tag{4.1}$$

в котором тройка (A,B,C) совпадает с одной из перестановок чисел $\pm x$, $\pm\delta$, $\pm\gamma$ при некотором выборе знаков (табл. 4.2).

Например, исходная схема Эль-Гамаля получается при $A = x$, $B = -\gamma$, $C = \delta$. На базе схем подписей из этого семейства построены стандарты цифровой подписи США и России (табл. 4.3). Так, в алгоритме DSA, входящем в американский стандарт DSS (Digital Signature Standard), используют значения $A = x$, $B = \gamma$, $C = \delta$, а в российском стандарте ГОСТ Р.34.10—94 — значения $A = -x$, $B = \delta$, $C = \gamma$. Заметим, что с июля 2002 г. в России введен в действие новый стандарт цифровой подписи ГОСТ

Таблица 4.2

Схемы цифровых подписей семейства Эль-Гамаля

A	B	C	Уравнение	Верификация
x	$\pm\gamma$	$\pm\delta$	$x \pm a\gamma = \pm r\delta$	$\alpha^x \beta^{\pm\gamma} = \gamma^{\pm\delta}$
x	$\pm\delta$	$\pm\gamma$	$x \pm a\delta = \pm r\gamma$	$\alpha^x \beta^{\pm\delta} = \gamma^{\pm\gamma}$
γ	$\pm\delta$	$\pm x$	$\gamma \pm a\delta = \pm rx$	$\alpha^\gamma \beta^{\pm\delta} = \gamma^{\pm x}$
γ	$\pm x$	$\pm\delta$	$\gamma \pm ax = \pm r\delta$	$\alpha^\gamma \beta^{\pm x} = \gamma^{\pm\delta}$
δ	$\pm\gamma$	$\pm x$	$\delta \pm a\gamma = \pm rx$	$\alpha^\delta \beta^{\pm\gamma} = \gamma^{\pm x}$
δ	$\pm x$	$\pm\gamma$	$\delta \pm ax = \pm r\gamma$	$\alpha^\delta \beta^{\pm x} = \gamma^{\pm\gamma}$

Таблица 4.3

Исходная схема цифровой подписи Эль-Гамаля и построенные на ее основе цифровые подписи американского (DSS) и российского (ГОСТ Р.34.10—94) стандартов

A	B	C	Уравнение	Верификация	Схема подписи
x	$-\gamma$	δ	$x - a\gamma = r\delta$	$\alpha^x\beta^{-\gamma} = \gamma^\delta$	Эль-Гамаля
x	γ	δ	$x + a\gamma = r\delta$	$\alpha^x\beta^\gamma = \gamma^\delta$	DSS
$-x$	δ	γ	$-x + a\delta = r\gamma$	$\alpha^{-x}\beta^\delta = \gamma^\gamma$	ГОСТ Р.34.10—94

Р.34.10—2001, основанный на вычислениях в группе точек эллиптической кривой над конечным полем, относящийся к семейству схем цифровых подписей Эль-Гамаля.

Еще одним достоинством цифровых подписей семейства Эль-Гамаля является возможность уменьшения длины подписи путем замены пары чисел (γ, δ) на пару чисел $(\gamma \bmod q,\ \delta \bmod q)$, где q — некоторый простой делитель числа $p-1$. При этом проверочное равенство (4.1) следует заменить на следующее модифицированное равенство:

$$(\alpha^A\beta^B \bmod p) \bmod q = \gamma^C \bmod q. \qquad (4.2)$$

Именно так сделано в алгоритме цифровой подписи DSA, принятом в 1994 г.

Цифровая подпись DSA. Приведем алгоритм формирования и проверки цифровой подписи алгоритма DSA (Digital Signature Algorithm), входящего в стандарт DSS, действующий в США с 1994 г. Заметим, что помимо алгоритма DSA стандартом предусмотрено использование алгоритма цифровой подписи RSA, основанного на вычислениях в кольце вычетов по составному модулю, и алгоритма ECDSA, основанного на вычислениях в группе точек эллиптической кривой.

Пусть p — простое число, q — простой делитель числа $p-1$, удовлетворяющие условиям:

$$2^{159} < q < 2^{160}, \quad 2^{511+64t} < p < 2^{512+64t}, \quad 0 \le t \le 8.$$

Выбираем α — элемент поля $\mathbf{Z}_p$, имеющий порядок q. Для этого выбираем случайное число $\theta \in \mathbf{Z}_p \setminus \{0\}$ и вычисляем $\alpha = \theta^{p-1/q}$. Если $\alpha = 1$, выбираем число θ заново.

Выберем случайное число a в интервале $1 \le a \le q-2$ и вычислим значение $\beta = \alpha^a \bmod p$. Число a является секретным ключом, а набор (p, q, α, β) — открытым ключом.

Алгоритм вычисления цифровой подписи для сообщения M имеет вид:

1) выбрать случайное целое число r, $1 \le r \le q-2$;

2) вычислить $\gamma = (\alpha^r \bmod p) \bmod q$;

3) вычислить $x = h(M)$ по алгоритму SHA-1;

4) проверить условие НОД$(x + a\gamma, q) = 1$: если оно не выполнено, то выбрать новое значение r;

5) вычислить $\delta = (x + a\gamma)r^{-1} \bmod q$;

6) подписью для сообщения M считать пару (γ, δ).

Алгоритм проверки цифровой подписи состоит из следующих действий:

1) вычислить $x = h(M, u)$, $e_1 = x\delta^{-1} \bmod q$, $e_2 = \gamma\delta^{-1} \bmod q$;

2) вычислить $u = (\alpha^{e_1}\beta^{e_2} \bmod p) \bmod q$;

3) проверить равенство $\gamma = u$; если оно верно, то подпись принимается; если — нет, то отвергается.

Вероятность того, что на первом шаге появится значение r, для которого на четвертом шаге будет нарушено условие взаимной простоты, равна приблизительно $1/2^q$, так как в силу простоты числа q необратимым будет только значение $x + a\gamma = 0 \bmod q$. Поскольку эта вероятность пренебрежимо мала, шаг 4 алгоритма можно опустить.

Цифровая подпись Шнорра. Схема цифровой подписи Шнорра (C. P. Schnorr) также основана на сложности вычисления значения логарифма в конечном поле.

Пусть p — простое число, q — простой делитель числа $p-1$, α — элемент поля $\mathbf{Z}_p$, имеющий порядок q, h — хеш-функция. Выберем случайное число a в интервале $1 \le a \le q-2$ и вычислим значение $\beta = \alpha^{-a} \bmod p$.

Число a является секретным ключом, набор (p, q, α, β) — открытым ключом.

Алгоритм вычисления цифровой подписи для сообщения M состоит из следующих действий:

1) выбрать случайное целое число r, $1 \le r \le q-2$;

2) вычислить $\gamma = \alpha^r \bmod p$;

3) вычислить $x = h(M, \gamma)$;

4) вычислить $\delta = r + ax \bmod q$;

5) подписью для сообщения M считать пару (x, δ).

Алгоритм проверки цифровой подписи следующий:

1) вычислить $\gamma' = \alpha^{\delta}\beta^{-x} \bmod p$;

2) вычислить $x' = h(M, \gamma')$;

3) проверить равенство $x' = x$; если оно верно, то подпись принимается; если — нет, то отвергается.

Данный вариант схемы цифровой подписи имеет следующее преимущество: при ее использовании (так же как и в случае схемы цифровой подписи Фейджа — Фиата — Шамира) уже нельзя обнаружить повторное использование случайного числа r, проверяя совпадение первых компонент цифровой подписи γ, как это было для схемы Эль-Гамаля. Повторяющиеся значения γ спрятаны в значение хеш-функции. Поэтому для разных сообщений M и M' значения первых компонент подписи $x = h(M, \gamma)$, $x' = h(M', \gamma)$ почти всегда будут различными. Введение простого числа q не упрощает вычислений, но позволяет сократить длину подписи.

4.4. Цифровые подписи на основе симметричных систем шифрования

Рассмотренные системы цифровой подписи, основанные на сложности задач разложения целых чисел на множители, вычисления квадратного корня или логарифмирования в конечных полях, имеют один потенциальный недостаток. Он состоит в возможности построения новых эффективных алгоритмов для решения этих математических задач. Поэтому в реальных схемах ключ выбирают с определенным превышением необходимой длины для обеспечения достаточного запаса стойкости. Это в свою очередь значительно усложняет алгоритмы вычисления и проверки подписи. Поэтому представляется весьма привлекательной задача построения схем цифровой подписи на основе симметричных систем шифрования, свободных от подобных недостатков.

Рассмотрим, например, схему цифровой подписи Диффи — Лампорта на основе симметричных систем шифрования.

Пусть требуется подписать сообщение $M = m_1 m_2 \ldots m_n$; $m_i \in \in \{0, 1\}$; $i = 1, \quad n$. Согласно схеме Диффи — Лампорта подписывающий сначала выбирает n пар случайных секретных ключей:

$$K = [(k_{10}, k_{11}), \quad , (k_{n0}, k_{n1})]$$

для используемой им симметричной системы шифрования, затем n пар случайных чисел:

$$S = [(S_{10}, S_{11}), \ldots, (S_{n0}, S_{n1})],$$

где $S_{ij} \in \{0, 1\}$; $i = 1, \quad , n$; $j = 0, 1$, и вычисляет значения

$$R_{ij} = E_{k_{ij}}(S_{ij}); \quad i = 1, \quad , n; \; j = 0, \, 1.$$

Наборы S и $R = [(R_{10}, R_{11}), \ldots, (R_{n0}, R_{n1})]$ являются открытыми и помещаются в общедоступном месте так, чтобы каждый мог прочитать их, но записать их туда мог бы только автор подписи.

Подпись для сообщения M имеет вид $(k_{1m_1}, \ldots, k_{nm_n})$. Чтобы убедиться в ее правильности, следует проверить равенства

$$R_{ij} = E_{k_{ij}}(S_{ij}); \quad j = m_i; \quad i = 1, \ldots, n.$$

Недостатком этой схемы является слишком большой размер подписи, который может превышать размер самого подписываемого сообщения. Имеется несколько способов избавиться от этого недостатка [2]. Рассмотрим некоторые из них.

Во-первых, можно хранить не $2n$ значений секретных ключей, а лишь один секретный ключ k. Для этого можно воспользоваться, например, следующей схемой формирования последовательности K:

$$k_{ij} = E_k(i, j); \quad j = 0, 1; \quad i = 1, \qquad n;$$

Во-вторых, можно аналогичным образом свернуть набор открытых значений S.

В-третьих, можно подписывать не само сообщение, а его свертку, если воспользоваться какой-либо хеш-функцией. Можно использовать и другие подходы, позволяющие сократить как длину подписи, так и размеры открытого и секретного ключей.

Вместе с тем подобные модификации не устраняют главного недостатка рассматриваемых подписей, состоящего в том, что после одного вычисления подписи часть секретного ключа становится известной. Поэтому рассмотренная схема цифровой подписи является по существу одноразовой.

4.5. Другие протоколы цифровой подписи

Рассмотрим некоторые модификации схемы цифровой подписи, обладающие дополнительными функциональными возможностями.

Схема цифровой подписи вслепую (blind signature scheme). Данная схема представляет собой разновидность протокола цифровой подписи, в котором помимо алгоритмов формирования и проверки подписи предусмотрены функция f преобразования сообщения перед применением алгоритма форми-

рования подписи («затемняющая» функция) и функция g «снятия затемнения». Эти функции для любого сообщения x должны удовлетворять условию: $g(\mathrm{Sig}(f(x))) = \mathrm{Sig}(x)$.

Таким образом, алгоритм формирования цифровой подписи выполняется вслепую в том смысле, что он применяется к специально сформированному сообщению, из которого без знания секретного ключа затемняющей функции нельзя получить никакой информации о том сообщении, которое будет извлечено из него получателем вместе с корректной подписью.

Данный вид схемы цифровой подписи был предложен Д. Шаумом (D. Chaum) в 1983 г. специально для использования в системах электронных платежей с цифровыми деньгами как средство обеспечения анонимности плательщика и неотслеживаемости. Дело в том, что при отсутствии «затемнения» противник, наблюдающий за процедурами снятия цифровых денег плательщиком со счета в банке и отправки их на депозит получателем, может однозначно отследить движение цифровой монеты и установить плательщика. Каждая цифровая монета представляет собой специальное электронное сообщение, обладающее уникальным номером и защищенное от искажений кодом обнаружения ошибок. Чтобы обеспечить свою анонимность, плательщик сначала применяет к монете «затемняющую» функцию, а затем отправляет «затемненную» монету в банк для снятия со своего счета. Банк вслепую подписывает монету и возвращает ему подписанную «затемненную» монету. Получив подписанную банком «затемненную» монету, плательщик снимает «затемнение» и получает корректно подписанную банком монету, которой он может расплатиться с получателем платежа.

В качестве примера Д. Шаум предложил использовать схему цифровой подписи RSA с принадлежащей банку парой открытого (n, e_T) и закрытого (n, d_T) ключей, которая используется для подписи монет одного номинала, и «затемняющую» функцию

$$f(x) = xr^{e_T} \bmod n,$$

где r — случайное число; $1 \le r \le n-1$; $(r, n) = 1$.

После подписания такого сообщения банком

$$f(x) \mapsto f(x)^{d_T} \bmod n$$

«затемнение» снимается плательщиком по формуле

$$g(f(x)^{d_T}) = \frac{f(x)^{d_T}}{r} \bmod n \equiv \frac{x^{d_T} r^{e_T d_T}}{r} \equiv x^{d_T} (\bmod n).$$

Теперь, применяя возведение в степень e_T, убеждаемся в правильности подписи.

Таким образом, протокол формирования подписи имеет вид

$$\begin{aligned} A \leftrightarrow T \quad & f(x), \\ A \leftrightarrow T \quad & f(x)^{d_T} \bmod n, \\ A \rightarrow B \quad & x^{d_T} \bmod n. \end{aligned}$$

Схема конфиденциальной цифровой подписи (undeniable signature scheme). Это схема цифровой подписи, в которой процедура проверки подписи предусматривает участие подписавшего. Поэтому факт подписания того или иного сообщения остается конфиденциальным и может быть установлен только в том случае, если подписывающий дает на это согласие. В схеме конфиденциальной цифровой подписи алгоритм проверки цифровой подписи замещается парой протоколов, с помощью которых подписавший доказывает с нулевым разглашением, что подпись корректна, либо, напротив, некорректна. Как и в других схемах цифровой подписи, споры о подлинности подписей решают с помощью процедуры арбитража.

Схема групповой цифровой подписи (group signature scheme). Это схема цифровой подписи, в которой правом вычисления значения подписи обладают только члены определенной группы участников, каждый из которых обладает своим секретным ключом. Проверка цифровой подписи осуществляется с помощью единственного открытого ключа. Подписавший сообщение член группы остается анонимным. Его анонимность может быть нарушена только в случае необходимости разрешения спорной ситуации.

Нотаризация цифровых подписей. Термин «нотаризация» происходит от слова «нотариус» — лицо, которое, как известно, должно подтверждать правильность копий документов для представления их третьим лицам — арбитрам. Центр нотаризации служит для предотвращения опасности отказа подписавшего документ от своей цифровой подписи в последующем, при изменении обстоятельств. Использование центра нотаризации предполагает наличие третьего лица — арбитра, не входящего в центр нотаризации, но доверяющего ему.

Заметим, что имеется существенная разница между следующими двумя внешне сходными задачами:

- сторона A убеждается, что цифровая подпись S была сделана в момент времени t_0;

• сторона A пытается убедить других в момент времени $t_1 \geq$ $\geq t_0$, что подпись S была действительна в момент времени t_0.

Первая задача может быть решена стандартным способом с использованием симметричной системы шифрования в отсутствие не доверяющих друг другу сторон. Особенностью второй задачи как раз и является наличие не доверяющих друг другу сторон. При этом поскольку цифровой подписи можно доверять только при условии неразглашения секретного ключа, возникает опасность, что подписавший сообщение может намеренно разгласить свой секретный ключ, а затем объявить цифровую подпись поддельной. Предотвратить эту угрозу может центр нотаризации, который функционирует следующим образом: абонент, получивший подпись на документе (или на значении вычисленной от него хеш-функции), представляет его в центр нотаризации. Центр нотаризации проверяет подпись, готовит документ, состоящий из исходного документа, информации, подтверждающей правильность его цифровой подписи, и временно́й метки. Затем подписывает этот составной документ своей цифровой подписью. Через конкретный промежуток времени, который может быть определен для оповещения в случае компрометации секретных ключей, документ, подписанный центром нотаризации, должен признаваться истинным всеми сторонами, доверяющими данному центру, даже в случае объявления о компрометации секретных ключей.

Контрольные задания

1. Что общего между собственноручной и цифровой подписями? Чем они отличаются?

2. Чем отличаются понятия цифровой, электронной и электронной цифровой подписей?

3. Какие задачи позволяет решить цифровая подпись?

4. В чем заключается принципиальная сложность практического применения систем цифровой подписи?

5. Почему в криптографических системах, основанных на открытых ключах, нельзя использовать одинаковые ключи для шифрования и цифровой подписи?

6. Докажите, что указанный выше способ подбора подписанных сообщений для схемы Эль-Гамаля действительно дает верные цифровые подписи.

7. Пользуясь приближенным методом последовательного деления отрезка $[1, n]$ пополам до получения единичного отрезка, покажите, что сложность вычисления квадратного корня m-й степени из натурального числа n разрешима за полиномиальное время.

Глава 5

Протоколы идентификации

5.1. Виды протоколов идентификации

Протокол (*схема*) *идентификации* (identification protocol) — протокол установления подлинности (аутентификации) сторон, участвующих во взаимодействии и не доверяющих друг другу. Заметим, что термин «идентификация» в данном случае использован в качестве краткого названия для отличия от других типов аутентификации: источника, сеанса, сообщения и др.

Одной из основных целей протоколов идентификации является обеспечение контроля доступа к определенным ресурсам, таким как банковские счета, телекоммуникационные каналы, компьютерные программы, базы данных, здания, сооружения и т. д. Поэтому часто результатом успешного выполнения протокола может являться ключ для получения доступа участника к определенному ресурсу.

В то же время протоколы идентификации тесно связаны с протоколами распределения ключей, так как для выполнения свойства аутентификации ключа (G7) протокола распределения ключей необходимо установить подлинность взаимодействующих сторон.

Определение. Протокол идентификации включает двух участников: *доказывающего* участника A, проходящего идентификацию, и *проверяющего* участника B, проверяющего аутентичность доказывающего. Целью протокола является установление того, что проверяемым действительно является участник A. С точки зрения проверяющего B возможными исходами протокола являются либо принятие решения об идентичности доказывающего A, либо завершение протокола без принятия такового решения.

Различают протоколы идентификации с односторонней и взаимной аутентификацией. В последнем случае в результате выполнения протокола обе стороны одновременно проверяют идентичность друг друга.

Как правило, подтверждение подлинности основано на какой-либо уникальной информации, известной обеим сторонам. В зависимости от того, какие уникальные данные используются в протоколе идентификации (запоминаемые, хранимые в некотором устройстве либо присущие самому участнику), протоколы идентификации могут быть разбиты на три большие категории:

1) протоколы, основанные на известной обеим сторонам информации; такой информацией могут быть пароли, личные идентификационные номера PIN (Personal Identification Number), секретные или открытые ключи, знание которых демонстрируется во время выполнения протокола;

2) протоколы, использующие некоторые физические приборы, с помощью которых и проводится идентификация; такими приборами могут быть магнитная или интеллектуальная пластиковая карта, устройство, генерирующее меняющиеся со временем пароли;

3) протоколы, использующие физические параметры, составляющие неотъемлемую принадлежность доказывающего; в качестве таковых могут выступать подписи, отпечатки пальцев, характеристики голоса, геометрия руки, структура сетчатки глаза; протоколы, использующие такие параметры, не носят криптографического характера и не будут далее рассматриваться.

Протоколы идентификации принято разделять на следующие классы:

- протоколы, основанные на паролях (слабая аутентификация) — одношаговые;
- протоколы, использующие технику «запрос — ответ» (сильная аутентификация) — двухшаговые;
- протоколы, основанные на технике доказательства знания — трехшаговые;
- протоколы доказательства знания с нулевым разглашением — многошаговые с гарантией неразглашения секретной информации.

Протоколы идентификации тесно связаны с протоколами цифровой подписи, но проще их. Последние имеют дело с меняющимися по содержанию сообщениями и обычно включают элементы, обеспечивающие невозможность отказа от подписанного сообщения. Для протоколов идентификации содержание сообщения, по существу, фиксировано — это заявление об аутентичности доказывающего A в текущий момент времени.

5.2. Протоколы идентификации, использующие пароли (слабая аутентификация)

Фиксированные пароли. Обычная парольная схема основывается на не зависящих от времени паролях и устроена следующим образом: для каждого пользователя имеется пароль, обычно представляющий собой последовательность длиной от 6 до 10 знаков некоторого алфавита, которую пользователь в состоянии запомнить. Эта последовательность выступает в качестве общего секрета пользователя и системы. Для того чтобы получить доступ к системному ресурсу B (база данных, принтер и т. д.), пользователь A представляет свой идентификатор, который обозначим тоже буквой A, пароль p и прямо или косвенно определяет необходимый ресурс. При этом идентификатор пользователя выступает как заявка на идентификацию, а пароль — как подтверждение этой заявки:

$$A \to B \quad A, p.$$

Различные парольные схемы отличаются между собой по методам хранения парольной информации в системе и методам ее проверки.

В число угроз данной схеме идентификации, допускающих возможность проникновения в систему, входят раскрытие пароля (вне системы) и перехват информации с линий связи (внутри системы). Угрозой является также подбор паролей, в том числе с использованием словарей.

Атаки на фиксированные пароли. Рассмотрим основные виды атак на фиксированные пароли.

Повторное использование паролей. Принципиальной слабостью схем, неоднократно использующих фиксированные пароли, является возможность получения противником этих паролей одним из следующих способов:

- путем просмотра при введении с клавиатуры;
- получением документов, содержащих эти пароли;
- перехватом их из каналов связи, используемых пользователями для связи с системой, или из самой системы, поскольку пароли используются в открытом виде.

Все это дает возможность противнику осуществить доступ к системе, выступая от имени законного пользователя.

Таким образом, фиксированные пароли нельзя использовать при их передаче по незащищенным каналам связи, особенно по эфиру.

Заметим, что при использовании процедуры идентификации на абонентском пункте для получения доступа к ресурсу сервера (например, при идентификации кредитной карточки банкоматом в режиме on-line) системный ответ («идентификация A» или «завершение протокола без идентификации») должен быть так же защищен, как и представляемый пароль. Он должен меняться во времени, чтобы противник не мог использовать системный ответ «идентификация A».

Тотальный перебор паролей. Простейшей атакой противника на систему с фиксированными паролями является перебор всех их возможных вариантов до тех пор, пока истинный пароль не будет найден. Особенно опасна эта атака в режиме off-line, поскольку в этом случае не требуется непосредственного контакта, доказывающего с проверяющим, поэтому число безуспешных попыток проверки пароля в единицу времени не может быть ограничено, как это обычно делается при проверке в режиме on-line.

Эффективность указанной атаки напрямую зависит от числа попыток до обнаружения первого пароля, обеспечивающего доступ в систему или к ее ресурсу, а также от временно́й сложности реализации каждой из таких попыток.

Атаки с помощью словаря. Вместо тотального перебора всех возможных паролей противник может прибегнуть к их перебору в порядке убывания вероятностей их использования. Этот метод обычно эффективен в том случае, если пароли выбирает сам пользователь и, как правило, они являются неравновероятными. Во многих случаях эти пароли содержатся в словарях. Обычные словари содержат, как правило, не более 150 тысяч слов, что намного меньше числа всех всего лишь 6-буквенных паролей.

В этом случае противником может быть реализована так называемая «атака с использованием словаря». Для повышения эффективности атаки противник создает и записывает на диск хешированный файл слов из используемого для выбора паролей словаря. Хешированные значения из файла системных паролей могут быть отсортированы (используя обычные алгоритмы сортировки) и сопоставлены со словами из созданного файла. Атаки с использованием словаря позволяют, как правило, найти не пароль фиксированного пользователя, а один или несколько паролей, обеспечивающих доступ в систему.

Защита от перехвата паролей. Остановимся подробнее на методах хранения паролей в системе. Наиболее очевидным из них является хранение паролей в открытом виде в файле, защищенном от записи и считывания системными средствами. При

обращении пользователя система сравнивает введенный пароль с паролем данного пользователя, хранимым в файле. Недостаток этого метода состоит в том, что пароли не защищены от привилегированных пользователей системы. Аналогичная угроза возникает при передаче пароля по сети при удаленном доступе ресурсу.

Для устранения этого недостатка используют зашифрованные файлы паролей пользователей. Для этого можно использовать непосредственное зашифрование паролей с помощью того или иного криптографического алгоритма:

$$A \to B \quad A, E_k(p),$$

что неудобно, так как при этом необходимо обеспечить конфиденциальность ключа либо вычисление значения хеш-функции пароля

$$A \to B \quad A, h(p).$$

Заметим, что использование шифрования либо хеширования перед передачей пароля по незащищенному каналу связи хотя и защищает сам пароль, но не защищает от возможности вхождения противника в систему путем повторной передачи перехваченного зашифрованного пароля. Для второго варианта противник также может использовать предварительно подготовленный словарь, в котором собраны значения $h(p)$ для ожидаемых значений пароля p. Поэтому для усложнения работы по созданию такого словаря можно немного изменить протокол:

$$A \to B \quad A, h(p, A).$$

Теперь противнику нужно составлять словарь отдельно для каждого идентификатора A.

Усложнение подбора паролей. Поскольку степень защиты парольной системы определяется сложностью подбора паролей, на практике используют целый ряд приемов, позволяющих существенно затруднить возможность подбора. Для этого применяют специальные правила составления паролей. Типичным требованием, предъявляемым при составлении паролей, является ограничение на минимальную длину пароля. Примером может служить требование, чтобы пароль имел длину не менее 9 букв или цифр. Так как сложность проникновения в систему определяется сложностью подбора простейшего из паролей зарегистрированных пользователей, он не должен быть легко

запоминаемым и не должен принадлежать какому-либо специфическому классу. Поэтому другими требованиями могут быть: наличие в пароле хотя бы одного символа из каждого регистра (верхнего, числового, не алфавитно-цифрового и т. д.), а также требование, чтобы пароль не мог быть словом из какого-либо доступного словаря или частью идентификационной информации доказывающего.

Лучшими из возможных паролей являются те, которые выбирают случайно и равновероятно из всех слов в данном алфавите. Такой выбор может быть осуществлен с помощью физического датчика случайных чисел.

Для того чтобы увеличить неопределенность используемых паролей и вместе с тем не нарушить такое их важное качество, как возможность запоминания человеком, часто используют парольные фразы. В этом случае в качестве пароля используют целое предложение вместо короткого слова. Парольные фразы хешируются до фиксированной длины и играют ту же роль, что и обычные пароли. При этом фраза не должна просто сокращаться до фиксированной длины. Идея этого метода состоит в том, что человеку легче запомнить фразу, чем случайную последовательность букв или цифр. Парольные фразы обеспечивают бóльшую безопасность, чем короткие пароли, но требуют большего времени для ввода.

Еще один способ затруднения подбора паролей получил название «подсоленные» пароли. Каждый пароль p дополняют n-битовой случайной последовательностью x, называемой «солью» (она изменяет «вкус» пароля), перед применением однонаправленной функции. Хешированный пароль дополненный «солью» записывают в файл паролей. Когда в последующем пользователь предъявляет пароль, система находит «соль» и применяет однонаправленную функцию к паролю, дополненному «солью»:

$$A \to B \quad A, x, h(p, x, A).$$

Хотя сложность подбора каждого варианта пароля при таком дополнении не изменяется, добавление «соли» усложняет атаку в целях получения хотя бы одного пароля за счет значительного увеличения объема словаря. Это происходит потому, что в словаре должно содержаться 2^n вариантов значений $h(p, x)$ для каждого пароля p и идентификатора A, что влечет за собой высокие требования к памяти, содержащей зашифрованный словарь, и соответственно увеличивает время на его изготовление.

Защита базы данных от компрометации. Предыдущий протокол обеспечивает защиту от подбора пароля противником, прослушивающим канал связи. Вместе с тем он может быть полностью скомпрометирован при хищении базы данных с паролями, находящейся у проверяющего. Действительно, в этой базе должны храниться строки $(A, x, h(p, x, A))$. Поэтому, получив однажды доступ к такой базе, противник в дальнейшем может выступать от имени любого из абонентов.

Для защиты от такой ситуации можно использовать протокол, в котором применяется не одна, а две хеш-функции h и h', первая на передающей стороне, вторая на проверяющей. В своей базе данных проверяющая сторона B хранит записи вида $(A, x, h'(h(p, x, A), A))$. После получения сообщения

$$A \to B \quad A, x, h(p, x, A)$$

проводится вычисление значения $q = h'(h(p, x, A), A)$ и в случае его несовпадения с тем, которое хранится в базе, идентификация отклоняется. Теперь противник, имея базу данных, не сможет подобрать сообщения для правильной идентификации в силу односторонности хеш-функции h'

Личные идентификационные номера. К категории фиксированных паролей относят *личные идентификационные номера.* Их обычно используют в приложениях, связанных с пластиковыми картами, для доказательства того, что данный участник является действительным владельцем карты и как таковой имеет право доступа к определенным системным ресурсам. Ввод личного идентификационного номера требуется во всех случаях, когда используется пластиковая карта. Это обеспечивает дополнительный рубеж безопасности, если карта утеряна или похищена.

В силу исторических обстоятельств (и для удобства пользователей) личный идентификационный номер является цифровым и содержит от 4 до 8 цифр. Для того чтобы обеспечить защиту от его тотального перебора, применяют дополнительные организационные меры. Например, большинство банкоматов при троекратном вводе неправильного номера блокируют кредитную карту. Для ее разблокирования требуется ввести уже более длинный номер. Поскольку люди, как правило, не могут запомнить ключи настолько длинные, чтобы с их помощью можно было бы обеспечить необходимую информационную безопасность системы, часто используется следующая двухступенчатая процедура идентификации пользователя:

$$A \to B \quad A, \mathrm{pin},$$
$$A \to B \quad A, h(p, \mathrm{pin}).$$

Сначала с помощью личного идентификационного номера проверяют личность лица, вводящего пластиковую карту, затем содержащаяся в пластиковой карте дополнительная ключевая информация используется для идентификации его в системе (как действительного владельца пластиковой карты, имеющего определенные права доступа в системе). Таким образом, пользователь, имеющий пластиковую карту, должен помнить только короткий личный идентификационный номер, в то время как более длинный ключ p, содержащийся на карте, обеспечивает необходимый уровень криптографической безопасности при идентификации в системе, использующей незащищенные каналы связи.

Защита от повторного воспроизведения. Рассмотренные выше протоколы не обеспечивают защиту от атаки методом повторной передачи перехваченного сообщения в более позднее время. Один из вариантов защиты от повторного воспроизведения состоит в использовании временно́го параметра t:

$$A \to B \quad A, x, t, h'(h(p, x, A), t).$$

Получив такое сообщение, проверяющая сторона обращается к своей базе данных, в которой хранятся записи $(A, x, h(p, x, A))$, вычисляет значение $q = h'(h(p, x, A), t)$ и в случае его несовпадения с полученным значением отклоняет идентификацию.

Одноразовые пароли. Другой способ повышения надежности идентификации основан на использовании *одноразовых паролей* — паролей, которые могут быть использованы для идентификации только один раз. Такие схемы обеспечивают защиту от противника, использующего перехват паролей из незащищенного канала.

Существует три схемы использования одноразовых паролей:

1) пользователи системы имеют общий список одноразовых паролей, который доставляется по защищенному от перехвата каналу связи;

2) первоначально пользователь и система имеют один общий секретный пароль; во время идентификации, использующей пароль t, пользователь создает и передает в систему новый пароль с номером $(t+1)$, зашифрованный на ключе, полученном из t-го пароля;

3) пользователи системы применяют одноразовые пароли на основе однонаправленной функции.

При выборе третьей схемы пользователь начинает с секретного пароля p и с помощью однонаправленной функции $H(p)$ вырабатывает итерации

$$H(w), H(H(p)), \ldots, \underbrace{H(H(\ldots(H(p)\ldots)))}_{n} = H^n(p).$$

Паролем для i-й идентификации ($1 \leq i \leq n$) является $p_i = H^{n-i}(p)$.

Для того чтобы идентифицировать себя при i-й попытке, участник A передает участнику B строку (A, i, p_i):

$$A \to B \quad A, i, p_i, \quad 1 \leq i \leq n.$$

Участник B проверяет соответствие полученного i номеру попытки и равенство $H(p_i) = p_{i-1}$. Если оба равенства выполнены, участник B идентифицирует A и запоминает (i, p_i) для следующей попытки идентификации.

Заметим, что последний протокол не защищает от активного противника, который перехватывает, сохраняет и блокирует передачу информации от участника A к B для последующей попытки подмены собой пользователя A. Поэтому этот протокол может быть использован только для идентификации пользователя системой, идентичность которой уже установлена.

5.3. Протоколы идентификации, использующие технику «запрос — ответ» (сильная аутентификация)

Случайные последовательности и метки времени. Идея построения криптографических протоколов идентификации типа «запрос — ответ» состоит в том, что доказывающий убеждает проверяющего в своей аутентичности путем демонстрации знания некоторого секрета без предъявления самого секрета. Знание секрета подтверждается выдачей ответов на меняющиеся с течением времени запросы проверяющего. Обычно запрос — это число, выбираемое проверяющим при каждой реализации протокола. Если канал связи контролируется противником, любое допустимое число реализаций протокола идентификации не должно давать противнику возможность извлечения информации, необходимой для последующей ложной идентификации. В таких протоколах обычно используют либо случайные

числа, либо числа из неповторяющихся (обычно возрастающих) последовательностей, либо метки времени. Остановимся подробно на последних двух вариантах.

Числа из неповторяющихся последовательностей используют как уникальные метки сообщений, обеспечивающие защиту от навязывания ранее переданных сообщений. Такие последовательности чисел используют независимо для каждой пары доказывающего и проверяющего. Кроме того, при передаче информации от участника A к B и от участника B к A также используют различные последовательности чисел. Стороны следуют заранее определенной политике по выработке таких последовательностей. Сообщение принимается только тогда, когда его число (метка сообщения) не было использовано ранее (или в определенный предшествующий период времени) и удовлетворяет согласованной политике. Простейшей политикой является такая: последовательность начинается с нуля и каждое следующее число увеличивается на единицу. Менее жесткая политика состоит в том, что принятые числа должны только монотонно возрастать. Это позволяет работать с учетом возможности потери сообщений из-за ошибок в каналах связи. Недостатком метода использования последовательностей чисел является необходимость запоминания информации, касающейся каждого доказывающего и каждого проверяющего, а также невозможность обнаружения сообщений, специально задержанных противником.

Метки времени используют для обеспечения гарантий своевременности и единственности сообщений, а также для обнаружения попыток навязывания ранее переданной информации. Они также могут быть использованы для обнаружения попыток задержки информации со стороны противника.

Протоколы, использующие метки времени, реализуют следующим образом: сторона, направляющая сообщение, снимает показания своих системных часов и криптографически «привязывает» их к сообщению. Получив такое сообщение, вторая сторона снимает показания своих системных часов и сравнивает их с показаниями, содержащимися в сообщении. Сообщение принимается, если его временна́я метка находится в пределах так называемого «временно́го окна» — фиксированного временно́го интервала, выбранного из расчета времени, необходимого для обработки и передачи максимально длинного сообщения и максимально допустимой рассинхронизации часов отправителя и получателя. В отдельных случаях для приема сообщения необходимо, кроме указанного выше, обеспечить выполнение усло-

вия, чтобы та же самая (или более ранняя) метка времени не приходила ранее от того же самого абонента.

Надежность методов, основанных на метке времени, зависит от надежности и точности синхронизации системных часов, что является главной проблемой в таких системах. Преимуществами указанных систем является меньшее число передаваемых для идентификации сообщений (как правило, одно), а также отсутствие требований по сохранению информации для каждой пары участников (как в числовых последовательностях). Временны́е метки в протоколах могут быть заменены запросом, включающим случайное число, и ответом.

«Запрос — ответ» с использованием симметричных алгоритмов шифрования. Механизм реализации идентификации с помощью алгоритмов «запрос — ответ» требует, чтобы доказывающий и проверяющий имели общий секретный ключ. В системах с небольшим числом абонентов такими ключами может быть обеспечена каждая пара корреспондентов заблаговременно. В больших системах установление общего ключа может быть обеспечено путем передачи его по защищенному каналу обоим корреспондентам из доверенного центра либо путем предварительного распределения ключей (см. далее).

Для описания алгоритмов введем следующие обозначения: r_A — случайное число, вырабатываемое A (обычно называемое nonce, сокращение от английского only once); t_A — временна́я метка A; k_{AB} — общий секретный ключ для A и B; E_k — алгоритм шифрования на ключе k; A — идентификатор участника A (для краткости используем тот же символ).

Приведем примеры протоколов.

1. *Односторонняя идентификация с использованием временно́й метки.* Заметим, что в данном случае можно ограничиться только одним сообщением протокола, не используя запросы. Доказывающий A передает проверяющему B свою временну́ю метку и идентификатор, зашифрованные на общем ключе:

$$A \to B \quad E_{k_{AB}}(t_A, B).$$

Проверяющий B, расшифровав данное сообщение, проверяет соответствие допустимому интервалу временно́й метки и совпадение полученного и собственного идентификаторов. Последнее необходимо для того, чтобы не позволить противнику немедленно переадресовать сообщение доказывающему A.

2. *Односторонняя идентификация с использованием случайных чисел.* В данном протоколе проверяющий B, расшифровав

сообщение, проверяет, соответствует ли полученное число случайному числу, отправленному им ранее участнику A; после этого он проверяет, соответствует ли его идентификатор полученному идентификатору:

1) $A \leftarrow B$: r_B;

2) $A \rightarrow B$: $E_{k_{AB}}(r_B, B)$.

Для предотвращения возможности криптоанализа алгоритма E_k с помощью специально подобранных открытых текстов участник A может ввести во второе сообщение свое случайное число так же, как в следующем протоколе.

3. *Взаимная идентификация с использованием случайных чисел.* Эта процедура описывается с помощью следующего протокола:

1) $A \leftarrow B \quad r_B$;

2) $A \rightarrow B \quad E_{k_{AB}}(r_A, r_B, B)$;

3) $A \leftarrow B \quad E_{k_{AB}}(r_A, r_B)$.

Получив сообщение от участника A, проверяющий B расшифровывает его и осуществляет те же проверки, что и в предыдущем протоколе. После этого проверяющий B использует случайное число r для формирования третьего сообщения. Доказывающий A, расшифровав третье сообщение, проверяет соответствие полученных случайных чисел тем, которые использовались ранее в п. 1, 2.

Эти три конструкции реализованы в рассмотренных далее протоколах на основе симметричных алгоритмов шифрования, предложенных в качестве стандартов Международной организацией по стандартизации (International Organization for Standardization — ISO).

Протокол односторонней аутентификации (ISO symmetric key two-pass unilateral authentication protocol):

1) $A \leftarrow B \quad r_B, M_1$;

2) $A \rightarrow B \quad M_3, E_{k_{AB}}(r_B, B, M_2)$.

Поля $M_1 - M_3$ могут содержать произвольные текстовые сообщения.

Протокол взаимной аутентификации двухпроходный (ISO symmetric key two-pass mutual authentication):

1) $A \rightarrow B \quad M_2, E_{k_{AB}}([t_A|r_A], B, M_1)$;

2) $A \leftarrow B \quad M_4, E_{k_{AB}}([t_B|r_B], A, M_3)$.

Данный протокол фактически состоит в двукратном независимом использовании предыдущего протокола. Текстовые поля $M_1 - M_4$ используют для передачи сообщений, которые могут применяться для связывания двух шагов.

Выбор случайного числа r_A или временно́й метки t_A зависит от конкретной реализации.

Протокол взаимной аутентификации трехпроходный (ISO symmetric key three-pass mutual authentication):

1) $A \leftarrow B \quad r_B, M_1$;
2) $A \rightarrow B \quad M_3, E_{k_{AB}}(r_A, r_B, B, M_2)$;
3) $A \leftarrow B \quad M_5, E_{k_{AB}}(r_B, r_A, M_4)$.

В данном протоколе связывание шагов внутри сеанса осуществляется с помощью техники «запрос — ответ».

«Запрос — ответ» с использованием асимметричных алгоритмов шифрования. В протоколе идентификации, построенном на основе системы шифрования с открытым ключом, доказывающий может продемонстрировать владение секретным ключом одним из двух способов:

1) путем расшифрования запроса, зашифрованного на его открытом ключе;

2) путем добавления к запросу своей цифровой подписи.

Рассмотрим эти два способа более подробно.

Протоколы идентификации, не использующие цифровую подпись. Пусть h — некоторая однонаправленная функция; E_A, D_A — алгоритмы соответственно зашифрования и расшифрования абонента A. Первый способ основан на следующем протоколе:

1) $A \leftarrow B \quad h(r), B, E_A(r, B)$;
2) $A \rightarrow B \quad r$.

Проверяющий B выбирает случайное число r, вычисляет $h(r)$ и запрос $c = E_A(r, B)$. Доказывающий A расшифровывает запрос c и проверяет совпадение значений хеш-функции и идентификаторов. В случае несовпадения участник A прекращает протокол. В противном случае участник A посылает случайное число r проверяющему B. Проверяющий B идентифицирует A, если полученное от A число r совпадает с имеющимся у него числом.

Использование однонаправленной хеш-функции предотвращает попытки криптоанализа с помощью выбранного открытого текста.

Протокол NSPK (*Needham — Schroeder Public Key Protocol*). Рассмотрим протокол осуществления взаимной аутентификации, предложенный Р. М. Нидхэмом и М. Д. Шредером в работе [69]:

1) $A \rightarrow B \quad E_B(r_A, A)$;
2) $A \leftarrow B \quad E_A(r_A, r_B)$;
3) $A \rightarrow B \quad E_B(r_B)$.

В данном случае дважды используется техника «запрос — ответ», причем случайные числа спрятаны внутри зашифрованных сообщений.

Как обнаружил Г. Лоу (G. Lowe) в 1995 г. [60], в таком виде криптографический протокол обладает следующим недостатком: нарушитель C, используя свой законный обмен с участниками A и B, может открыть второй сеанс протокола с участником B и при этом, чередуя сообщения обоих сеансов (interleaving attack), может найти случайное число r_B, сгенерированное участником B:

1) $A \to C \qquad E_C(r_A, A)$;
1′) $C(A) \to B \quad E_B(r_A, A)$;
2′) $C(A) \leftarrow B \quad E_A(r_A, r_B)$;
2) $A \leftarrow C \qquad E_A(r_A, r_B)$;
3) $A \to C \qquad E_C(r_B)$;
3′) $C(A) \to B \quad E_B(r_B)$.

В результате нарушитель C может убедить участника B в том, что он выступает от имени участника A.

Протокол NSL (Needham – Schroeder – Long Protocol). Один из вариантов исправления протокола NSPK, предложенный Лонгом, заключается в добавлении идентификатора B во второе сообщение:

1) $A \to B \quad E_B(A, r_A)$;
2) $A \leftarrow B \quad E_A(r_A, r_B, B)$;
3) $A \to B \quad E_B(r_B)$.

Если допустить, что можно заменить одно случайное число (r_B) на другое, полученное конкатенацией двух чисел (r_B, B), и это останется незамеченным (что на практике мало вероятно), то возможна следующая атака:

1) $A \to C(B) \qquad E_B(A, r_A)$;
1′) $C(A) \to B \quad E_B(A, C)$;
2′) $C(A) \leftarrow B \quad E_A(C, r_B, B)$;
1″) $A \leftarrow C \qquad E_A(C, (r_B, B))$;
2″) $A \to C \qquad E_C((r_B, B), r'_A, A)$;
3′) $C(A) \to B \quad E_B(r_B)$.

Здесь использовано три сеанса протокола с чередованием сообщений из разных сеансов (interleaving attack). При этом сообщение 2′ повторено (replay attack) как 1″ но два поля (r_B, B) рассматриваются как одно (r_B) (type flaw attack). В итоге нарушитель C может вычислить значение r_B из сообщения 2″ третьего сеанса и успешно завершить второй сеанс, пройдя аутентификацию B от имени A.

Еще один вариант исправления протокола NSPK состоит в обеспечении невозможности того, чтобы нарушитель в третьем шаге протокола смог вычислить значение r_B путем замены третьего шага протокола на следующий:

3) $A \to B \quad E_{h(r_B)}(B)$,

где h — хеш-функция.

Протокол ISO. Рассмотрим еще один из вариантов протокола взаимной аутентификации сторон, предложенный ISO и основанный на применении открытого шифрования:

1) $A \to B \quad A, r_A$;
2) $A \leftarrow B \quad r_B, A, D_B(r_B, r_A, A)$;
3) $A \to B \quad r'_A, B, D_A(r_A, r'_A, B)$.

В данном виде протокол имеет уязвимость. Как показывает следующий пример атаки [37], противник C может пройти аутентификацию B от имени участника A, при этом участник A ничего не подозревает, а идентификатор B уверен, что связан с A:

1) $C(A) \to B \quad A, r_C$;
2) $C(A) \leftarrow B \quad r_B, A, D_B(r_B, r_C, A)$;
1') $A \leftarrow C(B) \quad B, r_B$;
2') $A \to C(B) \quad r_A, B, D_A(r_A, r_B, A)$;
3) $C(A) \to B \quad r'_B, B, D_A(r_A, r_B, B)$.

Закончить аутентификацию у участника A от имени B нарушитель C не может, так как он не знает секретного ключа участника B. Поэтому для участника A протокол останется незавершенным.

Протоколы идентификации, использующие цифровую подпись. Пусть r_A, t_A — соответственно случайное число и метка времени доказывающего A; S_A — алгоритм цифровой подписи A. Будем считать, что алгоритм проверки цифровой подписи доказывающего известен проверяющему. Пусть cert_A — сертификат открытого ключа для алгоритма проверки цифровой подписи A. Как правило, сертификат имеет вид:

$$\mathrm{cert}_A = (\mathrm{pk}_A, A, S_T(\mathrm{pk}_A, A)),$$

где S_T — цифровая подпись центра сертификации T

Для идентификации могут быть использованы три протокола.

1. Односторонняя идентификация с использованием временны́х меток:

$$A \to B \quad \mathrm{cert}_A, t_A, B, S_A(t_A, B).$$

Получив сообщение, пользователь B проверяет, что временна́я метка находится в допустимом интервале, идентификатор B

совпадает с его собственным идентификатором, а также то, что цифровая подпись под этими двумя полями верна.

2. Односторонняя идентификация с использованием случайных чисел:

1) $A \leftarrow B \quad r_B$;

2) $A \rightarrow B \quad \mathrm{cert}_A, r_A, B, S_A(r_A, r_B, B)$.

Получив сообщение, пользователь B проверяет, что идентификатор B соответствует его идентификатору и что цифровая подпись под строкой (r_A, r_B, B) верна.

3. Взаимная идентификация с использованием случайных чисел:

1) $A \leftarrow B \quad r_B$;

2) $A \rightarrow B \quad \mathrm{cert}_A, r_A, B, S_A(r_A, r_B, B)$;

3) $A \leftarrow B \quad \mathrm{cert}_B, A, S_B(r_B, r_A, A)$.

Рассмотрим примеры таких протоколов. Международной организацией стандартов ISO предложены три протокола, аналогичные рассмотренным выше для случая симметричных систем шифрования.

Протокол односторонней аутентификации ISO1 (public key unilateral authentication protocol). Это одношаговый протокол односторонней аутентификации клиента на основе открытых ключей. Пусть участник A, выступающий в роли клиента, имеет пару открытый ключ — секретный ключ $(\mathrm{pk}_A, \mathrm{sk}_A)$, определяющую пару преобразований E_A и D_A соответственно. Аутентификация проводится с использованием случайного числа r_A:

$$A \rightarrow B \quad \mathrm{cert}_A, r_A, B, M, D_A(r_A, B, M),$$

где M — произвольное сообщение; $\mathrm{cert}_A = (\mathrm{pk}_A, A, D_T(\mathrm{pk}_A, A))$.

В данном случае в качестве цифровой подписи выступает подпись на основе асимметричного шифрования.

Для данного протокола может быть осуществлена соответствующая атака. Нарушитель просто перехватывает и повторяет подписанные сообщения:

0) $A \leftarrow C(B)$ start;

1) $A \rightarrow C(B)$ $\mathrm{cert}_A, r_A, B, M, D_A(r_A, B, M)$;

1′) $C(A) \rightarrow B$ $\mathrm{cert}_A, r_A, B, M, D_A(r_A, B, M)$;

1″) $C(A) \rightarrow B$ $\mathrm{cert}_A, r_A, B, M, D_A(r_A, B, M)$.

В результате нарушитель C может многократно успешно проходить аутентификацию на сервере участника B от имени клиента A.

Заметим, что допускается и другой вариант этого протокола, когда вместо случайного числа r_A используют временну́ю

метку t_A. В этом случае указанная выше атака уже неприменима.

Протокол взаимной аутентификации двухпроходный ISO2 (public key two-pass mutual authentication protocol). Это двухшаговый протокол взаимной аутентификации на основе открытых ключей. Он является, по сути, двукратным повторением протокола ISO1. Пусть, как и выше, участники A и B имеют соответственно пары открытый ключ — секретный ключ $(\mathrm{pk}_A, \mathrm{sk}_A)$ и $(\mathrm{pk}_B, \mathrm{sk}_B)$, определяющие пары преобразований E_A, D_A и E_B, D_B соответственно.

Протокол имеет вид:

1) $A \to B \quad \mathrm{cert}_A, r_A, B, M_2, D_A(r_A, B, M_1)$;

2) $A \leftarrow B \quad \mathrm{cert}_B, r_B, A, M_4, D_B(r_B, A, M_3)$,

где $M_1 - M_4$ — произвольные текстовые сообщения.

В данном случае нарушитель также осуществляет атаку простым прослушиванием и повторением подписанных сообщений:

0) $A \leftarrow C(B)$ start;

1) $A \to C(B)$ $\mathrm{cert}_A, r_A, B, M_2, D_A(r_A, B, M_1)$;

0′) $C(A) \to B$ start;

1′) $C(A) \leftarrow B$ $\mathrm{cert}_B, r_B, A, M_2, D_B(r_B, A, M_1)$;

2) $A \leftarrow C(B)$ $\mathrm{cert}_B, r_B, A, M_2, D_B(r_B, A, M_1)$.

В результате участник A будет уверен, что он связан с участником B, от имени которого на самом деле выступает нарушитель C. Участник B также будет считать, что связан с участником A.

Протокол взаимной аутентификации трехпроходный ISO3 (three-pass mutual authentication protocol). Данный протокол имеет вид:

1) $A \leftarrow B \quad r_B, M_1$;

2) $A \to B \quad \mathrm{cert}_A, r_A, r_B, B, M_3, D_A(r_A, r_B, B, M_2)$;

3) $A \leftarrow B \quad \mathrm{cert}_B, r_B, r_A, A, M_5, D_B(r_B, r_A, A, M_4)$.

5.4. Протоколы идентификации, использующие технику доказательства знания

В парольных схемах противник может запомнить передаваемые сообщения и в следующий раз использовать эту информацию. В протоколах типа «запрос — ответ» противник, контролируя канал связи, может навязывать специально подобранные запросы и, анализируя ответы, получить информацию о секрете.

Чтобы избежать этого, применяют *протоколы доказательства знания* (некоторой секретной информации), которые обладают дополнительным свойством нулевого разглашения секрета (см. далее).

Протоколы идентификации можно рассматривать как вид доказательства знания. Это вид интерактивного доказательства, поэтому остановимся подробнее на понятии интерактивного доказательства.

Интерактивное доказательство (interactive proof) — понятие теории сложности вычислений, составляющее основу понятия доказательства с нулевым разглашением. Интерактивное доказательство проводится путем выполнения протокола с двумя участниками, доказывающим и проверяющим. Участники обмениваются сообщениями (запросами и ответами), обычно зависящими от случайных чисел, которые могут содержаться в секрете. Цель доказывающего — убедить проверяющего в истинности некоторого утверждения. Проверяющий либо принимает, либо отвергает доказательство. В отличие от обычного математического понятия доказательства в данном случае оно носит не абсолютный, а вероятностный характер и характеризуется двумя вероятностями. Если доказываемое утверждение верно, то доказательство должно быть справедливым с вероятностью, стремящейся к единице при увеличении числа циклов протокола. Если же доказываемое утверждение ложно, то при увеличении числа циклов протокола вероятность правильности доказательства должна стремиться к нулю.

Заметим, что часто интерактивным доказательством называют последовательность сообщений, представляющих собой запись (стенограмму) работы протокола.

Доказательство знания (*протокол доказательства знания*) (proof of knowledge) — интерактивное доказательство, в котором доказывающий убеждает проверяющего в том, что он владеет секретной информацией. Протокол интерактивного доказательства должен учитывать возможность обмана со стороны обоих участников. Если участник A (доказывающий) на самом деле не знает доказываемого утверждения (либо от имени участника A выступает кто-либо другой), то участник B должен обнаружить факт обмана. Поэтому доказательство знания характеризуется двумя свойствами: полнотой и корректностью.

Полнота (completeness property) — свойство криптографического протокола, означающее, что при выполнении честными участниками протокол решает ту задачу, для которой он создан.

Корректность (soundness property) — способность криптографического протокола противостоять угрозам со стороны противника и (или) нарушителя, не располагающего необходимой секретной информацией, но пытающегося выполнить протокол за участника A, который по определению должен такой информацией владеть.

С другой стороны, протокол должен обеспечить возможность доказательства владения секретной информацией, не раскрывая ее. Если проверяющий B захочет получить какую-либо информацию об этом утверждении, помимо самого факта владения ею, то его попытки сделать это должны быть обречены на неуспех. Это обеспечивается еще одним важным криптографическим качеством протокола интерактивного доказательства — свойством нулевого разглашения.

Нулевое разглашение (zero-knowledge property) — свойство протокола доказательства знания, обеспечивающее такое его выполнение, что никакая информация о доказываемом утверждении, кроме факта его истинности, не может быть получена нечестным проверяющим из переданных сообщений за время, полиномиально зависящее от суммарной длины этих сообщений.

Протокол с нулевым разглашением (zero-knowledge protocol) — это синоним понятия «доказательство знания с нулевым разглашением».

Как правило, протоколы доказательства выполняют в виде последовательности независимых циклов (раундов), каждый из которых состоит из трех шагов определенного вида:

1) $A \rightarrow B \quad \gamma$ (заявка — witness);
2) $A \leftarrow B \quad x$ (запрос — challenge);
3) $A \rightarrow B \quad y$ (ответ — response).

После выполнения каждого такого цикла проверяющий принимает решение об истинности доказательства.

Идеи, лежащие в основе протоколов с нулевым разглашением, могут быть сформулированы следующим образом: доказывающий A, владеющий секретом s, выбирает случайный элемент r из заранее оговоренного множества как свой секретный параметр для данной итерации протокола, вычисляет, используя r как аргумент некоторой однонаправленной функции h, значение $\gamma = h(r, s)$ и предъявляет это значение проверяющему (в терминологии схемы привязки к биту (см. далее) число r называют *обязательством*, γ — *свидетельством*). Этим обеспечивается случайность и независимость различных итераций протокола. Проверяющий B определяет набор вопросов, на каждый из ко-

торых доказывающий готов дать ответ. Протокол построен так, что только доказывающий A, владеющий секретом s, в состоянии ответить на все эти вопросы и ни один ответ не дает информации о секрете. После этого проверяющий B выбирает один из этих вопросов и доказывающий A дает на него ответ, который затем проверяется участником B. Доказательство осуществляется проведением необходимого числа итераций протокола, т. е. повторений приведенного выше трехшагового протокола с разными значениями (γ, x, y), с целью снизить до приемлемого уровня вероятность обмана.

Другими словами, в основе протоколов с нулевым разглашением лежит комбинация идей протоколов типа «режь и выбирай» (этот термин происходит от стандартного метода, которым дети делят кусок пирога: один режет, а другой выбирает) и протоколов типа «запрос — ответ».

Протоколы идентификации можно рассматривать как вид протоколов доказательства знания. При этом свойства корректности, полноты и нулевого разглашения являются важными криптографическими характеристиками, которые в данном случае могут быть теоретически обоснованы.

Заметим, что протокол идентификации может быть легко получен из алгоритма формирования цифровой подписи, в котором вместо значения $x = h(M)$ нужно использовать запрос от проверяющего.

Рассмотрим примеры таких протоколов, построенных на основе схем цифровых подписей Фиата — Шамира и Шнорра.

Протокол идентификации Фиата — Шамира. Рассмотрим схему цифровой подписи Фиата — Шамира и преобразуем ее в протокол идентификации. Пусть p, q — различные простые числа; $n = pq$. В качестве секретного ключа доказывающий участник выбирает случайное число $0 \neq a \in \mathbf{Z}_n$, $(a, n) = 1$, открытым ключом объявляет значение $b = (a^{-1})^2 \bmod n$.

Протокол имеет вид:

$$\begin{aligned} A \to B \quad & \gamma = r^2 \bmod n, \\ A \leftarrow B \quad & x, \\ A \to B \quad & y = ra^x \bmod n. \end{aligned}$$

Доказывающий A выбирает случайное число r $(1 \leq r \leq n - 1)$ и отправляет участнику B число $\gamma = r^2 \bmod n$. Проверяющий B отвечает случайным запросом x, $x \in \{0, 1\}$. В ответ участник A высылает проверяющему B значение $y = ra^x \bmod n$. Теперь участник B проверяет равенство $y^2 b^x \bmod n = \gamma$.

Далее эти три шага повторяются независимо t раз, причем проверяющий B принимает доказательство владения участником A секретом a, если все эти итерации приводят к положительному ответу.

Полнота. Доказывающий A знает значение a, поэтому он в состоянии ответить на оба вопроса. При этом проверяющий B убеждается в справедливости соотношения

$$y^2 b^x \equiv (ra^x)^2 (a^{-1})^{2x} \bmod n \equiv \gamma \bmod n.$$

Корректность. Наличие запроса x требует, чтобы доказывающий A был в состоянии ответить на любой из двух вопросов, ответ на один из которых требует знания секрета a, а ответ на другой предотвращает попытку обмана. Противник, выдающий себя за A, может попытаться обмануть проверяющего, выбрав любое число z и передав проверяющиему B число $\gamma = z^2 \bmod n$. Тогда он сможет ответить на запрос «$x = 0$», направив правильный ответ «$y = z$», но не сможет ответить на запрос «$x = 1$», ответ на который требует знания корня квадратного из числа b по модулю n. С другой стороны, если он отправит проверяющему B число $\gamma = z^2 b^{-1} \bmod n$, то он сможет ответить на запрос «$x = 1$», но не сможет ответить на запрос «$x = 0$». Таким образом, обман удается с вероятностью, не превышающей $1/2(1 + 1/n)$. Следовательно, при t-кратной итерации протокола вероятность обмана можно считать равной 2^{-t}

Нулевое разглашение. Ответ «$y = r$» не зависит от секрета a доказывающего A, а ответ «$y = ra \bmod n$» также не несет информации о секрете a, так как случайное число r не известно проверяющему B. Обычно для доказательства свойства нулевого разглашения применяют следующий формальный прием. Покажем, что проверяющий может самостоятельно получить правильную тройку сообщений протокола (γ, x, y). Действительно, проверяющий B может сгенерировать случайные числа y, r $(1 \leq \leq y, r \leq n-1)$ и вычислить $\gamma = y^2 b^x \bmod n$. Тогда тройка (γ, x, y) является корректной тройкой сообщений, которая действительно может появиться при выполнении протокола, причем она получена без участника A. Поскольку таким способом может быть получена любая тройка сообщений протокола, множество допустимых троек протокола совпадает с множеством троек, которые проверяющий B может вычислить самостоятельно. Поэтому при наличии доказательства, состоящего из таких троек, которые проверяющий B может вычислить и сам, он не может получить из них информацию о секрете участника A.

Протокол идентификации Шнорра. Данный протокол получается простой модификацией алгоритма формирования цифровой подписи Шнорра, в которой вместо подписываемого сообщения M используют запрос x.

Пусть p — простое число; q — простой делитель числа $p-1$; α — элемент поля $\mathbf{Z}_p$ порядка q. Как и в алгоритме формирования цифровой подписи, в качестве секретного ключа выбирают случайное число a в интервале $1 \le a \le q-2$, а в качестве открытого — значение $\beta = \alpha^{-a} \bmod p$. Пусть cert_A — сертификат открытого ключа участника A.

Сообщения протокола имеют вид:

1) $A \to B \quad \mathrm{cert}_A, \gamma = \alpha^r \bmod p$;

2) $A \leftarrow B \quad x$;

3) $A \to B \quad y = (r + ax) \bmod q$.

Доказывающий A выбирает случайное целое число r ($1 \le r \le q-2$) и отправляет проверяющему B значение $\gamma = \alpha^r \bmod q$. Проверяющий B отвечает случайным запросом x ($1 \le x \le q-1$). Доказывающий A отвечает сообщением $y = (ax + r) \bmod q$. Теперь, если выполнено равенство

$$\gamma = \alpha^y \beta^x \bmod p, \tag{5.1}$$

то проверяющий B принимает доказательство; если равенство не выполнено, то — отвергает.

Заметим, что на втором шаге протокола в качестве запроса x на практике можно использовать случайное число, имеющее в двоичной записи некоторое число t бит (в первоначальном варианте протокола рекомендовалось выбирать случайное число с длиной записи $t = 40$ бит). Однако при теоретическом обосновании с помощью техники доказательства знания лучше использовать битовые запросы $x \in \{0, 1\}$ и повторять весь трехшаговый цикл заново некоторое число t раз.

Полнота. В данном случае полнота протокола заключается в принятии доказательства от истинного участника и легко вытекает из равенств

$$\alpha^y \beta^x = \alpha^{r+ax} (\alpha^{-a})^x \equiv \alpha^r \equiv \gamma \bmod p.$$

Корректность. В рассматриваемом случае корректность протокола означает большую вероятность непринятия доказательства, выдаваемого от имени честного участника кем-либо другим.

Если $x \in \{0, 1\}$, то он может обмануть проверяющего B, послав ему свое число γ, вычисленное при известном ему значе-

нии z. Если он выберет $\gamma = \alpha^z \bmod p$, то сможет правильно ответить $y = z$ на запрос «$x = 0$». Если же он выберет $\gamma = \alpha^z\beta \bmod p$, то сможет правильно ответить на запрос «$x = 1$». При ответе на другой запрос ему не останется ничего другого, как угадать значение y с вероятностью $1/q$, так как он не умеет находить логарифм $\log_\alpha \beta$. Поэтому вероятность успешной аутентификации не превосходит $\frac{1}{2}(1 + \frac{1}{q})$.

При $x \in \{0, 2^t\}$ корректность вытекает из теоремы 7.

Теорема 7. Если для некоторого числа γ и запроса x можно за полиномиальное время с вероятностью успеха не менее $1/2^{t-1}$ найти ответ y такой, что выполняется равенство (5.1), то за полиномиальное время можно найти a.

Доказательство. Среди всех 2^t возможных значений x найдутся не менее $2^t \quad 1/2^{t-1} \geq 2$ значений, дающих верное равенство (5.1). Поэтому существуют $x_1 \neq x_2$, $y_1 \neq y_2$ такие, что

$$\gamma = \alpha^{y_1}\beta^{x_1} \bmod p = \alpha^{y_2}\beta^{x_2} \bmod p,$$

отсюда

$$\alpha^{y_1 - y_2} \equiv \beta^{x_2 - x_1} \bmod p.$$

Поэтому значение

$$a = -\log_\alpha \beta = (x_2 - x_1)^{-1}(y_2 - y_1) \bmod q$$

можно вычислить, например, с помощью расширенного алгоритма Евклида.

Таким образом, вопрос о корректности протокола Шнорра свелся к вопросу о дискретном логарифмировании в поле $\mathbf{Z}_p$: если кто-либо сумеет пройти аутентификацию с вероятностью не менее $1/2^{t-1}$, то он сможет найти $\log_\alpha \beta$.

Замечание. Генерация числа r в протоколе должна выполняться очень качественным генератором. Если при аутентификации участник A использует плохой генератор случайных чисел, то при повторных прохождениях аутентификации может появиться пара троек (γ, x_1, y_1) и (γ, x_2, y_2) с одинаковыми значениями γ. Поэтому противник, наблюдающий за каналом связи, сможет воспользоваться такой ситуацией и найти секретный ключ a.

Нулевое разглашение. Покажем, что проверяющий B может самостоятельно получить любую тройку сообщений протокола (γ, x, y). Действительно, проверяющий B может сгенерировать

случайные числа x $(1 \le x \le q-1)$, y $(1 \le y \le q-1)$ и вычислить значение $\gamma = \alpha^y \beta^x \bmod p$. Тогда тройка (γ, x, y) является корректной тройкой сообщений протокола, причем она получена без участника A. Поскольку таким способом может быть получена любая тройка сообщений протокола, имея подобные тройки сообщений, проверяющий B не может получить из них информацию о секрете участника A.

Протокол Окамото. Рассмотрим модификацию протокола Шнорра, предложенную Т. Окамото (T. Okamoto) в 1985 г. Здесь за счет внесения избыточности в ключевое множество достигается свойство устойчивости к компрометации ключей участников.

Пусть, как и предыдущем случае, p — простое число; q — простой делитель числа $p-1$; α_1, α_2 — элементы поля $\mathbf{Z}_p$ порядка q, для которых вычисление логарифма $\log_{\alpha_1} \alpha_2$ является трудной задачей. В качестве секретного ключа выбирают пару случайных чисел (a_1, a_2), $0 \le a_1, a_2 \le q-2$, а в качестве открытого — значение $\beta = \alpha_1^{-a_1} \alpha_2^{-a_2} \bmod p$.

Сообщения протокола имеют вид:

1) $A \to B$ $\quad \mathrm{cert}_A, \gamma = \alpha_1^{r_1} \alpha_2^{r_2} \bmod p$;

2) $A \leftarrow B$ $\quad x$;

3) $A \to B$ $\quad \begin{cases} y_1 = (r_1 + a_1 x) \bmod q, \\ y_2 = (r_2 + a_2 x) \bmod q. \end{cases}$

Доказывающий A выбирает пару случайных чисел (r_1, r_2) в интервале $1 \le r_1, r_2 \le q-2$, вычисляет значение $\gamma = \alpha_1^{r_1} \alpha_2^{r_2} \bmod q$ и отправляет его участнику B. Проверяющий B отвечает случайным запросом x $(0 \le x \le 2^t < q-1)$. Участник A отвечает сообщением, состоящим из двух чисел:

$$y_1 = r_1 + a_1 x \bmod q, \quad y_2 = r_2 + a_2 x \bmod q.$$

Для проверки правильности участник B использует равенство

$$\gamma = \alpha_1^{y_1} \alpha_2^{y_2} \beta^x \bmod p. \tag{5.2}$$

Полнота. В данном случае полнота протокола вытекает из равенств

$$\alpha_1^{y_1} \alpha_2^{y_2} \beta^x = \alpha_1^{r_1 + a_1 x} \alpha_2^{r_2 + a_2 x} (\alpha_1^{-a_1} \alpha_2^{-a_2})^x \equiv$$

$$\equiv \alpha_1^{r_1} \alpha_2^{r_2} \equiv \gamma \bmod p.$$

Корректность. Перед тем как проверить корректность протокола, опишем классы эквивалентных ключей.

Лемма 3. Пусть $A(\beta) \subseteq \mathbf{Z}_q^2$ — множество секретных ключей $(a_1', a_2') \in \mathbf{Z}_q^2$, дающих такой же открытый ключ β, что и ключ (a_1, a_2). Обозначим $c = \log_{\alpha_1} \alpha_2$, тогда

$$A(\beta) = \{(a_1 + c\theta, a_2 - \theta) \quad \theta \in \mathbf{Z}_q\}.$$

Доказательство. В силу равенства

$$\alpha_1^{-a_1 - c\theta} \alpha_2^{-a_2 + \theta} \equiv \alpha_1^{-a_1} \alpha_2^{-a_2} \equiv \beta \bmod p,$$

множество $\{(a_1 + c\theta, a_2' - \theta) \quad \theta \in \mathbf{Z}_q\}$ содержится в множестве $A(\beta)$ и состоит из q элементов. Тогда с учетом равенства

$$\mathbf{Z}_q^2 = \bigcup_{\beta=0}^{q-1} A(\beta)$$

из количественных соображений получаем, что оно должно совпадать с $A(\beta)$.

Докажем более сильное утверждение.

Лемма 4. Секретные ключи, содержащиеся в множестве $A(\beta)$, неразличимы по информации, наблюдаемой в канале связи.

Доказательство. При каждом ключе (a_1, a_2) в канале связи противник может наблюдать величины γ, x, y_1, y_2. Кроме того, ему известно значение β, но неизвестны значения r_1, r_2. Покажем, что каждая из пар $(a_1', a_2') \in A(\beta)$ при всех возможных r_1, r_2 и фиксированном запросе x порождает то же множество троек (γ, y_1, y_2), что и пара (a_1, a_2).

Пусть $a_1' = a_1 + c\theta, a_2' = a_2 - \theta$ и тройка (γ, y_1, y_2) получена при ключе (a_1, a_2) и случайных числах r_1, r_2. Тогда эта же тройка может быть получена при ключе (a_1', a_2') и случайных числах

$$r_1' = r_1 - cx\theta \bmod q,$$
$$r_2' = r_2 + x\theta \bmod q.$$

Действительно:

$$\alpha_1^{r_1'} \alpha_2^{r_2'} = \alpha_1^{r_1 - cx\theta} \alpha_2^{r_2 + x\theta} \equiv \gamma \bmod p,$$
$$r_1' + a_1' x = (r_1 - cx\theta) + (a_1 + c\theta)x \equiv y_1 \bmod q,$$
$$r_2' + a_2' x = (r_2 + x\theta) + (a_2 - \theta)x \equiv y_2 \bmod q.$$

Теорема 8. Если для некоторого числа γ и запроса x можно за полиномиальное время с вероятностью успеха не менее $1/2^{t-1}$ найти ответ (y_1, y_2) такой, что выполняется равенство (5.2), то за полиномиальное время можно найти ключ (a_1', a_2'), эквивалентный ключу (a_1, a_2).

Доказательство. Среди всех 2^t возможных значений x найдутся не менее $2^t \cdot 1/2^{t-1} \geq 2$ значений, дающих верное равенство (5.2). Поэтому существуют $x \neq x'$, $(y_1, y_2) \neq (y_1', y_2')$ такие, что

$$\gamma = \alpha_1^{y_1}\alpha_2^{y_2}\beta^x \bmod p = \alpha_1^{y_1'}\alpha_2^{y_2'}\beta^{x'} \bmod p,$$

тогда

$$\alpha_1^{y_1-y_1'}\alpha_2^{y_2-y_2'} \equiv \beta^{x'-x} \bmod p.$$

Таким образом, можно найти $(x'-x)^{-1} \bmod q$ с помощью расширенного алгоритма Евклида, а затем принять

$$a_1' = (y_1 - y_1')(x' - x)^{-1} \bmod q,$$
$$a_2' = (y_2 - y_2')(x' - x)^{-1} \bmod q.$$

Покажем теперь, что данный протокол является устойчивым к компрометации секретного ключа одного из абонентов.

Теорема 9. Если для некоторого числа γ и запроса x можно за полиномиальное время с вероятностью успеха не менее $1/2^{t-1}$ найти ответ (y_1, y_2) такой, что выполняется равенство (5.2), то, зная секретный ключ (a_1, a_2), можно за полиномиальное время с вероятностью не менее $1 - 1/q$ найти $\log_{\alpha_1}\alpha_2$.

Доказательство. В условиях теоремы противнику известен ключ (a_1, a_2), и, кроме того, он может найти за полиномиальное время ключ (a_1', a_2'), эквивалентный ключу (a_1, a_2).

Если (a_1', a_2') и (a_1, a_2) не совпадают, то в силу равенства

$$\alpha_1^{-a_1'}\alpha_2^{-a_2'} \equiv \alpha_1^{-a_1}\alpha_2^{-a_2} \equiv \beta \bmod p,$$

имеем

$$\alpha_1^{a_1-a_1'} \equiv \alpha_2^{a_2-a_2'} \bmod p,$$

а значит

$$\log_{\alpha_1}\alpha_2 = (a_1 - a_1')(a_2 - a_2')^{-1} \bmod q.$$

Таким образом, вопрос о стойкости к компрометации ключей протокола Окамото свелся к вопросу о дискретном логарифмировании в поле $\mathbf{Z}_p$: если кто-либо сумеет пройти аутентификацию с вероятностью не менее $1/2^{t-1}$, то в случае компрометации ключей он сможет найти $\log_\alpha \beta$.

Нулевое разглашение. Аналогично предыдущему протоколу покажем, что проверяющий B может самостоятельно получить любой набор сообщений протокола вида (γ, x, y_1, y_2). Действительно, проверяющий B может сгенерировать случайные числа x, y_1, y_2 $(1 \le x, y_1, y_2 \le q-1)$ и вычислить значение $\gamma = \alpha_1^{y_1}\alpha_2^{y_2}\beta^x \bmod p$. Тогда набор сообщений (γ, x, y_1, y_2) является корректным для протокола, причем он получен без участника A. Поскольку таким способом может быть получен любой набор сообщений протокола такого вида, имея подобные наборы сообщений проверяющий B не может получить из них информацию о секрете участника A.

Протокол GQ. В 1988 г. Л. Гуиллоу и Ж. Куискатр (L. Guillou, J.-J. Quisquater) предложили протокол идентификации на основе схемы RSA. Он заключается в следующем: выбирают большие простые числа p, q и рассматривают кольцо $\mathbf{Z}_n$ $(n = pq)$. Далее выбирают элемент $b \ge 3$, удовлетворяющий условию $\text{НОД}(b, \varphi(n)) = 1$, где $\varphi(n)$ — функция Эйлера. Числа n, b являются открытыми параметрами.

Участник A генерирует случайный секретный ключ $u \in \mathbf{Z}_n$ $(\text{НОД}(u, n) = 1)$ и вычисляет открытый ключ

$$v = (u^{-1})^b \bmod n.$$

Сообщения протокола имеют вид:

1) $A \to B$ $\quad \text{cert}_A, \gamma = r^b \bmod n$;

2) $A \leftarrow B$ $\quad x$;

3) $A \to B$ $\quad y = ru^x \bmod n$.

Доказывающий A выбирает случайное число r в интервале $1 \le r \le n-2$, вычисляет значение $\gamma = r^b \bmod n$ и отправляет его проверяющему B. Проверяющий B отвечает случайным запросом x $(0 \le x < b)$. Доказывающий A отвечает сообщением $y = ru^x \bmod n$. Для проверки правильности проверяющий B использует равенство

$$\gamma = v^x y^b \bmod n. \tag{5.3}$$

Полнота. В данном случае полнота протокола вытекает из соотношения $v^x y^b = (u^{-b})^x (ru^x)^b \equiv r^b \equiv \gamma \bmod n$.

Корректность. Показать корректность протокола можно аналогично рассмотренным выше случаям.

Теорема 10. Если для некоторого числа γ и запроса x можно за полиномиальное время с вероятностью успеха p более $2/b$ найти ответ y такой, что выполняется равенство (5.3), то за полиномиальное время можно вычислить ключ u.

Доказательство. Среди всех b возможных значений x найдутся не менее $bp \geq 2$ значений, дающих верное равенство (5.3). Поэтому существуют $x_1 \neq x_2$, $y_1 \neq y_2$ такие, что

$$\gamma = v^{x_1} y_1^b \bmod n = v^{x_2} y_2^b \bmod n,$$

отсюда

$$v^{x_1 - x_2} \equiv \left(\frac{y_2}{y_1}\right)^b \bmod n.$$

Поскольку $0 \leq |x_1 - x_2| < b$, с помощью расширенного алгоритма Евклида можно вычислить обратный элемент $t = (x_1 - x_2)^{-1} \bmod b$ и элемент l такой, что выполняется равенство $(x_1 - x_2)t = lb + 1$. Возведем последнее сравнение в степень t:

$$v^{(x_1 - x_2)t} \equiv v^{lb+1} \equiv \left(\frac{y_2}{y_1}\right)^{bt} \bmod n,$$

тогда

$$v \equiv \left(\frac{y_2}{y_1}\right)^{bt} (v^{-1})^{lb} \bmod n.$$

Возведем обе части сравнения в степень $b^{-1} \bmod \varphi(n)$, и хотя мы и не можем эффективно вычислить эту степень, в результате получим искомое равенство, которое зависит только от тех величин, которые нам уже известны:

$$u = \left(\frac{y_1}{y_2}\right)^t v^l \bmod n.$$

Случай $x \in \{0, 1\}$ можно проанализировать аналогично предыдущим протоколам.

Нулевое разглашение. Покажем, что проверяющий B может самостоятельно получить любую тройку сообщений протокола (γ, x, y). Действительно, проверяющий B может сгенерировать

случайные числа x $(1 \le x \le b-1)$, y $(1 \le y \le n-1)$ и вычислить значение $\gamma = v^x y^b \bmod n$. Тогда тройка (γ, x, y) является корректной тройкой сообщений протокола, причем она получена без участника A. Поскольку таким способом может быть получена любая тройка сообщений протокола, то, имея подобные тройки сообщений, проверяющий B не может получить из них информацию о секрете участника A.

Протокол GQ с ключами, зависящими от идентификаторов. В заключение рассмотрим модифицированный протокол GQ, в котором индивидуальные ключи каждого участника вычисляют как значение некоторой функции от значения идентификатора участника.

Пусть, как и выше, $n = pq$, числа a, b удовлетворяют условию $ab = 1 \bmod n$. Число a известно только центру, выдающему ключи. Выбираем общеизвестную хеш-функцию h и каждому участнику A выдаем секретный ключ $u = \left(h(A)^{-1}\right)^a \in \mathbf{Z}_n$ и открытый ключ $v = h(A)$ (напомним, что для краткости сам участник и его идентификатор обозначаются одним символом). Тем самым выполняется равенство $v = (u^{-1})^b \bmod n$ и мы можем использовать ранее рассмотренный протокол:

1) $A \to B \quad A, \gamma = r^b \bmod n$;
2) $A \leftarrow B \quad x$;
3) $A \to B \quad y = ru^x \bmod n$.

Для проверки правильности участник B использует равенство (5.3). Заметим, что в предыдущем протоколе требовалось выполнение условия $\text{НОД}(u, n) = 1$, которое в данном случае может нарушаться. Тем не менее вероятность такого события очень мала.

Описанный подход, основанный на привязке открытого ключа к идентификатору участника (identity based protocol), решает проблему обеспечения невозможности его подмены и поэтому не требует создания специальных центров сертификации открытых ключей.

В заключение заметим следующее: идентификация может быть гарантирована только в момент времени после завершения протокола. При этом имеется опасность того, что противник подключится к линии связи после окончания процесса идентификации, выдавая себя за законного пользователя. Для исключения этой возможности следует совместить процесс идентификации с процессом установления общего сеансового ключа, который должен быть использован для защиты передаваемой информации до следующей реализации протокола идентификации. По-

добные протоколы будут рассмотрены при описании процедур распределения ключей.

Контрольные задания

1. На какие группы могут быть классифицированы алгоритмы идентификации?

2. В чем состоят недостатки систем с фиксированными паролями?

3. Чем определяется повышенная надежность идентификации при использовании пластиковой карты и личного идентификационного номера?

4. Каковы возможные уязвимости схемы использования одноразовых паролей?

5. В каких целях используют временну́ю метку в протоколе типа «запрос — ответ»?

6. Чем могут быть заменены временны́е метки в протоколах типа «запрос — ответ»?

7. Какая идея лежит в основе протоколов с нулевым разглашением?

8. Какие типы атак могут быть использованы при нападении на протоколы идентификации?

9. Пусть в построенной схеме идентификации для участника A *имеется только один правильный пароль; при остальных паролях* идентификация будет отклонена. Докажите, что перебор паролей в порядке убывания вероятностей их использования дает минимальное время ожидания до появления правильного пароля.

10. Докажите, что если участник A в схеме GQ применяет некачественный генератор случайных чисел для получения числа r, приводящий к повторному появлению использованных ранее чисел, то противник может вычислить секретный ключ u.

Протоколы с нулевым разглашением

6.1. Протоколы решения математических задач

В гл. 5 была показана роль техники протоколов доказательства знания с нулевым разглашением для построения протоколов идентификации. Эта область, безусловно, является одной из главных областей применения протоколов с нулевым разглашением. Однако данная техника имеет намного более широкое применение. Рассмотрим еще несколько типов протоколов с нулевым разглашением.

Заметим, что рассмотренный в подразд. 5.4 протокол идентификации Шнорра, по сути, представляет собой доказательство знания дискретного логарифма. Протокол Фиата — Шамира является протоколом доказательства знания квадратного корня в кольце вычетов по составному модулю. В настоящее время описано множество других протоколов, позволяющих доказывать знание решения различных математических задач, например доказывать знание наличия или отсутствия изоморфизма двух графов (см., например, [1, 83]). Ограничимся рассмотрением задачи проверки принадлежности элемента циклической подгруппе, которая является обобщением протокола Шнорра.

Протокол принадлежности подгруппе (subgroup membership protocol). Пусть элементы циклической подгруппы группы G задаются некоторой однонаправленной функцией $f \quad Z_n \to G$, удовлетворяющей тождеству $f(x+y) = f(x)f(y)$, причем $f(1)$ — образующий элемент циклической подгруппы $\langle f(1)\rangle$ группы G. Требуется для произвольного элемента $\beta \in G$ проверить, принадлежит он подгруппе $\langle f(1)\rangle$ или нет. Эту задачу можно сформулировать так: существует ли решение $a \in Z_n$ такое, что $f(a) = \beta$.

Пусть участник A знает такое решение a. Рассмотрим протокол, с помощью которого участник A может доказать то, что он знает решение a, не раскрывая этого значения:

1) $A \to B \quad \gamma = f(r)$ ($r \in Z_n$ — случайное);

2) $A \leftarrow B \quad x\ (x \in \{0,1\})$;

3) $A \rightarrow B \quad y = \begin{cases} r, & \text{если } x = 0, \\ r + a, & \text{если } x = 1. \end{cases}$

Участник B проверяет условие

$$f(y) = \begin{cases} \gamma, & \text{если } x = 0, \\ \gamma\beta, & \text{если } x = 1. \end{cases}$$

Протокол повторяется t раз, после чего проверяющий B принимает доказательство, если это условие всегда выполнено, и отвергает, если оно нарушено хотя бы один раз.

Полнота. Доказательство свойства полноты протокола очевидно.

Корректность. В данном случае корректность протокола вытекает из того, что противник, выступающий от имени A, может для известного ему значения z вычислить и отправить проверяющему одно из значений $\gamma = f(z)$ или $\gamma = f(z)\beta$. В первом случае он знает ответ на запрос «$x = 0$», во втором — на запрос «$x = 1$». Так как при других запросах ему нужно угадывать значение ответа, то вероятность обмана при одном цикле не превосходит $\frac{1}{2}(1 + \frac{1}{n}) \approx \frac{1}{2}$, а значит, после t итераций вероятность правильного ответа можно полагать равной $\frac{1}{2^t}$.

Нулевое разглашение. Свойство идеального нулевого разглашения вытекает из того, что поскольку значения r имеют равномерное распределение, то и значения y имеют равномерное распределение и поэтому не несут никакой информации о секрете. Кроме того, любую тройку сообщений протокола (γ, x, y) можно легко имитировать со стороны участника B. Действительно, участник B может для произвольной пары (x, y) вычислить значение $\gamma = f(y)\beta^{-x}$

6.2. Протокол привязки к биту

Протокол (схема) привязки к биту (bit commitment protocol (scheme)) является важным примитивным криптографическим протоколом, который входит в качестве составной части во многие прикладные протоколы. Это протокол с двумя участниками (отправителем и получателем), выполняемый в два этапа (привязка бита и раскрытие), посредством которого отправитель передает получателю бит информации (битовое обязательство) таким образом, что выполняются следующие два свойства:

1) *связывание* (binding) — после выполнения этапа привязки бита и передачи свидетельства получателю отправитель уже не может изменить его значение;

2) *сокрытие* (concealing) — получатель не может самостоятельно определить значение бита и узнает его только после выполнения отправителем этапа раскрытия.

Формально протокол можно записать следующим образом: пусть $a \in \{0,1\}$ — обязательство, $f \quad \{0,1\} \times X \to Y$ — некоторая функция, тогда сообщения протокола имеют вид:

1) $A \to B \quad \gamma = f(a,x)$ (привязка);

2) $A \to B \quad x$ (раскрытие).

Участник A выбирает случайный элемент x, вычисляет $\gamma = f(a,x)$ (свидетельство) и отправляет его участнику B. В дальнейшем для осуществления проверки обязательства участник A открывает значение x, и участник B убеждается в том, что элемент a действительно был выбран на первом шаге.

Свойства связывания и сокрытия можно рассматривать как в сложностном аспекте, если накладывать ограничения на вычислительные возможности сторон, так и в совершенном, если исходить из предположения, что вычислительные возможности неограниченны.

Пример 6.1. Функцию f можно построить на основе бесключевой хеш-функции h, полагая $f(a,x) = h(a \parallel x)$, где символом $\parallel$ обозначена конкатенация. В данном случае выполнение первого свойства обеспечивается тем, что хеш-функция h обладает свойством сложности подбора коллизий и второго прообраза, а второго — свойством односторонности. Поэтому свойства связывания и сокрытия выполнены в сложностном смысле.

Пример 6.2. Схема Гольдвассера — Микали (Goldwasser — Micali scheme). Данная схема аналогична той, которая применялась в протоколе идентификации Фиата — Шамира. Пусть p, q — различные простые числа; $n = pq$. Выберем элемент $m \in Z_n$, не являющийся квадратичным вычетом (уравнение $m = z^2 \bmod n$ не имеет решений относительно $z \in Z_n$). Доказывающий A выбирает случайное число x $(1 \le x \le n-1)$ и отправляет проверяющему B число $\gamma = m^a x^2 \bmod n$, затем предъявляет число x:

$$A \to B \quad \gamma = m^a x^2 \bmod n,$$
$$A \to B \quad x.$$

Теперь для проверки обязательства участник B проверяет равенство $m^a x^2 \bmod n = \gamma$.

Проверим свойство связывания. Предположим, что доказывающий A может найти числа x_1, x_2 такие, что $f(1,x_1) = f(0,x_2)$. Тогда

выполнено сравнение $mx_1^2 \equiv x_2^2 \bmod n$, что невозможно, так как тогда $m = (x_2 x_1^{-1})^2 \bmod n$. Поэтому свойство связывания выполнено всегда.

Функция $f(a, x) = m^a x^2 \bmod n$ является (вычислительно) односторонней по переменной x, так как для нахождения x по значению γ нужно извлечь квадратный корень по модулю большого составного числа n с неизвестным разложением на множители. Как известно, эта задача эквивалентна задаче разложения числа n на множители. Поэтому свойство сокрытия по числу x выполняется в сложностном смысле. В то же время можно вычислить символ Якоби $\left(\frac{\gamma}{n}\right)$. Если он принимает значение -1, то понятно, что $a = 1$. Модуль $n = pq$ составной, и значит $\left(\frac{\gamma}{n}\right) = -1$, если $\left(\frac{\gamma}{p}\right) = -1$ и $\left(\frac{\gamma}{q}\right) = 1$, либо $\left(\frac{\gamma}{p}\right) = -1$ и $\left(\frac{\gamma}{q}\right) = 1$. Поэтому в половине случаев можно узнать, что $a = 1$. Если же $\left(\frac{\gamma}{n}\right) = 1$, то вероятность угадывания значения a была бы равна $1/2$. Таким образом, с вероятностью $3/4$ можно узнать значение a.

Поэтому лучше выбрать элемент $m \in \mathbf{Z}_n$, являющийся квадратичным вычетом. Тогда свойства связывания и сокрытия будут выполняться в сложностном смысле.

Пример 6.3. Рассмотрим функцию $f(a, x) = \alpha^x \beta^a$, где α — элемент поля $\mathbf{Z}_p$, имеющий простой порядок q; β — элемент подгруппы $\langle\alpha\rangle$, для которого нахождение значения $\log_\alpha \beta$ является трудной задачей. В этом случае свойство сокрытия выполнено в сложностном смысле, так как задачи нахождения чисел a и x сводятся к поиску логарифма. Свойство связывания также выполняется в сложностном смысле, так как уравнение $f(0, x_1) = f(1, x_2)$ равносильно равенству $\alpha^{x_1 - x_2} = \beta$, поэтому сложность его решения также оценивается сложностью задачи логарифмирования.

6.3. Игровые протоколы

Подбрасывание монеты по телефону (coin-flipping by telephone). Эта задача является частным случаем задачи бросания жребия по телефону, она была впервые достаточно ярко сформулирована в 1982 г. М. Блюмом (M. Blum): «Муж и жена недавно развелись, живут в разных городах и хотят решить, кому достанется машина. Жребий бросает жена. Муж загадывает, что выпадет решка, и сообщает об этом жене по телефону, после чего слышит, как жена (на другом конце провода) говорит: «Ну хорошо... Я бросаю Орел! Ты проиграл!»

Протокол для решения этой задачи можно построить на основе протокола привязки к биту. Будем использовать те же обо-

значения. Пусть у участника A в результате бросания монеты выпало значение a. Участник A сообщает участнику B по телефону свидетельство γ. Теперь участник B угадывает, называя значение b. В заключение участник A предъявляет значение x, и они проверяют совпадение $a = b$. Сообщения протокола имеют вид

$$\begin{aligned} &A \to B \quad \gamma = f(a, x), \\ &A \leftarrow B \quad b, \\ &A \to B : x. \end{aligned}$$

В данном протоколе участник A играет роль жены, участник B — мужа.

Игра «камень — ножницы — бумага» по телефону. Еще один пример игры двух участников по телефону, в которой каждый из участников выбирает один из трех вариантов, дает известная игра «камень — ножницы — бумага». Участники независимо выбирают свои значения, а затем сравнивают их, пользуясь правилом: камень ломает ножницы, ножницы режут бумагу, бумага обертывает камень. В данном случае протокол имеет точно такой же вид, за исключением того, что значения a и b выбирают из множества элементов $\{0, 1, 2\}$, соответствующих выбираемым предметам.

Игра в покер по телефону (mental poker). Еще одним примером применения схем битовых обязательств могут служить протоколы карточных игр по телефону. Например, для игры в покер 52 карты можно представить шестиразрядными двоичными числами и применять протокол подбрасывания монеты по телефону последовательно бит за битом. Каждый из участников может перемешать колоду карт и сдать из нее карту другому участнику. После того как оба участника получат по пять карт, они открывают и сравнивают полученные наборы из пяти карт.

Протокол Гольдвассера — Микали (Goldwasser — Micaii). Рассмотрим протокол [10, 50], использующий в качестве теоретико-числового факта лемму 5.

Лемма 5. Если $n = pq$, где p, q — простые числа, то при всех значениях y сравнение $x^2 \equiv y \bmod n$ имеет четыре решения, причем если $p \equiv q \equiv 3 \bmod 4$, то

$$\begin{cases} a^2 \equiv b^2 (\mathrm{mod} n) \\ (a \pm b) \bmod n \neq 0 \end{cases} \Leftrightarrow \left(\frac{a}{n}\right) = -\left(\frac{b}{n}\right)$$

Доказательство. Используем тот факт, что при изоморфизме $Z_n \cong Z_p + Z_q$ корням уравнения $x^2 \equiv 1 \bmod n$ соответствуют представления $(\pm 1, \pm 1)$, а также то, что при $p \equiv q \equiv 3 \bmod 4$ выполняются следующие равенства для символа Якоби:

$$\left(\frac{-1}{p}\right) = (-1)^{\frac{p-1}{2}} = -1,$$

$$\left(\frac{-1}{q}\right) = (-1)^{\frac{q-1}{2}} = -1,$$

$$\left(\frac{-1}{n}\right) = \left(\frac{-1}{pq}\right) = \left(\frac{-1}{p}\right)\left(\frac{-1}{q}\right) = (-1)(-1) = 1.$$

Заметим, что с помощью указанной в лемме пары (a, b) можно найти факторизацию числа n, если вычислить НОД$(a \pm b, n)$.

Проанализируем протокол Гольдвассера — Микали.

1. *Кодировка карт участником* A. Участник A выбирает 52 пары больших простых чисел (p_i, q_i), удовлетворяющих условию $p_i \equiv q_i \equiv 3 \bmod 4$, и вычисляет произведения $n_i = p_i q_i$ $(i = 1, \ldots, 52)$. Далее участник A тасует колоду карт и присваивает i-й карте в перетасованной колоде число n_i. Затем он ставит в соответствие i-й карте в перетасованной колоде шестерку $(t_1, t_2, \ldots, t_6)$, где каждое t_j — случайное число с $\left(\frac{t_j}{n_i}\right) = 1$, причем t_j — квадратичный вычет по модулю n_i в том и только в том случае, если j-й бит двоичного представления i-й карты равен 1.

Участник A сообщает участнику B все 52 шестерки вместе с числами n_i (но не сообщает p_i, q_i).

2. *Кодировка карт участником* B. Участник B перемешивает свою колоду, порождает и сообщает участнику A аналогичное представление своих карт, используя пары больших простых чисел (r_i, s_i) и их произведения $m_i = r_i s_i$ $(i = 1, \ldots, 52)$.

3. *Участник* A *сдает из колоды карту участнику* B. Чтобы сдать одну карту участнику B применяется протокол подбрасывания 52 случайных чисел x_i, в результате которого участник B знает значения x_i для всех $i = 1, \ldots, 52$, а участник A не знает ни одного из них. Протокол подбрасывания состоит в следующем: участник B выбирает некоторое фиксированное число k, сообщает 52 пары чисел $(x_i^2 \bmod n_i, z_i)$ участнику A, где

$$z_i = \begin{cases} \left(\dfrac{x_i}{n_i}\right), & i \neq k, \\ -\left(\dfrac{x_i}{n_i}\right), & i = k. \end{cases}$$

Участник A может вычислить квадратные корни всех полученных чисел, поскольку знает разложение n_i. Заметим, что у участника A нет возможности узнать выделенное значение k. Поэтому участник A сообщает участнику B конкретное значение квадратного корня x_i, символ Якоби которого сообщен участнику B. Для значений $i \neq k$ участник B не получает никакой дополнительной информации. При $i = k$ участник B вычисляет с использованием леммы 5 второй корень, с помощью которого он теперь может разложить n_i, вычислив наибольший общий делитель разности корней и числа n_i. Теперь участник B может восстановить по полученной кодировке $(t_1, t_2, \ldots, t_6)$ k-й карты колоды участника A ее истинную кодировку шестиразрядным двоичным вектором. Это и будет одна из его карт.

Участник B вынимает из своей колоды сданную ему карту и сообщает участнику A, какая карта была удалена из колоды. Участник A не в состоянии расшифровать эту информацию и, таким образом, не знает, какая карта была удалена.

4. *Участник B сдает из колоды карту участнику A.* Участник B сдает участнику A одну карту из своей колоды, следуя той же самой процедуре. Поскольку в его колоде осталась только 51 карта, он применяет протокол подбрасывания 51 числа к участнику A.

Шаги 3 и 4 протокола продолжают (с учетом уменьшения числа карт в колоде), пока не будет сдано по пять карт обоим участникам (процедура без труда модифицируется для других разновидностей покера). После игры вся секретная информация раскрывается.

Любой полиномиальный алгоритм, угадывающий с вероятностью более $1/2$ конкретный бит i-й карты оппонента, может быть преобразован к вероятностному полиномиальному алгоритму для факторизации $n_i = p_i q_i$.

Нетрудно видеть, что после раскрытия секретной информации может быть обнаружено возможное шулерство. Например, можно обнаружить, что игрок взял на одном шаге более чем одну карту, хотя временно можно заявить более одного символа Якоби.

6.4. Протокол подписания контракта

Рассмотрим еще один тип прикладных протоколов, основанный на протоколе привязки к биту. Напомним, что *протокол подписания контракта* (contract signing protocol) позволяет двум участникам путем обмена сообщениями по каналам связи подписать контракт, существующий только в электронной форме.

Протокол подписания контракта должен обладать рядом свойств.

1. *Честный обмен* (fair exchange) — по окончании протокола оба участника либо должны обладать версиями правильно заключенного контракта, либо не должны обладать таковыми. В частности, если один из участников прервет выполнение протокола, то другой не должен получить действительный контракт.

2. *Эффективность* (effectiveness) означает, что два честных участника после завершения протокола без прерывания его выполнения будут иметь подписанный обеими сторонами контракт.

3. Протокол должен обеспечивать гарантии *завершения по времени* (timeless completion). Это означает, что оба участника должны быть уверены в том, что по истечении установленного промежутка времени протокол будет завершен.

4. *Невозможность отказа* (non-repudiality) для протоколов подписания контракта означает, что подписанный контракт содержит явное доказательство признания контракта обоими участниками.

5. Поскольку все известные протоколы подписания контракта предусматривают участие доверенной третьей стороны, выполняющей функции центра депонирования, необходимо выполнение еще одного свойства: *проверяемость третьей стороны* (third party verifiability) — если доверенная третья сторона будет действовать в пользу одного из участников, компрометируя честность другого, то обман должен быть доказан внешнему участнику.

Рассмотрим в качестве примера протокол подписания контракта ASW, предложенный в 1998 г. Н. Асоканом, В. Шоупом и М. Уайднером (N. Asokan, V. Shoup, M. Waidner [15]). Авторы назвали данный протокол оптимистичным, так как участие третьей стороны в нем требуется только в спорных ситуациях, в то время как при нормальной работе двое участников могут заключить контракт без ее участия.

Предположим, что два участника A и B собираются подписать контракт, текст которого мы обозначим как text. Для решения спорных ситуаций участники привлекают в качестве арбитра доверенную третью сторону T

Протокол подписания контракта включает в свою очередь три протокола:

1) протокол обмена;
2) протокол прерывания;
3) протокол принятия решения.

Протокол обмена (exchange protocol) выполняется за две фазы:

1) фаза привязки
   $A \to B \quad m_A = \mathrm{Sig}_A(\mathrm{pk}_A, \mathrm{pk}_B, T, \mathrm{text}, h(r_A))$,
   $A \leftarrow B \quad m_B = \mathrm{Sig}_B(m_A, h(r_B))$;
2) фаза раскрытия
   $A \to B \quad r_A$,
   $A \leftarrow B \quad r_B$.

Здесь pk — открытый ключ участника, который используется для проверки цифровой подписи.

На фазе привязки стороны сначала вырабатывают случайные числа r_A, r_B, называемые *секретными обязательствами по контракту*, затем вычисляют значения $h(r_A)$ и $h(r_B)$, которые называют *открытыми обязательствами*. После этого участники обмениваются значениями цифровых подписей под ними, играющими роль свидетельств. Далее осуществляется фаза раскрытия. Теперь каждая из сторон может вычислить значения хеш-функций и цифровых подписей и проверить соответствие открытых и секретных обязательств друг друга. В результате каждая из сторон будет иметь подписанный контракт в форме $(\mathrm{text}, m_A, m_B, r_A, r_B)$.

Протокол прерывания (abort protocol) процедуры подписания контракта имеет вид:

1) $A \to T \quad m_A, \mathrm{Sig}_A(\mathrm{aborted}, m_A)$;

2) $A \leftarrow T \quad \begin{cases} \mathrm{Sig}_T(m_A, m_B), & \mathrm{resolved}(m_A) = \mathrm{true}, \\ \mathrm{Sig}_T(\mathrm{aborted}, m_A), & \mathrm{resolved}(m_A) \neq \mathrm{true}. \end{cases}$

Протокол выполняется в случае, если участник A не получил сообщения m_B от участника B в течение установленного промежутка времени. В этом случае он обращается к центру доверия T с запросом о прерывании процедуры подписания контракта. Центр T проверяет по своей базе данных предыдущие записи, по которым он выступал в качестве арбитра. Если центр T ранее

не принимал решение о прерывании данного контракта, то отвечает значением $\mathrm{Sig}_T(m_A, m_B)$ и заносит в свою базу данных запись $\mathrm{resolved}(m_A) = \mathrm{true}$. В противном случае центр отправляет участнику A маркер прерывания $\mathrm{Sig}_T(\mathrm{aborted}, m_A)$ и заносит в свою базу данных запись о том, что данный контракт прерван: $\mathrm{aborted}(m_A) = \mathrm{true}$.

Протокол принятия решения (resolve protocol) имеет вид:

1) $A(B) \to T \quad m_A, m_B$;

2) $A(B) \leftarrow T \quad \begin{cases} \mathrm{Sig}_T(\mathrm{aborted}, m_A), & \mathrm{aborted}(m_A) = \mathrm{true}, \\ \mathrm{Sig}_T(m_A, m_B), & \mathrm{aborted}(m_A) \neq \mathrm{true}. \end{cases}$

Этот протокол необходим в случае, если один из участников A или B после выполнения этапа привязки в течение установленного промежутка времени не получил в ответ сообщение r_B или r_A соответственно для выполнения раскрытия полученного обязательства. В этом случае он также обращается к центру доверия T с просьбой о прерывании процедуры подписания контракта. Если центр T ранее уже принимал решение о прерывании данного контракта ($\mathrm{aborted}(m_A) = \mathrm{true}$), то он отвечает значением маркера прерывания $\mathrm{Sig}_T(\mathrm{aborted}, m_A)$. В противном случае центр отправляет участнику, который направил ему запрос, сообщение $\mathrm{Sig}_T(m_A, m_B)$ и заносит в свою базу данных запись о принятии решения о подписании контракта: $\mathrm{resolved}(m_A) = \mathrm{true}$. Сообщение $\mathrm{Sig}_T(m_A, m_B)$ представляет собой вторую форму котракта и называется замещающим контрактом.

Данный протокол, очевидно, удовлетворяет свойствам 2 — 4. Вместе с тем свойство 1 для него не выполнено.

Приведем атаку на данный протокол [22], в результате которой участник A, нарушая предписанный протоколом порядок действий, может аннулировать контракт, подписанный ранее между ним и участником B:

1) $A \to B \quad m_A$;
2) $A \leftarrow B \quad m_B$;
3) $A \to B \quad r_A$;
4) $A \leftarrow B \quad r_B$;
5) $A \to T \quad m_A, \mathrm{Sig}_A(\mathrm{aborted}, m_A)$;
6) $A \leftarrow T \quad \mathrm{Sig}_T(\mathrm{aborted}, m_A)$;
1′) $A \to B \quad m_A$;
2′) $A \leftarrow B \quad m'_B$.

Предположим, что нарушитель выполнил протокол обмена (пп. 1 — 4) с участником B. Если участник B не инициировал

протокол принятия решения, то нарушитель может опередить его и послать центру T запрос о прерывании (п. 5).

В результате центр внесет в базу данных запись о прерывании данного контракта (п. 6). Теперь нарушитель может открыть второй протокол обмена (пп. 7, 8) для подписания нового контракта с участником B, в котором он выполнит только первую фазу. После того как участник B, не дождавшись ответа, отправит центру запрос о принятии решения, центр ответит ему маркером прерывания, так как такая запись уже появилась в его базе данных после запроса нарушителя:

1) $B \to \mathbf{T} \quad m_A, m'_B$;

2) $B \leftarrow T \quad \mathrm{Sig}_T(\mathrm{aborted}, m_A)$.

В результате у нарушителя будет контракт $(\mathrm{text}, m_A, m_B, r_A, r_B)$, подписанный участником B, который он получил при первом выполнении протокола обмена, а у участника B будет подписанный первый вариант контракта и аннулированный второй контракт $(\mathrm{text}, m_A, m'_B, r_A, r'_B)$. Это произошло потому, что в запросе на прерывание участвует сообщение только первого участника, а в запросе на принятие решения — сообщения от обоих участников.

Данная ситуация, хотя и маловероятна, показывает, что свойство честности обмена для этого протокола не выполнено, так как согласно этому свойству в случае прерывания протокола одним из участников другой не должен иметь подписанную версию контракта. Действительно, это требование честности обмена можно при наличии доверенной третьей стороны уточнить следующим образом:

а) если честный участник получает от центра маркер прерывания, то никто (кроме самого центра) не может получить подписанный контракт;

б) если участник A (не обязательно честный) получил контракт, подписанный честным участником B, то участник B также должен получить подписанный контракт от центра.

Понятно, что для данного протокола первая часть требования не выполнена.

Еще одна слабость данного протокола заключается в том, что в нем не проводится аутентификация сторон. Из-за ее отсутствия противник, прослушивающий сеанс между двумя честными участниками, может в дальнейшем путем повторения сообщений участника A заключать любое число контрактов от его имени с участником B, каждый с новым значением случайного числа r_B.

6.5. Сертифицированная электронная почта

Приведем пример прикладного протокола, основанного на технике привязки, в котором проводится привязка не к биту, а сразу ко всему сообщению.

Сертифицированной электронной почтой называют такой протокол передачи электронных писем, который обеспечивает выполнение двух условий:

1) *доказательство получения* — отправитель A имеет средства для доказательства того, что участник B получил письмо, даже в том случае, если получатель B будет это отрицать;

2) *доказательство отсутствия источника* — получатель B имеет средства для доказательства того, что участник A не отправлял письмо, в случае если участник A будет утверждать, что он его отправил, хотя на самом деле он этого не делал.

В случае бумажной почты можно обеспечить только первое требование: получатель не сможет получить письма, пока он не подпишет подтверждения. Это подтверждение отправляется участнику A для доказательства факта вручения. Поэтому почта выступает в роли доверенной третьей стороны. При этом на самом деле отправитель A получает подтверждение только того, что участник B получил какое-то письмо, не обязательно то, которое было отправлено. Эта проблема легко решается в электронном случае при использовании электронной подписи, которая зависит от содержания письма.

Рассмотрим один из вариантов протокола сертифицированной электронной почты, который несложно реализовать на базе стандартного протокола электронной почты. Предполагается, что участники A и B обладают некоторым симметричным алгоритмом шифрования, могут вычислять хеш-функцию h, а участник B обладает секретным ключом для вычисления цифровой подписи, причем открытый ключ защищен от подмены центром сертификации. Протокол, предложенный в 1998 г. Б. Шнайером и Дж. Риорданом (B. Schneier, J. Riordan) [76], заключается в выполнении следующих действий:

1) участник A генерирует случайный ключ k и отправляет участнику B сообщение $E_k(\text{text})$;

2) участник B отправляет участнику A подписанное цифровой подписью $\text{Sig}_B(M)$ сообщение M вида: «Я хочу, чтобы участник A опубликовал ключ k для зашифрованного сообщения, хеш-свертка которого равна $h(E_k(\text{text}))$, в момент времени t, поместив его в общедоступном ресурсе X»;

3) участник A публикует пару $(h(E_k(\text{text})), k)$ в общедоступном ресурсе X при наступлении или после наступления времени t;

4) участник B расшифровывает полученное сообщение $E_k(\text{text})$.

Покажем, что данный протокол удовлетворяет приведенным выше двум условиям.

1. Если участник B отказывается от получения письма text, то для доказательства факта получения участник A предоставляет арбитру следующие данные: text, k, $E_k(\text{text})$, $\text{Sig}_B(M)$, X. Арбитр убеждается, что $E_k(\text{text})$ соответствует тексту text и хеш-свертке $h(E_k(\text{text}))$ из третьего сообщения, а также что публикация ключа k произошла в соответствии с требованиями, выдвинутыми участником B во втором сообщении. Далее арбитр проверяет правильность подписи $\text{Sig}_B(M)$. Если все соответствия соблюдены, то арбитр подтверждает правоту участника A.

2. Если теперь участник A на самом деле не отправлял письма text, но утверждает обратное, то для доказательства факта неотправления участник B просит участника A предоставить арбитру ту же информацию. Если арбитр найдет в этой информации какое-либо несоответствие, то он подтверждает правоту участника B.

В случае неправильного выполнения протокола участниками он также отвечает поставленной задаче:

- если участник B не ответит вторым сообщением либо даст неправильные дату и адрес, то он не получит ключ и в результате не сможет прочитать сообщение text;
- если участник A не отправит ключ в третьем сообщении протокола в соответствии с указанными в сообщении M временем и адресом, то в общедоступном ресурсе X будет отсутствовать запись $(h(E_k(\text{text})), k)$; это равносильно тому, что участник A не отправил сообщение text;
- если участник B не расшифрует сообщения на четвертом шаге, то это равносильно тому, что он подтвердил получение сообщения text, но отказался его читать.

Вместе с тем этот протокол не обеспечивает свойства конфиденциальности, так как сообщения $E_k(\text{text})$ и k доступны противнику. Поскольку здесь в случае корректного выполнения участниками протокола не требуется участие доверенной третьей стороны, для обмена можно использовать серверы, обеспечивающие анонимность.

Заметим, что протокол можно упростить, допуская отправление ключа k непосредственно участнику B, если потребовать, чтобы участник B возвращал подтверждение получения:

1) $A \to B \quad E_k(\text{text})$;
2) $A \leftarrow B \quad M, \text{Sig}_B(M)$;
3) $A \to B \quad k$;
4) $A \leftarrow B \quad \text{Sig}_B(k)$.

Если теперь участник B не отправит участнику A подтвержения $\text{Sig}_B(k)$, то участник A может опубликовать запись $(h(E_k(\text{text})), k)$ в общедоступном ресурсе X, как это требуется в исходном протоколе.

6.6. Аргумент с нулевым разглашением

Трехшаговое интерактивное доказательство с нулевым разглашением:

1) $A \to B : \gamma$ (заявка — witness),
2) $A \leftarrow B \quad x$ (запрос — challenge),
3) $A \to B \quad y$ (ответ — response) —

можно превратить в неинтерактивный (автономный) одношаговый протокол, называемый *аргументом с нулевым разглашением*, полагая $x = h(\gamma)$:

$$A \to B \quad (\gamma, x, y).$$

Проверка правильности доказательства участником B осуществляется аналогично.

Для данного доказательства уже не выполняется свойство совершенно нулевого разглашения, так как его нельзя имитировать со стороны участника B. Тем не менее в сложностном смысле такое доказательство не позволяет участнику B извлечь информацию о секрете, известном участнику A.

Такой протокол является удобным для различных приложений.

Приведем пример аргумента с нулевым разглашением, который потребуется нам в рассматриваемом далее протоколе электронного голосования. Его особенностью является то, что участник A выбирает свой голос $a \in \{1, -1\}$ (вместо значений 0, 1 взяты 1, -1 для удобства подсчета голосов), который должен оставаться неизвестным, а затем формирует доказательство, подтверждающее его выбор. Сначала рассмотрим интерактивный протокол доказательства:

$$
\begin{aligned}
&A \to B \quad \gamma = (\gamma_0, \gamma_1, \gamma_2),\\
&A \leftarrow B \quad x,\\
&A \to B \quad y = (d_1, d_2, y_1, y_2).
\end{aligned}
$$

Здесь участник A публикует значение свидетельства

$$\gamma_0 = f(a, r) = \alpha^r \beta^a,$$

скрывающего поданный им голос (битовое обязательство) $a \in \{1, -1\}$. Потом выбирает случайные элементы $1 \leq d, z, w \leq q-1$, вычисляет

$$\gamma_1 = \begin{cases} \alpha^z (\gamma_0 \beta)^{-d}, & \text{если } a = 1, \\ \alpha^w, & \text{если } a = -1, \end{cases}$$

$$\gamma_2 = \begin{cases} \alpha^w, & \text{если } a = 1, \\ \alpha^z (\gamma_0 \beta^{-1})^{-d}, & \text{если } a = -1 \end{cases}$$

и публикует их (отправляет участнику B). Затем, получив от участника B случайный запрос x, отвечает четверкой:

$$(d_1, d_2, y_1, y_2) = \begin{cases} (d, d', y, y'), & \text{если } a = 1, \\ (d', d, y', y), & \text{если } a = -1, \end{cases}$$

где $d' = x - d$; $y' = w - rd'$

Участник B проверяет равенства

$$\begin{cases} x = d + d', \\ \alpha^{y_1} = \gamma_1 (\gamma_0 \beta)^{d_1}, \\ \alpha^{y_2} = \gamma_2 (\gamma_0 \beta^{-1})^{d_2}. \end{cases}$$

Проверим полноту протокола.

Если $a = 1$, то

$$\begin{cases} (d_1, d_2, y_1, y_2) = (d, x - d, z, w - r(x - d)), \\ \gamma_1 (\gamma_0 \beta)^{d_1} = \alpha^w (\gamma_0 \beta^{-1})^{-d} (\gamma_0 \beta)^d = \alpha^z, \\ \gamma_2 (\gamma_0 \beta^{-1})^{d_2} = \alpha^z (\alpha^r \beta^1 \beta^{-1})^{-(x-d)} = \alpha^{w - a(x-d)}. \end{cases}$$

Если $a = -1$, то

$$\begin{cases} (d_1, d_2, y_1, y_2) = (x - d, d, w - r(x - d), z), \\ \gamma_1 (\gamma_0 \beta)^{d_1} = \alpha^w (\alpha^r \beta^{-1} \beta)^{x-d} = \alpha^{w - r(x-d)}, \\ \gamma_2 (\gamma_0 \beta^{-1})^{d_2} = \alpha^z (\gamma_0 \beta^{-1})^{-d} (\gamma_0 \beta^{-1})^d = \alpha^z \end{cases}$$

Чтобы получить из этого протокола аргумент с нулевым разглашением, полагаем $x = h(\gamma_0, \gamma_1, \gamma_2)$, где h — хеш-функция. Теперь в автономной версии протокола передается только одно сообщение:

$$A \to B \quad ((\gamma_0, \gamma_1, \gamma_2), h(\gamma_0, \gamma_1, \gamma_2), (d_1, d_2, y_1, y_2)).$$

Корректность такого протокола проверяется непосредственно. Более того, участник A может построить доказательство (ответ на запрос) только в том случае, если $a \in \{1, -1\}$. С другой стороны, участник B не получает никакой информации о значении a кроме того, что $a \in \{1, -1\}$.

6.7. Протокол (схема) электронного голосования

Рассмотрим более сложный тип протокола, представляющий схему для проведения электронного голосования.

Задача ставится следующим образом: участниками протокола являются m лиц с правом голоса и n счетных комиссий, которые создаются для обеспечения анонимности и предотвращения фальсификации итогов голосования. Требуется так организовать голосование, чтобы выполнялись следующие условия:

1) голосуют только уполномоченные избиратели;

2) любой участник имеет право отдать не более одного голоса;

3) ни один из участников не может узнать, как проголосовал другой;

4) никто не может дублировать чужой голос;

5) конечный результат будет подсчитан корректно;

6) любой участник может проверить правильность результата;

7) протокол должен работать и в случае, если некоторые из участников ведут себя нечестно.

Рассмотрим протокол, предложенный в 1996 г. (A. Cramer, M. Franclin, B. Shoenmakers, M. Young). Сначала каждая комиссия получает открытый ключ, а каждый голосующий — цифровую подпись.

Пусть p — простое число, α — элемент поля $\mathbf{Z}_p$, имеющий простой порядок q, $\beta \in \langle\alpha\rangle$ — элемент, для которого нахождение значения $\log_\alpha \beta$ является трудной задачей.

Будем использовать схему битовых обязательств с функцией

$$f(a, x) = \alpha^x \beta^a$$

1. *Заполнение бюллетеня избирателями.* Пусть j-й избиратель выбирает голос $a_j \in \{-1, 1\}$ и случайный элемент $r_j \in Z_q$. Затем он публикует свидетельство $\gamma_{0j} = f(a_j, r_j) = \alpha^{r_j}\beta^{a_j}$, $1 \leq \leq j \leq m$. В результате в общем доступе будут свидетельства всех участников $\gamma_{01}, \ldots, \gamma_{0m}$. Кроме того, j-й избиратель выполняет автономную версию протокола доказательства знания, рассмотренного выше в подразд. 6.3. Далее j-й избиратель подписывает свой голос и доказательство своей цифровой подписью.

2. *Передача бюллетеней в комиссии.* Для передачи бюллетеней с голосами избирателей счетным комиссиям используется пороговая схема разделения секрета Шамира (подробнее см. гл. 9): j-й избиратель выбирает два многочлена над полем Z_q степени $T < n$ (первый — для голосов, второй — для затемняющих факторов):

$$A_j(x) = a_j + a_{1j}x + \quad + a_{Tj}x^T,$$
$$R_j(x) = r_j + r_{1j}x + \quad + r_{Tj}x^T$$

значения которых в точках $1, \ldots, n$ будут являться долями секрета:

$$(u_{ij}, w_{ij}) = (A_j(i), R_j(i)), \quad i = \overline{1, n}.$$

Далее j-й избиратель шифрует доли (u_{ij}, w_{ij}) на открытом ключе i-й комиссии и отправляет эти значения в i-ю комиссию $(i = \overline{1, n})$. Помимо этого j-й избиратель передает i-й счетной комиссии сам многочлен $A_j(x)$, регистрируя с помощью той же схемы битовых обязательств его коэффициенты $f(a_{lj}, r_{lj}) = \alpha^{r_{lj}}\beta^{a_{lj}}$ $(1 \leq l \leq T)$.

3. *Подсчет голосов.* Каждая i-я комиссия подсчитывет бюллетени, публикует результат подсчета бюллетеней

$$u_i = \sum_{j=1}^{m} u_{ij},$$

а также обнародует сумму затемняющих факторов

$$w_i = \sum_{j=1}^{m} w_{ij}.$$

Теперь каждый участник (комиссия и избиратель) может убедиться в корректности опубликованных данных, проверив равенство:

$$\prod_{j=1}^{m}\left(f(a_j,r_j)\prod_{l=1}^{T}f(a_{lj},r_{lj})^{j^l}\right)=\prod_{j=1}^{T}\alpha^{w_{ij}}\beta^{u_{ij}}=\alpha^{w_i}\beta^{u_i}.$$

Действительно

$$\prod_{j=1}^{m}\left(\alpha^{r_j}\beta^{a_j}\prod_{l=1}^{T}(\alpha^{r_{lj}}\beta^{a_{lj}})^{j^l}\right)=\prod_{j=1}^{m}\alpha^{r_j+\sum\limits_{l=1}^{T}r_{ij}l^j}\beta^{a_j+\sum\limits_{l=1}^{T}a_{ij}l^j}=$$

$$=\prod_{j=1}^{m}\alpha^{R_j(i)}\beta^{A_j(i)}=\alpha^{\sum\limits_{j=1}^{m}w_{ij}}\beta^{\sum\limits_{j=1}^{m}u_{ij}}=\alpha^{w_i}\beta^{u_i}.$$

Для определения итога голосования берем T значений u_i, которые также являются значениями некоторого многочлена в точке i, так как

$$u_i=\sum_{j=1}^{m}u_{ij}=\sum_{j=1}^{m}A_j(i)=$$

$$=\left(\sum_{j=1}^{m}a_i\right)+\left(\sum_{j=1}^{m}a_{ij}\right)i+\quad+\left(\sum_{j=1}^{m}a_{Tj}\right)i^T$$

и интерполируем по ним результат $\sum\limits_{j=1}^{m}a_i$.

Если результат — отрицательное число, то голосов, равных -1, больше, если — положительное, то больше голосов, равных $+1$.

Далее проверяют, выполняются ли приведенные выше семь условий.

Контрольные задания

1. Что означает выражение «схема привязки к биту»?
2. Что означают свойства связывания и сокрытия для схем привязки к биту?
3. Покажите, что если при нескольких повторениях протокола проверки принадлежности подгруппе участник A использует случайное число r дважды, то участник B сможет определить число a.
4. Как можно переделать протокол доказательства знания в схему цифровой подписи?

5. Покажите, что протокол ASW удовлетворяет свойствам 2—4 протокола подписания контракта.

6. Предложите способ, позволяющий модифицировать протокол сертифицированной электронной почты так, чтобы для него выполнялось свойство конфиденциальности.

7. Покажите, что протокол электронного голосования обладает требуемыми свойствами.

8. Сколько нечестных комиссий может участвовать в протоколе электронного голосования, не нарушая его основных свойств?

Протоколы передачи ключей

7.1. Передача ключей с использованием симметричного шифрования

Протоколы распределения ключей отличаются как по назначению, так и по способам реализации. Можно выделить три типа протоколов распределения ключей:

1) протоколы передачи (ранее сгенерированных) ключей (обмена ключами);

2) протоколы совместной выработки общего ключа (открытое распределение ключей);

3) схемы предварительного распределения ключей.

Различают также протоколы распределения ключей между отдельными участниками и между группами участников информационного взаимодействия.

В гл. 7 начнем рассмотрение протоколов распределения ключей с протоколов передачи ключей. При этом считаем, что эти ключи уже были заранее сгенерированы кем-либо из участников протокола.

Имеются протоколы, в которых стороны осуществляют передачу ключей или обмен ключами при непосредственном взаимодействии, т. е. двусторонние протоколы или, иначе, протоколы типа «точка — точка» и протоколы с централизованным распределением ключей, в которых предусмотрена третья сторона, выполняющая функции доверенного центра.

Двусторонние протоколы

Рассмотрим сначала двусторонние протоколы передачи ключей с использованием симметричного шифрования. Различают протоколы, в которых стороны заранее располагают какой-либо известной им обеим секретной информацией, и протоколы, не требующие этого условия.

Пусть стороны A и B заранее обладают общей секретной информацией. Допустим, что это — секретный ключ k_{AB}, тогда для

передачи ключа k стороны могут использовать одностороннюю передачу:

$$A \to B \quad E_{k_{AB}}(k, t, B),$$

где E — алгоритм шифрования; t — метка времени; B — идентификатор участника B (напомним, что для краткости идентификаторы обозначены теми же символами, что и сами участники).

Подчеркнем, что если не передавать метки времени, то злоумышленник может осуществить повторную передачу того же сообщения. Если же не указывать идентификатор адресата, то злоумышленник может вернуть отправителю перехваченное сообщение, что в некоторых ситуациях может быть опасным, поскольку абонент A не сможет установить, что это сообщение получено не от абонента B. Заметим также, что временна́я метка и идентификатор могут служить дополнительным подтверждением правильности источника, так как соответствие форматов этих полей после их расшифрования принятым в системе свидетельствует о том, что зашифрование мог осуществить только абонент A.

Заметим, что в приведенном протоколе вместо шифрования можно использовать ключевую хеш-функцию, зависящую от общего ключа:

$$A \to B \quad k \oplus h_{k_{AB}}(t, B).$$

Правда, здесь теряется возможность проверки правильности формата и тем самым подтверждение правильности получения ключа можно будет осуществить только после дополнительного обмена сообщениями.

Если предъявить к протоколу требование проведения аутентификации сеанса в целях более надежной аутентификации источника и подтверждения единственности и своевременности передачи сообщений для защиты от повторной передачи, то можно использовать следующий протокол типа «запрос — ответ»:

1) $A \leftarrow B \quad r_B$;

2) $A \to B \quad E_{k_{AB}}(k, r_B, B)$,

где r_B — случайное число, сгенерированное участником B и переданное участнику A в начале сеанса.

При использовании хеш-функции подобный протокол может выглядеть так:

1) $A \leftarrow B \quad r_B$;

2) $A \to B \quad k \oplus h_{k_{AB}}(r_B, B)$.

Если требуется двусторонняя аутентификация, то можно модифицировать последний протокол, добавив третье сообщение

и предоставив возможность участнику A путем генерации случайного числа r_A и введения его в сообщение на втором шаге протокола, убедиться на третьем шаге в том, что он имеет дело именно с участником B, а также в том, что участник B получил правильное значение ключа k:

1) $A \leftarrow B \quad r_B$;
2) $A \rightarrow B \quad E_{k_{AB}}(k, r_A, r_B, B)$;
3) $A \leftarrow B \quad E_k(r_A)$.

Исходный протокол можно модифицировать так, чтобы искомый ключ k генерировался не одной стороной, а являлся результатом двустороннего обмена. Пусть участники A и B помимо случайных чисел r_A и r_B генерируют случайные числа k_A и k_B соответственно, тогда в результате выполнения протокола:

1) $A \leftarrow B \quad r_B$;
2) $A \rightarrow B \quad E_{k_{AB}}(k_A, r_A, r_B, B)$;
3) $A \leftarrow B \quad E_{k_{AB}}(k_B, r_B, r_A, A)$

каждая из сторон может вычислить общий ключ k с помощью некоторой функции f по правилу $k = f(k_A, k_B)$. Подчеркнем, что в этом протоколе ни одна из сторон не может предсказать заранее значения ключа k.

Протокол Andrew RPC Handshake (протокол рукопожатия для процедуры удаленного вызова Remote Procedure Calls). Рассмотрим более сложный пример двустороннего протокола передачи ключа, предложенный в 1985 г. (M. Satyanarayanan) [30]:

1) $A \rightarrow B \quad A, E_{k_{AB}}(r_A)$;
2) $A \leftarrow B \quad E_{k_{AB}}(r_A + 1, r_B)$;
3) $A \rightarrow B \quad E_{k_{AB}}(r_B + 1)$;
4) $A \leftrightarrow B \quad E_{k_{AB}}(k, r'_B)$.

Здесь первые три сообщения реализуют взаимную аутентификацию сторон, четвертое — передачу ключа, причем эти две части протокола никак не связаны между собой. В работе [37] был отмечен недостаток этого протокола. Поскольку в четвертом сообщении нет никаких средств для проверки новизны ключа, нарушитель может повторно передать последнее сообщение из старого протокола в новый, поэтому участник A будет использовать старое значение ключа из прошлого протокола. Там же был предложен исправленный вариант протокола:

1) $A \rightarrow B \quad A, r_A$;
2) $A \leftarrow B \quad E_{k_{AB}}(r_A + 1, k)$;
3) $A \rightarrow B \quad E_k(r_A + 1)$;
4) $A \leftrightarrow B \quad r'_B$.

Теперь первые два сообщения представляют передачу ключа с использованием техники «запрос — ответ», третье сообщение служит подтверждением для участника B правильности получения значения ключа k, а последнее сообщение содержит случайное число для последующего сеанса.

В 1996 г. предложена (G. Lowe) атака на этот протокол, позволяющая нарушителю C выступать от имени участника B; последний при этом вообще не участвует в обмене. Для этого нарушитель C перехватывает все сообщения, отправляемые от участника A к B, и открывает дополнительную параллельную сессию с участником A от имени B, где полностью повторяет передаваемые участником A сообщения:

1) $A \to C(B)$ A, r_A;
1′) $A \leftarrow C(B)$: A, r_A;
2′) $A \to C(B)$ $E_{k_{AB}}(r_A + 1, k)$;
2) $A \leftarrow C(B)$ $E_{k_{AB}}(r_A + 1, k)$;
3) $A \to C(B)$ $E_k(r_A + 1)$;
3′) $A \leftarrow C(B)$ $E_k(r_A + 1)$;
4′) $A \to C(B)$ r'_B;
4) $A \leftarrow C(B)$ r'_B.

В результате участник A будет думать, что он работает с участником B, в то время как сам участник B вообще не будет ни о чем догадываться.

Использование односторонней функции. В этом протоколе участник A имеет право самостоятельно генерировать новый сеансовый ключ k. Получив сообщение на втором шаге, участник B расшифровывает его и проверяет правильность полученного значения $h(r_B)$. Затем он вычисляет проверочное значение $h(r_A)$ и отправляет его участнику A, зашифровав на новом ключе k. Теперь участник A проверяет правильность значения $h(r_A)$ и убеждается в целостности сеанса и подлинности B:

1) $A \leftarrow B$ B, r_B;
2) $A \to B$ $A, E_{k_{AB}}(h(r_B), r_A, A, k)$;
3) $A \leftarrow B$ $B, E_k(h(r_A))$.

«Бесключевой» протокол Шамира. В заключение рассмотрим протокол, позволяющий передать ключ без использования какой-либо общей секретной информации. Этот протокол иногда называют трехпроходным протоколом Шамира — Ривеста — Адлемана (A. Shamir, R. L. Rivest, L. M. Adleman).

Пусть имеется некоторое коммутирующее шифрующее преобразование E. Это означает, что при всех сообщениях x и произвольных ключах k_1 и k_2 выполняется равенство

$$E_{k_1}(E_{k_2}(x)) = E_{k_2}(E_{k_1}(x)).$$

Тогда пользователи A и B могут реализовать следующий трехпроходный протокол для передачи секретного ключа k от A к B:

1) $A \to B \quad E_{k_A}(k)$;
2) $A \leftarrow B \quad E_{k_B}(E_{k_A}(k))$;
3) $A \to B \quad D_{k_A}(E_{k_B}(E_{k_A}(k))) = E_{k_B}(k)$.

Заметим, что в этом протоколе можно использовать не каждое коммутирующее преобразование E. Например, легко видеть, что для преобразования $E_{k_A}(k) = k \oplus \Gamma$ протокол оказывается заведомо нестойким. Поэтому в протоколе Шамира рекомендуется использовать преобразование вида $E_{k_A}(k) = k^a \bmod p$, в котором константа a определяется ключом k_A; p — большое простое число.

Вместе с тем в данном протоколе отсутствует аутентификация сторон. Поэтому противник может, заблокировав передачу к участнику B, выступить от имени участника B и получить от A ключ для связи с ним от имени B:

1) $A \to C(B) \quad E_{k_A}(k)$;
2) $A \leftarrow C(B) \quad E_{k_C}(E_{k_A}(k))$;
3) $A \to C(B) \quad E_{k_C}(k)$.

Заметим, что протокол не является стойким и к атаке повторением (replay attack). Например, в результате простой атаки:

1) $A \to C(B) \quad E_{k_A}(k)$;
2) $A \leftarrow C(B) \quad E_{k_A}(k)$;
3) $A \to C(B) \quad k$

ключ может появиться в канале связи в явном виде. Для ее проведения нарушитель C осуществляет доступ к сети от имени участника B, повторяя первое сообщение участника A.

Если для защиты протокола от этой атаки осуществлять проверку на втором шаге в целях отбраковки повторно переданных сообщений, то можно осуществить аналогичную атаку путем чередования сообщений двух различных сеансов (interleaving attack):

1) $A \to C(B) \quad E_{k_A}(k)$;
1′) $A \to C(B) \quad E_{k_A}(k')$;
2′) $A \leftarrow C(B) \quad E_{k_A}(k)$;
3′) $A \to C(B) \quad k$;
2) $A \leftarrow C(B) \quad M$;
3) $A \to C(B) \quad D_{k_A}(M)$.

Здесь M — произвольное фиктивное сообщение.

Трехсторонние протоколы

Рассмотрим протоколы передачи ключей между парами участников с использованием доверенной третьей стороны T, называемой центром. В этом качестве обычно выступает некоторый выделенный узел сети или сервер, которому доверяют все участники. Центр T хранит ключи всех абонентов сети, поэтому схема ключевых взаимоотношений графически представляет собой звезду.

Протокол Wide-Mouth Frog. Это простейший протокол передачи ключа k, сгенерированного участником A, от участника A участнику B, предложенный М. Бурроузом (M. Burrows) в 1989 г.:

$$\begin{array}{llll} 1) & A \rightarrow & T & A, E_{k_{AT}}(t_A, B, k); \\ 2) & & T \rightarrow B & E_{k_{BT}}(t_T, A, k). \end{array}$$

Здесь центр T выступает как центр перешифрования ключей; k_{AT}, k_{BT} — ключи, применяемые для связи участника A с центром T и участника B с центром T соответственно; t_A, t_T — метки времени. Участник B, получив второе сообщение, проверяет, чтобы метка времени t_T превосходила все предыдущие метки, указанные центром T Простейшая атака на этот протокол состоит в повторной передаче центру T первого сообщения в подходящем временно́м интервале, что вызовет ответное сообщение с новой меткой времени.

Более сложная атака состоит в записи всего сеанса и последующем использовании центра T для получения нужных ответных сообщений, которые потом можно использовать для введения в заблуждение участников A и B:

$$\begin{array}{llll} 1) & A \rightarrow & T & A, E_{k_{AT}}(t_A, B, k); \\ 2) & & T \rightarrow B & E_{k_{BT}}(t_T, A, k); \\ 1') & & T \leftarrow C(B) & B, E_{k_{BT}}(t_T, A, k); \\ 2') & C(A) \leftarrow & T & E_{k_{AT}}(t'_T, B, k); \\ 1'') & C(A) \rightarrow & T & A, E_{k_{AT}}(t'_T, B, k); \\ 2'') & & T \rightarrow C(B) & E_{k_{BT}}(t''_T, A, k). \end{array}$$

Теперь нарушитель C может отвечать участникам A и B от имени центра T:

$$\begin{array}{llll} 1) & A \leftarrow & C(T) & E_{k_{AT}}(t'_T, B, k); \\ 2) & & C(T) \rightarrow B & E_{k_{BT}}(t''_T, A, k). \end{array}$$

В результате оба участника A и B повторно примут ранее использованный ключ, не подозревая, что они взаимодействуют не друг с другом, а с нарушителем C.

Протокол Yahalom. В данном протоколе, предложенном Р. Яхалом (R. Yahalom), центр T выступает как центр генерации

и распределения ключей. Пусть, как и выше, k_{AT}, k_{BT} — ключи, применяемые для связи участника A с центром T и участника B с центром T соответственно:

1) $A \to B$ $\quad A, r_A;$
2) $B \to T$ $\quad B, E_{k_{BT}}(A, r_A, r_B);$
3) $A \leftarrow \leftarrow \leftarrow T$ $\quad t_a = E_{k_{AT}}(B, k, r_A, r_B);\ t_B = E_{k_{BT}}(A, k);$
4) $A \to B$ $\quad t_B, E_k(r_B).$

Слабость этого протокола заключается в том, что в последнем сообщении протокола нет никаких средств для проверки новизны ключа. Поэтому возможно повторное использование участником A старого ключа, что особенно опасно в случае его компрометации. Этот недостаток был исправлен в работе [37], где предложен исправленный вариант протокола — протокол BAN (M. Burrows, M. Abadi, R. Needham) — Yahalom, отличающийся тем, что в сообщение t_B вставляется случайное число r_B, позволяющее осуществить проверку целостности сеанса.

Протокол BAN — Yahalom. Протокол включает следующие шаги:

1) $A \to B$ $\quad A, r_A;$
2) $B \to T$ $\quad B, r_B, E_{k_{BT}}(A, r_A);$
3) $A \leftarrow \leftarrow \leftarrow T$ $\quad r_B,\ t_A = E_{k_{AT}}(B, k, r_A);\ t_B = E_{k_{BT}}(A, k, r_B);$
4) $A \to B$ $\quad t_B, E_k(r_B).$

Для этого протокола также была найдена [85] следующая атака (interleaving replay attack):

1) $C(A) \to B$ $\quad A, r_A;$
2) $B \to C(T)$ $\quad B, r_B, E_{k_{BT}}(A, r_A);$
1′) $C(A) \to B$ $\quad A, r_A, r_B;$
2′) $B \to C(T)$ $\quad B, r'_B, E_{k_{BT}}(A, r_A, r_B);$
3) $C(A) \leftarrow \leftarrow \leftarrow C(T)$ $\quad$ пропущен;
4) $C(A) \to B$ $\quad E_{k_{BT}}(B, k = r_A, r_B), E_k(r_B).$

Ее суть заключается в следующем: предположим, что можно заменить одно случайное число (nonse) r_A на другое, полученное конкатенацией двух чисел (r_A, r_B), и это останется незамеченным (type flaw). Кроме того, предположим, что нарушитель C может, не дожидаясь завершения первого сеанса протокола, открыть новый сеанс и при этом подменять сообщения из разных сеансов (interleaving attack). В этом случае нарушитель C может открыть два сеанса, повторить сообщения из первого сеанса во втором (replay attack) и с помощью ответных сообщений из второго сеанса завершить первый сеанс, успешно пройти аутентификацию у участника B от имени участника A.

Протокол Woo — Lam взаимной аутентификации и распределения ключей. Рассмотрим протокол, предложенный T. By и C. Лэм (T. Y. C. Woo, S. S. Lam) в 1994 г. [93], который осуществляет распределение ключей и взаимную аутентификацию с использованием симметричного шифрования:

1)	$A \rightarrow$	B	A, r_A;
2)	$A \leftarrow$	B	B, r_B;
3)	$A \rightarrow$	B	$E_{k_{AT}}(A, B, r_A, r_B)$;
4)		$B \rightarrow T$	$E_{k_{AT}}(A, B, r_A, r_B), E_{k_{BT}}(A, B, r_A, r_B)$;
5)		$B \leftarrow T$	$E_{k_{AT}}(B, r_A, r_B, k), E_{k_{BT}}(A, r_A, r_B, k)$;
6)	$A \leftarrow$	B	$E_{k_{AT}}(B, r_A, r_B, k), E_k(r_A, r_B)$;
7)	$A \rightarrow$	B	$E_k(r_B)$.

Приведем атаку на этот протокол, найденную в 1995 г. (J. Clark, J. Jacob). С ее помощью участник B может, применяя parallel session attack, заставить участника A принять в качестве нового использованный ранее ключ:

1)	$A \rightarrow$	B	A, r_1;
1′)	$A \leftarrow$	B	B, r_1;
2′)	$A \rightarrow$	B	A, r_2;
2)	$A \leftarrow$	B	B, r_2;
3)	$A \rightarrow$	B	$E_{k_{AT}}(A, B, r_1, r_2)$;
4)		$B \rightarrow T$	$E_{k_{AT}}(A, B, r_1, r_2), E_{k_{BT}}(A, B, r_1, r_2)$;
5)		$B \leftarrow T$	$E_{k_{AT}}(B, r_1, r_2, k), E_{k_{BT}}(A, r_1, r_2, k)$;
6)	$A \leftarrow$	B	$E_{k_{AT}}(B, r_1, r_2, k), E_k(r_1, r_2)$;
7)	$A \rightarrow$	B	$E_k(r_2)$;
3′)	$A \leftarrow$	B	$E_{k_{AT}}(A, B, r_1, r_2)$;
4′)	$A \rightarrow$	$B(T)$	$E_{k_{AT}}(A, B, r_1, r_2), E_{k_{BT}}(A, B, r_1, r_2)$;
5′)	$A \leftarrow$	$B(T)$	$E_{k_{AT}}(B, r_1, r_2, k), E_{k_{BT}}(A, r_1, r_2, k)$;
6′)	$A \rightarrow$	B	$E_{k_{AT}}(B, r_1, r_2, k), E_k(r_1, r_2)$;
7′)	$A \leftarrow$	B	$E_k(r_2)$.

Нарушитель B открывает второй сеанс с участником A, повторяя все случайные значения из первого сеанса. Затем он завершает первый сеанс, сохраняя все его сообщения. После этого он перехватывает сообщения от участника A к серверу T и возвращает участнику A сообщения первого сеанса в обратном порядке. В результате участник A принимает вторично ключ k.

Протокол NS. Один из первых протоколов такого типа был предложен P. Нидхэмом и M. Шредером (R. M. Needham, M. D. Schroeder) в 1978 г. [69].

Криптографический протокол NS заключается в выполнении следующих шагов:

1) $T \leftarrow A$ A, B, r_A;
2) $T \rightarrow A$ $E_{k_{AT}}(r_A, B, k, E_{k_{BT}}(k, A))$;
3) $A \rightarrow B$ $E_{k_{BT}}(k, A)$;
4) $A \leftarrow B$ $E_k(r_B)$;
5) $A \leftrightarrow B$ $E_k(r_B - 1)$.

В результате выполнения шагов 1 — 3 протокола пользователи A и B получают сгенерированный центром T общий ключ k для организации взаимодействия. Шаги 4, 5 предназначены для аутентификации пользователя A и подтверждения правильности получения ключа участником B.

В 1981 г. Д. Дэннинг и Дж. Сакко (D. Denning, G. Sacco) обнаружили [41] слабость этого протокола, заключающуюся в возможности повторной передачи абоненту B сообщения, отправленного на шаге 3. При таком повторе абонент B не имеет возможности установить, что полученный ключ k уже был использован. Поэтому в случае компрометации этого ключа злоумышленник может аутентифицироваться и передавать сообщения от имени пользователя A.

Протокол Denning — Sacco. Протокол, предложенный Д. Дэннинг и Дж. Сакко в качестве одного из вариантов улучшения протокола NS, основан на использовании метки времени:

1) $T \leftarrow A$ A, B;
2) $T \rightarrow A$ $E_{k_{AT}}(B, k, t, E_{k_{BT}}(A, k, t))$;
3) $A \rightarrow B$ $E_{k_{BT}}(A, k, t)$.

Теперь участник B может сравнить метку времени из третьего сообщения с показанием своих часов и убедиться в том, что это сообщение является новым. Протокол выполняется за три прохода, так как шаги 4, 5 теперь не нужны.

Протокол NS (исправленный). В 1987 г. Р. Нидхэм и М. Шредер предложили новую исправленную версию своего протокола, добавив в начале два дополнительных шага:

1) $A \rightarrow B$ A;
2) $A \leftarrow B$ $E_{k_{BT}}(A, r_B)$;
3) $T \leftarrow A$ $A, B, r_A, E_{k_{BT}}(A, r_B)$;
4) $T \rightarrow A$ $E_{k_{AT}}(r_A, B, k, E_{k_{BT}}(k, r_B, A))$;
5) $A \rightarrow B$ $E_{k_{BT}}(k, r_B, A)$;
6) $A \leftarrow B$ $E_k(n_0)$;
7) $A \rightarrow B$ $E_k(n_0 - 1)$.

Позже в 2005 г. Д. Лонг (D. Long) нашел атаку и на этот вариант протокола, заключающуюся в подмене типов полей в сообщениях протокола (type flaw). Она проходит в случае, если можно сделать идентификатор участника C, выступающего в роли

нарушителя, таким, что его длина будет равна суммарной длине полей (r_A, B, k). Атака осуществляется вначале путем перехвата открытия после третьего шага дополнительного сеанса, в котором нарушитель C выступает и от имени участника B, и от имени сервера T:

1)		$A \to C(B)$	A;
2)		$A \leftarrow C(B)$	X;
3)	$C(T) \leftarrow$	A	A, B, r_A, X;
1′)		$A \leftarrow C(B)$	$C = (r_A, B, k')$;
2′)		$A \to C(B)$	$E_{k_{AT}}(C, r'_A)$;
4)	$C(T) \to$	A	$E_{k_{AT}}((r_A, B, k'), r'_A)$;
5)		$A \to C(B)$	r'_A;
6)		$A \leftarrow C(B)$	$E_{k'}(n_0)$;
7)		$A \to C(B)$	$E_{k'}(n_0 - 1)$.

Здесь X — произвольное сообщение формата $E_{k_{BT}}(A, r_B)$, формируемое нарушителем в качестве отклика на шаге 2 (он не знает ключа k_{BT}). Если участник A сразу отправит сообщение Xсерверу T, не пробуя его расшифровать и не проводя сравнение идентификатора, то далее нарушитель открывает второй сеанс от имени участника B. Полученное от участника A значение $E_{k_{AT}}(C, r'_A)$ интерпретируется нарушителем как $E_{k_{AT}}(r_A, B, k', r'_A)$ и возвращается им обратно участнику A. Участник A расшифровывает его и извлекает значение r'_A, полагая, что это $E_{k_{BT}}(A, r'_B)$. Не имея возможности расшифровать и проверить его, так как он не знает ключа k_{BT}, участник A перенаправляет его участнику B. Теперь нарушитель знает ключ k', поэтому он может нормально завершить сеанс с участником A от имени участника B. В результате участник A уверен, что взаимодействует с участником B, а участник B при этом ничего не подозревает.

Протокол Kerberos. Другой подход к устранению недостатка протокола NS использован в протоколе Kerberos, который в настоящее время получил широкое распространение.

Рассмотрим сначала базовый протокол, применяемый в протоколе аутентификации и распределения ключей Kerberos. Он состоит из следующих шагов:

$A \to T$	A, B, r_A,
$A \leftarrow T$	$E_{k_{AT}}(k, r_A, L, B)$, билет,
$A \to B$	билет, аутентификатор,
$A \leftarrow B$	$E_k(t, k_B)$.

Здесь «билетом» названа величина $E_{k_{BT}}(k, A, L)$, «аутентификатором» — величина $E_k(A, t, k_A)$; t — метка времени; L —

период времени действия билета; r_A — случайное число, сгенерированное участником A и вставленное в передаваемое сообщение для взаимной аутентификации; k_A, k_B — случайные числа, сгенерированные абонентами A и B соответственно и используемые либо в качестве ключа шифрования информации другой стороне, либо для выработки общего ключа $k_{AB} = f(k_A, k_B)$ с помощью некоторой функции f

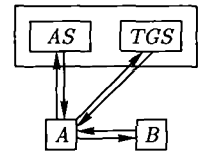

Рис. 7.1. Обмен в протоколе Kerberos

В полном протоколе Kerberos описанный выше базовый протокол используется два раза. Дело в том, что в нем предусмотрено два сервера (рис. 7.1). Первый — сервер аутентификации, обозначенный AS, выдает так называемые «билеты для получения билетов» (tgt), содержащие ключи, предназначенные для длительного использования. Второй — сервер выдачи билетов (TGS) — выдает обычные билеты для доступа к сетевым ресурсам и обращения к другим пользователям.

Сообщения, передаваемые согласно этому протоколу, выглядят следующим образом:

$$
\begin{array}{lcll}
A & \rightarrow & AS & A, TGS, r_A, \\
A & \leftarrow & AS & E_{k_{A,AS}}(k_{A,TGS}, r_A, L_1, TGS), \mathrm{tgt}, \\
A & \rightarrow & TGS & A, B, r'_A, \mathrm{tgt}, \text{ аутентификатор}_1, \\
A & \leftarrow & TGS & E_{k_{A,TGS}}(k, r'_A, L_2, B), \text{ билет}, \\
A & \rightarrow & B & \text{билет, аутентификатор}_2, \\
A & \leftarrow & B & E_k(t_2, k_B),
\end{array}
$$

где

$$
\begin{gathered}
\mathrm{tgt} = E_{k_{AS,TGS}}(k_{A,TGS}, A, L_1), \\
\text{аутентификатор}_1 = E_{k_{A,TGS}}(A, t_1, r'_A), \\
\text{билет} = E_{k_{B,TGS}}(k, A, L_2), \\
\text{аутентификатор}_2 = E_k(A, t_2, k_A).
\end{gathered}
$$

Благодаря введению второго сервера нагрузка на первый сервер уменьшается во много раз. Первый сервер должен быть наиболее защищенным, поскольку он хранит главные ключи всех пользователей. Серверов второго типа может быть несколько, и они могут соответствовать определенной подсети или определенному типу ресурса.

Протокол Otway — Rees. Приведем еще один протокол распределения ключей с использованием сервера, предложенный

Д. Отуэм и О. Ризом (D. Otway, O. Rees) в 1987 г. [72], предпочтительный для случая, когда сервер более удобно расположен для второго абонента. В отличие от предыдущего случая, для которого тип протоколов называется «pull» (тянуть), протоколы такого типа называют «push» (толкать). Протокол состоит в выполнении следующих действий:

1) $A \rightarrow B \quad r, A, B, E_{k_{AT}}(r_A, r, A, B)$;
2) $B \rightarrow T \quad r, A, B, E_{k_{AT}}(r_A, r, A, B), E_{k_{BT}}(r_B, r, A, B)$;
3) $B \leftarrow T \quad E_{k_{AT}}(r_A, k), E_{k_{BT}}(r_B, k)$;
4) $A \leftarrow B \quad E_{k_{AT}}(r_A, k)$.

Участник A генерирует два случайных числа: первое (r_A) используется, как и раньше, для взаимной аутентификации, второе (r) — для аутентификации сеанса связи (вместо него может быть использована метка времени).

Этот протокол можно дополнить еще одним шагом для обеспечения взаимной аутентификации сторон и подтверждения правильности полученного ключа:

4′) $A \rightarrow B \quad E_{k_{AT}}(r_A, k), E_k(r_A, r_B)$;
5) $A \leftarrow B \quad E_k(r_A)$.

При исследовании этого протокола была найдена атака на исходный протокол, основанная на подмене типа полей в передаваемых сообщениях (type flaw). Кратко эту атаку можно описать следующим образом: допустим, что суммарная длина полей r, A, B совпадает с длиной ключа k. Тогда на шаге 1 противник C перехватывает сообщение от A к B.

На шаге 4 противник C посылает пользователю A следующее сообщение от имени B:

4′) $A \leftarrow C(B) \quad E_{k_{AT}}(r_A, (C, A, B))$,

т. е. подменяет поле $E_{k_{AT}}(r_A, k)$ на $E_{k_{AT}}(r_A, (C, A, B))$ и тем самым противник C осуществляет подмену секретного ключа k на известное значение (C, A, B). Поэтому протокол не обеспечивает не только аутентификации сторон, но и аутентификации ключа, так как принятый участником A ключ будет известен противнику.

Семейство протоколов KriptoKnight

Интересный подход к построению протоколов на основе симметричных систем шифрования был предложен в 1992 г. в исследовательском отделе компании IBM в рамках программы NetSP (Network Security Program) [29, 67]. Протокол (точнее — семейство протоколов) KriptoKnight был задуман как гибкий и ком-

пактный набор протоколов для различных сценариев аутентифицированного обмена ключами в сети с учетом широкого спектра практических ситуаций. Этот набор протоколов предоставляет варианты, учитывающие близкое или удаленное расположение доверенного ключевого сервера, различные наборы доступных криптографических механизмов, позволяет уменьшить сложность применяемых преобразований, минимизировать число и длину передаваемых сообщений, чтобы их можно было использовать с минимальным привлечением системных ресурсов на сетевом уровне, а не только на уровне приложений, и в том числе для мобильных сетей связи с низкими скоростями передачи информации.

Заметим, что в семействе протоколов KriptoKnight максимально использовались хеш-функции, задаваемые генерируемым в системе одноразовым ключом, называемым Machine Authentication Code (MAC) [29]. Это позволило сделать систему свободно распространяемой вне зависимости от экспортных ограничений, связанных с криптографией.

В основе системы лежит модульный принцип построения различных протоколов, реализующих разные сценарии и учитывающих те или иные практические ситуации, при котором каждый протокол получали как комбинацию базовых модулей — простейших исходных протоколов, выполняющих конкретные функции. В качестве исходных было предложено применять ряд протоколов, приведенных ниже.

1. Протокол взаимной аутентификации 2PAP (Two Party Authentication Protocol):

1) $A \rightarrow B \quad r_A$;
2) $A \leftarrow B \quad r_B, h_{k_{AB}}(r_A, r_B, B)$;
3) $A \rightarrow B \quad h_{k_{AB}}(r_A, r_B)$.

2. Двусторонний протокол получения ключа 2PKDP (Two Party Key Distribution Protocol):

1) $A \rightarrow T \quad r_A$;
2) $A \leftarrow T \quad r_T, k \oplus h_{k_{AT}}(r_A, r_T, T)$;
3) $A \rightarrow T \quad h_{k_{AT}}(r_A, r_T)$.

В данном протоколе шаг 3 не обязателен. Заметим, что этот протокол имеет два недостатка:

1) участник A не получает подтверждения того, что он получил сообщение именно от T, т. е. аутентификация здесь односторонняя для T;

2) участник A не получает подтверждения правильности получения ключа.

Поэтому вместо него можно использовать двусторонний аутентифицированный протокол получения ключа.

3. Двусторонний аутентифицированный протокол получения ключа 2PAKDP (Two Party Authenticated Key Distribution Protocol):

1) $A \to T \quad r_A$;
2) $A \leftarrow T \quad r_T, h_{k_{AT}}(r_A, r_T, T), k \oplus E_{k_{AT}}(h_{k_{AT}}(r_A, r_T, T))$;
3) $A \to T \quad [h_{k_{AT}}(r_A, r_T)]$.

Здесь квадратные скобки обозначают необязательные сообщения.

Благодаря шифрованию можно открыто передать значение хеш-функции, по которому участник A может сначала проверить, что сообщение получено именно от T, так как только он знает ключ k_{AT}, а затем вычислить ключ k. Таким образом, устранен первый недостаток.

Для устранения второго недостатка можно усилить этот протокол, заменяя во втором сообщении значение r_T на k:

1) $A \to T \quad r_A$;
2) $A \leftarrow T \quad h_{k_{AT}}(r_A, k, T), k \oplus E_{k_{AT}}(h_{k_{AT}}(r_A, k, T))$;
3) $A \to T \quad [h_{k_{AT}}(r_A, k)]$.

Теперь участник A может зашифровать полученное значение хеш-функции и вычислить ключ k, затем с использованием полученного значения хеш-функции проверить не только то, что сообщение получено именно от T, но и правильность получения ключа k.

Используя приведенные выше модули, можно строить более сложные протоколы. Приведем два трехсторонних протокола распределения ключей для сценариев типа «push» и «pull», использующих технику «запрос — ответ».

Push-протокол 3PAKDP (Three Party Authenticated Key Distribution Protocol). Данный протокол для сценария $A - B - T$ предназначен для ситуации, когда ключевой сервер T расположен ближе к участнику B:

1) $A \to B \quad r_A$;
2) $B \to T \quad r_A, r_B, A$;
3) $B \leftarrow T \quad m_A = h_{k_{AT}}(r_A, k, B), k \oplus E_{k_{AT}}(m_A)$;
   $m_B = h_{k_{BT}}(r_B, k, A), k \oplus E_{k_{BT}}(m_B)$,
4) $A \leftarrow B \quad m_A, k \oplus E_{k_{AT}}(m_A),\ r_B, h_k(r_A, r_B, B)$;
5) $A \to B \quad h_k(r_A, r_B), [h_{k_{AT}}(r_A, k)]$;
6) $B \to T \quad [h_{k_{AT}}(r_A, k), h_{k_{BT}}(r_B, k)]$.

Здесь на шаге 4 участник A получает подтверждение правильности вычисленного им значения ключа k, а на шаге 5

участник B получает подтверждение правильности значения ключа k. На последнем шаге 6 центр T получает подтверждение того, что оба участника, а именно в точности A и B, получили правильное значение ключа.

Pull-протокол 3PAKDP. Этот протокол для сценария $T - A - B$ предназначен для ситуации, когда ключевой сервер T расположен ближе к участнику A:

1)		$A \to B$	r_A;
2)		$A \leftrightarrow B$	$r_B, M_B = h_{k_{BT}}(r_A, r_B, B)$;
3)	$T \leftarrow$	A	$r_A, r_B, B,\ h_{k_{AT}}(r_A, M_B, B), A$;
4)	$T \to$	A	$m_A = h_{k_{AT}}(r_A, k, B), k \oplus E_{k_{AT}}(m_A)$;
			$m_B = h_{k_{BT}}(r_B, k, A), k \oplus E_{k_{BT}}(m_B)$;
5)		$A \leftrightarrow B$	$m_B, k \oplus E_{k_{BT}}(m_B)$.

В данном случае шестой шаг протокола не нужен, так как значение M_B позволяет центру T на втором и третьем шагах осуществить одновременно аутентификацию сторон A и B.

Если же помимо этого необходимо организовать для центра T подтверждение правильности получения ключа обоими участниками, то нужно модифицировать сообщение на шаге 5 и добавить шаг 6 аналогично тому, как это сделано в предыдущем протоколе.

Заметим, что в семействе KriptoKnight имеются еще протоколы, использующие временны́е метки. Они удобны для локальных сетей, где нет проблемы синхронизации часов и все участники имеют одинаковый доступ к ключевому серверу. Всего описано пять различных протоколов. Кроме них предложено несколько вариантов протоколов для различных сценариев междоменного обмена, когда участники находятся в разных доменах, имеющих разные ключевые серверы.

7.2. Передача ключей с использованием асимметричного шифрования

Рассмотрим варианты использования асимметричного шифрования для передачи секретных ключей симметричных криптосистем.

Протоколы без использования цифровой подписи

Для передачи ключа k можно использовать следующий одношаговый протокол:

$$A \to B \quad E_B(k, t, A),$$

где E_B — алгоритм шифрования с открытым ключом участника B; t — метка времени.

Протокол NSPK. Рассмотренный в гл. 5 протокол взаимной аутентификации NSPK можно применять и для передачи и совместной выработки ключа, причем с одновременным подтверждением правильности получения ключа. Для этого нужно вместо случайных чисел (nonces) r_A, r_B использовать сами ключевые переменные k_A, k_B. Сообщения такого протокола имеют вид

$$\begin{aligned} A &\to B \quad E_B(k_A, A), \\ A &\leftrightarrow B \quad E_A(k_A, k_B), \\ A &\to B \quad E_B(k_B). \end{aligned}$$

В данном случае каждая из сторон генерирует свое значение ключа, которое можно использовать как ключ для одного направления либо вычислить общий ключ как одностороннюю функцию от значений k_A, k_B. Проводя расшифровку полученных сообщений на втором и третьем шагах, после проверки совпадения значений k_A и k_B стороны убеждаются в том, что они имеют дело именно с нужной стороной и что другая сторона правильно расшифровала полученное значение ключа.

Поскольку в таком виде, как отмечалось выше, протокол имеет слабость, необходимо заменить третий шаг протокола так, чтобы из передаваемых сообщений нельзя было извлечь значение переменной k_B.

Например, в протоколе Oakley Conservative семейства IPSec третий шаг протокола изменен так:

$$\begin{aligned} A &\to B \quad A, B, E_B(k_A), \\ A &\leftarrow B \quad E_A(k_A, k_B), B, A, \\ A &\to B \quad h_k(A, B), \end{aligned}$$

где ключ $k = h_0(k_A, k_B)$ вычисляется как значение хеш-функции h при нулевом ключе от конкатенации значений k_A и k_B.

Таким образом, секретное значение k_B спрятано односторонним образом в ключевую переменную k хэш-функции, что не позволяет нарушителю извлечь его из третьего сообщения.

Протокол Woo—Lam. В 1992 г. Т. Ву и С. Лэм (T. Woo, S. Lam) предложили [94] серию протоколов идентификации и распределения ключей. Рассмотрим один из предложенных ими

протоколов на основе открытых ключей. Участниками протокола являются абоненты A и B, а также центр генерации ключей — доверенный сервер T Каждый из участников имеет пару ключей асимметричного алгоритма шифрования, заверенную сертификатом открытого ключа.

Протокол состоит из следующих шагов:

1) $A \rightarrow B$ $E_B(r_A, A)$;
2) $B \rightarrow T$ $B, A, E_T(r_A)$;
3) $B \leftarrow T$ $E_B(D_T(r_A, k_{AB}, B))$;
4) $A \leftarrow B$ $E_A(D_T(r_A, k_{AB}, B), r_B)$,

где E_i — операция зашифрования данных на открытом ключе участника i ($i = A, B, T$); D_T — операция расшифрования данных на асимметричном секретном ключе доверенного сервера T; $E_{k_{AB}}$ — операция зашифрования данных симметричным криптографическим алгоритмом на ключе k_{AB}; k_{AB} — порождаемый доверенным сервером T результирующий ключ; r_A, r_B — случайные числа, порождаемые соответственно абонентами A и B.

Атака включает следующие шаги:

1) $A \rightarrow B$ $E_B(r_A, A)$;
2) $B \rightarrow C(T)$ $B, A, E_T(r_A)$;
2′) $C \rightarrow T$ $C, U, E_T(r_A)$;
3′) $C \leftarrow T$ $E_C(D_T(r_A, k_{CU}, C))$;
1″) $B \leftarrow C$ $E_B(r_A, C)$;
2″) $T \leftarrow B$ $B, C, E_T(r_A)$;
3″) $T \rightarrow B$ $E_B(D_T(r_A, k_{CB}, B))$;
4″) $B \rightarrow C$ $E_C(D_T(r_A, k_{CB}, B), r'_B)$;
3) $B \leftarrow C(T)$ $E_B(D_T(r_A, k_{CB}, B))$.

Здесь противник C перехватывает второе сообщение от участника B к доверенному серверу T и просит у T ключ для соединения с участником U, где U — идентификатор произвольного абонента сети, с которым противник имеет право устанавливать соединение. Получив на шаге 3′ сообщение $E_C(D_T(r_A, k_{CU}, C))$, противник извлекает из него значение r_A. Далее противник C не предпринимает никаких действий, связанных с абонентом U, а вместо этого устанавливает от своего лица соединение с участником B (третий сеанс). Извлекая из сообщения 4″ значение $D_T(r_A, k_{CB}, B)$, противник формирует сообщение $E_B(D_T(r_A, k_{CB}, B))$, которое он отправляет участнику B от имени доверенного сервера T, имитируя шаг 3 протокола в первом сеансе. Далее участники A и B завершают выполнение протокола. В результате описанных действий участники A и B вырабатывают результирующий ключ k_{CB}, известный про-

тивнику C. Противник провел успешную подмену доверенного сервера на шаге 3.

Авторы [93, 94] предложили исправленный вариант протокола:

1) $A \rightarrow B \qquad E_B(r_A, A);$
2) $\quad\; B \rightarrow T \quad B, A, E_T(r_A);$
3) $\quad\; B \leftarrow T \quad E_B(D_T(r_A, k_{AB}, A, B));$
4) $A \leftarrow B \qquad E_A(D_T(r_A, k_{AB}, A, B), r_B).$

Вставка второго идентификатора теперь делает невозможным смешивание сообщений из разных сеансов. В то же время нетрудно заметить, что у противника по-прежнему сохраняется возможность определения значения r_A, поэтому шифрование на шагах 1 и 2 протокола является излишним.

Смешанные протоколы

Протокол EKE. Рассмотрим протокол EKE (Encrypted Key Exchange), предложенный в 1992 г. С. Белловином и М. Мерриттом (S. Bellovin и M. Merritt) [27]. Для передачи ключа k от участника B участнику A они должны обладать согласованным заранее общим паролем P, играющим роль секретного ключа симметричной системы шифрования ($P = k_{AB}$). На этом ключе сначала передается сгенерированный стороной A открытый ключ E_A, который затем используется стороной B для шифрования сгенерированного им ключа k:

1) $A \rightarrow B \quad A, E_P(E_A);$
2) $A \leftarrow B \quad E_P(E_A(k));$
3) $A \rightarrow B \quad E_k(r_A);$
4) $A \leftarrow B \quad E_k(r_A, r_B);$
5) $A \rightarrow B \quad E_k(r_B).$

Первые два шага реализуют обмен ключами, последние три шага протокола служат для подтверждения правильности получения ключа и взаимной аутентификации.

Рассмотрим пример атаки на этот протокол, найденный средством AVISPA:

0) $A \leftarrow C(B);$
1) $A \rightarrow C(B) \qquad A, E_P(E_A);$
1′) $\quad\; C(B) \rightarrow A' \quad B, E_P(E_A);$
2′) $\quad\; C(B) \leftarrow A' \quad E_P(E_A(k));$
2) $A \leftarrow C(B) \qquad E_P(E_A(k));$
3) $A \rightarrow C(B) \qquad E_k(r_A);$
3′) $\quad\; C(B) \rightarrow A' \quad E_k(r_A);$

$4')$ $\quad C(B) \leftarrow A' \quad E_k(r_A, r_B);$
$4)$ $A \leftarrow \; C(B) \quad E_k(r_A, r_B);$
$5)$ $A \rightarrow \; C(B) \quad E_k(r_B);$
$5')$ $\quad C(B) \rightarrow A' \quad E_k(r_B).$

Для реализации этой атаки противник C использует два параллельных сеанса с участником A, в которых, маскируясь под участника B, он выступает как «противник в середине» между A — инициатором сеанса (первый сеанс) и A — получателем (во втором сеансе участник A обозначен A'). **В** результате противник C успешно завершает аутентификацию от имени участника **B**, правда, при этом ни одно из случайных чисел r_A, r_B и само значение ключа k для него не раскрыто.

Bilateral Key Exchange with Public Key. Это двусторонний протокол передачи ключа на основе открытых ключей с использованием односторонней функции:

1) $A \leftarrow B \quad B, E_A(r_B, B);$
2) $A \rightarrow B \quad E_B(h(r_B), r_A, A, k);$
3) $A \leftarrow B \quad E_k(h(r_A)).$

Здесь также используется два алгоритма шифрования: алгоритм E_A шифрования на открытом ключе и алгоритм E_k симметричного шифрования.

Протокол SPX. Данный протокол представляет исторический интерес как первая попытка создать инфраструктуру открытых ключей. Он предназначен для выработки совместного секретного ключа k участниками A и B. Данный протокол разработан в рамках архитектуры DSSA (Distributed System Security Architecture), предложенной фирмой DEC. Данная архитектура предусматривала использование как секретных ключей (для обеспечения контроля целостности, аутентификации источника данных и конфиденциальности), так и открытых ключей (для аутентификации сторон). Первоначально было предложено использовать алгоритмы DES для шифрования на секретных ключах и RSA — на открытых.

Для аутентификации сторон была разработана служба DASS (Distributed Authentication Security Service), которая явилась развитием разработанной ранее системы аутентификации и распределения ключей Sphinx.

Протокол SPX для аутентифицированного обмена ключами представляет собой трехсторонний протокол, в котором в качестве доверенной третьей стороны T выступает центр распределения сертификатов ключей CDC (Certificate Distribution Center). Участники A и B обладают парами ключей (E_A, D_A) и (E_B, D_B)

соответственно, где, как обычно, E_A обозначает для краткости не только алгоритм зашифрования, но и сам открытый ключ, D_A — секретный ключ. Пусть l_A, l_B — время жизни ключей E_A и E_B соответственно; a_A, a_B — сетевые адреса участников; t — временна́я метка.

Обозначим $m_A = (A, B, l_B, E_B)$, $m_B = (B, A, l_A, E_A)$. Тогда сообщения протокола имеют вид:

1) $A \to T \quad B$;
2) $A \leftarrow T \quad m_A, E_{k_{AT}}(h(m_A))$;
3) $A \to B \quad A, D_A(A, E_A, L), E_B(k), D_A(E_B(k)), t, E_k(t, a_A)$;
4) $B \to T \quad A$;
5) $B \leftarrow T \quad m_B, E_{k_{BT}}(h(m_B))$;
6) $B \to A \quad E_k(t, a_B)$.

Пары сообщений 1 и 2, 4 и 5 фактически реализуют получение участниками A и B из центра сертификации ключей T сертификатов открытых ключей для связи друг с другом. Сертификаты имеют вид

$$\mathrm{cert}_{AB} = (m_A, E_{k_{AT}}(h(m_A))),$$
$$\mathrm{cert}_{BA} = (m_B, E_{k_{BT}}(h(m_B))).$$

Заметим, что на самом деле сертификаты создаются центром сертификации ключей CA (Certification Authority), который представляет собой доверенный с высокой безопасностью центр, а центр CDC выполняет функции службы каталогов (directory services), работающей в реальном времени (on-line).

Для передачи ключа k фактически используют только сообщения 3 и 6.

Сообщение на третьем шаге протокола называют жетоном (token), а поле $D_A(A, E_A, L)$ — билетом. Оно позволяет участнику B убедиться в том, что сообщение получено именно от участника A и время сеанса соответствует времени жизни открытого ключа. Поля $E_k(t, a_A)$ и $E_k(t, a_B)$ называют аутентификаторами.

Участник A генерирует ключ k с временем жизни L и передает его в зашифрованном виде участнику B. Получив данное сообщение, участник B вычисляет ключ k и, используя поле $D_A(E_B(k))$, убеждается, что это значение получено именно от A. Затем проверяет правильность получения этого значения, используя аутентификатор $E_k(t, a_A)$.

Заметим, что если вместо значения $D_A(E_B(k))$ вставить $E_k(D_A)$, то участник A может тем самым делегировать (пе-

редать) свои права участнику B, передав ему свой секретный ключ D_A.

Неудобство таких сертификатов состоит в необходимости использования секретных ключей, общих с центром CDC, а также в том, что сертификаты привязаны к направлению, а не к отправителю.

Протоколы с использованием цифровой подписи

При использовании цифровой подписи аутентифицированный протокол передачи ключей может содержать только одно сообщение и иметь, например, один из следующих трех видов:

1) зашифрование и подпись ключа

$$A \to B \quad E_B(k,t),\ \mathrm{Sig}_A(B,k,t);$$

2) зашифрование подписанного ключа

$$A \to B \quad E_B(k,t,\ \mathrm{Sig}_A(B,k,t));$$

3) подпись зашифрованного ключа

$$A \to B \quad t, E_B(A,k),\ \mathrm{Sig}_A(B,t,E_B(A,k)).$$

Здесь $\mathrm{Sig}_A(M)$ — результат применения алгоритма формирования цифровой подписи к сообщению M при секретном ключе sk_A участника A.

Сертификаты открытых ключей

Многие протоколы и приложения, использующие криптосистемы с открытыми ключами для широкого спектра задач, должны быть способны работать с большим числом открытых ключей, применяемых в распределенных системах. Поэтому им приходится обращаться к услугам инфраструктуры открытых ключей (Public Key Infrastructures — PKI), которые предназначены для управления открытыми ключами для различных приложений. Большинство инфраструктур открытых ключей основаны на центрах сертификации (Certification Authorities — CAs), которые выдают сертификаты. Точнее — они осуществляют проверку и подтверждение подлинности открытых ключей. Как правило, при использовании открытых ключей хранятся и пересылаются не сами ключи, а их сертификаты. Центры серти-

фикации выдают сертификаты, осуществляют проверку их подлинности и ведут списки отозванных сертификатов (Certificate Revocation Lists — CRLs).

Примерами PKI являются PKIX (Public-Key Infrastructure X.509), SPKI (Simple Public Key Infrastructure) и DNSSEC (Domain Name System Security), которые стандартизованы сообществом IETF.

Сертификат представляет собой набор данных:

$$\mathrm{cert}_A = (A, k_A, t, \mathrm{Sig}_T(A, k_A, t)),$$

состоящий из идентификатора абонента A, его открытого ключа k_A и дополнительной информации, включающей время t выдачи сертификата, срок его действия, предназначение ключа, заверенный цифровой подписью доверенного центра T или заслуживающего доверия лица. Сертификат предназначен для исключения возможности подмены открытого ключа при его хранении или пересылке.

Получив такой сертификат и проверив цифровую подпись, можно убедиться в том, что открытый ключ действительно принадлежит данному абоненту.

Международный стандарт МККТТ X.509 (ISO 9594—8) определяет следующий протокол идентификации с одновременным распределением ключей:

1) $A \to B \quad \mathrm{cert}_A, d_A,\ \mathrm{Sig}_A(d_A)$;
2) $A \leftarrow B \quad \mathrm{cert}_B, d_B,\ \mathrm{Sig}_B(d_B)$;
3) $A \to B \quad r_B, B,\ \mathrm{Sig}_A(r_B, B)$,

где cert_A, cert_B — сертификаты сторон; Sig_A, Sig_B — алгоритмы формирования цифровых подписей сторон; d_A, d_B — наборы передаваемых и заверяемых цифровой подписью данных:

$$d_A = (t_A, r_A, B, \mathrm{data}_1, E_B(k_A)),$$

$$d_B = (t_B, r_B, A, r_A, \mathrm{data}_2, E_A(k_B)).$$

В поле data заносится дополнительная информация для аутентификации источника.

Стандарт X.509 предусматривает двухпроходный и трехпроходный варианты протокола. Третий проход протокола требуется для подтверждения стороне B того, что она действительно взаимодействует со стороной A.

Заметим, что в первоначальном варианте 1987 г. протокол в третьем сообщении не содержал идентификатора участника B:

3) $A \to B \quad r_B, B,\ \mathrm{Sig}_A(r_B)$.

Покажем, что в таком виде протокол имеет уязвимость. Воспользуемся примером атаки, приведенным в работе [37]. Предположим, что нарушитель C может начать второй сеанс протокола аутентификации с участником A в требуемый момент, тогда он может сделать следующее:

1) $C(A) \to B$ $\text{cert}_A, d_A = (t_A, r_A, B, x_A, E_B(k_A))$, $\text{Sig}_A(d_A)$;
2) $C(A) \leftarrow B$ $\text{cert}_B, d_B = (t_B, r_B, A, x_B, E_A(k_B))$, $\text{Sig}_B(d_B)$;
1′) $A \to C$ $\text{cert}_A, d'_A = (t'_A, r'_A, B, x'_A, E_B(k'_A))$, $\text{Sig}_A(d'_A)$;
2′) $A \leftarrow C$ $\text{cert}_C, d_C = (t_C, r_B, A, x_C, E_A(k_C))$, $\text{Sig}_C(d_C)$;
3′) $A \to C$ r_B, C, $\text{Sig}_A(r_B)$;
3) $C(A) \to B$ r_B, B, $\text{Sig}_A(r_B)$.

В первом сообщении нарушитель C просто повторяет старое сообщение участника A. При этом участник B не сможет заметить обмана, так как проверка значения t_A не осуществляется. Теперь нарушитель C сможет найти значение r_B и открыть сеанс с участником A. В результате участник A будет знать, что он работал с нарушителем C и сеанс завершился нормально, а участник B будет в полной уверенности, что к нему обращается участник A, поскольку протокол аутентификации успешно завершен.

Хотя большой опасности такая атака не представляет, так как нарушитель не сможет восстановить значение ключа k_B и поэтому не будет знать итогового ключа, тем не менее факт успешного прохождения им аутентификации свидетельствует о недостаточной защищенности такого протокола.

Если теперь вставить в третье сообщение идентификатор адресата, то перенаправить такое сообщение другому участнику будет невозможно. Поэтому нельзя завершить протокол без ошибок.

Контрольные задания

1. Каковы преимущества централизованного распределения ключей?
2. Какие шифры нельзя использовать в «бесключевом» протоколе Шамира?
3. Каков недостаток протокола NS?
4. С какой целью вводится второй сервер в протоколе Kerberos?
5. Как можно использовать цифровую подпись для защиты протоколов передачи ключей?
6. Каковы назначение и структура сертификата открытого ключа?
7. Перечислите основные атаки на протоколы распределения ключей.

Глава 8

Открытое распределение ключей

8.1. Виды протоколов открытого распределения ключей и их свойства

*Открытое распределение ключей** представляет собой протокол обмена сообщениями по открытому каналу связи, позволяющий участникам после завершения обмена выработать общий секретный ключ.

Более точно протоколы открытого распределения ключей следует называть протоколами динамической совместной выработки участниками общего ключа (key agreement) в отличие от статической выработки общего ключа, которая имеет место в схемах предварительного распределения ключей.

Важным преимуществом открытого распределения ключей является то, что в отличие от схем предварительного распределения ключей в данном случае ни один из абонентов заранее не может предугадать значения ключа, поскольку ключ существенно зависит от содержания сообщений, передаваемых в процессе обмена.

Однако главное преимущество открытого распределения ключей заключается в том, что участники вырабатывают ключ без какой-либо общей секретной информации, распределяемой заранее. Именно благодаря этому свойству такие протоколы сразу после появления в 1976 г. привлекли к себе огромное внимание и нашли широкое применение в различных финансовых и коммерческих приложениях.

Выше были рассмотрены примеры протоколов динамической совместной выработки участниками общего ключа с использованием симметричных систем шифрования. Участники A и B самостоятельно генерируют «половинки» ключей k_A, k_B соответственно и передают их друг другу зашифрованными на общем секретном ключе k_{AB}. Затем ключ, необходимый для установления защищенного соединения, формируется в виде $k = f(k_A, k_B)$

*Не путать с *распределением открытых ключей*.

для некоторой функции f В данном случае аутентификация сторон и источника осуществляется на основе предположения, что общим секретным ключом k_{AB} обладают только два участника A и B.

Рассмотрим способы совместной выработки участниками общего ключа с использованием систем шифрования с открытыми ключами. В данном случае установление защищенного соединения, как правило, начинается с обмена аутентификационными данными для обеспечения контроля доступа. Далее проводится обмен сообщениями по открытому каналу связи для выработки общего ключа. В данном случае, так же как и в симметричном случае, используется термин «обмен ключами» (key exchange). Необходимость в аутентификации при этом остается до окончания сеанса протокола выработки ключа, так как очевидно, что ключевой обмен также должен быть аутентифицирован. После того как общий ключ сформирован, он будет использован в механизмах защиты дальнейшего трафика и, в частности, с его помощью будет обеспечена дальнейшая аутентификация.

Комбинация начальной аутентификации сторон и аутентификации ключевого обмена обеспечивается *аутентифицированным протоколом обмена ключами* (authenticated key exchange protocol).

Понятие безопасного аутентифицированного протокола обмена ключами (secure authenticated key exchange protocol) было введено У. Диффи, П. ван Ооршотом и М. Вейнером (W. Diffie, P. C. van Oorschot, M. J. Wiener) [42]. Говорят, что подобный протокол безопасен, если при каждом выполнении протокола двумя участниками гарантируется выполнение двух условий (первая сторона — A правильно выполняет протокол и признает идентификатор другой стороны):

1) если сторона A принимает идентификатор стороны B, записи в сообщениях, передаваемых обеими сторонами, осуществляются правильно;

2) никто кроме сторон A и B не может определить передаваемый ключ.

Таким образом, протокол обмена ключами должен обеспечивать выполнение свойства аутентификации сообщения (G2), гарантирующего аутентификацию источника и целостность ключа, и свойства аутентификации ключа (G7). Заметим, что в настоящее время свойство аутентификации ключа формулируется как одностороннее, в котором речь идет о предоставлении гарантий только одному из участников (см. подразд. 1.2).

Эти два свойства образуют минимум, необходимый для протокола обмена ключами. Однако для протокола желательно выполнение и других свойств:

- *защищенность от чтения назад* (perfect forward secrecy, G9) — раскрытие долговременного ключа не приведет к компрометации сеансовых ключей, которые были сгенерированы ранее; часто данное свойство понимают в более сильном варианте так, что выполнено еще одно дополнительное свойство: раскрытие сеансового ключа не приводит к компрометации ни долговременного ключа, ни других сеансовых ключей; это свойство обеспечивается, например, генерацией сеансового ключа с использованием протокола Диффи — Хеллмана, в котором экспоненты используются только в данном сеансе;
- *формирование новых ключей* (fresh key derivation, G10); ранее использовался термин «back traffic protection» — гарантия, что генерация каждого сеансового ключа выполняется независимо: новые ключи не зависят от предыдущих ключей и раскрытие сеансового ключа не приводит к компрометации ни предыдущих сеансовых ключей, ни будущих сеансовых ключей;
- *прямая аутентификация* (direct authentication); иначе — подтверждение правильности ключа (key confirmation, G8) — если по завершении протокола величины, использованные для генерации общего секрета, аутентифицированы, либо если каждая сторона может убедиться, что она знает сеансовый ключ; наоборот, говорят, что аутентификация не прямая, если по завершении протокола она не гарантируется, а для ее проведения необходимо осуществить дополнительный обмен на этих ключах и убедиться, что значения ключей у обеих сторон согласованы;
- *защита идентификаторов* (identity protection) — осуществляющий перехват не сможет узнать идентификаторы сторон, проводящих обмен.

Наконец, использование временны́х меток для защиты от повторной передачи часто нежелательно, так как сильно зависит от синхронизации часов.

По сути, основным протоколом открытого распределения ключей является протокол Диффи — Хеллмана, а все различие заключается в способах его усиления для повышения защищенности к известным атакам. Поэтому рассмотрим подробно, как происходило развитие механизмов защиты этого протокола.

Протокол Диффи — Хеллмана позволяет генерировать общий секрет (ключ) без какой-либо предварительной информации. Однако он обладает очевидной слабостью — не защищен от

атаки «нарушитель в середине». Один из путей защиты от атаки «нарушитель в середине» состоит в аутентификации открытых значений, передаваемых в протоколе. Это можно сделать двумя способами:

1) с помощью обмена аутентифицированными открытыми значениями, например, вложенными в сертификаты;

2) путем аутентификации открытых значений после обмена ими, например, используя код аутентичности сообщения или подписав их.

К сожалению, в обоих случаях пропадает возможность генерации общего секрета без какой-либо предварительной информации. Однако если открытые значения используются только однократно, то такой аутентифицированный протокол обеспечивает свойство защищенности от чтения назад.

8.2. Протокол Диффи — Хеллмана и его усиления

Протокол DH (ephemeral DH). Первый алгоритм открытого распределения ключей был предложен в 1976 г. У. Диффи и М. Хеллманом (W. Diffie, M. E. Hellman) [43]. Для его выполнения стороны должны договориться о значениях большого простого числа p и образующего элемента α мультипликативной группы $Z_p^* = \{1, 2, \quad , p-1\}$. Для выработки общего ключа k они должны сгенерировать случайные числа x $(1 \le x \le p-2)$ и y $(1 \le y \le p-2)$ соответственно. Затем они должны обменяться сообщениями в соответствии с протоколом:

$$A \to B \quad \alpha^x \bmod p,$$
$$A \leftarrow B \quad \alpha^y \bmod p.$$

Искомый общий ключ теперь вычисляется по формуле

$$k = (\alpha^y)^x = (\alpha^x)^y \bmod p.$$

Недостатком этого протокола является возможность атаки типа «противник в середине», состоящей в следующем: предположим, что злоумышленник C может осуществить подмену передаваемых абонентами сообщений, тогда, выбрав числа x^* и y^* и подменив сообщения $\alpha^x \bmod p$ и $\alpha^y \bmod p$ на $\alpha^{x^*} \bmod p$ и $\alpha^{y^*} \bmod p$ соответственно он может сформировать ключи:

$$k_{AC} = (\alpha^x)^{y^*} \bmod p, \quad k_{CB} = (\alpha^y)^{x^*} \bmod p$$

для связи с пользователями A и B соответственно. В результате злоумышленник получает возможность полностью контролировать обмен сообщениями между абонентами A и B. При этом они не смогут обнаружить подмену и будут уверены, что связываются непосредственно друг с другом.

Далее рассмотрим варианты усиления протокола, устраняющие этот недостаток.

Предварительное распределение (static DH). Предположим, что имеется доверенный центр, который может с использованием сертификатов связать идентификаторы участников со значениями вида α^x, которыми участники обмениваются в протоколе DH. В этом случае противник, контролирующий канал связи, не сможет навязать свои значения для выработки ключа, так как это будет сразу обнаружено при проверке сертификатов. Однако такой подход удобен только для предварительного распределения.

Каждый участник должен предварительно получить в центре сертификат вида

$$\mathrm{cert}_A = (A, \alpha^x, \mathrm{Sig}_T(A, \alpha^x)),$$

где Sig_T — подпись доверенного центра T

Теперь для выработки совместного значения ключа участники должны просто обменяться сертификатами:

$$\begin{aligned} A &\to B \quad \mathrm{cert}_A, \\ A &\leftarrow B \quad \mathrm{cert}_B, \end{aligned}$$

либо прочитать их из общего списка действующих сертификатов.

Такой протокол является защищенным относительно атаки «противник в середине». Вместе с тем формируемый ключ остается неизменным, и, таким образом, исчезает все преимущество открытого распределения.

Протоколы MTI. Интересный подход к защите протокола DH от атаки «противник в середине» был предложен Т. Мацумото, И. Такашима и Х. Имаи (T. Matsumoto, Y. Takashima, H. Imai) в 1986 г. [61]. Они предложили серию протоколов, предполагающих наличие у абонентов открытых ключей и использующих различные модификации процедуры выработки общего ключа.

Рассмотрим протокол MTI/A0. Предположим, что пользователи A, B имеют секретные ключи a $(1 \le a \le p-2)$ и b $(1 \le b \le p-2)$ соответственно и публикуют свои открытые клю-

чи $\beta_A = \alpha^a \bmod p$, $\beta_B = \alpha^b \bmod p$. Для выработки общего секретного ключа k они должны сгенерировать случайные числа x $(1 \le x \le p-2)$ и y $(1 \le y \le p-2)$ соответственно, а затем обменяться следующими сообщениями:

$$A \to B \quad \alpha^x \bmod p,$$
$$A \leftarrow B \quad \alpha^y \bmod p.$$

Теперь участники A и B вычисляют общий ключ $k = \alpha^{xb+ya} \bmod p$ соответственно по формулам

$$k_A = (\alpha^y)^a \beta_B^x \bmod p,$$
$$k_B = (\alpha^x)^b \beta_A^y \bmod p.$$

Любая подмена сообщений приведет к тому, что все стороны получат различные значения ключа, что в свою очередь приведет к невозможности чтения передаваемой информации. Тем самым свойство аутентификации ключа протокола при атаке «противник в середине» не нарушено. Вместе с тем этот протокол не обеспечивает аутентификации сторон и подтверждения правильности получения ключа.

Другие варианты протокола MTI отличаются способом вычисления общего ключа (табл. 8.1).

Среди этих протоколов только в первом (MTI/A0) пересылаемые сообщения не зависят от открытых ключей участников. Покажем, что даже после усиления этого протокола путем добавления к пересылаемым сообщениям сертификатов он остается уязвимым к подмене содержания передаваемых сообщений, т. е. что этот протокол не обеспечивает аутентификации сторон.

Рассмотрим пример атаки на протокол MTI/A0 [64], в которой нарушитель C подменяет сертификат участника A на свой собственный.

Таблица 8.1

Варианты протокола MTI

Протокол	m_1	m_2	k_A	k_B	k_{AB}
MTI/A0	α^x	α^y	$m_2^a \beta_B^x$	$m_1^b \beta_A^y$	α^{ax+by}
MTI/B0	β_B^x	β_A^y	$m_2^{a^{-1}} \alpha^x$	$m_1^{b^{-1}} \alpha^y$	α^{x+y}
MTI/C0	β_B^x	β_A^y	$m_2^{a^{-1}x}$	$m_1^{b^{-1}y}$	α^{xy}
MTI/C1	β_B^{xa}	β_A^{yb}	m_2^x	m_1^y	α^{abxy}

Пусть участник A получил сертификат:

$$\mathrm{cert}_A = (A, \beta_A, \mathrm{Sig}_T(A, \beta_A)).$$

Нарушитель C регистрирует для себя сертификат:

$$\mathrm{cert}_C = (C, \beta_A^e, \mathrm{Sig}_T(C, \beta_A^e)),$$

где e — произвольное число, предназначенное для того, чтобы скрыть связь с ключом участника A.

Далее нарушитель B использует следующий протокол:

$$\begin{array}{lll} A \rightarrow C(B) & & \mathrm{cert}_A, \alpha^x \bmod p, \\ C & \rightarrow B & \mathrm{cert}_C, \alpha^x \bmod p, \\ C & \leftarrow B & \mathrm{cert}_B, \alpha^y \bmod p, \\ A \leftarrow C(B) & & \mathrm{cert}_B, (\alpha^y)^e \bmod p. \end{array}$$

В итоге участник B будет уверен, что полученный в результате выполнения этого протокола ключ $k = \alpha^{aey+bx} \bmod p$ будет общим ключом с участником C, а не с участником A, который ничего об этом не подозревает. При этом участник A вычислит тот же ключ k. Поэтому вся информация, полученная от участника A и зашифрованная на этом ключе, будет восприниматься участником B как полученная от участника C, а не от A. Таким образом, данный протокол, хотя и обеспечивает аутентификацию ключа для участника A, в то же время не обеспечивает аутентификации сторон, а также аутентификации ключа для участника B. (Подумайте, можно ли модифицировать эту атаку для других вариантов протокола MTI.)

Открытое распределение ключей с использованием самосертифицируемых ключей. Схема, предложенная М. Гиролтом (M. Girault), основана на схеме RSA с простыми числами $p = 2p_1 + 1$, $q = 2q_1 + 1$, где p_1, q_1 — простые числа. Выбираем в кольце Z_n $(n = pq)$ элемент α порядка НОД$(p - 1, q - 1) = 2p_1q_1$. Пусть $de = 1 \bmod 2p_1q_1$, причем d — секретная экспонента, известная только центру выдачи сертификатов, e — открытая.

Участник A генерирует свое секретное число a_A $(1 < a_A < 2p_1q_1)$ и вычисляет открытое значение $\beta_A = \alpha^{a_A}$, представляет его в центр и получает в центре значение $p_A = (\beta_A - A)^d \bmod n$.

Напомним, что идентификатор абонента и его имя обозначены одним символом.

По значению p_A можно убедиться, что оно принадлежит участнику A. Для этого нужно вычислить значение $(p_A^e + A) \bmod n$

и проверить, совпадает ли оно с β_A. Поэтому значение p_A можно использовать в качестве сертификата.

Теперь можно воспользоваться следующим протоколом выработки ключа, аналогичным протоколу MTI/A0:

$$A \to B \quad A, p_A,\ \gamma_A = \alpha^{r_A} \bmod n,$$
$$A \leftarrow B \quad B, p_B,\ \gamma_B = \alpha^{r_B} \bmod n.$$

Общий ключ $k_{AB} = \alpha^{r_A a_B + r_B a_A} \bmod n$ вычисляется теперь участниками A и B по формулам

$$k_{AB} = \gamma_B^{a_A}(p_B^e + B)^{r_A} \bmod n = \gamma_A^{a_B}(p_A^e + A)^{r_B} \bmod n.$$

Рассмотрим еще три варианта протоколов открытого распределения ключей, в которых развиваются подходы, примененные в протоколах серии MTI [31].

Протокол KEA. Алгоритм открытого распределения ключей KEA (Key Exchange Algorithm) был разработан Агентством национальной безопасности (АНБ) США и рассекречен в мае 1998 г. [68]. Это протокол распределения ключей из пакета криптоалгоритмов FORTEZZA, разработанного АНБ в 1994 г. Он похож на протокол MTI/A0 [61], только в качестве α берется не примитивный элемент поля Z_p, а элемент порядка q, где q — простой делитель числа $p-1$.

Протокол KEA выполняется следующим образом:

1) участники A и B получают аутентичные копии открытых ключей друг друга β_A и β_B; далее участники A и B обмениваются сообщениями:

$$A \to B \quad m_A = \alpha^x \bmod p,$$
$$A \leftarrow B \quad m_B = \alpha^y \bmod p;$$

2) участник A выбирает случайный элемент $x \in \overline{1, q-1}$ и отправляет участнику B сообщение

$$m_A = \alpha^x \bmod p;$$

3) участник B выбирает случайный элемент $y \in \overline{1, q-1}$ и отправляет участнику A сообщение

$$m_B = \alpha^y \bmod p;$$

4) участник A проверяет:

$$1 < m_B < p; \quad m_B^q \equiv 1 (\mathrm{mod} p);$$

если какая-либо из проверок неудачна, то участник A прерывает сеанс протокола с ошибкой; иначе, участник A вычисляет общий секрет

$$K = \beta_B^x + m_B^a \bmod p;$$

если $K = 0$, то участник A прерывает сеанс протокола с ошибкой;

5) участник B проверяет:

$$1 < m_A < p; \quad m_A^q \equiv 1(\mathrm{mod} p);$$

если какая-либо из проверок неудачна, то участник B прерывает сеанс протокола с ошибкой; иначе, участник B вычисляет общий секрет

$$K = \beta_A^y + m_A^b \bmod p;$$

если $K = 0$, то участник B прерывает сеанс протокола с ошибкой;

6) участники A и B вычисляют 80-битный ключ сеанса:

$$k = h(K) = h(\alpha^{ay} + \alpha^{bx}),$$

где $h = \mathrm{kdf}$ — функция вычисления ключа (key derivation function) из схемы симметричного шифрования SKIPJACK [68].

Перечислим возможные уязвимости [92] и способы защиты от них.

Проверка условия принадлежности подгруппе порядка q. Предположим, участник A не проверил, что $m_B^q \equiv 1(\mathrm{mod} p)$. Тогда противник на стороне участника B может получить информацию о статическом секретном ключе a участника A, используя вариант *атаки малой подгруппы* [59]. Она состоит в следующем: предположим, что число $p - 1$ имеет простой делитель l малой длины (например, 40 бит). Пусть $\gamma \in Z^*$ и имеет порядок l. Если участник B посылает участнику A сообщение $m_B = \gamma$, то участник A вычисляет

$$K = \alpha^{bx} + \gamma^a \bmod p, \quad k = h(K).$$

Предположим, что участник A отправляет участнику B зашифрованное сообщение $c = E_k(\mathrm{data})$, где E — алгоритм симметричного шифрования, а открытый текст data имеет некоторую распознаваемую структуру. Для каждого d $(0 \le d \le l - 1)$ участник B вычисляет

$$K' = \alpha^{bx} + \gamma^d \bmod p, \quad k' = h(K'), \quad (\mathrm{data})' = E_{k'}^{-1}(c).$$

Если $(\text{data})'$ обладает требуемой структурой, то участник B делает вывод, что $d = a \bmod l$, таким образом, получив некоторую информацию о ключе a. Эта процедура может быть повторена для различных малых простых делителей l.

Проверка условия принадлежности интервалу $[2, p-1]$. Предположим, что участник A не сделал проверку:

$$1 < \beta_B < p; \quad 1 < m_B < p,$$

тогда противник C может применить следующую атаку неизвестного общего ключа:

$$\begin{array}{lcll} A \rightarrow & C(B) & & m_A = \alpha^x \bmod p, \\ & C & \rightarrow B & m_A, \\ & C & \leftarrow B & m_B = \alpha^y \bmod p, \\ A \leftarrow & C(B) & & m'_B = 1 \bmod p. \end{array}$$

Противник C выбирает $\beta_C = 1 \bmod p$ и сертифицирует его в качестве своего открытого ключа. Затем противник C отправляет участнику B от своего имени перехваченный открытый ключ m_A участника A. Участник B отвечает противнику C значением m_B, противник C отправляет участнику A от имени участника B значение $m'_B = 1$. Теперь участник A вычисляет $K_{AB} = \alpha^{bx} + 1$, и участник B вычисляет $K_{BC} = \alpha^{bx} + 1$. Таким образом, не подозревая об этом, участник B выработал общий секрет с участником A. Поэтому в таком виде протокол не обеспечивает аутентификацию сторон. Раскрытия ключа здесь не происходит, так как участник C не знает секретного ключа участника B. Но свойство аутентификации ключа нарушено со стороны участника B, так как он думает, что выработал общий ключ с участником C, а на самом деле — с участником A.

Использование хеш-функции. Функция h предназначена для вычисления сеансового ключа из общего секретного ключа K. Одна из целей введения этой функции заключается в смешивании «сильных» бит и потенциально «слабых» бит числа K — «слабые» биты представляют собой информацию о ключе K, которая может быть безошибочно предугадана со значимой пользой для противника.

Еще одним поводом для введения функции h является наличие алгебраической зависимости между общим секретным ключом K, секретными и открытыми ключами. Нарушение этой зависимости может помочь отразить некоторые атаки с известным ключом (known-key attacks), например такие как атака тре-

угольника Бурместера (M. Burmester) [36]. Она состоит в следующем: противник C с ключевой парой (c, α^c) наблюдает за сеансом протокола между участниками A и B, обменивающимися сообщениями α^x и α^y; в результате получается общий секрет $K_{AB} = \alpha^{ay} + \alpha^{bx} \bmod p$. Затем участник C инициирует сеанс связи с участником A, заново отправляя сообщение α^y от своего имени; в результате получается секрет, который может вычислить только участник A:

$$K_{AC} = \alpha^{ay} + \alpha^{cx'} \bmod p,$$

где $\alpha^{x'}$ — сообщение A.

Аналогично участник C начинает сеанс протокола с участником B, отправляя сообщение α^x от своего имени; в результате получается секрет, вычисляемый только участником B:

$$K_{BC} = \alpha^{bx} + \alpha^{cy'} \bmod p,$$

где $\alpha^{y'}$ — сообщение B.

Если участник C может каким-либо образом узнать ключи K_{AC} и K_{BC}, то он сможет вычислить

$$K_{AB} = K_{AC} + K_{BC} - \alpha^{cx'} - \alpha^{cy'} \bmod p.$$

Условие $K \neq 0$ — лишнее, так как если $K = 0$, то $\alpha^{bx} \equiv \equiv -\alpha^{ay} \bmod p$. Однако это невозможно, поскольку тогда

$$(\alpha^{bx})^q \equiv (-\alpha^{ay})^q \equiv (-1)^q \equiv -1 (\mathrm{mod} p),$$

что противоречит условию

$$(\alpha^b)^q \equiv (\alpha^y)^q \equiv 1 (\mathrm{mod} p).$$

Замечание по безопасности. Протокол KEA не обеспечивает защиту от чтения назад, поскольку противник, знающий ключи a и b, может вычислить все сеансовые ключи, сформированные участниками A и B.

Имеются и другие варианты вычисления общего ключа, рекомендованные в проектах стандартов ANSI X9.42 [9], ANSI X9.63 [8] и IEEE P1363 [12].

Протокол «унифицированная модель» (unified model). Данный протокол, предложенный Р. Анни, Д. Джонсоном и М. Матиасом (R. Ankney, D. Johnson, M. Matyas) в 1995 г., отличается от протокола KEA тем, что в нем применяются сертификаты открытых ключей:

$$A \rightarrow B \quad \mathrm{cert}_A, m_A = \alpha^x \bmod p,$$
$$A \leftarrow B \quad \mathrm{cert}_B, m_B = \alpha^y \bmod p.$$

Общий ключ $k = h(\alpha^{ab}, \alpha^{xy})$ вычисляется участниками A и B по формулам

$$k = h(\beta_B^a, m_B^x) = h(\beta_A^b, m_A^y),$$

где h — бесключевая хеш-функция.

Протокол MQV. Этот протокол, предложенный Р. Лоу, А. Менезисом, М. Ку, Д. Солинасом и С. Ванстоуном (R. Law, A. Menezes, M. Qu, J. Solinas, S. Vanstone) в 1998 г., отличается от протокола KEA тем, что формирует общий ключ в виде $k = h(K)$, где $K = \alpha^{s_A s_B} \bmod p$ вычисляется участниками A и B по формулам

$$K = (m_B \beta_B^{\overline{m}_B})^{s_A} \bmod p = (m_A \beta_A^{\overline{m}_A})^{s_B} \bmod p,$$

где

$$s_A = x + a\overline{m}_A \bmod q;$$

$$s_B = y + b\overline{m}_B \bmod q;$$

$$\overline{m} = (m \bmod 2^{\lceil f \rceil}) + 2^{\lceil f \rceil};$$

f — битовая длина записи числа q.

8.3. Аутентифицированные протоколы

Рассмотрим способы построения аутентифицированных протоколов открытого распределения ключей, в которых решается задача обеспечения аутентификации сторон и ключа. Основная идея построения таких протоколов заключается в превращении базовых двухпроходных версий этих протоколов в трехпроходные протоколы путем добавления в передаваемые сообщения аутентификационных полей, представляющих собой цифровые подписи или коды аутентичности сообщений.

Применение цифровых подписей

Протокол STS. Первая попытка построения таких протоколов была предпринята в протоколе, называемом STS (station-to-station), созданном У. Диффи, П. ван Ооршотом и М. Вейнером (W. Diffie, P. van Oorschot, M. Wiener) в 1992 г. [42].

Протокол STS предполагает, что пользователи применяют цифровую подпись, которой подписывают передаваемые по протоколу Диффи — Хеллмана сообщения. Сообщения протокола STS можно записать в виде

$A \to B \quad A, B, m_A = \alpha^x \bmod p,$
$A \leftarrow B \quad B, A, m_B = \alpha^y \bmod p, E_k(\mathrm{Sig}_B(m_B, m_A)),$
$A \to B \quad A, B, E_k(\mathrm{Sig}_A(m_A, m_B)).$

Здесь Sig_A, Sig_B — цифровые подписи пользователей A и B соответственно; $k = \alpha^{xy} \bmod p$ — искомый общий ключ.

Вставка во второе и третье сообщения протокола значений цифровых подписей позволяет гарантировать достоверность получения сообщения именно от того пользователя, от которого это сообщение получено. Шифрование значений подписей пользователей с помощью симметричного алгоритма E введено для того, чтобы обеспечить взаимное подтверждение правильности вычисления значения ключа, так как при неверно вычисленном ключе невозможно получить верные значения цифровых подписей.

Как обнаружил Г. Лоу (G. Lowe) в 1996 г., в таком виде протокол STS обладает следующим недостатком: нарушитель C, используя свой законный обмен с участником B, может применить против участника A атаку, в результате которой он может убедить участника A в том, что он выступает от имени B:

$$\begin{array}{lll} A \to & C(B) & A, B, m_A, \\ & C \to B & C, B, m_A, \\ & C \leftarrow B & B, C, m_B,\ E_k(\mathrm{Sig}_B(m_B, m_A)), \\ A \leftarrow & C(B) & B, A, m_B,\ E_k(\mathrm{Sig}_B(m_B, m_A)), \\ A \to & C(B) & A, B, E_k(\mathrm{Sig}_A(m_A, m_B)). \end{array}$$

Сеанс с участником B остается незавершенным, так как нарушитель C, не зная секретного ключа участника A, не сможет подобрать правильный ответ для участника B. Поэтому любое его сообщение на третьем шаге будет участником B отвергнуто.

Хотя данная атака и не представляет реальной опасности, так как при этом нарушитель не будет знать секретного ключа $k = \alpha^{xy} \bmod p$ и поэтому не сможет читать передаваемые сообщения, передаваемые от A к B. Однако в результате участник A не будет ничего подозревать и примет нарушителя C за участника B.

Таким образом, данная атака, хотя и не затрагивает свойства аутентификации ключа для участника A, показывает, что данный протокол не обеспечивает аутентификации сторон, а также аутентификации ключа для участника B, так как последний будет полагать, что сформировал общий ключ с участником C, а на самом деле — с участником A.

Данная атака является разновидностью *атаки с неизвестным общим ключом* (unknown key-share attack — UKS attack), в результате которой участники A и B формируют общий ключ, причем хотя бы один из них уверен, что он на самом деле сформировал общий ключ с третьим участником C.

Этот недостаток протокола STS можно устранить, если проводить аутентификацию не только передаваемых сообщений m_A, m_B, но и идентификаторов сторон, например, заменив $\mathrm{Sig}_A(m_A, m_B)$ и $\mathrm{Sig}_B(m_B, m_A)$ на $\mathrm{Sig}_A(A, B, m_A, m_B)$ и $\mathrm{Sig}_B(B, \mathbf{A}, m_B, m_A)$ соответственно.

Протокол DHKE. В 1998 г. В. Шоуп (V. Shoup) предложил четыре варианта протоколов [79], аналогичных протоколу STS. Рассмотрим протокол DHKE-1 и на его примере продемонстрируем еще один тип атаки, называемой *двусторонней атакой с неизвестным общим ключом* (bilateral unknown key-share attack — BUKS attack) [38]. Пусть участники A и B обладают сертификатами своих открытых ключей, полученными из центра сертификации, а также могут сформировать свою цифровую подпись. Сообщения протокола DHKE-1 имеют вид:

$$\begin{aligned} A &\to B \quad \mathrm{cert}_A, m_A = \alpha^x \bmod p, \mathrm{Sig}_A(m_A, B), \\ A &\leftarrow B \quad \mathrm{cert}_B, m_B = \alpha^y \bmod \mathbf{p}, k, \mathrm{Sig}_B(m_A, m_B, k, A), \\ A &\to B \quad k_1. \end{aligned}$$

Здесь k — случайное число, используемое в качестве ключа хеш-функции при вычислении результирующего сеансового ключа k_2 по формуле

$$(k_1, k_2) = h_k(\alpha^{xy} \bmod p),$$

где $(k_1, k_2) = k_1 || k_2$ — конкатенация.

Значение k_1 передается участником A в открытом виде и применяется для проверки участником B правильности вычисления результирующего сеансового ключа k_2 и аутентификации участника A.

Двусторонняя атака с неизвестным общим ключом заключается в том, что два участника C и D, вступив в сговор, могут ввести в заблуждение участников A и B, сформировавших общий ключ. При этом участник A будет уверен, что он сформировал общий ключ с участником C, а участник B будет уверен, что он сформировал общий ключ с участником D.

Атака осуществляется путем открытия двух сеансов по следующему сценарию:

$$
\begin{array}{llll}
A \to & C & & \mathrm{cert}_A, m_A, \mathrm{Sig}_A(m_A, C), \\
& C \to & D & A, B, m_A, \\
& & D \to B & \mathrm{cert}_D, m_A, \mathrm{Sig}_D(m_A, B), \\
& & D \leftrightarrow B & \mathrm{cert}_B, m_B, k, \mathrm{Sig}_B(m_A, m_B, k, D), \\
& C \leftarrow & D & B, A, m_B, k, \\
A \leftarrow & C & & \mathrm{cert}_C, m_B, \mathrm{Sig}_C(m_A, m_B, k, A), \\
A \to & C & & k_1, \\
& C \to & D & k_1, \\
& & D \to B & k_1.
\end{array}
$$

Применение хеш-функций, задаваемых ключом

Другой способ получения аутентифицированных версий рассмотренных выше протоколов состоит в том, что для выполнения свойств аутентификации сторон и ключа, а также для взаимного подтверждения правильности получения ключа в передаваемые сообщения добавляют значения кодов аутентичности сообщений, вычисляемых как значения хеш-функции от передаваемых данных. Рассмотрим несколько примеров.

Модифицированный протокол STS. В 2004 г. К. Бойд и А. Матура (C. Boyd, A. Mathuria) [34] предложили следующую модификацию протокола STS:

$$
\begin{array}{ll}
A \to B & A, B, m_A = \alpha^x \bmod p, \\
A \leftarrow B & B, A, m_B = \alpha^y \bmod p, \mathrm{Sig}_B(m_B, m_A), h_{k_0}(m_B, m_A), \\
A \to B & A, B, \mathrm{Sig}_A(m_A, m_B), h_{k_0}(m_A, m_B),
\end{array}
$$

где $k_0 = f(k)$ — ключевой параметр хеш-функции, вычисляемый как значение некоторой функции от результирующего сеансового ключа $k = \alpha^{xy} \bmod p$.

Для данной версии протокола также можно применить двустороннюю атаку с неизвестным общим ключом:

$$
\begin{array}{llll}
A \to & C & & A, B, m_A, \\
& C \to & D & A, B, m_A, \\
& & D \to B & D, B, m_A, \\
& & D \leftarrow B & B, D, m_B, \mathrm{Sig}_B(m_B, m_A), h_{k_0}(m_B, m_A), \\
& C \leftarrow & D & B, A, m_B, h_{k_0}(m_B, m_A), \\
A \leftarrow & C & & C, A, m_B, \mathrm{Sig}_C(m_B, m_A), h_{k_0}(m_B, m_A), \\
A \to & C & & A, C, \mathrm{Sig}_A(m_A, m_B), h_{k_0}(m_A, m_B), \\
& C \to & D & A, B, h_{k_0}(m_A, m_B), \\
& & D \to B & D, B, \mathrm{Sig}_D(m_A, m_B), h_{k_0}(m_A, m_B).
\end{array}
$$

В результате участники C и D, вступившие в сговор, вводят в заблуждение участников A и B, сформировавших общий ключ. При этом участник A уверен, что он сформировал общий ключ с участником C, а участник B уверен, что он сформировал общий ключ с участником D.

Протокол «унифицированная модель с подтверждением». Данный протокол выглядит так:

$$\begin{aligned} A &\to B \quad \mathrm{cert}_A,\ m_A = \alpha^x \bmod p, \\ A &\leftarrow B \quad \mathrm{cert}_B,\ m_B = \alpha^y \bmod p,\ h_{k'}(2, B, A, m_B, m_A), \\ A &\to B \quad h_{k'}(3, A, B, m_A, m_B). \end{aligned}$$

Ключ k', отличный от ключа k, вычисляется с помощью независимой от функции h бесключевой хеш-функции. Например, можно модифицировать бесключевую хеш-функцию h путем дописывания перед аргументом каких-нибудь констант:

$$\begin{aligned} k &= h(10, \alpha^{ab}, \alpha^{xy}), \\ k' &= h(01, \alpha^{ab}, \alpha^{xy}). \end{aligned}$$

Такой способ модификации защищает протокол от атак, связанных с повтором и отражением перехваченных сообщений, так как проводится аутентификация направления (от отправителя к получателю) и передаваемых значений m_A, m_B. Поэтому каждая из сторон может убедиться в подлинности другой стороны и актуальности передаваемых сообщений. Тем самым выполняются свойства аутентификации сторон и ключа и подтверждается правильность формируемого ключа.

Подобным образом можно модифицировать протоколы KEA и MQV

Однопроходные версии протоколов

В некоторых случаях оказывается удобным свести процедуру выработки общего ключа к передаче только одного сообщения. Например, в протоколах без установления соединения желательно свести к минимуму число передаваемых сообщений.

Для построения таких протоколов можно применить способ, позволяющий модифицировать двухпроходные протоколы в однопроходные. Будем вместо сообщения $m_B = \alpha^y \bmod p$, отправляемого вторым участником, использовать его открытый ключ. Тогда, например, однопроходная модификация протокола MQV

будет выглядеть следующим образом: участник A генерирует значение x с условием $m_A = \alpha^x \bmod p \neq 1$, вычисляет значения

$$s_A = x + a\overline{m}_A \bmod q,$$

$$K = (\beta_B \beta_B^{\overline{\beta}_B})^{s_A} \bmod p.$$

Если в итоге получилось значение $K = 1$, он осуществляет повторную генерацию. Затем он отправляет участнику B сообщение

$$A \to B \quad \mathrm{cert}_A,\ m_A = \alpha^x \bmod p.$$

Получив такое сообщение, участник B сначала проверяет условие $m_A \neq 1$, затем вычисляет значения

$$s_B = b + b\overline{\beta}_B \bmod q,$$

$$K = (m_A \beta_A^{\overline{m}_A})^{s_B} \bmod p = \alpha^{s_A s_B} \bmod p,$$

проверяет условие $K \neq 1$.

Если требуется аутентифицировать данный протокол и подтвердить правильность получения ключа для участника B, то можно использовать следующий протокол:

$$A \to B \quad \mathrm{cert}_A,\ m_A = \alpha^x \bmod p,\ h_{k'}(A, B, m_A, \beta_B),$$

где $k' = h(01, K)$.

Теперь можно дополнительно проверить правильность полученного значения ключа и подлинность участника A. Вместе с тем в таком виде протокол не защищает от атаки с повторной передачей ранее переданного сообщения, которая приведет к повторному принятию ключа участником B. Для защиты от этой атаки нужно вставлять в сообщение, например, метку времени или элементы специальной последовательности.

Другие примеры аутентифицированных протоколов открытого распределения ключей будут рассмотрены в гл. 11.

Контрольные задания

1. Каков основной недостаток протокола распределения ключей Диффи — Хеллмана и каковы пути его устранения?

2. Покажите устойчивость к атаке «противник в середине» для всех четырех вариантов протокола MTI.

3. Какие из рассмотренных выше протоколов не обеспечивают свойства аутентификации ключа?

4. Какие из рассмотренных выше протоколов обеспечивают свойство взаимного подтверждения правильности получения ключа?

5. В чем состоит преимущество использования самосертифицируемых ключей?

6. Покажите, что протокол STS можно аутентифицировать, если проводить аутентификацию не только передаваемых сообщений α^x и α^y, но и идентификаторов сторон.

7. Покажите, что в качестве меры защиты протоколов от двусторонней атаки с неизвестным общим ключом можно вводить зависимость результирующего общего сеансового ключа от идентификаторов и открытых ключей сторон и передаваемых сообщений. Например, для протокола DHKE-1 и модифицированного протокола STS можно принять соответственно:

$$(k_1, k_2) = h_k(A, B, m_A, m_B, \mathrm{cert}_A, \mathrm{cert}_B, \alpha^{xy} \bmod p),$$

$$k = h(A, B, m_A, m_B, \mathrm{pk}_A, \mathrm{pk}_B, \alpha^{xy} \bmod p).$$

8. Как модифицировать протоколы KEA и MQV, чтобы для них выполнялось свойство взаимного подтверждения правильности получения ключа?

9. Как преобразовать протоколы KEA и «унифицированная модель с подтверждением» в однопроходные с подтверждением правильности получения ключа?

Глава 9

Предварительное распределение ключей

9.1. Схемы предварительного распределения ключей в сети связи

Большинство криптографических систем требует проведения предварительного распределения секретных ключей. Для предварительного распределения стороны могут обменяться ключами при личной встрече, либо поручить доставку ключей специально назначенному доверенному курьеру, либо использовать для передачи некоторый выделенный защищенный канал. В зависимости от назначения криптографической системы иногда оказывается удобным распределять не сами ключи, а некоторые вспомогательные ключевые материалы, на основании которых каждый участник или группа участников может самостоятельно вычислить необходимый ключ, используя для этого некоторую установленную заранее процедуру.

Рассмотрим ситуации, в которых необходимо проводить предварительное распределение ключей.

Проблема предварительного распределения ключей в сети связи с большим числом абонентов заключается в следующем. Если число абонентов сети засекреченной связи невелико, то и число распределяемых ключей невелико. Для больших же сетей распределение ключей становится очень серьезной проблемой. Она состоит в том, что для сети, в которой работают n абонентов, необходимо выработать заранее и хранить в дальнейшем $n(n-1)/2$ ключей. Кроме того, каждому абоненту сети необходимо передать ключи для связи с остальными $n-1$ абонентами, которые абонент должен постоянно хранить. Например, для сети со 100 абонентами нужно сгенерировать и хранить почти 5 000 ключей, причем каждый абонент при этом должен хранить у себя 99 ключей.

Для уменьшения объема хранимой ключевой информации применяют различные схемы предварительного распределения ключей в сети связи. Их суть заключается в том, что в действительности вначале происходит распределение не самих ключей,

а некоторых вспомогательных ключевых материалов, представляющих секретные данные меньшего объема. На основании этих материалов каждый абонент сети может самостоятельно вычислить по некоторому алгоритму необходимый для связи ключ. Такой подход позволяет уменьшить объем как хранимой, так и распределяемой секретной информации.

Схема предварительного распределения ключей в сети связи должна быть устойчивой относительно компрометации части ключей, в том числе вследствие обмана или сговора некоторых пользователей, и одновременно гибкой, т. е. быстро восстанавливаться как после частичной компрометации, так и после подключения новых пользователей.

Свойства схем предварительного распределения ключей

Определение 4. Пусть имеется сеть связи, в которой работают абоненты $A_1, \quad , A_n$, пусть также K, P, Q, R — некоторые конечные множества: K — множество ключей, P — множество исходных ключевых параметров, хранящихся в центре распределения ключевых материалов, Q — множество значений ключевых материалов абонентов, R — множество значений открытой информации, связанной с абонентами. *Схема предварительного распределения ключей* для сети с n абонентами $S(n) = (K, P, Q, R, \mathcal{A}_0, \mathcal{A}_1)$ задается двумя алгоритмами:

1) $\mathcal{A}_0 \quad P \times R \to Q$ — алгоритм формирования секретных ключевых материалов абонентов;

2) $\mathcal{A}_1 \quad Q \times R \to K$ — алгоритм вычисления ключа парной связи между двумя абонентами, удовлетворяющий условию:

$$\mathcal{A}_1(q_i, r_j) = \mathcal{A}_1(q_j, r_i), \quad q_i, q_j \in Q, \quad r_i, r_j \in R, \quad i \geq 1, \quad j \leq n.$$

Открытая информация $r_1, \ldots, r_n \in R$, связанная с абонентами $A_1, \ldots, A_n$ соответственно, располагается в открытом доступе. Каждый абонент A_i получает в центре распределения ключевых материалов набор секретных ключевых материалов:

$$\mathcal{A}_0(p, r_i) = q_i, \quad 1 \leq i \leq n.$$

Теперь абонент A_i вычисляет ключ k_{ij} для связи с абонентом A_j по правилу:

$$\mathcal{A}_1(q_i, r_j) = k_{ij}, \quad i \geq 1, \quad j \leq n.$$

При $i = j$ ключ k_{ii} обычно не рассматривают, хотя он может быть использован абонентом A_i для закрытия информации при хранении.

При различных $r_i \in R$ ключевые материалы абонентов могут иметь разную длину: пусть для определенности $\mathcal{A}_0(P, r_i) = Q_i \subseteq \subseteq K^{t_i} \subseteq Q$ $(1 \leq i \leq n)$.

На алгоритмы $\mathcal{A}_0$ и $\mathcal{A}_1$ логично наложить следующие априорные ограничения: пусть на множестве P задано равномерное распределение, тогда, во-первых, при всех $r_i \in R$ $(1 \leq i \leq n)$ алгоритм $\mathcal{A}_0$ должен давать равномерное распределение на множестве значений $Q_i = K^{t_i}$. Иначе противник сможет получить наиболее и наименее вероятные значения ключевых материалов абонентов. Во-вторых, при любом фиксированном $r_i \in R$ $(1 \leq i \leq n)$ и равномерном распределении на множестве $Q_i = K^{t_i}$ алгоритм $\mathcal{A}_1$ должен также давать равномерное распределение на множестве K.

Эти ограничения можно сформулировать в виде двух условий.

Условие 1. При любых $r_i \in R$, $q_i \in Q_i$, $1 \leq i \leq n$ уравнение $\mathcal{A}_0(p, r_i) = q_i$ имеет одинаковое число решений относительно $p \in P$.

Условие 2. При любых $r_i \in R$, $k \in K$, $1 \leq i \leq n$ уравнение $\mathcal{A}_1(q_i, r_i) = k$ имеет одинаковое число решений относительно $q_i \in Q_i$.

Определим основное свойство, характеризующее криптографическое качество схемы предварительного распределения ключей.

Определение 5. Пусть $1 \leq m \leq n - 2$. Говорят, что схема предварительного распределения ключей $S(n)$ является стойкой к m-кратной компрометации ключей*, если после того как противнику станут известны ключевые материалы m абонентов, он не сможет получить никакой информации о ключах парной связи остальных абонентов.

Найдем критерий того, что схема $S(n)$ обладает этим свойством.

Предположим, что противнику известны ключевые материалы $q_1, \ldots, q_m$. Поэтому он может вычислить связные ключи $k_{i,1}, \ldots, k_{i,m}$ для любого абонента A_i $(m + 1 \leq i \leq n)$, так как

*В англоязычной литературе принят термин *устойчива к сговору т абонентов*.

$\mathcal{A}_1(q_i, r_1) = k_{i,1} = k_{1,i} = \mathcal{A}_1(q_1, r_i)$. Теперь для нахождения неизвестного ему ключа $k_{i,m+1}$ $(m+1 < i \leq n)$ противник может составить следующую систему уравнений:

$$\begin{cases} \mathcal{A}_1(q_i, r_1) = k_{i,1}, \\ \\ \mathcal{A}_1(q_i, r_m) = k_{i,m}, \\ \mathcal{A}_1(q_i, r_{m+1}) = k_{i,m+1}. \end{cases} \tag{9.1}$$

Значения q_i, $k_{i,m+1}$ противнику неизвестны. Так как схема предварительного распределения ключей является стойкой к m-кратной компрометации ключей, то при любых значениях $k_{i,1}, \ldots, k_{i,m}$ ключ $k_{i,m+1}$ также может принимать любое значение из множества K. Поэтому чтобы противник не смог получить никакой информации о значении ключа $k_{i,m+1}$, каждое значение этого ключа должно появляться с такой же вероятностью, что и все остальные. Поэтому должно выполняться условие 3.

Условие 3. Система (9.1) при любых $r_1, \ldots, r_{m+1} \in R$, $k_{i,1}$, $k_{i,m+1} \in K$ $(1 \leq i \leq n)$ имеет одинаковое число решений относительно $q_i \in Q_i$.

Теорема 11. Пусть $1 \leq m \leq n-2$. Говорят, что схема предварительного распределения ключей между n абонентами, удовлетворяющая условию 1, является стойкой к m-кратной компрометации ключей в том и только в том случае, если выполняется условие 3.

Доказательство. Необходимость выполнения условия 3 показана выше. Докажем достаточность. В силу условия 1 случайная величина q_i имеет равномерное распределение. Поэтому условие 3 гарантирует, что при фиксированных значениях ключей $k_{i,1}, \ldots, k_{i,m} \in K$ ключ $k_{i,m+1}$ также будет распределен равномерно, и противник не сможет получить никакой информации о его значении.

Теорема доказана.

Следствие 1. Если при $1 \leq m \leq n-2$ схема предварительного распределения ключей между n абонентами является стойкой к m-кратной компрометации ключей, то:

1) каждый абонент должен хранить не менее $(m+1)\log_2 |K|$ бит ключевых материалов;

2) центр распределения ключевых материалов должен хранить не менее $\frac{1}{2}m(m+1)\log_2 |K|$ бит исходных ключевых параметров.

Доказательство. В силу условия 3 при любых значениях $k_{i,1}, \quad , k_{i,m+1}$ $(1 \le i \le n)$ из множества K система (9.1) имеет хотя бы одно решение. Поэтому должно выполняться неравенство

$$l_i = \log_2 |Q_i| \ge \log_2(|K|^{m+1}) = (m+1)\log_2 |K|.$$

Пусть $m < i < j$; рассмотрим систему уравнений

$$A_1(A_0(p,s), r_t) = k_{st}; \quad s < t; \; s,t \in \{1,\ldots,m,i,j\}.$$

При компрометации ключей абонентов $A_1, \ldots, A_m$ противник не должен получить никакой информации о ключе абонентов A_i и A_j. Поэтому при всех $|K|^{\frac{1}{2}m(m+1)}$ значениях правой части эта система должна иметь хотя бы одно решение. Поэтому

$$|P| \ge |K|^{\frac{1}{2}m(m+1)}$$

Следствие доказано.

Назовем схему предварительного распределения ключей *оптимальной*, если для нее достигаются нижние оценки из следствия 1 на объем ключевых материалов абонентов.

Схема Блома

В качестве примера рассмотрим схему предварительного распределения ключей между n абонентами, предложенную Р. Бломом (R. Blom) [33], в которой процедура вычисления ключа заключается в вычислении значения некоторого симметрического многочлена над конечным полем.

Выберем поле F, имеющее конечное, но достаточно большое число элементов, и зафиксируем n различных элементов $r_1, \ldots, r_n \in F$, отличных от нуля. Каждый элемент r_i припишем абоненту A_i $(i = \overline{1,n})$. Эти элементы не являются секретными и могут храниться на общедоступном сервере сети. Выберем теперь многочлен над полем F вида

$$f(x,y) = \sum_{s=0}^{m} \sum_{t=0}^{m} a_{st} x^s y^t,$$

где $a_{st} = a_{ts}$; $s \ne t$; $s,t = \overline{0,m}$.

Число m должно изменяться в пределах $1 \le m \le n-1$; оно является параметром метода. Коэффициенты a_{st} многочле-

на являются секретными и должны храниться только в центре распределения ключей. Каждый абонент A_i получает в качестве ключевых материалов набор $q_i = (a_0^{(i)}, a_1^{(i)}, \ldots, a_m^{(i)})$, состоящий из коэффициентов многочлена:

$$g_i(x) = f(x, r_i) = a_0^{(i)} + a_1^{(i)} x + \quad + a_m^{(i)} x^m$$

Для связи между абонентами A_i и A_j теперь можно использовать общий ключ k_{ij}:

$$k_{ij} = k_{ji} = f(r_i, r_j) = g_j(r_i) = g_i(r_j),$$

который легко могут вычислить оба абонента.

При использовании данной схемы каждый абонент должен хранить $m + 1$ секретных значений вместо $n - 1$, общее же число секретных коэффициентов многочлена f равно $m(m + 1)/2$.

Нам потребуется следующий известный факт.

Лемма 6. Если элементы $r_1, r_2, \ldots, r_{m+1}$ поля F попарно различны и отличны от нуля, то определитель матрицы

$$\begin{pmatrix} 1 & r_1 & r_1^2 & & r_1^m \\ 1 & r_2 & r_2^2 & & r_2^m \\ & & \ldots & & \\ 1 & r_{m+1} & r_{m+1}^2 & & r_{m+1}^m \end{pmatrix}$$

называемый определителем Вандермонда, равен

$$\prod_{1 \le i < j \le m} (r_j - r_i).$$

В частности, матрица обратима над полем F

Теорема 12. Схема Блома предварительного распределения ключей между n абонентами, использующая многочлен степени m $(1 \le m \le n - 1)$ является стойкой к m-кратной компрометации ключей.

Доказательство. Запишем выражение для общего ключа абонентов A_i, A_j, вычисляемого по формуле:

$$k_{ij} = f(r_i, r_j) = \sum_{s=0}^{m} \sum_{t=0}^{m} a_{st} r_i^s r_j^t,$$

в матричном виде:

$$k_{ij} = (1, r_i, r_i^2, \ldots, r_i^m) \begin{pmatrix} a_{00} & a_{01} & & a_{0m} \\ a_{10} & a_{11} & & a_{1m} \\ & & & \\ a_{m0} & a_{m1} & & a_{mm} \end{pmatrix} \begin{pmatrix} 1 \\ r_j \\ r_j^2 \\ \\ r_j^m \end{pmatrix}$$

Здесь матрица $\Lambda = (a_{st})_{(m+1)\times(m+1)}$ составлена из коэффициентов многочлена $f(x, y)$ и является симметричной.

Абонент A имеет секретный набор $(a_0^{(i)}, a_1^{(i)}, \ldots, a_m^{(i)})$, состоящий из коэффициентов многочлена

$$g_i(x) = f(x, r_i) = a_0^{(i)} + a_1^{(i)} x + \quad + a_m^{(i)} x^m,$$

или иначе

$$\left(a_0^{(i)}, a_1^{(i)}, a_2^{(i)}, \ldots, a_m^{(i)}\right) = (1, r_i, r_i^2, \ldots, r_i^m) \cdot \Lambda.$$

Предположим, что известны ключевые материалы, принадлежащие m абонентам. Покажем, что по ним нельзя получить никакой информации ни об одном из ключей для связи между оставшимися $(n - m)$ абонентами. Для этого можно воспользоваться теоремой 11, но проще провести прямое доказательство.

Пусть, например, известны секретные наборы ключевых материалов m абонентов: $A_1, A_2, \ldots, A_m$. Будем искать ключ для связи между абонентами A_i, A_j; $m + 1 \le i, j \le n$. Для этого рассмотрим матричное равенство

$$\begin{pmatrix} k_{11} & k_{12} & & k_{1m} & k_{1j} \\ k_{21} & k_{22} & & k_{2m} & k_{2j} \\ & & & & \\ k_{m1} & k_{m2} & & k_{mm} & k_{mj} \\ k_{i1} & k_{i2} & & k_{im} & k_{ij} \end{pmatrix} =$$

$$= \begin{pmatrix} 1 & r_1 & r_1^2 & & r_1^m \\ 1 & r_2 & r_2^2 & & r_2^m \\ & & \ldots & & \\ 1 & r_m & r_m^2 & & r_m^m \\ 1 & r_i & r_i^2 & & r_i^m \end{pmatrix} \Lambda \begin{pmatrix} 1 & 1 & & 1 & 1 \\ r_1 & r_2 & & r_m & r_j \\ r_1^2 & r_2^2 & & r_m^2 & r_j^2 \\ & & & & \\ r_1^m & r_2^m & & r_m^m & r_j^m \end{pmatrix}$$

В матрице левой части равенства неизвестен только искомый ключ k_{ij}. В правой части неизвестная матрица Λ умножается слева и справа на обратимые матрицы, так как

по лемме их определители не равны нулю. Выражая теперь матрицу Λ из данного равенства, замечаем, что при произвольно заданном значении ключа k_{ij} из поля F матрицу Λ находим как произведение трех матриц. Поскольку данное матричное уравнение имеет в качестве решения искомую матрицу Λ, а решение однозначно, в качестве решения получаем матрицу коэффициентов многочлена $f(x, y)$.

Таким образом, при каждом значении ключа k_{ij} может быть найден многочлен $f(x, y)$, при котором данный ключ получается в качестве решения. Это свидетельствует о невозможности получения информации об этом ключе на основании известных значений. В то же время если известны ключевые материалы, принадлежащие $m + 1$ абонентам, то, очевидно, из рассматриваемого равенства матрица $\Lambda = (a_{st})$ всегда находится однозначно.

Теорема доказана.

Следствие 2. Для заданного числа m $(1 \leq m \leq n - 1)$ схема Блома является оптимальной — она имеет минимальное по объему количество хранимых у абонентов ключевых материалов.

Схема на основе пересечения множеств

Рассмотрим еще одну схему предварительного распределения ключей, которая также позволяет значительно сократить общее число хранимых и передаваемых секретных ключей. Она называется KDP (Key Distribution Patterns) — схемой на основе шаблонов, или схемой на основе пересечений множеств [45].

Схема KDP(n, q). Пусть имеется n $(n > 2)$ абонентов (пользователей) и множество секретных ключей K. Составим из ключей этого множества упорядоченное множество q различных ключей и обозначим S — множество номеров выбранных ключей: $|S| = q$. Для простоты будем считать, что $S = \{1, 2, \ldots, q\}$. Выберем некоторое семейство $\{S_1, \ldots, S_n\}$ подмножеств множества S. Предварительно абоненту i при личной встрече или по защищенному каналу передается множество секретных ключей с номерами подмножества S_i $(i = \overline{1, n})$. Таким образом, семейство $\{S_1, \ldots, S_n\}$ представляет собой таблицу с номерами ключей, хранящихся у каждого пользователя. Хотя данная таблица является несекретной, она должна быть защищена от модификаций и подделок.

Если абонент i хочет связаться с абонентом j, то он использует для выработки общего ключа множество ключей, номера которых содержатся в пересечении $S_i \cap S_j$. Если каждый ключ представлен некоторой битовой строкой, то для формирования общего связного ключа можно взять, например, их сумму или значение некоторой хеш-функции от строки, составленной из ключей, номера которых входят в пересечение множеств $S_i \cap S_j$. (Покажите, что в случае суммы получается слабая схема распределения ключей, так как возможны линейные соотношения между значениями связных ключей у различных пар абонентов.)

Схемой распределения ключей типа KDP (или *KDP*(n, q)-*схемой*) назовем всякое семейство $\{S_1, \ldots, S_n\}$ подмножеств множества K, удовлетворяющее следующему условию: если при некоторых $i, j, r \in \{1, \ldots, n\}$ выполнено включение $S_i \cap S_j \subseteq S_r$, то либо $i = r$, либо $j = r$.

Это условие означает, что общий ключ двух абонентов не должен быть известным никакому другому абоненту.

Семейство Шпернера. Семейство подмножеств называется *семейством Шпернера*, если ни одно из них не содержится в другом.

Теорема 13. Если подмножества $\{S_1, \ldots, S_m\}$ множества S ($|S| = q$) образуют семейство Шпернера, то

$$m \leq \binom{q}{\lfloor \frac{q}{2} \rfloor}.$$

Равенство достигается только в случае, если множество $\{S_1, \ldots, S_m\}$ совпадает с множеством всех w-элементных подмножеств множества K, где $w = q/2$ при четном q и $w = (q+1)/2$ или $(q-1)/2$ при нечетном q (E. Sperner, 1928).

Доказательство. Пусть семейство $\{S_1, \ldots, S_m\}$ подмножеств множества K состоит из максимально возможного числа элементов. Покажем, что существует семейство Шпернера с не меньшим, чем m, числом множеств, каждое их которых имеет мощность $\left\lfloor \frac{q}{2} \right\rfloor$.

Обозначим $w = \max\limits_{i=\overline{1,n}} |S_i|$. Предположим, что $w > \left\lfloor \frac{q}{2} \right\rfloor$ и что подмножества $S_1, \quad , S_t$ ($t \leq n$) состоят из w элементов. Рассмотрим подмножества $R_1, \ldots, R_u$ множества K, каждое из которых состоит из $w - 1$ элементов и содержится хотя бы в одном из подмножеств $S_1, \ldots, S_t$. Нетрудно ви-

деть, что семейство $R_1, \ldots, R_u, S_{t+1}, \ldots, S_m$ также образует семейство Шпернера. Покажем, что $u \geq t$.

Каждое подмножество S_i содержит ровно w подмножеств R_j. В то же время каждое подмножество R_j входит не более чем в $(q - w + 1)$ подмножеств S_i. Поэтому, вычисляя двумя способами число ребер в двудольном графе, вершинами которого являются подмножества $S_1, \ldots, S_t$ и $R_1, \ldots, R_u$, а ребра соединяют вложенные подмножества $R_j \subseteq S_i$, получаем неравенство $wt \leq (q - w + 1)u$. Отсюда следует, что

$$t\left(\left\lfloor \frac{q}{2} \right\rfloor + 1\right) \leq wt \leq (q - w + 1)u \leq \left(q - \left\lfloor \frac{q}{2} \right\rfloor\right) u$$

или

$$u \geq \frac{\left\lfloor \frac{q}{2} \right\rfloor + 1}{q - \left\lfloor \frac{q}{2} \right\rfloor} t \geq t.$$

Таким образом, можно считать, что в семействе $\{S_1, S_n\}$ все элементы состоят не более чем из $\left\lfloor \frac{q}{2} \right\rfloor$ элементов.

Если в семействе $\{S_1, \ldots, S_n\}$ имеется множество с меньшим, чем $\left\lfloor \frac{q}{2} \right\rfloor$, числом элементов, то, переходя к семейству $\{S'_1, \ldots, S'_n\}$, состоящему из подмножеств $S'_i = K \backslash S_i$, которое также составляет семейство Шпернера, и, повторяя приведенные выше рассуждения, получаем семейство, в котором все множества имеют одинаковое число элементов.

Теорема доказана.

Связь между KDP(n, q)-схемами и семействами Шпернера. Рассмотрим теоремы 14, 15.

Теорема 14*. Семейство $\{S_1, \ldots, S_n\}$ подмножеств множества S $(|S| = q)$ образует KDP(n, q)-схему в том и только в том случае, если множество $\{S_i \cap S_j \quad 1 \leq i < j \leq n\}$ образует семейство Шпернера.

Теорема 15. Для любой KDP(n, q)-схемы каждый абонент должен иметь не менее $\log_2 n$ ключей. Если $n \geq 4$, то $q \geq 2 \log_2 n$.

Доказательство. Если у какого-либо абонента имеется множество, состоящее менее чем из $\log_2 n$ ключей, то с их помощью можно построить не более чем $(n - 2)$ раз-

*Докажите теорему 14 самостоятельно в качестве упражнения.

личных подмножеств ключей. Поэтому он не сможет сформировать различные ключи для связи с $n-1$ абонентами.

Вторая оценка получается из очевидного неравенства

$$\binom{q}{\lfloor q/2 \rfloor} \geq \binom{n}{2},$$

которое справедливо в силу теоремы 14.

Пример 9.1. Пусть $n = q$, $|S_i| = n - 1$ $(1 \leq i \leq n)$, тогда семейство множеств $S_i \cap S_j$ $(1 \leq i < j \leq n)$ образует семейство всех подмножеств мощности $n-2$ и поэтому является семейством Шпернера. По теореме 14 получаем $\mathrm{KDP}(n, n)$-схему.

Еще один пример из комбинаторики дают так называемые блок-схемы. *Структурой инцидентности* называется тройка (P, B, I), состоящая из множества P точек, множества B блоков и бинарного отношения $I \subseteq P \times B$. *Блок-схемой*, или 2-(v, k, λ)-*схемой* $(\lambda \geq 1)$, называют структуру инцидентности, имеющую v точек и b блоков, каждый из которых инцидентен r точкам, в которой каждая точка инцидентна k блокам, а каждая пара точек инцидентна λ блокам. Можно показать, что в этом случае выполняются соотношения $bk = vr$, $r(k-1) = (v-1)\lambda$.

Схема распределения ключей с полной матрицей, в которой каждая пара участников имеет один общий ключ, является 2-$(v, 2, 1)$-схемой при $v = n$, $b = n(n-1)/2$.

Менее тривиальный пример получаем для симметричной $(b = v)$ 2-$(v, k, 2)$-схемы. Такие блок-схемы называют *биплоскостями*. Можно показать, что в этом случае $r = k$, $b = (r^2 - r + 2)/2$. Поставим ей в соответствие $\mathrm{KDP}(n, n)$-схему следующим образом: пусть $n = b = v$, составим множество S_i из номеров блоков, инцидентных точке с номером i, $|S_i| = k$, $1 \leq i \leq n$, тогда все пересечения $S_i \cap S_j$ $(1 \leq i < j \leq n)$ будут состоять из двух точек и образовывать семейство Шпернера.

m-KDP(n, q)-схема. Рассмотрим теперь вопрос об устойчивости $\mathrm{KDP}(n, q)$-схемы к компрометации ключей. Легко видеть, что $\mathrm{KDP}(n, q)$-схема будет стойкой к m-кратной компрометации ключей в том и только в том случае, если выполняется следующее свойство: для любого подмножества абонентов $W \subset \{1, \ldots, n\}$, $|W| = m$ и любых $i, j \in \{1, \ldots, n\} \setminus W$ не должно выполняться включение

$$S_i \cap S_j \subseteq \bigcup_{t \in W} S_t.$$

В этом случае будем говорить, что это m-$\mathrm{KDP}(n, q)$-схема.

Теорема 16. $\mathrm{KDP}(n, q)$-схема является m-$\mathrm{KDP}(n, q)$-схемой в том и только в том случае, если множество всех объ-

единений из m различных пересечений $S_i \cap S_j$ $(1 \le i < j \le n)$ образует семейство Шпернера.

Доказательство. Рассмотрим множество из $\binom{n}{2} = M$ всех пересечений $S_i \cap S_j$. Покажем, что множество всех объединений из m таких пересечений образует семейство Шпернера. Если бы для некоторых различных объединений выполнялось включение

$$(A_1 \cap B_1) \cup \ldots \cup (A_m \cap B_m) \subseteq (C_1 \cap D_1) \cup \ldots \cup (C_m \cap D_m),$$

то это противоречило бы предыдущему условию. Действительно, предположим, что пара $\{A_i, B_i\}$ не совпадает ни с одной парой $\{C_j, D_j\}$ в правой части. Тогда, заменяя каждое пересечение $C_j \cap D_j$ на C_j либо на D_j, в зависимости от того, какое из этих множеств не совпадает ни с A_i ни с B_i, получаем включение вида $A_i \cap B_i \subseteq C_1 \cup \quad \cup C_m$, что противоречит условию.

Теорема 17. Для числа ключей любой m-KDP(n, q)-схемы справедлива оценка $q \ge m(\log_2 n - \log_2 m - 1)$.

Доказательство. Пусть A, B — множества номеров ключей двух абонентов, устанавливающих между собой связь; пусть $C_1, C_2, \ldots, C_w$ — множества номеров ключей договаривающихся между собой злоумышленников. Тогда для обеспечения скрытности передачи не должно выполняться условие $A \cap B \subseteq C_1 \cup C_2 \cup \quad \cup C_w$.

В силу теоремы 16 и оценки Шпернера должно выполняться неравенство

$$\binom{q}{\lfloor q/2 \rfloor} \ge \binom{n}{m},$$

из которого и получается требуемая оценка.

(g, m)-KDP(n, q)-схемы. Пусть $g \ge 2$. Под схемой предварительного распределения ключей для групп, состоящих не более чем из g участников, понимают два алгоритма: первый определяет значения распределяемых между n участниками наборов данных, которые будем называть ключевыми материалами, второй позволяет каждой группе, состоящей не более чем из g участников, вычислить значение ключа для организации закрытого сеанса.

Пусть $m \ge 1$. Схема предварительного распределения ключей для групп участников называется стойкой к m-кратной компрометации ключей, если любая группа, состоящая не более чем

из m участников, объединив свои ключевые материалы, не сможет определить ключи, применяемые группами из оставшихся участников.

Определение 6. Стойкая к m-кратной компрометации ключей схема распределения ключей на основе шаблонов для групп из g участников (g,m)-KDP(n,q) определяется набором подмножеств $\{S_1, \ldots, S_n\}$ множества $\{1, \ldots, q\}$, удовлетворяющим условию: если $i_1, \ldots, i_g, p_1, \ldots, p_m \in \{1, \ldots, n\}$ и выполнено включение

$$\bigcap_{j=1}^{g} S_{i_j} \subseteq \bigcup_{j=1}^{m} S_{p_j},$$

то $\{i_1, \ldots, i_g\} \cap \{p_1, \ldots, p_w\} \neq \varnothing$.

По сути данное определение совпадает с известным в комбинаторике понятием семейства без перекрытий.

Определение 7. Если X — множество из v элементов, $|X| = v$, $\mathcal{F}$ — множество его подмножеств (блоков), $|\mathcal{F}| = b$, то $(X, \mathcal{F})$ называют (i,j)-*семейством без перекрытий* (cover-free family) и обозначают (i,j)-CFF(v,b), если для любых блоков $B_1, \ldots, B_u \in \mathcal{F}$, $u \leq i$, и любых не совпадающих с ними блоков $A_1, \ldots, A_w \in \mathcal{F}$, $w \leq j$, выполняется условие

$$\bigcap_{k=1}^{u} B_k \not\subseteq \bigcup_{s=1}^{w} A_s.$$

Определим матрицу инцидентности системы множеств $(X, \mathcal{F})$ как (0,1)-матрицу размера $b \times v$, в которой столбцы соответствуют элементам множества X, строки — подмножествам из $\mathcal{F}$, причем единицы стоят на пересечении со столбцами, помеченными элементами подмножества, соответствующего строке.

Непосредственно из определения вытекает следующий далее критерий.

Лемма 7. Система множеств $(X, \mathcal{F})$ является семейством без перекрытий (i,j)-CFF(v,b) в том и только в том случае, если в ее матрице инцидентности для любых двух непересекающихся наборов, состоящих из i и j строк, найдется столбец, на пересечении которого с первым набором строк стоят единицы, а на пересечении со вторым — нули [84].

Следствие 3. Для параметров семейства (i,j)-CFF(v,b) выполняются неравенства $i+j \le b$ и $v \ge \min\{\binom{b}{i}, \binom{b}{j}\}$ [84].

Минимальным семейством без перекрытий для заданных значений i и j будет семейство (i,j)-CFF(v,b) при $i+j=b$, $v=\min\{\binom{b}{i}, \binom{b}{j}\}$, в матрице инцидентности которого каждый столбец имеет ровно i единиц. В данном случае $v \ge b$.

Рассмотрим рекурсивный способ построения таких семейств. Для частного случая он опубликован в работе [84].

Лемма 8. Пусть $i \le 1$, $j \le 1$. В таблице ортогонального массива $OA(n, ij+1, 1)$ для любых двух непересекающихся наборов, состоящих из i и j строк, найдется столбец, на пересечении с которым ни один из элементов, стоящих в строках из первого набора, не совпадает ни с одним элементов, стоящих в строках из второго набора.

Доказательство. Рассмотрим два произвольных непересекающихся набора из i и j строк. Выделим элементы, стоящие на пересечении всех столбцов с первой строкой из первого набора и всеми строками из второго набора. Из свойства ортогональности следует, что совпадающие элементы должны стоять в разных строках, иначе найдутся два столбца с совпадающими парами элементов. Отсюда следует, что максимальное число столбцов с такими совпадениями равно j. Теперь в оставшихся столбцах элементы первой строки из первого набора не совпадают ни с одним из элементов строк второго набора. Удаляем столбцы, в которых произошло совпадение элементов, и рассматриваем ортогональный массив из оставшихся столбцов. Повторяя рассуждения для оставшихся $i-1$ строк из первого набора, получаем, что совпадения элементов, стоящих на пересечении со строками из первого набора, с элементами из строк второго набора могут быть не более чем в ij столбцах. Теперь в оставшихся неудаленными столбцах элементы, стоящие в строках из первого набора, не совпадают с элементами из строк второго набора.

Утверждение 6. Пусть $i \ge 2$, $j \ge 1$. Если существуют семейство (i,j)-CFF(v,b) и ортогональный массив $OA(b, ij+1, 1)$, то существует и семейство (i,j)-CFF$((ij+1)v, b^2)$.

Доказательство. Рассмотрим матрицу инцидентности A семейства (i,j)-CFF(v,b). Построим (0,1)-матри-

цу B размера $b^2 \times (ij+1)v$ путем замены каждого элемента a $(1 \le a \le v)$ в матрице ортогонального массива $OA(b, ij+1, 1)$ на строку матрицы A с номером a. В силу леммы 8 для любых двух непересекающихся наборов, состоящих из i и j строк ортогонального массива, найдется столбец, на пересечении с которым ни один из элементов, стоящих в строках из первого набора, не совпадает ни с одним из элементов, стоящих в строках из второго набора. Поэтому после замены всех элементов этого столбца на соответствующие строки матрицы A в построенной матрице B в силу леммы 2 найдется столбец, в котором на пересечении со строками из первого набора стоят единицы, а на пересечении со строками из второго набора — нули. Поэтому по лемме 2 матрица B является матрицей инцидентности семейства (i,j)-CFF$((ij+1)v, b^2)$.

Следствие 4. Если существуют семейство (i,j)-CFF(v_0, b_0) и ортогональные массивы $OA(b_0^{2^t}, ij+1, 1)$, $t = 1, 2, \ldots$, то для любого натурального t существует семейство (i,j)-CFF(v,b) с параметрами $b = b_0^{2^t}$ и $v = (ij+1)^t v_0$, удовлетворяющими соотношению

$$v = v_0 (\log_{b_0} b)^{\log_2(ij+1)}.$$

Доказательство. В результате применения утверждения t раз строим семейство (i,j)-CFF(v,b) при $b = b_0^{2^t}$ и $v = (ij+1)^t v_0$. Выражая из этих равенств t и приравнивая выражения, получаем

$$\ln v = \log_2(ij+1) \ln \log_{b_0} b + \ln v_0,$$

откуда и вытекает требуемое равенство.

Например, так как ортогональные массивы $OA(p^m, p^m + 1, 1)$ легко строят на основе поля из p^m элементов, то можно построить семейства без перекрытий при любых i, j с условием $i + j \le p^m$.

Как следствие из доказанного утверждения получаем теорему 18.

Теорема 18. Пусть $g \ge 2$, $m \ge 1$. Если существует (g,m)-KDP(n,q)-схема и ортогональный массив $OA(n, gm+1, 1)$, то существует и (g,m)-KDP$(n^2, (gm+1)q)$-схема.

Используя данный подход, можно строить различные схемы предварительного распределения ключей. Например, при $g =$

$= m = 2$ можно в зависимости от условий применять следующие схемы

n.....	256	625	2 401	4 096	65 536
k.....	150	250	525	700	750

9.2. Групповые протоколы

Групповые протоколы распределения ключей предназначены для распределения ключей между группами участников. При этом каждый участник получает определенный набор ключевых материалов, позволяющих ему вместе с остальными членами группы вычислить общее значение ключа для данной группы участников.

Рассмотрим два таких протокола: схемы разделения секрета и протоколы вычисления ключа для закрытой телеконференции. В первом протоколе проводится распределение одного и того же ключа между различными группами участников, во втором — различные группы получают разные ключи.

Схемы разделения секрета

Схема разделения секрета представляет собой схему предварительного распределения ключей между уполномоченными группами участников, в которой ключ заранее определен и одинаков для каждой уполномоченной группы. При этом каждый участник получает свою долю или «часть секрета». Схема включает два протокола: протокол формирования долей (разделения секрета) и распределения их между участниками и протокол восстановления секрета группой участников. Схема должна удовлетворять двум условиям:

1) любая группа пользователей, которые имеют на это полномочия, может вычислить ключ;

2) никакая другая группа не должна иметь возможность восстановить ключ или получить о нем какую-либо информацию.

Основное назначение схемы разделения секрета — защита ключа от потери. Обычно для защиты от потери делают несколько копий ключа. С ростом числа копий ключа возрастает вероятность его компрометации. Если число копий мало, то велик риск потери ключа. Поэтому лучше «разделить» ключ между несколькими лицами так, чтобы допускалась возможность восстановления ключа при различных обстоятельствах несколь-

кими уполномоченными группами с заранее оговоренным составом участников. Тем самым исключается риск безвозвратной потери ключа.

Еще одно положительное качество схем разделения секрета заключается в разделении ответственности за принятие решения, которое будет принято автоматически после того, как все члены уполномоченной группы предъявят свои ключевые материалы. Такая коллективная ответственность нужна для многих приложений, включая принятие важных решений, касающихся применения систем оружия, подписания корпоративных чеков или допуска к банковскому хранилищу.

Случай единственной группы. В простейшем случае, когда имеется только одна группа, состоящая из t участников, уполномоченная формировать ключ, схему разделения секрета можно построить следующим образом. Предположим, например, что ключ представляет собой двоичный вектор k длиной m. Выберем случайным образом t векторов $s_1, \ldots, s_t$ так, чтобы

$$k = s_1 + \quad + s_t,$$

и распределим их в качестве долей между участниками. Теперь, собравшись вместе, они могут легко восстановить значение ключа k, в то время как никакая группа, состоящая из меньшего числа участников, не сможет этого сделать. Действительно, в данном случае отсутствие хотя бы одной доли приводит к полной неопределенности относительно значения секрета, поскольку для каждого значения искомого секрета найдется возможный вариант значения отсутствующей доли.

Заметим, что если бы мы в предыдущем примере просто разбили вектор на t частей, то такая схема не могла быть схемой разделения секрета, так как знание любой доли давало бы частичную информацию о ключе k.

Пороговая схема. Пусть $1 < t \leq n$. Схема разделения секрета между n участниками называется (n, t)-*пороговой*, если любая группа из t участников может восстановить ключ, в то время как никакая группа из меньшего числа участников не может получить никакой информации о ключе.

Для построения (n, t)-пороговой схемы А. Шамир (A. Shamir) [78] предложил использовать многочлен степени $t - 1$ над конечным полем с достаточно большим числом элементов. Как известно, многочлен степени $t - 1$ можно однозначно восстановить по его значениям в t различных точках, но при этом меньшее число точек использовать для интерполяции нельзя.

Пусть F — конечное поле с достаточно большим числом элементов; $r_1, \ldots, r_n$ различные элементы F, отличные от нуля. Каждый элемент r_i припишем i-му участнику ($i = \overline{1,n}$) и разместим их в общедоступном месте. Выберем также t случайных элементов $a_0, \ldots, a_{t-1}$ поля F и рассмотрим многочлен $f(x)$ над полем F степени $t-1$ ($1 < t \leq n$):

$$f(x) = \sum_{i=0}^{t-1} a_i x^i$$

Данные элементы являются секретными и известны только центру распределения долей. Ключом будет значение $k = f(0) = a_0$. Для разделения секрета центр вычисляет значения долей

$$s_1 = f(r_1), \ldots, s_n = f(r_n)$$

и распределяет их между участниками.

Таким образом, каждому участнику соответствует набор

$$(r_i, s_i), \quad i = \overline{1,n}.$$

Покажем, что выполняются условия (n,t)-пороговой схемы разделения секрета.

1. Любая группа из t участников может легко вычислить ключ. Пусть, например, известны доли первых t участников. Для восстановления ключа k можно воспользоваться интерполяционной формулой Лагранжа:

$$f(x) = \sum_{i=1}^{t} s_i \prod_{j \neq i} \frac{x - r_j}{r_i - r_j}.$$

Поскольку $k = f(0)$, то, подставляя значение $x = 0$, получаем соотношения

$$k = \sum_{i=1}^{t} s_i c_i, \quad c_i = \prod_{j \neq i} \frac{r_j}{r_j - r_i} \qquad i = \overline{1,n},$$

причем коэффициенты c_i не зависят от коэффициентов многочлена $f(x)$ и могут быть вычислены заранее.

2. Докажем, что никакая группа из меньшего числа участников не может получить никакой информации о секрете. Предположим, что первые $t-1$ участников захотят найти ключ. Покажем, что в качестве значения ключа $k = f(0)$ может появиться любой элемент поля F Проведем интерполяцию многочлена по

точкам $r_1, \ldots, r_{t-1}, 0$ при $f(0) = y$. Так как при любом значении $y \in F$ можно однозначно найти коэффициенты многочлена, то любой элемент поля может появиться в качестве ключа k, и тем самым данная группа участников не сможет получить никакой информации о ключе без знания многочлена.

Схема Шамира удобна тем, что она позволяет легко увеличивать число участников. Для этого не нужно ничего менять, кроме множества $\{r_1, \quad , r_n\}$, к которому следует добавить новые элементы $r_{n+1}, \ldots, r_{n+w}$. Заметим, что компрометация одной доли делает из (n, t)-пороговой схемы $(n-1, t-1)$-пороговую схему.

Протоколы установления ключей для конференц-связи

Еще один тип распределения ключей между группами пользователей дают протоколы распределения ключей для проведения конференц-связи. Несмотря на внешнюю схожесть с протоколами разделения секрета, они имеют несколько принципиальных отличий. Если протоколы разделения секрета осуществляют предварительное распределение одного и того же ключевого значения (секрета) по секретным каналам между привилегированными группами пользователей, то протоколы конференц-связи осуществляют динамическое распределение ключей по открытым каналам связи между привилегированными группами пользователей. При этом ключи должны быть различными для каждой группы.

Пример (g, m)-KDP(n, q)-схемы предварительного распределения ключей был описан выше. Поэтому рассмотрим протоколы динамического распределения ключей.

Тривиальный пример распределения ключей для проведения конференц-связи дает использование централизованного распределения ключей с помощью одного из трехсторонних протоколов передачи ключей, используемых для симметричных систем шифрования. Для реализации такого подхода нужно выделить одного из пользователей группы и возложить на него функции центра генерации и распределения ключей. Естественно, что при этом возрастают требования к доверенности и безопасности выделенного пользователя, что вносит серьезную асимметрию между участниками конференц-связи.

Другой подход основан на использовании идеи открытого распределения ключей.

Приведем примеры протоколов, в которых все участники группы имеют одинаковые полномочия и выполняют симметричные функции.

Протокол DH с тремя участниками. Простейший пример такого протокола для группы из трех участников можно получить, слегка модифицировав протокол открытого распределения ключей Диффи—Хеллмана. Участники протокола заранее договариваются о значениях большого простого числа p и образующего элемента α мультипликативной группы $Z_p^* = \{1, 2, \ldots, p-1\}$. Для выработки общего ключа k пользователи A, B, C должны сгенерировать случайные числа x, y, z $(1 \le x, y, z \le p-2)$ соответственно. Затем они должны обменяться сообщениями согласно следующему протоколу:

$$
\begin{aligned}
&A \to B : X = \alpha^x \bmod p,\\
&B \to C \quad Y = \alpha^y \bmod p,\\
&C \to A \quad Z = \alpha^z \bmod p,\\
&A \to B \quad Z' = Z^x \bmod p,\\
&B \to C \quad X' = X^y \bmod p,\\
&C \to A \quad Y' = Y^z \bmod p.
\end{aligned}
$$

Искомый общий ключ $k = \alpha^{xyz} \bmod p$ теперь вычисляется пользователями A, B, C по формулам

$$
\begin{aligned}
k &= \left(Y'\right)^x \bmod p,\\
k &= \left(Z'\right)^y \bmod p,\\
k &= \left(X'\right)^z \bmod p
\end{aligned}
$$

соответственно.

Можно обобщить эту схему на t участников, при этом в сети будет передано $t(t-1)$ сообщение.

Нетрудно видеть, что вмешательство нарушителя по типу «противник в середине» приведет к тому, что все участники получат в результате различные значения ключа. В то же время данный протокол не является аутентифицированным, так как он не позволяет обнаружить подмену, если в качестве одного из участников и вместо него выступит нарушитель.

Протокол Бурместера—Десмедта. Рассмотрим теперь протокол Бурместера—Десмедта (M. Burmester, Y. Desmedt) — протокол формирования общего ключа для конференц-связи группы из t участников $U_0, \ldots, U_{t-1}$ [35]. Как и в предыдущем

протоколе, каждый участник U_i должен сгенерировать секретное случайное число r_i $(1 \le r_i \le p-2)$ и вычислить открытую экспоненту $\beta_i = \alpha^{r_i} \bmod p$.

Обозначим

$$A_i = \alpha^{r_i r_{i+1}} = \beta_i^{r_{i+1}} = \beta_{i+1}^{r_i}.$$

Тогда общий ключ k будет иметь вид

$$k = \alpha^{r_0 r_1 + r_1 r_2 + \ldots + r_{t-1} r_0} \bmod p = A_0 A_1 \ldots A_{t-1} \bmod p.$$

Протокол состоит из следующих шагов:

1) каждый участник U_i рассылает β_i остальным $t-1$ пользователям;

2) каждый участник U_i вычисляет значение

$$X_i = (\beta_{i+1}/\beta_{i-1})^{r_i} \bmod p$$

и рассылает его остальным $t-1$ пользователям;

3) каждый участник U_i вычисляет значение общего ключа k по формуле

$$k_i = \beta_{i-1}^{t r_i} X_i^{t-1} X_{i+1}^{t-2} \ldots X_{i+t-3}^{2} X_{i+t-2}^{1} \bmod p.$$

Покажем, что данное значение является искомым ключом. В самом деле:

$$\begin{gathered} A_{i-1}^t X_i^{t-1} X_{i+1}^{t-2} \ldots X_{i+t-3}^2 X_{i+t-2}^1 = \\ = A_{i-1}(A_{i-1}X_i)(A_{i-1}X_iX_{i+1}) \ldots (A_{i-1}X_iX_{i+1}X_{i+t-2}) = \\ = A_{i-1}A_iA_{i+1} \ldots A_{i-2} = A_0A_1A_2 \ldots A_{t-1}, \end{gathered}$$

поэтому $k_i = k$.

Протокол требует передачи $2t(t-1)$ сообщений, причем каждый участник должен отправлять сообщения всем остальным. Можно модифицировать протокол для случая обмена сообщениями по схеме двунаправленного кольца.

Рассмотренный протокол не обеспечивает аутентификацию сторон, поскольку в нем не заложены процедуры для взаимной аутентификации.

Контрольные задания

1. Убедитесь, что схема Блома является оптимальной.

2. Почему ни одно из множеств $S_1, \ldots, S_n$ $\mathrm{KDP}(n, q)$-схемы не может состоять из двух элементов?

3. Пусть имеется KDP(n, q)-схема с матрицей инцидентности A. Обозначим строки матрицы A символами $a_1, \ldots a_n$. Покажите, что:

а) матрица A' размера $n^2 \times 3q$, составленная из строк вида (a_i, a_j, a_{i+j}), где $i, j \in \{1, \ldots, n\}$ и сумма индексов рассматривается по модулю n, является матрицей инцидентности KDP$(n^2, 3q)$-схемы;

б) если к матрице инцидентности A' из предыдущего упражнения добавить три строки вида

$$\begin{array}{ccccccccc} 0 & & 0 & 1 & & 1 & 1 & & 1 \\ 1 & & 1 & 0 & & 0 & 1 & & 1 \\ 1 & & 1 & 1 & & 1 & 0 & & 0 \end{array}$$

то получится KDP$(n^2 + 3, 3q)$-схема.

4. Объясните с использованием условия задачи 3, как можно построить KDP(n, q)-схемы с параметрами:

n	q
147...	27
364...	36
787...	45
21 612...	81
132 499...	108
619 372...	135
467 078 547...	243
17 555 985 004...	324
383 621 674 387...	405

5. $(n, k; \lambda)$-Разностной матрицей называется матрица (d_{st}) над кольцом вычетов $\mathbf{Z}_n$ размера $k \times n\lambda$, в которой при всех x, y, $1 \leq x < y \leq k$, в мультимножестве

$$\{d_{xz} - d_{yz} \bmod n \quad 1 \leq z \leq n\lambda\}$$

каждый из элементов $\mathbf{Z}_n$ встречается ровно λ раз.

Докажите, что если выполнено условие $(n, (k-1)!) = 1$, то матрица $D = (d_{uz})$ при $d_{xz} = uz \bmod n$, $1 \leq x \leq n$, $1 \leq z \leq k$ будет $(n, k; 1)$-разностной матрицей.

Докажите, что по уже построенной $(n, k; 1)$-разностной матрице $D = (d_{xz})$ можно получить ортогональный массив $OA(n, k, 1)$ с матрицей $B = (b_{(x,y),z})$ размера $n^2 \times k$, полагая при $1 \leq x, y \leq n$ и $1 \leq z \leq k$:

$$b_{(x,y),z} = d_{xz} + y \bmod n.$$

6. Структура инцидентности называется t-(v, k, λ)-схемой $(1 \leq t \leq k)$, если все блоки инцидентны k точкам и каждые t точек инцидентны λ блокам.

Докажите следующие свойства:

а) всякая t-(v, k, λ_t)-схема является $(t-1)$-(v, k, λ_{t-1})-схемой, причем $\lambda_t(v-t+1) = (k-t+1)\lambda_{t-1}$;

б) если для 3-(v, k, λ_3)-схемы выполнено условие $\lambda_2 > w\lambda_3$, то ей соответствует w-KDP(n, q)-схема;

в) всякой $(w+2)$-схеме соответствует w-KDP(n, q)-схема.

7. Что такое схема разделения секрета?

8. Предложите схему разделения секрета для двух групп из трех участников каждая, если:

- составы групп не пересекаются;
- имеется один участник, входящий в обе группы.

9. В чем общность и в чем отличия схемы разделения секрета и способов распределения ключей для протокола конференц-связи?

Аппаратные средства для защиты ключей в компьютере

10.1. Проблема защиты главного ключа

Рассмотрим проблему защиты ключевой информации на компьютере. Если криптографические средства реализованы программно, то ключи используются этими прикладными программами как обычные наборы данных. Поэтому потенциально их можно считать из памяти компьютера. Для исключения возможности этого необходимо сделать так, чтобы они появлялись в компьютере только в зашифрованном на других ключах виде. В этом случае все множество ключей можно хранить открыто на жестком диске в специальной базе данных. Вместе с тем для работы с такой базой необходимо иметь в открытом виде хотя бы один из ключей, на которых зашифрованы хранящиеся в базе ключи.

Выход из данной ситуации заключается в использовании специального программно-аппаратного средства, которое хранит значение главного ключа и выполняет все криптографические операции без обращения к основному процессору.

Рассмотрим вариант построения аппаратного устройства (исполненного в виде отдельной платы — криптоадаптера, устанавливаемого в компьютер) для защиты ключей в системах шифрования, реализующих криптографические алгоритмы, основанные на симметричных ключах, и ключевую систему на основе главного ключа. Данный вариант построения криптоадаптера реализован в рамках концепции TSS (Transaction Security System).

Система безопасности транзакций TSS, разработанная фирмой IBM, представляет собой набор аппаратных и программных средств, выполняющих определенные криптографические функции.

Данная система предназначена для обеспечения безопасности банковских рабочих мест с учетом сложной распределенной архитектуры информационной системы банка.

Для стандартизации основных криптографических функций, выполняемых аппаратными устройствами, и определения порядка их взаимодействия в рамках системы TSS разработана системная криптографическая архитектура SCA (System Cryptographic Architecture).

Архитектура SCA исходит из модели нарушителя, которым может быть лицо, являющееся техническим специалистом или обслуживающим персоналом банка. Поэтому злоумышленник обладает значительными возможностями по доступу к компьютеру и обрабатываемой информации. Целью нарушителя является овладение неизвестным ему главным ключом конкретного компьютера (терминала, хоста).

Основное требование к системе безопасности состоит в том, что принятые меры должны быть достаточны для противодействия угрозам со стороны одного лица, какими бы полномочиями это лицо не обладало.

Модель нарушителя. Будем исходить из самой неблагоприятной ситуации, когда нарушитель может вскрывать корпус компьютера и устанавливать программно-аппаратные закладки. В этом случае информация, подлежащая шифрованию, очевидно, будет ему доступна.

Определение. *Усиленную модель* нарушителя* (УМН) для случая локального несанкционированного доступа определим следующими условиями:

1) программное обеспечение и аппаратура ПЭВМ обладают рядом свойств:

- в аппаратуре ПЭВМ могут присутствовать аппаратные закладки;
- в программном обеспечении могут присутствовать программные закладки;
- программа начальной загрузки BIOS ПЭВМ аппаратно гарантированно не защищена от перезаписи;
- расширения BIOS ПЭВМ аппаратно не защищены от перезаписи;
- информация на диске не защищена методом стойкого шифрования;

---

*Понятно, что приведенная модель нарушителя описывает крайнюю ситуацию, в которой противник обладает почти максимальными возможностями для осуществления своих замыслов. Такая ситуация может возникнуть при неправильной системе организационно-технических мероприятий, либо если нарушителем является один из сотрудников обслуживающего технического персонала.

2) нарушителю известны:
   - тип используемого оборудования;
   - система защиты информации;

3) нарушитель обладает:
   - всеми исходными текстами программного обеспечения (включая средства защиты), работающего на данной ПЭВМ;
   - полной технологической документацией на атакуемый комплекс;
   - рабочим экземпляром комплекса;

4) нарушитель имеет возможность:
   - перехватывать, подменять и удалять передаваемые по телекоммуникациям данные;
   - имитировать терминалы;
   - получать доступ к включенной ПЭВМ на неограниченное время под контролем легальных пользователей;
   - получать доступ к включенной незаблокированной ПЭВМ на ограниченное время, достаточное для хищения всей обрабатываемой информации;
   - получать доступ к выключенной ПЭВМ на неограниченное время;
   - получать бесконтрольный доступ к включенной незаблокированной ПЭВМ на короткое время, достаточное для запуска одной программы со съемного накопителя;
   - получать кратковременный доступ к незаблокированной включенной ПЭВМ в любое удобное для себя время;
   - осуществлять атаку путем всевозможного сочетания функций вызова и команд, заложенных в криптографическом интерфейсе, вместе с подачей на вход произвольных наборов данных в целях получения доступа к ключам;
   - кратковременно вскрывать корпус ПЭВМ;

5) нарушитель не может:
   - нарушить целостность технических устройств, выполняющих криптографические функции;
   - преодолеть средства физической защиты криптоадаптера для считывания хранящегося там долговременного ключа;
   - быть «офицером безопасности» либо любым другим лицом, имеющим доступ по долгу службы к паролям или компонентам секретных ключей;

6) нарушитель не имеет ключевой информации.

10.2. Архитектура криптографической подсистемы

Архитектура криптоадаптера. Для реализации этого принципа средства, выполняющие криптографические функции, размещают в защищенной области специального аппаратного устройства — криптоадаптера. Основными функциями криптоадаптера, обозначаемого далее CF (Crypto Facility), являются:

- хранение главного ключа в физически защищенной памяти;
- выполнение криптографических функций по шифрованию и расшифрованию данных;
- шифрование и перешифрование ключей.

Главное требование к криптоадаптеру заключается в реализации *базового принципа*: ни один из ключей, используемых в системе, не должен появляться вне этого устройства в незашифрованном виде.

При этом обеспечивается защита от физического проникновения и попыток считывания памяти (через электрическое взаимодействие, рентгеновское излучение, электронное просвечивание, химическое воздействие и т. п.). В случае физического проникновения или попыток считывания памяти происходит автоматическое обнуление памяти. В наиболее ответственных случаях при вводе ключа применяют схемы разделения секрета.

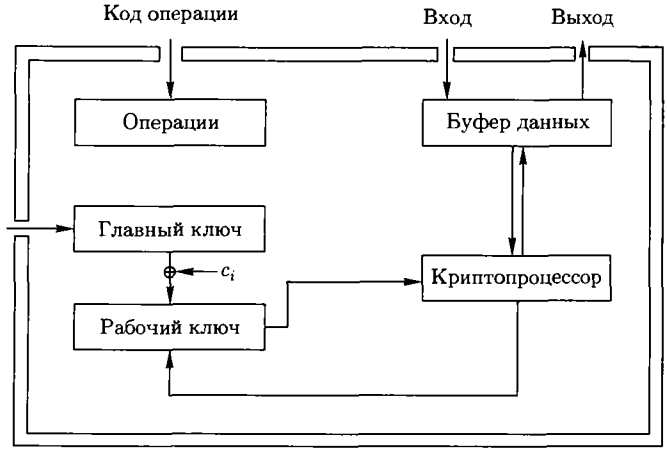

Рис. 10.1. Архитектура криптоадаптера

Для реализации устройства криптоадаптера (рис. 10.1) используется следующий подход:

- ввод в систему главного ключа осуществляется со специального считывающего устройства непосредственно в CF, минуя остальные устройства компьютера;
- генерацию и хранение ключей (в зашифрованном виде) можно осуществлять вне устройства CF; в самóм устройстве должен хранится только один главный ключ;
- в устройстве наряду с защищенной памятью должен быть реализован защищенный криптопроцессор, способный выполнять все необходимые криптографические операции;
- внутренние команды управления криптопроцессором должны быть недоступны извне; обращение к CF извне осуществляется только в соответствии со специальным интерфейсом, включающим ограниченный набор команд-инструкций;
- данный интерфейс должен включать и поддерживать только такие инструкции, при которых выполняется указанное выше главное требование к CF.

Интерфейс обращения к криптоадаптеру. Основной проблемой при создании такого устройства является разработка специального интерфейса для обращения к криптоадаптеру извне со стороны операционной системы и прикладных программ, выполняемых на компьютере. Необходимо выбрать набор команд-инструкций, описывающих этот интерфейс, таким образом, чтобы они, с одной стороны, позволяли выполнять все необходимые криптографические операции по шифрованию и расшифрованию различных массивов данных, а с другой стороны, позволяли осуществлять генерацию и перешифрование ключей с одного ключа на другой без раскрытия самих ключей.

Для этого создается специальное программное обеспечение криптографической подсистемы, состоящее из программ управления ключами и генерации ключей, а также из файла с базой данных, в которой хранятся зашифрованные ключи (рис. 10.2):

1) *программа управления ключами* (Key Manager — KM) предназначена для управления ключами: создания первичных ключей, извлечения хранимых ключей из открытой базы данных по имени и обращения к криптоадаптеру CF для выполнения операций прешифрования ключей;

2) *программа генерации ключей* (Key Generator — KG) предназначена для создания вторичных ключей;

3) *база данных*, в которой хранятся зашифрованные ключи (Cryptographic Key Data Set — CKDS), предназначена для хра-

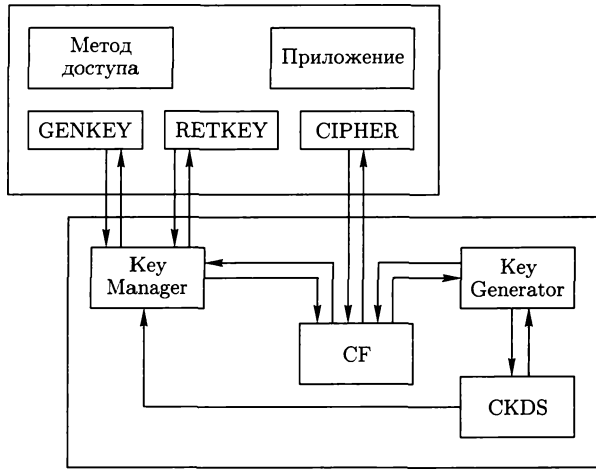

Рис. 10.2. Архитектура криптографической подсистемы

нения всех ключей, используемых на данном компьютере, за исключением главного и его вариантов; понятно, что копии файла с базой данных CKDS необходимо сохранять не только в системе, но и на сменных носителях.

На верхнем прикладном уровне интерфейс обращения программ приложений к криптографической подсистеме должен состоять из команд трех типов:

1) зашифрование/расшифрование ключей (CIPHER);
2) генерация ключей (GENKEY);
3) перешифрование ключей (RETKEY).

Они представляют собой обычные макроинструкции с некоторым списком параметров. Например, для команды зашифрование/расшифрование ключей нужно указать адреса открытой и зашифрованной информации, длину текста, адреса ключей, вид, режим и параметры алгоритма шифрования, способ дополнения текста (для блочного шифра) и т. п. Для команды генерации ключей следует указать адрес имени и адрес самого значения ключа, а для команды перешифрования ключей требуется указать, с какого ключа на какой нужно осуществить перешифрование.

Криптографическая подсистема преобразует эти команды в наборы кодов операций управления работой криптоадаптера (см. рис. 10.1).

Виды ключей. Заметим, что концепция главного ключа предполагает наличие в системе достаточно большого числа

ключей, которые предназначаются для различных целей. Помимо первичных ключей, которые могут использоваться в качестве сеансовых, коммуникационных, файловых ключей, в ней предусмотрены вторичные ключи, предназначенные для шифрования первичных ключей (вторичные коммуникационные, вторичные файловые и т. п.). Ключевая система предполагает хранение в открытом виде в защищенной области криптоадаптера только главного (матер-) ключа. Все остальные ключи хранятся в зашифрованном виде в базе данных CKDS, расположенной в незащищенных областях. Для этого используют ключи шифрования ключей (вторичные ключи). В открытом виде ключи могут присутствовать только внутри защищенной области и только в момент выполнения соответствующих криптографических функций.

Введем обозначения: k_{MH} — главный ключ (хост-) компьютера; k_{MT} — главный ключ терминала; k_{SC} — вторичный ключ для шифрования коммуникационных ключей; k_{SF} — вторичный ключ для шифрования файловых ключей; k_N — вторичный ключ; k — первичный ключ.

Примем следующие соглашения:

1) все первичные ключи в системе, предназначенные для выполнения операций шифрования и расшифрования каких-либо данных при внутреннем использовании, представляются в CF зашифрованными на главном ключе — $E_{k_{MH}}(k)$ (*внутреннее представление*); при этом генерация сеансовых ключей k изначально осуществляется в таком же виде: $r = E_{k_{MH}}(k)$;

2) все первичные ключи k хранятся в базе данных CKDS в незащищенной области компьютера и передаются между компьютерами в зашифрованном на соответствующем вторичном ключе виде: $E_{k_N}(k)$ (*внешнее представление*);

3) все вторичные ключи k_N хранятся в базе данных CKDS в незащищенной области компьютера в зашифрованном на так называемых вариантах главного ключа k_{MH1} и k_{MH2} виде $E_{k_{MHi}}(k_N)$. Вариант главного ключа k_{MHi} строится из главного ключа инверсией некоторых бит: $k_{MHi} = k_{MH} \oplus c_i$ $(i = 1, 2)$.

Операции управления работой криптоадаптера. Следуя [46, 65], рассмотрим основные виды инструкций, входящих в интерфейс обращения к криптоадаптеру; они имеют вид

$$\text{Код операции} : \{\text{Входные переменные}\} \to \{\text{Выход}\}.$$

Всего имеется шесть операций: две — для обработки данных и четыре — для работы с ключами.

1. Операция ECPH (encipher) имеет вид

$$\mathrm{ECPH}(E_{k_{MH}}(k), \mathrm{data}) = E_k(\mathrm{data}).$$

2. Операция DCPH (decipher) имеет вид

$$\mathrm{DCPH}(E_{k_{MH}}(k), E_k(\mathrm{data})) = \mathrm{data}.$$

3. Операция SMK (set-master-key) устанавливает главный ключ в рабочий регистр и имеет вид

$$\mathrm{SMK} = \{k_{MH}\}.$$

4. Операция EMK (encipher-under-master-key) имеет вид

$$\mathrm{EMK}(k) = E_{k_{MH}}(k).$$

Данная операция введена для преобразования во внутреннее представление ключей, введенных в криптоадаптер в явном виде (в случае если все ключи генерируются в системе, она не обязательна), а также для перешифрования базы данных с ключами в случае смены главного ключа. Поэтому ее свободное использование должно быть ограничено.

Ограничение 1. Одновременное с командой EMK наличие обратной команды DMK в том же устройстве недопустимо, так как ее выполнение приведет к появлению в открытом виде ключа k.

Ограничение 2. Операция EMK не может быть применена к главному ключу k_{MH} и к любому вторичному ключу k_N, иначе можно применить операцию DCPH для расшифрования любого первичного ключа:

$$\mathrm{DCPH}(E_{k_{MH}}(k_{MH}), E_{k_{MH}}(k)) = k,$$
$$\mathrm{DCPH}(E_{k_{MH}}(k_N), E_{k_N}(k)) = k.$$

В связи с тем, что все первичные ключи должны храниться в зашифрованном на вторичных ключах виде, необходимо предусмотреть еще две команды.

5. Операция RFMK (reencipher-from-master-key) имеет вид

$$\mathrm{RFMK}(E_{k_{MH1}}(k_N), E_{k_{MH}}(k)) = E_{k_N}(k).$$

6. Операция RTMK (reencipher-to-master-key) имеет вид

$$\mathrm{RTMK}(E_{k_{MH2}}(k_N), E_{k_N}(k)) = E_{k_{MH}}(k).$$

Ограничение 3. В этих командах нельзя допустить применения вторичных ключей, зашифрованных на главном ключе (см. ограничение 2), так как появление записей вида $E_{k_{MH}}(k_N)$ в системе дает возможность применить операцию DCPH и получить явно значение ключа:

$$\mathrm{DCPH}(E_{k_{MH}}(k_N), E_{k_N}(k)) = k.$$

Поэтому для шифрования вторичных ключей вместо главного ключа k_{MH} используют два производных от него варианта главного ключа k_{MH1} и k_{MH2}, которые получают из главного ключа инверсией некоторых бит. (Их не нужно хранить отдельно, так они быстро вычисляются.) Два ключа используют для того, чтобы осуществить *принцип разделения ключей*:

1) ключ k_{MH1} используют для шифрования при хранении тех вторичных ключей, которые применяют в макроинструкции RFMK для перешифрования с главного ключа на текущий вторичный ключ k_N, тем самым макроинструкция RFMK переводит зашифрованный ключ из разряда «внутренний», т. е. предназначенный для внутреннего использования, в разряд «внешний», т. е. предназначенный для внешнего использования или хранения;

2) ключ k_{MH2} используют для шифрования при хранении тех вторичных ключей, которые применяют в макроинструкции RTMK для перешифрования с текущего вторичного ключа k_N на главный ключ; тем самым макроинструкция RTMK переводит зашифрованный ключ из разряда «внешний» в разряд «внутренний», т. е. предназначенный для внутреннего использования при шифровании и расшифровании с помощью операций ECPH и DCPH.

10.3. Примеры протоколов распределения ключей

Перечисленного в подразд. 10.2 набора операций достаточно для организации закрытой передачи информации по сети, содержащей несколько компьютеров.

Рассмотрим простейший случай сети, в которой имеется несколько соединенных между собой хост-компьютеров, к каждому из которых подключено несколько терминалов. Будем также для простоты полагать, что терминалы не хранят информацию, а обрабатывают и обмениваются ею между собой. (Точно так же

описывается случай, когда на компьютере работают несколько процессов или программ приложений, использующих различные ключи доступа к своим данным.) Все рассматриваемые протоколы приведены в чистом виде, без использования механизмов аутентификации.

Протокол выработки общего сеансового ключа между терминалами, относящимися к одному хост-компьютеру. В простейшем случае имеется один хост-компьютер и несколько подключенных к нему терминалов, главные ключи всех терминалов могут храниться в зашифрованном виде на хост-компьютере, а передача текущих первичных ключей терминалам и обратно может осуществляться в зашифрованном на их главных ключах виде. Это можно сделать в связи с тем, что в терминалах, как правило, не хранятся никакие ключи, кроме главного ключа терминала. (В этом случае в криптоадаптере терминала достаточно реализовать только операции EMK, ECPH, DCPH.) На хост-компьютере для перешифрования сеансовых ключей используют макроинструкции RFMK, RTMK.

Протокол имеет следующий вид: хост H_i генерирует ключ $r = E_{k_{MH}}(k)$ и отправляет терминалам T_1, T_2 сообщения:

$$\begin{aligned} H_i \to T_1 \quad & E_{k_{MT1}}(k) = \mathrm{RFMK}(E_{k_{MH1i}}(k_{MT1}), r), \\ H_i \to T_2 \quad & E_{k_{MT2}}(k) = \mathrm{RFMK}(E_{k_{MH1i}}(k_{MT2}), r). \end{aligned}$$

Теперь терминалы T_1, T_2 могут начать сеанс на ключе k.

Протокол выработки общего сеансового ключа между терминалами, относящимися к разным хост-компьютерам. Пусть терминал T_1, связанный с хостом H_i, запрашивает у него ключ для связи с терминалом T_2, связанным с хостом H_j. Хост H_i генерирует сеансовый ключ k в виде $R = E_{k_{MH0i}}(k)$ и передает его терминалу T_1 и хосту H_j (здесь $k_{MH0i} = k_{MHi}$ — главный ключ хоста H_i; k_{MH1i}, k_{MH2i} — его варианты). Для передачи ключей между хостами H_i и H_j используют шифрование на специальных коммуникационных ключах k_{SCij} и k_{SCji}. Между хостами и подключенными к ним терминалами передача осуществляется в зашифрованном на главных ключах терминалов виде.

1) хост H_i генерирует ключ $r = E_{k_{MH0i}}(k)$ и отправляет сообщения:

$$\begin{aligned} H_i \to T_1 \quad & E_{k_{MT1}}(k) = \mathrm{RFMK}(E_{k_{MH1i}}(k_{MT1}), r), \\ H_i \to H_j \quad & E_{k_{SCij}}(k) = \mathrm{RFMK}(E_{k_{MH1i}}(k_{SCij}), r); \end{aligned}$$

2) хост H_j вычисляет $r' = \mathrm{RTMK}(E_{k_{MH2j}}(k_{SCij}), E_{k_{SCij}}(k))$ и отправляет сообщение:

$$H_j \to T_2 \quad E_{k_{MT2}}(k) = \mathrm{RFMK}(E_{k_{MH1j}}(k_{MT2}), r').$$

Теперь терминалы T_1, T_2 могут начать сеанс на ключе k.

Протокол передачи файла. Предположим, что нужно передать файл data, зашифрованный на первичном файловом ключе k, от хоста H_i к хосту H_j.

Если файл data новый, то протокол имеет следующий вид:

1) хост H_i генерирует ключ $r = E_{k_{MH0i}}(k)$, выполняет операции

$$E_k(\mathrm{data}) = \mathrm{ECPH}(r, \mathrm{data}),$$
$$E_{k_{SFij}}(k) = \mathrm{RFMK}(E_{k_{MH1i}}(k_{SFij}), r)$$

и отправляет сообщение:

$$H_i \to H_j \quad E_{k_{SFij}}(k), E_k(\mathrm{data});$$

2) хост H_j вычисляет

$$E_{k_{MH0j}}(k) = \mathrm{RTMK}(E_{k_{MH2j}}(k_{SFij}), E_{k_{SFij}}(k)),$$
$$\mathrm{data} = \mathrm{DCPH}(E_{k_{MH0j}}(k), E_k(\mathrm{data})).$$

Если же файл data уже хранится в зашифрованном виде $(E_{k_{SFii}}(k), E_k(\mathrm{data}))$, то протокол передачи этого файла отличается первым шагом:

1) хост H_i вычисляет

$$E_{k_{MH0i}}(k) = \mathrm{RTMK}(E_{k_{MH2i}}(k_{SFii}), E_{k_{SFii}}(k)),$$
$$E_{k_{SFij}}(k) = \mathrm{RFMK}(E_{k_{MH1i}}(k_{SFij}), E_{k_{MH0i}}(k))$$

и отправляет сообщение:

$$H_i \to H_j \quad E_{k_{SFij}}(k), E_k(\mathrm{data});$$

2) хост H_j вычисляет

$$E_{k_{MH0j}}(k) = \mathrm{RTMK}(E_{k_{MH2j}}(k_{SFij}), E_{k_{SFij}}(k)),$$
$$\mathrm{data} = \mathrm{DCPH}(E_{k_{MH0j}}(k), E_k(\mathrm{data})).$$

Замена старого главного ключа на новый. Можно показать, что приведенных в подразд. 10.2 операций достаточно для замены старого главного ключа на новый. В связи с этим

приведем еще одно важное ограничение, существенно влияющее на безопасность ключей. Дело в том, что для выполнения некоторых операций необходимо иметь значения главного ключа и производных от него вариантов главного ключа k_{MH1} и k_{MH2}, зашифрованные друг на друге:

$$E_{k_{MH}}(k_{MHi}), E_{k_{MHi}}(k_{MHj}), \ldots; \quad i,j = 1,2.$$

Например, для осуществления операции замены старого главного ключа k^*_{MH} на новый k_{MH} и проведения соответствующего перешифрования всех вторичных ключей k_N (k'_N) необходимо выполнить следующую последовательность операций:

$$\begin{gathered}
\mathrm{EMK}(k_{MH1}) = E_{k_{MH}}(k_{MH1}),\\
\mathrm{EMK}(k_{MH2}) = E_{k_{MH}}(k_{MH2}),\\
\mathrm{ECPH}(E_{k_{MH}}(k_{MH1}), k_{MHi}) = E_{k_{MH1}}(k_{MHi}), \quad i = 1,2,\\
\mathrm{ECPH}(E_{k_{MH}}(k_{MH2}), k^*_{MHi}) = E_{k_{MH2}}(k^*_{MHi}), \quad i = 1,2,\\
\mathrm{RTMK}(E_{k_{MH2}}(k^*_{MH1}), E_{k^*_{MH1}}(k_N)) = E_{k_{MH}}(k_N),\\
\mathrm{RTMK}(E_{k_{MH1}}(k_{MH1}), E_{k_{MH}}(k_N)) = E_{k_{MH1}}(k_N),\\
\mathrm{RTMK}(E_{k_{MH2}}(k^*_{MH2}), E_{k^*_{MH2}}(k'_N)) = E_{k_{MH}}(k'_N),\\
\mathrm{RTMK}(E_{k_{MH1}}(k_{MH2}), E_{k_{MH}}(k'_N)) = E_{k_{MH2}}(k'_N).
\end{gathered}$$

Замечание. Отсутствие ограничений на проведение операций шифрования на производных ключах k_{MH1}, k_{MH2} неизбежно приводит к возможности раскрытия ключей.

Действительно, пусть, например, нарушитель может зашифровать произвольное известное ему значение X на ключе k_{MH1}. Тогда он может выполнить следующие действия:

$$\begin{gathered}
\mathrm{RFMK}(E_{k_{MH1}}(X), E_{k_{MH}}(k)) = E_X(k),\\
\mathrm{EMK}(X) = E_{k_{MH}}(X),\\
\mathrm{DCPH}(E_{k_{MH}}(X), E_X(k)) = k.
\end{gathered}$$

В результате происходит раскрытие ключа k.

Ограничение 4. Необходимо ограничить возможность шифрования на производных вариантах главного ключа k_{MH1}, k_{MH2}. Значения $E_{k_{MHi}}(k_{MHj})$, $i,j = 1,2$ называют *ключами активации системы*; их вычисляют один раз при вводе главного ключа, а затем хранят в специальных регистрах памяти в криптоадаптере. В дальнейшем их используют как готовые значения, а все операции шифрования на ключах k_{MH1}, k_{MH2} блокируют.

Ограничение 5. Нельзя предоставлять свободный доступ к значениям из множеств $\{E_{k_{MH}}(k_{MHi}) \quad i = 1,2\}$ и

$$\{E_{k_{MHi}}(k_{MHj}) \quad i,j = 0,1,2;\ i = j \neq 2\},$$

где $k_{MH0} = k_{MH}$, так как они могут быть использованы в качестве аргументов для выполнения операций зашифрования/расшифрования на ключах k_{MHi} (см. замечание); $i = 0,1,2$.

Приведем примеры:

- с помощью ключа $E_{k_{MH}}(k_{MHj})$, $j = 0,1,2$ можно зашифровать/расшифровать на ключе k_{MHj} любой набор данных X, используя операции ECPH, DCPH:

$$\text{ECPH}(E_{k_{MH}}(k_{MHj}), X) = E_{k_{MHj}}(X),$$
$$\text{DCPH}(E_{k_{MH}}(k_{MHj}), X) = D_{k_{MHj}}(X);$$

- с помощью ключа $E_{k_{MH1}}(k_{MHj})$, $j = 1,2$ можно зашифровать на ключе k_{MHj} любой набор данных X, используя операцию RFMK:

$$\text{EMK}(X) = E_{k_{MH}}(X),$$
$$\text{RFMK}(E_{k_{MH1}}(k_{MHj}), E_{k_{MH}}(X)) = E_{k_{MHj}}(X);$$

- наконец, с помощью ключа $E_{k_{MH2}}(k_{MHj})$, $j = 1,2$ можно для любого набора данных X, используя операцию RTMK, вычислить:

$$\text{RTMK}(E_{k_{MH2}}(k_{MHj}), X) = E_{k_{MH}}(D_{k_{MHj}}(X)).$$

Поэтому если в качестве X взять из базы данных запись $E_{k_{MHj}}(k_N)$, то в результате получится $E_{k_{MH}}(k_N)$, что недопустимо.

Таким образом, данный интерфейс с учетом введенных ограничений удовлетворяет поставленным условиям.

Дальнейшее развитие системы TSS. В более поздних версиях этого интерфейса при дальнейшем развитии концепции TSS были сделаны следующие дополнения:

- вместо накладывания двух масок c_i на главный ключ k_{MH} для получения вариантов главного ключа $k_{MHi} = k_{MH} \oplus c_i$ ($i = 1,2$) складывали главный ключ k_{MH} с контрольным вектором cv, который определяли в зависимости от контекста, т. е. назначения и условий использования ключа k (внешний/внутренний, период времени и область действия и т. д.), тем самым в криптографических операциях реально применялся ключ $k_{MHcv} = k_{MH} \oplus cv$;

• данный прием был распространен на все виды ключей, в том числе первичные и вторичные, т. е. при шифровании использовался не сам ключ, а его сумма с контрольным вектором $k \oplus cv$; благодаря этому нововведению каждый ключ мог использоваться для расшифрования только в рамках определенного контекста; кроме того, автоматически решалась проблема перекрывания ключей, так как зашифрованные на одном ключе сообщения при разном контексте реально шифруются теперь на разных ключах;

• помимо этого в систему была включена поддержка функций вычисления и проверки кодов аутентичности сообщений (имитовставок) для контроля целостности, а также шифрования персональных идентификаторов (PIN).

Таким образом, в архитектуре SCA каждая криптографическая функция снабжается собственным ключом. Не допускается использование одного ключа для выполнения различных криптографических функций. Это условие обеспечивается механизмом контрольного вектора. Контрольный вектор указывает назначение каждого конкретного ключа (ключ шифрования файлов, ключ выработки кода аутентичности сообщения, ключ шифрования персональных идентификаторов (PIN) и т. п.). Он хранится в открытом виде вместе с ключом. Без контрольного вектора ключ не может быть использован. Поэтому контрольный вектор служит дополнительным средством проверки целостности ключей при их хранении и использовании.

Контрольные задания

1. В каких случаях следует применять аппаратные средства для защиты ключей в компьютере?

2. Охарактеризуйте понятие «модель нарушителя».

3. Объясните базовый принцип работы криптоадаптера.

4. Каковы функции и особенности реализации криптоадаптера?

5. Объясните состав и назначение операций управления работой криптоадаптера.

6. Покажите, что если при создании первичных файловых ключей k вместо главного ключа использовать вторичные файловые ключи k_{SF} и генерировать случайные числа в виде $r = E_{k_{SF}}(k)$, то при пересылке файлов от одного хоста к другому можно обойтись одной операцией RTMK без RFMK.

7. Как модифицировать рассмотренные выше протоколы, чтобы они стали защищенными от атак, использующих повторение и отражение ранее переданных сообщений?

Семейство протоколов IPsec

11.1. Структура протокола IPsec

Протокол IPsec в настоящее время широко распространен. Он применяется для обеспечения безопасности межсетевого взаимодействия и является одним из вариантов защиты протокола IP (Internet Protocol). Он обеспечивает различные функции безопасности. В данной главе рассматривается иерархия протоколов обмена ключами в IPsec. Поскольку в основе ключевого обмена при выработке сеансовых ключей лежит протокол Диффи — Хеллмана, основная проблема защиты ключевого обмена заключается в выборе варианта усиления протокола DH в целях обеспечения свойства аутентифициии ключей.

Рассмотрим различные варианты такой защиты, предложенные в процессе исторического развития протокола IPsec.

Проанализируем общую структуру протокола IPsec, следуя [57]. На самом деле, говорить об IPsec как о протоколе некорректно, так как термин IPsec (IP Security Protocol — протокол защиты IP) обозначает совокупность различных механизмов, разработанных для защиты трафика на уровне IP (IPv4 или IPv6). Это, по сути, множество общих схем, протоколов и конкретных алгоритмов обеспечения следующих функций (служб) безопасности:

- целостность без установления соединения;
- аутентификация источника данных;
- защита от повторной передачи;
- конфиденциальность (конфиденциальность данных и частичная защита от анализа трафика при использовании тунельного режима).

Эти услуги обеспечены на уровне IP и, таким образом, составляют защиту уровня IP и всех протоколов верхнего уровня. Протокол IPsec является дополнительным на уровне IPv4, но обязательным для любой реализации на уровне IPv6.

Протокол IPsec разработан рабочей группой при IETF в 1992 г. и опубликован в 1995 г. в документах RFC. В первой версии не содержалось части, описывающей управление ключами. Эта часть была включена в новое описание только в ноябре 1998 г.

В отличие от протоколов TLS/SSL, SSH и некоторых других, которые работают на транспортном уровне, протокол IPsec предназначен для защиты обмена на уровне IP. Это обеспечивается применением двух механизмов защиты (AH и ESP), которые добавляют к традиционной обработке в протоколе на уровне IP:

- механизм AH (Authentication Header) предназначен для обеспечения целостности и аутентификации дейтаграмм IP без их шифрования (т. е. без конфиденциальности) в протоколе без установления соединения; механизм AH основан на принципе добавления к стандартному заголовку IP дополнительного поля (MAC), которое позволяет получателю проверить подлинность данных, включенных в дейтаграмму;
- механизм ESP (Encapsulating Security Payload) предназначен для того, чтобы гарантировать конфиденциальность, но может также обеспечить подлинность данных; принцип механизма ESP состоит в преобразовании исходной дейтаграммы IP в новую дейтаграмму, в которой данные и первоначальный заголовок зашифрованы, а затем аутентифицированы.

Приведем разрешенные алгоритмы для использования в протоколе IPsec:

1) алгоритмы для использования при механизме защиты AH:
   - HMAC-MD5, HMAC-SHA-1, AES-XCBC-MAC-96, AES-XCBC-PRF-128 (другие возможные алгоритмы — KPDK-MD5, DES-MAC, HMAC-RIPE-MD);

2) алгоритмы для использования при механизме защиты ESP:
   - шифрование: triple DES (MUST), DES-CBC, RC5, CAST, IDEA, triple IDEA, Blowfish, RC4, AES-CBC, AES-CTR и NULL для случаев, когда шифрование не требуется;
   - аутентификация: HMAC-MD5 (MUST), HMAC-SHA-1 (MUST), DES-MAC, HMAC-RIPE-MD, KPDK-MD5, AES-XCBC-MAC-96, AES-XCBC-PRF-128 и NULL для случаев, когда аутентификация не требуется.

В настоящее время описанию протоколов IPsec посвящено около 30 документов RFC, причем их развитие активно продолжается комитетом IETF, в работе у которого находится бо-

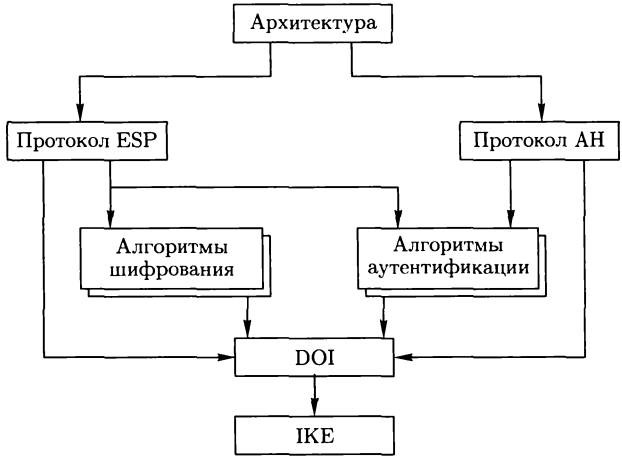

Рис. 11.1. Структура документов протокола IPsec

лее 20 предварительных версий документов (Internet-Drafts) вида draft-ietf-ipsec-.txt, посвященных различным аспектам работы протоколов IPsec.

Общая структура документов, описывающих IPsec, представлена на рис. 11.1.

Понятие защищенной ассоциации

В протоколе IPv6 не предусмотрено средств для того, чтобы поддерживать обмен ключами на уровне IP, при котором данные, необходимые для управления ключами, передавались бы с использованием специального заголовка IPv6. Поэтому протокол IPsec использует для управления ключевой системой верхние уровни, куда данные, связанные с ключевым управлением, транспортируются в соответствии с протоколом верхнего уровня типа UDP или TCP. Это позволяет осуществить явное отделение механизма управления ключами от других механизмов защиты. Таким образом, имеется возможность варьировать методы управления ключами, не внося изменений в выполнение механизмов защиты.

Упомянутые выше механизмы используют криптографические методы, поэтому требуется обмен некоторыми параметрами (используемыми алгоритмами шифрования, ключами, выбранными механизмами), с которыми обе стороны должны согласиться. Чтобы управлять этими параметрами, протокол IPsec использует понятие защищенной ассоциации.

Защищенная ассоциация (Security Association — SA) — однонаправленное соединение, которое обеспечивает функции защиты трафика. Для каждого направления создается своя связанная с ним защищенная ассоциация. Ее можно рассматривать как набор параметров, которые описывают, как данное соединение должно быть защищено. Для защиты обычной двунаправленной передачи требуется две защищенные ассоциации, по одной для каждого направления. Безопасность обеспечивается с помощью либо механизма AH, либо механизма ESP. Если для защиты трафика используют различные способы защиты, например и механизм AH, и механизм ESP, то защищенные ассоциации создают для каждого способа защиты; и их совокупность называют связкой SA (SA bundle).

Фактически, каждая защищенная ассоциация однозначно определена следующей тройкой:

1) адрес назначения пакетов;

2) идентификатор протокола защиты (AH или ESP);

3) индекс параметров защиты (SPI), представляющий собой блок из 32 бит, передаваемый в открытом виде в заголовке каждого пакета.

Протокол IPsec сохраняет активные защищенные ассоциации в базе данных, названной *базой данных защищенных ассоциаций* (Security Association Database — SAD). Она содержит все параметры, связанные с каждой защищенной ассоциацией, и позволяет узнать, как обработать каждый полученный или посланный пакет.

Ключи и управление защищенными ассоциациями

Как упомянуто выше, защищенная ассоциация содержит все параметры, необходимые для операций IPsec, включая используемые ключи. Управление ключами для различных механизмов защиты в протоколе IPsec осуществляется только через защищенную ассоциацию. Если ситуация достаточно проста, защищенная ассоциация может быть конфигурирована вручную, но общее правило заключается в том, чтобы использовать определенный протокол, который позволяет осуществлять динамический обмен значениями SA и, в частности, обмен сеансовыми ключами.

Протокол, позволяющий договариваться об используемой защищенной ассоциации, называют *протоколом установления защищенных ассоциаций и управления ключами* (Internet Secu-

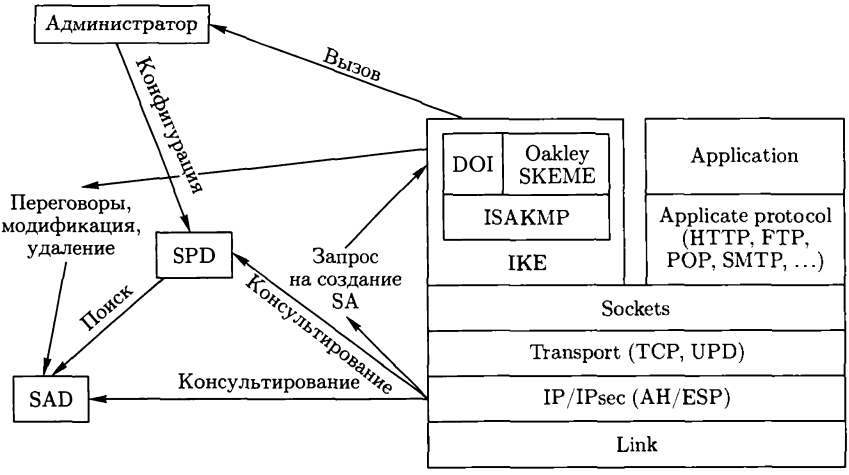

Рис. 11.2. Организация работы протокола IPsec

rity Association and Key Management Protocol — ISAKMP). Фактически, ISAKMP не является протоколом, это — универсальная структура (набор шаблонов), которая позволяет использовать несколько протоколов обмена ключами и которая может использоваться для других механизмов защиты в протоколе IPsec. В рамках структуры стандартизации IPsec применение протокола ISAKMP связано с протоколами SKEME и OAKLEY, а их комбинация объединена в результирующем протоколе IKE (Internet Key Exchange). Рассмотрим эти протоколы более подробно.

Организация работы протокола IPsec может быть представлена в виде схемы, изображенной на рис. 11.2. Обработка исходящего трафика происходит в определенном порядке. Если необходимо отправить какую-либо информацию, протокол IPsec консультируется с базой данных политик безопасности (Secure Policy Database — SPD), а затем ищет нужную защищенную ассоциацию в базе данных защищенных ассоциаций. Если таковая там имеется, то осуществляется передача. Если — нет, то протокол IPsec вызывает IKE для установления необходимой защищенной ассоциации.

Входящий трафик обрабатывается аналогично: при получении из сети пакета уровнем IPsec он проверяется на наличие механизмов защиты средствами IPsec, и если таковые присутствуют, то он консультируется с SAD для определения параметров алгоритмов защиты. Затем по SPD проверяется, правильно ли

реализованы эти механизмы и является ли извлеченный после деинкапсуляции пакет обычным пакетом IP. Если — является, то пакет выходит из IPsec. Если — нет, например выясняется, что это пакет IKE, то он должен обрабатываться средствами IKE и в случае неудачной попытки установления соединения сигнализировать об этом администратору.

11.2. Управление ключами в протоколе IPsec

Термин «управление» в данном контексте включает генерацию, распределение, хранение и уничтожение ключей.

В зависимости от роли в протоколе IPsec различают следующие типы ключей:

- *ключи шифрования ключей* (KEK) — верхний уровень иерархии ключей; предназначены для шифрования остальных ключей; как правило, являются долговременными; системы с открытыми ключами часто применяют только для шифрования передаваемых сеансовых ключей;
- *главные ключи* (master keys) — ключи, применяемые не для шифрования, а для генерации ключей; например, главный ключ можно использовать для генерации двух ключей: одного для шифрования и одного для подписи;
- *сеансовые ключи* (session keys) — ключи, имеющие короткий период действия (ключи могут меняться с каждым сообщением), в противоположность определенным выше, применяются для шифрования сообщений и расположены внизу иерархии; как правило, являются ключами симметричных криптосистем.

Необходимо различать протоколы с установлением соединения и протоколы без установления соединения. В первом случае аутентифицированные протоколы установки ключей используют на начальной стадии, до самого обмена. Затем результирующий ключ используют для защиты трафика IP. Таким образом, при указанном подходе установку и управление ключами нужно проводить на псевдосеансовом уровне, выше уровня IP, так как сам протокол IP является протоколом без установления соединения. Во втором случае используют аутентифицированный протокол, не требующий установления какого-либо соединения. Например, применяемые для шифрования ключи передают вместе с самими пакетами, закрытыми на открытом ключе получателя. Недостаток такого подхода состоит в увеличении длины каждого передаваемого пакта.

Как правило, в обоих случаях применяют протокол Диффи — Хеллмана, и различие состоит во времени действия используемых открытых значений и способе их аутентификации и передачи.

Протокол SKIP (Simple Key Management for Internet Protocols). Данный протокол предложен в 1994 г. А. Азиз и У. Диффи (A. Aziz, W. Diffie), сотрудниками компании «Sun Microsystems». Протокол SKIP был разработан в качестве протокола управлениями ключами для IPsec, и до 1996 г. было опубликовано несколько документов серии Internet-drafts. В сентябре 1996 г. предпочтение было отдано протоколу ISAKMP/OAKLEY, который к настоящему времени переименован в IKE и выбран как обязательный для всех применений, но использование протокола SKIP не запрещено. Он применяется в продуктах фирм «Sun Microsystems» и «Check Point».

Протокол SKIP является примером протокола, который не требует установления соединения. Для расшифрования полученного зашифрованного пакета не требуется никакой предварительной информации, так как такая информация содержится в самом пакете. Поэтому он расположен непосредственно на сетевом уровне, а не выше TCP и UDP подобно многим протоколам управления ключами.

Протокол SKIP основан на генерации общего секрета с использованием протокола Диффи — Хеллмана, в котором открытые значения предварительно аутентифицированы. Поэтому должно выполняться единственное требование, чтобы каждая из сторон заранее обладала аутентифицированными открытыми значениями. Аутентификация может быть проведена различными способами, например с использованием сертификатов X.509, secure DNS, подписи PGP и т. д. Кроме того, для информационного обмена с выбранной стороной каждая сторона должна вначале получить его открытое значение. Возможными путями распределения таких значений является служба каталогов (directory service) либо протокол выдачи сертификатов (Certificate Discovery Protocol — CDP).

Пусть A и B — две стороны. Их секретные значения обозначим соответственно x и y, а открытые значения — $\alpha^x \bmod p$ и $\alpha^y \bmod p$. Каждая из сторон вычисляет общий секрет по формуле

$$\alpha^{xy} \bmod p = (\alpha^y \bmod p)^x \bmod p = (\alpha^x \bmod p)^y \bmod p.$$

Значение $\alpha^{xy} \bmod p$ служит долговременным секретом и используется для вычисления секретного ключа k_{AB}. Как прави-

ло, значение $\alpha^{xy} \bmod p$ имеет не менее 1 024 бит, в то время как значение k_{AB} может быть от 40 до 256 бит. В протоколе SKIP ключ k_{AB} формируется из младших разрядов вектора $\alpha^{xy} \bmod p$. Ключ k_{AB} используется только для шифрования ключей, имеющих малый срок действия. Фактически ключ k_{AB} используется для зашифрования ключа k_p, называемого *пакетным ключом* (packet key), который используется для формирования двух ключей: одного (k_1) для шифрования содержимого пакета и другого (k_2) для аутентификации пакетов (или частей пакетов) IP с помощью некоторого кода аутентификации:

$$A \rightarrow B \quad E_{k_{AB}}(k_p), E_{k_1}(\text{text}), h_{k_2}(\text{text}).$$

Алгоритм шифрования и код аутентификации заранее не определены и могут быть выбраны из некоторого достаточно широкого списка.

Поскольку SKIP является протоколом без установления соединения, в нем происходит увеличение длины пакетов — добавляется так называемый «SKIP-заголовок», совпадающий по своей структуре с IP-заголовком и отличающийся лишь тем, что в нем передается ключ k_p (зашифрованный на ключе k_{AB}) и указываются используемые алгоритмы.

Расширение SKIP PFS. Приведенный выше протокол не обеспечивает свойства защиты от чтения назад (perfect forward secrecy), так как очевидно, что в случае компрометации ключа k_{AB} все сеансовые ключи k_p, использованные ранее, будут скомпрометированы. Поэтому было разработано расширение протокола SKIP, которое гарантирует дополнительную безопасность. Это расширение использует так называемый протокол ephemeral DH, в котором открытые значения имеют короткий срок действия. Защита протокола обеспечивается за счет обмена сертификатами, соответствующими этим значениям, вырабатываемыми с помощью протокола CDP. Для закрытия обмена используется ключ k_{AB}. В отличие от основного протокола SKIP протокол SKIP PFS требует проведения дополнительного предварительного обмена сообщениями между сторонами до того, как они получат возможность обмениваться зашифрованными сообщениями.

Протокол Photuris. Предложенный в 1995 г. П. Карном (P. Karn), сотрудником компании «Qualcomm», и У. Симпсоном (W. Simpson), сотрудником компании «DayDreamer», протокол Photuris использует те же принципы, что и протокол STS (Station-To-Station). Протокол Photuris описан в нескольких доку-

ментах серии Internet-drafts и *.RFC и разработан для использования в системе протоколов IPsec. Он используется лишь несколькими приложениями, так как применяемый в настоящее время протокол IKE намного более общий.

В противоположность SKIP, Photuris — протокол, ориентированный на соединение. Он требует осуществления нескольких предварительных обменов (для выбора опций и генерации ключей) до обмена шифрованными сообщениями (ему выделен UDP порт 468).

Протокол Photuris основан на генерации общего секрета с помощью протокола Диффи — Хеллмана. Этот общий секрет имеет короткий срок действия и используется для генерации сеансового ключа, на котором будет шифроваться весь последующий обмен. Для защиты от атаки «противник в середине» пересылаемые открытые значения затем аутентифицируют с помощью долговременного ключа. Эти долговременные ключи используют только для аутентификации, поэтому протокол Photuris обеспечивает выполнение свойства защиты от чтения назад.

Использование в алгоритме Диффи — Хеллмана трудоемких операций делает его уязвимым к атакам типа «отказ в обслуживании». Чтобы затруднить проведение таких атак, протокол Photuris использует предварительный обмен значениями специальных системных идентификаторов (cookies) до выполнения протокола Диффи — Хеллмана. Значения cookies определяют с помощью легко вычисляемой функции, зависящей от системных установок обеих сторон: их IP-адресов, номеров UDP-портов и т. п. Это не позволяет атакующему правильно определить значения cookies и использовать их для проведения атаки «шторм пакетов» со случайных различных IP-адресов или портов. Кроме того, чтобы сделать невозможной для атакующего генерацию поддельных значений cookies, которые были бы приняты стороной в качестве ее собственных, в качестве аргументов используют дополнительные локальные секреты, о которых знают стороны и не знает нарушитель.

Протокол Photuris состоит из трех фаз (стадий):

1) обмен значениями cookies, которые вычисляют по определенным алгоритмам из некоторой системной информации; предназначен для затруднения проведения атак DoS; каждая из сторон вычисляет свое значение cookie и вставляет его во все свои сообщения;

2) обмен открытыми значениями для генерации общего секрета;

3) обмен аутентификаторами для взаимной идентификации сторон и проверки подлинности переданных на предыдущей стадии открытых значений; все сообщения при этом обмене закрыты на секретном ключе, полученном из общего секрета и значений cookies.

В последующих сообщениях можно изменять как сеансовые ключи, так и параметры безопасности. Конфиденциальность этих сообщений обеспечивается так же, как и на третьей стадии. Одновременно стороны могут договориться о методе генерации общего секрета и параметрах безопасности для защищенной ассоциации.

Протокол SKEME. Разработанный специально для системы протоколов IPsec в 1996 г. Х. Кравчуком (H. Krawczyk), сотрудником компании «IBM», и Т. Ватсоном (T. J. Watson), сотрудником компании «Research Center», протокол SKEME представляет собой ориентированный на установление соединения протокол, являющийся расширением протокола Photuris. В отличие от Photuris протокол SKEME обеспечивает различные режимы ключевого обмена. Подобно протоколам STS и Photuris основной (базовый) режим протокола SKEME основан на использовании открытых ключей и генерации общего секрета методом Диффи — Хеллмана. Однако протокол SKEME не обязательно использует открытые ключи, но может использовать и предварительно распределенный секретный ключ. Этот ключ может быть получен непосредственно в центре распределения ключей (KDC), подобно Kerberos, либо при посредничестве центра, когда KDC выступает в качестве доверенной третьей стороны. Использование этого секретного ключа для аутентификации передаваемых открытых значений протокола Диффи — Хеллмана, а не самого сеансового ключа, снижает уровень требований к доверенности для KDC. Наконец, протокол SKEME позволяет отказаться от использования протокола Диффи — Хеллмана в случаях, когда не требуется свойство PFS.

Протокол SKEME включает четыре различных режима:

1) основной режим (basic mode), который обеспечивает обмен ключами на основе открытых ключей и гарантирует свойство PFS благодаря использованию протокола Диффи — Хеллмана;

2) режим обмена ключами на основе открытых ключей без использования протокола Диффи — Хеллмана;

3) режим обмена ключами на основе предварительно распределенного секретного ключа с использованием протокола Диффи — Хеллмана;

4) режим быстрой замены ключа, основанный только на симметричных алгоритмах.

Помимо этого протокол SKEME состоит из трех фаз: SHARE, EXCH, AUTH.

Во время фазы SHARE стороны обмениваются «половинами» ключей (half-keys), зашифрованными с помощью соответствующих открытых ключей. Эти две половины используются для вычисления секретного ключа k_0:

1) $A \rightarrow B \quad A, E_B(k_A)$;
2) $A \leftarrow B \quad B, E_A(k_B)$.

В результате стороны вычисляют общий ключ $k_0 = h(k_A, k_B)$. Если требуется обеспечить анонимность, то идентификаторы обеих сторон также шифруют:

1) $A \rightarrow B \quad E_B(A, k_A)$;
2) $A \leftarrow B \quad E_A(B, k_B)$.

Если общий секрет уже существует, то эта фаза опускается.

Фаза обмена EXCH используется в зависимости от выбранного режима для обмена открытыми значениями для протокола Диффи — Хеллмана или случайными (nonces) значениями. Общий секрет для протокола Диффи — Хеллмана будет определен только в конце этой фазы:

3) $A \rightarrow B \quad m_A = \alpha^x \bmod p$;
4) $A \leftarrow B \quad m_B = \alpha^y \bmod p$.

Открытые или случайные значения аутентифицируются во время выполнения фазы аутентификации AUTH с использованием секретного ключа, полученного на фазе SHARE:

5) $A \rightarrow B \quad h_{k_0}(m_B, m_A, A, B)$;
6) $A \leftarrow B \quad h_{k_0}(m_A, m_B, B, A)$.

Сообщения этих трех фаз не обязательно следуют друг за другом, они могут объединяться для минимизации числа передаваемых сообщений. Например, в основном режиме они представлены так:

1) $A \rightarrow B \quad E_B(A, k_A), m_A$;
2) $A \leftarrow B \quad E_A(B, k_B), m_B, h_{k_0}(m_A, m_B, B, A)$;
3) $A \rightarrow B \quad h_{k_0}(m_B, m_A, A, B)$.

В результате будет выработан следующий сеансовый ключ: $k = h(\alpha^{xy} \bmod p)$.

К этим трем фазам может быть добавлена еще одна фаза — обмен специально сформированными значениями системных идентификаторов (cookies), которая выполняется перед фазой SHARE для защиты от атак DoS аналогично криптографическому протоколу Photuris.

Протокол OAKLEY. Предложенный Х. Орман (H. Orman), Университет Аризона, протокол OAKLEY является вместе с ISAKMP и SKEME основным протоколом обмена ключами в системе IPsec. Протокол OAKLEY представляет собой протокол, имеющий много общего со SKEME: он также имеет несколько режимов, использует для защиты от атак на отказ в обслуживании (атак DoS) системные идентификаторы cookies и не требует вычисления общего секрета по протоколу Диффи — Хеллмана до завершения выполнения протокола. Он отличается от предыдущих протоколов тем, что позволяет сторонам явно договориться об используемых механизмах обмена ключами, шифрования и аутентификации. Главным достоинством протокола OAKLEY является то, что он позволяет сторонам безопасно выбрать набор параметров для защиты обмена: имя ключа, секретный ключ, идентификаторы сторон, алгоритмы шифрования, аутентификации и хеш-функции.

Протокол OAKLEY имеет дополнительные возможности, он позволяет в дополнение к традиционному обмену ключами Диффи — Хеллмана использовать получение нового ключа из старого или распределять зашифрованный ключ. Эти опции появляются благодаря некоторым дополнительным режимам.

Главный принцип протокола OAKLEY в том, что инициатор обмена сначала определяет столько информации в своем сообщении, сколько он пожелает. Получатель в свою очередь отвечает таким объемом информации, как он желает. Диалог продолжается до тех пор, пока стороны не придут к общему соглашению. Выбор количества включаемой в каждое сообщение информации зависит от выбранных опций (использование независимых от состояния cookies, защита идентификаторов, PFS, невозможность отказа от своих действий и т. д.).

Протокол имеет три составных части:

1) обмен значениями сформированных для данной архитектуры специальных идентификаторов (cookies); значения cookies вычисляют путем редуцирования значений хеш-функции от адресов, идентификаторов и некоторых параметров конкретного способа реализации (по умолчанию постоянны и не зависят от состояния); идентификаторы cookies предназначены для защиты от атак на отказ в обслуживании; значения cookies входят в заголовки передаваемых пакетов; далее они обозначены c_A, c_B;

2) обмен открытыми значениями по протоколу Диффи — Хеллмана (может отсутствовать);

3) аутентификация (опции: анонимность идентификаторов, невозможность отказа от своих действий).

Рассмотрим более подробно различные режимы протокола OAKLEY.

Мы не будем обсуждать заголовки и поля передаваемых сообщений, предназначенные для выбора защищенных ассоциаций, будем анализировать только те поля сообщений, которые влияют на криптографическую часть протокола. Это позволяет упростить записи и сделать их более компактными и понятными для анализа.

В описании протокола OAKLEY [71] приведены четыре варианта протокола формирования общего ключа для различных режимов использования этого протокола. Выделим из описания только криптографическую часть этих протоколов. Пусть, как и выше, $m_A = \alpha^x \bmod p$, $m_B = \alpha^y \bmod p$.

1. *Энергичный режим* (OAKLEY aggressive) определяется следующим протоколом:

1) $A \to B : m_A, A, B, r_A, \mathrm{Sig}_{\mathrm{sk}_A}(A, B, r_A, m_A)$;

2) $A \leftarrow B : m_B, B, A, r_B, r_A, \mathrm{Sig}_{\mathrm{sk}_B}(B, A, r_B, r_A, m_B, m_A)$;

3) $A \to B : m_A, A, B, r_A, r_B, \mathrm{Sig}_{\mathrm{sk}_A}(A, B, r_A, r_B, m_A, m_B)$,

где x, y — случайные значения, генерируемые участниками, как и в обычном протоколе DH; $(\mathrm{pk}_A, \mathrm{sk}_A)$, $(\mathrm{pk}_B, \mathrm{sk}_B)$ — пары открытых/секретных ключей участников A и B.

Результирующий ключ, сформированный в результате выполнения фазы установления защищенной ассоциации и выработки ключа для использования в механизме AH или ESP (см. далее) протокола IPsec, вычисляется по формуле

$$\mathrm{sKEYID} = h_{(r_A, r_B)}(\alpha^{xy} \bmod p, c_A, c_B), \qquad (11.1)$$

где c_A, c_B — значения системных идентификаторов cookies из заголовков передаваемых пакетов.

Вместо цифровой подписи может использоваться код аутентичности сообщения, вычисленный с помощью одной из оговоренных в протоколе ISAKMP защищенных ассоциаций хеш-функций h, задаваемых ключом, вид которой установлен в *.rfc документах. Порядок такой замены следующий:

$$\mathrm{Sig}_{\mathrm{sk}_A}(\mathrm{data}) = h_{k32}(\mathrm{data}),$$

где строка $k32$ — общий ключ A и B, распределяемый предварительно.

В данном протоколе для усиления обычного протокола DH использованы две известные конструкции: техника «запрос—

ответ» для проведения взаимной аутентификации сторон и сеанса с использованием случайных значений (nonces) r_A и r_B и аутентификация сообщений путем добавления цифровой подписи (кода аутентичности сообщения) под передаваемыми открытыми значениями алгоритма Диффи — Хеллмана или добавления кода аутентичности сообщения, вычисленного с помощью хеш-функции h, задаваемой общим ключом.

Заметим, что если в качестве результирующего ключа был использован ключ

$$\text{sKEYID} = h_{(r_A, r_B)}(\alpha^{xy} \bmod p),$$

не зависящий от значений системных идентификаторов c_A и c_B, то для этого режима протокола в случае использования цифровой подписи была бы применима двусторонняя атака с неизвестным общим ключом:

$$\begin{aligned}
A &\to C && m_A, A, C, r_A, \mathrm{Sig}_{\mathrm{sk}_A}(A, C, r_A, m_A),\\
&\quad C \to D && m_A, A, B, r_A,\\
&\qquad D \to B && m_A, D, B, r_A, \mathrm{Sig}_{\mathrm{sk}_D}(D, B, r_A, m_A),\\
&\qquad D \leftarrow B && m_B, B, D, r_B, r_A, \mathrm{Sig}_{\mathrm{sk}_B}(B, D, r_B, r_A, m_B, m_A),\\
&\qquad D \to B && m_A, D, B, r_A, r_B, \mathrm{Sig}_{\mathrm{sk}_D}(D, B, r_A, r_B, m_A, m_B),\\
&\quad C \leftarrow D && m_B, B, A, r_B,\\
A &\leftarrow C && m_B, A, C, r_B, r_A, \mathrm{Sig}_{\mathrm{sk}_C}(B, C, r_B, r_A, m_B, m_A),\\
A &\to C && m_A, A, C, r_A, r_B, \mathrm{Sig}_{\mathrm{sk}_A}(A, C, r_A, r_B, m_A, m_B).
\end{aligned}$$

В результате противник, в качестве которого выступает коалиция участников C и D, открыв и успешно завершив два сеанса протокола между A и C и между B и D, вводит в заблуждение участников A и B относительно того, с кем они на самом деле сформировали общий ключ. Тем самым свойство аутентификации ключа оказывается нарушенным. Хотя результирующий ключ sKEYID останется противнику неизвестным, оба участника A и B примут его в качестве общего ключа.

В случае использования энергичного режима с кодом аутентичности сообщения вместо цифровой подписи подобная атака уже неприменима, так как в качестве ключевой переменной хеш-функции используется распределяемый предварительно между участниками A и B общий ключ $k32$.

2. *Энергичный режим с защитой идентификаторов* (OAKLEY aggressive with hidden identities) определяется следующим протоколом:

1) $A \to B \quad m_A, B', E_{B'}(A, B, E_B(r_A))$;
2) $A \leftarrow B \quad m_B, E_A(B, A, r_B), h_{k_{AB}}(B, A, m_B, m_A)$;
3) $A \to B \quad m_A, h_{k_{AB}}(A, B, m_A, m_B)$,

где B' — заранее согласованный с A идентификатор прикрытия для B с известным абоненту A открытым ключом.

Результирующий ключ вычисляют по формуле (11.1), а ключ для хеш-функции по формуле $k_{AB} = h_0(r_A, r_B)$.

В данном протоколе использованы две базовые конструкции. Во-первых, это техника «запрос — ответ» для проведения взаимной аутентификации сторон и сеанса с использованием случайных значений r_A, r_B:

1) $A \to B \quad B', E_{B'}(A, B, E_B(r_A))$;
2) $A \leftarrow B \quad E_A(B, A, r_B)$.

Заметим, что в третьем сообщении протокола значения r_A, r_B спрятаны в ключевую переменную хэш-функции, а не зашифрованы. Оно выбрано таким образом, чтобы из него нельзя было извлечь явное значение r_A, хотя оно участвует в процессе аутентификации на третьем шаге. Это известный прием для защиты от возможного некорректного поведения участника B, который может попытаться в дальнейшем выступить от имени A.

Кроме того, здесь применяется аутентификация сообщений путем добавления кодов аутентичности сообщений, вычисленных с помощью ключевой хэш-функции h на общем ключе k_{AB}:

1) $A \to B \quad m_A$;
2) $A \leftarrow B \quad m_B, h_{k_{AB}}(B, A, m_B, m_A)$;
3) $A \to B \quad m_A, h_{k_{AB}}(A, B, m_A, m_B)$.

Заметим, что потенциальной слабостью этого режима протокола является выбор значения k_{AB} как обычной свертки значений r_A и r_B, не зависящей от значения $\alpha^{xy} \bmod p$. Если бы, как это было в одной из предыдущих версий протокола, в качестве результирующего ключа использовался ключ sKEYID, не зависящий от значений системных идентификаторов c_A и c_B, то был бы применим пример двусторонней атаки с неизвестным общим ключом на данный протокол [38]:

$A \to C \qquad\qquad m_A, C', E_{C'}(A, B, E_C(r_A))$,
$C \to D \qquad\qquad m_A, A, B, r_A$,
$D \to B \qquad m_A, B', E_{B'}(D, B, E_B(r_A))$,
$D \leftarrow B \qquad m_B, E_D(B, D, r_B), h_{k_{AB}}(B, D, m_B, m_A)$,
$D \to B \qquad m_A, h_{k_{AB}}(D, B, m_A, m_B)$,
$C \leftarrow D \qquad m_B, B, A, r_B$,
$A \leftarrow C \qquad\qquad m_B, E_A(C, A, r_B), h_{k_{AB}}(B, A, m_B, m_A)$,
$A \to C \qquad\qquad m_A, h_{k_{AB}}(A, C, m_A, m_B)$.

Поскольку противник, в качестве которого выступает коалиция участников C и D, может вычислить значение ключа k_{AB}, в результате он может подделать значения кодов аутентичности сообщений, тем самым введя в заблуждение участников A и B относительно того, с кем они на самом деле сформировали общий ключ.

3. *Энергичный режим с защитой идентификаторов без использования протокола Диффи — Хеллмана* (OAKLEY aggressive with private identities and without Diffie — Hellman) определяется следующим протоколом:

1) $A \to B \quad B', E_{B'}(A, B, k_A), r_A$;
2) $A \leftarrow B \quad E_A(B, A, k_B), r_B, h_{k_{AB}}(B, A, r_B, r_A)$;
3) $A \leftrightarrow B \quad h_{k_{AB}}(A, B, r_A, r_B)$,

где k_A, k_B — секретные ключи отправителя и получателя соответственно, генерируемые как попсе.

Ключ для хеш-функции вычисляют по формуле

$$k_{AB} = h_0(k_A, k_B),$$

результирующий ключ находят согласно

$$\text{sKEYID} = h_{k_{AB}}(c_A, c_B).$$

Для выработки общего ключа путем открытого ключевого обмена, но без использования протокола Диффи — Хеллмана, здесь используют сообщения:

$$A \to B \quad B', E_{B'}(A, B, k_A),$$
$$A \leftarrow B \quad E_A(B, A, k_B).$$

Кроме того, для защиты идентификаторов применяют ссылку на заранее оговоренного третьего абонента с известным абоненту A открытым ключом, выполняющего функцию прикрытия для абонента B. Остальное — это стандартная техника «запрос — ответ» с использованием хеш-функции для аутентификации чисел r_A и r_B:

$$A \to B \quad r_A,$$
$$A \leftarrow B \quad r_B, h_{k_{AB}}(B, A, r_B, r_A),$$
$$A \to B \quad h_{k_{AB}}(A, B, r_A, r_B).$$

4. *Консервативный режим* (OAKLEY conservative) описывается протоколом вида

3) $A \to B : m_A$;
4) $A \leftarrow B \quad m_B$;
5) $A \to B : m_A, E_k(A, B, E_B(r_A))$;
6) $A \leftarrow B \quad m_B, E_k(E_A(r_B, r_A), B, A, h_{k_{AB}}(B, A, m_B, m_A))$;
7) $A \to B \quad m_A, E_k(h_{k_{AB}}(A, B, m_A, m_B))$.

Здесь, как и в агрессивном режиме, $k = h_0(\alpha^{xy} \bmod p)$, $k_{AB} = h_0(r_A, r_B)$, а результирующий ключ вычисляют по формуле (11.1).

Заметим, что шифрование сообщений 5 — 7 на ключе k не защищает протокол от вмешательства третьей стороны, так как ключевой обмен осуществляется на шаге 3 и 4 в открытом неаутентифицированном виде, что допускает подмену передаваемых сообщений, приводящую в результате к навязыванию другого значения ключа. Поэтому на аутентификацию влияют только внутренние поля сообщений 5 — 7.

Два поля $E_B(r_A)$ и $E_A(r_B, r_A)$ взяты из трехшагового протокола:

1) $A \to B \quad A, B, E_B(r_A)$;
2) $A \leftarrow B \quad E_A(r_B, r_A), B, A$;
3) $A \to B \quad h_{k_{AB}}(A, B, \ldots)$.

Первые два шага здесь стандартны, а третий модифицирован для совмещения с дальнейшей аутентификацией и одновременной защитой от нечестного поведения третьего участника C, который может выступить промежуточным звеном в обмене между участниками A и B. Известная атака на подобный протокол заключается в том, что, используя свой законный обмен с абонентами A и B, абонент C может получить возможность в дальнейшем вступать во взаимодействие с абонентом A от имени B. Для защиты от этой атаки на третьем шаге секретное случайной значение r_B спрятано односторонним образом в ключевую переменную k_{AB} хеш-функции, что не позволяет его извлечь из третьего сообщения.

Другой стандартной конструкцией, применяемой в этом протоколе, является обмен аутентификаторами на шагах 6 и 7 протокола:

$$\begin{aligned} A &\to B \quad m_A, \\ A &\leftarrow B \quad m_B, h_{k_{AB}}(B, A, m_B, m_A), \\ A &\to B \quad h_{k_{AB}}(A, B, m_A, m_B), \end{aligned}$$

которые представляют собой значения кодов аутентичности сообщения, вычисленных с помощью хеш-функции, задаваемой ключом k_{AB}. Они подтверждают целостность и подлинность ключевого обмена. Неявно здесь также используется техника «запрос — ответ» с применением аутентификаторов, где в качестве запросов выступают значения m_A, m_B.

Важным является одновременное использование в качестве ключевых переменных значений $k = h_0(\alpha^{xy} \bmod p)$, защищаю-

щих этот режим от двусторонней атаки с неизвестным общим ключом, а также значения $k_{AB} = h_0(r_A, r_B)$, с помощью которых осуществляется привязка кодов аутентичности сообщений к одному сеансу протокола, что обеспечивает его целостность и невозможность использования противником сообщений из разных сеансов протокола. Симметричность данных сообщений не играет роли, так как в исходном протоколе на шагах 6, 7 они используются несимметричным образом.

Протокол ISAKMP

Протокол IKE (Internet Key Exchange) разработан специально для системы IPsec; он обеспечивает аутентификацию ключей и обмен ключами для большинства ситуаций. Протокол состоит из нескольких элементов: ISAKMP и двух протоколов OAKLEY и SKEME. Конкретное применение к системе IPsec протокола IKE определяется в документе, называемом IPsec DOI (Domain of Interpretation).

Протокол установления защищенных ассоциаций и управления ключами ISAKMP разработан для обмена, установления, изменения и удаления защищенной ассоциации и ее атрибутов. Он представляет собой общий шаблон, не зависящий от конкретных механизмов, обмен параметрами которых происходит при работе протокола. Протокол ISAKMP может быть использован для обмена в форме SA параметрами, относящимися к любым механизмам: IPsec, TLS и т. п. Он создан для поддержки обмена защищенными ассоциациями для любого протокола уровня IP (и выше). Это возможно благодаря способу обмена. Прежде всего, протокол ISAKMP работает независимо от механизмов безопасности: данные ключевого обмена передаются отдельно. Протокол ISAKMP можно реализовать непосредственно выше уровня IP либо выше любого протокола транспортного уровня. В частности, к нему отнесен UDP порт 500. Кроме того, протокол ISAKMP определяет порядок обмена защищенными ассоциациями, но он ничего не говорит о параметрах, которые их составляют. Документ IPsec DOI (см. далее) должен определять обмен параметрами и соглашения об использовании протокола ISAKMP и представляет собой отдельное руководство. Идентификатор документа DOI используется для интерпретации содержания сообщений протокола ISAKMP. Спецификации IPsec DOI определены в RFC 2407.

Протокол ISAKMP имеет две фазы, которые явно разделяют защиту трафика самого протокола ISAKMP и процедуру обмена параметрами для выбора SA.

В течение первой фазы осуществляется обмен множествами атрибутов, связанных с безопасностью, аутентификацией сторон, и генерируются некоторые ключи. Эти элементы составляют первую защищенную ассоциацию, известную как ISAKMP SA. В отличие от IPsec SA защищенная ассоциация ISAKMP SA двунаправлена и используется для защиты всех сообщений самого протокола ISAKMP.

Вторая фаза используется для обмена параметрами, связанными с защищенной ассоциацией, для конкретного используемого механизма безопасности (например, AH или ESP). Обмен на этой фазе защищен (конфиденциальность, аутентификация и др.) с помощью ISAKMP SA. На этой фазе ISAKMP SA можно использовать для обмена несколькими IPsec SA.

Параметры защищенной ассоциации ISAKMP SA могут относиться только к ISAKMP либо могут содержать некоторые специфические элементы другого конкретного протокола безопасности, определенного в соответствующем DOI. В первом случае защищенная ассоциация называется внутренней (generic) ISAKMP SA, и ее можно использовать для обмена SA в любом протоколе безопасности. Во втором случае ISAKMP SA может быть использована для обмена SA только при таком же самом DOI.

Построение сообщения из вложений (payloads). Протокол ISAKMP не зависит от метода генерации ключей и используемых алгоритмов аутентификации и шифрования, в частности он не зависит ни от какого конкретного протокола обмена ключами. Это позволяет явно отделить фрагменты управления защищенными ассоциациями от фрагментов ключевого обмена. Таким образом, совместно с протоколом ISAKMP может быть использовано множество протоколов обмена ключами, обладающих различными свойствами. Это возможно благодаря тому, что ISAKMP не привязан к конкретному типу сообщений. Скорее ISAKMP представляет собой некий общий принцип построения, согласно которому сообщения ISAKMP состоят из заголовка и нескольких различных следующих за ним вложений (payloads), из которых составляются сообщения протокола ISAKMP.

Каждое сообщение ISAKMP начинается с заголовка фиксированной длины. Он включает значения двух cookies, соответствующих инициатору и ответчику, которые помимо защиты от

атак DoS дают возможность идентифицировать действующую защищенную ассоциацию ISAKMP SA (в отличие от IPsec SA она не определяется значениями SPI). Поле, называемое Exchange Type, указывает тип действующего обмена, его порядок и число сообщений. Заголовок ISAKMP также включает поле Next Payload, которое указывает вид первого вложения в сообщение. Каждое вложение в сообщение начинается со своего собственного заголовка, который указывает его длину и вид следующего вложения. Последнее вложение в сообщение обозначают знаком 0 в поле видов следующего вложения. Конструкция сообщений ISAKMP, таким образом, представляет собой сцепление вложений. Всего может быть 13 различных видов вложений.

Типы обмена. Протокол ISAKMP определяет пять типов обмена. Тип обмена представляет собой спецификацию множества сообщений, образующих данный обмен, и предназначен для обеспечения реализуемых функций безопасности: анонимности, PFS, взаимной аутентификации и т. п. Документ RFC 2408 определяет, какие виды вложений соответствуют каждому типу обмена. Другие типы обмена можно определить с использованием DOI и конкретных протоколов обмена ключами.

Спецификация ISAKMP содержит пять типов обмена:

1) базовый обмен (base exchange);

2) обмен с защитой идентификаторов (identity protection exchange);

3) обмен с аутентификацией (authentication-only exchange);

4) энергичный обмен (aggressive exchange);

5) информационный обмен (informational exchange).

Будем использовать следующие обозначения: HDR — заголовок ISAKMP; SA — выбор SA; KE — ключевой обмен; ID — идентификатор; AUTH — аутентификатор (HASH или SIG); NONCE — случайное число; символ $*$ означает, что далее все сообщение зашифровано.

Базовый обмен (base exchange) предназначен для реализации ключевого обмена и обмена связанной с ним аутентификационной информацией. При этом идентификаторы передаются в открытом виде, так как общий ключ появляется только после второго сообщения протокола:

1) $A \rightarrow B$ HDR, SA, NONCE (выбор параметров SA);

2) $A \rightarrow B$ HDR, SA, NONCE;

3) $A \leftarrow B$ HDR, KE, ID_A, AUTH (проверка аутентичности);

4) $A \rightarrow B$ HDR, KE, ID_B, AUTH.

Данные сообщений 3 и 4 защищены на общем ключе с помощью вложения AUTH, реализованного с использованием алгоритма, выбранного благодаря сообщениям 1 и 2.

Обмен с защитой идентификаторов (identity protection exchange) предназначен для отделения информации ключевого обмена от идентификаторов и связанной с ним аутентификационной информации. Идентификаторы передают защищенными на предварительно распределенном ключе:

1) $A \rightarrow B$ HDR, SA (выбор параметров SA);
2) $A \leftarrow B$ HDR, SA;
3) $A \rightarrow B$ HDR, KE, NONCE (ключевой обмен);
4) $A \rightarrow B$ HDR, KE, NONCE;
5) $A \leftarrow B$ HDR*, ID_A, AUTH (проверка аутентичности);
6) $A \rightarrow B$ HDR*, ID_B, AUTH.

Обмен с аутентификацией (authentication only exchange) предназначен для обмена аутентификационной информацией. Его преимуществом является то, что он позволяет провести аутентификацию без трудоемких операций вычисления ключей. Используется обычно на второй фазе с использованием SA выбранной на первой фазе:

1) $A \rightarrow B$ HDR, SA, NONCE;
2) $A \leftarrow B$ HDR, SA, NONCE, ID_B, AUTH;
3) $A \rightarrow B$ HDR, ID_A, AUTH.

Энергичный обмен (aggressive exchange) предназначен для выбора SA, ключевого обмена и обмена связанной с ним аутентификационной информацией, передаваемыми одновременно. Это позволяет сократить число передаваемых сообщений. При этом идентификаторы не защищены и число сообщений запрос—ответ при выборе SA сокращено до одного:

1) $A \rightarrow B$ HDR, SA, KE, NONCE, ID_A;
2) $A \leftarrow B$ HDR, SA, KE, NONCE, ID_B, AUTH;
3) $A \rightarrow B$ HDR*, AUTH.

Информационный обмен (informational exchange) предназначен для односторонней передачи информации, которая может понадобиться для управления SA. Использует только вложения вида Notification или Delete (N/D) и защищено с помощью ISAKMP SA, которая была ранее выбрана:

$$A \rightarrow B \quad \mathrm{HDR}^*, \mathrm{N/D}.$$

Протокол IPsec DOI

Протокол ISAKMP определяет порядок обмена защищенными ассоциациями, но ничего не говорит о параметрах, их составляющих.

Документ Domain of Interpretation (DOI), структура которого описана в RFC 2407, объединяет конкретные параметры обмена и соглашения об использовании ISAKMP в одном руководстве. Указание на DOI используется для интерпретации сообщений ISAKMP с использованием следующих вложений:

1) вложение SA: ситуация; использование вложения этого типа определяет одну из трех ситуаций, при которых происходит выбор SA:

- identity only — аутентификация с помощью вложений типа ID;
- secrecy — с указанием уровня конфиденциальности;
- integrity — с указанием уровня целостности;

2) вложение P: протокол; варианты выбора протокола в IPsec DOI следующие:

- ISAKMP;
- AH;
- ESP;
- IPCOMP — протокол сжатия данных, работающий на уровне IP; применяется до шифрования;

3) вложение T: transform and attributes; для каждого из четырех упомянутых протоколов возможно использование различных способов преобразования, вложение transform указывает на этот выбор:

- для ISAKMP – это указание на протокол обмена ключами;
- для IPsec — это только IKE;
- для AH – это алгоритмы MD5, SHA, DES, AES, применение которых дает возможность реализовать варианты: HMAC-MD5, KPDK-MD5, HMAC-SHA, DES-MAC, AES-XCBC-MAC-96, AES-XCBC-PRF-128;
- для ESP – это DES_IV32, DES_IV64 (DES в режиме CBC с инициализационным вектором длиной 32 или 64 бита соответственно), DES (в режиме CBC), triple DES, RC5, IDEA, CAST, BLOWFISH, triple IDEA, RC4, AES-CBC, AES-CTR и NULL (для использования ESP без шифрования);
- для IPCOMP – это OUI, DEFLATE, LZS и V42BIS;

в дополнение к этим атрибутам можно выбирать группу для алгоритма Диффи Хеллмана, время действия SA, длину ключа (см. RFC 2407);

4) вложение ID; указывает тип протокола (UDP, TCP), используемый им порт и принятые способы идентификации;

5) вложение N; вводит три статуса сообщения:

- responder-lifetime;
- replay-status;
- initial-contact.

Протокол IKE

Если протокол IPsec DOI определяет содержание вложений в пакеты протокола ISAKMP применительно к IPsec, то протокол IKE использует ISAKMP для построения практического протокола. Протокол управления ключами, ассоциированный с ISAKMP, строится на основе либо OAKLEY, либо SKEME. Более точно, протокол IKE использует некоторые режимы из OAKLEY и заимствует из SKEME использование открытых ключей для шифрования при аутентификации, а также метод быстрой смены ключей с помощью обмена случайными значениями nonces. Кроме того, протокол IKE не зависит от конкретного DOI, а может использовать любой DOI.

Протокол IKE включает четыре режима:

1) основной режим (main mode);

2) энергичный режим (aggressive mode);

3) быстрый режим (quick mode);

4) режим перехода к новой группе (new group mode).

Основной и энергичный режимы используют в первой фазе, быстрый режим – во второй фазе. Режим перехода к новой группе для использования в протоколе Диффи — Хеллмана является исключением: он не выполняется ни на первой, ни на второй фазе обмена, а может использоваться только один раз при установлении защищенной ассоциации ISAKMP SA.

Фаза 1 (выбор ISAKMP SA): основной или энергичный режим. В первой фазе протокола IKE происходит выбор следующих алгоритмов: шифрования, функции хэширования, метода аутентификации, а также выбор группы для алгоритма Диффи — Хеллмана. При этом в конце первой фазы должны быть сгенерированы три ключа: для шифрования, аутентификации сторон и порождения других ключей. Эти ключи зависят от значений специальных системных идентификаторов (cookies),

случайных чисел (nonces) и открытых значений или общего секрета из протокола Диффи — Хеллмана. При их вычислении используют функции хэширования, определенные в протоколе ISAKMP SA и зависящие от выбранного способа аутентификации (более подробно см. RFC 2409).

В первой фазе можно использовать основной или энергичный режим. Основной режим является реализацией режима обмена с защитой идентификаторов протокола ISAKMP (см. выше). Он состоит из шести сообщений, первые два позволяют сторонам договориться о SA, следующие два — об обмене открытыми значениями алгоритма Диффи — Хеллмана, последние два — об аутентификации этих открытых значений.

Первые два сообщения позволяют договориться о параметрах протокола IKE: алгоритмах шифрования, функции хэширования, методе аутентификации и выборе группы для алгоритма Диффи — Хеллмана. Четыре возможных метода аутентификации — это цифровая подпись, два способа аутентификации с использованием открытых ключей и использование предварительно распределенного ключа. Выбор метода влияет только на вид сообщений и метод генерации сеансовых ключей. Два следующих сообщения позволяют выбрать общий секрет с помощью протокола Диффи — Хеллмана. Общий секрет используется для генерации сеансовых ключей, два из них — для защиты последующего обмена путем использования шифрования и хэширования, выбранных при предыдущем обмене. Последние два сообщения осуществляют аутентификацию предыдущих сообщений, прежде всего открытых значений для протокола Диффи – Хеллмана.

Энергичный режим протокола IKE является реализацией энергичного режима протокола ISAKMP и состоит только из трех сообщений (см. выше).

Фаза 2 (выбор защищенных ассоциаций для механизмов защиты AH и ESP): быстрый режим. Все сообщения на второй фазе защищены (аутентифицированы и конфиденциальны) благодаря ключам, выработанным в первой фазе. Аутентичность сообщений проверяют добавлением вложения типа HASH в сообщение после ISAKMP-заголовка, а конфиденциальность обеспечивают шифрованием всех вложений в сообщения.

Быстрый режим используют для выбора защищенных ассоциаций для протоколов, аналогичных IPsec. Каждая договоренность приводит к двум защищенным ассоциациям, по одной для

каждого направления. Более точно, сообщения, составляющие этот обмен, выполняют следующие функции:

- выбор множества параметров IPsec (связка SA);
- обмен случайными числами, используемыми для генерации новых сеансовых ключей, которые формируют с помощью общего секрета, выработанного на фазе 1 с использованием протокола Диффи — Хеллмана; дополнительно можно включить новый обмен протокола Диффи — Хеллмана для обеспечения свойства PFS (этого нельзя сделать при генерации нового ключа из старого и случайных значений);
- идентификация трафика, защищаемого данной связкой SA.

В терминах вложений он выглядит следующим образом:

1) $A \rightarrow B$: HDR*, HASH, SA, NONCE [,KE], [,ID_A, ID_B];
2) $A \leftarrow B$: HDR*, HASH, SA, NONCE [,KE], [,ID_A, ID_B];
3) $A \rightarrow B$: HDR*, HASH.

Режим перехода к новой группе. Группа в алгоритме Диффи — Хеллмана может быть выбрана либо с помощью вложения вида SA при выполнении основного режима, либо позднее с помощью режима выбора новой группы. В обоих случаях используются два способа выбора группы:

1) ссылкой на одну из представленных в описании протокола OAKLEY групп;

2) заданием характеристик новой группы: типа группы (MODP, ECP, EC2N), простого числа или неприводимого полинома, образующих и т. п.

В настоящее время разработка протоколов управления ключами активно продолжается. В частности, в 2002 г. опубликованы предварительные версии (drafts) второй версии протокола IKEv2, профиль безопасности для протоколов ISAKMP и PKIX, протокол SIGMA — улучшенная версия протокола Photuris, совместно использующего цифровую подпись (SIGN) с аутентификацией (MAC) и др.

11.3. Атаки на протокол IPsec

После появления в 1995 г. первой версии протокола IPsec опубликовано несколько работ, где указываются возможные слабости процедуры шифрования ESP в случае, если она применяется без аутентификации.

Так, в работе [28] указаны возможные уязвимости режима CBC и их использования для построения атак на конфиденциальность сообщений, передаваемых с использованием протоко-

лов UDP и TCP. Дело в том, что при шифровании начальный блок открытого текста переходит в начало шифрованного текста, причем первый блок определяется вектором инициализации, который передается в открытом виде вместе с сообщением. Поскольку при инкапсуляции пакета транспортного уровня в пакет IP в начале открытого текста будет стоять заголовок пакета UDP или TCP, причем он имеет стандартную структуру, то можно изменять поля (адрес, номер пакета, длину сообщения, контрольную сумму) в заголовке путем искажения или повтора ранее переданных пакетов либо путем замены вложений на переданные ранее, тем самым навязывая информацию или осуществляя захват (соединения) сеанса (session hijacking) путем подмены одной из сторон информационного обмена. Приведены также некоторые слабости протокола обмена ключами SKIP, связанные со сложностью автоматической замены долговременного ключа.

В работе [28] данные идеи развиваются в отношении метода дешифрования, использующего вероятные открытые тексты. Дело в том, что перебор ключей при использовании алгоритма DES может быть сокращен, если угадан весь (или часть) блока отрытого текста. Это оказывается особенно эффективным, если данный блок открытого текста соответствует не одному, а нескольким открытым сообщениям. Вместе с тем эта ситуация оказывается достаточно типичной при инкапсуляции пакетов транспортного уровня и особенностей строения заголовков этих пакетов. В работе [62] анализируется безопасность протокола IPsec относительно класса атак, известных как IV-атаки, основанных на модификации вектора инициализации (IV) для передаваемых пакетов, зашифрованных в режиме CBC. Показано, что при неправильном использовании протокола эта атака является серьезной. Обсуждаются особенности применения этих атак для протоколов транспортного уровня UDP и TCP, используемых совместно с протоколом IPv4 или IPv6, а также с протоколом туннелирования L2TP, пакеты которого инкапсулируются в протокол UDP.

Все эти результаты показывают, что использование шифрования без аутентификации в режиме ESP может приводить к различным слабостям. Поэтому использование аутентификации в этом режиме должно быть обязательным.

Первоначальная версия протокола IPsec обладала большей слабостью к атаке с использованием повторной передачи в случае, если протокол IKE конфигурирован с использованием пред-

варительно распределенного ключа, чем при стандартной конфигурации с использованием сертификатов. Однако применение техники использования nonses в более поздних версиях IPsec позволяет обнаруживать и предотвращать подобные атаки.

Приведем в качестве примера одну из атак на протокол IKEv2. Это протокол аутентификации и открытого распределения ключей. Он применяется до установления соединения по протоколу IPsec. Протокол IKEv2 используется в двух вариантах в зависимости от метода аутентификации: с цифровой подписью или с хеш-функцией, задаваемой ключом.

Рассмотрим вариант протокола IKEv2-DS, основанный на использовании цифровой подписи.

Протокол IKEv2-DS выполняется в два этапа. Сначала на этапе IKE_SA_INIT участники обмениваются случайными значениями nonces и выполняют алгоритм Диффи—Хеллмана, устанавливая начальную защищенную ассоциацию, называемую IKE_SA. На втором этапе IKE_SA_AUTH осуществляется аутентификация предыдущих сообщений, обмен идентификаторами пользователей и устанавливается первая защищенная ассоциация (child security association, или CHILD_SA), которая будет в дальнейшем использоваться для подходящего туннеля IPsec. Участник A (B) генерирует случайное число r_A (r_B) и секретные значения x (y) для алгоритма Диффи—Хелмана. Кроме того, в поле $SAa1$ содержатся предложения A по выбору криптографического алгоритма, а в поле $SAb1$ — соответствующий выбор B для установления защищенной ассоциации IKE_SA. Аналогично, поля $SAa2$ и $SAb2$ предназначены для установления защищенной ассоциации CHILD_SA:

- IKE_SA_INIT:
  1) $A \to B \quad SAa1, \alpha^x \bmod p, r_A$;
  2) $A \leftarrow B \quad SAb1, \alpha^y \bmod p, r_B$;
- IKE_SA_AUTH:
  3) $A \to B \quad E_k(A, \mathrm{AUTH}a, SAa2)$;
  4) $A \leftarrow B \quad E_k(B, \mathrm{AUTH}b, SAb2)$,

где

$$k = h(r_A, r_B, SAa1, \alpha^{xy} \bmod p),$$
$$\mathrm{AUTH}a = D_{sk_A}(SAa1, \alpha^x \bmod p, r_A, r_B),$$
$$\mathrm{AUTH}b = D_{sk_B}(SAb1, \alpha^y \bmod p, r_A, r_B).$$

Заметим, что поскольку мы не учитываем возможность переговоров по выбору защищенной ассоциации, то полагаем, что $SAa1 = SAb1$ и $SAa2 = SAb2$, т. е. участник A только один раз

направляет свои предложения, а участник B сразу их принимает.

Для данного варианта протокола IKEv2 с помощью AVISPA найдена атака, в которой нарушитель C использует технику «противник в середине». Нарушитель C убеждает участника B, что он связан с A, в то время как на самом деле участник A в этом сеансе вообще не участвует. Нарушитель просто передает сообщения из разных сеансов с A, в которых участник A знает, что он связывается с участником C:

0) $A \leftarrow C$ start;
1) $A \rightarrow C$ $SA1, \alpha^x \bmod p, r_A$;
1′) $C(A) \rightarrow B$ $SA1, \alpha^x \bmod p, r_A$;
2′) $C(A) \leftarrow B$ $SA1, \alpha^y \bmod p, r_B$;
2) $A \leftarrow C$ $SA1, \alpha^y \bmod p, r_B$;
3) $A \rightarrow C$ $E_k(a, D_{sk_A}(SA1, \alpha^x \bmod p, r_A, r_B), SA2)$;
3′) $C(A) \rightarrow B$ $E_k(a, D_{sk_A}(SA1, \alpha^x \bmod p, r_A, r_B), SA2)$;
4′) $C(A) \leftarrow B$ $E_k(b, D_{sk_B}(SA1, \alpha^y \bmod p, r_B, r_A), SA2)$,

где $k = h(r_A, r_B, SA1, \alpha^{xy} \bmod p)$.

Эффективность данной атаки проблематична, так как нарушитель не может узнать ключ, который участник B формирует для связи с A. Поэтому конфиденциальность ключа не нарушается. Данная атака может быть исключена введением подтверждения правильности получения ключа. В реальном протоколе IKEv2-DSX введен дополнительный обмен сообщениями, зашифрованными на выработанном ключе, поэтому здесь эта атака невозможна.

В общем случае для современных версий протокола IPsec, использующих протокол управления ключами IKE, не известны факты успешного применения каких-либо атак, позволяющих осуществлять эффективное вмешательство в работу протокола.

Контрольные задания

1. Каково назначение протокола IPsec?
2. Охарактеризуйте понятие «защищенная ассоциация».
3. Объясните назначение и опишите структуру документов, описывающих протокол IPsec.
4. Объясните назначение документа IPsec DOI.
5. Каково назначение протокола ISAKMP?
6. Сравните свойства протоколов SKIP, Photuris, SKEME, OAKLEY. Какие особенности этих протоколов использованы при разработке протокола IKE? Объясните назначение фаз и режимов протокола IKE.
7. Предложите вариант двусторонней атаки с неизвестным общим ключом на протокол SKEME.

Управление ключами

12.1. Проблема управления ключами

Рассмотрим вопросы управления ключами, которые решают в основном организационными мерами. Поскольку ключи играют в криптографических системах очень важную роль, порядок обращения с ними во многом определяет надежность системы. Поэтому, следуя пособиям [9, 64], остановимся кратко на некоторых задачах, связанных с возникающими здесь проблемами.

Управление ключами состоит из процедур, обеспечивающих:

- включение пользователей в систему;
- выработку, распределение и введение в аппаратуру ключей;
- контроль за использованем ключей;
- смену и уничтожение ключей;
- архивирование, хранение и восстановление ключей.

Управление ключами играет важнейшую роль в криптографии как основа для обеспечения конфиденциальности обмена информацией, идентификации и целостности данных. Основным свойством хорошо спроектированной системы управления ключами является сведение сложных проблем обеспечения безопасности многочисленных ключей к проблеме обеспечения безопасности нескольких ключей, которая может быть относительно просто решена путем обеспечения их физической изоляции в выделенных помещениях и защищенном от проникновения оборудовании.

Целью управления ключами является нейтрализация таких угроз, как:

- компрометация конфиденциальности секретных ключей;
- компрометация аутентичности секретных или открытых ключей, т. е. использование ключей не теми, для кого они предназначены;
- несанкционированное использование секретных или открытых ключей, например использование ключа, срок действия которого истек.

Управление ключами обычно осуществляется в контексте определенной политики безопасности. Политика безопасности прямо или косвенно определяет те угрозы, которым должна противостоять система. Кроме того, она определяет:

- правила и процедуры, которыми необходимо руководствоваться и которые необходимо выполнять в процессе автоматического или ручного управления ключами;
- ответственность и подотчетность всех субъектов, участвующих в управлении;
- все виды записей, которые должны сохраняться для подготовки необходимых сообщений и проведения проверки действий, связанных с безопасностью ключей.

Одним из инструментов, используемых для обеспечения конфиденциальности ключей, является разделение ключей по уровням.

Главный ключ — высший ключ в иерархии, который не защищается криптографически. Его защиту осуществляют с помощью физических или электронных средств.

Ключи для шифрования ключей — секретные или открытые ключи, используемые для засекречивания перед передачей или при хранении других шифровальных ключей. Эти ключи сами могут быть зашифрованы с помощью других ключей.

Ключи для шифрования данных используют для защиты данных пользователей.

Ключи более высоких уровней используют для защиты ключей или данных на более низких уровнях, что уменьшает ущерб при компрометации ключей и снижает объем ключевой информации, нуждающейся в физической защите.

Одной из важных характеристик системы управления ключами являются сроки действия ключей. Под *сроком действия ключа* понимают интервал времени, в течение которого он может быть использован законным пользователем.

Сокращение сроков действия ключей необходимо для ограничения:

- объема информации, зашифрованной на данном ключе, которая может быть использована для криптографического анализа;
- размера ущерба при компрометации ключей;
- объема машинного времени, которое может быть использовано для криптографического анализа (в случае если отсутствуют требования по длительному сохранению ключа и информации, защищенной на этом ключе).

С учетом срока действия ключей в дополнение к указанной выше классификации ключей по уровням может быть введена следующая классификация:

1) ключи с длительным сроком действия; к ним относится главный ключ, часто — ключи для шифрования ключей;

2) ключи с коротким сроком действия; к ним относятся ключи для шифрования данных.

Как правило, в телекоммуникационных приложениях используют ключи с коротким сроком действия, а для защиты хранимых данных — ключи с длительным сроком действия.

Следует иметь в виду, что термин «короткий срок действия» относится только к сроку действия ключа, а не к промежутку времени, в течение которого ключ должен оставаться в секрете. Например, к ключу, используемому для шифрования в течение только одного сеанса связи, часто предъявляют требование, чтобы зашифрованная с его помощью информация не могла быть вскрыта на протяжении нескольких десятков лет. В то же время код аутентичности сообщения проверяется немедленно после передачи сообщения, поэтому ключ кода аутентификации в данном случае может сохраняться в тайне в течение короткого срока.

12.2. Жизненный цикл ключей

Ключевая информация должна быть сменена до момента истечения срока действия ключа. Для этого может быть использована действующая ключевая информация, протоколы распределения ключей и ключевые уровни (см. выше).

Для того чтобы ограничить ущерб от компрометации ключей, по возможности следует избегать зависимости между действующей и устанавливаемой ключевой информацией. Например, не рекомендуется защищать очередной сеансовый ключ с помощью действующего сеансового ключа. При хранении секретных ключей должны быть приняты меры по обеспечению их конфиденциальности и аутентичности. При хранении открытых ключей должны быть приняты меры, позволяющие проверить их аутентичность. Конфиденциальность и аутентичность могут быть обеспечены криптографическими, организационными и техническими мерами.

В зависимости от конкретных приложений могут выдвигаться различные требования к необходимости и длительности хранения используемых ключей. Например, открытые ключи, ис-

пользуемые для проверки цифровой подписи, в ряде случаев необходимо хранить длительное время, даже возможно дольше, чем соответствующие секретные ключи, чтобы была возможность проверки отдельных подписей. В то же время во многих приложениях секретные ключи цифровой подписи не следует хранить длительное время, а тем более архивировать их, поскольку попадание их к посторонним лицам может повлечь отказ от подписанного документа со стороны владельца ключа подписи. Проблема отсутствия секретного ключа цифровой подписи из-за его преждевременного уничтожения (без компрометации) сравнительно легко может быть решена путем генерации нового ключа, поскольку уничтоженный ключ не требуется для проверки ранее произведенных подписей. Аналогично этому открытые ключи, используемые для засекречивания информации, не нуждаются в архивировании. В то же время секретные ключи, используемые для расшифрования, должны определенное время храниться, поскольку в противном случае засекреченная информация будет утрачена.

Все криптосистемы за исключением простейших, в которых используемые ключи зафиксированы раз и навсегда, нуждаются в периодической замене ключей. Эта замена проводится с помощью определенных процедур и протоколов, в ряде которых используются и протоколы взаимодействия с третьей стороной. Далее приведена последовательность стадий, которые проходят ключи от момента установления до следующей замены, называемая *жизненным циклом ключей.*

1. *Регистрация пользователей.* Эта стадия включает обмен первоначальной ключевой информацией, такой как общие пароли или PIN-коды, путем личного общения или пересылки через доверенного курьера.

2. *Инициализация.* На этой стадии пользователь устанавливает аппаратное оборудование и (или) программные средства в соответствии с установленными рекомендациями и правилами.

3. *Генерация ключей.* При генерации ключей должны быть приняты меры по обеспечению их необходимых криптографических качеств. Ключи могут генерироваться как самостоятельно пользователем, так и специальным защищенным элементом системы, а затем передаваться пользователю по защищенному каналу.

4. *Установка ключей.* Ключи устанавливают в оборудование тем или иным способом. При этом первоначальная ключевая информация, полученная на стадии регистрации пользователей,

может либо непосредственно вводиться в оборудование, либо использоваться для установления защищенного канала, по которому передается ключевая информация. Эта же стадия используется в последующем для смены ключевой информации.

5. *Регистрация ключей.* Ключевая информация связывается регистрационным центром с именем пользователя и сообщается другим пользователям ключевой сети. При этом для открытых ключей сертификационным центром создаются ключевые сертификаты, и эта информация публикуется тем или иным способом.

6. *Обычный режим работы.* На этой стадии ключи используют для защиты информации в обычном режиме.

7. *Хранение ключа.* Эта стадия включает процедуры, необходимые для хранения ключа в надлежащих условиях, обеспечивающих его безопасность до момента его замены.

8. *Замена ключа.* Данная стадия осуществляется до истечения срока действия ключа и включает процедуры, связанные с генерацией ключей, протоколами обмена ключевой информацией между корреспондентами, а также с доверенной третьей стороной. Для открытых ключей на этой стадии обычно проводят обмен информацией по защищенному каналу с сертификационным центром.

9. *Архивирование.* В отдельных случаях ключевая информация после ее использования для защиты информации может быть подвергнута архивированию для ее извлечения в специальных целях (например, при рассмотрении вопросов, связанных с отказами от цифровой подписи).

10. *Уничтожение ключей.* После окончания сроков действия ключей их выводят из обращения, и все имеющиеся копии ключей уничтожают. При этом необходимо следить, чтобы в случае уничтожения секретных ключей тщательно уничтожалась и вся информация, по которой возможно их частичное восстановление.

11. *Восстановление ключей.* Если ключевая информация уничтожена, но не скомпрометирована (например, из-за неисправности оборудования или из-за того, что оператор забыл пароль), должны быть предусмотрены меры, дающие возможность восстановить ключ из хранимой в соответствующих условиях его копии.

12. *Отмена ключей.* В случае компрометации ключевой информации возникает необходимость прекращения использования ключей до окончания срока их действия. При этом должны быть предусмотрены необходимые меры оповещения абонентов

сети. При отмене открытых ключей, снабженных сертификатами, одновременно прекращается действие сертификатов.

12.3. Услуги, предоставляемые доверенной третьей стороной

В жизненном цикле управления ключами важную роль играет так называемая доверенная третья сторона. Перечислим функции доверенной третьей стороны.

Сервер имен абонентов обеспечивает придание каждому из абонентов индивидуального имени.

Регистрационный центр обеспечивает включение каждого из абонентов в данную сеть засекреченной связи и выдачу ему соответствующей ключевой информации.

Центр производства ключей обеспечивает изготовление (генерацию) необходимого количества ключей и доставку их участникам с помощью протоколов распределения ключей.

Ключевой (идентификационный) сервер обеспечивает установку общего сеансового ключа между двумя абонентами путем передачи этого ключа по защищенному каналу, образуемому сервером с каждым из абонентов. При этом может осуществляться и идентификация абонентов.

Центр управления ключами обеспечивает хранение, архивирование, замену и отмену ключей, а также аудит действий, связанных с жизненным циклом ключей.

Сертификационный центр обеспечивает аутентичность открытых ключей путем придания им сертификатов, заверенных цифровой подписью.

Центр установки временны́х меток обеспечивает привязку временно́й метки к электронному сообщению или транзакции, заверяя тем самым их наличие в определенный момент времени.

Центр нотаризации обеспечивает невозможность отказа от сделанного в определенный момент заявления, зафиксированного в электронной форме.

12.4. Особенности управления ключами в симметричных системах шифрования

Использование симметричных систем шифрования при построении сетевых протоколов связано, как правило, с необходимостью использования большого числа различных ключей. Это

объясняется не только большим числом абонентов сети, но и необходимостью разделения ключей:

- по уровням;
- срокам действия;
- функциональному предназначению и т. п.

При этом многие ключи имеют достаточно длительные сроки действия или хранения. Для минимизации объема хранимой ключевой информации применяют шифрование одних ключей на другие с использованием определенной иерархии на множестве ключей. В результате получают достаточно большие базы данных, в которых накапливаются значительные объемы зашифрованной ключевой информации. Секретность всех баз данных обеспечивает один главный ключ, для защиты которого нужно принять специальные меры. Хранение больших ключевых баз данных во многих случаях может оказаться нецелесообразным как с точки зрения удобства работы, так и с точки зрения безопасности.

Чтобы избежать хранения большого количества ключевой информации у участников протокола, применяют специальные приемы.

Как отмечалось выше, для уменьшения объема ключевой информации у обычных абонентов сети:

- используют схемы предварительного распределения ключей;
- наделяют отдельных участников функциями доверенной третьей стороны; они осуществляют централизованное управление ключами и др.

Особенно остро эта проблема стоит для центров, выполняющих роль доверенной третьей стороны, поскольку там должны храниться ключи всех остальных участников обмена.

Сертификаты секретных ключей. Чтобы не хранить в центре зашифрованные ключи остальных участников, можно использовать так называемые *сертификаты секретных ключей.* Они имеют следующий вид:

$$\mathrm{scert}_A = E_{k_T}(k_{AT}, A, t),$$

где A, T — идентификаторы участника и центра соответственно; k_T — ключ центра; k_{AT} — общий ключ участника A и центра T; t — срок действия или другая дополнительная информация.

Такой сертификат может постоянно храниться только у участника A. В случае необходимости выполнения протокола

между участником A и центром T он просто вставляется в сообщение, отправляемое участником A центру T

Для хранения общедоступной базы данных сертификатов секретных ключей не нужно принимать специальных мер для связывания идентификаторов с сертификатами секретных ключей, так как идентификатор содержится внутри сертификата.

С применением таких сертификатов протокол конфиденциальной передачи сообщения M от участника A участнику B через центр перешифрования T может выглядеть следующим образом:

1) $A \to T$ $\text{seert}_A, E_{k_{AT}}(B, M), \text{scert}_B$;
2) $A \leftrightarrow T$ $E_{k_{BT}}(M, A)$;
3) $A \to B$ $E_{k_{BT}}(M, A)$.

Заметим, что данный протокол обеспечивает защиту идентификаторов, так как они передаются внутри зашифрованных сообщений. Тем самым выполняется свойство несвязываемости, важное для обеспечения анонимности отправителя. Атака отражением невозможна, так как в прямом и обратном сообщениях переставлены поля. Для защиты от повтора можно вставлять в сообщения временные метки или использовать случайные последовательности.

Использование ключей, зависимое от назначения. Еще один прием, позволяющий сократить число используемых ключей, состоит в применении принципа, зависимого от назначения управления ключами. Прием заключается в том, что один и тот же ключ применяют в различных ситуациях по-разному. Для этого используют следующие приемы:

- дополнение ключа;
- использование открытых масок;
- применение контекстных векторов.

При дополнении к ключу дописывают вектор, сформированный по некоторому правилу в зависимости от данного конкретного применения. Данный способ нежелателен, так как может приводить к реальному сокращению ключевого пространства.

При использовании маски v_i из набора $v_1, \ldots, v_t$ ключ k преобразуется к виду $k \oplus v_i$. Такой способ получения различных вариантов ключа применяют в системах защиты главных ключей (см. гл. 10). Маски могут храниться в открытом виде.

Наиболее интересен способ, основанный на введении контрольных векторов; при этом при шифровании используют не сам ключ, а его сумму с контрольным вектором $k \oplus cv$. Если дли-

на контрольного вектора отличается от длины ключа, то можно применить хеш-функцию: $k \oplus h(cv)$. Контрольный вектор можно определять по некоторому правилу в зависимости от назначения (шифрование, код аутентификации, цифровая подпись, период действия ключа, регион, адресат и т. д.). В результате каждый ключ можно использовать неоднократно, причем каждый раз благодаря изменению контекста реально будут применяться различные ключи. Для обратного преобразования нужно знать контекст. Поэтому он может храниться вместе с сообщением либо для исключения возможности манипуляции — вычисляться автоматически при каждом применении.

12.5. Особенности управления ключами в асимметричных системах шифрования

Использование в сетевых протоколах асимметричных систем шифрования избавляет от необходимости хранения большого числа различных ключей. Вместе с тем здесь остро стоит проблема связывания идентификаторов с открытыми ключами участников. Для решения этой проблемы применяют сертификаты открытых ключей, а для управления сертификатами создают инфраструктуру открытых ключей, обеспечивающую процесс их выдачи, проверки и подтверждения подлинности, а также отзыва в случае необходимости.

Как отмечалось выше, для надежной проверки и обслуживания сертификатов создают удостоверяющие центры, выполняющие все функции по управлению сертификатами, либо центры с разделенными функциями: центры сертификации и центры регистрации. При большом числе центров между ними устанавливаются отношения доверия. Для этого в зависимости от ситуации можно использовать древовидную иерархическую структуру, в которой вышестоящий центр заверяет нижестоящих, либо сетевую структуру, в которой используют либо кольцевые цепочки однонаправленных сертификатов, либо двунаправленные двусторонние взаимные заверения открытых ключей между каждой парой связанных центров — кросс-сертификаты.

При использовании инфраструктур открытых ключей также возникают определенные проблемы.

1. Получение участником сертификата. Возможны два варианта: участник сам генерирует пару открытый ключ/секретный ключ и затем получает сертификат открытого ключа, либо генерацию и сертификацию ключей осуществляет центр.

В первом варианте, прежде чем выдать сертификат, центр должен убедиться, что данный открытый ключ не был ранее связан ни с одним из других участников, а также проверить подлинность идентификационных данных участника.

Второй вариант генерации ключей центром в этом смысле более надежен, но представляет большие неудобства при удаленном взаимодействии центра и участника, так как для передачи секретной половины ключа необходим закрытый канал.

2. Отзыв сертификатов. Если проверка сертификатов осуществляется удостоверяющим центром в реальном режиме (on-line), то отозванные сертификаты сразу выводят из оборота. В случае отложенного режима (off-line) удостоверяющему центру приходится создавать списки отозванных сертификатов и периодически рассылать их адресатам. Список отозванных сертификатов представляет собой подписанный центром документ, в котором указывают открытые ключи, серийные номера отозванных сертификатов, время отзыва и другую информацию. При большом числе сертификатов такая рассылка затруднительна. Поэтому необходимо прибегать к специальным приемам, например:

- проводить сегментирование такого списка (это можно сделать, если множество сертификатов можно разбить по независимым группам пользователей либо по областям применения);
- осуществлять рассылку только вновь отозванных сертификатов (в предположении, что участники накапливают эти списки);
- вводить правила, определяющие периодичность обязательного обращения к этим спискам.

3. Большое число сертификатов у одного участника. Как и для симметричных систем шифрования, в данном случае часто возникают ситуации, в которых у участника оказывается достаточно большое число m открытых ключей, предназначенных для различных целей. Если значение m очень велико, то при обычном порядке регистрации и получения сертификатов удостоверяющий центр должен обеспечить сертификатами все открытые ключи участника, что может привести к излишне большим размерам списка хранимых в центре сертификатов.

Есть простой способ сведения данной задачи к регистрации только одного сертификата для каждого участника. Он основан на применении бинарных деревьев, вершины которых помечены значениями хеш-функции. Суть его такова. Пусть участник име-

ет m значений $P_1, \quad , P_m$, для которых должна быть установлена аутентичность. Строим бинарное дерево глубиной $\lceil \log_2 m \rceil$ с m листьями. Теперь нумеруем листья $1, \ldots, m$ и помечаем i-й лист значением $h(P_i)$, $1 \leq i \leq m$. Далее индуктивно переходим на более высокие уровни, поднимаясь к корню дерева, и помечаем каждую вершину значением $h(x||y)$, где x, y — метки входящих в нее левой и правой ветвей. В итоге получаем в удостоверяющем центре сертификат на хеш-значение, соответствующее корню дерева.

Для проверки аутентичности значения P_i нужно выписать последовательно метки вершин, смежных с вершинами, лежащими на пути от i-го листа к корню дерева, а затем вычислить значение $h(P_i)$ и соответствующие значения для всех вершин, лежащих на этом пути. Если в результате для корня получается значение, содержащееся в сертификате, то проверка прошла успешно.

Неудобство такого подхода заключается в необходимости добавления новых ветвей с пересчетом соответствующх значений и получении нового сертификата в случае увеличения числа m.

Контрольные задания

1. В чем состоят цели управления ключами?
2. В чем состоит политика безопасности?
3. Как могут быть классифицированы ключи в зависимости от предназначения и сроков их действия?
4. Какие стадии включает жизненный цикл ключей?
5. Какие услуги могут быть предоставлены доверенной третьей стороной?

СПИСОК ЛИТЕРАТУРЫ

1. *Агибалов Г. П.* Избранные теоремы начального курса криптографии учеб. пособие / Г. П. Агибалов. — Томск Изд-во НТЛ, 2005.

2. *Березин Б. В.* Цифровая подпись на основе традиционной криптографии / Б. В. Березин, П. В. Дорошкевич // Защита информации. — 1992. — № 2. — С. 148 — 167.

3. *Деднев М. А.* Защита информации в банковском деле и электронном бизнесе / М. А. Деднев, Д. В. Дыльнов, М. А. Иванов. — М. КУДИЦ-ОБРАЗ, 2004.

4. *Запечников С. В.* Криптографические протоколы и их применение в финансовой и коммерческой деятельности учеб. пособие для вузов / С. В. Запечников. — М. Горячая линия — Телеком, 2007.

5. *Зубов А. Ю.* Математика кодов аутентификациии / А. Ю. Зубов. — М. Гелиос АРВ, 2007.

6. Интеллектуальные карты и криптографические особенности их применений в банковском деле учеб. пособие / [А. А. Варфоломеев, С. В. Запечников, В. В. Маркелов, М. Б. Пеленицын]. — М. МИФИ, 2000.

7. *Косачев А. С.* Анализ подходов к верификации функций безопасности и мобильности / А. С. Косачев, В. Н. Пономаренко. — М. Ин-т системного программирования РАН, 2004.

8. Криптография в банковском деле / [М. И. Анохин, Н. П. Варновский, В. М. Сидельников, В. В. Ященко]. — М. МИФИ, 1997.

9. Основы криптографии учеб. пособие / [А. П. Алферов, А. Ю. Зубов, А. С. Кузьмин, А. В. Черемушкин]. — 3-е изд., доп. — М. Гелиос АРВ, 2005.

10. *Саломаа А.* Криптография с открытым ключом / А. Саломаа. — М. Мир, 1996.

11. Словарь криптографических терминов / под ред. Б. А. Погорелова, В. Н. Сачкова. — М. МЦНМО, 2006.

12. *Шнайер Б.* Прикладная критография. Протоколы, алгоритмы, исходные тексты на языке Си / Б. Шнайер. — М. ТРИУМФ, 2002.

13. *Armando A.* SAT-based Model-Checking of Security Protocols Using Planning Graph Analysis / A. Armando, L. Compagna, P. Ganty // Proc. 12th Int. Formal Methods Europe Symp. (FME'03). — Pisa, 2003. — Lecture Notes in Computer Science. — 2003. — V 2805. — P. 875 — 893.

14. *Armando A.* SATMC a SAT-Based Model Checker for Security Protocols / A. Armando, L. Compagna // Proc. 9th Eur. Conf. on Logics

in Artificial Intelligence (JELIA'04). — Lisabon, 2004. — Lecture Notes in Artifician Inteligence. — 2004. — V 3229. — P. 730 — 733.

15. *Asokan N.* Asynhronous Protocols for Optimisic Exchange / N. Asokan, V. Shoup, M. Waidner // Proc. IEEE Symp. on Research in Security and Privacy. — 1998. — P. 86 — 99.

16. Automated Validation of Internet Security Protocols and Applications (AVISTA). IST-2001-39252. Deliverable 2.1 The High-Level Protocol Specification Language. — 2003. — http://www.avispa-project.org/publications.html.

17. Automated Validation of Internet Security Protocols and Applications (AVISTA). IST-2001-39252. Deliverable 2.3 The Intermediate Format. — 2003. — http://www. avispa-project.org/publications.html.

18. Automated Validation of Internet Security Protocols and Applications (AVISTA). IST-2001-39252. The AVISPA Tool. — 2005. — http://www.avispa-project.org/publications.html.

19. Automated Validation of Internet Security Protocols and Applications (AVISTA). IST-2001-39252. The AVISPA v.1.1 User Manual. — 2006. — http://www.avispa-project.org/ publications.html.

20. Automated Validation of Internet Security Protocols and Applications (AVISTA). IST-2001-39252. Deliverable D6.1 List of Selected Problems. — 2005. — http://www.avispa-project.org/publications.html.

21. Automated Validation of Internet Security Protocols and Applications (AVISTA). IST-2001-39252. Deliverable D6.2 Specification of the Problems in the High-Level Specification Language. — 2005. — http://www.avispa-project.org/publications.html.

22. *Backes M.* Compositional Analysis of Contract Signing Protocols / M. Backes, A. Datta, A. Derek // Proc. 18th IEEE Computer Security Foundations Workshop (CSFW-18). — Aix-en-Provence, 2005. — Theoretical Computer Science. — 2005. — P. 33 — 56.

23. *Baignères T.* A Classical Introduction to Cryptography. Exercise Book / T. Baignères, P. Junod, Li Yu. — N.Y. Springer Science ; Bisiness Medis, 2006.

24. *Basin D.* Constraint Differentiation a New Reduction Technique for Constraint-Based Analysis of Security Protocols / D. Basin, S. Modersheim, L. Vigan'o // Proc. Computer and Communication Security (CCS'03) / eds Vijay Atlury, Peng Liu. — ACM Press. — 2003. — P. 335 — 344. — http://www.avispa-project.org/publications.html.

25. *Basin D.* OFMC a Symbolic Model-Checker for Security Protocols / D. Basin, S. Modersheim, L. Vigan'o // Int. J. Information Security. — 2005. —V 4. — N 5. — P. 181 — 208.

26. *Bellare M.* Keying Hash Functions for Message Authentication / M. Bellare, R. Canetti, H. Krawchyk // Advances in Cryptology. — CRYPTO'96 / ed. N. Koblitz. — Lecture Notes in Computer Science. — 1996. — V 1109. — P. 1 — 15.

27. *Bellovin S. M.* Encrypted Key Exchange : Password Based Protocols Secure Against Dictionary Attacks / S. M. Bellovin, M. Merritt // Proc.

1992 IEEE Symp. on Research in Security and Privacy. — IEEE Computer Society. — 1992. — P. 72—84.

28. *Bellovin S. M.* Problem Areas for the IP Security Protocols // Proc. Sixth Usenix Unix Security Symp., 1996. — P. 1—16. — http://www.research.att.com/~smb/papers/badesp.ps.

29. *Bird R.* The KryptoKnight Family of Light-Weight Protocols for Authentication and Key Distribution / R. Bird, I. Gopal, A. Herzberg // IEEE/ACM Transactions on Networking. — 1995. — V 3. — N 1. — P. 31—41.

30. *Birrell A. D.* Secure Communication Using Remote Procedure Calls // ACM Transactions on Operating Systems. — 1985. — V 3. — N 1. — P. 1—14.

31. *Blake-Wilson S.* Authenticated Diffe—Hellman Key Agreement Protocols / S. Blake-Wilson, A. Menezes // Proc. ACM Symp. on Applied Computing (SAC'98), 1998 / eds S. Tavares, H. Meijer // Lecture Notes in Computer Science. — 1999. — V 1556. — P. 339—361.

32. *Blom R.* An Optimal Class of Symmetric Key Generation Systems // Advances in Cryptology. — EUROCRYPT'84. — Lecture Notes in Computer Science. — 1984. — V 209. — P. 335—338.

33. *Blom R.* Non-Public Key Distribution // Advances in Cryptology. — Proc. Eurocrypt'82. Plenum. — N.Y., 1983. — P. 231—236.

34. *Boyd C.* Protocols for Authentication and Key Establishment / C. Boyd, A. Mathuria. — Berlin : Springer-Verlag, 2004.

35. *Burmester M.* A Secure and Efficient Conference Key Distribution System / M. Burmester, Y. Desmedt // Advances in Cryptology. — EUROCRYPT'89. — Lecture Notes in Computer Security. — 1990. — V 434. — P. 122—133.

36. *Burmester M.* On the Risk of Opening Distributed Keys / M. Burmester // Advances in Cryptology. — CRYPTO'94. — Lecture Notes in Computer Security. — 1994. — V 839. — P. 308—317.

37. *Burrows M.* A Logic of Authentication / M. Burrows, M. Abadi, R. Needham // ACM Transactions on Computer Systems. — 1990. — V. 8. — N 1. — P. 18—36.

38. *Chen L.* Bilateral Unknown Key-Share Attacks in Key Agreement Protocols / L. Chen, Q. Tang. — Cryptology ePrint Archive. Report 2007/209. — http://www.iacr.org/eprint.

39. *Clark J.* A Survey of Authentication Protocol Literature: Version 1.0. 17 Nov. 1997 / J. Clark, J. Jacob. — http://www.cs.york.ac.uk/jac/papers/ drareview.ps.gz, 1997.

40. *Cohen E.* TAPS a First-Order Verifier for Cryptographic Protocols // Proc. 13th IEEE Computer Security Foundations Workshop (CSFW'13). — IEEE Computer Society Press. — 2000. — P. 144—158.

41. *Denning D. E.* Timestamps in Key Distribution Protocols / D. E. Denning, G. M. Sacco // Communications of the ACM. — 1981. — V. 24. — N 8. — P. 533—536.

42. *Diffie W.* Authentication and Authenticated Key Exchanges / W. Diffie, P. C. van Oorschot, M. J. Wiener // Designs, Codes and Cryptography. — 1992. — V 2. — P. 107—125.

43. *Diffie W.* New Directions in Cryptography / W. Diffie, M. E. Hellman // IEEE Transactions on Information Theory. — 1976. — V IT-22. — N 6. — P. 644—654.

44. *Dolev D.* On the Security of Publickey Protocols / D. Dolev, A. Yao // IEEE Transactions on Information Theory. — 1983. — V. IT-29. — N 2. — P. 198—208.

45. *Dyer M.* On Key Storage in Secure Networks / M. Dyer, T. Fenner, A. Frieze // J. Cryptology. — 1995. — V 8. — P. 189—200.

46. *Ehrsam W. F.* A Cryptographic Key Management Scheme for Implementing the Data Encryption Standard / W. F. Ehrsam, S. M. Matyas, C. H. Meyer // IBM Syst. J. — 1978. — V 17. — N 2. — P. 106—125.

47. *Genet T.* Reachability Analysis of Term Rewriting Systems with Timbuk / T. Genet, V Viet Triem Tong // Proc. 8th Int. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning. — Lecture Notes in Artificial Intelligence. — 2001. — V. 2250. — P. 695—706.

48. *Genet T.* Rewriting for Cryptographic Protocol Verification / T. Genet, F. Klay // Int. Conf. on Automated Deduction (CADE'17). — 2000. — Lecture Notes in Artificial Intelligence. — 2000. — V 1831. — P. 271—290. — http://siteseer.nj.nec.com/genet99rewriting.html.

49. *Gnesi S.* Towards Model Checking a Spi-Calculus Dialect / S. Gnesi, D. Latella, G. Lenzini // Istituto di Scienza e Tecnologie dell'Informazione «A. Faedo» (ISTI-CNR). — Pisa, 2002. — Technical report (2002-TR-10).

50. *Goldwasser S.* Probabilistic Encription / S. Goldwasser, S. Micali // J. Computer and System Science. — 1984. — V 28. — P. 270—299.

51. *Goubault-Larrecq J.* A Method for Automatic Cryptographic Protocol Verification // Fifth Workshop on Formal Methods for Parallel Programming Theory and Applications (FMPPTA'2000). — Lecture Notes in Computer Security. — 2000. — V 1800. — P. 977—984. — http://www.dyade.fr/fr/ actions/vip/publications.html.

52. *Heather J.* How to Prevent Type Flaw Attacks on Security Protocols / J. Heather, G. Lowe, S. Schneider // Proc. 13th IEEE Computer Security Foundations Workshop (CSFW'13). — 2000. — P. 255—268.

53. *Huima A.* Efficient Infinite-State Analysis of Security Protocols // Proc. Workshop on Formal Methods and Security Protocols (FLOC.99), 1999.

54. Index of the Security Protocols Repository (SPORE) // Laboratorie Spécification et Vérification. — http://www.lsv.ens-cachan.fr/spore/table.html.

55. *Kindred D.* Theory Generation for Security Protocols. — Doctor of Philosophy Thesis, 1999.

56. *Krawchyk H.* HMAC : Keyed-Hashing for Message Authentication / H. Krawchyk, M. Bellare, R. Canetti. — RFC 2104. — 1997.

57. *Labouret G.* A Technical Paper Giving an Overview of the IPsec Standard // Initial Version on 7 December 1998. Last revised on 16 Juin 2000. — http://www.hsc.fr/ressources/articles/ipsec-tech/ipsec-tech.pdf.

58. *Lenstra A.* Colliding X.509 Certificates / A. Lenstra, X. Wang, B. de Weger // Cryptology ePrint Archive. Report 2005/067. — http://www.iacr.org/eprint/

59. *Lim C.* A Key Recovery Attack on Discrete Log-Based Schemes Using a Prime Order Subgroup / C. Lim, P. Lee // Advances in Cryptology. CRYPTO'97. — Lecture Notes in Computer Security. — 1997. — V. 1294. — P. 249 — 263.

60. *Lowe G.* An Attack on the Needham — Schroeder Publick-Key Authentication Protocol // Information Proccesing Letters. — 1995. — V 56. — N 3. — P. 131 — 133.

61. *Matsumoto T.* On Seeking Smart Public-Key-Distribution Systems / T. Matsumoto, Y. Takashima, H. Imai // Trans. IECE of Japan. — 1986. — V E69. — P. 99 — 106.

62. *McCubbin C. B.* Initialization Vector Attacks on the IPsec Protocol Suite / C. B. McCubbin, A. A. Selcuk, Sidhu Deepinder // IEEE 9th Int. Workshops on Enabling Technologies Infrastructure for Collaborative Enterprises (WET ICE'00). — Gaithersburg, 2000. — http://csdl.computer.org/dl/proceedings/wetice/2000/0798/00/07980171.pdf

63. *Meadows C.* The NRL Protocol Analyzer : an Overview // J. Logic Programming. — 1996. — V 26. — N 2. — P. 113 — 131.

64. *Menezes A. J.* Handbook of Applied Cryptography / A. J. Menezes, P. C. van Oorschot, S. A. Vanstone — N.Y. CRC Press, 1997.

65. *Meyer C. H.* Cryptography a New Dimension in Computer Data Security. A Guide for the Design and Implementation of Secure Systems / C. H. Meyer, S. M. Matyas — N.Y. John Willey&Sons, 1982.

66. *Mitchell C. J.* Key Strorage in Secure Networks / C. J. Mitchell, F. C. Piper // Discrete Applied Math. — 1988. — V. 21. — P. 215 — 228.

67. *Molva R.* KryptoKnight authentication and key distribution syste / R. Molva, G. Tsudik, E. van Herreweghen // Proc. 2nd Eur. Symp. on Research in Computer Security (ESORICS'92). — Toulouse, 1992. — P. 155 — 174.

68. National Security Agency. SKIPJACK and KEA algorithm specification. Version 2.0. 29 May 1998. — http://csrc.nist.gov/encryption/skipjack-kea.htm

69. *Needham R. M.* Using Encryption for Authentication in Large Networks of Computers / R. M. Needham, M. D. Schroeder // Commun. ACM. — 1978. — V 21. — P. 993 — 999.

70. *Oehl F.* Automatic Approximation for the Verification of Cryptographic Protocols / F. Oehl, G. Cece, O. Kouchnarenko // Proc. Int. Conf. on Formal Aspects of Security (FASec). — London, 2003. — Lecture Notes in Computer Science. — V 2629. — P. 33 — 48.

71. *Orman H.* The OAKLEY Key Determination Protocol. — RFC 2412. — 1998.

72. *Otway D.* Efficient and Timely Mutual Authentication / D. Otway, O. Rees // Operating Systems Review. 1987. — V. 21. — P. 8 – 10.

73. *Paulson L. C.* Mechanized Proofs for a Recursive Authentication Protocol // Proc. 10th Computer Security Foundations Workshop (CSFW'10). — IEEE Computer Society Press. — 1997. — P. 84 — 95.

74. *Paulson L. C.* The Inductive Approach to Verifying Cryptographic Protocols // J. Computer Security. - 1998. — V. 6. — N 1 — 2. — P. 85 — 128.

75. *Preneel B.* Analysis and Design of Cryptographic Hash Functions. — Ph. D. Dissertation. — Katholieke Universiteit Leuven, 1993.

76. *Schneier B.* A Certified E-Mail Protocol / B. Schneier, J. Riordan // 13th Annual Computer Security Applications Conf. — 1998. — http://citeseer.ist.psu.edu/article/schneier98certified.html.

77. *Shamir A.* An Efficient Indentification Scheme Based on Permuted Kernels // Proc. CRYPTO'89. — Lecture Notes in Computer Scince. — 1990. — V 435. — P. 606 — 609.

78. *Shamir A.* How to Share a Secret // Commun. ACM. — 1979. — V 22. — N 11. — P. 612 — 613.

79. *Shoup V.* On Formal Models for Secure Key Exchange. Technical Report. IBM Research Report RZ 3120. 1998 // Cryptology ePrint Archive, Report 2000/012 (version 4). — 1999. — http://www.iacr.org/eprint/2000/ 012.

80. *Song D.* Athena : a Novel Approach to Efficient Automatic Security Protocol Analysis / D. Song, S. Berezin, A. Perrig // J. Computer Security. — 2001. — V 9. — N 1. — P. 47 — 74.

81. *Stevens M.* Fast Collision Attack on MD5 // Cryptology ePrint Archive. Report 2006/104. — http://eprint.iacr.org.

82. *Stinson D. R.* Combinatorial Characterizations of Authentication Codes // Designs, Codes and Cryptography. — 1992. — V 2. — P. 175 — 187

83. *Stinson D. R.* Cryptography Theory and Practice / D. R. Stinson. — N.Y. CRC Press, 1995.

84. *Stinson D. R.* Secure frameproof codes, key distribution patterns, group testing algorithms and related structures / D. R. Stinson, Tvan Trung, R. Wei // J. Statist. Plan. Infer. — 2000. — V 86. — N 2. — P. 595 — 617.

85. *Syverson P.* Formal Semantic for Logic of Cryptographic Protocols // Proc. Computer Security Foundations Workshop III. — IEEE Computer Society Press. — 1990. — P. 32 — 41.

86. *Talbot J.* Complexity and Cryptography. An Introduction / J. Talbot, D. Welsh. — N.Y. Cambridge University Press, 2006. — http://www.cambridge.org/9780521852319.

87. *Thayer F. F.* Strand Spaces Why is a Security Protocol Correct / F. F. Thayer, J. Herzog, J. Guttman // Proc. 1998 IEEE Symp. on Security and Privacy. — IEEE Computer Society Press. — 1998. — P. 160 — 171.

88. *Turuani M.* Securite des Protocoles Cryptographiques : Decidabilite et Complexite. — Ph. D., Universite Henri Poincare. — Nancy, 2003.

89. *Wang X.* Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD / X. Wang, D. Feng, X. Lai // CRYPTO 2004. Cryptology ePrint Archive, Report 2004/199, first version (August 16, 2004), second version (August 17, 2004). — http://eprint.iacr.org.

90. *Wang X.* How to Break MD5 and Other Hash Functions / X. Wang, H. Yu // Advances in Cryptology. — EUROCRYPT 2005. — Lecture Notes in Computer Security. — 2005. — V 3494. — P. 19—35. — http://www.infosec.sdu.edu.cn/paper/md5-attack.pdf.

91. *West C. H.* General Technique for Communications Protocol Validation // IBM J. Research and Development. — 1978. — V 22. — P. 393—404.

92. *Wilson S. B.* Authenticated Diffe—Hellman Key Agreement Protocols / S. B. Wilson, A. Menezes eds S. Tavares, H. Meijer // Proc. SAC'98. — Lecture Notes in Computer Security. — 1999. — V 1556. — P. 339—361.

93. *Woo T. Y. C.* A Lesson on Authentication Protocol Design / T. Y. C. Woo, S. S. Lam // Operating Systems Review. — 1994. — V. 28. — N 3. — P. 24—37.

94. *Woo T. Y. C.* Authentication for Distributed Systems / T. Y. C. Woo, S. S. Lam // Computer. — 1992. — V 25. — N 1. — P. 39—52.

ОГЛАВЛЕНИЕ