Высшее профессиональное образование

П. Б. Хорев

МЕТОДЫ И СРЕДСТВА ЗАЩИТЫ ИНФОРМАЦИИ В КОМПЬЮТЕРНЫХ СИСТЕМАХ

Учебное пособие

Информатика и вычислительная техника

ACADEMA

ВЫСШЕЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАНИЕ

П. Б. ХОРЕВ

МЕТОДЫ И СРЕДСТВА ЗАЩИТЫ ИНФОРМАЦИИ В КОМПЬЮТЕРНЫХ СИСТЕМАХ

Рекомендовано
Учебно-методическим объединением вузов
по университетскому политехническому образованию
в качестве учебного пособия для студентов высших учебных заведений,
обучающихся по направлению 230100 (654600)
«Информатика и вычислительная техника»

Москва
ACADEMIA
2005

УДК 681.3.067(075.8)
ББК 32.973-018.2я73
Х792

Рецензенты:
профессор кафедры вычислительных машин, систем и сетей МЭИ (ТУ),
д-р техн. наук *Ю. Н. Мельников*;
доцент кафедры вычислительных машин, комплексов,
систем и сетей МГТУ ГА, канд. техн. наук *А. И. Терентьев*

Хорев П. Б.

Х792 Методы и средства защиты информации в компьютерных системах: Учеб. пособие для студ. высш. учеб. заведений / Павел Борисович Хорев. — М.: Издательский центр «Академия», 2005. — 256 с.

ISBN 5-7695-1839-1

Основное внимание в учебном пособии уделено программным и программно-аппаратным методам и средствам защиты информации. Рассмотрены, в частности, модели безопасности, используемые в защищенных версиях операционной системы Windows (в том числе использование функций криптографического интерфейса приложений CryptoAPI) и операционных системах «клона» Unix.

Для студентов высших учебных заведений. Может быть полезно специалистам в области информационной безопасности.

ISBN 5-7695-1839-1

УДК 681.3.067(075.8)
ББК 32.973-018.2я73

ПРЕДИСЛОВИЕ

Проблема защиты информации: надежное обеспечение ее сохранности и установленного статуса использования — является одной из важнейших проблем современности.

Еще 25...30 лет назад задача защиты информация могла быть эффективно решена с помощью организационных мер (выполнения режимных мероприятий, использования средств охраны и сигнализации) и отдельных программно-аппаратных средств разграничения доступа и шифрования. Этому способствовала концентрация информационных ресурсов и средств для их обработки на автономно функционирующих вычислительных центрах. Появление персональных ЭВМ, локальных и глобальных компьютерных сетей, спутниковых каналов связи, эффективных средств технической разведки и получения конфиденциальной информации существенно обострило проблему защиты информации.

Особенностями современных информационных технологий, прямо или косвенно влияющими на безопасность информации, являются:

- увеличение числа автоматизированных процедур в системах обработки данных и важности принимаемых на их основе решений;
- территориальная распределенность компонентов компьютерных систем и передача информации между этими компонентами;
- усложнение используемых программных и аппаратных средств компьютерных систем;
- накопление и долговременное хранение больших массивов данных на электронных носителях, зачастую не имеющих твердых копий;
- интеграция в единых базах данных информации различного назначения и различных режимов доступа;
- непосредственный доступ к ресурсам компьютерных систем большого количества пользователей различных категорий и с различными полномочиями в системе;
- рост стоимости ресурсов компьютерных систем.

Рост количества и качества угроз безопасности информации в компьютерных системах не всегда приводит к адекватному ответу в виде создания надежных систем защиты информации и безопасных информационных технологий. В большинстве коммерческих и госу-

дарственных организаций, не говоря уже об отдельных пользователях, в качестве средств защиты применяются только антивирусные программы, не всегда своевременно обновляемые, и разграничение прав пользователей компьютерной системы на основе паролей.

В связи с этим следует не только увеличивать количество специалистов в области информационной безопасности и защиты информации, но и обучать современным методам и средствам защиты информации специалистов других сфер — в первую очередь специалистов в области информатики и вычислительной техники. Решению этой задачи и посвящено данное учебное пособие.

В первой главе кратко изложена сущность комплексного подхода к обеспечению информационной безопасности компьютерных систем. Поскольку последующие главы книги посвящены в основном программно-аппаратным и криптографическим методам и средствам защиты информации, в первой главе излагаются основы организационной, правовой и инженерно-технической защиты информации.

Во второй главе рассмотрены методы и средства защиты информации от локального и удаленного несанкционированного доступа к информации в компьютерных системах на основе аутентификации пользователей и процессов, а также применения специализированных протоколов типа S/Key, CHAP и Kerberos.

В третьей главе рассмотрены методы и средства защиты информации от несанкционированного доступа к ней в операционных системах Windows и Unix. Рассмотрены также особенности дискреционного и мандатного управления доступом к объектам компьютерных систем. К особенностям представленного в этой главе материала относится включение в него анализа методов и средств защиты в открытых версиях операционной системы Windows (Windows 9x/ME/XP Home Edition), сведений об основных функциях привилегированного режима (режима администратора) из набора Windows API, сведений о подсистемах аудита событий безопасности.

В четвертой главе рассмотрены методы симметричной и асимметричной криптографии и их применение при решении задачи защиты информации в компьютерных системах. В отдельный раздел этой главы вынесен вопрос об основах современной компьютерной стеганографии.

В пятой главе изложены основы криптографического интерфейса приложений Windows (CryptoAPI) и применения его функций для задач обеспечения конфиденциальности, аутентичности и целостности хранимых и передаваемых данных. Отдельно рассмотрены вопросы защиты документов Microsoft Office и шифрующая файловая система Windows, базирующиеся на CryptoAPI.

В шестой главе описаны методы и средства защиты информации от вредоносных программ — компьютерных вирусов и про-

граммных закладок. Приведена классификация вредоносных программ, методов и средств защиты от них.

В седьмой главе изложены методы защиты программного обеспечения и других информационных ресурсов от их несанкционированного использования и копирования (защиты авторских прав).

Поскольку данное учебное пособие предназначено, в первую очередь, для студентов, обучающихся по специальностям группы «Информатика и вычислительная техника», автор включил в него достаточное количество примеров программ, иллюстрирующих использование различных методов и средств защиты информации.

Автор выражает искреннюю признательность коллегам по кафедре информационной безопасности Московского государственного социального университета, а также жене и дочери за помощь и поддержку в период подготовки учебного пособия.

ГЛАВА 1

КОМПЛЕКСНЫЙ ПОДХОД К ОБЕСПЕЧЕНИЮ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

1.1. Основные понятия защиты информации

Под *информацией*, применительно к задаче ее защиты, понимают сведения о лицах, предметах, фактах, событиях, явлениях и процессах независимо от формы их представления. В зависимости от формы представления информация может быть разделена на речевую, телекоммуникационную и документированную.

Речевая информация возникает в ходе ведения в помещениях разговоров, работы систем связи, звукоусиления и звуковоспроизведения. *Телекоммуникационная* информация циркулирует в технических средствах обработки и хранения информации, а также в каналах связи при ее передаче. К *документированной* информации, или *документам*, относят информацию, представленную на материальных носителях вместе с идентифицирующими ее реквизитами.

К *информационным процессам* относят процессы сбора, обработки, накопления, хранения, поиска и распространения информации.

Под *информационной системой* понимают упорядоченную совокупность документов и массивов документов и информационных технологий, реализующих информационные процессы.

Информационными ресурсами называют документы и массивы документов, существующие отдельно или в составе информационных систем.

Процесс создания оптимальных условий для удовлетворения информационных потребностей граждан, организаций, общества и государства в целом называют *информатизацией.*

Информацию разделяют на открытую и ограниченного доступа. К информации ограниченного доступа относятся государственная тайна и конфиденциальная информация. В соответствии с российским законодательством к конфиденциальной относится следующая информация:

- служебная и профессиональная тайна (врачебная, адвокатская, тайна суда и следствия и т. п.);
- коммерческая тайна;
- персональные данные (сведения о фактах, событиях и обстоятельствах жизни гражданина, позволяющие идентифицировать его личность).

Информация может являться объектом публичных, гражданских и иных правовых отношений. *Обладатель* информации — лицо, самостоятельно создавшее информацию либо получившее на основании закона или договора право разрешать или ограничивать доступ к информации. Обладателем информации может быть гражданин (физическое лицо), юридическое лицо, государство (Российская Федерация или ее субъект), муниципальное образование. Под *пользователем* информации будем понимать субъекта, обращающегося к информационной системе за получением необходимой ему информации и пользующегося ею.

К *защищаемой* относится информация, имеющая обладателя и подлежащая защите в соответствии с требованиями правовых документов или требованиями, устанавливаемыми обладателем информации.

Защитой информации называют деятельность по предотвращению утечки защищаемой информации, несанкционированных и непреднамеренных воздействий на защищаемую информацию.

Под *утечкой* понимают неконтролируемое распространение защищаемой информации путем ее разглашения, несанкционированного доступа к ней и получения разведками. *Разглашение* — это доведение защищаемой информации до неконтролируемого количества получателей информации (например, публикация информации на открытом сайте в сети Интернет или в открытой печати). *Несанкционированный доступ* — получение защищаемой информации заинтересованным субъектом с нарушением правил доступа к ней.

Несанкционированное воздействие на защищаемую информацию — воздействие с нарушением правил ее изменения (например, намеренное внедрение в защищаемые информационные ресурсы вредоносного программного кода или умышленная подмена электронного документа).

Под *непреднамеренным воздействием* на защищаемую информацию понимают воздействие на нее из-за ошибок пользователя, сбоя технических или программных средств, природных явлений, иных нецеленаправленных воздействий (например, уничтожение документов в результате отказа накопителя на жестком магнитном диске компьютера).

Целью защиты информации (ее желаемым результатом) является предотвращение ущерба собственнику, владельцу или пользователю информации. Под *эффективностью* защиты информации понимают степень соответствия результатов защиты информации поставленной цели. *Объектом защиты* может быть информация, ее носитель или информационный процесс, в отношении которых необходимо обеспечивать защиту в соответствии с поставленной целью.

Под *качеством информации* понимают совокупность свойств, обусловливающих пригодность информации удовлетворять опре-

деленные потребности ее пользователей в соответствии с назначением информации. Одним из показателей качества информации является ее *защищенность* — поддержание на заданном уровне тех параметров информации, которые характеризуют установленный статус ее хранения, обработки и использования.

Основными характеристиками защищаемой информации являются конфиденциальность, целостность и доступность. *Конфиденциальность* информации — это известность ее содержания только имеющим соответствующие полномочия субъектам. Конфиденциальность является субъективной характеристикой информации, связанной с объективной необходимостью защиты законных интересов одних субъектов от других.

Шифрованием информации называют процесс ее преобразования, при котором содержание информации становится непонятным для не обладающих соответствующими полномочиями субъектов. Результат шифрования информации называют *шифротекстом*, или *криптограммой*. Обратный процесс восстановления информации из шифротекста называют *расшифрованием* информации. Алгоритмы, используемые при шифровании и расшифровании информации, обычно не являются конфиденциальными, а конфиденциальность шифротекста обеспечивается использованием при шифровании дополнительного параметра, называемого *ключом шифрования*. Знание ключа шифрования позволяет выполнить правильное расшифрование шифротекста.

Целостностью информации называют неизменность информации в условиях ее случайного и (или) преднамеренного искажения или разрушения. Целостность является частью более широкой характеристики информации — ее достоверности, включающей помимо целостности еще полноту и точность отображения предметной области.

Хешированием информации называют процесс ее преобразования в хеш-значение фиксированной длины (дайджест). Одним из применений хеширования является обеспечение целостности информации.

Под *доступностью* информации понимают способность обеспечения беспрепятственного доступа субъектов к интересующей их информации. *Отказом в обслуживании* называют состояние информационной системы, при котором блокируется доступ к некоторому ее ресурсу. Совокупность информационных ресурсов и системы формирования, распространения и использования информации называют *информационной средой* общества.

Под *информационной безопасностью* понимают состояние защищенности информационной среды, обеспечивающее ее формирование и развитие.

Политика безопасности — это набор документированных норм, правил и практических приемов, регулирующих управление, защиту и распределение информации ограниченного доступа.

Целью данного учебного пособия является представление методов и средств защиты информации в компьютерных системах. *Компьютерной*, или *автоматизированной*, системой обработки информации называют организационно-техническую систему, включающую в себя:

- технические средства вычислительной техники и связи;
- методы и алгоритмы обработки информации, реализованные в виде программных средств;
- информацию (файлы, базы данных) на различных носителях;
- обслуживающий персонал и пользователей, объединенных по организационно-структурному, тематическому, технологическому или другим признакам.

1.2. Угрозы информационной безопасности и каналы утечки информации

Под *угрозой* безопасности информации в компьютерной системе (КС) понимают событие или действие, которое может вызвать изменение функционирования КС, связанное с нарушением защищенности обрабатываемой в ней информации.

Уязвимость информации — это возможность возникновения на каком-либо этапе жизненного цикла КС такого ее состояния, при котором создаются условия для реализации угроз безопасности информации.

Атакой на КС называют действие, предпринимаемое нарушителем, которое заключается в поиске и использовании той или иной уязвимости. Иначе говоря, атака на КС является реализацией угрозы безопасности информации в ней.

Угрозы информационной безопасности могут быть разделены на угрозы, не зависящие от деятельности человека (*естественные* угрозы физических воздействий на информацию стихийных природных явлений), и угрозы, вызванные человеческой деятельностью (*искусственные* угрозы), которые являются гораздо более опасными.

Искусственные угрозы исходя из их мотивов разделяются на *непреднамеренные* (случайные) и *преднамеренные* (умышленные).

К непреднамеренным угрозам относятся:

- ошибки в проектировании КС;
- ошибки в разработке программных средств КС;
- случайные сбои в работе аппаратных средств КС, линий связи, энергоснабжения;
- ошибки пользователей КС;
- воздействие на аппаратные средства КС физических полей других электронных устройств (при несоблюдении условий их электромагнитной совместимости) и др.

К умышленным угрозам относятся:

• несанкционированные действия обслуживающего персонала КС (например, ослабление политики безопасности администратором, отвечающим за безопасность КС);

• несанкционированный доступ к ресурсам КС со стороны пользователей КС и посторонних лиц, ущерб от которого определяется полученными нарушителем полномочиями.

В зависимости от целей преднамеренных угроз безопасности информации в КС угрозы могут быть разделены на три основные группы:

• угроза нарушения конфиденциальности, т.е. утечки информации ограниченного доступа, хранящейся в КС или передаваемой от одной КС к другой;

• угроза нарушения целостности, т.е. преднамеренного воздействия на информацию, хранящуюся в КС или передаваемую между КС (заметим, что целостность информации может быть также нарушена, если к несанкционированному изменению или уничтожению информации приводит случайная ошибка в работе программных или аппаратных средств КС; санкционированным является изменение или уничтожение информации, сделанное уполномоченным лицом с обоснованной целью);

• угроза нарушения доступности информации, т.е. отказа в обслуживании, вызванного преднамеренными действиями одного из пользователей КС (нарушителя), при котором блокируется доступ к некоторому ресурсу КС со стороны других пользователей КС (постоянно или на большой период времени).

Опосредованной угрозой безопасности информации в КС является угроза раскрытия параметров подсистемы защиты информации, входящей в состав КС. Реализация этой угрозы дает возможность реализации перечисленных ранее непосредственных угроз безопасности информации.

Результатом реализации угроз безопасности информации в КС может быть утечка (копирование) информации, ее утрата (разрушение) или искажение (подделка), блокирование информации. Поскольку сложно заранее определить возможную совокупность угроз безопасности информации и результатов их реализации, модель потенциальных угроз безопасности информации в КС должна создаваться совместно собственником (владельцем) КС и специалистами по защите информации на этапе проектирования КС. Созданная модель должна затем уточняться в ходе эксплуатации КС.

Рассмотрим возможные каналы утечки информации в КС. *Косвенными* каналами утечки называют каналы, не связанные с физическим доступом к элементам КС:

• использование подслушивающих (радиозакладных) устройств;

• дистанционное видеонаблюдение;

• перехват побочных электромагнитных излучений и наводок (ПЭМИН).

Побочные электромагнитные излучения создаются техническими средствами КС при обработке информации, существуют в диапазоне от единиц герц до 1,5 ГГц и могут распространять обрабатываемую информацию с дальностью до 1 км. Наиболее опасными с точки зрения ПЭМИН являются дисплеи, кабельные линии связи, накопители на магнитных дисках, матричные принтеры. Для перехвата ПЭМИН используется специальная портативная аппаратура, включающая в себя широкополосный автоматизированный супергетеродинный приемник с устройством регистрации информации на магнитном носителе и (или) дисплеем.

Побочные электромагнитные наводки представляют собой сигналы в цепях электропитания и заземления аппаратных средств КС и в находящихся в зоне воздействия ПЭМИН работающих аппаратных средств КС кабелях вспомогательных устройств (звукоусиления, связи, времени, сигнализации), металлических конструкциях зданий, сантехническом оборудовании. Эти наведенные сигналы могут выходить за пределы зоны безопасности КС.

Другим классом каналов утечки информации являются *непосредственные* каналы, связанные с физическим доступом к элементам КС. К непосредственным каналам утечки, не требующим изменения элементов КС, относятся:

• хищение носителей информации;

• сбор производственных отходов с информацией (бумажных и магнитных носителей);

• намеренное копирование файлов других пользователей КС;

• чтение остаточной информации после выполнения заданий других пользователей (областей оперативной памяти, удаленных файлов, ошибочно сохраненных временных файлов);

• копирование носителей информации;

• намеренное использование для несанкционированного доступа к информации незаблокированных терминалов других пользователей КС;

• маскировка под других пользователей путем похищения их идентифицирующей информации (паролей, карт и т. п.);

• обход средств разграничения доступа к информационным ресурсам вследствие недостатков в их программном обеспечении и др.

К непосредственным каналам утечки, предполагающим изменение элементов КС и ее структуры, относятся:

• незаконное подключение специальной регистрирующей аппаратуры к устройствам или линиям связи (пассивное для фиксации и сохранения передаваемых данных или активное для их уничтожения, искажения или подмены);

• злоумышленное изменение программ для выполнения ими несанкционированного копирования информации при ее обработке;

• злоумышленный вывод из строя средств защиты информации.

Пассивное подключение нарушителя к устройствам или линиям связи легко предотвратить (например, с помощью шифрования передаваемой информации), но невозможно обнаружить. Активное подключение, напротив, легко обнаружить (например, с помощью хеширования и шифрования передаваемой информации), но невозможно предотвратить.

Помимо утечки информации в КС возможны также ее несанкционированное уничтожение или искажение (например, заражение компьютерными вирусами), а также несанкционированное использование информации при санкционированном доступе к ней (например, нарушение авторских прав владельцев или собственников программного обеспечения или баз данных).

Наличие в КС значительного числа потенциальных каналов утечки информации является объективным фактором и обусловливает уязвимость информации в подобных системах с точки зрения ее несанкционированного использования.

Поскольку наиболее опасные угрозы информационной безопасности вызваны преднамеренными действиями нарушителя, которые в общем случае являются неформальными, проблема защиты информации относится к формально не определенным проблемам. Отсюда следуют два основных вывода:

• надежная защита информации в КС не может быть обеспечена только формальными методами (например, только программными и аппаратными средствами);

• защита информации в КС не может быть абсолютной.

При решении задачи защиты информации в КС необходимо применять так называемый системно-концептуальный подход. В соответствии с ним решение задачи должно подразумевать:

• системность целевую, при которой защищенность информации рассматривается как составная неотъемлемая часть ее качества;

• системность пространственную, предполагающую взаимосвязанность защиты информации во всех элементах КС;

• системность временную, предполагающую непрерывность защиты информации;

• системность организационную, предполагающую единство организации всех работ по защите информации в КС и управления ими.

Концептуальность подхода к решению задачи защиты информации в КС предусматривает ее решение на основе единой концепции (совокупности научно обоснованных решений, необхо-

димых и достаточных для оптимальной организации защиты информации в КС).

Обеспечение информационной безопасности КС является непрерывным процессом, целенаправленно проводимым на всех этапах ее жизненного цикла с комплексным применением всех имеющихся методов и средств.

Существующие методы и средства защиты информации можно подразделить на четыре основные группы:

- методы и средства организационно-правовой защиты информации;
- методы и средства инженерно-технической защиты информации;
- криптографические методы и средства защиты информации;
- программно-аппаратные методы и средства защиты информации.

1.3. Организационно-правовое обеспечение информационной безопасности

К методам и средствам *организационной* защиты информации относятся организационно-технические и организационно-правовые мероприятия, проводимые в процессе создания и эксплуатации КС для обеспечения защиты информации. Эти мероприятия должны проводиться при строительстве или ремонте помещений, в которых будет размещаться КС; проектировании системы, монтаже и наладке ее технических и программных средств; испытаниях и проверке работоспособности КС.

Основные свойства методов и средств организационной защиты:

- обеспечение полного или частичного перекрытия значительной части каналов утечки информации (например, хищения или копирования носителей информации);
- объединение всех используемых в КС средств в целостный механизм защиты информации.

Методы и средства организационной защиты информации включают в себя:

- ограничение физического доступа к объектам КС и реализация режимных мер;
- ограничение возможности перехвата ПЭМИН;
- разграничение доступа к информационным ресурсам и процессам КС (установка правил разграничения доступа, шифрование информации при ее хранении и передаче, обнаружение и уничтожение аппаратных и программных закладок);
- резервное копирование наиболее важных с точки зрения утраты массивов документов;

• профилактику заражения компьютерными вирусами.

Перечислим основные виды мероприятий, которые должны проводиться на различных этапах жизненного цикла КС:

1) на этапе создания КС: при разработке ее общего проекта и проектов отдельных структурных элементов — анализ возможных угроз и методов их нейтрализации; при строительстве и переоборудовании помещений — приобретение сертифицированного оборудования, выбор лицензированных организаций; при разработке математического, программного, информационного и лингвистического обеспечения — использование сертифицированных программных и инструментальных средств; при монтаже и наладке оборудования — контроль за работой технического персонала; при испытаниях и приемке в эксплуатацию — включение в состав аттестационных комиссий сертифицированных специалистов;

2) в процессе эксплуатации КС — организация пропускного режима, определение технологии автоматизированной обработки документов, организация работы обслуживающего персонала, распределение реквизитов разграничения доступа пользователей к элементам КС (паролей, ключей, карт и т. п.), организация ведения протоколов работы КС, контроль выполнения требований служебных инструкций и т. п.;

3) мероприятия общего характера — подбор и подготовка кадров, организация плановых и предупреждающих проверок средств защиты информации, планирование мероприятий по защите информации, обучение персонала, участие в семинарах, конференциях и выставках по проблемам безопасности информации и т. п.

Основой проведения организационных мероприятий является использование и подготовка законодательных и нормативных документов в области информационной безопасности, которые на правовом уровне должны регулировать доступ к информации со стороны потребителей. В российском законодательстве позже, чем в законодательстве других развитых стран, появились необходимые правовые акты (хотя далеко не все).

Можно выделить четыре уровня правового обеспечения информационной безопасности. *Первый уровень* образуют международные договоры, к которым присоединилась Российская Федерация, и федеральные законы России:

• международные (всемирные) конвенции об охране промышленной собственности, охране интеллектуальной собственности, авторском праве;

• Конституция РФ (ст. 23 определяет право граждан на тайну переписки, телефонных, телеграфных и иных сообщений);

• Гражданский кодекс РФ (в ст. 139 устанавливается право на возмещение убытков от утечки с помощью незаконных методов информации, относящейся к служебной и коммерческой тайне);

• часть четвертая Гражданского кодекса РФ, введенная в действие с 1 января 2008 г. (ст. 1271 определяет знак охраны авторского права — латинскую букву «С» в окружности с указанием правообладателя и года первого опубликования произведения; ст. 1301 — ответственность за нарушение исключительного права на произведение в виде возмещения убытков или, по выбору правообладателя, выплаты компенсации в размере от 10 000 до 5 000 000 р. или в двухкратном размере стоимости экземпляров произведения);

• Уголовный кодекс РФ (ст. 272 устанавливает ответственность за неправомерный доступ к компьютерной информации, ст. 273 — за создание, использование и распространение вредоносных программ для ЭВМ, ст. 274 — за нарушение правил эксплуатации ЭВМ, систем и сетей);

• Федеральный закон «Об информации, информационных технологиях и защите информации» от 20.07.2006 № 149-ФЗ (ст. 9 устанавливает порядок и условия ограничения доступа к информации, составляющей государственную тайну, коммерческую тайну, служебную тайну, профессиональную тайну и персональные данные, ст. 16 определяет сущность и порядок защиты информации);

• Федеральный закон «О государственной тайне» от 21.07.93 № 5485-1 (ст. 5 устанавливает перечень сведений, составляющих государственную тайну; ст. 8 — степени секретности сведений и грифы секретности их носителей: «особой важности», «совершенно секретно» и «секретно»; ст. 20 — органы по защите государственной тайны, межведомственную комиссию по защите государственной тайны для координации деятельности этих органов; ст. 28 — порядок сертификации средств защиты информации, относящейся к государственной тайне);

• Федеральные законы «О лицензировании отдельных видов деятельности» от 08.08.2001 № 128-ФЗ, «О связи» от 16.02.95 № 15-ФЗ, «Об электронной цифровой подписи» от 10.01.02 № 1-ФЗ, «О коммерческой тайне» от 29.07.2004 № 98-ФЗ, «О персональных данных» от 27.07.2006 № 152-ФЗ.

Второй уровень правового обеспечения информационной безопасности составляют подзаконные акты, к которым относятся указы Президента РФ и постановления Правительства РФ, а также письма Высшего Арбитражного Суда РФ и постановления пленумов Верховного Суда РФ. Примерами таких актов могут являться Указ Президента РФ «Об утверждении перечня сведений конфиденциального характера» от 06.03.97 № 188 или Постановление Правительства РФ «О перечне сведений, которые не могут составлять коммерческую тайну» от 05.12.91 № 35.

Третий уровень правового обеспечения информационной безопасности составляют государственные стандарты (ГОСТы) в области защиты информации, руководящие документы, нормы, методики и классификаторы, разработанные соответствующими

государственными органами. В качестве примеров можно привести следующие документы:

- ГОСТ Р 50922—96 «Защита информации. Основные термины и определения», ГОСТ Р 50739—95 «Средства вычислительной техники. Защита от несанкционированного доступа к информации. Общие технические требования», ГОСТ 28147—89 «Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования» и др.;
- руководящие документы Государственной технической комиссии при Президенте Российской Федерации (Гостехкомиссии России) «Концепция защиты средств вычислительной техники и автоматизированных систем от несанкционированного доступа к информации», «Автоматизированные системы. Защита от несанкционированного доступа к информации. Классификация автоматизированных систем и требования по защите информации» и др.

Четвертый уровень правового обеспечения информационной безопасности образуют локальные нормативные акты, положения, инструкции, методические рекомендации и другие документы по комплексной защите информации в КС конкретной организации. К таким нормативным документам относятся:

- приказ об утверждении перечня сведений, составляющих коммерческую тайну предприятия;
- трудовые и гражданско-правовые договоры (подряда, поручения, комиссии и т. п.), в которые включены пункты об обязанности возмещения ущерба за разглашение сведений, составляющих коммерческую тайну предприятия, и др.

1.4. Инженерно-технические методы и средства защиты информации

Под инженерно-техническими средствами защиты информации понимают физические объекты, механические, электрические и электронные устройства, элементы конструкции зданий, средства пожаротушения и другие средства, обеспечивающие:

- защиту территории и помещений КС от проникновения нарушителей;
- защиту аппаратных средств КС и носителей информации от хищения;
- предотвращение возможности удаленного (из-за пределов охраняемой территории) видеонаблюдения (подслушивания) за работой персонала и функционированием технических средств КС;
- предотвращение возможности перехвата ПЭМИН, вызванных работающими техническими средствами КС и линиями передачи данных;
- организацию доступа в помещения КС сотрудников;

• контроль над режимом работы персонала КС;

• контроль над перемещением сотрудников КС в различных производственных зонах;

• противопожарную защиту помещений КС;

• минимизацию материального ущерба от потерь информации, возникших в результате стихийных бедствий и техногенных аварий.

Важнейшей составной частью инженерно-технических средств защиты информации являются технические средства охраны, которые образуют первый рубеж защиты КС и являются необходимым, но недостаточным условием сохранения конфиденциальности и целостности информации в КС.

Рассмотрим немного подробнее методы и средства защиты информации от утечки по каналам ПЭМИН. Основной задачей является уменьшение соотношения сигнал/шум в этих каналах до предела, при котором восстановление информации становится принципиально невозможным. Возможными методами решения этой задачи могут быть:

1) снижение уровня излучений сигналов в аппаратных средствах КС;

2) увеличение мощности помех в соответствующих этим сигналам частотных диапазонах.

Для применения *первого метода* необходим выбор системно-технических и конструкторских решений при создании технических средств КС в защищенном исполнении, а также рациональный выбор места размещения этих средств относительно мест возможного перехвата ПЭМИН (для соблюдения условия максимального затухания информационного сигнала). Требования к средствам вычислительной техники в защищенном исполнении определяются в специальных ГОСТах.

Реализация *второго метода* возможна путем применения активных средств защиты в виде генераторов сигналоподобных помех или шума.

Отметим перспективные методы и средства защиты информации в КС от утечки по каналам ПЭМИН:

• выбор элементной базы технических средств КС с возможно более малым уровнем информационных сигналов;

• замена в информационных каналах КС электрических цепей волоконно-оптическими линиями;

• локальное экранирование узлов технических средств, являющихся первичными источниками информационных сигналов;

• включение в состав информационных каналов КС устройств предварительного шифрования обрабатываемой информации.

Отметим, что при использовании технических средств КС для обработки информации ограниченного доступа необходимо проведение специальных проверок, целью которых является обнару-

жение и устранение внедренных специальных электронных устройств подслушивания, перехвата информации или вывода технических средств из строя (аппаратных закладок). При проведении таких проверок может потребоваться практически полная их разборка, что иногда может привести к возникновению неисправностей в работе технических средств и дополнительным затратам на их устранение.

Рассмотрим средства обнаружения электронных подслушивающих (радиозакладных) устройств, простейшими из которых являются нелинейные локаторы. Они с помощью специального передатчика в сверхвысокочастотном диапазоне радиоволн облучают окружающее пространство и регистрируют вторичный, переизлученный сигнал, поступающий от различных полупроводниковых элементов, находящихся как во включенном, так и в выключенном состоянии. Нелинейные локаторы могут не выявить радиозакладное устройство, если оно вмонтировано в электронное устройство (системный блок компьютера, телевизор, телефонный аппарат и т. п.), так как сигнал отклика от подслушивающего устройства будет замаскирован откликом от электронной аппаратуры. В этом случае потребуется применение более сложных устройств контроля постороннего радиоизлучения — индикаторов электромагнитного излучения, сканирующих приемников, компьютерных анализаторов.

1.5. Программные и программно-аппаратные методы и средства обеспечения информационной безопасности

К аппаратным средствам защиты информации относятся электронные и электронно-механические устройства, включаемые в состав технических средств КС и выполняющие (самостоятельно или в едином комплексе с программными средствами) некоторые функции обеспечения информационной безопасности. Критерием отнесения устройства к аппаратным, а не к инженерно-техническим средствам защиты является обязательное включение в состав технических средств КС.

К основным аппаратным средствам защиты информации относятся:

- устройства для ввода идентифицирующей пользователя информации (магнитных и пластиковых карт, отпечатков пальцев и т. п.);
- устройства для шифрования информации;
- устройства для воспрепятствования несанкционированному включению рабочих станций и серверов (электронные замки и блокираторы).

Примеры вспомогательных аппаратных средств защиты информации:

- устройства уничтожения информации на магнитных носителях;
- устройства сигнализации о попытках несанкционированных действий пользователей КС и др.

Под программными средствами защиты информации понимают специальные программы, включаемые в состав программного обеспечения КС исключительно для выполнения защитных функций.

К основным программным средствам защиты информации относятся:

- программы идентификации и аутентификации пользователей КС;
- программы разграничения доступа пользователей к ресурсам КС;
- программы шифрования информации;
- программы защиты информационных ресурсов (системного и прикладного программного обеспечения, баз данных, компьютерных средств обучения и т. п.) от несанкционированного изменения, использования и копирования.

Заметим, что под *идентификацией*, применительно к обеспечению информационной безопасности КС, понимают однозначное распознавание уникального имени субъекта КС. *Аутентификация* означает подтверждение того, что предъявленное имя соответствует данному субъекту (подтверждение подлинности субъекта).

Примеры вспомогательных программных средств защиты информации:

- программы уничтожения остаточной информации (в блоках оперативной памяти, временных файлах и т. п.);
- программы аудита (ведения регистрационных журналов) событий, связанных с безопасностью КС, для обеспечения возможности восстановления и доказательства факта происшествия этих событий;
- программы имитации работы с нарушителем (отвлечения его на получение якобы конфиденциальной информации);
- программы тестового контроля защищенности КС и др.

К преимуществам программных средств защиты информации относятся:

- простота тиражирования;
- гибкость (возможность настройки на различные условия применения, учитывающие специфику угроз информационной безопасности конкретных КС);
- простота применения — одни программные средства, например шифрования, работают в «прозрачном» (незаметном для пользователя) режиме, а другие не требуют от пользователя никаких новых (по сравнению с другими программами) навыков;

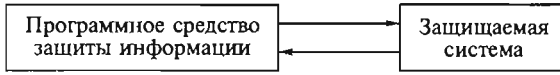

Рис. 1.1. Пример пристыкованного программного средства защиты

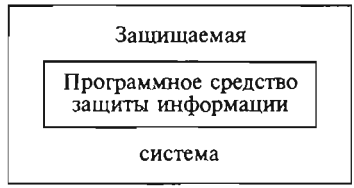

Рис. 1.2. Пример встроенного программного средства защиты

• практически неограниченные возможности их развития путем внесения изменений для учета новых угроз безопасности информации.

К недостаткам программных средств защиты информации относятся:

• снижение эффективности КС за счет потребления ее ресурсов, требуемых для функционирование программ защиты;

• более низкая производительность (по сравнению с выполняющими аналогичные функции аппаратными средствами защиты, например шифрования);

• пристыкованность многих программных средств защиты (а не их встроенность в программное обеспечение КС, рис. 1.1 и 1.2), что создает для нарушителя принципиальную возможность их обхода;

• возможность злоумышленного изменения программных средств защиты в процессе эксплуатации КС.

1.6. Требования к комплексным системам защиты информации

Поскольку потенциальные угрозы безопасности информации весьма многообразны, цели защиты информации могут быть достигнуты только путем создания комплексной системы защиты информации (КСЗИ), под которой понимается совокупность методов и средств, объединенных единым целевым назначением и обеспечивающих необходимую эффективность защиты информации в КС.

Основные требования к комплексной системе защиты информации:

• разработка на основе положений и требований существующих законов, стандартов и нормативно-методических документов по обеспечению информационной безопасности;

• использование комплекса программно-технических средств и организационных мер для защиты КС;

• надежность, производительность, конфигурируемость;

• экономическая целесообразность (поскольку стоимость КСЗИ включается в стоимость КС, стоимость средств защиты не должна быть выше возможного ущерба от потери информации);

• выполнение на всех этапах жизненного цикла обработки информации в КС (в том числе при проведении ремонтных и регламентных работ);

• возможность совершенствования;

• обеспечение разграничения доступа к конфиденциальной информации с отвлечением нарушителя на ложную информацию (обеспечение не только пассивной, но и активной защиты);

• взаимодействие с незащищенными КС по установленным для этого правилам разграничения доступа;

• обеспечение проведения учета и расследования случаев нарушения безопасности информации в КС;

• сложная для пользователя (не должна вызывать у него психологического противодействия и стремления обойтись без применения ее средств);

• возможность оценки эффективности ее применения.

Впервые основные категории требований к защищенности КС были сформулированы в документе Министерства обороны США «Trusted Computer System Evaluation Criteria» («Критерии оценки безопасности компьютерных систем», или «Оранжевая книга»), 1985 г. В этом документе предложены три основные категории требований.

1. Политика:

• наличие явной и хорошо определенной политики обеспечения безопасности;

• использование маркировки объектов КС для управления доступом к ним.

2. Подотчетность:

• индивидуальная идентификация субъектов КС;

• сохранение и защита информации аудита.

3. Гарантии:

• включение в состав КС программно-аппаратных средств для получения гарантий выполнения требований категорий 1 и 2;

• постоянная защищенность средств обеспечения безопасности информации в КС от их преодоления и (или) несанкционированного изменения.

В «Оранжевой книге» были введены семь классов защищенности КС — от минимальной защиты (класс D1) до верифициро-

ванной (формально доказанной) защиты (класс А1). Требования «Оранжевой книги» явились первой попыткой создать единый стандарт безопасности КС, рассчитанный на проектировщиков, разработчиков (программистов), пользователей подобных систем и специалистов по их сертификации.

Отличительной чертой этого стандарта является ориентация на государственные (в первую очередь военные) организации и операционные системы.

В 1992 г. Гостехкомиссия России опубликовала первый комплект руководящих документов по защите средств вычислительной техники (СВТ) и автоматизированных систем (АС) от несанкционированного доступа.

СВТ не решают непосредственно прикладных задач, а используются в качестве элементов АС. Примерами СВТ являются плата расширения BIOS с соответствующим аппаратным и программным интерфейсом для аутентификации пользователей АС или программа «прозрачного» шифрования информации на жестком диске.

В руководящих документах Гостехкомиссии России определены семь классов защищенности СВТ от несанкционированного доступа к обрабатываемой (сохраняемой, передаваемой) с помощью этих средств информации (наиболее защищенным является первый класс),

АС рассматривается как комплекс СВТ и имеет дополнительные характеристики: полномочия пользователей, модель нарушителя, технология обработки информации. Типичным примером АС является многопользовательская и многозадачная операционная система.

В руководящих документах Гостехкомиссии России определены девять классов защищенности АС от несанкционированного доступа, объединенных в три группы:

- однопользовательские АС с информацией, размещенной на носителях одного уровня конфиденциальности (класс 3Б и 3А);
- многопользовательские АС с одинаковыми полномочиями пользователей и информацией на носителях разного уровня конфиденциальности (классы 2Б и 2А);
- многопользовательские АС с разными полномочиями пользователей и информацией разного уровня конфиденциальности (в порядке возрастания защищенности от класса 1Д до класса 1А).

Под несанкционированным доступом к информации в руководящих документах Гостехкомиссии России понимается доступ к информации, нарушающий установленные правила разграничения доступа и использующий штатные возможности СВТ и АС. Руководящие документы Гостехкомиссии России, подобно «Оранжевой книге», ориентированы прежде всего на применение в КС силовых структур Российской Федерации.

Дальнейшее развитие стандартов в области информационной безопасности КС привело к появлению европейских «Критериев оценки безопасности информационных технологий» (Information Technology Security Evaluation Criteria), американских «Федеральных критериев безопасности информационных технологий» (Federal Criteria for Information Technology Security), канадских «Критериев оценки безопасности компьютерных продуктов» (Canadian Trusted Computer Product Evaluation Criteria) и завершилось на сегодняшний день принятием «Общих критериев оценки безопасности информационных технологий» (Common Criteria for Information Technology Security Evaluation).

«Общие критерии...» адресованы трем группам специалистов (пользователям, разработчикам и экспертам по классификации КС) и представляют собой новый межгосударственный уровень в стандартизации безопасности информационных технологий.

В Российской Федерации «Общие критерии...» изданы в качестве ГОСТа (ГОСТ РИСО/МЭК 15408—2001 «Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий»).

В «Общих критериях...» предложена система функциональных требований к защищенным КС и критерии их независимого ранжирования.

Иначе говоря, в этих стандартах не устанавливается линейная шкала уровней безопасности КС, характерная для «Оранжевой книги». Это объясняется тем, что для одних КС наиболее важным требованием является идентификация и аутентификация пользователей, а для других — реализация конкретной политики разграничения доступа к ресурсам или обеспечение доступности информации.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. В каких формах может быть представлена информация?
2. Какая информация называется документированной?
3. Что относится к информации ограниченного доступа?
4. Что понимается под зашитой информации?
5. Что относится к основным характеристикам защищаемой информации?
6. Что такое угроза безопасности информации? Каковы основные виды угроз?
7. Какие существуют каналы утечки конфиденциальной информации?
8. Что такое ПЭМИН?
9. В чем сущность системно-концептуального подхода к защите информации в компьютерных системах?
10. Почему проблема защиты информации не может быть решена с помощью только формальных методов и средств?
11. В чем сущность организационной защиты информации?

12. Каковы уровни правового обеспечения информационной безопасности?

13. Какие законодательные акты составляют основу российского информационного права?

14. Что относится к средствам инженерно-технической защиты информации и для чего они предназначены?

15. В чем заключаются достоинства и недостатки программных средств защиты информации?

16. Какие требования предъявляются к комплексным системам защиты информации?

17. Какие существуют международные и российские стандарты в области безопасности компьютерных систем и информационных технологий?

ГЛАВА 2

МЕТОДЫ И СРЕДСТВА ЗАЩИТЫ ИНФОРМАЦИИ ОТ НЕСАНКЦИОНИРОВАННОГО ДОСТУПА

2.1. Способы несанкционированного доступа к информации в компьютерных системах и защиты от него

В руководящих документах Гостехкомиссии России приведены следующие основные способы несанкционированного доступа к информации в КС:

- непосредственное обращение к объекту с конфиденциальной информацией (например, с помощью управляемой пользователем программы, читающей данные из файла или записывающей их в него);
- создание программных и технических средств, выполняющих обращение к объекту в обход средств защиты (например, с использованием случайно или намеренно оставленных разработчиком этих средств, так называемых люков);
- модификация средств защиты для осуществления несанкционированного доступа (например, внедрение программных закладок);
- внедрение в технические средства СВТ или АС программных или технических механизмов, нарушающих структуру и функции этих средств для осуществления несанкционированного доступа (например, путем загрузки на компьютере иной, незащищенной операционной системы).

Модель нарушителя в руководящих документах Гостехкомиссии России определяется исходя из следующих предположений:

- нарушитель имеет доступ к работе со штатными средствами КС;
- нарушитель является специалистом высшей квалификации (знает все о КС и, в частности, о системе и средствах ее защиты).

Можно выделить следующие уровни возможностей нарушителя, предоставляемые ему штатными средствами КС (каждый следующий уровень включает в себя предыдущий):

1) запуск программ из фиксированного набора (например, подготовка документов или получение почтовых сообщений);

2) создание и запуск собственных программ (возможности опытного пользователя или пользователя с полномочиями отладки программ);

3) управление функционированием КС — воздействие на ее базовое программное обеспечение, состав и конфигурацию КС (например, внедрение программной закладки);

4) весь объем возможностей лиц, осуществляющих проектирование, реализацию и ремонт средств КС, вплоть до включения в состав КС собственных СВТ с новыми функциями.

С учетом различных уровней возможностей нарушителя выделяют следующие вспомогательные способы несанкционированного доступа к информации в КС, позволяющие нарушителю использовать перечисленные ранее основные способы:

- ручной или программный подбор паролей путем их полного перебора или при помощи специального словаря (взлом КС);
- подключение к КС в момент кратковременного прекращения работы легального пользователя, работающего в интерактивном режиме и не заблокировавшего свой терминал;
- подключение к линии связи и перехват доступа к КС после отправки пакета завершения сеанса легального пользователя, работающего в удаленном режиме;
- выдача себя за легального пользователя с применением похищенной у него или полученной обманным путем (с помощью так называемой социальной инженерии) идентифицирующей информации — «маскарад»;
- создание условий для связи по компьютерной сети легального пользователя с терминалом нарушителя, выдающего себя за легального объекта КС (например, одного из ее серверов), — «мистификация»;
- создание условий для возникновения в работе КС сбоев, которые могут повлечь за собой отключение средств защиты информации или нарушение правил политики безопасности;
- тщательное изучение подсистемы защиты КС и используемой в ней политики безопасности, выявление ошибочных участков в программных средствах защиты информации в КС, введение программных закладок, разрешающих доступ нарушителю.

Приведем пример использования способа несанкционированного доступа к информации в КС, основанный на создании аварийной ситуации. Если у нарушителя есть физический доступ хотя бы к одной рабочей станции локальной вычислительной сети (ЛВС) организации или к линии связи, то он сможет внедрить на рабочей станции программную закладку (или подключить к линии связи специальное устройство), перехватывать все пакеты подключения легального пользователя этой рабочей станции к серверу ЛВС и искажать имя пользователя в этих пакетах (иначе говоря, создать условия, при которых легальный пользователь КС никогда не сможет подключиться к серверу).

В этой ситуации на атакуемую рабочую станцию рано или поздно придет администратор ЛВС для того, чтобы разобраться в причинах сбоев при подключении к серверу. Если при этом администратор пошлет пакет подключения к серверу под своей привилегированной учетной записью, в которой оставлено имя адми-

нистратора по умолчанию (например, «Supervisor» в операционной системе Novell Netware или «Администратор» в операционных системах Windows NT/2000/XP Professional), то тем самым цель нарушителя (перехват пароля администратора) будет достигнута.

Причиной успеха описанной в данном примере атаки является нарушение администратором системы правил политики безопасности, в соответствии с которыми он должен использовать привилегированную учетную запись только для выполнения административных функций и только с защищенной рабочей станции, а для выполнения других действия требуется создать другую учетную запись администратора с отличным от принятого по умолчанию именем.

В соответствии с руководящими документами Гостехкомиссии России основными направлениями обеспечения защиты СВТ и АС от несанкционированного доступа являются создание системы разграничения доступа (СРД) субъектов к объектам доступа и создание обеспечивающих средств для СРД.

К основным функциям СРД относятся:

- реализация правил разграничения доступа субъектов и их процессов к информации и устройствам создания ее твердых копий;
- изоляция процессов, выполняемых в интересах субъекта доступа, от других субъектов;
- управление потоками информации в целях предотвращения ее записи на носители несоответствующего уровня конфиденциальности;
- реализация правил обмена информацией между субъектами в компьютерных сетях.

К функциям обеспечивающих средств для СРД относятся:

- идентификация и аутентификация субъектов и поддержание привязки субъекта к процессу, выполняемому для него;
- регистрация действий субъекта и активизированного им процесса;
- исключение и включение новых субъектов и объектов доступа, изменение полномочий субъектов;
- реакция на попытки несанкционированного доступа (сигнализация, блокировка, восстановление объекта после несанкционированного доступа);
- учет выходных печатных форм в КС;
- контроль целостности программной и информационной части СРД и обеспечивающих ее средств.

Итак, основными способами защиты от несанкционированного доступа к информации в компьютерных системах являются аутентификация, *авторизация* (определение прав доступа субъекта к объекту с конфиденциальной информацией) и шифрование информации.

Под *протоколом* в общем случае понимают конечную последовательность однозначно и точно определенных действий, выполняемых двумя или более сторонами для достижения желаемого результата за конечное время. Рассмотрим протокол идентификации пользователя при его входе в КС (под «С» понимается система, под «П» — пользователь):

1. С: запрос имени, под которым пользователь зарегистрирован в базе данных учетных записей КС (логического имени пользователя или так называемого логина).
2. П: ввод логического имени (ID).
3. С: проверка наличия ID в регистрационной базе данных. Если пользователь с таким именем зарегистрирован, то запрос его идентифицирующей информации, в противном случае — возврат к п. 1.
4. П: ввод идентифицирующей информации (P).
5. С: проверка совпадения P с идентифицирующей информацией для пользователя ID в регистрационной базе данных. Если совпадение есть, то допуск пользователя к работе в КС, в противном случае — возврат к п. 3.

Присвоение каждому пользователю КС уникального логического имени, под которым он регистрируется в базе данных учетных записей, не только позволяет предоставить разным пользователям КС различный уровень прав в ней, но и дает возможность полного учета всех входов пользователя в систему в журнале аудита.

Приведем типичную структуру учетной записи i в регистрационной базе данных КС:

- относительный номер учетной записи RID_i;
- логическое имя пользователя ID_i;
- полное имя пользователя и его должность в организации D_i;
- случайное значение S_i, генерируемое при регистрации пользователя в КС (используется для предотвращения возможности получения одним пользователем полномочий другого пользователя при случайном совпадении идентифицирующей информации);
- идентифицирующая пользователя информация P_i;
- информация о правах пользователя в КС R_i.

Доступ к базе данных учетных записей КС как по чтению, так и по записи должен быть разрешен только привилегированному пользователю (администратору). Рассмотрим возможные угрозы безопасности информации в КС, если доступ к регистрационной базе данных будет разрешен всем зарегистрированным в КС пользователям.

Если разрешен доступ по записи (без права добавления данных в регистрационную базу), то тогда возможна следующая ситуация. Пользователь i после входа в КС изменяет идентифицирующую информацию в учетной записи пользователя j на идентифицирующую информацию из своей учетной записи, сохраняя при этом старую информацию из учетной записи j, после чего завершает

сеанс работы с КС и возобновляет его уже как пользователь *j*. Применив полномочия другого пользователя, нарушитель восстанавливает идентифицирующую информацию в учетной записи *j*, после чего завершает сеанс работы с КС.

Если к регистрационной базе данных КС разрешен доступ по чтению, то пользователь-нарушитель сможет скопировать ее на собственный носитель или просто в другой файл и осуществить попытку подбора идентифицирующей информации (например, пароля) привилегированного пользователя для осуществления несанкционированного доступа с помощью «маскарада».

Для удобства назначения полномочий пользователям КС они могут объединяться в группы в соответствии с должностным положением пользователей в организации и (или) их принадлежностью одному из ее структурных подразделений. Информация о группах пользователей также может размещаться в регистрационной базе данных КС.

Рассмотрим способы аутентификации пользователей в КС, которые можно подразделить на три группы. К *первой группе* относятся способы аутентификации, основанные на том, что пользователь знает некоторую подтверждающую его подлинность информацию (парольная аутентификация и аутентификация на основе модели «рукопожатия»).

Ко *второй группе* относятся способы аутентификации, основанные на том, что пользователь имеет некоторый материальный объект, который может подтвердить его подлинность (например, пластиковую карту с идентифицирующей пользователя информацией).

К *третьей группе* относятся способы аутентификации, основанные на таких данных, которые позволяют однозначно считать, что пользователь и есть тот самый субъект, за которого себя выдает (биометрические данные, особенности клавиатурного почерка и росписи мышью и т. п.).

В соответствии с «Оранжевой книгой» (см. подразд. 1.6) в защищенных КС, начиная с класса С1, должен использоваться хотя бы один из способов аутентификации (например, пароль), а данные аутентификации должны быть защищены от доступа неавторизованного пользователя.

В руководящих документах Гостехкомиссии России (см. подразд. 1.6) в АС, отнесенных к классу защищенности 1Д, должна осуществляться идентификация и проверка подлинности субъектов при входе в систему по паролю условно-постоянного действия длиной не менее шести буквенно-цифровых символов. Для классов защищенности 1Г и 1В дополнительно требуется использовать идентификатор (код, логическое имя) пользователя. Для отнесения АС к классу защищенности 1Б дополнительно необходимо использовать пароль временного действия длиной не менее восьми буквенно-цифровых символов. В требованиях к классу защи-

щенности 1А определена необходимость применения пользователями при входе в АС биометрических характеристик или специальных устройств (жетонов, карт, электронных ключей) и пароля временного действия длиной не менее восьми буквенно-цифровых символов.

2.2. Аутентификация пользователей на основе паролей и модели «рукопожатия»

При выборе паролей пользователи КС должны руководствоваться двумя, по сути взаимоисключающими, правилами — пароли должны трудно подбираться и легко запоминаться (поскольку пароль ни при каких условиях не должен нигде записываться, так как в этом случае необходимо будет дополнительно решать задачу защиты носителя пароля).

Сложность подбора пароля определяется, в первую очередь, мощностью множества символов, используемого при выборе пароля (N), и минимально возможной длиной пароля (k). В этом случае число различных паролей может быть оценено снизу как $C_p = N^k$. Например, если множество символов пароля образуют строчные латинские буквы, а минимальная длина пароля равна 3, то $C_p = 26^3 = 17\,576$ (что совсем немного для программного подбора). Если же множество символов пароля состоит из строчных и прописных латинских букв, а также из цифр и минимальная длина пароля равна 6, то $C_p = 62^6 = 56\,800\,235\,584$.

Сложность выбираемых пользователями КС паролей должна устанавливаться администратором при реализации установленной для данной системы политики безопасности. Другими параметрами политики учетных записей при использовании парольной аутентификации должны быть:

- максимальный срок действия пароля (любой секрет не может сохраняться в тайне вечно);
- несовпадение пароля с логическим именем пользователя, под которым он зарегистрирован в КС;
- неповторяемость паролей одного пользователя.

Требование неповторяемости паролей может быть реализовано двумя способами. Во-первых, можно установить минимальный срок действия пароля (в противном случае пользователь, вынужденный после истечения срока действия своего пароля поменять его, сможет тут же сменить пароль на старый). Во-вторых, можно вести список уже использовавшихся данным пользователем паролей (максимальная длина списка при этом может устанавливаться администратором).

К сожалению, обеспечить реальную уникальность каждого вновь выбираемого пользователем пароля с помощью приведенных выше

мер практически невозможно. Пользователь может, не нарушая установленных ограничений, выбирать пароли «A1», «A2», ... где A — первый пароль пользователя, удовлетворяющий требованиям сложности.

Обеспечить приемлемую степень сложности паролей и их реальную уникальность можно путем назначения паролей всем пользователям администратором КС с одновременным запретом на изменение пароля самим пользователем. Для генерации паролей администратор при этом может использовать программный генератор, позволяющий создавать пароли различной сложности (пример окна настройки одного из подобных генераторов приведен на рис. 2.1).

Однако при таком способе назначения паролей возникают проблемы, связанные с необходимостью создания защищенного канала для передачи пароля от администратора к пользователю, трудностью проверки сохранения пользователем не им выбранного пароля только в своей памяти и потенциальной возможностью администратора, знающего пароли всех пользователей, злоупотребления своими полномочиями. Поэтому наиболее целесообразным является выбор пароля пользователем на основе установленных администратором правил с возможностью задания администратором нового пароля пользователю в случае, если тот забыл свой пароль.

Еще одним аспектом политики учетных записей пользователей КС должно стать определение противодействия системы попыткам подбора паролей.

Могут применяться следующие правила:

- ограничение числа попыток входа в систему;

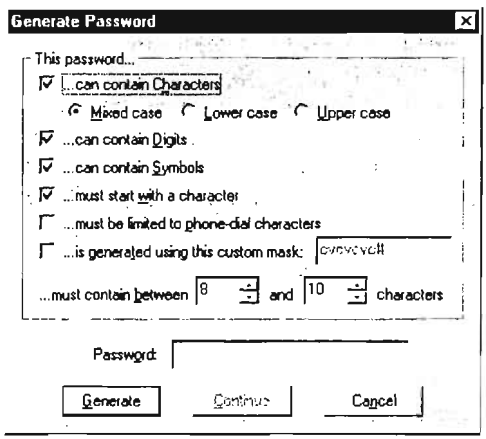

Рис. 2.1. Пример окна настройки программного генератора паролей

• скрытие логического имени последнего работавшего пользователя (знание логического имени может помочь нарушителю подобрать или угадать его пароль);

• учет всех попыток (успешных и неудачных) входа в систему в журнале аудита.

Реакцией системы на неудачную попытку входа пользователя могут быть:

• блокировка учетной записи, под которой осуществляется попытка входа, при превышении максимально возможного числа попыток (на заданное время или до ручного снятия блокировки администратором);

• нарастающее увеличение временной задержки перед предоставлением пользователю следующей попытки входа.

Постоянная блокировка учетной записи при обнаружении попытки подбора пароля (до снятия блокировки администратором) менее целесообразна, поскольку она позволит нарушителю намеренно заблокировать работу в КС легального пользователя (реализовать угрозу нарушения доступности информации).

При любой реакции системы на попытку подбора пароля необходимо в настройках параметров политики учетных записей обеспечить сброс значения счетчика попыток входа в систему под конкретной учетной записью через заданный промежуток времени, иначе значения счетчика будут суммироваться для разных сеансов работы пользователя.

При первоначальном вводе или смене пароля пользователя обычно применяются два классических правила:

• символы вводимого пароля не отображаются на экране (это же правило применяется и для ввода пользователем пароля при его входе в систему);

• для подтверждения правильности ввода пароля (с учетом первого правила) этот ввод повторяется дважды.

Одним из следствий первого правила является нецелесообразность назначения пользователю пароля системой, поскольку в этом случае пароль должен быть выведен пользователю в открытом виде или записан на специальном носителе (второй способ противоречит принципу сохранения пароля только в памяти пользователя).

Однако отказ от отображения символов вводимого пароля может создать проблему, так как увеличивается вероятность того, что случайная ошибка, допущенная при вводе пароля, останется не замеченной, а это может привести к блокировке учетной записи легального пользователя. Поэтому, если вход пользователя в КС происходит в защищенном помещении, в которое не могут попасть посторонние лица, от правила скрытия символов вводимого пароля можно и отказаться.

Очевидно, что в базе данных учетных записей пользователей КС пароли не могут храниться в открытом виде (иначе к ним мо-

жет получить доступ как минимум администратор системы). Для хранения паролей возможно их предварительное шифрование или хеширование.

Шифрование паролей имеет два недостатка:

- поскольку при шифровании необходимо использовать ключ, требуется обеспечить его защищенное хранение в КС (знание ключа шифрования пароля позволит выполнить его расшифрование и осуществить несанкционированный доступ к информации);
- существует опасность расшифрования любого пароля и получения его в открытом виде.

Хеширование является необратимым преобразованием и знание хеш-значения пароля не даст нарушителю возможности его получения в открытом виде (он сможет только пытаться подобрать пароль при известной функции хеширования). Поэтому гораздо более безопасным является хранение паролей в хешированном виде. Недостатком является то, что не существует даже теоретической возможности восстановить забытый пользователем пароль.

Несмотря на то, что с помощью применения перечисленных выше правил парольную аутентификацию можно сделать более безопасной, она все-таки остается весьма уязвимой. Для ее усиления могут использоваться так называемые одноразовые пароли. Пусть пользователь КС получает список паролей P_1, P_2, ..., P_i, ..., P_n. Каждый из паролей действует только на один сеанс входа (P_1 — на первый, P_2 — на второй и т.д.). В этом случае знание уже использовавшегося пользователем пароля ничего не даст нарушителю, а при каждом входе легального пользователя возможна проверка на использование данного пароля кем-либо еще.

Недостатки схемы одноразовых паролей:

- организация защищенного хранения длинного списка паролей (либо его запоминание, что маловероятно);
- неясность с номером следующего пароля, если после ввода предыдущего пароля из списка вход пользователя в систему не был осуществлен из-за сбоя в работе КС.

Эти недостатки могут быть устранены, если список паролей генерировать на основе некоторой необратимой функции, например функции хеширования. Пусть P — начальный пароль пользователя, а F — необратимая функция. Обозначим: $F^i(P) = F(F(...F(P)...))$ (функция F применяется последовательно i раз). Тогда список одноразовых паролей создается следующим образом: $P_1 = F^n(P)$, $P_2 = F^{n-1}(P)$, ..., $P_{n-1} = F(F(P))$, $P_n = F(P)$.

При сбое в процессе входа пользователя в КС всегда осуществляется выбор следующего пароля из списка, а система последовательно применяет функцию F к введенному пользователем паролю, вплоть до совпадения с последним принятым от него паролем (и тогда пользователь допускается к работе в системе) или до

превышения длины списка паролей (в этом случае попытка входа пользователя в КС отвергается).

Но в любом варианте парольной аутентификации подтверждение подлинности пользователя осуществляется на основе ввода им некоторой конфиденциальной информации, которую можно подсмотреть, выманить, подобрать, угадать и т.п. Рассмотрим аутентификацию пользователей на основе модели «рукопожатия», во многом свободную от указанных недостатков.

В соответствии с этой моделью пользователь П и система С согласовывают при регистрации пользователя в КС функцию f, известную только им. Протокол аутентификации пользователя в этом случае выглядит следующим образом:

1. С: генерация случайного значения x; вычисление $y = f(x)$; вывод x.
2. П: вычисление $y' = f'(x)$; ввод y'.
3. С: если y и y' совпадают, то пользователь допускается к работе в системе, иначе попытка входа в систему отклоняется.

К функции f предъявляется требование, чтобы по известным x и $f(x)$ нельзя было угадать f.

Преимущества аутентификации на основе модели «рукопожатия» перед парольной аутентификацией:

- между пользователем и системой не передается никакой конфиденциальной информации, которую нужно сохранять в тайне;
- каждый следующий сеанс входа пользователя в систему отличен от предыдущего, поэтому даже длительное наблюдение за этими сеансами ничего не даст нарушителю.

К недостаткам аутентификации на основе модели «рукопожатия» относится большая длительность этой процедуры по сравнению с парольной аутентификацией.

Парольная аутентификация совершенно неприменима в случае взаимного подтверждения подлинности пользователей компьютерной сети. Действительно, пусть А и Б обозначают двух пользователей сети, имеющих соответственно пароли $Р_А$ и $Р_Б$. Тогда протокол взаимной аутентификации А и Б мог бы выглядеть следующим образом:

1. А→Б: А, запрос $Р_Б$.
2. Б→А: Б, запрос $Р_А$.
3. А→Б: А, $Р_А$.
4. Б→А: Б, $Р_Б$.

Но в момент отправки своего пароля (неважно, в открытой или защищенной форме) А не может быть уверен в подлинности Б, который может воспользоваться паролем А, чтобы выдать себя за А при взаимодействии еще с одним пользователем компьютерной сети В.

Модель «рукопожатия» вполне приемлема для взаимной аутентификации:

1. А: выбор значения x; вычисление $y = f(x)$.
2. А→Б: А, x.
3. Б: вычисление $y' = f(x)$.
4. Б→А: Б, y'.
5. А: если y и y' совпадают, то А может доверять Б.

Затем процедура аутентификации повторяется с переменой ролей (теперь Б начинает процесс и выбирает значение x), чтобы Б мог быть также уверен в подлинности А.

Для повышения безопасности протокола взаимной аутентификации перед отправкой по сети значения x и y' (пп. 2 и 4 протокола) могут быть зашифрованы на секретном ключе, которым должны предварительно обменяться по защищенному каналу А и Б. В этом случае потенциальному нарушителю, который имеет возможность перехвата всех передаваемых по сети данных и желает выдать себя за одного из легальных пользователей сети, придется не только определить функцию f, но и предварительно взломать шифротекст.

При интерактивном доступе пользователя к системе функция f может быть задана таблицей своих значений. Рассмотрим два примера. В первом примере система предлагает пользователю ответить при регистрации его в КС на несколько вопросов, имеющих частично объективное и частично вымышленное содержание (например: «девичья фамилия Вашей матери», «в каком городе Вы проживали в июне 2002 г.», «где находится клуб», «когда откроется пул» и т. п.). При входе в систему пользователю предлагается ответить на другой список вопросов, среди которых есть некоторые из заданных ему при регистрации. Для правильной аутентификации пользователь должен дать те же ответы, которые он давал на аналогичные вопросы при регистрации.

Второй пример — аутентификация на основе модели «рукопожатия». При регистрации в КС пользователю предлагается набор небольших изображений (например, пиктограмм), среди которых он должен выбрать заданное число картинок. При последующем входе в систему ему выводится другой набор изображений, часть из которых он видел при регистрации. Для правильной аутентификации пользователь должен отметить те картинки, которые он выбрал при регистрации.

2.3. Аутентификация пользователей по их биометрическим характеристикам, клавиатурному почерку и росписи мышью

К основным биометрическим характеристикам пользователей КС, которые могут применяться при их аутентификации, относятся:

- отпечатки пальцев;
- геометрическая форма руки;
- узор радужной оболочки глаза;
- рисунок сетчатки глаза;
- геометрическая форма и размеры лица;
- тембр голоса;
- геометрическая форма и размеры уха и др.

Наиболее распространенными являются программно-аппаратные средства аутентификации пользователей по их отпечаткам пальцев. Для считывания этих отпечатков обычно применяются оснащенные специальными сканерами клавиатуры и мыши. Наличие достаточно больших банков данных с отпечатками пальцев граждан является основной причиной достаточно широкого применения подобных средств аутентификации в государственных структурах, а также в крупных коммерческих организациях. Недостатком таких средств является потенциальная возможность применения отпечатков пальцев пользователей для контроля над их частной жизнью.

Если по объективным причинам (например, из-за загрязненности помещений, в которых проводится аутентификация) получение четкого отпечатка пальца невозможно, то может применяться аутентификация по геометрической форме руки пользователя. В этом случае сканеры могут быть установлены на стене помещения.

Наиболее достоверными (но и наиболее дорогостоящими) являются средства аутентификации пользователей, основанные на характеристиках глаза (узоре радужной оболочки или рисунке сетчатки). Вероятность повторения этих признаков оценивается в 10^{-78}.

Наиболее дешевыми (но и наименее достоверными) являются средства аутентификации, основанные на геометрической форме и размере лица пользователя или на тембре его голоса. Это позволяет использовать эти средства и для аутентификации при удаленном доступе пользователей к КС.

Основные достоинства аутентификации пользователей по их биометрическим характеристикам:

- трудность фальсификации этих признаков;
- высокая достоверность аутентификации из-за уникальности таких признаков;
- неотделимость биометрических признаков от личности пользователя.

Для сравнения аутентификации пользователей на основе тех или иных биометрических характеристик применяются оценки вероятностей ошибок первого и второго рода. Вероятность ошибки первого рода (отказа в доступе к КС легальному пользователю) составляет $10^{-6} \ldots 10^{-3}$. Вероятность ошибки второго рода (допуска к работе в

КС незарегистрированного пользователя) в современных системах биометрической аутентификации составляет $10^{-5} \ldots 10^{-2}$.

Общим недостатком средств аутентификации пользователей КС по их биометрическим характеристикам является их более высокая стоимость по сравнению с другими средствами аутентификации, что обусловлено, в первую очередь, необходимостью приобретения дополнительных аппаратных средств. Способы аутентификации, основанные на особенностях клавиатурного почерка и росписи мышью пользователей, не требуют применения специальной аппаратуры.

Одним из первых идею аутентификации пользователей по особенностям их работы с клавиатурой и мышью предложил С.П.Расторгуев. При разработке математической модели аутентификации на основе клавиатурного почерка пользователей было сделано предположение, что временные интервалы между нажатиями соседних символов ключевой фразы и между нажатиями конкретных сочетаний клавиш в ней подчиняются нормальному закону распределения. Сутью данного способа аутентификации является проверка гипотезы о равенстве центров распределения двух нормальных генеральных совокупностей (полученных при настройке системы на характеристики пользователя и при его аутентификации).

Рассмотрим вариант аутентификации пользователя по набору ключевой фразы (одной и той же в режимах настройки и подтверждения подлинности).

Процедура настройки на характеристики регистрируемого в КС пользователя:

1) выбор пользователем ключевой фразы (ее символы должны быть равномерно разнесены по клавиатуре);

2) набор ключевой фразы несколько раз;

3) исключение грубых ошибок (по специальному алгоритму);

4) расчет и сохранение оценок математических ожиданий, дисперсий и числа наблюдений для временных интервалов между наборами каждой пары соседних символов ключевой фразы.

Процедура аутентификации пользователя может проводиться в двух вариантах. Первый вариант процедуры аутентификации:

1) набор ключевой фразы пользователем несколько раз;

2) исключение грубых ошибок (по специальному алгоритму);

3) расчет оценок математических ожиданий и дисперсий для временных интервалов между нажатиями каждой пары соседних символов ключевой фразы;

4) решение задачи проверки гипотезы о равенстве дисперсий двух нормальных генеральных совокупностей для каждой пары соседних символов ключевой фразы (по специальному алгоритму);

5) если дисперсии равны, то решение задачи проверки гипотезы о равенстве центров распределения двух нормальных гене-

ральных совокупностей при неизвестной дисперсии для каждой пары соседних символов ключевой фразы (по специальному алгоритму);

6) вычисление вероятности подлинности пользователя как отношения числа сочетаний соседних клавиш, для которых подтверждены гипотезы (пп. 4 и 5), к общему числу сочетаний соседних символов ключевой фразы;

7) сравнение полученной оценки вероятности с выбранным пороговым значением для принятия решения о допуске пользователя.

Второй вариант процедуры аутентификации:

1) набор ключевой фразы один раз;

2) решение задачи проверки гипотезы о равенстве дисперсий двух нормальных генеральных совокупностей для временных интервалов между нажатиями соседних символов ключевой фразы;

3) если дисперсии равны, то исключение временных интервалов между нажатиями соседних символов ключевой фразы, которые существенно отличаются от эталонных (полученных при настройке);

4) вычисление вероятности подлинности пользователя как отношения числа оставшихся интервалов к общему числу интервалов в ключевой фразе;

5) сравнение полученной оценки вероятности с выбранным пороговым значением для принятия решения о допуске пользователя.

Вместо использования постоянной для пользователя КС ключевой фразы можно проводить аутентификацию с помощью набора псевдослучайного текста. В этом случае клавиатура разделяется на поля и вводится понятие расстояния d_{ij} между клавишами i и j, под которым понимается число клавиш, расположенных на соединяющей i и j прямой линии. Клавиша i принадлежит полю m, если $\forall j \in m \; d_{ij} \le k$.

Величину k назовем степенью поля m (если $k = 0$, то m — отдельная клавиша). Обозначим через x_{ij} временной интервал между нажатиями клавиш, принадлежащих полям i и j.

Введем следующие допущения:

- характеристики нажатия клавиш одного поля тем ближе друг к другу, чем меньше k;
- для пользователя, работающего двумя руками, получение характеристик клавиатурного почерка возможно с помощью исследования работы только с одной половиной клавиатуры;
- ключевой фразой может быть любой набор символов;
- число полей должно быть одним и тем же в режимах настройки и аутентификации.

Процедура настройки при наборе псевдослучайного текста:

1) генерация и вывод пользователю текста из фиксированного множества слов, символы которых максимально разбросаны по клавиатуре;

2) набор текста пользователем;

3) фиксация и сохранение значений x_{ij}, которые затем используются для расчета статистических характеристик клавиатурного почерка.

Процедура аутентификации совпадает с процедурой аутентификации, используемой при наборе ключевой фразы.

Достоверность аутентификации на основе клавиатурного почерка пользователя ниже, чем при использовании его биометрических характеристик.

Однако этот способ аутентификации имеет и свои преимущества:

- возможность скрытия факта применения дополнительной аутентификации пользователя, если в качестве ключевой фразы используется вводимая пользователем парольная фраза;
- возможность реализации данного способа только с помощью программных средств (снижение стоимости средств аутентификации).

Теперь рассмотрим способ аутентификации, основанный на росписи мышью (с помощью этого манипулятора, естественно, нельзя выполнить реальную роспись пользователя, поэтому данная роспись будет достаточно простым росчерком). Назовем линией росписи ломаную линию, полученную соединением точек от начала росписи до ее завершения (соседние точки при этом не должны иметь одинаковых координат). Длину линии росписи рассчитаем как сумму длин отрезков, соединяющих точки росписи.

Введем понятие разрыва в линии росписи, признаком которого будет выполнение условия

$$d_{i,\,i-1} > \frac{d}{k},$$

где $d_{i,\,i-1}$ — расстояние между двумя соседними точками линии росписи; d — длина всей линии; k — число точек в линии.

Для устранения разрывов в линии росписи С. П. Расторгуевым предложен алгоритм ее сглаживания, состоящий в добавлении в линию в точках ее разрывов дополнительных точек. Каждая дополнительная точка a с координатами x_a и y_a, добавляемая между точками $i - 1$ и i линии росписи, должна удовлетворять условию

$$\min\,(d_{i-1,\,a} + d_{a,\,i}) \;\forall\; a\; d_{i-1,\,a} \le \sqrt{2}.$$

По сглаженной линии росписи можно выделить все замкнутые контуры в ней (по специальному алгоритму).

Процедура настройки на характеристики пользователя может состоять из следующих этапов:

1) ввод нескольких эталонных росписей;

2) для каждой росписи получение числа точек в ней и длины ее линии, определение числа и местоположения разрывов в линии росписи;

3) для каждой линии росписи выполнение сглаживания, получение числа и местоположения замкнутых контуров;

4) расчет среднего значения полученных характеристик росписи и их допустимых отклонений.

Процедура аутентификации состоит из следующих этапов:

1) ввод росписи;

2) расчет числа точек и длины линии росписи;

3) получение числа и местоположения разрывов в линии росписи;

4) сглаживание линии росписи;

5) получение числа и местоположения замкнутых контуров;

6) сравнение полученных характеристик росписи с эталонными;

7) принятие решения о допуске пользователя к работе в КС.

Подобно аутентификации на основе клавиатурного почерка подлинность пользователя по его росписи мышью подтверждается прежде всего темпом его работы с этим устройством ввода.

К достоинствам аутентификации пользователей по их росписи мышью, подобно использованию клавиатурного почерка, относится возможность реализации этого способа только с помощью программных средств; к недостаткам — меньшая достоверность аутентификации по сравнению с применением биометрических характеристик пользователя, а также необходимость достаточно уверенного владения пользователем навыками работы с мышью.

Общей особенностью способов аутентификации, основанных на клавиатурном почерке и росписи мышью является нестабильность их характеристик у одного и того же пользователя, которая может быть вызвана:

1) естественными изменениями, связанными с улучшением навыков пользователя по работе с клавиатурой и мышью или, наоборот, с их ухудшением из-за старения организма;

2) изменениями, связанными с ненормальным физическим или эмоциональным состоянием пользователя.

Изменения характеристик пользователя, вызванные причинами первого рода, не являются скачкообразными, поэтому могут быть нейтрализованы изменением эталонных характеристик после каждой успешной аутентификацией пользователя.

Изменения характеристик пользователя, вызванные причинами второго рода, могут быть скачкообразными и привести к отклонению его попытки входа в КС. Однако эта особенность аутентификации на основе клавиатурного почерка и росписи мышью

может стать и достоинством, если речь идет о пользователях КС военного, энергетического и финансового назначения.

Перспективным направлением развития способов аутентификации пользователей КС, основанных на их личных особенностях, может стать подтверждение подлинности пользователя на основе его знаний и навыков, характеризующих уровень образования и культуры.

2.4. Программно-аппаратная защита информации от локального несанкционированного доступа

Отмеченные в подразд. 2.2 недостатки парольной аутентификации пользователей КС могут быть устранены применением так называемой двухфакторной аутентификации, при которой пользователь для входа в систему должен не только ввести пароль, но и предъявить элемент аппаратного обеспечения, содержащий подтверждающую его подлинность ключевую информацию. Такими элементами аппаратного обеспечения могут быть:

- магнитные диски, не требующие установки на компьютере пользователя КС никаких дополнительных аппаратных средств, но наиболее уязвимые с точки зрения копирования хранящейся на них ключевой информации;
- элементы Touch Memory (аналогичные изделия других производителей именуются iButton), включающие в себя энергонезависимую память в виде постоянного запоминающего устройства (ПЗУ) с уникальным для каждого изделия серийным номером и (в более дорогих вариантах) оперативного запоминающего устройства (ОЗУ) для хранения идентифицирующей пользователя информации, а также встроенный элемент питания со сроком службы до 10 лет (элемент Touch Memory напоминает миниатюрную батарейку диаметром 16 мм и толщиной 3...6 мм, он имеет один сигнальный контакт и один контакт заземления, а для контакта элемента с устройством чтения достаточно простого касания);
- пластиковые карты с магнитной полосой, на которой помимо ключевой информации могут размещаться и дополнительные реквизиты пользователя (его фамилия, имя, отчество, фотография, название организации и ее подразделения и т.п.); подобные карты наиболее дешевы, но и наименее защищены от копирования и подделки;
- карты со штрихкодом, покрытым непрозрачным составом, считывание информации с которых происходит в инфракрасных лучах; эти карты также относительно дешевы, но уязвимы для подделки;
- смарт-карты, носителем ключевой информации в которых является специальная бескорпусная микросхема, включающая в

себя только память для хранения ключевой информации (простые смарт-карты) или микропроцессор (интеллектуальные карты), позволяющий реализовывать достаточно сложные процедуры аутентификации;

• маркеры eToken (USB-брелки), представляющие собой подключаемое к USB-порту компьютера устройство, которое включает в себя аналогичную смарт-карте микросхему с процессором и защищенной от несанкционированного доступа памятью (в отличие от пластиковых карт не требуется установка устройства их чтения с кабелем для подключения этого устройства к компьютеру).

С помощью только программных средств принципиально нельзя обеспечить надежную защиту информации от несанкционированного доступа к ней в КС.

Рассмотрим порядок работы программ после включения питания компьютера и до загрузки операционной системы:

1) программа самопроверки устройств компьютера POST (Power On — Self Test);

2) программа BIOS Setup (может быть вызвана пользователем во время выполнения программы POST, обычно для этого необходимо нажать клавишу Delete);

3) программы BIOS;

4) программы расширения BIOS (BIOS Extension), если соответствующая плата установлена на компьютере;

5) программа начальной загрузки, которая размещается в первом секторе нулевой головки нулевого цилиндра жесткого диска компьютера (Master Boot Record, MBR) и в функции которой входят определение активного раздела жесткого диска и вызов программы загрузки операционной системы;

6) программа загрузки операционной системы, которая размещается в первом секторе активного раздела жесткого диска, загрузочного компакт-диска или загрузочной дискеты;

7) оболочка операционной системы.

Очевидно, что если программа начальной загрузки содержит вредоносный код (программную закладку), то и загруженная затем операционная система будет фактически функционировать под управлением программы нарушителя. Кроме того, если нарушитель получит доступ к коду процедуры хеширования идентифицирующей информации пользователя и данным с хеш-значением этой информации, то он сможет подобрать пароль любого пользователя КС и осуществить несанкционированный доступ к информации. Поэтому для гарантированной работы программно-аппаратного средства защиты от несанкционированной загрузки операционной системы достаточно, чтобы программа защиты и хеш-значения паролей пользователей были аппаратно защищены от чтения программными средствами во время сеанса работы пользователя.

Рассмотрим вариант комплекса программно-аппаратных средств для защиты от локального несанкционированного доступа к информации в КС.

Определим модель (возможности) нарушителя:

- установка системы защиты производится в его отсутствие;
- нарушитель не может вскрыть системный блок компьютера;
- нарушитель не может перезаписать информацию в ПЗУ BIOS при работающем компьютере;
- нарушитель не имеет пароля установки системы защиты;
- нарушитель не имеет пароля пользователя КС;
- нарушитель не имеет копии ключевой информации пользователя, хранящейся в элементе аппаратного обеспечения (например, в элементе Touch Memory).

Выполнение первых двух условий может быть обеспечено только с помощью методов организационной защиты информации (см. подразд. 1.3).

Программные средства системы защиты информации должны быть записаны на плате расширения BIOS, для каждой из которых определен уникальный пароль установки. Установка системы защиты информации производится на компьютере, свободном от вредоносных программ типа закладок и вирусов.

После установки платы расширения BIOS выполняется процедура установки системы защиты информации:

1) после включения питания компьютера программа, записанная на плате расширения BIOS, выдает запрос на ввод пароля;

2) после ввода пароля установки PS (как правило, администратором системы) происходят загрузка операционной системы и запуск собственно программы установки (проверочные функции системы защиты при этом отключаются);

3) по запросу программы установки вводятся пароль пользователя P, ключевая информация с элемента аппаратного обеспечения (например, серийный номер элемента Touch Memory) KI и имена подлежащих проверке системных и пользовательских файлов F_1, F_2, ..., F_n;

4) для каждого указанного файла вычисляется и сохраняется проверочная информация в виде

$$E_k(H(PS,\ P,\ KI,\ F_i)),$$

где E — функция шифрования; k — ключ шифрования; H — функция хеширования.

Рассмотрим процедуру входа пользователя в КС при использовании данной системы защиты:

1) после включения питания компьютера программа на плате расширения BIOS запрашивает пароль пользователя и просит установить элемент аппаратного обеспечения с его ключевой информацией;

2) осуществляется проверка целостности выбранных при установке системы защиты файлов путем вычисления хеш-значения для них по приведенному выше правилу и сравнения с расшифрованными эталонными хеш-значениями;

3) в зависимости от результатов проверки выполняется либо загрузка операционной системы, либо запрос на повторный ввод пароля.

После завершения работы пользователя элемент аппаратного обеспечения с его ключевой информацией изымается из компьютера.

Доступ же к хеш-значению пароля фактически заблокирован, так как программное обеспечение для его вычисления исчезает из адресного пространства компьютера и не может быть прочитано никакими программными средствами без извлечения платы расширения BIOS.

Если у нарушителя нет пароля пользователя или копии элемента аппаратного обеспечения с его ключевой информацией, то он не сможет выполнить загрузку операционной системы. Если у нарушителя есть пароль установки системы защиты, что позволит ему загрузить операционную систему без проверочных функций, или он получил доступ к терминалу с уже загруженной операционной системой, то он сможет осуществить несанкционированный доступ (НСД) к информации, но не сможет внедрить программные закладки для постоянного НСД. Наличие у нарушителя пароля установки без знания им пароля пользователя или его ключевой информации не позволит нарушителю переустановить систему защиты для постоянного НСД.

Для защиты от несанкционированного доступа к информации в ситуации, когда нарушитель получил доступ к работающему терминалу, необходимо использовать средства разграничения доступа к ресурсам КС или средства шифрования.

2.5. Аутентификация пользователей при удаленном доступе. Защита информации от несанкционированного доступа в сетях

Простейшим протоколом, который может быть использован для удаленного доступа пользователя к КС, является протокол PAP (Password Authentication Protocol). Пусть С обозначает сервер КС, П — пользователя КС с логическим именем ID и паролем P, а К — удаленный компьютер, с которого пользователь пытается получить доступ к КС при помощи соответствующей клиентской программы.

1. К→С: ID, P′ (запрос аутентификации).

2. С: выборка Р из регистрационной базы данных; сравнение Р и Р′.

3. С→К: если пароли совпадают, то подтверждение аутентификации, в противном случае — отказ в аутентификации и разрыв соединения.

Независимо от формы передачи информации о пароле (в открытом виде, зашифрованном или хешированном) нарушитель может ее перехватить и использовать для несанкционированного доступа к информации в КС с помощью «маскарада». Поэтому протокол PAP может использоваться только совместно с протоколом S/Key.

Идея протокола S/Key основывается на модели одноразовых паролей, получаемых последовательным применением необратимой функции (см. подразд. 2.2). Сервер С инициализирует список из М одноразовых паролей на основе пароля Р пользователя П, вычисляет и сохраняет в регистрационной базе данных проверочное значение $Y_{M+1} = F^{M+1}(P)$. В регистрационной базе данных КС для каждого пользователя также сохраняются значения ID, Р и М, причем только пароль пользователя Р является конфиденциальной информацией.

При очередной аутентификации пользователя удаленный компьютер (клиент) К посылает серверу логическое имя пользователя ID, а сервер в ответ направляет клиенту значение М. Клиент вычисляет значение Y'_M и отправляет его серверу, который вычисляет $Y'_{M+1} = F(Y'_M)$ и сравнивает Y'_{M+1} с извлеченным из базы данных учетных записей значением Y_{M+1}. При совпадении этих значений пользователь допускается к работе в КС, а в его учетной записи значение М уменьшается на единицу, а вместо Y_{M+1} записывается Y'_M.

Для того чтобы можно было сгенерировать новый список одноразовых паролей без личного присутствия пользователя (с удаленного компьютера), а также для повышения безопасности этого списка вычисление одноразовых паролей может быть организовано на базе не только пароля Р, но и генерируемого сервером случайного числа.

Протокол S/Key состоит из двух частей: генерации списка одноразовых паролей (парольной инициализации) и собственно аутентификации.

Рассмотрим процедуру парольной инициализации.

1. С→К: запрос ID.

2. К→С: ID.

3. С: генерация случайного числа (кода инициализации) N.

4. С→К: запрос числа М одноразовых паролей, которые будут использоваться до следующей парольной инициализации (возможно задание этого значения администратором системы, обычно М выбирается в диапазоне 300... 1000).

5. K→C: M.

6. C: по логическому имени пользователя ID извлечение из регистрационной базы данных значения P; вычисление $Y_{M+1} = F^{M+1}(N,P)$; сохранение N, M, Y_{M+1} вместе с ID и P в регистрационной базе данных.

В этом варианте парольной инициализации не требуется передача по сети пароля пользователя P для генерации нового списка одноразовых паролей.

Рассмотрим процедуру аутентификации по протоколу S/Key.

1. K→C: ID.

2. C: извлечение из регистрационной базы данных соответствующих ID значений P, N, M, Y_{M+1}.

3. C→K: N, M.

4. П→K: P′.

5. K: вычисление $Y'_M = F^M(N, P')$.

6. K→C: Y'_M.

7. C: вычисление $Y'_{M+1} = F(Y'_M)$; сравнение Y'_{M+1} и Y_{M+1}; если эти значения совпадают, то пользователь допускается к работе, а в регистрационной базе данных соответствующее ID значение Y_{M+1} заменяется значением Y'_M, а значение M уменьшается на единицу.

Для ускорения процедуры аутентификации некоторое значение одноразовых паролей (например, 50) может быть вычислено на клиентском компьютере заранее, а для сохранения конфиденциальности — сохраняться на этом компьютере в зашифрованном виде с использованием ключа шифрования, равного паролю пользователя P.

Парольная инициализация должна выполняться:

- после назначения или изменения пароля пользователя P;
- после использования для аутентификации последнего пароля из списка (когда M станет равным нулю);
- при вероятной компрометации списка паролей, когда номер пароля, запрашиваемый сервером, меньше номера, ожидаемого клиентом.

Еще одним протоколом удаленной аутентификации пользователей КС является протокол CHAP (Challenge Handshake Authentication Protocol), основанный на модели «рукопожатия». Идеей протокола CHAP является передача клиентом пароля в хешированном виде с использованием полученного от сервера случайного числа.

1. C: генерация случайного числа N.

2. C→K: идентификатор сервера IDS, N и его длина в байтах (вызов).

3. П→K: P′.

4. K: вычисление хеш-значения D′ = H(IDS, N, P′).

5. K→C: ID, D′ (отклик).

6. С: извлечение из регистрационной базы данных соответствующего ID значения P; вычисление хеш-значения D = H(IDS, N, P); сравнение D′ и D.

7. С→К: если значения совпадают, то подтверждение аутентификации, в противном случае — отказ в аутентификации и разрыв соединения.

Используемое в протоколе CHAP значение N должно быть уникальным и непредсказуемым. Если N не уникально, то нарушитель сможет повторно использовать перехваченный им пакет с откликом клиента для несанкционированного доступа к информации на сервере в форме «маскарада». Если значение N предсказуемо, то нарушитель сможет подобрать его и, сформировав пакет с вызовом, послать его клиенту от лица сервера. Полученный от клиента пакет с откликом нарушитель сохраняет для последующей отправки от лица клиента, когда реальный сервер направит клиенту аналогичный пакет с вызовом.

Обычно в качестве N выбирается последовательность битов, представляющая собой значение текущих даты и времени в секундах, к которой присоединяется случайное число, полученное от программного или аппаратного генератора псевдослучайных чисел.

Если в распределенной компьютерной системе имеется несколько серверов, предоставляющих свои сервисы клиентам, то для надежной аутентификации пользователей КС, которые обращаются к ее серверам с различных рабочих станций, и самих серверов может использоваться протокол аутентификации Kerberos. Этот протокол предполагает использование центрального сервера аутентификации СА, в функции которого и входит идентификация серверов и пользователей КС, а также сервера выдачи мандатов (или билетов, ticket) СВМ на доступ пользователей КС к ее серверам.

Для обеспечения конфиденциальности передаваемой по сети информации в протоколе Kerberos используются функция шифрования E и сеансовые (используемые однократно) ключи шифрования для передачи данных между клиентом и сервером выдачи мандатов (Kcts), а также между клиентом и сервером КС с необходимым клиенту сервисом (Kcs). Сервер выдачи мандатов имеет постоянный ключ шифрования Kts, который известен и каждому серверу КС. Каждый из предоставляющих свои сервисы клиентам серверов также имеет постоянный ключ шифрования Ks, который в свою очередь известен серверу выдачи мандатов. Постоянные ключи шифрования серверов КС распределяются в системе по защищенному от несанкционированного доступа каналу.

Для исключения возможности повторного использования нарушителем мандата, выданного легальному пользователю, в протоколе Kerberos в передаваемые по сети мандаты добавляются штампы времени TS и информация о периоде действия мандатов TA (как правило, 8 ч).

Рассмотрим одну из версий протокола Kerberos (IDS — идентификатор сервера КС; IDT — идентификатор сервера выдачи мандатов; T_{ts} — мандат на получение мандата у СВМ; T_s — мандат на получение сервиса у сервера КС; AD — адрес рабочей станции пользователя; A — аутентификатор клиента).

1. K→CA: ID, IDT, TS_1.
2. CA: вычисление $T_{ts} = E_{Kts}(Kcts, ID, AD, IDT, TS_2, TA_2)$.
3. CA→K: $E_P(Kcts, IDT, TS_2, TA_2, T_{ts})$.
4. K: вычисление $A_1 = E_{Kcts}(ID, AD, TS_3)$.
5. K→CBM: IDS, T_{ts}, A_1.
6. CBM: вычисление $T_s = E_{Ks}(Kcs, ID, AD, IDS, TS_4, TA_4)$.
7. CBM→K: $E_{Kcts}(Kcs, ID, TS_4, T_s)$.
8. K: вычисление $A_2 = E_{Kcs}(ID, AD, TS_5)$.
9. K→C: T_s, A_2.
10. C→K: $E_{Kcs}(TS_5 + 1)$.

Пункты 1 ... 3 протокола Kerberos (обмен службы аутентификации) предназначены для получения клиентом мандата на получение мандата, который включает в себя помимо идентификаторов адреса и отметки времени, а также сеансовый ключ для обмена между клиентом и сервером выдачи мандатов. Только клиент, работающий в интересах пользователя и знающий его пароль P, может расшифровать полученное от сервера аутентификации сообщение, содержащее запрашиваемый мандат.

Пункты 4 ... 7 протокола Kerberos (обмен службы выдачи мандатов) предназначены для получения клиентом мандата на получение сервиса. При обращении к серверу выдачи мандатов клиент добавляет к сообщению аутентификатор, который будет использоваться только один раз, в отличие от предполагающего многократное использование мандата, и имеет весьма ограниченный срок действия. Сервер выдачи мандатов может проверить подлинность клиента с помощью аутентификатора, который он расшифровывает при помощи сеансового ключа, извлеченного из мандата на получение мандата.

Пункты 8 ... 10 протокола Kerberos (обмен аутентификации клиента и сервера) предназначены для получения клиентом требуемого сервиса. Вместе с мандатом на получение сервиса клиент высылает серверу и свой аутентификатор. Сервер проверяет подлинность клиента с помощью этого аутентификатора, который он расшифровывает, используя извлеченный из мандата сеансовый ключ. Для подтверждения собственной подлинности перед клиентом сервер отвечает ему сообщением, содержащим увеличенную на единицу отметку времени, которая была извлечена сервером из аутентификатора клиента. Это сообщение зашифровывается на сеансовом ключе, извлеченном из полученного сервером мандата. Это убеждает клиента, что данное сообщение могло быть отправлено только сервером.

Теперь клиент и сервер имеют общий сеансовый ключ, который они могут использовать для обмена зашифрованными сообщениями и обмена новыми сеансовыми ключами.

К достоинствам протокола Kerberos относятся:

- более быстрое подсоединение клиента к серверу, так как серверу не требуется обращаться к серверу аутентификации для подтверждения подлинности клиента, что приводит к улучшению масштабируемости распределенной КС;
- возможность делегирования клиентом своих полномочий серверу для выполнения запроса;
- упрощение администрирования распределенной КС.

Основные причины, облегчающие нарушителю реализацию угроз безопасности информации в распределенных КС:

- отсутствие выделенного канала связи между объектами распределенной КС (наличие широковещательной среды передачи данных, например среды Ethernet), что позволяет нарушителю анализировать сетевой трафик в подобных системах;
- возможность взаимодействия объектов распределенной КС без установления виртуального канала между ними, что не позволяет надежно идентифицировать объект или субъект распределенной КС и организовать защиту передаваемой информации;
- использование недостаточно надежных протоколов идентификации объектов распределенной КС перед установлением виртуального канала между ними, что позволяет нарушителю при перехвате передаваемых сообщений выдать себя за одну из сторон соединения;
- отсутствие контроля создания и использования виртуальных каналов между объектами распределенной КС, что позволяет нарушителю добиться реализации угрозы отказа в обслуживании в КС (например, любой объект распределенной КС может анонимно послать любое число сообщений от имени других объектов КС);
- отсутствие возможности контроля маршрута получаемых сообщений, что не позволяет подтвердить адрес отправителя данных и определить инициатора удаленной атаки на КС;
- отсутствие полной информации об объектах КС, с которыми требуется создать соединение, что приводит к необходимости отправки широковещательного запроса или подключения к поисковому серверу (нарушитель при этом имеет возможность внедрения ложного объекта в распределенную КС и выдать один из ее объектов за другой);
- отсутствие шифрования передаваемых сообщений, что позволяет нарушителю получить несанкционированный доступ к информации в распределенной КС.

Выделим основные методы создания безопасных распределенных КС:

- использование выделенных каналов связи путем физического соединения каждой пары объектов распределенной КС или при-

менения топологии «звезда» и сетевого коммутатора, через который осуществляется связь между объектами;

• разработка дополнительных средств идентификации объектов распределенной КС перед созданием виртуального канала связи между ними и применение средств шифрования передаваемой по этому каналу информации;

• контроль маршрута поступающих сообщений;

• контроль создания и использования виртуального соединения между объектами распределенной КС (например, ограничение числа запросов от одного из объектов и разрыв соединения после истечения определенного интервала времени);

• разработка распределенной КС с полной информацией об ее объектах, если это возможно, или организация взаимодействия между объектом КС и поисковым сервером только с созданием виртуального канала.

Среди программно-аппаратных и программных средств обеспечения информационной безопасности распределенных КС можно выделить межсетевые экраны (МСЭ), средства анализа защищенности и средства обнаружения атак.

Межсетевые экраны (брандмауэры, firewall) реализуют набор правил, которые определяют условия прохождения пакетов данных из одной части распределенной КС (открытой) в другую (защищенную). Обычно межсетевые экраны устанавливаются между сетью Интернет и локальной вычислительной сетью организации (рис. 2.2), хотя они могут размещаться и внутри корпоративной сети. В зависимости от уровня взаимодействия объектов сети основными разновидностями МСЭ являются фильтрующие маршрутизаторы, шлюзы сеансового и прикладного уровней. Как правило, в состав МСЭ включаются компоненты, соответствующие двум или всем трем указанным разновидностям.

Основной функцией фильтрующих маршрутизаторов, работающих на сетевом уровне эталонной модели, является фильтрация

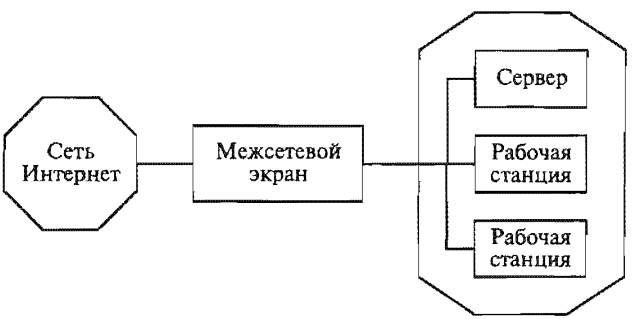

Рис. 2.2. Пример размещения межсетевого экрана

пакетов данных, входящих в защищенную часть сети или исходящих из нее.

При фильтрации используется информация из заголовков пакетов:

- IP-адрес отправителя пакета;
- IP-адрес получателя пакета;
- порт отравителя пакета;
- порт получателя пакета;
- тип протокола;
- флаг фрагментации пакета.

Под портом понимается числовой идентификатор (от 0 до 65 535), используемый клиентской и серверной программами для отправки и приема сообщений.

Правила фильтрации определяют, разрешается или блокируется прохождение через МСЭ пакета с задаваемыми этими правилами параметрами. На рис. 2.3 и 2.4 приведен пример создания такого правила. К основным достоинствам фильтрующих маршрутизаторов относятся простота их создания, установки и конфигурирования; прозрачность для приложений пользователей КС и минимальное влияние на их производительность; невысокая стоимость.

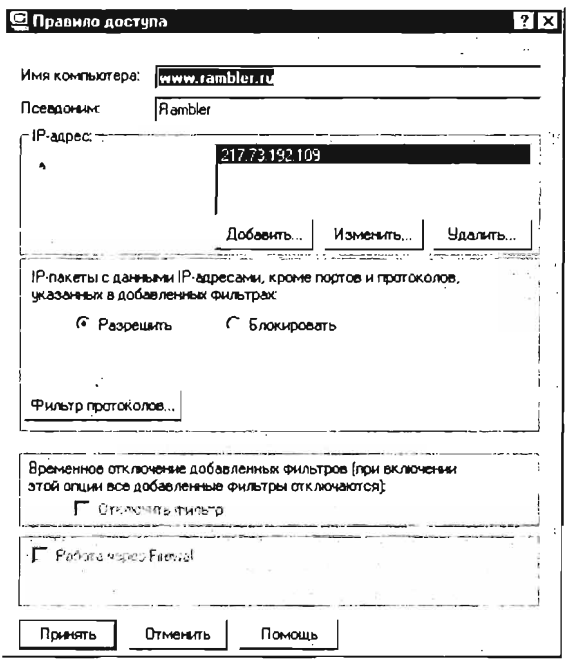

Рис. 2.3. Пример создания правила фильтрации пакетов для межсетевого экрана

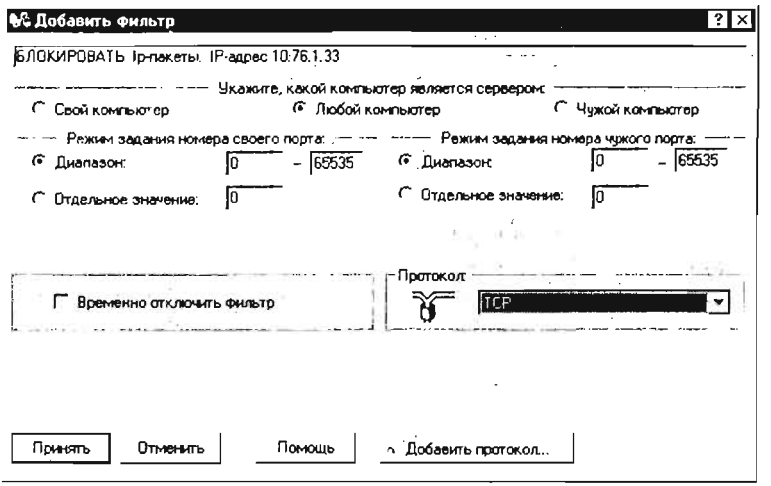

Рис. 2.4. Добавление информации о протоколе и порте в правило фильтрации

Недостатки фильтрующих маршрутизаторов:

- отсутствие аутентификации на уровне пользователей КС;
- уязвимость для подмены IP-адреса в заголовке пакета;
- незащищенность от угроз нарушения конфиденциальности и целостности передаваемой информации;
- сильная зависимость эффективности набора правил фильтрации от уровня знаний администратора МСЭ конкретных протоколов;
- открытость IP-адресов компьютеров защищенной части сети.

Шлюзы сеансового уровня выполняют две основные функции:

- контроль виртуального соединения между рабочей станцией защищенной части сети и хостом ее незащищенной части;
- трансляцию IP-адресов компьютеров защищенной части сети.

Шлюз сеансового уровня устанавливает соединение с внешним хостом от имени авторизованного клиента из защищенной части сети, создает виртуальный канал по протоколу TCP, после чего копирует пакты данных в обоих направлениях без их фильтрации. Когда сеанс связи завершается, МСЭ разрывает установленное соединение с внешним хостом.

В процессе выполняемой шлюзом сеансового уровня процедуры трансляции IP-адресов компьютеров защищенной части сети происходит их преобразование в один IP-адрес, ассоциированный с МСЭ. Это исключает прямое взаимодействие между хостами защищенной и открытой сетей и не позволяет нарушителю осуществлять атаку путем подмены IP-адресов.

К достоинствам шлюзов сеансового уровня относятся также их простота и надежность программной реализации; к недостаткам —

отсутствие возможности проверять содержимое передаваемой информации, что позволяет нарушителю пытаться передать пакеты с вредоносным программным кодом через подобный МСЭ и обратиться затем напрямую к одному из серверов (например, Web-серверу) атакуемой КС.

Шлюзы прикладного уровня не только исключают прямое взаимодействие между авторизованным клиентом из защищенной части сети и хостом из ее открытой части, но и фильтруют все входящие и исходящие пакеты данных на прикладном уровне (на основе анализа содержания передаваемых данных).

Основные функции шлюзов прикладного уровня:

- идентификация и аутентификация пользователя КС при попытке установить соединение;
- проверка целостности передаваемых данных;
- разграничение доступа к ресурсам защищенной и открытой частей распределенной КС;
- фильтрация и преобразование передаваемых сообщений (обнаружение вредоносного программного кода, шифрование и расшифрование и т. п.);
- регистрация событий в специальном журнале;
- кэширование запрашиваемых извне данных, размещенных на компьютерах внутренней сети (для повышения производительности КС).

Шлюзы прикладного уровня позволяют обеспечить наиболее высокую степень защиты КС от удаленных атак, поскольку любое взаимодействие с хостами открытой части сети реализуется через программы-посредники, которые полностью контролируют весь входящий и исходящий трафик.

Достоинствами шлюзов прикладного уровня также являются:

- скрытость структуры защищенной части сети для остальных хостов (доменное имя компьютера со шлюзом прикладного уровня может быть единственным известным внешним серверам именем);
- надежная аутентификация и регистрация проходящих сообщений;
- более простые правила фильтрации пакетов на сетевом уровне, в соответствии с которыми маршрутизатор должен пропускать только трафик, предназначенный для шлюза прикладного уровня, и блокировать весь остальной трафик;
- возможность реализации дополнительных проверок, что уменьшает вероятность использования ошибок в стандартном программном обеспечении для реализации угроз безопасности информации в КС.

Основными недостатками шлюзов прикладного уровня являются более высокая стоимость, сложность разработки, установки и конфигурирования, снижение производительности КС, «непрозрачность» для приложений пользователей КС.

Межсетевые экраны являются основой для создания виртуальных частных сетей (Virtual Private Network, VPN), которые предназначены для скрытия топологии внутренних сетей организаций, обменивающихся информацией по сети Интернет, и защиты трафика между ними. При этом используются специальные системы маршрутизации.

Общим недостатком МСЭ любого вида является то, что эти программно-аппаратные средства защиты в принципе не могут предотвратить многих видов атак (например, угрозы несанкционированного доступа к информации с использованием ложного сервера службы доменных имен сети Интернет, угрозы анализа сетевого трафика, угрозы отказа в обслуживании). Нарушителю реализовать угрозу доступности информации в КС, использующей МСЭ, может оказаться даже проще, так как достаточно атаковать только хост с МСЭ для фактического отключения от внешней сети всех компьютеров защищенной части сети.

В руководящем документе Гостехкомиссии России «Средства вычислительной техники. Межсетевые экраны. Защита от несанкционированного доступа к информации. Показатели защищенности от несанкционированного доступа к информации» установлено пять классов защищенности МСЭ (наиболее защищенным является первый класс). Например, для пятого класса защищенности требуется фильтрация пакетов на сетевом уровне на основе IP-адресов отправителя и получателя, а для второго класса — фильтрация на сетевом, транспортном и прикладном уровнях со скрытием субъектов и объектов защищаемой сети и трансляцией сетевых адресов.

Рассмотрим пример возможного применения программно-аппаратных средств защиты от угроз безопасности информации, передаваемой по протоколу TCP/IP через сеть Интернет между различными ЛВС одной организации (рис. 2.5).

Определим модель (возможности) нарушителя:

- знает топологию всей сети;

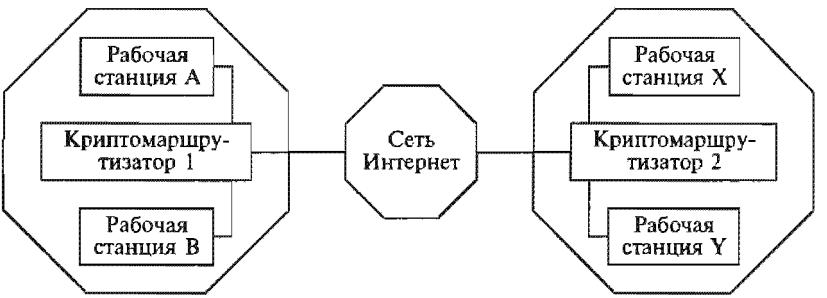

Рис. 2.5. Защита информации при ее передаче по открытой сети Интернет

• знает IP-адреса хостов защищаемых подсетей (ЛВС организации);

• имеет образцы программного и аппаратного обеспечения установленных в подсетях рабочих станций и серверов;

• владеет сведениями о внедренных в стандартное программное обеспечение рабочих станций и серверов закладках, случайно или намеренно оставленной отладочной информации, ключах шифрования и т. п.;

• не знает сеансовых и базовых ключей шифрования, используемых специализированными криптомаршрутизаторами;

• не имеет возможности осуществлять локальный несанкционированный доступ к информации.

Характеристики программно-аппаратного средства защиты — криптомаршрутизатора:

• физическое разделение внешних (с сетью Интернет) и внутренних (с хостами обслуживаемой подсети) интерфейсов (например, с помощью двух разных сетевых карт);

• возможность шифрования всех исходящих (в другие ЛВС организации) и расшифрования всех входящих (из этих ЛВС) пакетов данных.

Обозначим через CR_1 и CR_2 криптомаршрутизаторы 1 и 2 соответственно, а через AD(A), AD(X), AD(CR_1) и AD(CR_2) — IP-адреса рабочих станций и криптомаршрутизаторов.

Алгоритм работы криптомаршрутизатора CR_1 при передаче пакета данных от рабочей станции A к рабочей станции X:

1) по таблице маршрутов ищется адрес криптомаршрутизатора, который обслуживает подсеть, содержащую получателя пакета (AD(CR_2));

2) определяется интерфейс, через который доступна подсеть, содержащая CR_2;

3) выполняется шифрование всего пакета от A (вместе с его заголовком) на сеансовом ключе связи CR_1 и CR_2, извлеченном из таблицы маршрутов;

4) к полученным данным добавляется заголовок, содержащий AD(CR_1), в качестве адреса отправителя и AD(CR_2) в качестве адреса получателя пакета;

5) сформированный пакет отправляется через сеть Интернет.

Алгоритм работы криптомаршрутизатора CR_2 при получении пакета для рабочей станции X:

1) из таблицы маршрутов извлекается сеансовый ключ связи CR_1 и CR_2;

2) выполняется расшифрование данных полученного пакета;

3) если после расшифрования структура вложенного пакета некорректна или адрес его получателя не соответствует обслуживаемой CR_2 подсети (не совпадает с AD(X)), то полученный пакет уничтожается;

4) расшифрованный пакет, содержащий AD(A) в поле отправителя и AD(X) в поле получателя, передается X через внутренний интерфейс ЛВС.

В рассмотренном варианте защиты от несанкционированного доступа достигается полная «прозрачность» функционирования криптомаршрутизаторов для работы любого сетевого программного обеспечения, использующего стек протоколов TCP/IP. Обеспечивается скрытость адресного пространства подсетей организации и его независимость от адресов в сети Интернет.

Степень защиты передаваемой информации полностью определяется стойкостью к взлому используемой функции шифрования (если возможности нарушителя укладываются в рамки рассмотренной ранее модели). Пользователи защищаемых подсетей не замечают никакого изменения в работе сети, кроме некоторого замедления за счет шифрования и расшифрования передаваемых пакетов.

При работе с большим числом защищаемых подсетей необходимо выделить специальный криптомаршрутизатор с функциями центра распределения ключей шифрования для связи между парами криптомаршрутизаторов, которые в этом случае могут работать в двух режимах (загрузки конфигурации и основном) и имеют на защищенном носителе один маршрут и один ключ шифрования для связи с центром распределения ключей.

После успешной установки соединения центра распределения ключей с одним из криптомаршрутизаторов ему высылается таблица маршрутов, зашифрованная общим с центром ключом. После получения и расшифрования таблицы маршрутов криптомаршрутизатор переходит в основной режим работы.

Рассмотрим вариант программно-аппаратных средств защиты при удаленном доступе к КС через коммутируемый канал связи посредством модема. Модель вероятного нарушителя:

- возможны попытки локального несанкционированного доступа к информации в КС организации для внедрения закладок в программное обеспечение рабочих станций и серверов ее ЛВС;
- возможен перехват нарушителем всего трафика вне пределов защищенной зоны организации;
- нарушитель имеет образцы аппаратного и программного обеспечения ЛВС организации и всю документацию к нему;
- нарушитель может получить информацию только вне пределов защищенной зоны;
- нарушитель не может создать обходные каналы утечки информации через незащищенную зону;
- нарушитель не имеет ключей шифрования.

В качестве программно-аппаратного средства защиты в данной ситуации может использоваться устройство шифрования информации, управляемое модемом и размещаемое между модемом и

компьютером, с которого осуществляется удаленный доступ к сети. Данное устройство шифрования должно иметь входной и выходной разъемы интерфейса RS-232. Алгоритм работы устройства шифрования:

1) после установления соединения производится синхронизация с подобным устройством на противоположном конце линии связи;

2) после синхронизации устройств шифрования информация передается в компьютер и из него;

3) при сбое синхронизации пп. 1 и 2 повторяются;

4) при отсутствии соединения с КС устройство шифрования отключается, что не позволяет нарушителю оказывать удаленное воздействие на программно-аппаратное обеспечение защищаемой КС.

Основным достоинством рассмотренного варианта защиты информации является полная независимость от используемого программно-аппаратного обеспечения.

К недостаткам можно отнести невозможность организации взаимодействия с абонентами сети, не имеющими аналогичного устройства шифрования.

Основными функциями программных *средств анализа защищенности* КС (сканеров уязвимости, Vulnerability-Assessment) являются:

- проверка используемых в системе средств идентификации и аутентификации, разграничения доступа, аудита и правильности их настроек с точки зрения безопасности информации в КС;
- контроль целостности системного и прикладного программного обеспечения КС;
- проверка наличия известных неустраненных уязвимостей в системных и прикладных программах, используемых в КС, и др.

Средства анализа защищенности работают на основе сценариев проверки, хранящихся в специальных базах данных, и выдают результаты своей работы в виде отчетов, которые могут быть конвертированы в различные форматы.

К недостаткам средств анализа защищенности КС относятся:

- зависимость их от конкретных систем;
- недостаточная надежность (их применение может иногда вызывать сбои в работе анализируемых систем);
- малый срок эффективной эксплуатации (не учитываются новые обнаруженные уязвимости, которые и являются наиболее опасными);
- возможность использования нарушителями в целях подготовки к атаке на КС.

Программные *средства обнаружения атак* (Intrusion Detection Systems, IDS) применяются для решения двух основных задач:

- обнаружение признаков атак на основе анализа журналов безопасности операционной системы, журналов МСЭ и других служб;

• инспекция пакетов данных непосредственно в каналах связи (с использованием мультиагентных систем).

В обоих случаях средствами обнаружения атак используются базы данных с зафиксированными сетевыми событиями и шаблонами известных атак. Эти средства работают в реальном масштабе времени и реагируют на попытки использования известных уязвимостей КС или несанкционированного исследования защищенной части сети организации, а также ведут журнал регистрации зафиксированных событий для последующего анализа.

К основным недостаткам средств обнаружения атак относятся:

• неспособность эффективно функционировать в высокоскоростных сетях;

• возможность пропуска неизвестных атак;

• необходимость постоянного обновления базы данных с шаблонами атак;

• сложность определения реакции этих средств на обнаруженные попытки атаки.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие существуют способы несанкционированного доступа к информации в компьютерных системах?
2. Какие способы аутентификации пользователей могут применяться в компьютерных системах?
3. В чем заключаются основные недостатки парольной аутентификации и как она может быть усилена?
4. В чем сущность, достоинства и недостатки аутентификации на основе модели «рукопожатия»?
5. Какие биометрические характеристики пользователей могут применяться для их аутентификации? В чем преимущества подобного способа подтверждения подлинности?
6. В чем специфика аутентификации пользователей на основе их клавиатурного почерка и росписи мышью?
7. Какие элементы аппаратного обеспечения могут применяться для хранения идентифицирующей информации для пользователей компьютерных систем?
8. Что представляют собой элементы Touch Memory?
9. Что называют двухфакторной аутентификацией?
10. Почему с помощью только программных средств нельзя обеспечить необходимую степень защищенности от локального несанкционированного доступа к информации в компьютерных системах?
11. В чем заключается достаточное условие надежной программно-аппаратной защиты от локального несанкционированного доступа к информации?
12. Что лежит в основе работы протокола S/Key? Что такое парольная инициализация?

13. На чем основан протокол CHAP? Какие требования выдвигаются к используемому в нем случайному числу?

14. Для чего предназначен протокол Kerberos?

15. Какие применяются разновидности межсетевых экранов?

16. Что такое VPN и для чего они предназначены?

17. Каковы общие недостатки всех межсетевых экранов?

18. В чем состоят функции средств анализа защищенности компьютерных систем и каковы их основные недостатки?

19. В чем сущность систем обнаружения атак на компьютерные системы?

ГЛАВА 3

ЗАЩИТА ИНФОРМАЦИИ ОТ НЕСАНКЦИОНИРОВАННОГО ДОСТУПА В ОПЕРАЦИОННЫХ СИСТЕМАХ

3.1. Защита информации от несанкционированного доступа в открытых версиях операционной системы Windows

К открытым версиям операционной системы Windows относятся операционные системы Windows 95/98/ME/XP Home Edition. В этих операционных системах нельзя обеспечить высокую степень защиты от несанкционированного доступа к информации, поскольку эта цель изначально не ставилась перед началом их проектирования. Тем не менее, в силу широкого распространения открытых версий операционных систем Windows не только в домашних КС, но и в КС организаций необходимо уметь правильно использовать все, хотя и скромные, программно-аппаратные средства защиты информации, которые имеются в этих операционных системах.

Для предотвращения угрозы, связанной с несанкционированной загрузкой операционной системы неавторизованным пользователем, можно применить защиту на основе пароля, устанавливаемого программой BIOS Setup с помощью функции Set User Password (или аналогичной ей) при выбранном для параметра Security Option (или аналогичного ему) значении System в окне настроек функции Advanced BIOS Features (или аналогичной ей). В этом случае перед загрузкой операционной системы или при попытке вызвать программу BIOS Setup пользователю будет предложено ввести пароль. Максимальная длина пароля, устанавливаемого программой BIOS Setup, равна восьми символам.

К достоинствам защиты информации от несанкционированного доступа с помощью пароля программы BIOS Setup относятся простота установки и надежность защиты (число попыток ввода пароля перед загрузкой операционной системы ограничено тремя, что делает практически невозможным подбор пароля).

Недостатки защиты информации на основе пароля программы BIOS Setup:

- все пользователи, работающие на этой рабочей станции, получают общий пароль, что не позволяет разграничить их права в системе;
- сложность замены пароля, если он забыт пользователем (в лучшем случае потребуется отключение энергонезависимой CMOS-

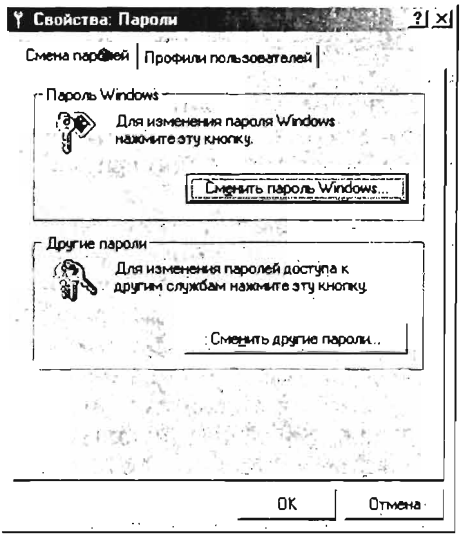

Рис. 3.1. Смена пароля на вход в открытых версиях Windows

памяти компьютера, а в худшем — замена его системной (материнской) платы;

• поскольку пароль, установленной программой BIOS Setup, сохраняется в CMOS-памяти компьютера в незащищенном виде,

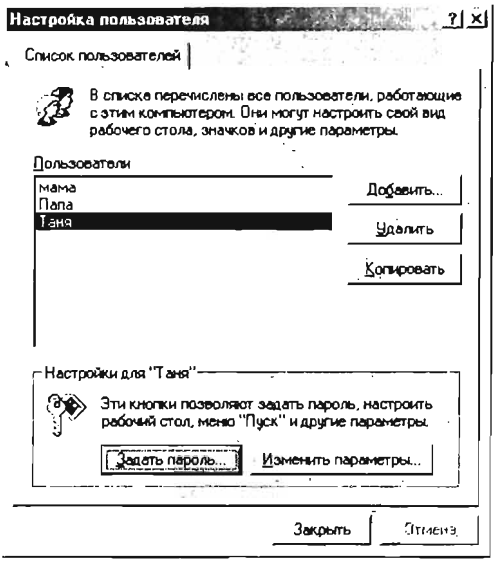

Рис. 3.2. Установка пароля пользователя открытой версии Windows

нарушитель может прочитать его с помощью специальной программы после получения доступа к компьютеру после загрузки операционной системы;

• существование так называемых технических паролей программы BIOS Setup, позволяющих осуществить загрузку операционной системы неавторизованному пользователю, знающему подобный пароль.

В открытых версиях операционной системы Windows могут быть установлены пароли на вход в систему (функции «Пароли» или «Пользователи» панели управления Windows, рис. 3.1 и 3.2). Однако эти пароли фактически служат для разграничения личных профилей пользователей, в которые входят выбранные для каждого из них настройки аппаратного и программного обеспечения КС, структура рабочего стола пользователя, структура его главного меню в системе, список последних использовавшихся им документов и другая информация о настройках и действиях пользователя.

Необходимость использования пользователями своих индивидуальных профилей устанавливается с помощью функции «Пароли» панели управления Windows (рис. 3.3), а информация о месте хранения этих профилей содержится в разделе реестра Windows HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\ProfileList\ЛогическоеИмяПользователя (параметр ProfileImagePath) (рис. 3.4). При подключении компьютера к локальной сети информация о профилях пользователей может храниться в их домашних папках на сервере сети.

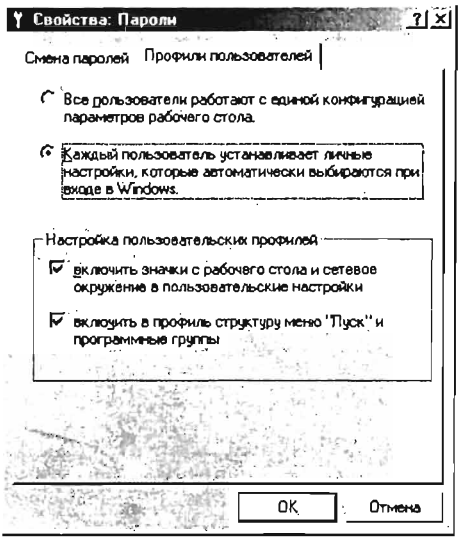

Рис. 3.3. Установка индивидуальных профилей пользователей Windows

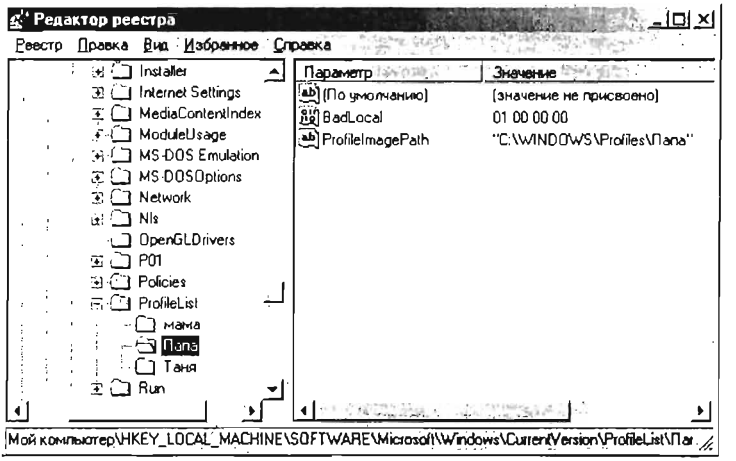

Рис. 3.4. Информация о месте хранения профилей пользователей

Напомним, что в разделах реестра (системной базы данных) Windows HKEY_LOCAL_MACHINE и HKEY_CURRENT_USER хранится информация о настройках аппаратных и программных средств компьютера, которые соответственно являются общими для всех пользователей компьютера или установлены для его текущего пользователя. Просмотр и редактирование содержимого реестра возможны с помощью системной программы regedit (редактор реестра).

В открытых версиях операционной системы Windows пользователь может начать сеанс работы как «Пользователь с профилем по умолчанию» (Default), нажав кнопку «Отмена» в окне входа в систему, или начать сеанс от лица пользователя с произвольным логическим именем с автоматическим созданием для него нового профиля, введя имя и пароль нового пользователя в окне входа в систему. Во втором случае профиль для вновь зарегистрированного пользователя будет создан на основе профиля по умолчанию.

В состав полного набора установочных файлов открытых версий операционной системы Windows входит системная программа poledit (редактор системных правил). После отдельной установки этой программы с помощью функции «Установка и удаление программ» панели управления Windows (вкладка «Установка Windows») редактор системных правил помещается в подменю Программы | Стандартные | Служебные программы меню Пуск.

С помощью редактора системных правил можно ввести определенные ограничения на права конкретного пользователя КС или всех пользователей системы (рис. 3.5). Для конкретного пользователя КС могут быть установлены:

- запрет на выполнение программ в сеансе MS-DOS;

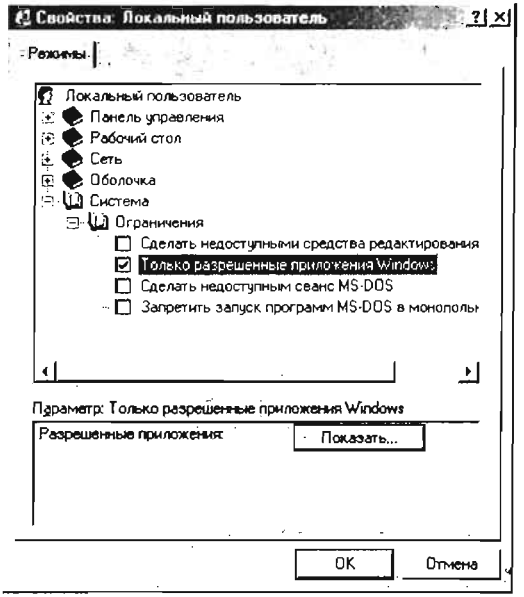

Рис. 3.5. Пример использования редактора системных правил Windows

- возможность запуска только разрешенных приложений из отдельного списка;
- запрет использования средств редактирования реестра;
- запрет использования панели управления или ее отдельных функций;
- удаление команд «Выполнить» и «Поиск» из меню «Пуск»;
- запрет на завершение работы с операционной системой Windows (в том числе на ее перезагрузку);
- запрет на настройку панели задач;
- запрет на отображение структуры локальной сети;
- скрытие дисков в папке «Мой компьютер» и в окне проводника;
- скрытие всех объектов на рабочем столе и др.

Для всех пользователей данного компьютера могут быть установлены:

- отключение кэширования вводимых пользователями паролей (их сохранения на жестком диске компьютера в зашифрованном виде);
- требование усложнения выбираемых пользователями паролей — минимальная длина и (или) включение в их состав букв и цифр;
- проверка пароля на вход в систему на сервере локальной сети;
- запрет совместного доступа к файлам и принтерам компьютера со стороны других компьютеров локальной сети;

• запрет удаленного доступа к этому компьютеру и др.

Информация об установленных ограничениях сохраняется в реестре Windows в разделах HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies (рис. 3.6) и HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Policies. Поэтому обязательным является запрет на использование средств редактирования реестра непривилегированными пользователями, а программа для редактирования системных правил должна запускаться администратором с защищенной рабочей станции или со специального носителя.

При работе в сети ограничения на права отдельного пользователя или группы пользователей сети, а также всех пользователей конкретного компьютера целесообразно поместить в отдельный файл системных правил, который может быть помещен, например, в папку Netlogon на сервере, работающем под управлением одной из защищенных версий операционной системы Windows. В этом случае после входа пользователя КС в сеть с любого компьютера информация об установленных для него ограничениях из файла системных правил на сервере заместит соответствующие разделы реестра на использованном для входа в систему компьютере. Администратор КС может создавать с помощью обычного текстового редактора файлы шаблонов ограничений для различных категорий пользователей и их групп (adm-файлы), в которых в том числе могут быть отражены и дополнительные ограничения на использование функций не только системных, но и прикладных программ.

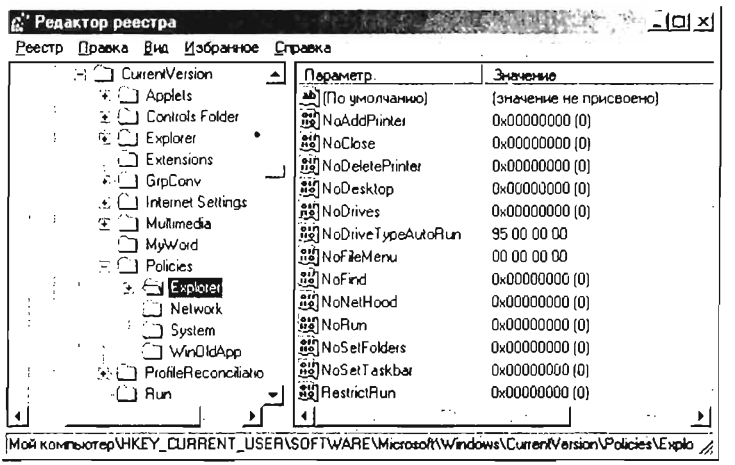

Рис. 3.6. Отражение в реестре установленных ограничений прав пользователей

Администратор КС может установить максимальные ограничения для пользователя с профилем по умолчанию и тем самым для всех незарегистрированных им пользователей, которые попытаются войти в систему.

Нарушитель может попытаться прервать обычную процедуру загрузки операционной системы, чтобы загрузить ее в так называемом безопасном режиме или в режиме MS-DOS (при этом возможен обход установленных администратором ограничений). Чтобы исключить эту возможность, необходимо добавить в текстовый файл msdos.sys следующую секцию:

```
[Options]
BootKeys=0
```

Тем самым у нарушителя не будет возможности прервать обычную процедуру загрузки операционной системы, нажав определенную комбинацию клавиш на клавиатуре (Ctrl или F8 для вызова меню загрузки либо другую для загрузки в отличном от нормального режиме).

Чтобы не дать нарушителю возможности произвести загрузку с собственного носителя (системной дискеты или загрузочного компакт-диска) и обойти установленные для незарегистрированных пользователей ограничения, необходимо исключить такую возможность с помощью программы BIOS Setup. Ее функция Advanced BIOS Features (или аналогичная в программах других производителей) позволяет установить порядок применения устройств при загрузке операционной системы. Для обеспечения наибольшей безопасности загрузка должна всегда начинаться (First Boot Device) с жесткого диска (HDD-0), а при невозможности загрузки с жесткого диска — продолжаться (Second Boot Device) с дискеты (Floppy) или компакт-диска (CDROM).

Для повышения безопасности при использовании открытых версий операционной системы Windows можно также использовать исключение из состава аппаратных средств компьютера накопителей на гибких магнитных дисках и компакт-дисках с помощью средств программы BIOS Setup (функция Standard CMOS Features или аналогичная).

Доступ к изменению настроек, устанавливаемых программой BIOS Setup, должен быть разрешен только администратору КС. Для этого с помощью функции Set Supervisor Password этой программы (или аналогичной) необходимо установить пароль администратора. Без ввода этого пароля будут невозможны вызов программы BIOS Setup и изменение ее настроек.

Чтобы заблокировать терминал рабочей станции на период временного отсутствия пользователя, в открытых версиях операционной системы Windows должны использоваться программы-заставки (рис. 3.7) с установленным паролем для возобновления

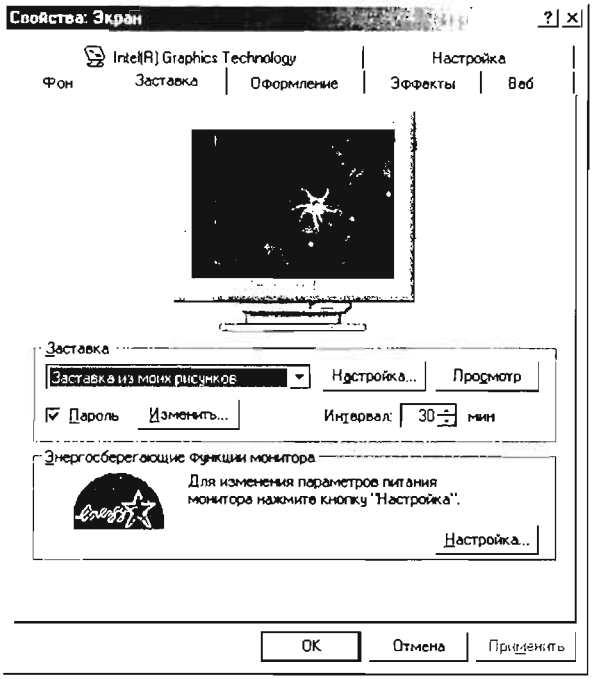

Рис. 3.7. Установка программы-заставки

работы после запуска программы-заставки. Поскольку программа-заставка автоматически запускается только по истечении заданного интервала времени после прекращения работы пользователя с устройствами ввода, необходимо вынести ярлык к установленной программе-заставке на рабочий стол пользователя (программы-заставки имеют расширение «.scr» и размещаются обычно в папке System папки с файлами операционной системы Windows).

С помощью организационных мер требуется также обеспечить применение пользователями этих средств для блокировки терминалов своих рабочих станций.

Пароли пользователей открытых версий операционной системы Windows сохраняются в файлах с расширением «.pwl» и с именем, совпадающим с логическим именем зарегистрированного пользователя КС, в хешированном виде. К сожалению, при этом используется нестойкая функция шифрования, поэтому в сети Интернет свободно распространяются программы для подбора паролей из этих файлов, которые могут быть использованы нарушителем. Поэтому необходимо, но недостаточно применение следующих мер:

• запрет (с помощью редактора системных правил) запуска непривилегированными пользователями любых программ, кроме необходимых им для выполнения своих служебных обязанностей;

• запрет сохранения паролей привилегированных пользователей на рабочих станциях под управлением открытых версий Windows (с помощью редактора системных правил или организационных мер).

Имеются и другие недостатки рассмотренных средств защиты от несанкционированного доступа к информации в открытых версиях операционной системы Windows:

• влияние ограничений на запускаемые пользователями приложения и отображение дисков распространяются только на приложения компании Microsoft (например, с помощью оболочки Windows Commander пользователь сможет запускать любые установленные на его компьютере приложения), поэтому необходим тщательный и обоснованный выбор устанавливаемого программного обеспечения и накладываемых на его использование ограничений;

• возможен обход установленных ограничений на запуск приложений путем переименования файлов с помощью проводника Windows (например, нарушитель может переименовать файл с программой для несанкционированного доступа к информации в winword.exe), поэтому необходимо обеспечить скрытие дисков в папке «Мой компьютер» и в окне проводника Windows (для невозможности копирования на жесткий диск компьютера файлов с вредоносным кодом), что может оказаться неудобным и даже невозможным для легальных пользователей;

• для легальных пользователей, зарегистрированных в КС и имеющих доступ к жесткому диску компьютера, и запущенных ими программ невозможно осуществить разграничение доступа к файлам, папкам, принтерам и разделам реестра на этом компьютере (любой вошедший в систему пользователь с подобными правами имеет полный доступ к любым ресурсам этого компьютера, а разграничение возможно только для доступа с других компьютеров локальной сети на основе одного пароля для полного доступа к папке и файлам в ней или двух паролей для чтения и записи соответственно);

• если нарушителю удастся получить доступ к компьютеру с загруженной операционной системой, то он сможет попытаться изменить настройки, установленные программой BIOS Setup, путем прямого редактирования содержимого CMOS-памяти с помощью специальных программных средств.

Отсутствие возможности разграничения доступа пользователей к ресурсам КС в рассматриваемых операционных системах является главной причиной отнесения их к классу открытых (слабо защищенных от несанкционированного доступа к информации)

систем. Разграничение доступа к файлам и папкам с защищаемой информацией в такого рода системах возможно только с помощью их шифрования (см. гл. 4 и 5).

3.2. Дискреционное и мандатное управление доступом к объектам компьютерных систем

Прежде чем рассмотреть средства защиты информации от несанкционированного доступа, имеющиеся в защищенных операционных системах, остановимся на основных моделях управления доступом к объектам КС со стороны их субъектов. Разделение всех элементов КС на множества субъектов и объектов осуществляется на основе обладания субъектами свойством «быть активными» («получать управление»). Понятие субъекта отличается от понятия пользователя КС, который обычно является физическим лицом, обладающим некоторой идентифицирующей его информацией (но возможны и псевдопользователи, например сама система). Пользователь управляет работой субъекта (порожденного им из объекта с программой процесса) с помощью его интерфейсных элементов (команд меню, кнопок и т. п.).

Дискреционное управление доступом (Discretionary Access Control, DAC) к объектам КС предполагает выполнение следующих требований:

- все субъекты и объекты КС должны быть однозначно идентифицированы;
- для любого объекта КС должен быть определен пользователь-владелец;
- владелец объекта должен обладать правом определения прав доступа к объекту со стороны любых субъектов КС;
- в КС должен существовать привилегированный пользователь, обладающий правом полного доступа к любому объекту (или правом становиться владельцем любого объекта).

Последнее свойство определяет невозможность существования в КС потенциально недоступных объектов, владелец которых отсутствует. Но реализация права полного доступа к любому объекту посредством предварительного назначения себя его владельцем не позволяет привилегированному пользователю (администратору) использовать свои полномочия незаметно для реального владельца объекта.

Дискреционное управление доступом к объектам КС реализуется обычно в виде матрицы доступа, строки которой соответствуют субъектам КС, а столбцы — ее объектам. Элементы матрицы доступа определяют права доступа субъектов к объектам. В целях сокращения затрат памяти матрица доступа может задаваться в виде списков прав субъектов (для каждого из них создается спи-

сок всех объектов, к которым разрешен доступ со стороны данного субъекта) или в виде списков контроля доступа (для каждого объекта КС создается список всех субъектов, которым разрешен доступ к данному объекту).

К достоинствам дискреционного управления доступом к объектам КС относятся относительно простая реализация (проверка прав доступа субъекта к объекту производится в момент открытия этого объекта в процессе субъекта) и хорошая изученность (в наиболее распространенных операционных системах универсального назначения применяется разграничение доступа на основе дискреционного управления).

К недостаткам дискреционного управления доступом к объектам КС относится прежде всего статичность разграничения доступа — права доступа к уже открытому субъектом объекту в дальнейшем не изменяются, независимо от изменения состояния КС.

При использовании дискреционного управления доступом к объектам КС не существует возможности проверки, не приведет ли разрешение доступа к объекту для некоторого субъекта к нарушению безопасности информации в КС (например, владелец файла с конфиденциальной информацией, дав разрешение на его чтение другому пользователю, делает этого пользователя фактически владельцем защищаемой информации). Иначе говоря, дискреционное управление доступом к объектам КС не обеспечивает защиты от утечки конфиденциальной информации.

Дискреционное управление доступом к объектам КС не позволяет обеспечить надежную защиту от проникновения в КС вредоносных программ.

Пример. Пусть LU — добросовестный пользователь КС, а NU — пользователь-нарушитель. Обозначим через CO объект с конфиденциальной информацией, владельцем которого является LU, и через PO — объект с исходным кодом программы, содержащей закладку («троянским конем», см. гл. 6), владельцем которого является NU. Нарушитель предоставляет LU право записи в объект PO (матрица доступа принимает вид, соответствующий табл. 3.1).

Таблица 3.1

Субъект	Объект CO	Объект PO
LU	Право владения Право чтения Право записи	Право записи
NU	—	Право владения Право чтения Право записи

Нарушитель NU создает ситуацию, при которой владелец объекта с конфиденциальной информацией запускает программу из объекта PO (например, предоставляя LU возможность использования новой компьютерной игры). Пользователь LU запускает программу из PO, активизируя тем самым закладку, которая, имея в этом случае права доступа LU, читает информацию из объекта CO и записывает ее в объект PO. После этого нарушитель NU читает конфиденциальную информацию из объекта PO.

К недостаткам дискреционного управления доступом к объектам КС относится также автоматическое назначение прав доступа субъектам (из-за большого числа объектов в КС в качестве субъектов доступа остаются только пользователи КС, а значение элемента матрицы доступа вычисляется с помощью функции, определяющей права доступа порожденного пользователем субъекта к данному объекту КС).

Мандатное управление доступом (Mandatory Access Control, MAC) к объектам КС во многом лишено отмеченных недостатков. В этом случае необходимо обеспечить выполнение следующих требований:

- все субъекты и объекты КС должны быть однозначно идентифицированы;
- должен существовать линейно упорядоченный набор меток конфиденциальности и соответствующих им степеней допуска (нулевая метка или степень соответствуют открытому объекту и степени допуска к работе только с открытыми объектами);
- каждому объекту КС должна быть присвоена метка конфиденциальности;
- каждому субъекту КС должна быть присвоена степень допуска;
- в процессе своего существования каждый субъект должен иметь свой уровень конфиденциальности, равный максимуму из меток конфиденциальности объектов, к которым данный субъект получил доступ;
- в КС должен существовать привилегированный пользователь, имеющий полномочия на удаление любого объекта системы;
- понизить метку конфиденциальности объекта может только субъект, имеющий доступ к данному объекту и обладающий специальной привилегией;
- право на чтение информации из объекта получает только тот субъект, чья степень допуска не больше метки конфиденциальности данного объекта (правило «не читать выше»);
- право на запись информации в объект получает только тот субъект, чей уровень конфиденциальности не меньше метки конфиденциальности данного объекта (правило «не записывать ниже»).

Основной целью мандатного управления доступом к объектам КС является предотвращение утечки информации из объектов с

высокой меткой конфиденциальности в объекты с низкой меткой конфиденциальности (противодействие созданию каналов передачи информации «сверху вниз»).

Для мандатного управления доступом к объектам КС формально доказано следующее важное утверждение: если начальное состояние КС безопасно и все переходы из одного состояния системы в другое не нарушают правил разграничения доступа, то любое состояние КС также безопасно.

Достоинствами мандатного управления доступом к объектам КС также являются:

- более высокая надежность работы самой КС, так как при разграничении доступа к объектам контролируется и состояние самой системы, а не только соблюдение установленных правил;
- большая простота определения правил разграничения доступом по сравнению с дискреционным управлением (эти правила более ясны для разработчиков и пользователей КС).

Продолжим приведенный ранее пример с использованием нарушителем программной закладки, обеспечивающей канал утечки конфиденциальной информации. Обозначим через C(LU), C(NU), C(CO) и C(PO) соответственно степени допуска пользователей LU и NU и метки конфиденциальности объектов CO и PO. Пусть C(LU) > C(NU). Теперь, если LU может записать в CO конфиденциальную информацию, должно выполняться условие $C(NU) < C(LU) \leq C(CO)$. Для того чтобы содержащаяся в PO программная закладка могла прочитать информацию из CO и записать ее в PO, необходимо выполнение условие $C(PO) \geq C(CO)$. Тогда C(PO) > C(NU) и нарушитель NU не имеет права на чтение информации из PO, что делает атаку данного вида бессмысленной.

Недостатки мандатного управления доступом к объектам КС:

- сложность программной реализации, которая увеличивает вероятность внесения ошибок и появления каналов утечки конфиденциальной информации;
- снижение эффективности работы КС, так как проверка прав доступа субъекта к объекту выполняется не только при открытии объекта в процессе субъекта, но и перед выполнением любой операции чтения из объекта или записи в объект;
- создание дополнительных неудобств работе пользователей КС, связанных с невозможностью изменения информации в неконфиденциальном объекте, если тот же самый процесс использует информацию из конфиденциального объекта (его уровень конфиденциальности больше нуля).

Для устранения последнего недостатка необходимо разработать программное обеспечение КС с учетом особенностей мандатного управления доступом к объектам КС.

Из-за отмеченных недостатков мандатного управления доступом в реальных КС множество объектов, к которым применяется

мандатное управление, является подмножеством множества объектов, доступ к которым осуществляется на основе дискреционного управления.

В «Оранжевой книге» (см. подразд. 1.6) выделено два класса защищенности для КС, в которых реализовано дискреционное управление доступом к объектам КС (классы С1 и С2), и три класса, в которых должно быть реализовано мандатное управление доступом к объектам КС (классы В1, В2 и В3).

Для класса защищенности КС С1 требуется определить права доступа между поименованными объектами и субъектами (пользователями или группами); для класса С2 — разграничить доступ с определением прав для каждого отдельного пользователя КС.

Для класса защищенности КС В2 по сравнению с В1 требуется обеспечить уведомление каждого работающего пользователя о любых изменениях его степени допуска и возможность запроса пользователем КС полной информации о его степени допуска и ее изменениях. В классе В3 не предъявляется дополнительных требований в области мандатного управления доступом к объектам КС.

В руководящих документах Гостехкомиссии России (см. подразд. 1.6) для АС класса защищенности 1Г должно быть обеспечено дискреционное управление доступом к объектам КС, а начиная с класса 1В — мандатное управление доступом к объектам КС.

3.3. Подсистема безопасности защищенных версий операционной системы Windows

К защищенным версиям операционной системы Windows относятся Windows NT/2000/XP Professional. В состав этих операционных систем входит подсистема безопасности, архитектура которой представлена на рис. 3.8.

Ядром подсистемы безопасности является локальная служба безопасности (Local Security Authority, LSA), размещающаяся в файле lsass.exe. После загрузки операционной системы автоматически запускается процесс входа (winlogon.exe), который остается активным до перезагрузки операционной системы или выключения питания компьютера, а его аварийное завершение приводит к аварийному завершению всей операционной системы. Этим обеспечивается практическая невозможность подмены процесса входа при функционировании системы.

После нажатия пользователем комбинации клавиш Ctrl+Alt+ +Delete процесс входа обращается к провайдеру аутентификации (динамически компонуемой библиотеке функций, DLL) для приема от пользователя его логического имени (ID) и аутентифицирующей информации (P). Стандартный провайдер аутентификации размещается в файле msgina.dll и в качестве аутентифициру-

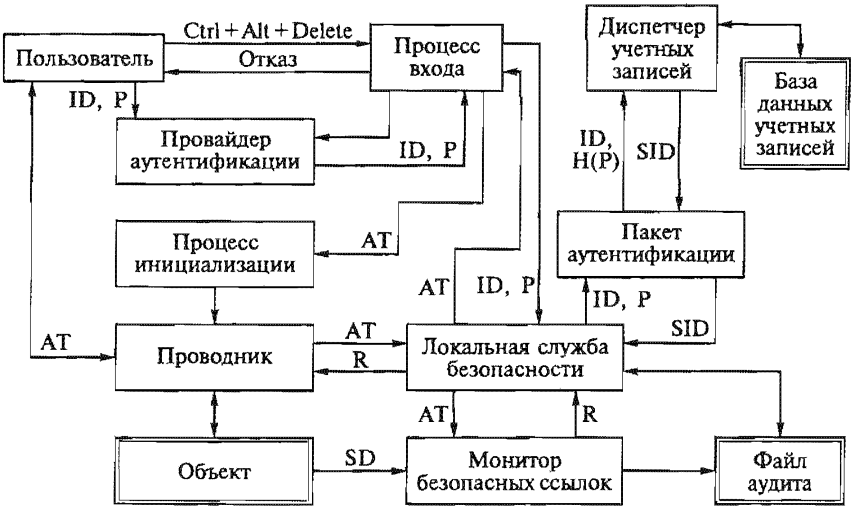

Рис. 3.8. Архитектура подсистемы безопасности защищенных версий Windows

ющей информации использует пароли пользователей. Возможно использование других провайдеров аутентификации (например, считывающих ключевую информацию со смарт-карт). В этом случае необходимо, чтобы интерфейс к предоставляемым провайдером аутентификации функциям соответствовал определениям, содержащимся в файле winwlx.h (входит в состав любой системы программирования на языке C++ для Windows). Путь к используемому процессом входа в систему провайдеру аутентификации должен быть записан в разделе реестра HKEY_LOCAL_MACHINE\Software\Microsoft\WindowsNT\CurrentVersion\WinLogon как значение параметра GinaDLL.

Введенные пользователем логическое имя и пароль передаются процессом входа в службу LSA, которая обращается к пакету аутентификации (библиотеке функций) для подтверждения подлинности пользователя. Если пользователь зарегистрирован на локальном компьютере, то пакет аутентификации вычисляет хеш-значение пароля H(P) и обращается к диспетчеру учетных записей (Security Account Manager, SAM) для проверки правильности введенного пароля и возможности для пользователя с введенным логическим именем начать работу в системе (не истек ли срок действия пароля, не заблокирована ли учетная запись пользователя и т. п.). Пакет аутентификации также является заменяемым компонентом подсистемы безопасности (стандартный пакет аутентификации размещается в файле msv1_0.dll).

Диспетчер учетных записей обращается к базе данных учетных записей (базе данных SAM) для извлечения информации из учетной записи пользователя с введенным логическим именем. База данных учетных записей содержится в разделе реестра HKEY_LOCAL_MACHINE\SAM (а также в файле Windows\System32\Config\SAM). К базе данных SAM не может быть получен доступ для чтения или изменения с помощью штатных средств операционной системы даже администратором. Для ее редактирования предназначены специальные функции из набора Windows API и специальное системное приложение (в Windows XP — функция «Администрирование» панели управления).

Пароль пользователя в базе данных SAM хранится в виде двух хеш-значений, каждое из которых имеет длину 128 бит. Первое хеш-значение пароля пользователя вычисляется по алгоритму Windows NT.

1. Строка символов пароля P усекается до 14 знаков (при необходимости) и преобразуется в кодировку Unicode, в которой каждый символ представляется двумя байтами.

2. Вычисляется хеш-значение преобразованного пароля H(P) длиной 128 бит (используется функция хеширования MD4).

3. Полученное хеш-значение зашифровывается по алгоритму DES с помощью ключа, равного относительному номеру учетной записи пользователя, E_{RID} (H(P)).

4. Полученный результат шифрования записывается в базу данных учетных записей.

Второе хеш-значение пароля пользователя вычисляется по алгоритму LAN Manager.

1. Все буквенные символы (латинского алфавита) строки пароля P преобразуются к верхнему регистру.

2. Строка символов пароля дополняется нулями, если она короче 14 байтов, и делится на две семибайтовые половины P_1 и P_2.

3. Каждое из значений P_1 и P_2 используется в качестве ключа для шифрования по алгоритму DES магической строки M = "KGS!@# $ %", в результате которого получаются два значения, из 64 бит каждое: $H_1 = E_{P1}(M)$ и $H_2 = E_{P2}(M)$.

4. Выполняется шифрование по алгоритму DES на ключе, равном относительному номеру учетной записи, результата сцепления H_1 и H_2: $E_{RID}(H_1 \| H_2)$.

5. Полученный результат шифрования помещается в базу данных SAM.

Алгоритм получения хеш-значения LAN Manager является менее стойким (искусственно уменьшается мощность алфавита, из которого выбираются символы пароля, а разделение пароля на две половинки облегчает его подбор в том случае, если длина пароля не превышает семи знаков, так как результат шифрования магической строки на нулевом ключе заранее известен нарушите-

лю). Использование более слабого алгоритма LAN Manager объясняется необходимостью обеспечения совместимости с открытыми версиями операционной системы Windows, которые могут быть установлены на рабочих станциях локальной сети организации.

Недостатком хранения паролей пользователей в защищенных версиях Windows является также то, что, не принимая во внимание заключительного шифрования на ключе, равном относительному номеру учетной записи, в обоих алгоритмах одинаковому паролю, случайно выбранному разными пользователями, будет соответствовать одно и то же хеш-значение. Это создает теоретическую возможность применения полномочий одного пользователя другим. Относительный же номер учетной записи может быть достаточно легко получен нарушителем (встроенная учетная запись администратора, например, всегда имеет относительный номер 500).

Для устранения подобной уязвимости достаточно было бы вычислять хеш-значение пароля с использованием дополнительного случайного числа, генерируемого при регистрации пользователя в КС (см. подразд. 2.1).

Несмотря на то что доступ к базе данных SAM с помощью штатных средств Windows для нарушителя практически невозможен, он, тем не менее, может, например, скопировать весь содержащий ее жесткий диск на подготовленный носитель (помешать этому можно только с помощью организационных мер). Затем информация с жесткого диска восстанавливается на компьютере нарушителя, который теперь может загрузить одну из открытых версий операционной системы Windows и получить доступ к базе данных учетных записей, как к обычному файлу (с помощью специальных программных средств).

Следующим шагом нарушителя будет программный подбор паролей привилегированных пользователей, получив которые он сможет осуществить несанкционированный доступ к информации в КС. Для предотвращения подобной угрозы администратор защищенных версий Windows может дополнительно зашифровать хеш-значения паролей пользователей в базе данных SAM с помощью системной программы syskey. Программа syskey обеспечит шифрование хеш-значений паролей с помощью первичного ключа длиной 128 бит, хранящегося в реестре также в зашифрованном виде.

После запуска программы syskey администратор должен выбрать способ хранения системного ключа длиной 128 бит, который будет использован для шифрования первичного ключа:

- в системном реестре (преимущество этого способа заключается в отсутствии необходимости присутствия привилегированного пользователя при перезагрузке операционной системы, а недостаток — в наименьшей защищенности хранения системного ключа);

• в файле startup.key (длиной 16 байт) в корневом каталоге специальной дискеты (в этом случае придется позаботиться о защищенном хранении этой дискеты и ее резервной копии);

• без физического сохранения системного ключа, который будет генерироваться из специальной парольной фразы длиной не менее 12 символов.

Во втором и третьем способах при загрузке операционной системы перед появлением приглашения нажать комбинацию клавиш Ctrl+Alt+Delete появится диалоговое окно с приглашением вставить дискету с системным ключом шифрования или ввести парольную фразу.

Если пользователь не зарегистрирован на локальном компьютере, то пакет аутентификации отправляет логическое имя и хеш-значение пароля пользователя сервису NetLogon для аутентификации пользователя на контроллере домена (сервере аутентификации группы компьютеров, разделяющих общую политику безопасности и информацию об учетных записях пользователей).

Если хеш-значение введенного пользователем пароля не совпадает с эталоном, извлеченным из базы данных учетных записей, или работа пользователя в системе невозможна (пользователь с такой учетной записью не зарегистрирован, учетная запись пользователя заблокирована или отключена), то пакет аутентификации возвращает LSA код ошибки, который локальная служба безопасности передает процессу входа для выдачи пользователю сообщения об отказе в доступе к системе.

Если проверка подтвердила подлинность пользователя и отсутствие препятствий для начала его работы в КС, то пакет аутентификации получает от SAM уникальный идентификатор безопасности пользователя SID (security identifier), который затем передается в LSA.

Идентификатор безопасности представляет собой структуру переменной длины, которая однозначно определяет пользователя или группу и сохраняется в регистрационной базе данных. В SID содержится следующая информация:

• идентификатор авторизации (48 бит), состоящий из идентификатора домена КС и относительного номера учетной записи (RID) в его регистрационной базе данных (32 бита);

• уровень пересмотра;

• переменное число идентификаторов субавторизации.

Для записи SID применяется специальная нотация: S-R-I-S-S... (S определяет эту запись как SID, R задает уровень пересмотра, I — идентификатор авторизации и S — идентификаторы субавторизации), например: S-1-4138-86. Некоторые значения SID являются предопределенными в системе, например: S-1-0-0 (группа без пользователей-членов), S-1-1-0 (группа, включающая в себя всех пользователей), S-1-2-0 (группа, включающая в себя пользовате-

лей, которые входят в КС локально), S-1-3-0 (пользователь, являющийся создателем-владельцем нового объекта в КС), S-1-3-1 (первичная группа пользователя, являющегося создателем-владельцем нового объекта), S-1-5-1 (группа, включающая в себя пользователей, осуществляющих удаленный доступ к КС через модем), S-1-5-2 (группа, включающая в себя пользователей, осуществляющих доступ по сети), S-1-5-18 (псевдопользователь — операционная система) и др.

Получив идентификатор безопасности пользователя, локальная служба безопасности LSA создает для него маркер доступа AT (access token), который идентифицирует пользователя во всех его действиях с объектами КС.

В маркере доступа содержится следующая информация:

- SID пользователя;
- идентификаторы безопасности его групп;
- полномочия пользователя;
- идентификаторы безопасности пользователя и его первичной группы, которые будут использованы при создании пользователем новых объектов в КС;
- дискреционный список контроля доступа по умолчанию для вновь создаваемого объекта;
- источник выдачи маркера доступа;
- тип маркера доступа — первичный (созданный при входе пользователя в КС) или используемый для олицетворения;
- текущий уровень олицетворения и др.

Полномочия (привилегии) пользователя и группы назначаются администратором КС и представляют собой права субъектов на выполнение действий, относящихся к системе в целом, а не к отдельным ее объектам. Каждой привилегии соответствует уникальный локальный идентификатор LUID (locally unique identifier), определяемый строкой символов (например, SE_SYSTEMTIME_NAME).

Для отображения пользователю содержания конкретной привилегии с ней также связывается текстовая строка (например, «Изменение системного времени» или «Change the system time»). Внутреннее представление информации о привилегии зависит от конкретной реализации.

Перечислим наиболее важные привилегии, которые могут быть назначены пользователям и группам:

- завершение работы системы (SE_SHUTDOWN_NAME);
- изменение системного времени (SE_SYSTEMTIME_NAME);
- отладка программ (SE_DEBUG_NAME) — обладание этой привилегией проверяется только при использовании для отладки систем программирования корпорации Microsoft;
- архивирование файлов и каталогов (SE_BACKUP_NAME);
- восстановление файлов и каталогов (SE_RESTORE_NAME);

• управление аудитом и журналом безопасности (SE_SECURITY_NAME);
• создание журналов безопасности (SE_AUDIT_NAME);
• смена владельцев файлов или иных объектов (SE_TAKE_OWNERSHIP_NAME);
• принудительное удаленное завершение работы (SE_REMOTE_SHUTDOWN_NAME);
• профилирование загруженности системы (SE_SYSTEM_PROFILE_NAME);
• профилирование одного процесса (SE_PROF_SINGLE_PROCESS_NAME);
• работа в режиме операционной системы (SE_TCB_NAME);
• создание маркерного объекта (SE_CREATE_TOKEN_NAME);
• замена маркера уровня процесса (SE_ASSIGNPRIMARY-TOKEN_NAME);
• увеличение приоритета диспетчирования (SE_INC_BASE_PRIORITY_NAME);
• изменение параметров среды оборудования (SE_SYSTEM_ENVIRONMENT_NAME);
• загрузка и выгрузка драйверов устройств (SE_LOAD_DRIVER_NAME) и др.

Обладание перечисленными привилегиями должно быть ограничено, поскольку оно позволяет субъектам обходить те или иные средства обеспечения безопасности информации в КС. Например, привилегии архивирования и восстановления файлов и каталогов позволяют их обладателям обходить правила разграничения доступа к объектам КС. Привилегия увеличения приоритета диспетчирования позволяет ее обладателю нарушить доступность информации в операционной системе, создав процесс с максимально высоким приоритетом и организовав в нем бесконечный цикл. Привилегия создания журналов безопасности позволяет ее обладателю записывать в файл аудита информацию для компрометации других пользователей и скрытия собственных действий и т.д.

В маркер доступа не помещаются сведения о привилегиях входа пользователей в КС, которые проверяются пакетом аутентификации на этапе их авторизации в системе:
• локальный вход в систему;
• доступ к компьютеру из сети;
• вход в качестве пакетного задания;
• вход в качестве службы;
• вход в систему через службу терминалов.

При взаимодействии клиентов и серверов в защищенных версиях операционной системы Windows может использоваться механизм олицетворения (impersonation). В этом случае при необходимости обращения с запросом к серверу в процессе клиента создается поток (thread), которому назначается маркер доступа

пользователя, инициировавшего этот запрос. Этот механизм позволяет серверу действовать от лица клиента при доступе к объектам, к которым сам сервер не имеет доступа.

Созданный LSA маркер доступа AT передается процессу входа, который с помощью провайдера аутентификации завершает процесс авторизации пользователя в КС, запуская процесс его инициализации (userinit.exe) и передавая ему AT. Процесс инициализации на основе содержащегося в AT идентификатора безопасности пользователя загружает из реестра Windows его профиль и загружает программную оболочку — проводник Windows (explorer.exe), передавая ему маркер доступа пользователя. После этого процесс инициализации завершает свою работу.

Концепция рабочего стола пользователя (desktop) в защищенных версиях Windows отличается от аналогичного понятия в открытых версиях этой операционной системы. Рабочий стол в защищенных версиях Windows представляет собой совокупность окон, одновременно видимых на экране. Только процессы, окна которых расположены на одном рабочем столе, могут взаимодействовать между собой, используя средства графического интерфейса пользователя Windows (GUI).

Процесс входа (winlogon), получающий от пользователя имя и пароль, выполняется на отдельном рабочем столе (рабочем столе аутентификации). Никакой другой процесс, в том числе и программная закладка, внедренная нарушителем для перехвата паролей, не имеет доступа к этому рабочему столу. Поэтому приглашение пользователю на вход в систему, выводимое этой закладкой, может располагаться только на рабочем столе прикладных программ, с которым связаны все запущенные пользователем программы.

Переключение экрана компьютера с одного рабочего стола на другой производится при нажатии комбинации клавиш Ctrl+Alt++Delete. В защищенных версиях Windows эта комбинация обрабатывается иначе — сообщение о нажатии комбинации клавиш Ctrl+Alt+Delete посылается только процессу входа, который остается активным до перезагрузки операционной системы или выключения питания. Для всех других процессов (в частности, для всех прикладных программ, запущенных пользователем) нажатие этой комбинации клавиш совершенно незаметно.

При загрузке системы на экране компьютера вначале отображается рабочий стол аутентификации. Однако пользователь вводит имя и пароль не сразу, а только после нажатия комбинации клавиш Ctrl+Alt+Delete. Когда пользователь завершает сеанс работы с системой, на экран также выводится рабочий стол аутентификации, и новый пользователь может ввести пароль для входа в систему только после нажатия комбинации клавиш Ctrl+Alt++Delete.

Если в систему внедрена программная закладка для перехвата паролей, то для осуществления ею этого перехвата в закладке необходимо обработать нажатие пользователем комбинации клавиш Ctrl+Alt+Delete. В противном случае при нажатии пользователем этой комбинации клавиш произойдет переключение на рабочий стол аутентификации, рабочий стол прикладных программ станет неактивным и закладка просто не сможет ничего перехватить — сообщения о нажатии пользователем клавиш будут приходить на другой рабочий стол. Однако для всех прикладных программ факт нажатия пользователем комбинации клавиш Ctrl+Alt+Delete всегда остается незамеченным. Поэтому введенный пользователем пароль будет воспринят не программной закладкой, а процессом входа.

Конечно, программная закладка может имитировать не первое приглашение операционной системы, когда пользователю предлагается нажать комбинацию клавиш Ctrl+Alt+Delete для начала работы, а то приглашение, которое высвечивается после нажатия пользователем этой комбинации клавиш.

Однако в обычных условиях (при отсутствии внедренной программной закладки) это второе приглашение автоматически отменяется через достаточно короткое время (от 30 с до 1 мин, в зависимости от версии Windows). Если второе приглашение присутствует на экране компьютера долгое время, то этот факт должен насторожить пользователя. Кроме того, как показывает опыт, пользователи, имеющие достаточный опыт работы с защищенными версиями Windows, приобретают привычку начинать работу с системой с нажатия комбинации клавиш Ctrl+Alt+Delete независимо от того, что отображается на экране.

Защита рассматриваемых версий операционной системы Windows от программных закладок такого рода может считаться весьма надежной.

Для управления списками пользователей и групп в КС, назначения им полномочий, определения параметров политики безопасности в операционных системах Windows 2000/XP администратором используются функции «Администрирование» и «Локальная политика безопасности» панели управления, а в операционной системе Windows NT для этого предназначалась системная программа «Диспетчер пользователей» (User Manager).

На рис. 3.9 и 3.10 приведены примеры определения свойств учетной записи пользователя, а также свойств и состава группы пользователей. Для учетной записи пользователя помимо его логического имени для входа в систему и полного имени и (или) описания задаются следующие характеристики:

- признак необходимости смены пароля при следующем входе в систему (устанавливается, как правило, при создании учетной записи администратором или задании для пользователя нового пароля);

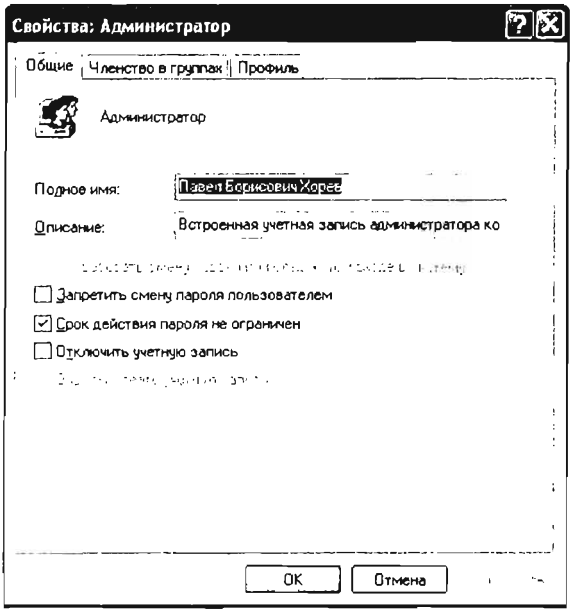

Рис. 3.9. Определение свойств учетной записи пользователя

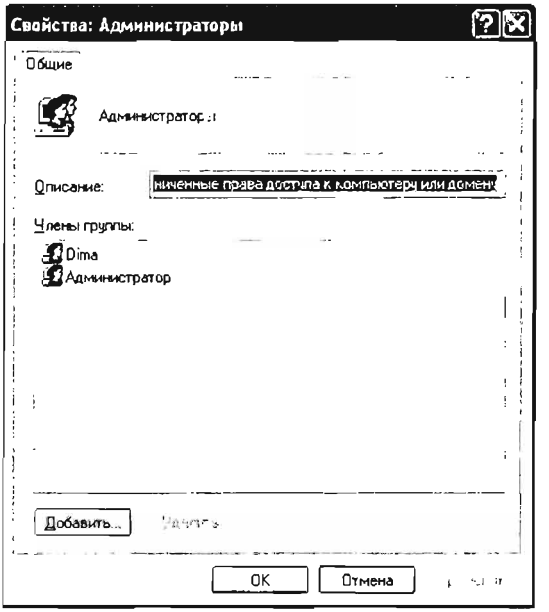

Рис. 3.10. Определение свойств и состава группы пользователей

• признак запрещения смены пароля пользователем (может устанавливаться для обобщенных учетных записей, объединяющих нескольких пользователей с минимальными правами в системе, например студентов);

• признак неограниченности срока действия пароля (должен применяться в исключительных случаях, так как отрицательно влияет на безопасность информации в КС);

• признак отключения учетной записи (используется при необходимости временно запретить пользователю возможность доступа к КС);

• признак блокировки учетной записи (устанавливается автоматически при превышении максимального числа неудачных попыток входа в систему и снимается администратором вручную или по истечении заданного периода времени).

У администратора имеется возможность задания пользователю нового пароля (с помощью соответствующей команды контекстного меню учетной записи пользователя), однако это право следует применять осторожно (только в том случае, если пользователь забыл свой пароль). Задание нового пароля приведет, в частности, к невозможности для пользователя работать с документами и папками, зашифрованными им средствами шифрующей файловой системы (см. подразд. 5.6).

В защищенных версиях операционный системы Windows предопределены несколько псевдопользователей, например «Система» (SYSTEM) или «Создатель-владелец» объекта КС (CREATOR_OWNER). Автоматически создаются учетные записи «Администратор» (Administrator) и «Гость» (Guest). Для запрещения анонимного входа в КС учетная запись «Гость» должна быть отключена. Для предотвращения угрозы компрометации пароля встроенной учетной записи «Администратор», для которой нарушителю известны не только логическое имя, но и относительный номер RID, эта учетная запись также должна быть отключена после создания другой учетной записи с полномочиями администратора КС.

В свойствах группы указываются ее имя, описание и список входящих в ее состав пользователей и других групп. К предопределенным группам относятся: «Все» (Everyone, включает пользователей и псевдопользователей КС), «Интерактивные» (Interactive), «Входящие по сети» (Network), «Удаленные» (Dial_Up), «Входящие в первичную группу создателя-владельца объекта КС» (CREATOR_GROUP). Автоматически создаются несколько групп пользователей: «Пользователи» (Users), «Опытные пользователи» (Power Users), «Администраторы» (Administrators), «Операторы резервного копирования» (Backup Operators). На серверах КС вместо группы опытных пользователей создаются автоматически группы администраторов учетных записей (Account Operators), адми-

нистраторов печати (Print Operators) и операторов сервера (Server Operators).

В параметрах локальной политики безопасности определены две группы параметров учетных записей — параметров политики паролей (рис. 3.11) и параметров политики блокировки учетных записей. Параметры политики паролей соответствуют рассмотренным в подразд. 2.2 требованиям к усилению парольной аутентификации. Включение параметра «Пароль должен удовлетворять требованиям сложности» обеспечит выбор пользователями паролей, удовлетворяющих следующим условиям:

- минимальная длина пароля должна равняться шести символам;
- пароль должен включать в себя символы хотя бы трех алфавитов из четырех: строчных латинских букв, прописных латинских букв, цифр и специальных знаков;
- пароль не должен совпадать с логическим именем пользователя или его частью.

С учетом изложенного ранее алгоритма хеширования паролей LAN Manager для повышения безопасности парольной аутентификации требуется дополнительно к данным требованиям сложности паролей установить равной восьми символам их минимальную длину.

К параметрам политики блокировки учетной записи относятся:

- пороговое значение блокировки (максимальное число ошибок входа в систему);

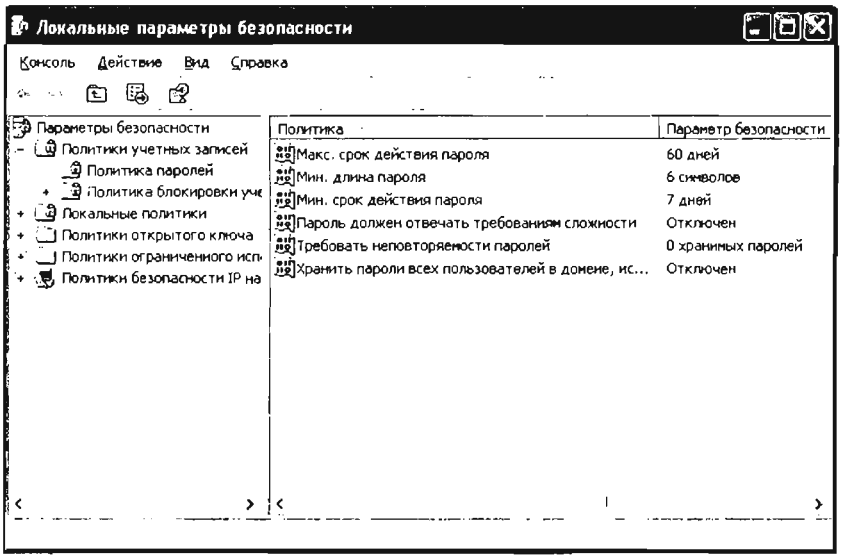

Рис. 3.11. Определение параметров парольной аутентификации

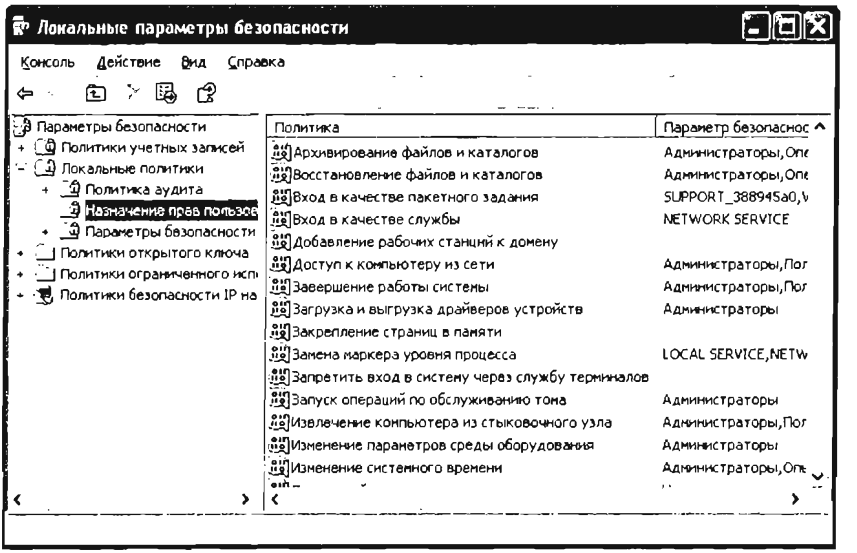

Рис. 3.12. Назначение полномочий пользователям и группам КС

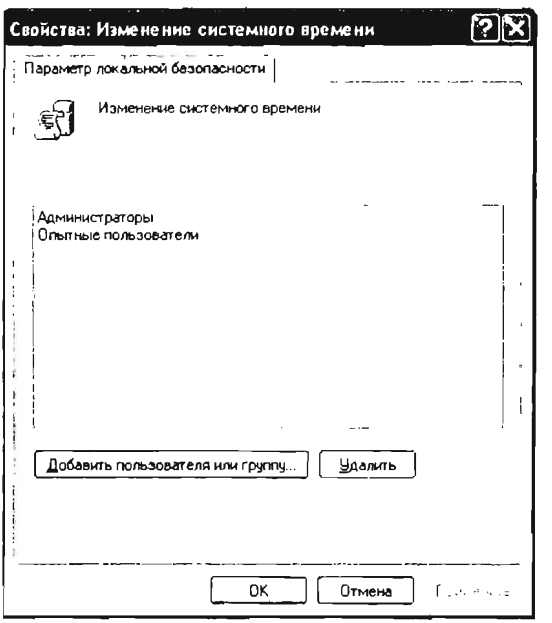

Рис. 3.13. Пример предоставления привилегии группам пользователей

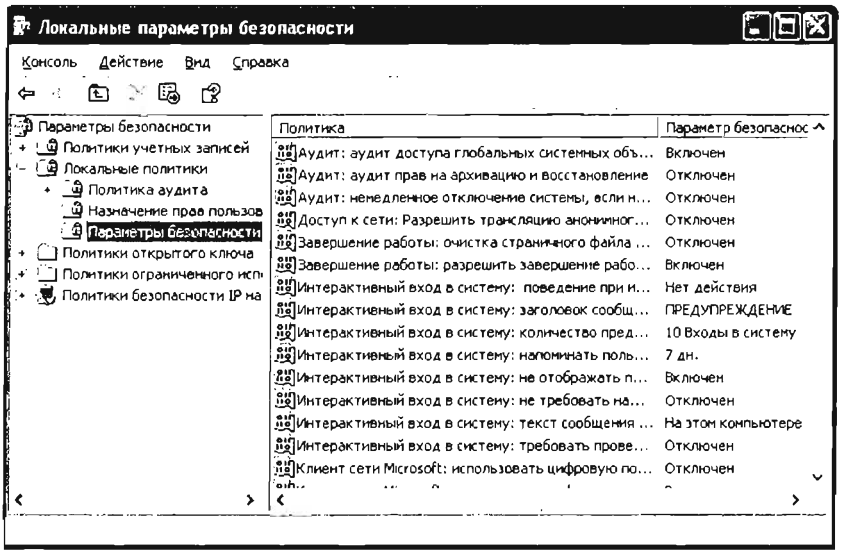

Рис. 3.14. Определение параметров политики безопасности

• длительность блокировки учетной записи;

• интервал времени, через который происходит сброс счетчика блокировок.

На рис. 3.12 приведено окно назначения полномочий пользователям и группам КС, а на рис. 3.13 — пример предоставления привилегии группам пользователей (определения списка обладателей привилегии на изменение системного времени).

На рис. 3.14 показан пример определения параметров политики безопасности, а на рис. 3.15 — пример изменения значения одного из параметров безопасности (параметра «Интерактивный вход в систему: не отображать последнего имени пользователя»).

Для работы с учетными записями, идентификаторами безопасности и маркерами доступа пользователей предназначены следующие функции из набора Windows API:

• `BOOL LogonUser(LPTSTR lpszUsername, LPTSTR lpszDomain, LPTSTR lpszPassword, DWORD dwLogonType, DWORD dwLogonProvider, PHANDLE phToken); /*` *попытка входа пользователя с именем* `lpszUsername` *и паролем* `lpszPassword` *в домен или сервер* `lpszDomain` *с типом доступа* `dwLogonType` *(интерактивным, по сети, как пакетное задание или сервис) с использованием провайдера входа* `dwLogonProvider`; *если попытка входа не отклоняется, то в* `phToken` *записывается указатель на созданный маркер доступа* `*/`

• `BOOL CreateProcessAsUser(HANDLE hToken, LPCTSTR lpApplicationName, LPTSTR lpCommandLine, LPSECURITY_`

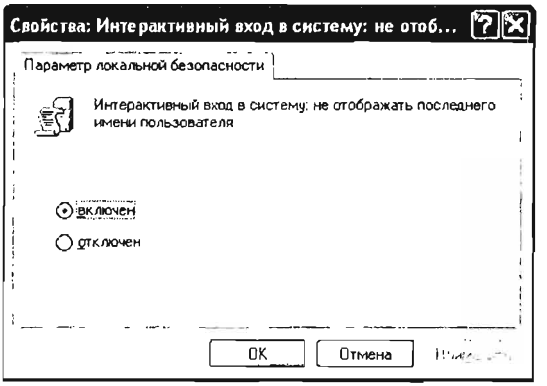

Рис. 3.15. Пример изменения значения одного из параметров безопасности

ATTRIBUTES lpProcessAttributes, LPSECURITY_ATTRIBUTES lpThreadAttributes, BOOL bInheritHandles, DWORD dwCreationFlags, LPVOID lpEnvironment, LPCTSTR lpCurrentDirectory, LPSTARTUPINFO lpStartupInfo, LPPROCESS_INFORMATION lpProcessInformation); /* *создание и запуск нового процесса от лица пользователя с маркером доступа* hToken */

- BOOL LookupAccountName(LPCTSTR lpSystemName, LPCTSTR lpAccountName, PSID Sid, LPDWORD cbSid, LPTSTR ReferencedDomainName, LPDWORD cbReferencedDomainName, PSID_NAME_USE peUse); /* *получение указателя на идентификатор безопасности* Sid (*длиной* cbSid) *пользователя с логическим именем* lpAccountName, *зарегистрированного на компьютере* lpSystemName; *при корректном завершении в* ReferencedDomainName *записывается имя домена* (*длиной* cbReferencedDomainName), *а в* peUse — *тип учетной записи* (*пользователя, группы, домена и др.*) */
- BOOL LookupAccountSid(LPCTSTR lpSystemName, PSID Sid, LPTSTR Name, LPDWORD cbName, LPTSTR ReferencedDomainName, LPDWORD cbReferencedDomainName, PSID_NAME_USE peUse); /* *получение логического имени пользователя* Name (*длиной* cbName) *по его идентификатору безопасности на компьютере* lpSystemName; *при корректном завершении возвращаются также имя домена и тип учетной записи* */
- NET_API_STATUS NetGetDisplayInformationIndex(LPWSTR ServerName, DWORD Level, LPWSTR Prefix, LPDWORD Index); /* *получение в* Index *индекса первой учетной записи типа* Level, *начинающейся с префикса* Prefix, *в базе данных на сервере* ServerName */

• NET_API_STATUS NetQueryDisplayInformation(LPWSTR ServerName, DWORD Level, DWORD Index, DWORD EntriesRequested, DWORD PreferredMaximumLength, LPDWORD ReturnedEntryCount, PVOID *SortedBuffer); /* *получение в* SortedBuffer *массива (с максимальным числом элементов* EntriesRequested *и максимальным размером* PreferredMaximumLength) *зарегистрированных на сервере* ServerName *учетных записей типа* Level, *начиная с записи с индексом* Index; *при корректном завершении в* ReturnedEntryCount *помещается фактическое число записанных в* SortedBuffer *элементов; для учетной записи пользователя или группы указываются имя, характеристики (только для пользователя), полное имя (описание), атрибуты (только для группы), относительный номер и индекс следующей учетной записи* */

• NET_API_STATUS NetUserAdd(LPWSTR servername, DWORD level, LPBYTE buf, LPDWORD parm_err); /* *добавление в регистрационную базу данных на сервере* servername *учетной записи со значениями параметров уровня* level *из буфера* buf */

• NET_API_STATUS NetUserSetInfo(LPWSTR servername, LPWSTR username, DWORD level, LPBYTE buf, LPDWORD parm_err); /* *установка значений параметрам (уровня* level) *учетной записи пользователя с логическим именем* username, *зарегистрированного на сервере* servername; *в буфере* buf *должны быть записаны устанавливаемые значения (логическое имя, пароль, срок действия пароля и т.п.)* */

• NET_API_STATUS NetUserGetInfo(LPWSTR servername, LPWSTR username, DWORD level, LPBYTE *bufptr); /* *получение в* bufptr *значений параметров (уровня* level) *учетной записи пользователя с логическим именем* username, *зарегистрированного на сервере* servername */

• NET_API_STATUS NetUserEnum(LPWSTR servername, DWORD level, DWORD filter, LPBYTE *bufptr, DWORD prefmaxlen, LPDWORD entriesread, LPDWORD totalentries, LPDWORD resume_handle); /* *получение в буфере* bufptr *длины* prefmaxlen *информации обо всех учетных записях пользователей, отобранных с помощью правила* filter; *при корректном завершении в* entriesread *помещается число фактически прочитанных записей, а в* totalentries — *общее число подобных записей* */

• NET_API_STATUS NetUserGetGroups(LPWSTR servername, LPWSTR username, DWORD level, LPBYTE *bufptr, DWORD prefmaxlen, LPDWORD entriesread, LPDWORD totalentries); /* *получение информации обо всех группах, в которые входит пользователь с логическим именем* username */

• NET_API_STATUS NetUserChangePassword(LPWSTR domainname, LPWSTR username, LPWSTR oldpassword, LPWSTR newpassword); /* *изменение пароля* (с oldpassword *на* newpassword) *пользователя с логическим именем* username, *зарегистрированного в домене* domainname */

• BOOL PrivilegeCheck(HANDLE ClientToken, PPRIVILEGE_SET RequiredPrivileges, LPBOOL pfResult); /* *проверка обладания привилегиями, заданными в* RequiredPrivileges, *пользователем с маркером доступа* ClientToken; *результат проверки записывается в* RequiredPrivileges */

• BOOL GetTokenInformation(HANDLE TokenHandle, TOKEN_INFORMATION_CLASS TokenInformationClass, LPVOID TokenInformation, DWORD TokenInformationLength, PDWORD ReturnLength); /* извлечение информации типа TokenInformationClass (имя пользователя или его группы, привилегии пользователя и т.п.) из маркера доступа TokenHandle; при успешном завершении извлеченная информация помещается в буфер TokenInformation длины TokenInformationLength (в ReturnLength помещается фактический размер запрошенной информации) */

Для разграничения доступа субъектов к объектам КС в защищенных версиях операционной системы Windows используется дискреционное управление доступом (см. подразд. 3.2).

С объектом разграничения доступа связывается дескриптор безопасности SD (security descriptor), содержащий следующую информацию:

- идентификатор безопасности (SID) владельца объекта;
- идентификатор безопасности первичной группы владельца;
- дискреционный список контроля доступа (discretionary access control list, DACL);
- системный список контроля доступа (system access control list, SACL).

Список SACL управляется администратором системы (см. подразд. 3.4). Список DACL управляется владельцем объекта и предназначен для идентификации пользователей и групп, которым предоставлен или запрещен определенный тип доступа к объекту. Каждый элемент списка DACL (access control entry, ACE) определяет права доступа к объекту для одного пользователя или группы. Каждый ACE содержит следующую информацию:

- идентификатор безопасности SID субъекта, для которого определяются права доступа;
- маска доступа (access mask, AM), которая специфицирует контролируемые данным ACE права доступа;
- тип ACE;

• признак наследования прав доступа к объекту, определенных для родительского объекта.

Элементы списка DACL могут быть двух типов: элементы, запрещающие специфицированные в них права доступа (Access-allowed ACE), и элементы, запрещающие определенные в них права доступа (Access-denied ACE). Элементы для запрещения субъектам использования определенных прав доступа должны размещаться в «голове» списка, до первого из элементов, разрешающих использование субъектом тех или иных прав доступа.

Право доступа субъекта к объекту означает возможность обращения субъекта к объекту с помощью определенного метода (типа) доступа. В защищенных версиях операционной системы Windows различают специальные, стандартные и общие (generic) права доступа к объектам.

Специальные права доступа к объектам определяют возможность обращения к объекту по свойственному только данной категории объектов методу: чтение данных из объекта, запись данных в объект, чтение атрибутов объекта, выполнение программного файла и т.д.

Стандартные права доступа к объектам определяют возможность доступа к объекту по методу, применимому к любому объекту, — изменение владельца объекта, изменение списка DACL объекта, удаление объекта и т.д.

Каждое из *общих* прав доступа к объектам представляет собой комбинацию специальных и стандартных прав и предоставляет возможность обращения к объекту с помощью некоторого набора методов доступа.

Определены следующие общие права доступа:

• чтение, включающее в себя чтение DACL объекта, чтение данных из объекта, чтение его атрибутов и расширенных атрибутов, использование объекта для синхронизации;

• запись, включающая в себя чтение DACL объекта, запись и добавление данных в объект, запись его атрибутов и расширенных атрибутов, использование объекта для синхронизации;

• выполнение, включающее в себя чтение DACL объекта, чтение его атрибутов, выполнение программного файла и использование объекта для синхронизации;

• все действия с объектом.

Процесс преобразования общего права доступа к объекту в набор специальных и стандартных прав называется отображением права доступа. Преимуществом общих прав доступа является то, что при их установке владелец объекта может ничего не знать об особенностях этого типа объектов (например, владелец может не знать, что чтение данных из объекта и чтение его атрибутов реализуются с помощью разных методов доступа).

При попытке доступа к объекту субъект может сделать запрос о предоставлении ему максимально возможных прав доступа, указав

в качестве маски доступа константу MAXIMUM_ALLOWED. В этом случае субъект может открыть объект без детального анализа определенных для него прав доступа, который будет выполняться операционной системой в процессе проверки конкретных прав доступа субъекта к объекту. Маска доступа MAXIMUM_ALLOWED, естественно, не может использоваться в элементах DACL.

Маска доступа, содержащаяся в элементе DACL, представляет собой значение длиной 32 бита. Первые 16 битов определяют специальные права доступа; биты с 16 до 23 — стандартные права доступа; бит 24 — право ACCESS_SYSTEM_SECURITY (см. подразд. 3.4); бит 25 — право MAXIMUM_ALLOWED; биты 26 и 27 зарезервированы для дальнейшего использования; биты с 28 по 31 определяют общие права доступа, отображаемые в специальные и стандартные права при попытке доступа к объекту.

Маркер доступа субъекта, обращающегося к некоторому объекту КС, поступает в локальную службу безопасности LSA (см. рис. 3.8). От LSA маркер доступа поступает к монитору безопасных ссылок (security reference monitor, SRM), который просматривает DACL из дескриптора безопасности SD соответствующего объекта и принимает решение R о предоставлении доступа субъекту или отказе в доступе (рис. 3.16). Получив от SRM результат R, LSA передает его субъекту, запросившему доступ к объекту.

Монитор безопасных ссылок использует следующий алгоритм проверки запрошенных субъектом прав доступа к объекту.

1. Если SID из маркера доступа субъекта AT не совпадает с SID, содержащемся в элементе ACE списка контроля доступа к объекту, то осуществляется переход к следующему ACE, в противном случае — переход к п. 2.

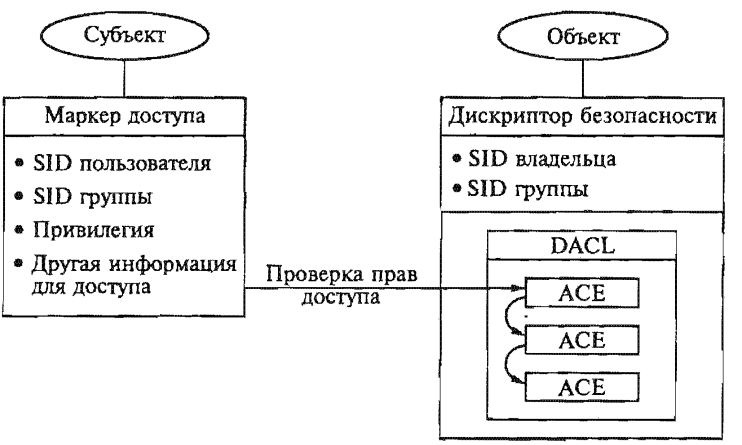

Рис. 3.16. Проверка прав доступа субъекта к объекту

2. Если в элементе ACE запрещается доступ к объекту для субъекта с данным SID, но этот субъект является владельцем объекта (его идентификатор безопасности совпадает с SID владельца из дескриптора безопасности SD объекта) и запрашиваемая маска доступа содержит только попытку доступа к объекту по методу «чтение (или) изменение дискреционного списка контроля доступа к объекту», то доступ субъекта к объекту разрешается, в противном случае — осуществляется переход к п. 3.

3. Если в элементе ACE запрещается доступ к объекту для субъекта с данным SID, то сравниваются запрашиваемая маска доступа и маска доступа, определенная в ACE. Если при сравнении находится хотя бы один общий метод доступа, то попытка доступа субъекта к объекту отклоняется, в противном случае — происходит переход к следующему ACE.

4. Если в элементе ACE разрешается доступ к объекту для субъекта с данным SID, то также сравниваются запрашиваемая маска доступа и маска доступа, определенная в ACE. Если при этом маски доступа полностью совпадают, то доступ субъекта к объекту разрешается, в противном случае — происходит переход к следующему ACE.

5. Если достигнут конец списка DACL из дескриптора безопасности объекта, то попытка доступа субъекта к объекту отклоняется.

Из приведенного алгоритма ясно, что если DACL объекта пуст, то любой доступ к нему запрещен всем субъектам, за исключением владельца объекта, которому разрешены чтение и (или) изменение списка контроля доступа к объекту. Эта особенность может быть использована для защиты от несанкционированного доступа к объекту с конфиденциальной информацией со стороны запущенных пользователем программ (например, компьютерных игр), которые не должны иметь доступ к данному объекту, но содержат вредоносные закладки, пытающиеся прочитать и сохранить данные из объектов, чьим владельцем является запустивший их на выполнение пользователь. Для предотвращения подобной угрозы владелец объекта должен исключить из списка DACL данного объекта все элементы (или явно запретить любые права доступа к объекту для всех субъектов, включая самого себя). После этого любой запущенной им программе, содержащей закладку, будет отказано в доступе к объекту. При необходимости же получить действительно необходимый доступ к данному объекту его владелец сможет снова внести изменения в его DACL.

К сожалению, рассмотренный способ защиты от несанкционированного доступа не предотвратит угрозы, исходящей из закладки, внедренной в программу, использующую информацию из защищаемого объекта (например, в бухгалтерскую программу).

Если у объекта КС нет дескриптора безопасности (например, у папок и файлов, размещенных на дисках под управлением файло-

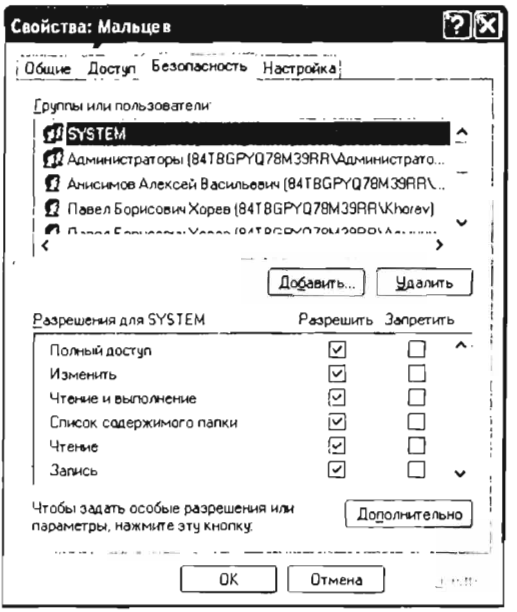

Рис. 3.17. Определение прав доступа к папке

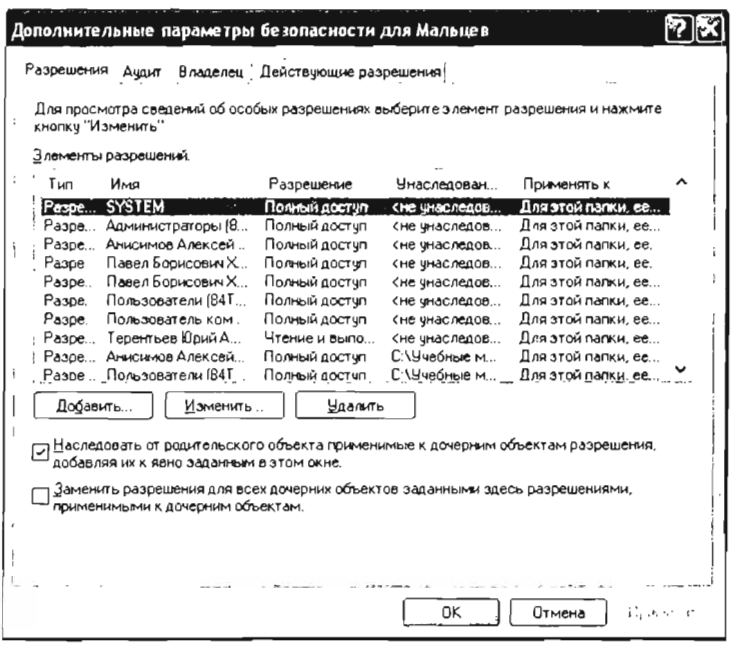

Рис. 3.18. Просмотр и редактирование любых прав доступа к объекту

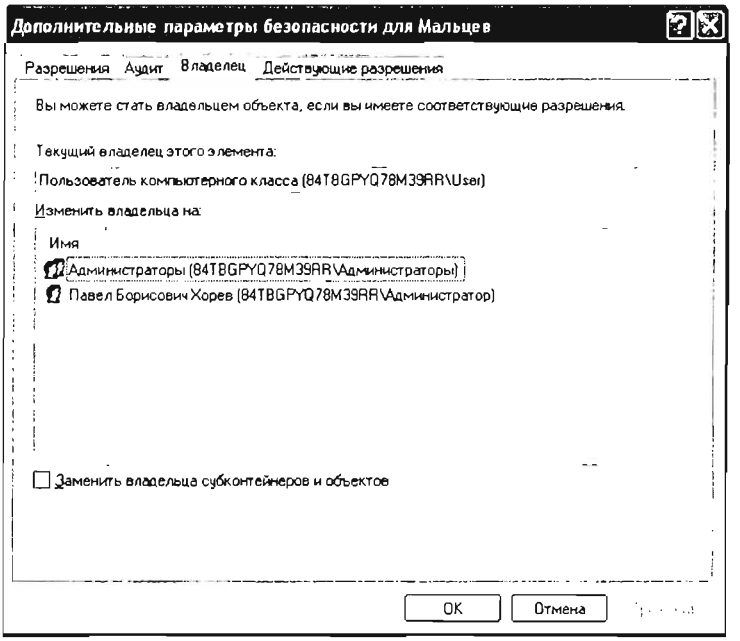

Рис. 3.19. Просмотр и изменение информации о владельце объекта

вой системы FAT), то любые пользователи и группы могут получить любые права доступа к данному объекту.

Пользователи КС для назначения субъектам КС прав доступа к файлам и папкам на дисках с файловой системой NTFS, чьими создателями-владельцами они являются, должны применять средства проводника Windows. Для этого (рис. 3.17) выполняются команды «Общий доступ и безопасность» или «Свойства» контекстного меню папки либо команда «Свойства» контекстного меню файла (в операционной системе Windows XP необходимо выключить режим «Использовать простой общий доступ к файлам» на вкладке «Вид» окна свойств папки). Кнопки «Добавить» и «Удалить» позволяют изменять число элементов ACE в списке DACL, а в окне «Разрешения для *имя субъекта*» можно устанавливать общие права доступа к объекту конкретным пользователям и группам.

Нажатие кнопки «Дополнительно» (см. рис. 3.17) позволяет отобразить окно настроек дополнительных параметров безопасности для объекта. На вкладке «Разрешения» (рис. 3.18) можно просмотреть и при необходимости изменить любые (в том числе и специальные) права доступа к объекту. На вкладке «Владелец» (рис. 3.19) можно просмотреть и при наличии соответствующей привилегии изменить информацию о владельце объекта (записать в SID владельца в дескрипторе безопасности объекта свой SID).

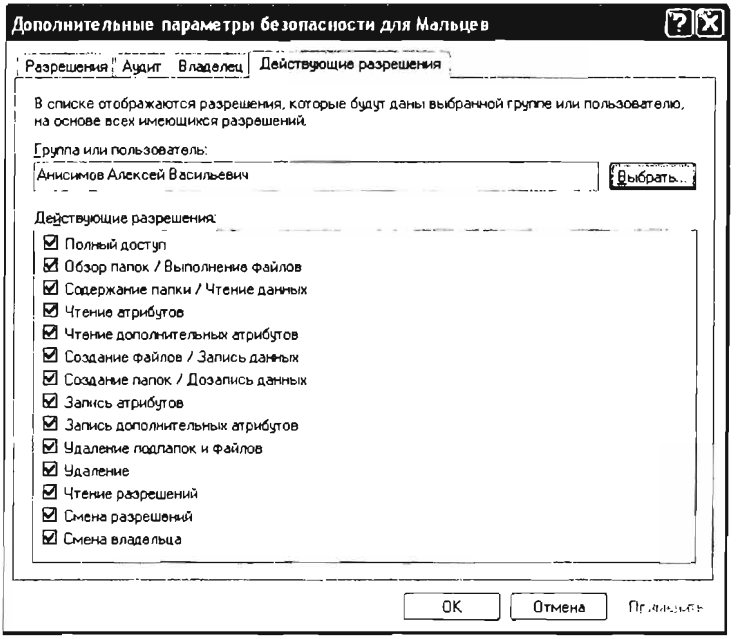

Рис. 3.20. Проверка прав доступа к объекту для конкретного субъекта

На вкладке «Действующие разрешения» (рис. 3.20) можно проверить, какие права доступа к объекту установлены для конкретного пользователя или группы, которые выбираются с помощью кнопки «Выбрать».

Установка прав доступа субъектов КС к установленным в системе принтерам производится аналогично с помощью команды «Свойства» контекстного меню принтера. Разграничение доступа субъектов к разделам реестра Windows XP производится с помощью системной программы regedit (команда «Разрешения» меню «Правка»), а в операционных системах Windows NT/2000 — с помощью системной программы regedt32 (меню «Безопасность»). Отметим, что разграничение доступа к разделам реестра возможно при любой используемой для хранения реестра файловой системе.

Заметим, что проводник Windows и редактор системного реестра правильно отображают список DACL объекта только при выполнении двух условий:

- все элементы ACE, запрещающие доступ к объекту, находятся в «голове» списка DACL;
- в элементах ACE, запрещающих доступ субъектов к объекту, запрещены все права доступа, а не избранные.

Если эти условия не соблюдаются, то системные программы Windows предлагают пользователю, обладающему соответствую-

щими правами, изменить список DACL для выполнения указанных условий.

Среди объектов доступа в защищенных версиях операционной системы Windows различаются контейнерные и неконтейнерные объекты. Контейнерный объект, например папка, имеет логические связи с другими объектами (вложенными папками и файлами), которые могут наследовать определенные права доступа от своего родительского объекта. По умолчанию изменение прав доступа к папке автоматически распространяется на права доступа к файлам этой папки, но не на вложенные в нее другие папки. При наследовании права доступа, установленные для дочерних объектов, могут добавляться к правам доступа, установленным для родительского объекта, или полностью заменяться ими (см. рис. 3.18).

Назначение дескрипторов безопасности вновь создаваемым объектам в защищенных версиях операционной системы Windows производится:

1) на основе явно заданного субъектом и корректного по форме дескриптора безопасности (например, при вызове системных функций CreateFile или CreateDirectory при создании файлов или папок, при вызове системной функции RegCreateKeyEx при создании раздела реестра и т. п.);

2) на основе механизма наследования (если при создании объекта дескриптор безопасности не задается);

3) из маркера доступа субъекта, создающего объект (если наследование невозможно).

Рассмотрим подробно механизм наследования прав доступа из дескриптора безопасности родительского объекта. Наследование дескриптором безопасности дочернего объекта элементов ACE из списка DACL дескриптора безопасности объекта-родителя зависит от следующих флагов, содержащихся в заголовке ACE:

- CONTAINER_INHERIT_ACE — данный ACE наследуется контейнерным дочерним объектом;
- OBJECT_INHERIT_ACE — данный ACE наследуется неконтейнерным дочерним объектом;
- INHERIT_ONLY_ACE — данный ACE не используется при проверке прав доступа к родительскому объекту и используется только при наследовании;
- NO_PROPAGATE_INHERIT_ACE — флаги OBJECT_INHERIT_ACE и CONTAINER_INHERIT_ACE сбрасываются при наследовании данного ACE.

Если в маске доступа ACE родительского объекта содержатся общие права доступа, то перед наследованием данного ACE происходит их отображение.

Приведем прототипы функций из набора Windows API, предназначенных для работы с дескрипторами безопасности объектов и проверки прав доступа субъектов к объектам:

• DWORD GetNamedSecurityInfo(LPTSTR pObjectName, SE_OBJECT_TYPE ObjectType, SECURITY_INFORMATION SecurityInfo, PSID *ppsidOwner, PSID *ppsidGroup, PACL *ppDacl, PACL *ppSacl, PSECURITY_DESCRIPTOR *ppSecurityDescriptor); /* *получение информации типа* SecurityInfo *(о собственнике, его группе, DACL и (или) SACL) из дескриптора безопасности объекта* pObjectName *типа* ObjectType *(файл или папка, раздел реестра, принтер и т.п.) как значений параметров* ppsidOwner *(SID владельца объекта)*, ppsidGroup *(SID его группы)*, ppDacl *(DACL) и* ppSacl *(SACL); в* ppSecurityDescriptor *помещается указатель на дескриптор безопасности объекта */*

• DWORD GetSecurityInfo(HANDLE handle, SE_OBJECT_TYPE ObjectType, SECURITY_INFORMATION SecurityInfo, PSID *ppsidOwner, PSID *ppsidGroup, PACL *ppDacl, PACL *ppSacl, PSECURITY_DESCRIPTOR *ppSecurityDescriptor); /* *получение информации из дескриптора безопасности объекта, заданного не своим именем, а дескриптором* handle */

• DWORD SetNamedSecurityInfo(LPTSTR pObjectName, SE_OBJECT_TYPE ObjectType, SECURITY_INFORMATION SecurityInfo, PSID psidOwner, PSID psidGroup, PACL pDacl, PACL pSacl); /* *установка значений в дескрипторе безопасности объекта с именем* pObjectName */

• DWORD SetSecurityInfo(HANDLE handle, SE_OBJECT_TYPE ObjectType, SECURITY_INFORMATION SecurityInfo, PSID psidOwner, PSID psidGroup, PACL pDacl, PACL pSacl); /* *установка значений в дескрипторе безопасности объекта с дескриптором* handle */

• BOOL GetAce(PACL pAcl, DWORD dwAceIndex, LPVOID *pAce); /* *получение ACE с индексом* dwAceIndex *из списка* pAcl */

• BOOL AddAce(PACL pAcl, DWORD dwAceRevision, DWORD dwStartingAceIndex, LPVOID pAceList, DWORD nAceListLength); /* *добавление к списку ACL* pAcl, *начиная с индекса* dwStartingAceIndex, *элементов ACE, содержащихся в списке* pAceList *длины* nAceListLength *байтов; значение параметра* dwAceRevision *определяет версию ACL */*

• BOOL AddAccessAllowedAce(PACL pAcl, DWORD dwAceRevision, DWORD AccessMask, PSID pSid); /* *добавление к DACL* pAcl *ACE, разрешающего субъекту с идентификатором безопасности* pSid *доступ к объекту с использованием прав, определенных в маске* AccessMask */

• BOOL AddAccessDeniedAce(PACL pAcl, DWORD dwAceRevision, DWORD AccessMask, PSID pSid); /* *добав-*

ление к DACL pAcl *ACE, запрещающего субъекту с идентификатором безопасности* pSid *доступ к объекту с использованием прав, определенных в маске* AccessMask */

- BOOL DeleteAce(PACL pAcl, DWORD dwAceIndex); /* *удаление из ACL* pAcl *элемента с индексом* dwAceIndex */
- DWORD GetExplicitEntriesFromAcl(PACL pacl, PULONG pcCountOfExplicitEntries, PEXPLICIT_ACCESS * pListOfExplicit Entries); /* *получение* pcCountOfExplicit Entries *элементов ACE в списке* pListOfExplicitEntries *из ACL* pacl */
- VOID BuildExplicitAccessWithName(PEXPLICIT_ACCESS pExplicitAccess, LPTSTR pTrusteeName, DWORD AccessPermissions, ACCESS_MODE AccessMode, DWORD Inheritance); /* *подготовка данных для ACE* pExplicitAccess *типа* AccessMode: *имени субъекта* pTrusteeName, *прав его доступа* AccessPermissions *и флагов наследования* Inheritance */
- DWORD SetEntriesInAcl(ULONG cCountOfExplicitEntries, PEXPLICIT_ACCESS pListOfExplicitEntries, PACL OldAcl, PACL * NewAcl); /* *добавление* cCountOfExplicitEntries *элементов ACE из списка* pListOfExplicitEntries *в существующий ACL* OldAcl; *в* NewAcl *возвращается указатель на измененный DACL */
- DWORD GetEffectiveRightsFromAcl(PACL pacl, PTRUSTEE pTrustee, PACCESS_MASK pAccessRights); /* *получение в* pAccessRights *прав доступа субъекта* pTrustee *из DACL* pacl */
- VOID MapGenericMask(PDWORD AccessMask, PGENERIC_MAPPING GenericMapping); /* *отображение общих прав доступа из* GenericMapping *в маску специальных и стандартных прав доступа* AccessMask */
- BOOL AccessCheck(PSECURITY_DESCRIPTOR pSecurity Descriptor, HANDLE ClientToken, DWORD DesiredAccess, PGENERIC_MAPPING GenericMapping, PPRIVILEGE_SET PrivilegeSet, LPDWORD PrivilegeSetLength, LPDWORD GrantedAccess, LPBOOL AccessStatus); /* *проверка разрешения субъекту с маркером доступа* ClientToken *доступа с применением специальных и стандартных прав* DesiredAccess *и общих прав* GenericMapping *к объекту с дескриптором безопасности* pSecurityDescriptor; *при успешном завершении в буфер* PrivilegeSet *длиной* PrivilegeSetLength *будут помещены привилегии субъекта, использованные при проверке, в* GrantedAccess – *маска прав доступа к объекту, разрешенных для данного субъекта, и в* AccessStatus – *признак успешной проверки запрошенных прав доступа */*

• `BOOL AreAllAccessesGranted(DWORD GrantedAccess, DWORD DesiredAccess); /* проверка принадлежности всех запрашиваемых прав доступа из DesiredAccess маске разрешенных прав доступа GrantedAccess */`

• `BOOL AreAnyAccessesGranted(DWORD GrantedAccess, DWORD DesiredAccess); /* проверка принадлежности хотя бы одного из запрашиваемых прав доступа DesiredAccess маске разрешенных прав доступа GrantedAccess */`

К недостаткам разграничения доступа к объектам КС, реализованного в защищенных версиях операционной системы Windows, можно отнести то, что пользователи КС не имеют средств управления правами доступа субъектов к объектам других типов (процессам и потокам, сервисам и их диспетчеру, рабочим столам и оконным станциям, портам, маркерам доступа, объектам синхронизации процессов и потоков и др.).

Для предоставления пользователям возможности запуска только определенных приложений в операционных системах Windows NT/2000 могут использоваться средства редактора системных правил (см. подразд. 3.1). В операционной системе Windows XP Professional администратор может дополнительно использовать параметры политики ограниченного использования программ, которые предоставляют собой механизм идентификации программ и управления возможностями их выполнения.

Политика ограниченного использования программ состоит из применяемого по умолчанию правила, используемого при запуске приложений пользователями, и списка (возможно, пустого) программ, являющихся исключением из данного правила.

Применяемыми по умолчанию правилами могут быть:

• разрешение на выполнение пользователями всех программ, за исключением некоторых;

• запрещение выполнения пользователями всех программ, кроме некоторых.

Список программ, являющихся исключением из применяемого по умолчанию правила, формируется на основе одного из следующих правил идентификации программ:

• правила хеша (рис. 3.21), в соответствии с которым программа идентифицируется вычисленным для нее хеш-значением (файл программы может быть переименован или перемещен в другую папку, но его хеш-значение при этом не изменится);

• правило пути, в соответствии с которым программа идентифицируется полным путем к своему файлу (например: C:\Program Files\Microsoft Office\Office10\WINWORD.EXE) или путем к папке, в которой этот файл находится (например: C:\Windows\System32);

• правило сертификата, в соответствии с которым программа идентифицируется сертификатом ее издателя (см. подразд. 4.8),

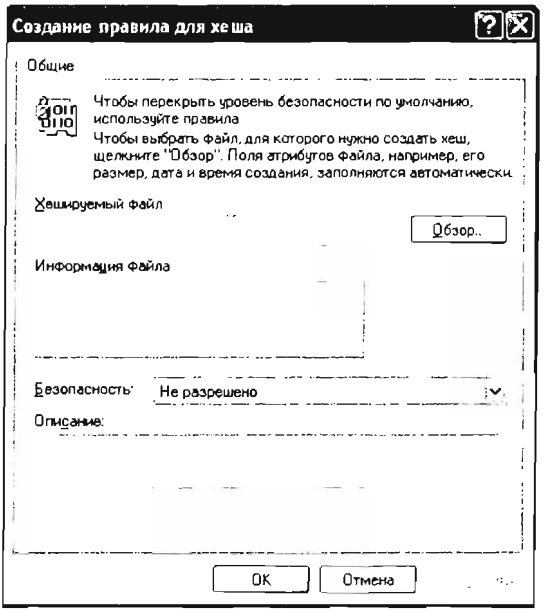

Рис. 3.21. Пример определения правила идентификации программ

который был использован для создания электронной цифровой подписи этой программы;

• правило зоны безопасности, в соответствии с которым программа идентифицируется зоной безопасности сети Интернет, из которой она поступила.

Для предотвращения угрозы, связанной с возможным доступом нарушителя к терминалу временно отсутствующего пользователя, необходимо использовать функцию «Блокировка», которая становится доступной после нажатия комбинации клавиш Ctrl+Alt+Delete. Возобновить сеанс работы с защищенными версиями Windows в этом случае сможет только пользователь, заблокировавший рабочую станцию, или администратор системы.

В соответствии с требованиями «Оранжевой книги» (см. подразд. 1.6) для КС, отнесенных к классу защищенности С2 и выше, требуется, чтобы никакая (в том числе и зашифрованная) информация о действиях одного субъекта не могла быть получена другим субъектом, получившем доступ к КС после первого субъекта. В руководящих документах Гостехкомиссии России (см. подразд. 1.6) для класса защищенности 1Г и выше предъявляется аналогичное требование к обязательной очистке освобождаемых областей оперативной памяти ЭВМ и внешних накопителей. Рассмотрим способы аппаратной защиты инициированных субъектами процессов в защищенных версиях операционной системы Windows.

Ядро и драйверы устройств этих операционных систем выполняются в нулевом кольце защиты, а весь остальной системный код и приложения пользователей — в третьем кольце защиты. Первое и второе кольца защиты не используются. Каждый процесс получает адресное пространство размером 4 гигабайта, рассматриваемое как шесть совпадающих друг с другом сегментов: код, данные и стек для третьего кольца защиты, код, данные и стек для нулевого кольца защиты.

Наиболее вероятной ошибкой в программах, которая приводит к несанкционированному изменению данных, хранящихся в оперативной памяти по произвольному адресу, является использование указателей, не получивших начального значения и связанных с близкими к нулевым адресами памяти. Но в защищенных версиях Windows все адреса от –65 536 до 65 535 принадлежат зарезервированным зонам, оперативная память из которых не должна использоваться ни в какой ситуации. Поэтому выявление программных ошибок, связанных с неправильным использованием указателей, не является трудоемким.

Еще одна зарезервированная зона адресов памяти предназначена для предотвращения переполнения адресного пространства процесса.

Адресное пространство процесса содержит также системную область адресов, обратиться к которой можно только из кода, содержащегося в нулевом кольце защиты. Включение системного пространства адресов в адресное пространство процесса позволяет упростить обработку аппаратных прерываний в коде, выполняющемся в третьем кольце защиты.

Оставшаяся часть адресного пространства процесса (помимо зарезервированных зон и области системных адресов) предназначена для кода и данных самого процесса. В эту же часть проецируется и выделенная процессу дополнительная оперативная память.

Таким образом, в защищенных версиях операционной системы Windows реализован подход, в соответствии с которым каждому процессу выделяется индивидуальное адресное пространство, которое аппаратно изолировано от адресных пространств других процессов. В этом случае, какой бы адрес оперативной памяти не использовался в процессе, невозможно обращение к памяти, выделенной другому процессу, так как одному и тому же значению адреса в разных адресных пространствах соответствуют различные физические адреса оперативной памяти компьютера.

Дополнительно используемый в защищенных версиях Windows подход обеспечивает надежную защиту от случайных, вызванных ошибками в программах обращений процессов к памяти других выполняющихся системных и пользовательских приложений. Но поскольку часть оперативной памяти должна быть разделяемой (проецироваться в адресные пространства различных процессов),

сохраняется угроза преднамеренного несанкционированного доступа одного процесса к части адресного пространства другого процесса, относящейся к разделяемой памяти. Исключение возможности использования разделяемой памяти приведет к существенному усилению требований операционной системы к размеру оперативной памяти компьютера, а производительность КС значительно снизится.

3.4. Аудит событий безопасности в защищенных версиях операционной системы Windows

В соответствии с требованиями «Оранжевой книги» (см. разд. 1.6) в КС, отнесенных к классу защищенности С2 и выше, должен быть организован аудит событий, связанных с безопасностью системы. К основным требованиям политики аудита в КС относятся:

- ассоциирование пользователя с любым событием аудита, что обеспечивает индивидуальную ответственность пользователей за их действия в КС;
- обязательность аудита стандартного набора событий — идентификации и аутентификации пользователя, размещения объектов в адресном пространстве процессов, уничтожения объектов, действий привилегированного пользователя, любого изменения

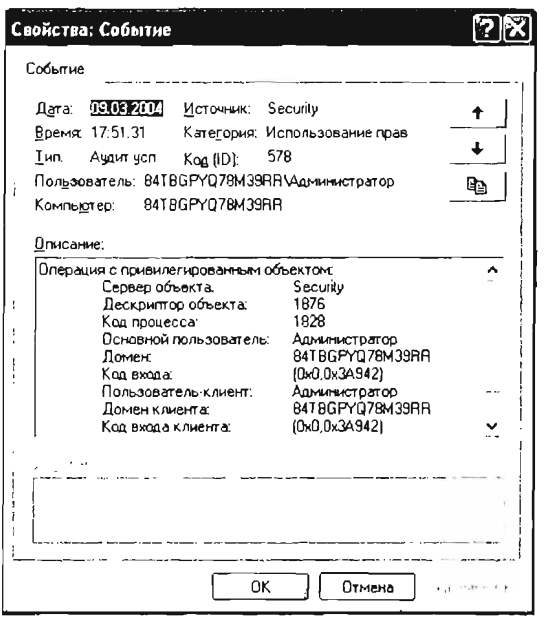

Рис. 3.22. Просмотр одной записи журнала аудита событий безопасности

меток конфиденциальности объектов читаемого вывода (при использовании в КС мандатного управления доступом к объектам);

- наличие необходимого набора атрибутов записи журнала аудита — даты и времени события, логического имени инициировавшего событие пользователя, типа события, признака успешного или неудачного завершения вызвавшего событие действия, имени связанного с событием объекта;
- возможность фильтрации записей журнала аудита;
- поддержка и защита от несанкционированного доступа журнала аудита.

Аналогичные по сути требования предъявляются к КС и в руководящих документах Гостехкомиссии России (см. разд. 1.6), начиная с класса защищенности 1Д и выше. Из дополнительных требований к подсистеме регистрации и учета, предъявляемых в классе защищенности 1Г, можно, например, указать необходимость наличия в записи регистрационного журнала кода или пароля, предъявленного при неудачной попытке входа в систему, и необходимость регистрации запуска (завершения) процессов, предназначенных для обработки защищаемых файлов.

Рассмотрим выполнение требований в подсистеме аудита защищенных версий операционной системы Windows. Журнал аудита содержится в файле windows\System32\Config\ secevent.evt, доступ осуществляется с помощью функции «Просмотр событий» панели управления Windows. На рис. 3.22 приведен пример просмотра одной записи журнала аудита событий безопасности.

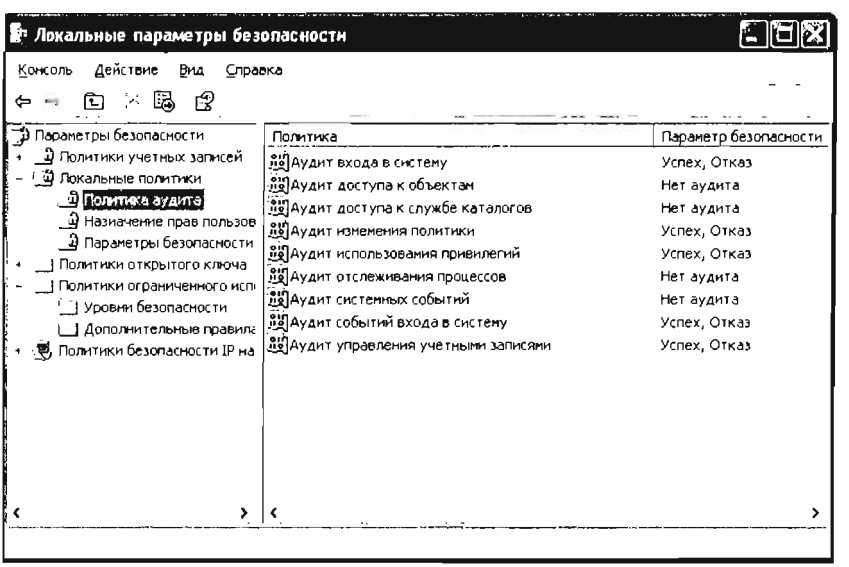

Рис. 3.23. Определение параметров политики аудита

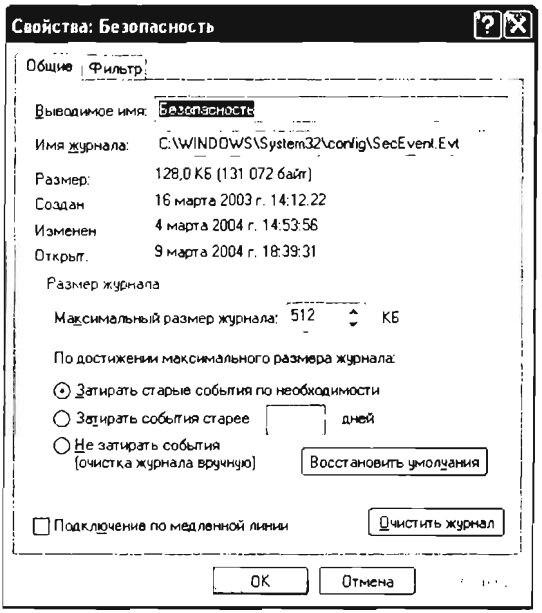

Рис. 3.24. Определение свойств журнала аудита событий безопасности

Значения параметров политики аудита могут быть заданы в окне задания значений параметрам локальной политики безопасности (см. рис. 3.14), в специальном окне определения значений параметрам аудита (рис. 3.23) и в окне свойств самого журнала аудита событий безопасности при его просмотре (рис. 3.24).

С помощью задания значений параметрам безопасности и аудита администратор указывает, какие события должны регистрироваться в журнале аудита.

Возможна регистрация следующих событий:

- вход пользователей в систему;
- доступ субъектов к объектам;
- доступ к службе каталогов Active Directory;
- изменение политики безопасности;
- использование привилегий;
- отслеживание процессов;
- системные события;
- попытки входа в систему;
- управление учетными записями пользователей и групп;
- доступ к глобальным системным объектам;
- использование прав на архивацию и восстановление объектов.

Для каждой категории регистрируемых событий администратор может указать тип события (успешное и (или) неудачное завершение) либо отменить его регистрацию в журнале аудита. При

аудите использования привилегий регистрируются попытки использования не всех возможных привилегий, а лишь тех, которые считаются потенциально опасными с точки зрения разработчиков подсистемы безопасности защищенных версий Windows (например, создание маркерного объекта или создание журналов безопасности). Следует отметить, что не все объективно опасные привилегии входят в этот список.

К системным событиям, которые могут регистрироваться в журнале аудита, относятся:

- перезагрузка операционной системы;
- завершение работы операционной системы;
- загрузка пакета аутентификации;
- запуск процесса входа (Winlogon);
- сбой при регистрации события в журнале аудита;
- очистка журнала аудита;
- загрузка пакета оповещения об изменении в списке пользователей.

Параметрами журнала аудита, которые может изменить администратор, являются (см. рис. 3.24) максимальный размер журнала и реакция операционной системы на его переполнение:

- затирать старые события при необходимости (реакция по умолчанию);

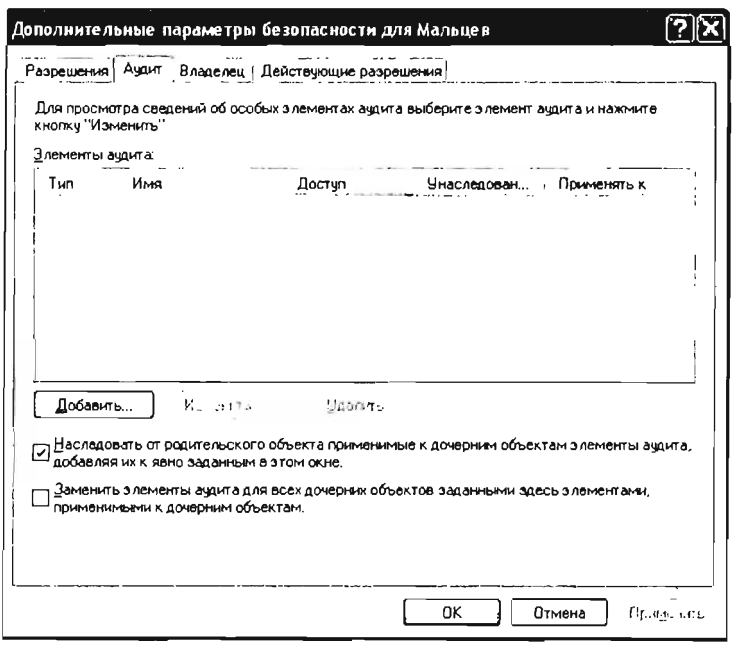

Рис. 3.25. Определение параметров аудита попыток доступа к объекту

• затирать старые события, которые произошли ранее установленного числа дней (в этом случае новые события не регистрируются, пока не истечет заданное число дней с момента регистрации самого старого события);

• не затирать события (очистка журнала вручную).

При выборе последнего типа реакции и возникновении ситуации переполнения журнала аудита последующая загрузка операционной системы становится невозможной для всех пользователей, кроме администратора. Он должен будет очистить журнал аудита (кнопка «Очистить журнал» в окне свойств журнала аудита), восстановить значение параметра реакции системы на переполнение журнала аудита и перезагрузить операционную систему. Если эти действия не будут выполнены, то регистрация новых событий в журнале аудита станет невозможной.

Для регистрации попыток доступа субъекта к объекту в дескрипторе безопасности объекта предусмотрен системный список контроля доступа SACL (см. подразд. 3.3), содержимое которого формируется администратором системы (рис. 3.25). Элементы ACE списка SACL имеют один и тот же тип SYSTEM_AUDIT_ACE и содержат заголовок ACE, маску регистрируемых в журнале аудита прав доступа и SID пользователя или группы, чьи попытки доступа к объекту должны регистрироваться (если в ACE не указан SID, то регистрируются попытки доступа к объекту всех пользователей). Еще один тип элементов ACE списка SACL — SYSTEM_ALARM_ACE — зарезервирован для дальнейшего использования.

В заголовке ACE списка SACL могут использоваться два дополнительных флага: FAILED_ACCESS_ACE_FLAG (будут регистрироваться неудачные попытки получения доступа к объекту) и SUCCESSFUL_ACCESS_ACE_FLAG (будут регистрироваться успешные попытки получить доступ к объекту).

Существует специальное право доступа к объекту ACCESS_SYSTEM_SECURITY, в соответствии с которым субъект может прочитать и изменить системный список контроля доступа к объекту. Обладание этим правом полностью зависит от обладания соответствующей привилегией субъекта доступа и не может изменяться в зависимости от конкретного объекта.

В дополнение к приведенным в подразд. 3.3 системным функциям для работы со списками контроля доступа к объектам предусмотрены отдельные функции для работы с SACL:

• `BOOL AddAuditAccessAce(PACL pAcl, DWORD dwAceRevision, DWORD dwAccessMask, PSID pSid, BOOL bAuditSuccess, BOOL bAuditFailure); /* добавление в SACL pAcl версии dwAceRevision элемента ACE, содержащего маску регистрируемых прав доступа dwAccessMask,`

идентификатор безопасности субъекта pSid, *признак регистрации успешных попыток доступа* bAuditSuccess *и признак регистрации неудачных попыток доступа* bAuditFailure */

• BOOL AccessCheckAndAuditAlarm(LPCTSTR SubsystemName, LPVOID HandleId, LPTSTR ObjectTypeName, LPTSTR ObjectName, PSECURITY_DESCRIPTOR SecurityDescriptor, DWORD DesiredAccess, PGENERIC_MAPPING GenericMapping, BOOL ObjectCreation, LPDWORD GrantedAccess, LPBOOL AccessStatus, LPBOOL pfGenerateOnClose); /* *проверка подсистемой* SubsystemName (*например, "Win32"*) *наличия у субъекта* HandleId *специальных и стандартных* (DesiredAccess) *и общих* (GenericMapping) *прав доступа к существующему или новому* (*признак* ObjectCreation) *объекту типа* ObjectTypeName *с именем* ObjectName *и дескриптором безопасности* SecurityDescriptor *с регистрацией соответствующего события аудита; в* GrantedAccess *будет помещена маска разрешенных субъекту прав доступа, в* AccessStatus *— результат проверки и в* pfGenerateOnClose *— признак необходимости зарегистрировать событие закрытия объекта */

Добавление записей в журнал аудита производится с помощью вызовов системных функций из набора Windows API ObjectOpenAuditAlarm (генерация события аудита при попытке открытия или создания нового объекта), ObjectCloseAuditAlarm (генерация события аудита при закрытии объекта), ObjectDeleteAuditAlarm (генерация события аудита при удалении объекта), ObjectPrivilegeAuditAlarm (генерация события аудита при попытке субъекта применить привилегии при доступе к уже открытому объекту), PrivilegedServiceAuditAlarm (генерация события аудита при попытке использования субъектом привилегий), AccessCheck AndAuditAlarm (рассмотрена ранее) и др.

Для регистрации событий в журнале аудита субъект должен обладать соответствующей привилегией, которая по умолчанию предоставлена только псевдопользователю System. Назначать эту привилегию другим субъектам нецелесообразно, поскольку это может привести к записи в журнал аудита сфабрикованной информации, компрометирующей других пользователей КС и скрывающей следы возможного несанкционированного доступа к информации.

При определении политики аудита следует понимать, что адекватной угрозам безопасности информации в КС будет не политика, которая предполагает регистрацию многих событий, а политика, при которой будут регистрироваться необходимые события.

К таким событиям можно отнести следующие:

• вход и выход пользователей;

• доступ к объектам с конфиденциальной информацией со стороны пользователей, в отношении которых имеются обоснованные подозрения о попытках несанкционированного доступа;

• изменения в списке зарегистрированных пользователей и политике безопасности КС;

• запуск и завершение процессов при наличии обоснованных подозрений о заражении системных программ вредоносным кодом.

Для обеспечения безопасности информации в КС целесообразно разделить полномочия администраторов КС и аудиторов (пользователей с правами доступа к файлу аудита). Если этого не сделать, то возникнет ситуация, при которой установка параметров политики безопасности и проверка ее соблюдения сосредоточатся в одних руках.

Для создания отдельной от администраторов группы аудиторов одному из администраторов системы (будущему аудитору) необходимо:

1) создать группу «Аудиторы» (см. подразд. 3.3) и включить в ее состав пользователей с соответствующими правами;

2) назначить владельцем файла аудита одного из членов группы аудиторов (см. подразд. 3.3);

3) разрешить полный доступ к файлу аудита членам группы «Аудиторы» и полный доступ псевдопользователю System, а всем другим пользователям доступ запретить;

4) предоставить группе «Аудиторы» привилегию управления аудитом и журналом безопасности, отменив использование этой привилегии для членов группы «Администраторы»;

5) исключить свою учетную запись из состава группы «Администраторы».

После этого просматривать и очищать журнал аудита, а также управлять списками SACL объектов доступа смогут только члены группы аудиторов КС. Полномочия на изменение значений параметров политики аудита при этом сохранятся у членов группы администраторов КС.

Недостатки подсистемы аудита защищенных версий операционной системы Windows:

• защита от несанкционированного доступа к файлу аудита обеспечивается только средствами разграничения доступа без применения шифрования;

• отсутствуют простые средства разграничения полномочий администраторов и аудиторов КС;

• некоторые важные события безопасности (например, запуск драйвера устройства) не могут регистрироваться в журнале аудита;

• объекты КС, которые недоступны для стандартных программ администрирования (см. подразд. 3.3), имеют SACL, что приводит к регистрации в файле аудита лишних событий.

3.5. Защита информации от несанкционированного доступа в операционных системах семейства Unix

Учетные записи пользователей операционных систем клона Unix хранятся в файле /etc/passwd. Учетная запись пользователя *i* имеет следующий формат:

логическое имя пользователя ID_i : *хеш-значение его пароля* $H(P_i)$: *системный идентификатор пользователя* UID_i : *системный идентификатор первичной группы пользователя* GID_i : *полное имя и должность пользователя* D_i : *домашний (рабочий) каталог пользователя* HD_i : *командный процессор (оболочка), применяемый пользователем,* SH_i

Для добавления учетной записи нового пользователя администратором применяется команда adduser с параметром, равным уникальному логическому имени пользователя длиной до восьми символов. При работе в системе пользователь полностью идентифицируется своим системным идентификатором UID, поэтому два пользователя с одинаковым идентификатором будут обладать совершенно одинаковыми правами в системе. Как правило, каждому пользователю назначается уникальный UID.

В большинстве Unix-подобных систем предопределены следующие учетные записи пользователей:

- root — суперпользователь (администратор), обладающий всеми правами в системе;
- daemon или sys — псевдопользователь, связанный с сервисами операционной системы;
- guest — анонимный пользователь КС;
- nobody — псевдопользователь, не являющийся владельцем объектов и обладающий минимальными правами в системе;
- agent, ftp, uucp, news, lp — псевдопользователи, связанные соответственно с сервисами электронной почты, FTP, UUCP, Usenet и печати и др.

В учетных записях псевдопользователей в поле хеш-значения пароля помещается '*', что не позволяет применять эти логические имена для входа в систему. Поскольку привилегии пользователя в КС определяются не его логическим именем, а значением UID, вход в систему пользователя с именем root и с системным идентификатором, отличным от нуля, не обеспечит ему привилегий суперпользователя. С другой стороны, вход в систему пользователя с произвольным логическим именем и с UID, равным нулю, даст ему все полномочия суперпользователя.

Для повышения безопасности КС целесообразно не применять учетную запись root для регулярной работы в системе. Вместо нее следует создать другие учетные записи для пользователей, выполняющих те или иные административные функции (учет пользова-

телей, работа с различными системными сервисами и т. п.). В Unix-системах возможно определение списка терминалов, с которых не может входить в КС суперпользователь.

Пользователи изменяют свои пароли с помощью команды passwd. Этой же командой может воспользоваться и суперпользователь для задания нового пароля (вместо забытого старого) пользователю с указанным в качестве параметра команды passwd именем учетной записи.

Пароли пользователей записываются в файл учетных записей в хешированном виде. Применяется следующий алгоритм хеширования:

1) на основе времени суток генерируется случайное значение, которое затем преобразуется в строку из двух символов и запоминается в файле учетных записей как первые два символа поля с хеш-значением пароля;

2) магическое значение длиной 64 бита, состоящее из нулей или пробелов, зашифровывается по алгоритму DES (см. подразд. 4.4), причем в качестве ключа шифрования длиной 56 бит используется пароль пользователя, а случайное значение (salt) применяется для модификации алгоритма шифрования;

3) полученное значение длиной 64 бита вновь зашифровывается на том же ключе (общее число повторений равно 25);

4) полученное окончательное значение преобразуется в 11 символов (каждым шести битам соответствует один символ из множества {'.', '/', '0'—'9', 'A'—'Z', 'a'—'z'});

5) полученная строка символов записывается в файл учетных записей после случайного значения.

Поскольку пароль используется в алгоритме хеширования в качестве ключа DES-шифрования длиной 56 бит, его минимальную длину целесообразно выбирать равной восьми символам (56 бит в кодировке ASCII). Другим требованием к сложности паролей может быть обязательное наличие в нем строчных и прописных букв, цифр и знаков препинания. Пользователям также может быть запрещено устанавливать собственные пароли, а для входа в систему использовать только сгенерированные ею пароли. В последнем случае генератор паролей может запускаться при выполнении команды passwd, доступ к которой должен быть ограничен.

Для временного отключения учетной записи пользователя администратор может заменить хеш-значение пароля в его учетной записи символом '*'. Для удаления учетной записи пользователя следует удалить его учетную запись из файла /etc/passwd, а также удалить все файлы пользователя и его домашний каталог с помощью команд:

```
find / usr / имя домашнего каталога -exec rm {} \;
rmdir имя домашнего каталога
```

Все учетные записи псевдопользователей должны быть отключены.

По умолчанию к файлу /etc/passwd разрешен доступ по чтению для всех пользователей КС. Это необходимо, поскольку сведения об идентификаторах пользователя и группы, о домашнем каталоге и логическом имени пользователя из этого файла должны быть доступны различным программам. Для защиты хеш-значений паролей от чтения непривилегированными пользователями выполняется процедура затенения (shadow) паролей. В этом случае хеш-значения паролей перемещаются из файла /etc/passwd в файл /etc/shadow (/etc/security/passwd.adjunct или /etc/master.passwd в других операционных системах). Запись в файле теневых паролей имеет следующий формат:

логическое имя пользователя : хеш-значение пароля : дата последней замены пароля : число дней до возможной смены пароля : число дней до обязательной смены пароля : количество дней до предупреждения пользователя о необходимости замены пароля : число дней после истечения срока действия пароля, по истечении которых происходит отключение учетной записи : дата отключения учетной записи : зарезервированное поле

В исходном файле учетных записей при использовании затенения паролей в поле хеш-значения пароля помещаются специальные символы или случайная строка символов (для усложнения задачи подбора паролей). Доступ к файлу теневых паролей имеет только привилегированный пользователь.

Для работы с теневыми паролями учетных записей пользователей предназначены специальные команды:

- useradd — добавление учетной записи нового пользователя;
- usermod — замена информации в учетной записи пользователя;
- userdel — удаление учетной записи пользователя и его файлов;
- passwd — отключение учетной записи пользователя (и снятие отключения), установка максимального и минимального сроков действия пароля пользователя, установка числа дней для предупреждения о необходимости замены пароля, установка числа дней между истечением срока действия пароля и отключением учетной записи;
- pwck — проверка корректности исходного файла паролей и файла теневых паролей (корректности числа полей, уникальности логических имен, корректности системных идентификаторов и групп, корректности первичной группы, домашнего каталога и оболочки пользователя, наличия учетных записей с пустыми паролями).

Для изменения устанавливаемой по умолчанию подсистемы аутентификации может использоваться архитектура сменных мо-

дулей аутентификации PAM (pluggable authentication modules). Существуют четыре типа модулей PAM:

- Auth — выполняет функции аутентификации;
- Account — устанавливает возможность аутентификации при выполнении некоторых условий (например, вход в КС может быть разрешен пользователю только в будние дни и рабочие часы);
- Password — задает ограничения на пароли пользователей;
- Session — устанавливает возможность доступа пользователя к определенным сервисам после разрешения входа модулем Account.

Возможно объединение различных модулей одного типа для выполнения нескольких процедур аутентификации. Конфигурационные файлы PAM размещаются в каталоге /etc/pam.d. Приведем пример содержимого файла /etc/pam.d/passwd:

```
# %PAM-1.0
auth required /lib/security/pam_pwdb.so shadow nullok
account required /lib/security/pam_pwdb.so
password required /lib/security/pam_cracklib.so retry=3
password required /lib/security/pam_pwdb.so use_authok
nullok
```

Во второй строке обеспечивается вывод приглашения к вводу пароля и проверка введенной информации в файле теневых паролей. В третьей строке обеспечивается то же самое для случая, когда теневые пароли не используются. Четвертая строка обеспечивает выполнение программы для проверки устойчивости нового пароля от попыток его подбора. В последней строке определяется модуль, который будет вызываться для изменения пароля пользователя.

Атрибут модуля required обеспечивает обязательность вызова модуля вне зависимости от успешности завершения других модулей (при этом порядок вызова модулей не имеет значения). Другими возможными атрибутами модулей PAM являются optional, sufficient и requisite. Если в конфигурационном файле PAM содержатся только модули с атрибутом optional, то успешность (или неудача) их выполнения будут означать успешность (или неудачу) процедуры аутентификации в целом. Атрибут sufficient подобен атрибуту optional, но имеет больший вес для результата аутентификации в целом. Модуль с атрибутом requisite при своем успешном выполнении передает управление другому модулю аутентификации, а в противном случае — непосредственно подсистеме аутентификации (при неудачном завершении такого модуля прочие модули аутентификации не выполняются).

Информация о группах пользователей в операционных системах семейства Unix помещается в файл /etc/group. Каждая запись в этом файле имеет следующий формат:

имя группы : пароль группы : системный идентификатор группы GID : список разделенных запятыми логических имен пользователей – членов группы

В большинстве Unix-систем предопределены следующие группы:

- wheel — группа, членами которой являются администраторы КС (ее GID равен нулю);
- uucp — группа, единственным членом которой является псевдопользователь uucp (ее GID равен 10);
- users — группа, членами которой являются все пользователи КС, чьи учетные записи содержатся в файле паролей (ее GID равен 100).

Полномочия членов группы в системе определяются не именем группы, а ее системным идентификатором. Поэтому если в файле паролей учетная запись пользователя nick содержит GID, равный 100, а в файле групп указано, что nick входит в группу wheel, то пользователь nick не будет иметь в системе административных привилегий. Первичной группой пользователя nick будет в этом случае группа users.

В некоторых Unix-системах доступна команда newgrp, с помощью которой пользователь может включить себя в другую первичную группу, чье имя указано в качестве параметра команды. При успешном завершении этой команды в учетной записи пользователя изменяется значения поля GID. Для предотвращения угрозы превышения нарушителем своих полномочий в системе путем несанкционированного изменения GID введен пароль группы. Если пользователь пытается получить идентификатор группы, членом которой он не является, то при выполнении команды newgrp запрашивается пароль группы, хранящийся в файле /etc/group.

Пароли групп сохраняются в хешированном виде, а алгоритм хеширования полностью соответствует рассмотренному ранее алгоритму хеширования паролей пользователей. Однако в большинстве операционных систем отсутствуют средства назначения паролей группам. Поэтому должен применяться следующий порядок действий:

1) с помощью команды passwd создается пароль для произвольной учетной записи пользователя;

2) с помощью текстового редактора хеш-значение созданного пароля переносится из файла паролей пользователей в файл групп.

Добавление и удаление группы могут происходить редактированием файла групп. Для повышения безопасности информации следует применять затенение паролей групп, аналогичное созданию теневых паролей пользователей (в этом случае хеш-значения паролей групп перемещаются в файл /etc/gshadow или аналогичный). Для проверки корректности исходного файла групп и файла

с их теневыми паролями предназначена команда grpck — проверка числа полей в записях, уникальности имен групп, правильности списков членов групп.

В некоторых Unix-системах не поддерживаются пароли групп, а в соответствующем поле файла групп помещается '*' или это поле остается пустым.

На период своего временного отсутствия пользователи операционных систем семейства Unix должны блокировать терминалы с помощью соответствующих команд (например, lock). Использование средств автоматического блокирования терминала по истечении заданного периода времени менее безопасно из-за возможного внедрения программной закладки, имитирующей блокирование экрана и сохраняющей введенный затем пароль пользователя.

Разграничение доступа к объектам в операционных системах семейства Unix — файлам, каталогам, связям и специальным файлам (символьным или блочным устройствам ввода-вывода и именованным каналам) — осуществляется на основе хранящихся в индексе соответствующего объекта сведений о владельце объекта и его группе, а также о векторе доступа к объекту.

Вектор доступа представляет собой список контроля доступа фиксированной (а не произвольной, как в защищенных версиях Windows) длины. Первый элемент списка определяет права доступа к объекту его владельца, второй — членов его первичной группы, а третий — всех остальных пользователей системы. Суперпользователь root имеет полный доступ ко всем объектам в системе. Каждый элемент вектора доступа имеет длину 3 или 4 бита.

В Unix-системах возможны три метода доступа к объекту: чтение (*r*), запись (*w*) и выполнение (*x*). Для каталогов запись определяет создание и (или) удаление файлов, а выполнение — поиск файла в каталоге по заданному имени.

Пример вектора доступа к файлу:

```
rwxr-xr-
```

(владелец файла имеет право на полный доступ к нему, члены группы владельца — на чтение и выполнение файла, а все остальные пользователи — только на чтение файла).

Для просмотра разрешений на доступ к объекту и информации о его владельце используется команда ls с параметром -l, для изменения разрешений на доступ к объекту — команда chmod, для изменения владельца объекта — команда chown, а для изменения первичной группы владельца — команда chgrp. При изменении прав доступа к объекту может применяться абсолютная форма, при которой элементы вектора доступа записываются в виде восьмеричных цифр. Например, для задания каталогу /usr/home разрешений в соответствии с приведенным ранее примером можно использовать следующую команду:

```
chmod 457 /usr/home
```

Используемое в операционных системах семейства Unix разграничение доступа к объектам достаточно надежно. Для того чтобы без использования системных команд (вручную) изменить права доступа к объекту для конкретного пользователя, необходимо иметь доступ к области индексов файловой системы, которые определены в специальном файле (например, /dev/root). Но индекс этого файла также хранится в области индексов. Поэтому если не изменять права доступа ко всем системным объектам, которые установлены по умолчанию при установке операционной системы (что может сделать только суперпользователь), то можно гарантировать безопасность работы подсистемы разграничения доступа.

Для каждого элемента вектора доступа к объекту может быть задан четвертый бит, определяющий права на выполнение программного файла или каталога. Если четвертый бит установлен в элементе вектора для владельца файла (SUID), то программный файл будет выполняться в сеансе любого пользователя с правами владельца этого файла. Это необходимо, например, при вызове команды passwd пользователем для изменения своего пароля. Обычный пользователь может иметь право смены своего пароля, но не может иметь право записи в файл паролей.

Если четвертый бит установлен в элементе вектора доступа для членов группы владельца (SGID), то данный программный файл будет выполняться в сеансе любого пользователя с правами членов группы владельца данного файла. Если SGID установлен в векторе доступа к каталогу, то все создаваемые пользователем файлы в этом каталоге будут иметь такой же идентификатор группы владельца, как и у каталога.

Нарушитель может с помощью SUID или SGID попытаться получить постоянные полномочия администратора КС, если ему удастся хотя бы один раз получить (перехватить) его пароль. Например:

1) нарушитель создает копию командного процессора в своем домашнем каталоге или еще где-нибудь (например, в максимально низком узле дерева каталогов /usr/tmp)

```
% cp /bin/sh /tmp/break-acct
```

2) нарушитель назначает владельцем созданного файла администратора и с применением его полномочий (после входа в систему под его учетной записью) устанавливает SUID

```
% chown root/tmp/break-acct
% chmod 4755 /tmp/break-acct
```

3) до тех пор, пока созданный файл не будет уничтожен, нарушитель будет пользоваться правами администратора.

Для предотвращения приведенной выше угрозы необходимо регулярно проверять файловую систему на наличие незарегистрированных файлов с установленными SUID или SGID:

```
# find / \(-perm -004000
-o -perm -002000 \) -type f -print
```

Если четвертый бит установлен в элементе вектора доступа для всех остальных пользователей (Sticky), то операционная система создает специальный текстовый образ программного файла. Чаще этот бит используется для каталогов и определяет запрет на удаление или переименование файлов других пользователей в этом каталоге. Это особенно важно для каталогов /tmp и /usr/tmp, чтобы одни пользователи не могли повредить работе других. Бит Sticky для каталогов может быть установлен только администратором.

В целях безопасности информации в КС биты SUID, SGID и Sticky для обычных файлов, не являющихся каталогами, сбрасываются при их изменении даже владельцем файла.

Права доступа к вновь создаваемым объектам определяются в Unix-системах на основе значения системной переменной umask, которое устанавливается в файлах пользователей .login, .cshrc или .profile либо в системном файле /etc/profile. Значение umask определяет маскируемые биты в элементах вектора доступа к создаваемому объекту (значение umask определяет биты, которые не будут установлены в векторе доступа).

Например, чтобы у всех вновь создаваемых объектов вектор доступа был равен 0667 (владелец объекта имел бы полный доступ к нему, а члены группы владельца и все остальные пользователи — право на чтение и выполнение объекта), значением переменной umask должно быть 0022. Для того чтобы только владелец создаваемого объекта имел к нему полный доступ, а всем остальным пользователям доступ к объекту был запрещен, следует установить umask в значение 0077.

Несанкционированное изменение субъектом значения переменной umask может привести к трудно предсказуемым последствиям для безопасности информации в КС.

Для ограничения полномочий пользователя в системе администратор может создать в файле паролей учетную запись с правом выполнения единственной системной команды или программы (не командного процессора), например:

```
date::60000:100:Запуск программы date:/tmp:/sbin/date
uptime::60001:100:Запуск программы uptime:/tmp:/usr/
ucb/uptime
finger::60002:100:Запуск программы finger:/tmp:/usr/
ucb/finger
sync::60003:100:Запуск программы sync:/tmp:/sbin/sync
```

Важным условием при этом является то, что разрешенная для пользователя программа не предоставляет ему возможности ввода с клавиатуры и не позволяет пользователю начать работу в системе в интерактивном режиме. Например, команда mail позволит пользователю выполнить любую команду:

```
% mail Sarah
Subject: test message
~!date
Wed Aug 1 09:56:42 EDT 2004
```

Для ограничения полномочий пользователей в КС могут также применяться ограниченные оболочки (restricted shell). Например, в System V UNIX стандартная оболочка sh может быть загружена в ограниченном режиме с указанием параметра *–r* в командной строке или связыванием файла оболочки с именем rsh. После запуска ограниченной оболочки выполняются команды из файла $HOME/.profile, после чего пользователь не может:

- изменять текущий каталог;
- изменять значение системной переменной PATH;
- применять команды, содержащие '/';
- перенаправлять вывод программ, применяя операции '>' или '>>'.

В качестве дополнительного ограничения пользователь не может прервать работу ограниченной оболочки во время обработки ею файла $HOME/.profile — в этом случае работа командного процессора немедленно завершается.

Возможна дополнительная модификация файла $HOME/.profile для усиления наложенных на права пользователя ограничений — добавление команд trap, exec и exit (обеспечение работы только с заданной программой) или команды tty (подтверждение работы в интерактивном режиме).

В версиях Unix, родственных Berkeley, для запуска ограниченной оболочки необходимо установить связь между файлом оболочки и файлом rsh, например:

```
% ln /bin/sh /usr/etc/rsh
```

В этом случае файл rsh не должен помещаться в стандартный каталог системных программ, чтобы ограниченная оболочка не запускалась вместо команды удаленного доступа rsh (remote shell).

Оболочка Korn может быть запущена в ограниченном режиме указанием параметра *-r* в командной строке или организацией связи ее файла ksh с файлом rksh или krsh. В этом случае на пользователя накладываются дополнительные, по сравнению с System V UNIX, ограничения — он не может изменять значения переменных среды ENV и SHELL, а также не может изменять свою первичную группу с помощью команды newgrp.

Свободно распространяемая оболочка bsh (bash) не поддерживает ограниченного режима работы.

Ограничение полномочий некоторой учетной записи производится в следующем порядке:

1) создание в файле паролей учетной записи пользователя с ограниченной оболочкой, например:

```
player::100:100:Пользователь-гость:/usr/rshhome:/bin/
rsh
```

2) создание для пользователя специального каталога только с разрешенными для него программами, например

```
# mkdir /usr/rshhome /usr/rshhome/bin
# ln /usr/games/rogue /usr/rshhome/bin/rogue
# ln /usr/bin/talk /usr/rshhome/bin/talk
# chmod 555 /usr/rshhome/bin
# chmod 555 /usr/rshhome
```

3) создание файла .profile в домашнем каталоге пользователя, например:

```
# cat > /usr/rshhome/.profile
/bin/echo Это учетная запись гостя.
/bin/echo Вы можете запускать программы:
/bin/echo rogue (игра),
/bin/echo talk (общение с другими пользователями).
/bin/echo
/bin/echo Для выхода введите logout.
PATH=/usr/rshhome/bin
SHELL=/bin/rsh
export PATH SHELL
^D
# chmod 444 /usr/rshhome/.profile
# chown player /usr/rshhome/.profile
# chmod 500 /usr/rshhome
```

Использование ограниченных оболочек требует внимательного выбора разрешенных для запуска пользователями программ. Многие системные команды и прикладные программы разрешают запуск других системных команд. Например, если разрешено выполнение команды man для чтения документации, то пользователь сможет запустить текстовый редактор, затем загрузить оболочку и выполнять любые программы.

Другим способом ограничения полномочий отдельных пользователей является применение программы chroot, которая изменяет представление файловой системы для вызвавшего эту программу процесса. В этом случае в сеансе работы пользователя доступна только часть файловой системы. В версии System V Release 4 (SVR4) дан-

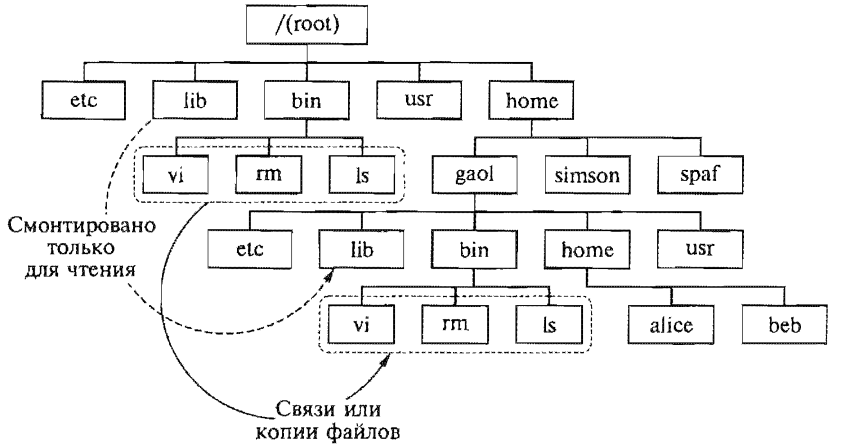

Рис. 3.26. Пример ограниченной файловой системы

ный вариант защиты может быть реализован автоматически, если в поле оболочки учетной записи в файле паролей вместо пути к файлу указывается '*'. Ограниченная файловая система, корнем которой будет теперь являться домашний каталог пользователя, должна иметь все необходимые файлы и команды для программы входа и работы прикладных программ пользователя. Естественно, что в новых файлах паролей и групп не должны содержаться те же самые пароли, что и в настоящих регистрационных файлах.

Точно так же каталоги /etc, /lib и /usr/lib, /bin в ограниченной файловой системе не должны содержать те же файлы, которые содержатся в стандартных каталогах. В них должны быть копии или связи только необходимых для работы данного пользователя файлов. При этом символические связи будут ограничены доступной частью файловой системы. Можно выполнить подключение файловой системы в режиме «только чтение» (с помощью команды mount) для обхода этого ограничения (рис. 3.26).

Системные журналы в Unix-системах сохраняются в каталогах /usr/adm (старые версии Unix), /var/adm (более современные версии Unix) или /var/log (некоторые версии Solaris, Linux и др.). К файлам системных журналов относятся следующие файлы:

- acct — регистрация команд, выполненных каждым пользователем;
- loginlog — регистрация неудачных попыток входа в КС;
- sulog — регистрация использования команды su;
- wtmp — регистрация входов и выходов пользователей, загрузки и завершения работы операционной системы;
- vold.log — регистрация ошибок, вызванных внешними устройствами (дисководами на гибких дисках, дисководами на компакт-дисках) и др.

Для регулярного архивирования и сохранения файлов системных журналов могут применяться командные сценарии, например:

```
#!/bin/ksh
BFILE=$(date+backup.%y.%m.%d.tar.Z)
cd /var/adm
tar cf -. | compress >../adm.backups/$BFILE
exit 0
```

В этом примере системные журналы из каталога /var/adm архивируются и сохраняются в каталоге /var/adm.backups ежедневно.

Многие Unix-системы имеют средства централизованного сбора информации о событиях (сообщениях) безопасности (сервис syslog). Каждое сообщение аудита в этом случае содержит следующую информацию:

- имя программы, при выполнении которой было сгенерировано сообщение;
- источник сообщения (модуль операционной системы);
- приоритет (важность) сообщения;
- содержание сообщения.

Пример сообщения о попытке входа суперпользователя с незарегистрированного терминала (источник сообщения — модуль авторизации и его приоритет — критическая ошибка не показаны):

```
login: Root LOGIN REFUSED on ttya
```

В конфигурационном файле /etc/syslog.conf определяются значения параметров политики аудита, например:

```
*.err;kern.debug;auth.notice      /dev/console
daemon,auth.notice                /var/adm/messages
lpr.*                             /var/adm/lpd-errs
auth.*                            root,nosmis
auth.*                            @prep.ai.mit.edu
*.emerg                           *
*.alert                           |dectalker
mark.*                            /dev/console
```

Каждая строка конфигурационного файла состоит из двух частей, разделяемых символом табуляции:

- селектора, в котором указываются приоритеты и источники регистрируемых сообщений (например, все сообщения об ошибках или все отладочные сообщения ядра системы);
- описания действия, которое должно быть выполнено при поступлении сообщения указанного типа (например, записать в системный журнал или отослать на терминал пользователя).

Запись о выбранном сообщении состоит также из двух частей: источника регистрируемого сообщения и его приоритета (напри-

мер, kern.debug указывает на все отладочные сообщения, которые генерируются ядром системы). Будут также регистрироваться все сообщения с приоритетом, большим чем у отладочных. Задание '*' вместо источника или приоритета означает «все». Два и более селектора могут быть объединены и разделяться точкой с запятой.

В поле действия могут быть указаны следующие реакции на поступившее сообщение:

- запись в файл аудита или вывод на устройство (описание действия начинается с '|');
- отправка сообщения пользователю (описание действия содержит его имя или список имен);
- отправка сообщения всем пользователям (в описание действия помещается '*');
- передача сообщения программе (описание действия начинается с '|');
- отправка сообщения сервису syslog на другой хост (в этом случае описание действия содержит имя хоста, начинающееся с '@').

Для просмотра файлов системных журналов может применяться программа Swatch, работа которой также управляется конфигурационным файлом.

В заключение приведем список событий безопасности, которые могут регистрироваться в версии Unix SCO Release 5.0:

- загрузка и завершение работы системы;
- вход и выход из системы;
- создание и завершение процесса;
- открытие, создание, закрытие, изменение и удаление объекта;
- выполнение программы;
- изменение прав доступа к объекту;
- отказ в доступе к объекту;
- действия привилегированных пользователей;
- превышение полномочий процессами;
- отказ в предоставлении ресурса;
- посылка сигналов и отправка сообщений процессам;
- изменение процесса (например, его системного идентификатора);
- изменение в политике аудита;
- изменение данных безопасности системы;
- использование защищенных подсистем;
- использование привилегий субъектами.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. В чем достоинства и недостатки использования пароля программы BIOS Setup?
2. Что защищается паролем пользователя в открытых версиях операционной системы Windows?

3. Как сохраняются пароли пользователей в открытых версиях операционной системы Windows?

4. Для чего предназначен редактор системных правил Windows? Где сохраняются результаты его работы?

5. Почему операционные системы Windows 9x/ME/XP Home Edition не могут считаться защищенными?

6. В чем сущность, достоинства и недостатки дискреционного управления доступом к объектам КС?

7. Какие правила применяются при предоставлении доступа субъекта к объекту в соответствии с мандатным управлением доступом к объектам КС? В чем их смысл?

8. Какие основные компоненты входят в состав подсистемы безопасности защищенных версий операционной системы Windows?

9. Где хранится регистрационная база данных пользователей защищенных версий Windows?

10. Как организовано хранение паролей в защищенных версиях Windows?

11. Какое дополнительное средство защиты паролей пользователей может быть применено в защищенных версиях Windows?

12. Какие параметры политики паролей и как могут быть определены в защищенных версиях Windows?

13. Что такое маркер доступа субъекта защищенных версий Windows и какая информация в нем содержится?

14. Как осуществляется разграничение доступа к объектам в защищенных версиях Windows?

15. Из чего состоит дескриптор безопасности объекта в защищенных версиях Windows? При каком обязательном условии файлы и папки могут иметь дескрипторы безопасности?

16. Как происходит изменение прав доступа к объектам защищенных версий Windows и кто это может делать?

17. Что относится к параметрам политики аудита в защищенных версиях Windows?

18. Где хранится информация об учетных записях пользователей и групп в операционных системах семейства Unix?

19. Как сохраняются пароли пользователей в операционных системах семейства Unix? Что такое теневые пароли?

20. Как разграничивается доступ субъектов к объектам в операционных системах семейства Unix?

21. Какую роль выполняют дополнительные биты SUID в векторах доступа к объектам операционных систем семейства Unix?

22. Как назначаются права доступа к вновь создаваемым объектам в защищенных версиях Windows и операционных системах семейства Unix?

23. Какая модель управления доступом к объектам применяется в защищенных версиях Windows и операционных системах семейства Unix?

24. Для чего предназначены защищенные оболочки в операционных системах семейства Unix?

25. Как устанавливаются параметры аудита в операционных системах семейства Unix?

ГЛАВА 4

КРИПТОГРАФИЧЕСКИЕ МЕТОДЫ И СРЕДСТВА ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

4.1. Элементы теории чисел

Целые числа a и b *сравнимы по модулю* n (целому числу, неравному нулю), если выполняется условие

$$a = b + kn$$

для некоторого целого числа k. В этом случае обычно используется следующая запись:

$$a = b \ \{\text{mod } n\}.$$

Сравнимость a и b по модулю n означает, что n делит $(a - b)$ нацело:

$$n \mid (a - b).$$

Если $b \geq 0$, $a = b \ \{\text{mod } n\}$ и $|b| < n$, то b называют *вычетом* числа a по модулю n. Вычет равен остатку от целочисленного деления числа a на число n. Операцию нахождения вычета числа a по модулю n называют *приведением* числа a по модулю n.

Очевидно, что

$$-a \ \{\text{mod } n\} = -\text{a} + n \ \{\text{mod } n\};$$
$$n = 0 \ \{\text{mod } n\}.$$

Примеры:

$$3 + 10 \ \{\text{mod } 12\} = 1 \ \{\text{mod } 12\} \text{ («арифметика» часов);}$$
$$-5 \ \{\text{mod } 7\} = 2 \ \{\text{mod } 7\}.$$

Полным набором вычетов по модулю n называется множество целых чисел от нуля до $n - 1$:

$$\{0, 1, 2, \ldots, n - 1\}.$$

Вычеты по модулю n с применением операций сложения и умножения образуют *коммутативное кольцо*, в котором справедливы законы ассоциативности, коммутативности и дистрибутивности. Операции над вычетами обладают также свойствами:

- аддитивности

$$(a + b) \ \{\text{mod } n\} = (a \ \{\text{mod } n\} + b \ \{\text{mod } n\}) \ \{\text{mod } n\};$$

• мультипликативности

$$(ab)\ \{\mathrm{mod}\ n\} = (a\ \{\mathrm{mod}\ n\}b\ \{\mathrm{mod}\ n\})\ \{\mathrm{mod}\ n\};$$

• сохранения степени

$$a^b\ \{\mathrm{mod}\ n\} = (a\ \{\mathrm{mod}\ n\})^b\ \{\mathrm{mod}\ n\}.$$

Данные свойства операций над вычетами позволяют либо сначала вычислять вычеты, а затем выполнять операцию, либо сначала выполнять операцию, а затем вычислять вычеты. Операция вычисления вычета является *гомоморфным отображением* кольца целых чисел в кольцо вычетов по модулю n.

Наибольшим общим делителем (НОД) целых чисел a и b называется наибольшее целое число, на которое делятся без остатка a и b. Например: НОД(56, 98) = 14 и НОД(150, 19) = 1.

Простым числом называется целое число, которое делится без остатка только на единицу и на себя. Например: 7, 13, 139.

Целые числа a и b называются *взаимно простыми*, если выполняется условие НОД(a, b) = 1.

Целое число y называется *мультипликативно обратным* целому числу x по модулю n, если выполняется условие $xy\ \{\mathrm{mod}\ n\} = 1$. Мультипликативно обратное целое число существует только тогда, когда x и n — взаимно простые числа. Если целые числа a и n не являются взаимно простыми, то сравнение $a^{-1} = x\ \{\mathrm{mod}\ n\}$ не имеет решения. Если из полного набора вычетов по модулю n выделить подмножество вычетов, взаимно простых с n, то получим *приведенный набор вычетов*. Например:

• {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10} — полный набор вычетов по модулю 11. Приведенным набором вычетов будет то же подмножество целых чисел, за исключением нуля;

• {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} — полный набор вычетов по модулю 10. Приведенным набором вычетов будет подмножество целых чисел {1, 3, 7, 9}.

Очевидно, что если n является простым числом, то приведенный набор вычетов по модулю n всегда содержит $(n - 1)$ элемент (все целые числа от единицы до $n - 1$).

Значением *функции Эйлера* $\varphi(n)$ будет число элементов в приведенном наборе вычетов по модулю n. Если n — простое число, то $\varphi(n) = n - 1$ и $\varphi(n^2) = n(n - 1)$. Если $n = pq$ (p и q — простые числа и $p \neq q$), то $\varphi(n) = (p - 1)(q - 1)$.

В соответствии с *малой теоремой Ферма*, если a — целое число, n — простое число и НОД(a, n) = 1, то

$$a^{n-1} = 1\ \{\mathrm{mod}\ n\}.$$

Обобщением малой теоремы Ферма является теорема Эйлера: если целые числа a и n являются взаимно простыми (НОД(a, n) = 1), то

$$a^{\varphi(n)} = 1\ \{\mathrm{mod}\ n\}.$$

4.2. Основные понятия криптологии. Симметричные и асимметричные криптосистемы

Назовем *открытым текстом* (plaintext) информацию, содержание которой может быть понятно любому субъекту. Под *шифрованием* (см. подразд. 1.1) понимается процесс преобразования открытого текста в шифротекст (cipher text) или криптограмму с целью сделать его содержание непонятным для посторонних лиц:

$$C = E_k(P),$$

где C — шифротекст; E — функция шифрования; k — ключ шифрования (дополнительный параметр функции шифрования); P — открытый текст.

Очевидно, что без введения ключа шифрования применение одной и той же функции шифрования к одному и тому же открытому тексту приводило бы всегда к получению одного и того же шифротекста. В этом случае защищенность шифротекста целиком и полностью определялась бы неизвестностью для посторонних функции шифрования, что практически невозможно обеспечить (особенно при современном уровне развития информационных технологий).

Под *расшифрованием* понимается процесс обратного преобразования шифротекста в открытый текст:

$$P = D_{k'}(C),$$

где D — функция расшифрования; k' — ключ расшифрования (дополнительный параметр функции расшифрования).

Совокупность реализуемых функциями E и D алгоритмов, множества возможных ключей, множеств возможных открытых текстов и шифротекстов принято называть *криптосистемой*. Если при шифровании и расшифровании используются одни и те же ключи ($k = k'$), то такую криптосистему называют *симметричной*. Очевидно, что ключ шифрования (он же — ключ расшифрования) в этом случае должен быть секретным.

Если при шифровании и расшифровании используются различные ключи, то такую криптосистему называют *асимметричной*. В этом случае один из этих ключей должен оставаться секретным (secret key), а другой может быть открытым (public key). Поэтому асимметричные криптосистемы иногда называют *криптосистемами с открытым ключом*

Науку о защите информации с помощью шифрования называют *криптографией* (криптография в переводе означает загадочное письмо или тайнопись). Криптография известна из глубокой древности, а одним из первых ее методов следует, по-видимому, считать создание письменности.

Процесс получения открытого текста из шифротекста без знания ключа расшифрования называют обычно *дешифрованием* (или взломом шифра), а науку о методах дешифрования — *криптоанализом*. Совместным изучением методов криптографии и криптоанализа занимается *криптология*.

Характеристика надежности шифротекста от вскрытия называется *криптостойкостью*. Криптостойкость шифра может оцениваться двумя величинами:

- минимальным объемом шифротекста, статистическим анализом которого можно его вскрыть и получить открытый текст без знания ключа;
- числом MIPS-часов или MIPS-лет — временем работы условного криптаналитического компьютера производительностью один миллион операций в секунду, необходимым для вскрытия шифротекста.

Очевидным правилом при выборе криптосистемы для защиты конфиденциальной информации в КС должно быть следующее: ценность конфиденциальной информации должна быть ниже стоимости вскрытия ее шифротекста нарушителем.

При разработке криптографических алгоритмов широко применяются вычисления в кольце вычетов по модулю (см. подразд. 4.1). Это вызвано следующими причинами:

- выполнение обратных вычислений (логарифмирования, извлечения корня, разложения на сомножители) гораздо более трудоемко, чем прямые вычисления (возведение в степень или умножение), что соответствует требованию существенно большей трудоемкости дешифрования без знания ключа по сравнению с расшифрованием по известному ключу;
- при вычислениях по модулю ограничивается диапазон возможных значений для всех промежуточных величин и результатов (например, $a^{25}\{\mathrm{mod}\ n\} = ((((a^2 a)^2)^2)^2) a\{\mathrm{mod}\ n\}$).

Криптография применяется:

- при защите конфиденциальности информации, передаваемой по открытым каналам связи;
- аутентификации (подтверждении подлинности) передаваемой информации;
- защите конфиденциальной информации при ее хранении на открытых носителях;
- обеспечении целостности информации (защите информации от внесения несанкционированных изменений) при ее передаче по открытым каналам связи или хранении на открытых носителях;
- обеспечении неоспоримости передаваемой по сети информации (предотвращении возможного отрицания факта отправки сообщения);
- защите программного обеспечения и других информационных ресурсов от несанкционированного использования и копирования.

В подразд. 1.1 было приведено определение *хеширования* — процесса преобразования исходного текста M произвольной длины в хеш-значение (дайджест или образ) H(M) фиксированной длины. К функциям хеширования предъявляются следующие требования:

• постоянство длины хеш-значения независимо от длины исходного текста:

$\forall$ M Length[H(M)] = const;

• полная определенность (для двух одинаковых исходных текстов должно получаться одно и то же хеш-значение):

$\forall$ $M_1 = M_2 H(M_1) = H(M_2)$;

• необратимость (невозможность восстановления исходного текста по его хеш-значению):

$\neg\exists H^{-1}(H(M)) = M$;

• стойкость к взлому (практическая невозможность подбора другого исходного текста для известного хеш-значения):

$\neg\exists M' \neq M$ $H(M') = H(M)$.

К основным применениям хеширования при обеспечении информационной безопасности КС относятся:

• защита парольной и иной идентифицирующей пользователей КС информации (см. подразд. 2.1);

• создание дайджеста файла или электронного сообщения, применяемого в системах электронной цифровой подписи (см. подразд. 4.7).

4.3. Способы создания симметричных криптосистем. Абсолютно стойкий шифр

К основным способам симметричного шифрования относятся перестановки, подстановки и гаммирование. При использовании *перестановки* биты (или символы) открытого текста переставляются в соответствии с задаваемым ключом шифрования правилом

$$\forall\ i,\ 1 \le i \le n\ C_i = P_{k[i]},$$

где $P = \{P_1, P_2, ..., P_i, ..., P_n\}$ — открытый текст; n — длина открытого текста; $C = \{C_1, C_2, ..., C_i, ..., C_n\}$ — шифротекст; $k = \{k_1, k_2, ..., k_i, ..., k_n\}$ — ключ шифрования.

При расшифровании применяется обратная перестановка:

$$\forall\ i,\ 1 \le i \le n\ P_{k[i]} = C_i.$$

Очевидно, что при шифровании перестановкой ключ должен удовлетворять условию

$$\forall\ k_i \in k\ 1 \le k_i \le n \wedge \forall\ k_i, k_j \in k\ k_i \ne k_j.$$

Пример. Пусть надо зашифровать слово «связной» (n = 7) с помощью ключа k = {4, 2, 1, 7, 6, 3, 5}. В результате шифрования мы получаем шифротекст «звсйоян».

Если длина ключа меньше длины открытого текста, то можно разбить открытый текст на блоки, длина которых равна длине ключа, и последовательно применить ключ перестановки к каждому блоку открытого текста. Если длина открытого текста не кратна длине ключа, то последний блок может быть дополнен пробелами или нулями.

Можно использовать и другой прием. После разбиения открытого текста длиной n на блоки, длина которых равна длине ключа m, открытый текст записывается в таблицу с числом столбцов, равным длине ключа (каждый блок открытого текста записывается в столбец таблицы). Число строк таблицы в этом случае будет равно наименьшему целому числу, не меньшему n/m. Затем столбцы полученной таблицы переставляются в соответствии с ключом перестановки, а шифротекст считывается из строк таблицы последовательно.

Пример. Необходимо зашифровать открытый текст: «связной прилетает в пятницу» (n = 27) — с помощью ключа перестановки k = {3, 5, 4, 2, 1} (m = 5). После разбиения открытого текста на блоки и занесения его в таблицу размером 6 строк и 5 столбцов получаем:

с	й	е	в	и
в		т		ц
я	п	а	п	у
з	р	е	я	
н	и	т	т	
о	л		н	

После применения ключа перестановки к столбцам таблицы получаем:

е	и	в	й	с
т	ц			в
а	у	п	п	я
е		я	р	з
т		т	и	н
		н	л	о

После считывания текста по строкам таблицы получаем окончательный шифротекст: «еивйстц вауппяе ярзт тин нло».

При расшифровании шифротекст записывается в таблицу того же размера по строкам, затем происходит обратная перестановка столбцов в соответствии с ключом, после чего расшифрованный текст считывается из таблицы по столбцам.

Если в качестве ключа перестановки используется последовательность не цифр, а произвольных символов (например, пароль пользователя КС), то его необходимо предварительно преобразовать в последовательность целых чисел от единицы до m (m — длина ключа):

1) символы ключа сортируются в лексикографическом порядке;

2) каждый символ исходного ключа заменяется целым числом, равным номеру его позиции в отсортированном ключе.

Приведем пример функций на языке С, выполняющих шифрование и расшифрование перестановкой заданной строки символов максимальной длины MAXLEN (в этом примере открытый текст не разбивается на блоки):

```
#include <stdlib.h>
#include <string.h>
...
// функция, используемая при сортировке символов ключа
int lt(const void *a, const void *b)
{ if(*(char*)a<*(char*)b) return -1;
else if(*(char*)a==*(char*)b) return 0;
else return 1; }
/* функция шифрования открытого текста str длиной n по
ключу key с записью полученного шифротекста в res */
void Crypt(const char *str, const char *key, char *res,
unsigned n)
{char tmp[MAXLEN]; // отсортированный ключ
/* выравнивание длин открытого текста и ключа: усечение
или дополнение пробелами ключа */
if(n<strlen(key)) strncpy(tmp, key, n);
else strcpy(tmp, key);
while(strlen(tmp)<n)
strncat(tmp, " ", 1);
char Key[MAXLEN]; // преобразованный ключ
strcpy(Key, tmp);
// сортировка символов ключа
qsort(tmp, n, sizeof(char), lt);
unsigned Perm[MAXLEN]; // ключ перестановки
// цикл шифрования
for(unsigned i=0; i<n; i++)
{// получение очередного элемента ключа перестановки
Perm[i]=strchr(tmp, Key[i])-tmp;
```

```
// «сброс» уже найденного символа отсортированного ключа
tmp[Perm[i]]='\1';
// получение очередного символа шифротекста
res[i]=str[Perm[i]]; }
res[n]='\0'; }
/* функция расшифрования шифротекста str длиной n по
ключу key с записью восстановленного открытого текста
в res */
void Encrypt(const char *str, const char *key, char
*res, unsigned n)
{char tmp[MAXLEN];
if(n<strlen(key)) strncpy(tmp, key, n);
else strcpy(tmp, key);
while(strlen(tmp)<n)
strncat(tmp," ",1);
char Key[MAXLEN];
strcpy(Key, tmp);
qsort(tmp, n, sizeof(char), lt);
unsigned Perm[MAXLEN];
// цикл расшифрования
for(unsigned i=0; i<n; i++)
{ Perm[i]=strchr(tmp, Key[i])-tmp;
tmp[Perm[i]]='\1';
// восстановление очередного символа открытого текста
res[Perm[i]]=str[i]; }
res[n]='\0'; }
```

Достоинством шифрования перестановкой является высокая скорость получения шифротекста. При шифровании двоичных файлов (например, программных) можно дополнительно сократить временные затраты, если ограничиться перестановками только наиболее важных участков шифруемого файла.

К недостаткам шифрования перестановкой относятся:

- сохранение частотных характеристик открытого текста после его шифрования (символы открытого текста лишь меняют свои позиции в шифротексте);
- малое число возможных ключей шифрования.

При шифровании с помощью *подстановки* (замены) символы открытого текста заменяются символами того же (*одноалфавитная подстановка*) или другого (*многоалфавитная подстановка*) алфавита в соответствии с определяемым ключом шифрования правилом.

При использовании одноалфавитной подстановки каждый символ открытого текста заменяется в шифротексте символом, номер которого в используемом алфавите больше номера символа открытого текста на величину ключа шифрования (используется

сложение в кольце вычетов по модулю, равному мощности применяемого алфавита):

$$\forall\ i,\ 1 \le i \le n\ \ C_i = P_i + k\ \{\mathrm{mod}\ m\},$$

где $P = \{P_1, P_2, ..., P_i, ..., P_n\}$ — открытый текст; n — длина открытого текста; $A = \{A_1, A_2, ..., A_m\}$ — алфавит символов открытого текста ($\forall\ i,\ 1 \le i \le n\ P_i \in A$); $C = \{C_1, C_2, ..., C_i, ..., C_n\}$ — шифротекст; k — ключ шифрования ($0 \le k < m$); $\forall a_i \in A,\ 1 \le i \le m\ a_i + k = a_{i+k}$.

При расшифровании символ шифротекста заменяется символом, номер которого в используемом алфавите больше номера символа шифротекста (применяется операция сложения в кольце вычетов по модулю m) на величину $m - k$ (m — мощность используемого алфавита; k — ключ шифрования):

$$\forall\ i,\ 1 \le i \le n\ C_i = P_i + m - k\ \{\mathrm{mod}\ m\}.$$

Пример. При шифровании открытого текста «наступайте» с помощью одноалфавитной подстановки по ключу 3 (так называемой подстановки Цезаря) получаем шифротекст «ргфхцгтмхз».

Приведем пример функций на языке С, выполняющих шифрование и расшифрование одноалфавитной подстановкой заданной строки символов.

```
/* функция шифрования открытого текста str длиной n по
ключу key с записью полученного шифротекста в res */
void Crypt(const char *str, char key, char *res, unsigned n)
{ for(unsigned i=0; i<n; i++)
res[i]=(str[i]+key) % 256;
res[n]='\0'; }
/* функция расшифрования шифротекста str длиной n по
ключу key с записью восстановленного открытого текста
в res */
void Encrypt(const char *str, char key, char *res,
unsigned n)
{ for(unsigned i=0; i<n; i++)
res[i]=(str[i]+256-key) % 256;
res[n]='\0'; }
```

Основные недостатки шифрования с помощью одноалфавитной подстановки:

- не скрывается частота появления различных символов открытого текста в шифротексте (одинаковые символы открытого текста остаются одинаковыми и в шифротексте);
- малое число возможных ключей.

При использовании многоалфавитной подстановки каждый символ открытого текста заменяется в шифротексте символом, номер которого в применяемом алфавите больше номера символа открытого текста на величину, равную очередному элементу ключа

шифрования (используется сложение в кольце вычетов по модулю, равному мощности алфавита):

$$\forall\ i,\ 1 \le i \le n\ C_i = P_i + k_i \ \{\mathrm{mod}\ m\},$$

где $P = \{P_1, P_2, ..., P_i, ..., P_n\}$ — открытый текст; n — длина открытого текста; $A = \{A_1, A_2, ..., A_m\}$ — алфавит символов открытого текста ($\forall\ i,\ 1 \le i \le n\ P_i \in A$); $C = \{C_1, C_2, ..., C_i, ..., C_n\}$ — шифротекст; $k = \{k_1, k_2, ..., k_i, ..., k_n\}$ — ключ шифрования ($\forall\ i,\ 1 \le i \le n\ 0 \le k_i < m$); $\forall a_i \in A,\ 1 \le i \le m\ a_i + k = a_{i+k}$.

Расшифрование выполняется аналогично шифрованию, за исключением того, что увеличение номера в алфавите, соответствующего символу шифротекста, происходит на величину $m - k_i$ (m — мощность алфавита; k_i — очередной элемент ключа):

$$\forall\ i,\ 1 \le i \le n\ \ C_i = P_i + m - k_i \ \{\mathrm{mod}\ m\}.$$

Если длина ключа меньше длины открытого текста, то необходимо разбить открытый текст на блоки, длина которых равна длине ключа, и последовательно применить ключ подстановки к каждому блоку открытого текста. Если длина открытого текста не кратна длине ключа, то последний блок следует дополнить необходимым числом одних и тех же символов.

Приведем пример функций на языке С, выполняющих шифрование и расшифрование многоалфавитной подстановкой заданной строки символов (в этом примере открытый текст не разбивается на блоки).

```
#include <string.h>
...
/* функция шифрования открытого текста str длиной n по
ключу key с записью полученного шифротекста в res */
void Crypt(const char *str, const char *key, char *res,
unsigned n)
{char tmp[MAXLEN]; // ключ скорректированной длины
/* выравнивание длин открытого текста и ключа: усечение
или дополнение пробелами ключа */
if(n<strlen(key)) strncpy(tmp, key, n);
else strcpy(tmp, key);
while(strlen(tmp)<n)
strncat(tmp, " ", 1);
// цикл шифрования
for(unsigned i=0; i<n; i++)
res[i]=(str[i]+tmp[i]) % 256;
res[n]='\0'; }
/* функция расшифрования шифротекста str длиной n по
ключу key с записью восстановленного открытого текста
в res */
```

```
void Encrypt(const char *str, const char *key, char
*res, unsigned n)
{char tmp[MAXLEN];
if(n<strlen(key)) strncpy(tmp, key, n);
else strcpy(tmp, key);
while(strlen(tmp)<n)
strncat(tmp, " ", 1);
// цикл расшифрования
for(unsigned i=0; i<n; i++)
res[i]=(str[i]+256-tmp[i]) % 256;
res[n]='\0'; }
```

Достоинством многоалфавитной подстановки является то, что в шифротексте маскируется частота появления различных символов открытого текста, поэтому криптоаналитик не может при вскрытии шифра использовать частотный словарь букв естественного языка.

Разновидностью шифрования с применением многоалфавитной подстановки является побайтное шифрование, при котором каждый следующий байт открытого текста складывается с предыдущим байтом, а нулевой байт открытого текста — с последним байтом.

На рис. 4.1 и 4.2 приведены алгоритмы побайтного шифрования и расшифрования открытого текста $P=\{P_0, P_1, ..., P_i, ..., P_n\}$. При модификации алгоритма побайтного шифрования к очеред-

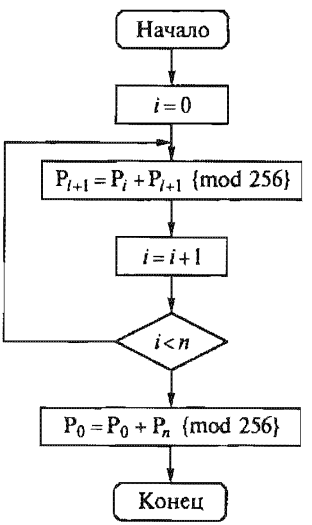

Рис. 4.1. Алгоритм побайтного шифрования

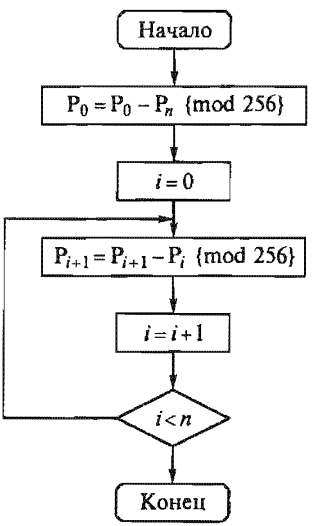

Рис. 4.2. Алгоритм побайтного расшифрования

ному байту открытого текста добавляется значение байта с определяемым ключом шифрования смещением. Недостаточная криптостойкость данного алгоритма шифрования при его использовании в чистом виде определяется тем, что ключ шифрования содержится в шифротексте.

При *гаммировании* шифротекст получается путем наложения на открытый текст гаммы шифра с помощью какой-либо обратимой операции (как правило, поразрядного сложения по модулю 2):

$$\forall\ i,\ 1 \le i \le n\ \ C_i = P_i \oplus G_i,$$

где $P = \{P_1, P_2, ..., P_i, ..., P_n\}$ — открытый текст; n — длина открытого текста; $C = \{C_1, C_2, ..., C_i, ..., C_n\}$ — шифротекст; $G = \{G_1, G_2, ..., G_i, ..., G_n\}$ — гамма шифра; $\oplus$ — операция поразрядного сложения по модулю 2.

Расшифрование в этом случае заключается в повторном наложении той же гаммы шифра на шифротекст:

$$\forall\ i,\ 1 \le i \le n\ \ P_i = C_i \oplus G_i.$$

Гамма шифра вычисляется с помощью программного или аппаратного датчика (генератора) псевдослучайных чисел, параметры которого определяются ключом шифрования. Одним из наиболее простых датчиков псевдослучайных чисел является линейный конгруэнтный датчик:

$$\forall\ i,\ 1 \le i \le n\ \ G_i = aG_{i-1} + c\ \{\mathrm{mod}\ m\},$$

где a, c и G_0 — определяемые ключом параметры датчика псевдослучайных чисел; $m = 2^s$ (s — длина машинного слова в битах, обычно 32 или 64).

Доказано, что максимальный период генерируемой линейным конгруэнтным датчиком псевдослучайной последовательности достигается при нечетном значении параметра c и $a = 1\ \{\mathrm{mod}\ 4\}$.

Другим примером датчика псевдослучайных чисел являются линейные последовательные машины (М-последовательности). Определяемое ключом шифрования исходное двоичное значение $a_0, a_1, ..., a_{k-1}$ помещается в сдвиговый регистр. На каждом такте работы датчика происходит сдвиг k двоичных разрядов, вытесняемый бит a_0 добавляется к гамме шифра, а затем замещает бит a_{k-1} по следующему правилу:

$$a_k = -\sum_{j=0}^{k-1} h_j a_j,$$

где $h = \{h_0, h_1, ..., h_{k-1}\}$ — определяемые ключом шифрования коэффициенты передачи ($\forall j,\ 0 \le j < k\ h_j = 0 \vee h_j = 1$); $\sum$ — операция суммирования по модулю 2.

Если в качестве коэффициентов передачи выбираются коэффициенты неприводимого многочлена степени k (который нельзя раз-

ложить на сомножители-многочлены степени меньше k), то данный тип генератора псевдослучайных чисел обеспечивает выдачу последовательности двоичных чисел с периодом, равным $2^k - 1$.

Приведем пример функций на языке С, выполняющих шифрование и расшифрование гаммированием заданной строки символов (используется линейный конгруэнтный датчик псевдослучайных чисел):

```
/* функция шифрования открытого текста str длиной n по
ключу a, c и g0 с записью полученного шифротекста в res
*/
void Crypt(const char *str, char a, char c, char g0,
char *res, unsigned n)
{char Gamma=g0; // гамма шифра
// цикл шифрования
for(unsigned i=0; i<n; i++)
{ Gamma=(a*Gamma+c) % 256;
res[i]=str[i]^Gamma; }
res[n]='\0'; }
/* функция расшифрования шифротекста str длиной n по
ключу a, c и g0 с записью восстановленного открытого
текста в res */
void Encrypt(const char *str, char a, char c, char g0,
char *res, unsigned n)
{ char Gamma=g0; // гамма шифра
// цикл расшифрования
for(unsigned i=0; i<n; i++)
{ Gamma=(a*Gamma+c) % 256;
res[i]=str[i]^Gamma; }
res[n]='\0'; }
```

Гаммирование лежит в основе *потоковых* шифров, в которых открытый текст преобразуется в шифротекст последовательно по одному биту. Криптостойкость потоковых шифров полностью определяется структурой используемого генератора псевдослучайной последовательности (чем меньше период псевдослучайной последовательности, тем ниже криптостойкость потокового шифра).

Основным преимуществом потоковых шифров является их высокая производительность. Эти шифры наиболее пригодны для шифрования непрерывных потоков открытых данных (например, в сетях передачи данных или связи). К наиболее известным современным потоковым шифрам относятся:

- RC4 (Rivest Cipher 4), разработанный Р. Ривестом (R. Rivest); в шифре RC4 может использоваться ключ переменной длины;
- SEAL (Software Encryption ALgorithm) — приспособленный для программной реализации потоковый шифр, использующий ключ длиной 160 бит;

• WAKE (Word Auto Key Encryption).

Важнейшим показателем качества построенного шифра является отсутствие каких-либо закономерностей в шифротексте (частотная и позиционная равномерность кодов символов в нем). Поэтому из-за недостаточной криптостойкости в настоящее время не используются шифры перестановок или подстановок в чистом виде.

Большинство современных симметричных криптосистем относятся к разряду *блочных* шифров. В этих криптосистемах открытый текст разбивается на блоки, как правило, фиксированной длины, к каждому блоку применяется функция шифрования, использующая перестановки битов блока и многократное повторение операций подстановки и гаммирования, после чего над зашифрованными блоками может выполняться дополнительная операция перед включением их в шифротекст.

К наиболее распространенным способам построения блочных шифров относится *сеть Фейстела*, при использовании которой каждый блок открытого текста представляется сцеплением двух полублоков одинакового размера $L_0 \| R_0$. Затем для каждой итерации (раунда) i выполняется следующие действия:

$$1)\ L_i = R_{i-1};$$

$$2)\ R_i = L_{i-1} \oplus f(R_{i-1}, k_i),$$

где f — функция шифрования; k_i — ключ, используемый на i-м раунде шифрования (k_i определяется исходным ключом шифрования открытого текста и называется внутренним ключом).

К основным характеристикам современных блочных шифров относятся длина блока, длина ключа шифрования и число раундов. В табл. 4.1 приведены значения данных характеристик для наиболее известных блочных шифров.

Может ли быть построен идеальный (абсолютно стойкий) шифр? Ответ на этот вопрос был дан К. Шенноном. Для того чтобы шифр обладал абсолютной стойкостью к взлому, он должен обладать двумя свойствами:

1) ключ шифрования должен вырабатываться совершенно случайным образом (в частности, один и то же ключ должен применяться для шифрования только одного открытого текста);

2) длина шифруемого открытого текста не должна превышать длину ключа шифрования.

К сожалению, в большинстве случаев обеспечить выполнение этих условий практически невозможно, хотя короткие и наиболее важные сообщения следует шифровать именно так. Для открытых текстов большой длины главной проблемой симметричной криптографии является генерация, хранение и распространение ключа шифрования достаточной длины.

Таблица 4.1

Шифр	Длина блока в битах	Число раундов	Длина ключа в битах
DES (DataEncryption Standard)	64	16	64 (8 контрольных)
3-DES (Triple-DES)	64	48	168
DESX (DES eXtended)	64	16	184
ГОСТ 28147—89	64	32	256
IDEA (International Data Encryption Algorithm)	64	8	128
AES (Advanced Encryption Standard)	128	14	128, 192, 256
RC2 (Rivest Cipher 2)	64	Переменное	Переменная
RC5 (Rivest Cipher 5)	32, 64, 128	Переменное	Переменная
RC6 (Rivest Cipher 6)	Переменная	Переменное	Переменная
CAST (C. Adams, S. Tavares)	64	16	128
Blowfish	64	16	Переменная
SAFER+	128	8, 12, 16	128, 192, 256
Skipjack	64	32	80

Очевидно, что за счет увеличения длины ключа шифрования можно уменьшить требования к сложности алгоритма блочного шифрования (например, уменьшить число раундов), и, наоборот, более короткий ключ требует увеличения сложности криптоалгоритма.

Генерация случайного ключа шифрования возможна с помощью программного или аппаратного датчика псевдослучайных чисел и случайных событий, создаваемых пользователем при нажатии клавиш на клавиатуре или движением мыши (рис. 4.3). Ключ шифрования будет в этом случае создан из порций, взятых из псевдослучайной последовательности в момент возникновения инициированных пользователем событий.

Для того чтобы на основе одного ключа шифрования сгенерировать несколько различных сеансовых ключей (session keys), которые будут использованы для шифрования отдельных сообщений (файлов), могут применяться добавляемые к ключу случайные значения (salt values). Особенно полезны случайные значения

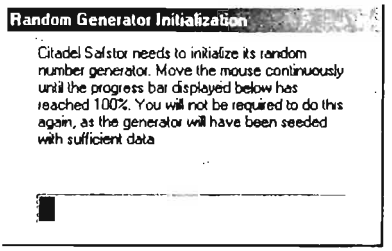

Рис. 4.3. Окно отображения хода генерации ключа шифрования

при шифровании большого числа практически идентичных пакетов данных — будет обеспечено получение совершенно разных шифротекстов. В отличие от ключей шифрования случайные значения могут быть открытыми и передаваться (храниться) вместе с шифротекстом.

Если для хранения ключей шифрования используются открытые магнитные носители, то эти ключи должны храниться только в зашифрованном с помощью мастер-ключа виде. Мастер-ключ не зашифровывается, но хранится в защищенной части аппаратуры КС, причем его потеря в результате аппаратной ошибки не должна приводить к потере зашифрованных с его помощью данных.

Для распределения ключей шифрования между субъектами распределенной КС могут применяться центры распределения ключей (Key Distribution Center, KDC). В этом случае на каждом объекте КС должен храниться ключ шифрования для связи с KDC. Недостатком применения центра распределения ключей является то, что в KDC возможно чтение всех передаваемых в КС сообщений. Для организации анонимного распределения ключей симметричного шифрования могут использоваться протоколы, основанные на криптографии с открытым ключом (см. подразд. 4.6).

Другой способ распределения ключей шифрования между субъектами КС состоит в прямом обмене данными между ними. Основной проблемой при этом является взаимное подтверждение подлинности субъектов сети. Для решения этой задачи могут применяться рассмотренные в подразд. 2.5 протоколы. В этом случае в настоящее время также активно используются методы асимметричной криптографии (например, криптосистема Диффи-Хеллмана, см. подразд. 4.6).

4.4. Криптографическая система DES и ее модификации

Алгоритм DES до 2001 г. являлся федеральным стандартом США на защиту информации, не относящейся к государственной тайне. Он был поддержан национальным институтом США по стандартам и технологиям (National Institute of Standards and

Technologies) и американской ассоциацией банкиров (American Bankers Association). Алгоритм DES допускает программную и аппаратную реализацию.

Рассмотрим примерную схему алгоритма DES. Пусть K — ключ шифрования (длина 64 бита, из которых 8 битов контрольных); IP — начальная перестановка битов в блоке открытого текста P длиной 64 бита; IP^{-1} — обратная к IP перестановка; L и R — соответственно левый и правый полублоки (длиной 32 бита) блока P; k_i — внутренний ключ шифрования i-го раунда длиной 48 бит (k_i = KS(i, K)); f — функция шифрования, на вход которой поступает блок длиной 32 бита, а на выходе формируется блок длиной также 32 бита.

Шифрование очередного блока открытого текста P по алгоритму DES производится следующим образом (C — сформированный блок шифротекста).

1. $L_0R_0 = IP(P)$.
2. Сеть Фейстела (см. подразд. 4.3) с числом раундов, равным 16.
3. $C = IP^{-1}(R_{16}L_{16})$.

Аргументами функции шифрования f являются R_{i-1} и k_i (i — номер раунда).

Алгоритм выполнения функции f.

1. Расширение R_{i-1} до 48 бит путем копирования 16 крайних элементов из 8 четырехбитных подблоков исходного R_{i-1} (R'_{i-1} — преобразованное значение R_{i-1}).
2. $R'_{i-1} = R_{i-1} \oplus k_i$.
3. Выполнение блока подстановки (S-бокса), в котором каждый из 8 шестибитных подблоков R'_{i-1} используется для выбора строки и столбца соответствующей номеру подблока матрицы элементов со значениями от 0 до 15 (размер матрицы — 4 строки и 16 столбцов). На выходе блока подстановки получаем текст длиной 32 бита.
4. Выполнение блока перестановки (P-бокса), иначе называемого блоком проволочной коммутации.

Алгоритм функции KS выбора внутреннего ключа k_i.

1. Начальная перестановка исходного ключа шифрования из 56 бит (K′ — преобразованный блок ключа).
2. $K' = C_0 \| D_0$ (длина полублоков C и D равна 28 битам).
3. Независимый циклический сдвиг налево C_0 и D_0 на зависящее от номера раунда i число разрядов (C_i и D_i — преобразованные полублоки).
4. Конечная перестановка $C_i \| D_i$ для получения 48-битного k_i.

При расшифровании блока C шифротекста по алгоритму DES все действия выполняются в обратном порядке, начиная с перестановки $IP^{-1}(R_{16}L_{16})$. Затем для i = 16, 15, ..., 2, 1 выполняются следующие действия:

1. $R_{i-1} = L_i$.

2. $L_{i-1} = R_i \oplus f(L_i, k_i)$.

Последним шагом алгоритма расшифрования будет применение перестановки IP к блоку L_0R_0, в результате которого будет восстановлен блок P открытого текста.

В криптосистеме на основе алгоритма DES используются четыре режима работы. В режиме *электронной кодовой книги* (Electronic Code Book, ECB) каждый блок открытого текста зашифровывается независимо от других блоков:

$$\forall\ i,\ 1 \le i \le n\ C_i = E_k(P_i),$$

где $P = \{P_1, P_2, ..., P_i, ..., P_n\}$ — открытый текст; n — число 64-битных блоков открытого текста; $C = \{C_1, C_2, .., C_i, ..., C_n\}$ — шифротекст; k — ключ шифрования.

Расшифрование в режиме ECB выполняется следующим образом:

$$\forall\ i,\ 1 \le i \le n\ P_i = D_k(C_i),$$

где D — функция расшифрования.

В режиме ECB одинаковые блоки открытого текста остаются одинаковыми и в шифротексте, что облегчает задачу криптоанализа шифротекста. Кроме того, если длина открытого текста не кратна 64 битам, то необходимо выполнить расширение (padding) последнего блока P_n до 8 байт. Обычно последний блок в этом случае дополняется байтами, значением каждого из которых является число добавляемых байт (например, если необходимо добавить три байта, то последний блок дополняется байтами со значением \x03). Но в любом случае криптоаналитику при этом проще вскрыть содержимое последнего блока шифротекста.

В режиме ECB изменение одного бита в шифротексте приведет к изменению всего блока в расшифрованном открытом тексте. Кроме того, нарушитель может свободно удалять, повторять или переставлять блоки шифротекста для воздействия на расшифрованный открытый текст.

Режим ECB целесообразно использовать для шифрования коротких открытых текстов (других ключей шифрования, паролей и т. п.), в которых не содержатся одинаковые блоки из восьми байт.

В режиме *сцепления блоков шифра* (Cipher Block Chaining, CBC) каждый блок открытого текста перед шифрованием складывается по модулю 2 с предыдущим блоком шифротекста, а первый блок — с вектором инициализации (синхропосылкой) IV (дополнительным параметром шифра, который должен сохраняться и передаваться вместе с ключом шифрования):

$$\forall\ i,\ 1 \le i \le n\ C_i = E_k(P_i \oplus C_{i-1}),\ C_0 = IV.$$

Расшифрование в режиме CBC выполняется следующим образом:

$$\forall\ i,\ 1 \le i \le n\ P_i = C_{i-1} \oplus D_k(C_i),\ C_0 = IV.$$

При использовании режима CBC одинаковые блоки открытого текста становятся различными в шифротексте. Последний блок шифротекста C_n является функцией ключа шифрования, вектора инициализации и всех блоков открытого текста — кодом аутентификации сообщения (Message Authentication Code, MAC). Блок MAC может использоваться для проверки подлинности и целостности полученного сообщения с помощью тех же значений ключа и вектора инициализации.

При изменении одного бита в шифротексте будет полностью искажен соответствующий блок восстановленного открытого текста, а также аналогичный бит предыдущего блока.

Режим *обратной связи по шифротексту* (Cipher FeedBack, CFB) использует регистр замены (сдвига), в который первоначально помещается вектор инициализации. После шифрования блока в регистре замены происходит его сдвиг влево на величину, равную длине порции данных (например, 1/4 длины регистра замены), и сложение по модулю 2 вытесняемой части регистра с очередной порцией открытого текста. Результат последней операции образует очередную порцию шифротекста и одновременно помещается в освободившуюся часть регистра сдвига:

$$\forall\ i,\ 1 \le i \le m\ C_i = P_i \oplus E_k(C_{i-1}),\ C_0 = IV$$

где m — число порций открытого текста.

Расшифрование в режиме CFB производится следующим образом:

$$\forall\ i,\ 1 \le i \le m\ P_i = C_i \oplus E_k(C_{i-1}),\ C_0 = IV.$$

В режиме CFB искажение одного бита шифротекста приведет к искажению последовательности расшифрованных блоков открытого текста. Поскольку последний блок шифротекста зависит от всех блоков открытого текста, а также от вектора инициализации и ключа шифрования, он также может использоваться в качестве кода аутентификации сообщения.

В режиме *обратной связи по выходу* (Output FeedBack, OFB) также используются регистр замены и вектор инициализации. После шифрования блока в регистре замены и сдвига вытесняемая часть замещает свободную область регистра и одновременно складывается по модулю 2 с очередной порцией открытого текста. Результат последней операции и образует очередную порцию шифротекста:

$$\forall\ i,\ 1 \le i \le m\ C_i = P_i \oplus S_i,\ S_i = E_k(S_{i-1}),\ S_0 = IV.$$

где m — число порций открытого текста.

Расшифрование в режиме OFB производится следующим образом:

$$\forall\ i,\ 1 \le i \le m\ \ P_i = C_i \oplus S_i,\ S_i = E_k(S_{i-1}),\ S_0 = IV.$$

Особенность режима OFB по сравнению с CFB заключается в том, что любые искажения внутри одного блока шифротекста не распространяются на следующие восстановленные блоки открытого текста. Режим OFB часто используется для генерации псевдослучайных чисел (см. подразд. 4.3), а также применяется в спутниковых системах связи.

Число возможных ключей шифрования в криптосистеме DES равно $2^{56} = 128^8 = 72\,057\,594\,037\,927\,936$. Если же при выборе ключа шифрования ограничиться лишь печатаемыми символами (например, взятыми из пароля пользователя), то число возможных ключей уменьшится до $96^8 = 7\,213\,895\,789\,838\,336$.

Для повышения криптостойкости алгоритма DES, вызванной недостаточными на сегодняшний день длиной ключа шифрования и числом раундов, используются различные модификации этой криптосистемы. Среди них наиболее известны 3-DES и DESX.

В тройном DES (3-DES) к одному и тому же блоку открытого текста P функция шифрования применяется трижды с тремя разными ключами ($k1$, $k2$ и $k3$), что обеспечивает увеличение длины ключа окончательного шифрования и числа раундов в три раза:

$$C = E_{k3}(D_{k2}(E_{k1}(P))).$$

Расшифрование выполняется следующим образом:

$$P = D_{k1}(E_{k2}(D_{k3}(C))).$$

На втором шаге тройного DES используется не функция шифрования, а функция расшифрования, поскольку при $k1 = k2 = k3$ результат шифрования по алгоритму 3-DES совпадает с шифрованием по алгоритму DES на ключе $k1$. Использование двойного DES более уязвимо для криптоанализа.

Недостатком алгоритма 3-DES является снижение производительности шифрования в три раза по сравнению с алгоритмом DES. Этого недостатка лишен алгоритм DESX:

$$C = k_2 \oplus E_k(k_1 \oplus P),$$

где k — ключ DES-шифрования длиной 56 бит; k_1 и k_2 — дополнительные ключи шифрования длиной 64 бита каждый.

Общая длина ключа шифрования, используемого в алгоритме DESX, составляет, таким образом, 184 бита. Расшифрование шифротекста по алгоритму DESX производится следующим образом:

$$P = D_k(C \oplus k_2) \oplus k_1.$$

Многие операционные системы семейства Unix включают в свой состав системную программу des, реализующую шифрование (расшифрование) по алгоритму DES информации со стандартного устройства ввода с передачей результата на стандартное устройство вывода:

```
% des -e|-d [-h] [-k key] [-b]
```

Параметры *e* и *d* указывают на необходимость выполнения соответственно шифрования или расшифрования. Параметр *k* может использоваться для определения ключа шифрования непосредственно в командной строке вызова des, но это небезопасно. Параметр *b* может использоваться для замены принятого по умолчанию режима CBC на режим ECB. Параметр *h* позволяет использовать ключи шифрования, записанные в шестнадцатеричном виде (в противном случае будут возможны только ключи, состоящие из символов, которые можно вводить с клавиатуры).

Пример использования программы des для шифрования файла message в текущем каталоге:

```
% des -e < message > message.des
Enter key: ключ шифрования
Enter key again: ключ шифрования
% cat message.des
«UI}mE8NZlOi\Iy|
```

В данном примере ключ шифрования вводится дважды по запросу программы des и не отображается на экране.

Пример использования программы des для расшифрования файла message.des:

```
% des -d < message.des
Enter key: ключ шифрования
Enter key again: ключ шифрования
Это конфиденциальное сообщение.
```

В операционных системах Windows, как в открытых, начиная с версии Windows 95 OSR2, так и в защищенных доступ к шифрованию по алгоритму DES и другим возможен с помощью функций криптографического интерфейса CryptoAPI (см. гл. 5).

4.5. Криптографическая система ГОСТ 28147—89

Используемая в Российской Федерации криптосистема определена в стандарте ГОСТ 28147—89 «Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования данных» (в 1989 г. с этого алгоритма был снят гриф секретности, но разработан он был значительно раньше).

В алгоритме ГОСТ 28147—89 используется ключ шифрования k длиной 256 бит, который может рассматриваться как массив из восьми 32-битных элементов k_0, k_1, ..., k_7 (внутренних ключей). Дополнительным ключевым элементом алгоритма является таблица замен H, представляющая собой матрицу из восьми строк и шестнадцати столбцов, элементы которой — целые числа от 0 до 15. Каждая строка таблицы замен должна содержать 16 различных чисел. Таким образом, общий размер таблицы замен составляет 512 бит.

Алгоритм основной функции шифрования f, используемой в алгоритме шифрования ГОСТ 28147—89 (N — преобразуемый блок длиной 64 бита; K — один из внутренних ключей шифрования длиной 32 бита).

1. $N = N_1 \| N_2$ (два полублока по 32 бита).
2. $S = N_1 + K \{\mathrm{mod}\ 2^{32}\}$, $S = S_0 S_1 \ldots S_7$ (8 элементов по 4 бита).
3. $\forall\ i = 0, 1, \ldots, 7$: $S_i = H[i, S_i]$.
4. Циклический сдвиг S на 11 бит влево.
5. $S = S \oplus N_2$.
6. $N_2 = N_1$; $N_1 = S$.

Алгоритм шифрования блока P открытого текста.

1. $\forall\ m = 1, 2, 3$:
$\forall\ j = 0, 1, \ldots, 7$: $P = f(P, k_j)$.
2. $\forall\ j = 7, 6, \ldots, 0$: $P = f(P, k_j)$.
3. $P = P_1 \| P_2$ (два полублока по 32 бита).
4. $P_1 \leftrightarrow P_2$.

Таким образом, общее число раундов в алгоритме шифрования ГОСТ 28147—89 равно 32.

Алгоритм расшифрования блока C шифротекста.

1. $\forall\ j = 0, 1, \ldots, 7$: $C = f(C, k_j)$.
2. $\forall\ m = 1, 2, 3$:
$\forall\ j = 7, 6, \ldots, 0$: $C = f(C, k_j)$.
3. $C = C_1 \| C_2$ (два полублока по 32 бита).
4. $C_1 \leftrightarrow C_2$.

Криптосистема ГОСТ 28147—89 может использоваться в трех основных и одном дополнительном режимах. В режиме *простой замены*, который соответствует режиму ECB криптосистемы DES, каждый блок открытого текста шифруется индивидуально, независимо от других блоков:

$$\forall\ i,\ 1 \le i \le n\ C_i = E_k(P_i),$$

где $P = \{P_1, P_2, \ldots, P_i, \ldots, P_n\}$ — открытый текст; n — число 64-битных блоков открытого текста; $C = \{C_1, C_2, \ldots, C_i, \ldots, C_n\}$ — шифротекст; E — функция, реализующая алгоритм шифрования; k — ключ шифрования.

Расшифрование в режиме простой замены выполняется следующим образом:

$$\forall\ i,\ 1 \le i \le n\ P_i = D_k(C_i),$$

где D — функция расшифрования.

В соответствии с ГОСТ 28147—89 режим простой замены должен применяться только для шифрования ключей и таблиц замен, поскольку их длина кратна 64 битам, а наличие повторяющихся 64-битных блоков исключается.

В режиме *гаммирования* используется дополнительный параметр — синхропосылка $S = S_1S_0$ длиной 64 бита (S_0 — младшая часть двоичного числа S):

$\forall\ i,\ 1 \le i \le n$:

$S_0^i = S_0^{i-1} + A \{\text{mod } 2^{32}\}$; $S_1^i = (S_1^{i-1} + B - 1) \{\text{mod } 2^{32} - 1\} + 1$, $C_i = P_i \oplus E_k(S^i)$.

В ГОСТ 28147—89 определены следующие значения констант, используемых рекуррентным генератором псевдослучайных чисел: $A = 1\,010\,101_{16}$ и $B = 1\,010\,104_{16}$.

Расшифрование в режиме гаммирования выполняется следующим образом:

$$\forall\ i,\ 1 \le i \le n\ P_i = C_i \oplus E_k(S^i).$$

Режим гаммирования криптосистемы ГОСТ 28147—89 похож на режим OFB криптосистемы DES. Изменение бита в шифротексте приводит к изменению соответствующего бита открытого текста, что создает возможность внесения целенаправленных изменений в шифротекст. Однако эта особенность режима гаммирования не может считаться его недостатком, поскольку имеет отношение не к стойкости шифра, а к его аутентичности, которая в криптосистеме ГОСТ 28147—89 может быть обеспечена применением дополнительного режима генерации имитовставки.

В соответствии с ГОСТ 28147—89 синхропосылка может передаваться (храниться) вместе с шифротекстом, увеличивая его длину на 8 байт, или генерироваться по известному правилу без передачи (хранения).

В режиме *гаммирования с обратной связью* также используется синхропосылка:

$$\forall\ i,\ 1 \le i \le n\ C_i = P_i \oplus E_k(C_{i-1}),\ C_0 = S.$$

Процедура расшифрования шифротекста при использовании режима гаммирования с обратной связью:

$$\forall\ i,\ 1 \le i \le n\ P_i = C_i \oplus E_k(C_{i-1}),\ C_0 = S.$$

В отличие от режима гаммирования в данном случае искажение одного бита шифротекста приводит к искажению аналогичного бита в соответствующем блоке открытого текста и непредсказуемому изменению следующего блока открытого текста (нарушитель лишается возможности внесения целенаправленных изменений в шифротекст). Режим гаммирования с обратной связью крип-

тосистемы ГОСТ 28147—89 похож на режим CFB криптосистемы DES.

Дополнительный режим *генерации имитовставки* используется с одним из основных режимов и предназначен для обеспечения подлинности и целостности шифротекста. Алгоритм выработки имитовставки для очередного блока текста P:

$\forall\ m = 1, 2$:

$\forall\ j = 0, 1, \ldots, 7$: $\mathrm{P} = f(\mathrm{P}, k_j)$.

Вычисление имитовставки в шифротекст для открытого текста P:

$$\forall\ i,\ 1 \le i \le n\ \mathrm{S}_i = \mathrm{EI}_k(\mathrm{S}_{i-1} \oplus \mathrm{P}_i),\ \mathrm{S}_0 = 0,$$

где EI — функция, реализующая алгоритм выработки имитовставки.

В качестве имитовставки I берется младшая часть (32 бита) полученного двоичного числа S. Для внесения изменений в шифротекст нарушитель теперь должен решить две задачи:

- вычислить, не зная ключа шифрования, значение имитовставки для открытого текста;
- подобрать открытый текст под данное значение имитовставки.

Первоначально алгоритм ГОСТ 28147—89 предназначался для аппаратной реализации. Но с ростом производительности современных компьютеров стала возможной и эффективная программная реализация этой криптографической системы.

4.6. Принципы построения асимметричных криптографических систем

В основе асимметричных криптографических систем лежит понятие однонаправленной функции f, обладающей свойствами:

- простое (не требующее больших ресурсов) вычисление значения функции $y = f(x)$;
- существование обратной функции f^{-1};
- сложное (требующее ресурсов за пределами возможностей современных компьютеров) вычисление значения обратной функции $x = f^{-1}(y)$.

Фактически в асимметричной криптографии используется подкласс однонаправленных функций — однонаправленные функции с обходными путями, для которых обратная функция может быть вычислена так же просто, как и прямая, только если известна специальная информация об обходных путях. Эта специальная информация исполняет роль секретного ключа.

Пусть pk — открытый ключ функции шифрования E, а sk — секретный ключ функции расшифрования D. Тогда должны выполняться следующие условия, чтобы E и D образовывали асимметричную криптосистему.

1. $D_{sk}(E_{pk}(P)) = P$ (расшифрование должно восстанавливать открытый текст P).

2. Функции E_{pk} и D_{sk} должны быть просты в реализации.

3. При раскрытии преобразования, выполняемого с помощью E_{pk}, не должно раскрываться преобразование, выполняемое с помощью D_{sk} (из открытого ключа нельзя получить секретный ключ).

4. $D_{pk}(E_{sk}(P)) = P$ (возможно использование секретного ключа для шифрования, а открытого — для расшифрования).

Четвертое условие является необязательным и не все асимметричные криптосистемы им обладают.

К основным применениям асимметричных криптосистем относятся:

- передача ключа симметричного шифрования по открытой сети (отправитель зашифровывает этот ключ с помощью открытого ключа получателя, который только и сможет расшифровать полученное сообщение с помощью своего секретного ключа);
- системы электронной цифровой подписи для защиты электронных документов (создатель документа удостоверяет его подлинность с помощью своего секретного ключа, после чего любой владелец соответствующего открытого ключа сможет проверить аутентичность данного документа, см. подразд. 4.7).

Для использования асимметричной криптосистемы для получения и проверки электронной цифровой подписи она должна удовлетворять четвертому из приведенных ранее условий.

В отличие от классической симметричной криптографии криптография с открытым ключом появилась сравнительно недавно — во второй половине XX в. К особенностям современных асимметричных криптосистем, которые не позволяют им полностью заменить симметричные криптосистемы, относятся:

- большая продолжительность процедур шифрования и расшифрования (примерно в 1000 раз больше);
- необходимость использования существенно более длинного ключа шифрования для обеспечения той же криптостойкости шифра (например, симметричному ключу длиной 56 бит будет соответствовать асимметричный ключ длиной 384 бита, а симметричному ключу длиной 112 бит — асимметричный ключ длиной 1792 бита).

К наиболее известным асимметричным криптографическим системам относятся RSA (Rivest, Shamir, Adleman), Диффи-Хеллмана, Эль-Гамаля и криптосистема на основе эллиптических кривых.

В криптосистеме RSA ключи шифрования выбираются следующим образом:

1) выбираются два больших простых числа p и q;

2) вычисляется значение модуля $n = pq$;

3) выбирается достаточно большое целое число y, которое является взаимно простым с $\varphi(n)$ и вместе с n образует секретный ключ шифрования (y, n) ($\varphi(n)$ — функция Эйлера);

4) вычисляется целое число x, которое является мультипликативно обратным числу y по модулю $\varphi(n)$ и вместе с n образует открытый ключ шифрования (x, n).

Шифрование по алгоритму RSA выполняется следующим образом:

$$C = P^x \{\mathrm{mod}\ n\},$$

где P — открытый текст; C — шифротекст.

Для расшифрования шифротекста производится следующее действие:

$$P = C^y \{\mathrm{mod}\ n\}.$$

В приведенных функциях шифрования и расшифрования предполагается, что $P<n$ (в противном случае открытый текст перед шифрованием разбивается на блоки).

Докажем, что при расшифровании по алгоритму RSA восстанавливается открытый текст. Вначале рассмотрим случай, когда P и n являются взаимно простыми.

$C^y \{\mathrm{mod}\ n\} = (P^x)^y \{\mathrm{mod}\ n\} = P^{xy} \{\mathrm{mod}\ n\} = P^{1+\varphi(n)k} \{\mathrm{mod}\ n\} = PP^{\varphi(n)k} \{\mathrm{mod}\ n\} = P1^k \{\mathrm{mod}\ n\} = P.$

Сделаем необходимые пояснения. Равенство $xy = 1 + \varphi(n)k$ (k — произвольное целое число) следует из условия выбора значения x. Равенство $P^{\varphi(n)} = 1 \{\mathrm{mod}\ n\}$ следует из теоремы Эйлера.

Теперь рассмотрим случай, когда P и n имеют общий делитель. Поскольку $P<n$ и $n = pq$, можно положить $P = kp$, где k — некоторое целое число. Тогда P взаимно просто с q и $P^{q-1} = 1 \{\mathrm{mod}\ q\}$ (из малой теоремы Ферма). Далее можно записать $1^{p-1} = P^{\varphi(n)} \{\mathrm{mod}\ q\}$ и $1^k = P^{k\varphi(n)} \{\mathrm{mod}\ q\}$. Из условия выбора x следует, что $xy - 1 = k\varphi(n)$, поэтому $1 = P^{xy-1} \{\mathrm{mod}\ q\}$ и $P = P^{xy} \{\mathrm{mod}\ q\}$. Пусть $a = P^{xy-1} \{\mathrm{mod}\ q\}$. Тогда $\exists$ целое число r, для которого выполняется равенство $P^{xy} = kp(rq + a)$. Тогда $kprq + kpa = P \{\mathrm{mod}\ q\}$ и, поскольку $kprq = 0 \{\mathrm{mod}\ q\}$, $kpa = kp \{\mathrm{mod}\ q\}$ и $(a - 1)pk = 0 \{\mathrm{mod}\ q\}$. На основании выбора n и того, что $P<n$, $k<q$ и $a<q$. Поэтому $a = 1$. Отсюда $P^{xy} \{\mathrm{mod}\ n\} = kp = P$.

Криптостойкость алгоритма RSA основывается на сложности задачи разложения на множители большого целого числа n (задачи факторизации n). Если криптоаналитику удастся разложить n на множители p и q, то он сможет вычислить значение $\varphi(n) = (p - 1)(q - 1)$, затем определить значение y и раскрыть тем самым параметры шифрования. На современном уровне развития компьютерных технологий значение n должно содержать не менее 1024 бит.

При выборе параметров шифрования по алгоритму RSA простые числа p и q должны выбираться случайным образом. По теореме о простых числах вблизи целого числа m в среднем есть одно простое число на $\ln m$ целых чисел. Поэтому даже для очень больших простых чисел с сотнями десятичных цифр потребуется про-

верить на простоту только несколько сотен чисел. При подобной проверке могут использоваться различные тесты: Ферма, Соловея-Штрассена или Рабина.

Тест Ферма, например, основан на малой теореме Ферма. Если p — проверяемое целое число, то выбирается $a<p$ и проверяется условие $a^{p-1} = 1 \{\text{mod } p\}$. Рекомендуется выполнить эту проверку для 100 значений a, чтобы с надежностью удостовериться в том, что p — простое число.

Выбираемые простые числа p и q не должны принадлежать специальным классам целых чисел (числам Ферма или Мерсенна), так как эти числа хорошо изучены, что облегчает задачу факторизации n.

Выбор секретного ключа y в криптосистеме RSA выполняется на основе вычисления НОД(y, $\varphi(n)$) с помощью алгоритма Евклида.

1. $a = y$; $b = \varphi(n)$.
2. Если $a \neq b$, то переход к п. 3, в противном случае — к п. 5.
3. Если $a>b$, то $a = a - b$; в противном случае $b = b - a$.
4. Переход к п. 2.
5. Конец (НОД(y, $\varphi(n)$) = a).

Если НОД(y, $\varphi(n)$) = 1, то значение y выбрано. Для вычисления значения x как мультипликативно обратного к y по модулю $\varphi(n)$ применяется частный режим работы расширенного алгоритма Евклида, использующего равенство НОД(y, $\varphi(n)$) = 1 и трехмерные векторы u, v и t:

1. $u = \{0, 1, \varphi(n)\}$; $v = \{1, 0, y\}$.
2. Если $u_3 = 1$, то переход к п. 4; в противном случае — переход к п. 3.
3. $s = [u_3/v_3]$ (целая часть); $t = u - vq$; $u = v$; $v = t$; переход к п. 2.
4. Конец ($x = u_1$).

Пример. Пусть $p = 5$, $q = 11$, $\varphi(n) = 40$, $y = 23$. Требуется найти $x = y^{-1} \{\text{mod } 40\}$. Применяя частный режим расширенного алгоритма Евклида, запишем промежуточные вычисления в табл. 4.2.

Таблица 4.2

s	u_1	u_2	u_3	v_1	v_2	v_3
	0	1	40	1	0	23
1	1	0	23	–1	1	17
1	–1	1	17	2	–1	6
2	2	–1	6	–5	3	5
1	–5	3	5	7	–4	1
	7	–4	1			

Итак, в результате получаем $x = 7 = 23^{-1} \{\text{mod } 40\}$.

Шифрование в криптосистеме RSA может быть выполнено по следующему алгоритму.

1. $x = x_k x_{k-1} x_i x_1 x_0$ (двоичное представление x)
2. C = 1.
3. $\forall\ i = k, k-1, \ldots, 1, 0$:
   a. $C = C^2 \{\text{mod } n\}$.
   b. Если $x_i = 1$, то $C = CP \{\text{mod } n\}$.
4. Конец.

Для расшифрования может использоваться тот же алгоритм с заменой x на y.

Криптосистемы *Диффи-Хеллмана* и *Эль-Гамаля* основаны на вычислительной сложности задачи дискретного логарифмирования: вычисление $y = a^x \{\text{mod } p\}$ (p — простое число или степень простого числа, $1<x<p-1$, $1<a<p-1$, $\forall b$, $1<b<p-1$ $\exists c$ $a^c = b \{\text{mod } p\}$) выполняется просто, но вычисление $x = \log_a y \{\text{mod } p\}$ выполняется достаточно сложно.

Алгоритм Диффи-Хеллмана предназначен только для генерации ключа симметричного шифрования, который затем будет использован субъектами A и B для защищенного обмена сообщениями по открытой сети.

1. A: выбирает xa и вычисляет $y_a = a^{xa} \{\text{mod } p\}$.
2. B: выбирает xb и вычисляет $y_b = a^{xb} \{\text{mod } p\}$.
3. A→B: y_a.
4. B→A: y_b.
5. A: вычисляет $k_a = (y_b)^{xa} \{\text{mod } p\}$.
6. B: вычисляет $k_b = (y_a)^{xb} \{\text{mod } p\}$.
7. Конец ($k_a = k_b$ и созданный ключ может теперь использоваться для защищенного обмена сообщениями между A и B).

Значения a и p в алгоритме Диффи-Хеллмана не являются секретными, поскольку, даже зная их, нарушитель не сможет решить задачу дискретного логарифмирования и найти значения xa и xb, чтобы вычислить сгенерированный ключ симметричного шифрования.

В криптосистеме Эль-Гамаля значение a вместе с значениями p и y составляет открытый ключ, а секретным ключом является значение x ($y = a^x \{\text{mod } p\}$). Шифрование открытого текста P в криптосистеме Эль-Гамаля выполняется по следующему алгоритму.

1. Выбор случайного целого числа k ($1<k<p-1$ и НОД(k, $p-1$) = = 1).
2. $C_1 = a^k \{\text{mod } p\}$.
3. $C_2 = Py^k \{\text{mod } p\}$.
4. Конец (шифротекстом являются значения C_1 и C_2).

Расшифрование в криптосистеме Эль-Гамаля производится путем составления сравнения $PC_1^x = C_2 \{\text{mod } p\}$ и решения его относительно P. Действительно: $PC_1^x \{\text{mod } p\} = P(a^k)^x \{\text{mod } p\} = P(a^x)^k \{\text{mod } p\} = Py^k \{\text{mod } p\} = C_2 \{\text{mod } p\}$.

Если $P \geq p$, то открытый текст должен быть разбит на блоки, длина которых равна длине числа p. В п. 3 алгоритма шифрования вместо операции умножения может использоваться операция сложения по модулю 2 ($C_2 = P \oplus y^k \{\text{mod } p\}$). Тогда при расшифровании восстановление открытого текста выполняется следующим образом:

```
P = (C1^x {mod p}) ⊕ C2 (так как C1^x {mod p} = y^k {mod p}).
```

Недостатком этого варианта является то, что открытый текст должен разбиваться на блоки заранее неизвестной длины $y^k \{\text{mod } p\}$.

В криптосистеме на основе *эллиптических кривых* используются алгебраические структуры, определенные на множестве точек на эллиптической кривой:

$$\{(x, y) \mid y^2 = x^3 + ax + b\} \cup \{(\infty, \infty)\}.$$

Для точек на эллиптической кривой вводится операция сложения (добавления новой точки по двум известным), которая допускает простую реализацию и играет ту же роль, что и операция дискретного возведения в степень в криптосистемах Диффи-Хеллмана и Эль-Гамаля. Определив, таким образом, операцию сложения, можно ввести и операцию умножения точки эллиптической кривой G на целое число x:

$$G + G + \ldots + G(x \text{ раз}) = xG.$$

В реальных криптосистемах вычисление $y^2 = x^3 + ax + b$ производится по простому модулю p. Если y и G — две точки на эллиптической кривой, связанные соотношением $y = xG$, то значение x будет являться секретным ключом, а значение y вместе с значениями G, a, b и p составит открытый ключ.

Криптостойкость системы на основе эллиптических кривых определяется вычислительной сложностью нахождения целого числа x по известным точкам эллиптической кривой y и G. Основным применением асимметричных криптосистем на основе эллиптических кривых являются системы электронной цифровой подписи.

4.7. Электронная цифровая подпись и ее применение

Механизм *электронной цифровой подписи* должен обеспечить защиту от следующих угроз безопасности электронных документов, передаваемых по открытым компьютерным сетям или хранящихся на открытых носителях:

• подготовка документа от имени другого субъекта («маскарада»);
• отказ автора документа от факта его подготовки (ренегатства);
• изменение получателем документа его содержания (подмены);
• изменение содержания документа третьим лицом (активного перехвата);
• повторная передача по компьютерной сети ранее переданного документа (повтора).

Электронная цифровая подпись (ЭЦП) представляет собой относительно небольшой по объему блок данных, передаваемый (хранящийся) вместе (реже — отдельно) с подписываемым с ее помощью документом. Механизм ЭЦП состоит из двух процедур: получение (простановка) подписи с помощью секретного ключа автора документа и проверка ЭЦП при помощи открытого ключа автора документа.

Алгоритм получения ЭЦП под документом P.

1. Вычисление хеш-значения H(P) для документа P.
2. Шифрование H(P) с помощью секретного ключа автора документа ska — $E_{ska}(H(P))$ (полученный шифротекст и будет являться ЭЦП).

Алгоритм проверки ЭЦП C под документом P.

1. Вычисления хеш-значения H(P) для документа P.
2. Расшифрование ЭЦП с помощью открытого ключа автора документа pka — $D_{pka}(C) = D_{pka}(E_{ska}(H(P))) = H(P)$.
3. Сравнение вычисленного и расшифрованного хеш-значений для документа P.

Перед получением ЭЦП в подписываемый документ должны быть включены дополнительные сведения:

• дата и время простановки подписи;
• срок окончания действия секретного ключа данной подписи;
• реквизиты (фамилия, имя, отчество подписывающего лица, его должность и название представляемой организации);
• идентификатор секретного ключа (для возможности выбора лицом, проверяющим ЭЦП, нужного открытого ключа).

В системе ЭЦП подпись под электронным документом невозможно подделать без знания секретного ключа автора документа, поэтому компрометация секретного ключа недопустима.

Известны следующие системы ЭЦП:

• RSA (на основе асимметричной криптосистемы RSA);
• DSS (Digital Signature Standard, стандарт США на основе асимметричной криптосистемы Эль-Гамаля);
• ГОСТ Р 34.10—94 (российский стандарт ЭЦП на основе асимметричной криптосистемы Эль-Гамаля);
• ГОСТ Р 34.10—2001 (российский стандарт ЭЦП, использующий асимметричную криптосистему на основе эллиптических кривых).

Алгоритмы получения и проверки ЭЦП в системе RSA не отличаются от алгоритмов шифрования и расшифрования в аналогичной криптосистеме, за исключением того, что получение ЭЦП производится с применением секретного ключа y (см. подразд. 4.6), а проверка ЭЦП — с применением открытого ключа x.

Алгоритмы получения и проверки ЭЦП в системе Эль-Гамаля отличаются от алгоритмов шифрования и расшифрования в аналогичной криптосистеме. Алгоритм получения ЭЦП под документом P.

1. Выбор случайного целого числа k ($1 < k < p-1$ и НОД(k, $p-1$) = = 1) (k — случайная составляющая ЭЦП, изменяющая подпись под вновь отправляемым по сети тем же самым документом).
2. $C_1 = a^k \{\text{mod } p\}$.
3. Определение C_2 из сравнения $P = xC_1 + kC_2 \{\text{mod } p-1\}$ (C_1 и C_2 образуют ЭЦП для P).

Проверка ЭЦП в системе Эль-Гамаля сводится к проверке сравнения $y^{C1}C_1^{C2} \{\text{mod } p\} = a^P \{\text{mod } p\}$. Действительно: $y^{C1}C_1^{C2} \{\text{mod } p\} = a^{xC1}a^{kC2} \{\text{mod } p\} = a^{xC1+kC2} \{\text{mod } p\} = a^{P(\text{mod } p-1)} \{\text{mod } p\} = = a^{P+m(p-1)} \{\text{mod } p\} = a^P(a^m)^{p-1} \{\text{mod } p\} = a^P \{\text{mod } p\}1$ (из малой теоремы Ферма).

Алгоритм получения ЭЦП под документом P в системе на базе эллиптических кривых (целое число g выбирается из условия $gG = 0$).

1. Выбор случайного целого числа k ($1 < k < g$) — случайной составляющей ЭЦП, изменяющей подпись под вновь отправляемым по сети тем же самым документом).
2. Вычисление $c = kG$.
3. $r = x_c \{\text{mod } g\}$ (x_c — x-координата точки c).
4. $s = rx + kP \{\text{mod } g\}$.
5. Если $r \neq 0$ и $s \neq 0$, то эти значения образуют ЭЦП, в противном случае выбирается другое k и пп. 2 ...4 повторяются.

Алгоритм проверки ЭЦП в системе на основе эллиптических кривых.

1. Вычисление $V = P^{-1} \{\text{mod } g\}$.
2. $z_1 = sV \{\text{mod } g\}$; $z_2 = -rV \{\text{mod } g\}$.
3. Вычисление $c = z_1G + z_2y$; $R = x_c \{\text{mod } g\}$.
4. Проверка $R = r$.

Действительно: $z_1 = sV \{\text{mod } g\} = rxP^{-1} + kPP^{-1}\{\text{mod } g\} = rxP^{-1} + + k \{\text{mod } g\}$ и $z_2 = -rV \{\text{mod } g\} = -rP^{-1} \{\text{mod } g\}$. Тогда $z_1G = rxP^{-1}G + + kG$ и $z_2y = -rP^{-1}y = -rP^{-1}xG$. Следовательно, $c = rxP^{-1}G + kG - - rP^{-1}xG = kG$.

Защищенность системы ЭЦП от угрозы аутентичности и целостности подписанных документов зависит не только от стойкости

алгоритмов используемой асимметричной криптосистемы, но и от стойкости функции хеширования (см. подразд. 4.2). На функции хеширования, используемые в системах ЭЦП, налагаются очевидные дополнительные условия:

- чувствительность к любым изменениям в документе (вставкам, удалением, перестановкам, заменам фрагментов и отдельных символов);
- минимальность вероятности того, что хеш-значения двух разных документов, независимо от их длин, совпадут.

К наиболее известным функциям хеширования относятся:

- MD2, MD4, MD5 (Message Digest) — получают хеш-значение длиной 128 бит и используются в системе ЭЦП RSA;
- SHA (Secure Hash Algorithm) — получает хеш-значение длиной 160 бит и используется в системе ЭЦП DSS;
- ГОСТ Р 34.11—94 — получает хеш-значение длиной 256 бит и используется в российских стандартах ЭЦП;
- RIPEMD (Race Integrity Primitives Evaluation Message Digest) — получает хеш-значение длиной 128 или 160 бит (две модификации).

4.8. Использование симметричных и асимметричных криптографических систем

Рассмотрим примерную схему использования симметричной криптографической системы для создания защищенного канала связи между субъектами распределенной КС.

1. Безопасное создание, распространение и хранение ключа симметричного шифрования k (см. подразд. 4.3).

2. Вычисление хеш-значения (имитовставки, кода аутентификации сообщения) для открытого текста P и присоединение его к тексту:

a. Вычисление H(P).

b. $P' = P \| H(P)$.

3. Получение шифротекста для преобразованного открытого текста $C = E_k(P')$.

4. Передача шифротекста по незащищенному каналу связи.

5. Расшифрование полученного шифротекста (неявная аутентификация полученного шифротекста, так как только имевший ключ шифрования k мог выполнить шифрование) $P' = D_k(C)$.

6. Отделение хеш-значения (имитовставки, кода аутентификации сообщения) от открытого текста $P' = P \| H(P)$.

7. Вычисление хеш-значения (имитовставки, кода аутентификации сообщения) для полученного открытого текста H(P).

8. Сравнение полученного и вычисленного хеш-значений (проверка целостности полученного открытого текста).

Использование асимметричных криптосистем с симметричными позволяет отказаться от распределения и хранения ключей симметричного шифрования. Рассмотрим схему организации защищенного канала связи между субъектами распределенной КС, использующую асимметричное и симметричное шифрование.

1. Безопасное создание и распространение открытых и секретных ключей асимметричного шифрования. Секретный ключ передается его владельцу. Открытый ключ хранится в базе данных открытых ключей, которая администрируется удостоверяющим центром (Certification Authority, CA). Все субъекты распределенной КС доверяют CA и полученным от него сертификатам открытых ключей других субъектов. Если создатель ключей асимметричного шифрования и CA — разные КС (организации, лица), то субъекты уверены, что создатель уничтожил все копии их секретных ключей шифрования.

2. Вычисление хеш-значения для открытого текста P, его шифрование секретным ключом отправителя ska и добавление к открытому тексту:

a) Вычисление H(P).
b) Вычисление $S = E_{ska}(H(P))$.
c) $P' = P \parallel S$.

3. Создание секретного симметричного (сеансового) ключа шифрования k и шифрование с его помощью открытого текста и ЭЦП под ним $C = E_k(P')$.

4. Защищенная передача получателю сеансового ключа шифрования:

a) отправитель имеет открытый ключ асимметричного шифрования удостоверяющего центра CA pkca, который должен быть предварительно распределен по защищенному каналу связи;

b) отправитель запрашивает у CA открытый ключ получателя pkb, подписанный секретным ключом CA $E_{skca}(pkb)$;

c) отправитель расшифровывает полученный открытый ключ получателя $pkb = D_{pkca}(E_{skca}(pkb))$;

d) отправитель шифрует сеансовый ключ симметричного шифрования с помощью открытого ключа получателя $E_{pkb}(k)$;

e) отправитель присоединяет зашифрованный сеансовый ключ к полученному при выполнении п. 3 шифротексту $C' = C \parallel E_{pkb}(k)$.

5. Передача C′ по незащищенному каналу связи.

6. Выделение получателем зашифрованного сеансового ключа $C' = C \parallel E_{pkb}(k)$.

7. Расшифрование получателем сеансового ключа с помощью своего секретного ключа асимметричного шифрования skb $k = D_{skb}(E_{pkb}(k))$.

8. Расшифрование полученного шифротекста с помощью восстановленного сеансового ключа $P' = D_k(C)$.

9. Отделение ЭЦП от расшифрованного открытого текста $P' = P \| S$.

10. Получение получателем открытого ключа асимметричного шифрования отправителя pka:

a) получатель имеет открытый ключ асимметричного шифрования удостоверяющего центра CA pkca, который должен быть предварительно распределен по защищенному каналу связи;

b) получатель запрашивает у CA открытый ключ отправителя pka, подписанный секретным ключом CA $E_{skca}(pka)$;

c) получатель расшифровывает полученный открытый ключ отправителя $pka = D_{pkca}(E_{skca}(pka))$.

11. Расшифрование ЭЦП с помощью открытого ключа отправителя $H(P) = D_{pka}(S)$.

12. Вычисление хеш-значения для расшифрованного открытого текста H(P).

13. Сравнение расшифрованного из ЭЦП и вновь вычисленного хеш-значений для проверки аутентичности и целостности полученного открытого текста.

Альтернативой использования единого удостоверяющего центра может быть сетевая модель доверительных отношений. В этом случае часть субъектов распределенной КС обмениваются открытыми ключами напрямую (по защищенному каналу), после чего сертификаты открытых ключей других субъектов заверяются либо секретным ключом непосредственно известного субъекта, либо по иерархической лестнице (цепочке сертификатов), вплоть до достижения такого субъекта.

В подразд. 4.3 говорилось, что возможным решением задачи распределения ключей симметричного шифрования между субъектами КС является использование центра распределения ключей KDC.

Рассмотрим протокол, позволяющий обеспечить анонимное распределение сеансовых ключей, при котором KDC не может определить, какой именно ключ получил субъект A (skc и pkc — соответственно секретный и открытый ключи асимметричного шифрования центра распределения ключей, а ska и pka — секретный и открытый ключи асимметричного шифрования субъекта A).

1. A: выбирает ska и pka (в данном протоколе открытый ключ также должен оставаться конфиденциальным).
2. C: генерирует последовательность сеансовых ключей k_1, k_2, ...
3. C: вычисляет $E_{pkc}(k_1)$, $E_{pkc}(k_2)$, ...
4. C→A: $E_{pkc}(k_1)$, $E_{pkc}(k_2)$, ...
5. A: случайно выбирает $E_{pkc}(k_i)$.
6. A: вычисляет $E_{pka}(E_{pkc}(k_i))$.
7. A→C: $E_{pka}(E_{pkc}(k_i))$.
8. C: вычисляет $E_{pka}(k_i) = D_{skc}(E_{pka}(E_{pkc}(k_i)))$.
9. C→A: $E_{pka}(k_i)$.

10. A: вычисляет $k_i = D_{ska}(E_{pka}(k_i))$.

В протоколе анонимного распределения ключей предполагается, что для используемой в нем асимметричной криптосистемы выполняется следующее условие: $D_{skc}(E_{pka}(E_{pkc}(k_i))) = E_{pka}(D_{skc}(E_{pkc}(k_i))) = E_{pka}(k_i)$. Это условие, например, выполняется для криптосистемы RSA.

Асимметричные криптосистемы служат основой для создания *инфраструктуры открытых ключей* (Public Key Infrastructure, PKI), которая представляет собой интегрированный набор служб и средств администрирования для разработки и развертывания приложений, применяющих криптографию с открытым ключом, а также для управления ими. Пользователи КС получают в удостоверяющем центре сертификаты открытых ключей в результате обработки соответствующего запроса, который может быть подан различными способами:

- через веб-страницу;
- с помощью соответствующего мастера;
- автоматически при входе пользователя в систему.

Сертификат открытого ключа пользователя КС содержит обычно следующую информацию:

- номер версии сертификата;
- идентификатор (серийный номер) сертификата;
- сведения об алгоритме ЭЦП, для которого может использоваться данный открытый ключ;
- сведения о поставщике сертификата;
- сроки действия сертификата;
- сведения о субъекте, которому был выдан сертификат;
- открытый ключ асимметричного шифрования;
- идентификатор открытого ключа удостоверяющего центра;
- сведения об используемой в сертификате функции хеширования;
- читаемое имя сертификата;
- описание сертификата и др.

Данная информация зашифровывается с помощью секретного ключа удостоверяющего центра, после чего и становится собственно сертификатом, который содержит открытый ключ субъекта и одновременно может служить основой для его аутентификации в КС (см. подразд. 2.1), поскольку шифрование информации в сертификате могло быть выполнено только удостоверяющим центром.

Секретные ключи субъектов КС после их генерации могут храниться в защищенном виде на различных типах носителей:

- в реестре операционной системы;
- на дискетах;
- на интеллектуальных смарт-картах (см подразд. 2.4);
- в элементах Touch Memory (см. подразд. 2.4) и др.

Сертификат открытого ключа пользователя КС может быть скомпрометирован до истечения срока его действия в результате:

- потери носителя с секретным ключом (даже при его последующем обнаружении);
- увольнения сотрудника, имевшего доступ к секретным ключам;
- нарушения правил хранения и уничтожения (по истечении срока действия сертификатов) секретных ключей;
- получения сертификата незаконным путем (в обход установленной процедуры);
- возникновения подозрений о возможной утечке информации или ее искажении в системе конфиденциальной связи;
- изменения статуса субъекта в КС (например, понижения его полномочий);
- возникновения случаев, когда нельзя достоверно установить, что произошло с носителями секретных ключей (в том числе случаев, когда ключевой носитель вышел из строя и доказательно не опровергнута возможность того, что данный факт произошел в результате несанкционированных действий злоумышленника).

При компрометации сертификата открытого ключа удостоверяющим центром должна быть выполнена процедура его отзыва. В инфраструктуре открытых ключей для этого предусмотрена поддержка списков отозванных сертификатов (Certificate Revocation List, CRL), публикация которых в КС является обязанностью ее администратора. Клиенты могут получить эту информацию и записать ее в свою локальную кэш-память, чтобы использовать для проверки сертификатов других субъектов КС.

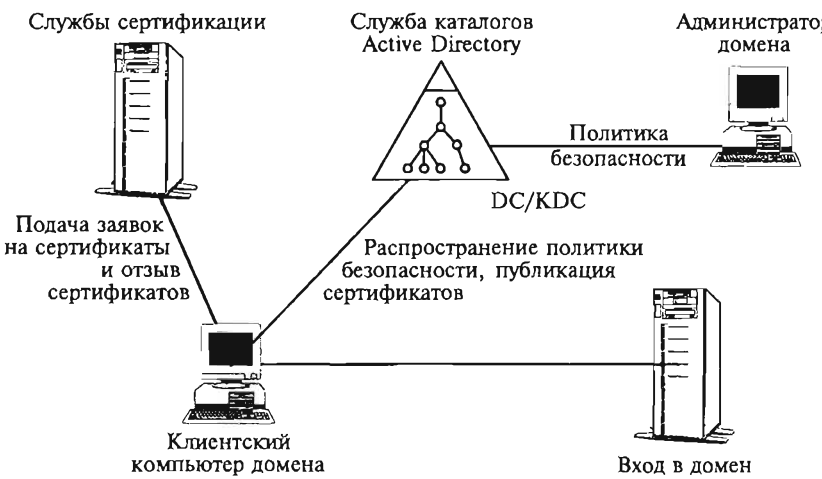

Рис. 4.4. Компоненты инфраструктуры открытых ключей операционных систем Windows 2000/XP Professional

На рис. 4.4 показаны основные компоненты инфраструктуры открытых ключей операционных систем Windows 2000/XP Professional.

Примером программы, объединяющей возможности симметричной и асимметричной криптографии, является программа PGP (Pretty Good Privacy). В PGP допускается использование и удостоверяющего центра, роль которого могут сыграть как любой пользователь PGP, так и сервер компании (Network Associates, Inc.), и сетевой модели доверия. При использовании программы PGP для каждого открытого ключа хранится присвоенный ему пользователем программы уровень доверия (полного или частичного). Для признания истинности полученного пользователем открытого ключа требуется скрепление его одной ЭЦП с ключом полного доверия или двумя ЭЦП с ключами частичного доверия.

Если электронные документы организации должны подписываться несколькими лицами, то секретный ключ при использовании PGP может быть разделен на несколько частей для того, чтобы одно лицо не могло представлять всю организацию.

В программе PGP могут использоваться системы ЭЦП DSS (вместе с функцией хеширования SHA) и RSA (вместе с функцией хеширования MD5).

В пакете PGP секретный и открытый ключи пользователя создаются с помощью программы PGPkeys и хранятся в файлах с расширениями«.skr» и «.pkr» соответственно. Секретный ключ хранится в зашифрованном виде с помощью ключа симметричного шифрования, основанного на заданной пользователем ключевой фразе. Получение ЭЦП для выбранного файла выполняется в программе PGP с помощью функции Sign (рис. 4.5). При этом возможен выбор секретного ключа ЭЦП (для его расшифрования потребуется ввод ключевой фразы), а также указание на размещение ЭЦП в отдельном файле (detached signature) с расширением «.sig» и преобразование ЭЦП в читаемый вид (text output) для

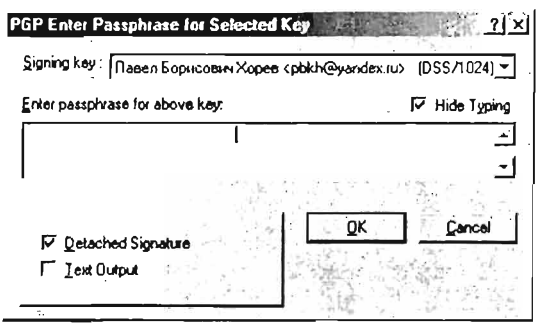

Рис. 4.5. Получение ЭЦП в программе PGP

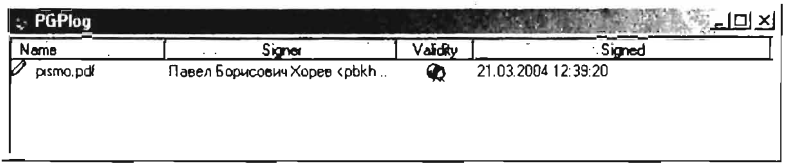

Рис. 4.6. Результат проверки ЭЦП в программе PGP

передачи, например, в обычном письме, по телефону или факсу. Для проверки ЭЦП в программе PGP используется функция Verify (пример результата проверки приведен на рис. 4.6).

Для симметричного шифрования в программе PGP используются одноразовые сеансовые ключи, для получения которых применяется программный датчик псевдослучайных чисел. Первый из сеансовых ключей генерируется с помощью создаваемых пользователем случайных событий (перемещением мыши на клавиатуре). Все последующие сеансовые ключи создаются на основе времени суток и других истинно случайных величин. В программе PGP могут использоваться симметричные криптосистемы CAST, 3-DES и IDEA (см. подразд. 4.3).

Шифрование файлов выполняется с помощью функции Encrypt программы PGP (рис. 4.7). При этом возможно включение зашифрованного сеансового ключа в текст зашифрованного файла (при шифровании сеансового ключа будет использован открытый ключ

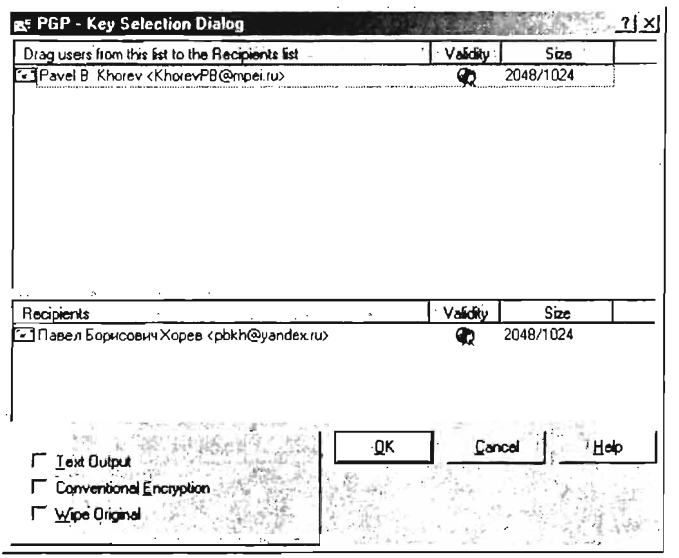

Рис. 4.7. Шифрование файлов в программе PGP

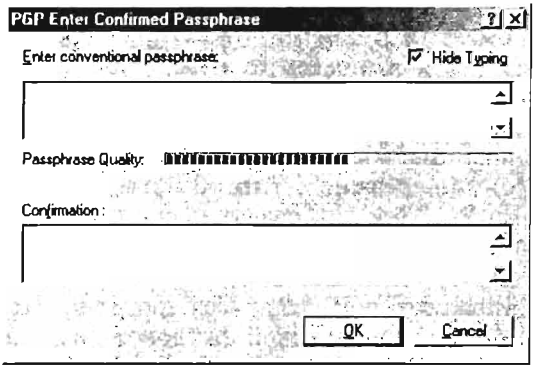

Рис. 4.8. Ввод ключевой фразы в программе PGP

получателя этого файла) или генерация сеансового ключа на основе специальной ключевой фразы, которая в этом случае должна быть известна и получателю зашифрованного файла (режим conventional encryption, пример ввода ключевой фразы на рис. 4.8). Другими возможными режимами шифрования в программе PGP являются создание зашифрованного файла в читаемом формате (text output) и надежное уничтожение файла с открытым текстом (wipe original). Зашифрованные программой PGP файлы получают расширение «.pgp».

При расшифровании файла с шифротекстом в программе PGP потребуется ввести ключевую фразу для расшифрования секретного ключа получателя или генерации сеансового ключа (рис. 4.9). Для расшифрования применяется функция Decrypt.

В программе PGP возможно одновременное выполнение шифрования и получения ЭЦП с помощью функции Encrypt Sign.

К другим возможностям пакета PGP относятся:

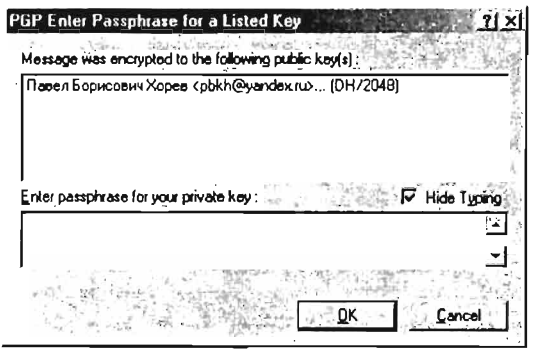

Рис. 4.9. Расшифрование файлов в программе PGP

• функция Wipe для надежного удаления файла с конфиденциальной информацией путем многократной перезаписи данных в сектора диска, ранее занимаемые этим файлом;

• функция Freespace Wipe для надежной очистки всех свободных областей магнитного диска (например, в которых ранее хранилась конфиденциальная информация) с возможностью выбора числа повторений перезаписи этих областей (рис. 4.10);

• программа PGPdisk для создания на жестком диске компьютера виртуального диска с зашифрованными файлами, для доступа к которому необходим ввод ключевой фразы создавшего этот диск пользователя; в отличие от других программных средств шифрования (например, шифрующей файловой системы операционной системы Windows, см. подразд. 5.6) при использовании программы PGPdisk скрывается число и имена зашифрованных файлов;

• модули расширения для почтовых программ Microsoft Outlook Express, Microsoft Outlook и Eudora;

• программа Secure Viewer для отображения на экране расшифрованной информации с помощью специального шрифта, препятствующего получению конфиденциальной информации по каналам ПЭМИН (см. подразд. 1.2);

• программа Photographic User ID для добавления в сертификат открытого ключа пользователя его фотографии и др.

Программа PGP существует и в версии для операционных систем семейства Unix (вызывается с помощью команды pgp).

Пример использования программы PGP для шифрования файла message в текущем каталоге с помощью сеансового ключа, гене-

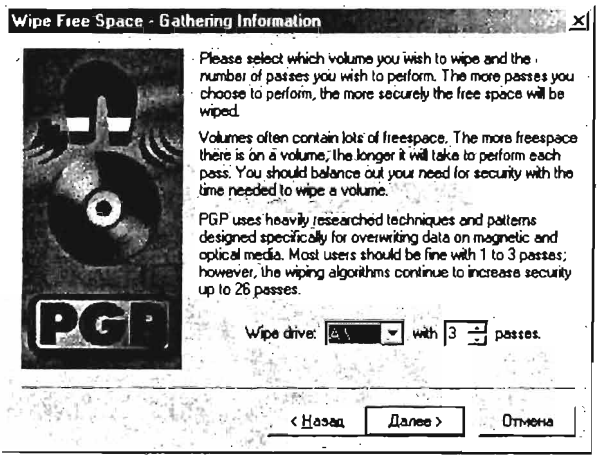

Рис. 4.10. Использование функции Freespace Wipe программы PGP

рируемого из введенной ключевой фразы (файл с шифротекстом получает имя message.pgp):

```
% pgp -c message
Pretty Good Privacy(tm) 2.6.1 - Public-key encryption
for the masses.
(c) 1990-1994 Philip Zimmermann, Phil's Pretty Good
Software. 29 Aug 94
Distributed by the Massachusetts Institute of Technology.
Uses RSAREF.
Export of this software may be restricted by the U.S.
government.
Current time: 2004/02/12 03:32 GMT
You need a pass phrase to encrypt the file.
Enter pass phrase: some days green tomatoes
Enter same pass phrase again: some days green tomatoes
Just a moment...
Ciphertext file: message.pgp
```

Пример расшифрования файла (требуется ввод ключевой фразы):

```
% pgp message.pgp
Pretty Good Privacy(tm) 2.6.1 - Public-key encryption
for the masses.
(c) 1990-1994 Philip Zimmermann, Phil's Pretty Good
Software. 29 Aug 94
Distributed by the Massachusetts Institute of Technology.
Uses RSAREF.
Export of this software may be restricted by the U.S.
government.
Current time: 2004/02/12 03:47 GMT
File is conventionally encrypted.
You need a pass phrase to decrypt this file.
Enter pass phrase: some days green tomatoes
Just a moment...Pass phrase appears good.
Plaintext filename: message
```

Если при расшифровании вводится неверная ключевая фраза, то восстановление открытого текста невозможно:

```
% pgp message.pgp
Pretty Good Privacy(tm) 2.6.1 - Public-key encryption
for the masses.
(c) 1990-1994 Philip Zimmermann, Phil's Pretty Good
Software. 29 Aug 94
Distributed by the Massachusetts Institute of Technology.
Uses RSAREF.
```

```
Export of this software may be restricted by the U.S.
government.
Current time: 2004/02/12 03:48 GMT
File is conventionally encrypted.
You need a pass phrase to decrypt this file.
Enter pass phrase: I am the walrus
Just a moment...
Error: Bad pass phrase.
You need a pass phrase to decrypt this file.
Enter pass phrase: Love will find a way
Just a moment...
Error: Bad pass phrase.
For a usage summary, type: pgp -h
For more detailed help, consult the PGP User's Guide.
```

Для доступа к другим функциям программы PGP команда pgp должна содержать дополнительные параметры: kg (создание пользовательских ключей асимметричного шифрования), ks (создание сертификата открытого ключа пользователя), kx (отзыв сертификата), ka (добавление в базу сертификата, полученного от другого пользователя), sat (получение ЭЦП вместе с подписываемыми данными в файле с расширением «.asc»), sb (получение ЭЦП в отдельном файле с расширением «.sig»)

В соответствии с требованиями «Оранжевой книги» (см. подразд. 1.6) для отнесения КС к классу защищенности B2 и выше необходимо обеспечить защищенный канал связи между подсистемой защиты информации и пользователем при его входе в КС и аутентификации, инициатором создания которого может выступать только пользователь.

В соответствии с требованиями руководящих документов Гостехкомиссии России (см. подразд. 1.6) для автоматизированных систем, отнесенных к классу защищенности 1Б, должно быть обеспечено шифрование конфиденциальной информации с использованием аттестованных (сертифицированных) криптографических средств. Доступ субъектов к операциям шифрования и к соответствующим криптографическим ключам должен дополнительно контролироваться посредством подсистемы управления доступом. Для АС, отнесенных к классу защищенности 1А, дополнительно необходимо обеспечить шифрование информации, принадлежащей различным субъектам доступа (группам субъектов), на разных ключах.

4.9. Компьютерная стеганография и ее применение

Применение методов криптографии позволяет скрыть от непосвященных содержание конфиденциальной информации, но не

способно скрыть самого факта ее наличия или передачи. Методы *стеганографии* направлены на скрытие самого присутствия конфиденциальной информации. Подобно криптографии стеганография известна с глубокой древности, а ее примерами могут быть условные обозначения на рисунке, письмо с использованием симпатических чернил и т. п.

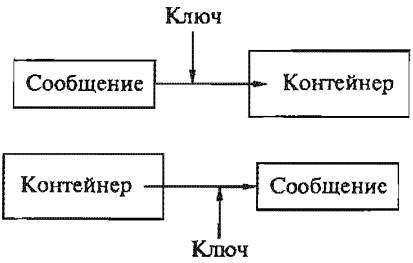

Рис. 4.11. Методы стеганографии

Применительно к стеганографии различают *сообщение* (объект, существование и содержание которого должно быть скрыто) и *контейнер* (объект, в котором скрывается сообщение). При помещении сообщения в контейнер может использоваться секретный ключ, определяющий порядок помещения сообщения в контейнер. Этот же ключ должен быть задан при извлечении сообщения из контейнера (рис. 4.11).

Большое развитие получили методы стеганографии в последние годы в связи с бурным развитием компьютерных технологий.

Принципы компьютерной стеганографии:

- обеспечение аутентичности и целостности файла-сообщения;
- открытость методов компьютерной стеганографии;
- сохранение основных свойств файла-контейнера после помещения в него сообщения (после этого файл-контейнер можно открывать, сжимать, восстанавливать без потери качества и изменения содержания информации в контейнере);
- сложность извлечения сообщения из файла контейнера при известности факта скрытия сообщения, но без знания ключа.

Основные задачи защиты информации, которые могут решаться с помощью методов компьютерной стеганографии:

- защита от несанкционированного доступа к конфиденциальной информации;
- преодоление систем сетевого мониторинга и управления сетевыми ресурсами (например, систем промышленного шпионажа, регистрирующих частоту обмена конфиденциальными сообщениями даже при отсутствии возможности их расшифрования);
- камуфлирование конфиденциального программного обеспечения (защита его от использования незарегистрированными пользователями путем скрытия в мультимедийных файлах);
- защита авторских прав создателей (владельцев) электронных документов путем нанесения на файлы с этими документами (фото, аудио и видеоматериалами) специальной метки (водяного знака), распознаваемого только специальным программным обеспечением.

Известны две основные группы методов компьютерной стеганографии. К *первой группе* методов компьютерной стеганографии относятся методы, использующие специальные свойства форматов электронных документов:

- зарезервированные для дальнейшего применения поля;
- специальное форматирование текстовых документов;
- неиспользуемые места дисковой памяти (например, последние байты и секторы последнего выделенного файлу кластера);
- имитирующие функции для генерации осмысленного текста файла-контейнера для скрываемого сообщения и др.

К недостаткам методов компьютерной стеганографии данной группы относится небольшой размер сообщения, которое может быть скрыто в контейнере.

Ко *второй группе* методов компьютерной стеганографии относятся методы, использующие естественную избыточность оцифрованных графических изображений, звука и видеоинформации. Эти методы обычно называют методами последнего значащего бита (Last Significant Bit, LSB). Например, полноцветные графические файлы в формате RGB кодируют каждую точку (пиксель) изображения тремя байтами для представления соответственно красной, зеленой и синей составляющих. Изменение каждого из трех младших битов (для хранения битов скрываемого сообщения) приведет к изменению цветовых характеристик данной точки изображения менее чем на 1 %, что абсолютно незаметно для человеческого глаза. Этот метод позволяет скрыть в графическом файле размером 800 килобайт сообщение размером до 100 килобайт.

Аналогично одна секунда оцифрованного звука с частотой дискретизации 44100 герц и уровнем отсчета 8 бит в стереорежиме позволяет по методу LSB скрыть сообщение размером до 10 килобайт.

Примерами известных программ компьютерной стеганографии являются Steganos (скрытие сообщений внутри графических, зву-

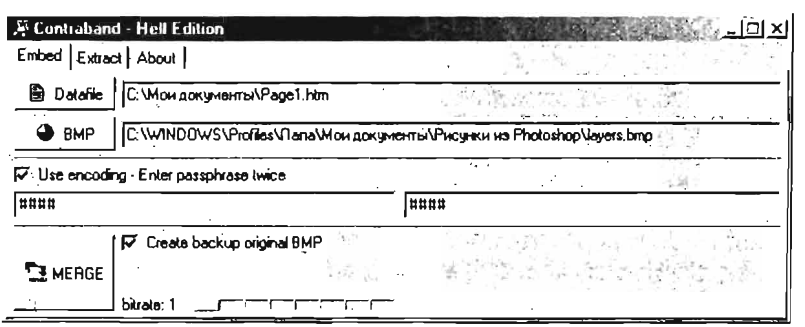

Рис. 4.12. Пример скрытия сообщения в файле-контейнере

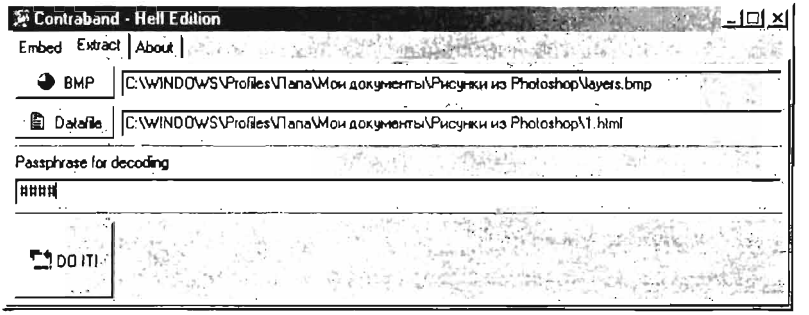

Рис. 4.13. Пример извлечения сообщения из файла-контейнера

ковых, текстовых файлов и файлов в формате HTML) и Contraband (скрытие сообщений внутри bmp-файлов). На рис. 4.12 и 4.13 приведены примеры скрытия и извлечения сообщения (файла в формате HTML) из файла-контейнера.

Возможно объединение методов криптографии и стеганографии, при котором сообщение предварительно зашифровывается перед помещением в контейнер.

Рассмотрим пример использования методов стеганографии для защиты программного обеспечения от его несанкционированного запуска. Использование приложения возможно только при его запуске с указанием ключевой фразы в качестве обязательного параметра командной строки. В самой программе ключевая фраза используется в качестве ключа шифрования магической строки (чтобы значение ключевой фразы не присутствовало в коде программы в явном виде). Полученное таким образом при запуске программы хеш-значение сравнивается с эталонным хеш-значением. По результатам этого сравнения и принимается решение о легальности запуска защищаемой программы.

Приведем пример фрагмента программы на языке C++, в котором происходит вычисление хеш-значения на основе полученного параметра командной строки (ключа шифрования перестановкой (см. подразд. 4.3) магической строки, задаваемой константой MAGICTEXT) и его сравнение с эталонным хеш-значением (константа HASHVALUE):

```
#include <string.h>
#include <stdlib.h>
#define MAXLEN 20 // максимальная длина ключевой фразы
...
// функция, используемая при сортировке символов ключа
// шифрования
int lt(const void *a, const void *b)
```

```
{ if(*(char*)a<*(char*)b) return -1;
else if(*(char*)a==*(char*)b) return 0;
else return 1; }
void main()
{ char tmp[MAXLEN], // упорядоченный ключ шифрования
key[MAXLEN], // ключевая фраза
res[MAXLEN], // результирующее хеш-значение (шифро
//текст)
str[MAXLEN] = MAGICTEXT; // магическая строка
/* проверка количества параметров командной строки при
попытке запуска приложения и копирование ключевой фра-
зы */
if(_argc>1) strncpy(key, _argv[1], MAXLEN);
else strcpy(key,"");
/* преобразование ключа шифрования (выравнивание его
длины с длиной магической строки) */
if(strlen(MAGICTEXT)<strlen(key))
strncpy(tmp, key, strlen(MAGICTEXT));
else strcpy(tmp, key);
while(strlen(tmp)<strlen(MAGICTEXT)) strncat(tmp," ",1);
char Key[MAXLEN]; // преобразованный ключ
strcpy(Key, tmp);
// сортировка символов ключа шифрования
qsort(tmp, strlen(tmp), sizeof(char), lt);
unsigned Perm[MAXLEN]; /* ключ шифрования для переста-
новки */
// цикл шифрования магической строки
for(unsigned i=0; i<strlen(MAGICTEXT); i++)
{ Perm[i]=strchr(tmp, Key[i])-tmp;
tmp[Perm[i]]='\1';
res[i]=str[Perm[i]]; }
res[strlen(MAGICTEXT)]='\0';
// сравнение полученного и эталонного хеш-значений
if(!strcmp(res, HASHVALUE))
// продолжение работы приложения при совпадении
...
/* завершение работы защищенного приложения при несанк-
ционированном запуске */
else exit(-1); }
```

Для скрытия процедуры санкционированного запуска приложения создается контейнер — документ в текстовом процессоре Microsoft Word с произвольным содержанием, но включающий в себя три макроса.

1. Играющий роль скрываемого сообщения макрос с произвольным именем для ввода пути к защищаемому приложению и

параметра его запуска (ключевой фразы), запуска программы, завершения работы текстового процессора Microsoft Word:

```
Sub CallProg()
'
' CallProg Макрос
' Макрос создан 26.03.2004 Павел Б. Хорев
'
s = InputBox("Параметры запуска", "Инициализация", "")
If s <> "" Then Shell s
Application. Quit
End Sub
```

Этот макрос с помощью команды меню Microsoft Word Сервис | Настройка | Клавиатура связывается с ключевой комбинацией клавиш на клавиатуре (например, Alt+Ctrl+P).

2. Макрос со стандартным именем для блокировки команды меню Microsoft Word Сервис | Настройка:

```
Sub ToolsCustomize()
'
' ToolsCustomize Макрос
' Настройка интерфейса пользователя Word (меню,
' клавиатура и " панели инструментов)
'
' Dialogs(wdDialogToolsCustomize).Show
End Sub
```

3. Макрос со стандартным именем для блокировки команды меню (и дублирующей ее кнопки панели управления) Сервис | Макрос | Макросы:

```
Sub ToolsMacro()
'
' ToolsMacro Макрос
' Запуск, создание, удаление или изменение макроса
'
' Dialogs(wdDialogToolsMacro).Show
End Sub
```

При открытии документа-контейнера неуполномоченным пользователем будет выдано сообщение о наличии в документе макросов, возможно содержащих вирусы (см. подразд. 6.3), с предложением выбора варианта дальнейшей работы с документом — с отключением макросов или без их отключения. Если неуполномоченный пользователь выберет вариант работы с документом без отключения макросов, то для осуществления несанкционированного запуска защищаемого приложения он должен подобрать:

- ключевую комбинацию клавиш для выполнения запускающего макроса (просмотр списка макросов документа и назначенных комбинаций клавиш будет заблокирован);

• правильный путь к защищаемому приложению и ключевую фразу, необходимую для его запуска.

Если неуполномоченный пользователь откроет документ-контейнер с отключением содержащихся в нем макросов, то он также не сможет просмотреть их список и назначенные для их выполнения комбинации клавиш.

В качестве средства дополнительной защиты от несанкционированного запуска приложения можно использовать его шифрование с помощью, например, криптографического интерфейса приложений Windows (см. гл. 5). В этом случае макрос, скрываемый в документе-контейнере, будет вызывать программу для расшифрования и запуска защищаемого приложения, а в качестве источника для генерации ключа шифрования будет взята введенная пользователем ключевая фраза.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что называется вычетом целого числа по некоторому модулю?
2. Почему операции над вычетами находят широкое применение в криптографии?
3. В чем разница между симметричными и асимметричными криптографическими системами?
4. Какие основные способы применяются при создании алгоритмов симметричной криптографии?
5. В чем разница между потоковыми и блочными шифрами?
6. Какой шифр может считаться идеальным (по К. Шеннону)?
7. Какие симметричные криптосистемы используются сегодня?
8. В каких режимах может использоваться криптосистема DES?
9. Какие из режимов DES могут использоваться для проверки аутентичности и целостности шифротекста?
10. В каких режимах может использоваться криптосистема ГОСТ 28147—89? Как в ней обеспечивается аутентичность и целостность шифротекстов?
11. Что лежит в основе асимметричной криптографии?
12. В чем особенности и основные сферы применения асимметричных криптосистем?
13. На чем основана криптостойкость систем RSA и Эль-Гамаля?
14. Что такое электронная цифровая подпись, как она получается и проверяется?
15. Какова роль в системах ЭЦП функций хеширования?
16. Какую роль исполняют удостоверяющие центры? Что такое сертификат открытого ключа?
17. Какие основные возможности имеет программа PGP? Что лежит в ее основе?
18. В чем заключаются принципы и методы компьютерной стеганографии?
19. Для решения каких задач применяются методы компьютерной стеганографии?
20. В чем разница между криптографией и стеганографией?

ГЛАВА 5

КРИПТОГРАФИЧЕСКИЙ ИНТЕРФЕЙС ПРИЛОЖЕНИЙ ОПЕРАЦИОННОЙ СИСТЕМЫ WINDOWS (CRYPTOAPI)

5.1. Принципы построения и использования CryptoAPI

Криптографический интерфейс приложений операционной системы Windows представляет собой набор констант, типов данных и функций, предназначенных для выполнения операций шифрования, расшифрования, получения и проверки ЭЦП, генерации, хранения и распределения ключей шифрования. Эти услуги для приложений предоставляют провайдеры криптографического обслуживания (Cryptographic Service Provider, CSP) — динамически компонуемые библиотеки (DLL), экспортирующие единый набор объектов, определяемый интерфейсом CryptoAPI.

Взаимодействие между приложением и CSP строится на основе следующих принципов:

- приложение не имеет прямого доступа к изготовлению и хранению ключей шифрования (нет риска их потери из-за ошибок в приложении);
- приложение не определяет детали выполнения криптографических операций, а лишь указывает на требуемые от CSP действия (например, зашифровать по заданному алгоритму данные и получить для них ЭЦП);
- приложение не обрабатывает данных, по которым проводится аутентификация пользователя, а предоставляет это CSP, который имеет лучшие возможности для проверки подлинности пользователя (например, может использовать аутентификацию на основе биометрических признаков или ключевой информации, размещенной на смарт-картах или других элементах аппаратного обеспечения); в этом случае из-за ошибок в приложении не может произойти утечки конфиденциальной информации о пользователе (например, о его пароле).

На рис. 5.1 приведена архитектура криптографической подсистемы Windows. Вызовы функций CryptoAPI обрабатываются моду-

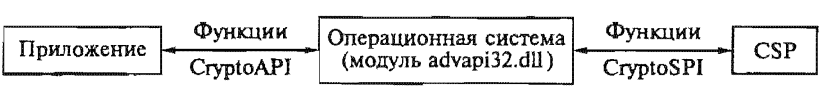

Рис. 5.1. Криптографическая подсистема Windows

лем операционной системы advapi32.dll, который преобразует их в вызовы функций интерфейса провайдера криптографического обслуживания (Cryptographic Service Provider Interface, CryptoSPI). Для обеспечения аутентичности и подлинности CSP он снабжается ЭЦП, которая периодически проверяется операционной системой в ходе сеанса работы пользователя. Получение ЭЦП производится корпорацией Microsoft на основе хеш-значения CSP, представленного изготовителем этой библиотеки.

Каждый CSP характеризуется своим присвоенным производителем именем (строкой символов) и типом, определяющим поддерживаемые этим провайдером криптографические алгоритмы (см. гл. 4). В табл. 5.1 приведены некоторые предопределенные типы криптопровайдеров.

К основным атрибутам CSP относятся:

- обязательно поддерживаемый алгоритм ЭЦП (всегда единственный);
- длина ключей асимметричного шифрования;
- формат ЭЦП;

Таблица 5.1

Символическая константа типа (ее численное значение)	Алгоритм обмена сеансовыми ключами	Алгоритм ЭЦП	Алгоритмы симметричного шифрования	Функции хеширования
PROV_RSA_FULL (1)	RSA	RSA	RC2, RC4, DES, 3-DES	MD2, MD4, MD5, SHA
PROV_RSA_SIG (2)	—	RSA	—	MD5, SHA
PROV_DSS (3)	—	DSS	—	SHA
PROV_FORTEZZA (4)	Диффи-Хеллмана	DSS	Skipjack	SHA
PROV_MS_EXCHANGE (5)	RSA	RSA	CAST	MD5
PROV_SSL (6)	RSA	RSA	Разные	Разные
PROV_RSA_SCHANNEL (12)	RSA	RSA	RC4, DES, 3-DES	MD5, SHA
PROV_DSS_DH (13)	Диффи-Хеллмана	DSS	CYLINK MEK	MD5, SHA
PROV_DH_SCHANNEL (18)	Диффи-Хеллмана	DSS	DES, 3-DES	MD5, SHA

• форматы блоков, в которых открытый и (возможно) секретный ключи асимметричного шифрования экспортируются из CSP;
• возможно поддерживаемый алгоритм обмена сеансовыми ключами симметричного шифрования (всегда единственный);
• возможно поддерживаемые алгоритмы симметричного шифрования (конкретный CSP может поддерживать только часть таких алгоритмов, определенных для соответствующего типа криптопровайдера);
• схема генерации сеансового ключа из хеш-значения;
• длины сеансовых ключей (в зависимости от алгоритма);
• формат блока сеансового ключа при его экспорте из CSP;
• режимы симметричного шифрования, принятые по умолчанию (например, режим CBC).

Информация об установленных на компьютере криптопровайдерах содержится в реестре Windows в разделе HKEY_LOCAL_MACHINE\Software\Microsoft\Cryptography\Defaults. В подразделе Provider (рис. 5.2) размещается информация о пути к библиотеке, ЭЦП и типе каждого CSP (по его имени), а в подразделе Provider Types — сведения о полных именах криптопровайдера и его типа.

Для каждого зарегистрированного у него пользователя или конкретного приложения CSP хранит контейнер ключей асимметричного шифрования, который может включать в себя две пары ключей — открытый и секретный ключи для обмена сеансовыми ключами, а также открытый и секретный ключи для ЭЦП. Ключи симметричного шифрования (сеансовые ключи) не сохраняются CSP и об их сохранении (или правильной повторной генерации) должно позаботиться приложение.

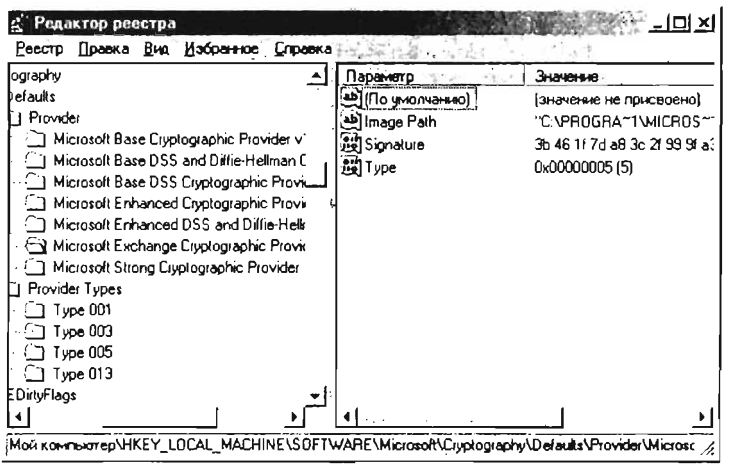

Рис. 5.2. Информация о криптопровайдерах в реестре Windows

Существуют две версии CryptoAPI: 1.0 и 2.0. В CryptoAPI 2.0 введены дополнительные функции для поддержки инфраструктуры открытых ключей (см. подразд. 4.8). Но поскольку дополнительные функции CryptoAPI 2.0 используют вызовы функций CryptoAPI 1.0, в данном учебном пособии мы ограничимся изучением методов использования функций CryptoAPI 1.0, которые приведены в табл. 5.2.

Для возможности доступа пользователя или его приложения к услугам, предоставляемым конкретным CSP, необходимо создать для этого субъекта пустой контейнер ключей (иначе говоря, зарегистрировать его у этого криптопровайдера). После этого в начале каждого сеанса работы с CSP приложение должно получить у

Таблица 5.2

Название функции	Краткое описание функции
CryptAcquireContext CryptReleaseContext	Получение и освобождение дескриптора CSP
CryptGetProvParam CryptSetProvParam	Получение и установка атрибутов CSP
CryptSetProvider	Установка используемого по умолчанию CSP
CryptGenKey CryptDestroyKey	Создание ключей шифрования и освобождение их дескрипторов
CryptDeriveKey	Создание сеансового ключа из хеш-значения
CryptExportKey CryptImportKey	Экспорт и импорт ключа шифрования в CSP
CryptGetUserKey	Получение дескриптора открытого ключа
CryptGetKeyParam CryptSetKeyParam	Получение и установка параметров ключа шифрования
CryptEncrypt CryptDecrypt	Симметричное шифрование и расшифрование данных
CryptCreateHash CryptDestroyHash	Создание пустого хеш-значения и освобождение его дескриптора
CryptHashData CryptHashSessionKey	Хеширование (добавление к хеш-значению) произвольных данных и сеансового ключа
CryptGetHashParam CryptSetHashParam	Получение и установка параметров хеш-значения
CryptSignHash CryptVerifySignature	Получение и проверка ЭЦП
CryptGenRandom	Генерация случайного значения

криптопровайдера дескриптор контейнера ключей соответствующего субъекта. Создание контейнера ключей и получение его дескриптора выполняются с помощью функции CryptoAPI CryptAcquireContext:

```
BOOL CryptAcquireContext(HCRYPTPROV *phProv, LPCSTR pszContainer, LPCSTR pszProvider, DWORD dwProvType, DWORD dwFlags); /* получение в *phProv дескриптора существующего контейнера ключей pszContainer у криптопровайдера с именем pszProvider и типом dwProvType (dwFlags=0) или (если функция CryptAcquireContext возвращает FALSE с кодом ошибки NTE_BAD_ KEYSET) регистрация нового субъекта в криптопровайдере (dwFlags=CRYPT_NEWKEYSET) */
```

В защищенных версиях операционной системы Windows (см. подразд. 3.3) для создания нового контейнера ключей приложение должно иметь полномочия администратора (требуется запись в раздел реестра HKEY_LOCAL_MACHINE). Полученный дескриптор контейнера ключей должен указываться приложением при вызове других функций CryptoAPI. Для освобождения дескриптора контейнера ключей, полученного с помощью функции CryptAcquireContext, предназначена следующая функция CryptoAPI:

```
BOOL CryptReleaseContext(HCRYPTPROV hProv, DWORD dwFlags); /* освобождение дескриптора контейнера ключей hProv (dwFlags=0) */
```

Вместе с любой версией Windows, начиная с Windows 95 OSR2, поставляется криптопровайдер «Microsoft Base Cryptographic Provider v1.0» (типа PROV_RSA_FULL). Начиная с версий Windows 2000/ME, в состав операционных систем Windows включаются также криптопровайдеры «Microsoft Enhanced Cryptographic Provider v1.0» и «Microsoft Strong Cryptographic Provider» (того же типа). Криптопровайдер «Microsoft Enhanced Cryptographic Provider v1.0» может быть установлен на компьютерах под управлением операционных систем Windows 9x путем загрузки с веб-сервера Microsoft.

Криптопровайдер «Microsoft Base Cryptographic Provider v1.0» использует ключи асимметричного шифрования длиной 512 бит и ключи симметричного шифрования по алгоритмам RC2 и RC4 длиной 40 бит, что недостаточно на современном уровне развития компьютерных технологий. Криптопровайдеры «Microsoft Enhanced Cryptographic Provider v1.0» и «Microsoft Strong Cryptographic Provider» используют ключи асимметричного шифрования длиной 1024 бита и ключи симметричного шифрования длиной 56 (DES), 128 (RC2 и RC4) и 168 (3-DES) бит.

Существуют криптопровайдеры, в которых реализованы алгоритмы симметричного шифрования и ЭЦП, соответствующие

российским стандартам в области информационной безопасности (ГОСТ 28147—89, ГОСТ Р 34.10—94, ГОСТ Р 34.10—2001 и ГОСТ Р 34.11—94). Примером такого криптопровайдера является средство криптографической защиты информации «КриптоПро CSP».

Если при вызове функции CryptAcquireContext вместо имени криптопровайдера указывается NULL, то предполагается получение дескриптора контейнера ключей у криптопровайдера по умолчанию («Microsoft Base Cryptographic Provider v1.0» или «Microsoft Enhanced Cryptographic Provider v1.0», в зависимости от версии Windows).

Если при вызове этой функции вместо имени контейнера ключей задается NULL, то имя контейнера ключей совпадает с логическим именем пользователя, начавшего данный сеанс работы с операционной системой.

В параметре dwFlags при вызове функции CryptAcquireContext могут также использоваться флаги CRYPT_MACHINE_KEYSET (контейнер ключей, создаваемый и используемый встроенными в Windows криптопровайдерами, хранится не в профиле пользователя в реестре, а в разделе реестра HKEY_LOCAL_MACHINE), CRYPT_DELETEKEYSET (удаление контейнера ключей), CRYPT_VERIFYCONTEXT (дескриптор контейнера ключей получается приложением только для проверки ЭЦП, поэтому доступ к секретным ключам приложение не имеет) и др.

Дополнительно установленные криптопровайдеры могут хранить контейнеры ключей на защищенных от несанкционированного чтения носителях (например, на смарт-картах) или в реестре в зашифрованном виде (в этом случае при выполнении функции CryptAcquireContext может потребоваться ввод аутентифицирующей пользователя информации: PIN-кода или специального пароля).

Определения констант, типов данных и прототипов функций CryptoAPI содержатся в файле wincrypt.h, поставляемом вместе с любой системой программирования на языке C++ для Windows. При разработке приложений, использующих CryptoAPI, на языке Object Pascal в системе программирования Borland Delphi потребуется написать специальный интерфейсный модуль, пример которого приведен далее (включены только наиболее необходимые константы и функции):

```
unit Wincrypt;
interface
uses Windows;
const
// коды криптографических алгоритмов
CALG_MD2=$8001;
CALG_MD4=$8002;
```

```
CALG_MD5=$8003;
CALG_SHA=$8004;
CALG_DES=$6601;
CALG_RC2=$6602;
CALG_3DES=$6603;
CALG_RC4=$6801;
// константы для функции CryptAcquireContext
PROV_RSA_FULL=1;
CRYPT_NEW_KEYSET=$8;
CRYPT_MACHINE_KEYSET=$20;
CRYPT_DELETEKEYSET=$10;
CRYPT_VERIFYCONTEXT=$F0000000;
// константы для функции создания ключа
CRYPT_EXPORTABLE=$1;
CRYPT_CREATE_SALT=$4;
AT_KEYEXCHANGE=1;
AT_SIGNATURE=2;
// константы для функции установки параметров ключа
KP_MODE=4;
// константы типов блоков данных для экспорта ключей
SIMPLEBLOB=1;
PUBLICKEYBLOB=6;
PRIVATEKEYBLOB=7;
// коды режимов шифрования
CRYPT_MODE_CBC=1;
CRYPT_MODE_ECB=2;
CRYPT_MODE_OFB=3;
CRYPT_MODE_CFB=4;
type
ALG_ID=Cardinal;
// функции CryptoAPI
function CryptAcquireContext(var hProv: THandle;
pszContainer, pszProvider: PChar; dwProvType, dwFlags:
Longint): Longbool; stdcall; external 'advapi32' name
'CryptAcquireContextA';
function CryptCreateHash(hProv: THandle; Algid: ALG_ID;
hKey: THandle; dwFlags: Longint; var hHash:
THandle):Longbool; stdcall; external 'advapi32';
function CryptHashData(hHash: THandle; pbData: Pointer;
dwDataLen, dwFlags: Longint): Longbool; stdcall; external
'advapi32';
function CryptDestroyHash(hHash: THandle):Longbool;
stdcall; external 'advapi32';
function CryptDeriveKey(hProv: THandle; Algid: ALG_ID;
hBaseData: THandle; dwFlags: Longint; var hKey:
THandle):Longbool; stdcall; external 'advapi32';
```

```
function CryptDestroyKey(hKey: THandle):Longbool;
stdcall; external 'advapi32';
function CryptReleaseContext(hProv: THandle; dwFlags:
Longint):Longbool; stdcall; external 'advapi32';
function CryptEncrypt(hKey, hHash: THandle; Final:
Longbool; dwFlags: Longint; pbData: Pointer; var
dwDataLen: Longint; dwBufLen: Longint): Longbool;
stdcall; external 'advapi32';
function CryptDecrypt(hKey, hHash: THandle; Final:
Longbool; dwFlags: Longint; pbData: Pointer; var
dwDataLen: Longint): Longbool; stdcall; external
'advapi32';
function CryptSetKeyParam(hKey: THandle; dwParam:
Longint; pbData: Pointer; dwFlags: Longint): Longbool;
stdcall; external 'advapi32';
function CryptGenKey(hProv: THandle; Algid: ALG_ID;
dwFlags: Longint; var phKey: THandle): Longbool; stdcall;
external 'advapi32';
function CryptGetUserKey(hProv: THandle; dwKeySpec:
Longint; var phUserKey: THandle): Longbool; stdcall;
external 'advapi32';
function CryptSignHash(hHash: THandle; dwKeySpec:
Longint; sDescription: PChar; dwFlags: Longint;
pbSignature: Pointer; var pdwSigLen: Longint): Longbool;
stdcall; external 'advapi32' name 'CryptSignHashA';
function CryptVerifySignature(hHash: THandle;
pbSignature: Pointer; dwSigLen: Longint; hPubKey:
THandle; sDescription: PChar; dwFlags: Longint):
Longbool; stdcall; external 'advapi32' name
'CryptVerifySignatureA';
implementation
end.
```

5.2. Создание и передача криптографических ключей с помощью функций CryptoAPI

Для создания пар ключей асимметричного шифрования должна использоваться следующая функция CryptoAPI:

```
BOOL CryptGenKey(HCRYPTPROV hProv, ALG_ID Algid, DWORD
dwFlags, HCRYPTKEY *phKey); /* создание в контейнере
ключей с дескриптором hProv пары ключей ЭЦП
(Algid=AT_SIGNATURE) или обмена сеансовыми ключами
(Algid=AT_KEYEXCHANGE) и запись дескриптора открытого
ключа созданной пары в *phKey; если секретный ключ
```

созданной пары должен иметь возможность экспорта из CSP, то dwFlags=CRYPT_EXPORTABLE (открытые ключи всегда являются экспортируемыми) */

Получение дескриптора открытого ключа *phUserKey из соответствующего контейнера ключей hProv возможно с помощью следующей функции:

```
BOOL CryptGetUserKey(HCRYPTPROV hProv, DWORD dwKeySpec,
HCRYPTKEY *phUserKey); /* параметр dwKeySpec определяет тип запрашиваемого ключа – обмена (AT_KEYEXCHANGE)
или ЭЦП (AT_SIGNATURE) */
```

Если функция CryptGetUserKey возвращает FALSE с кодом ошибки NTE_NO_KEY, то запрашиваемый открытый ключ в контейнере ключей криптопровайдера с дескриптором hProv не существует и соответствующая пара ключей асимметричного шифрования (для обмена сеансовыми ключами или ЭЦП) должна быть создана с помощью функции CryptGenKey.

Для создания сеансового ключа также может использоваться функция CryptGenKey со значением параметра Algid, равным коду алгоритма используемого в приложении симметричного шифрования (CALG_DES, CALG_RC2, CALG_3DES, CALG_RC4 или др.). Значение параметра dwFlags может быть объединением следующих флагов: CRYPT_CREATE_SALT (использование случайного значения для модификации ключа шифрования, см. подразд. 4.3), CRYPT_EXPORTABLE (сеансовый ключ может экспортироваться из CSP) и др.

Экспорт ключа шифрования заключается в его получении внутри специального блока данных (блоба, от Bit Large OBject, BLOB) в открытом (тип блоба PUBLICKEYBLOB для экспорта открытых ключей асимметричного шифрования) или зашифрованном (типы блоба SIMPLEBLOB для экспорта сеансовых ключей или PRIVATEKEYBLOB для экспорта секретного и открытого ключей асимметричного шифрования). Ключом шифрования секретного ключа при его экспорте должен быть специальный сеансовый ключ, а ключом шифрования экспортируемого сеансового ключа — открытый ключ получателя блоба с сеансовым ключом. Для экспорта и импорта ключей шифрования предназначены следующие функции CryptoAPI:

```
BOOL CryptExportKey(HCRYPTKEY hKey, HCRYPTKEY hExpKey,
DWORD dwBlobType, DWORD dwFlags, BYTE *pbData, DWORD
*pdwDataLen); /* экспорт ключа шифрования с дескриптором hKey, зашифрованного с помощью ключа с дескриптором hExpKey, в блобе *pbData длиной *pdwDataLen типа
dwBlobType; при успешном завершении в *pdwDataLen помещается фактическая длина блоба */
```

```
BOOL CryptImportKey(HCRYPTPROV hProv, CONST BYTE *pbData,
DWORD dwDataLen, HCRYPTKEY hPubKey, DWORD dwFlags,
HCRYPTKEY *phKey); /* расшифрование ключа шифрования
из блоба *pbData длиной dwDataLen с помощью ключа с
дескриптором hPubKey и импорт восстановленного ключа в
контейнер ключей hProv; при успешном завершении в *phKey
помещается дескриптор импортированного ключа */
```

Расшифрование импортируемого сеансового ключа выполняется с помощью секретного ключа получателя блоба, а импортируемого секретного ключа — с помощью специального сеансового ключа.

Другим способом создания сеансового ключа симметричного шифрования является его генерация из хеш-значения введенной пользователем ключевой фразы. В этом случае используются следующие функции CryptoAPI:

```
BOOL CryptCreateHash(HCRYPTPROV hProv, ALG_ID Algid,
HCRYPTKEY hKey, DWORD dwFlags, HCRYPTHASH *phHash); /*
создание в криптопровайдере с дескриптором hProv пустого
хеш-значения с дескриптором *phHash (в Algid должен
содержаться код алгоритма хеширования — CALG_MD2,
CALG_MD4, CALG_MD5 или CALG_SHA, параметры hKey и dwFlags
не используются и должны быть равны нулю */
BOOL CryptHashData(HCRYPTHASH hHash, CONST BYTE *pbData,
DWORD dwDataLen, DWORD dwFlags); /* добавление к хеш-
значению с дескриптором hHash данных (например, ключевой
фразы) из буфера *pbData длиной dwDataLen; параметр
dwFlags не используется и должен быть равен нулю */
BOOL CryptDeriveKey(HCRYPTPROV hProv, ALG_ID Algid,
HCRYPTHASH hBaseData, DWORD dwFlags, HCRYPTKEY *phKey);/*
создание сеансового ключа с дескриптором *phKey из хеш-
значения с дескриптором hBaseData; значения параметров
Algid (код алгоритма шифрования) и dwFlags (флаги) могут
быть такими же, как при вызове функции CryptGenKey
*/
```

После вызова функции CryptDeriveKey в используемое ею хеш-значение не могут быть добавлены новые данные. После завершения работы с хеш-значениями и ключами шифрования их дескрипторы должны быть разрушены (занимаемая ими оперативная память освобождена) в приложении с помощью следующих функций:

```
BOOL CryptDestroyKey(HCRYPTKEY hKey); /* разрушение
дескриптора ключа шифрования hKey */
BOOL CryptDestroyHash(HCRYPTHASH hHash); /* разруше-
ние дескриптора хеш-значения hHash */
```

Рассмотрим пример создания сеансового ключа (на основе вводимой пользователем ключевой фразы) и пар ключей асимметричного шифрования:

```
#include <windows.h>
#include <conio.h>
#include <iostream.h>
#define _WIN32_WINNT 0x0400
#include <wincrypt.h>
/* функция преобразования сообщений на русском языке
для вывода в консольном приложении Windows */
char * MessageOut(char * str)
{ static char OutString[MAX_PATH]; /* буфер для преоб-
разованного сообщения */
CharToOem(str, OutString);
return OutString; }
void main()
{HCRYPTPROV hProv = 0; // дескриптор CSP
BYTE pbData[1024]; // буфер для получаемых данных
DWORD dwDataLen; // длина получаемых данных
/* получение дескриптора провайдера по умолчанию для
текущего пользователя (при необходимости создание но-
вого контейнера ключей) */
if(!CryptAcquireContext(&hProv, NULL, NULL,
PROV_RSA_FULL,0))
if((unsigned)GetLastError()==NTE_BAD_KEYSET)
if(!CryptAcquireContext(&hProv, NULL, NULL,
PROV_RSA_FULL, CRYPT_NEWKEYSET)) {
cout<<MessageOut("Ошибка при ");
cout<<MessageOut("CryptAcquireContext (создании"\
"нового "\ "контейнера ключей)!\n");
getch();
return; }
else;
else {
cout<<MessageOut("Ошибка при CryptAcquireContext ")
<<hex<<
GetLastError()<<dec;
cout<<MessageOut(" (получении дескриптора"\
"существующего контейнера)!\n");
getch();
return; }
// получение хеш-значения для ключевой фразы
cout<<MessageOut("Введите ключевую фразу для шифрова-
ния \n");
char KeyFrase[MAX_PATH]; /* ключевая фраза для генера-
ции сеансового ключа */
```

```
cin>>KeyFrase;
HCRYPTHASH hash=0; // дескриптор хеш-значения
if(!CryptCreateHash(hProv, CALG_SHA, 0, 0, &hash)) {
cout<<MessageOut("Ошибка при CryptCreateHash!\n");
getch();
return; }
if(!CryptHashData(hash, (BYTE*)KeyFrase, strlen
(KeyFrase),0)) {
cout<<MessageOut("Ошибка при CryptHashData!\n");
getch();
return; }
// создание сеансового ключа
HCRYPTKEY hKey=0;
if(!CryptDeriveKey(hProv, CALG_RC4, hash, CRYPT_
EXPORTABLE | CRYPT_CREATE_SALT, &hKey)) {
cout<<MessageOut("Ошибка при CryptDeriveKey!\n");
getch();
return; }
// разрушение хеш-значения
if(hash) CryptDestroyHash(hash);
/* получение (создание при первом обращении) пары клю-
чей обмена */
HCRYPTKEY hExpKey=0; /* дескриптор открытого ключа
обмена */
HCRYPTKEY hSignKey=0; /* дескриптор открытого ключа
ЭЦП */
if(!CryptGetUserKey(hProv, AT_KEYEXCHANGE, &hExpKey))
if((unsigned)GetLastError()==NTE_NO_KEY)
if(!CryptGenKey(hProv, AT_KEYEXCHANGE, 0, &hExpKey)) {
cout<<MessageOut("Ошибка при создании ключей"\
"обмена через CryptGenKey!\n");
getch();
return; }
else;
else {
cout<<MessageOut("Ошибка при получении ключей"\
"обмена через CryptGetUserKey!\n");
getch();
return; }
/* получение (создание при первом обращении) пары клю-
чей ЭЦП */
if(!CryptGetUserKey(hProv, AT_SIGNATURE, &hSignKey))
if((unsigned)GetLastError()==NTE_NO_KEY)
if(!CryptGenKey(hProv, AT_SIGNATURE, 0, &hSignKey)) {
cout<<MessageOut("Ошибка при создании ключей"\
подписи через CryptGenKey!\n");
```

```
getch();
return; }
else;
else {
cout<<MessageOut("Ошибка при получении ключей"\
"подписи через CryptGetUserKey!\n");
getch();
return; }
/* выполнение криптографических операций (см. подразд.
5.3 и 5.4)*/
...
// разрушение сеансового ключа
if(hKey) CryptDestroyKey(hKey);
// разрушение дескрипторов открытых ключей
if(hExpKey) CryptDestroyKey(hExpKey);
if(hSignKey) CryptDestroyKey(hSignKey);
// освобождение CSP
if(hProv) CryptReleaseContext(hProv, 0);
return; }
```

Для организации обмена открытыми ключами асимметричного шифрования возможны как использование удостоверяющих центров (см. подразд. 4.8), так и передача открытых ключей напрямую между пользователями:

1) отправитель экспортирует из криптопровайдера свой открытый ключ в форме блоба (типа PUBLICKEYBLOB) с помощью функции CryptExportKey;

2) блоб посылается получателю по защищенному каналу связи (например, в зашифрованном виде с помощью сеансового ключа, генерируемого из специальной ключевой фразы, или в открытом виде с возможной конвертацией в текстовый формат);

3) получатель блоба импортирует его в свой криптопровайдер с помощью функции CryptImportKey.

Для передачи сеансового ключа также могут применяться два способа. *Первый способ*: отправитель создает случайный сеансовый ключ (с помощью функции CryptGenKey), зашифровывает его при помощи ранее полученного открытого ключа получателя (используется функция CryptExportKey) в виде блоба типа SIMPLEBLOB, отправляет блоб получателю, который расшифровывает сеансовый ключ с помощью своего секретного ключа обмена (используется функция CryptImportKey). Для снижения риска повторной посылки нарушителем перехваченного им ранее блоба с сеансовым ключом и зашифрованного с помощью этого ключа сообщения можно использовать отметки времени (аналогично протоколу Kerberos, см. подразд. 2.5) или последовательную нумерацию сообщений на основе случайного начального значения.

Второй способ передачи сеансового ключа заключается в использовании специального трехфазного протокола (предполагается, что отправитель A и получатель B имеют открытые ключи обмена PKA, PKB и логические имена ID_A, ID_B друг друга).

1. Первая фаза протокола:

a) A: генерация случайного сеансового ключа k_A (CryptGenKey); шифрование его с помощью открытого ключа B E_{PKB} (k_A) (CryptExportKey);

b) A→B: E_{PKB} (k_A);

c) B: расшифрование блоба с сеансовым ключом $k_A=D_{SKB}(E_{PKB}$ (k_A)) (CryptImportKey).

2. Вторая фаза протокола:

a) B: генерация случайного сеансового ключа k_B (CryptGenKey); шифрование его с помощью открытого ключа A E_{PKA} (k_B) (CryptExportKey);

b) B→A: E_{PKA} (k_B);

c) B: вычисление хеш-значения $H_{1B}=H(k_A, ID_B, k_B, ID_A$, «Фаза 2») (CryptCreateHash, CryptHashSessionKey, CryptHashData, CryptGetHashParam);

d) B→A: H_{1B};

e) A: расшифрование блоба с сеансовым ключом $k_B=D_{SKA}(E_{PKA}$ (k_B)) (CryptImportKey); вычисление $H_{1A}=H(k_A, ID_B, k_B, ID_A$, «Фаза 2») (CryptCreateHash, CryptHashSessionKey, CryptHashData, CryptGetHashParam); сравнение H_{1B} и H_{1A}; при несовпадении хеш-значений (получатель не является подлинным или произошло искажение передаваемых по сети сообщений третьим лицом) выполнение протокола прекращается, а сеанс связи завершается.

3. Третья фаза протокола:

a) A: вычисление хеш-значения $H_{2A}=H(k_B, ID_A, ID_B$, «Фаза 3») (CryptCreateHash, CryptHashSessionKey, CryptHashData, CryptGetHashParam);

b) A→B: H_{2A};

c) B: вычисление хеш-значения $H_{2B}=H(k_B, ID_A, ID_B$, «Фаза 3») (CryptCreateHash, CryptHashSessionKey, CryptHashData, CryptGetHashParam); сравнение H_{2A} и H_{2B}; при совпадении хеш-значений сеансовые ключи k_A и k_B могут использоваться для обмена зашифрованными с их помощью сообщениями.

При выполнении данного трехфазного протокола потребуется использование еще двух функций CryptoAPI:

```
BOOL CryptHashSessionKey(HCRYPTHASH hHash, HCRYPTKEY
hKey, DWORD dwFlags); /* добавление в хеш-значение с
дескриптором hHash сеансового ключа шифрования с дес-
криптором hKey; значение параметра dwFlags не исполь-
зуется и должно быть равно нулю */
```

```
BOOL CryptGetHashParam(HCRYPTHASH hHash, DWORD dwParam,
BYTE *pbData, DWORD *pdwDataLen, DWORD dwFlags); /*
получение в буфере *pbData длиной *pdwDataLen хеш-
значения с дескриптором hHash; после завершения в
*pdwDataLen помещается действительный размер хеш-зна-
чения; значение параметра dwFlags не используется и
должно быть равно нулю;dwParam=HP_HASHVAL */
```

5.3. Использование функций CryptoAPI для шифрования и расшифрования данных

После получения приложением сеансового ключа (случайного или сгенерированного из ключевой фразы, см. подразд. 5.2) он может быть использован для шифрования данных с помощью функции CryptEncrypt:

```
BOOL CryptEncrypt(HCRYPTKEY hKey, HCRYPTHASH hHash,
BOOL Final, DWORD dwFlags, BYTE *pbData, DWORD
*pdwDataLen, DWORD dwBufLen); /* шифрование на сеансо-
вом ключе с дескриптором hKey порции данных из буфера
*pbData длиной dwBufLen (*pdwDataLen – длина порции
данных, после выполнения функции в эту переменную за-
писывается фактическая длина зашифрованных данных);
hHash=0, dwFlags=0, Final – признак последней порции
шифруемых данных */
```

Для расшифрования зашифрованных с помощью сеансового ключа данных применяется функция CryptDecrypt:

```
BOOL CryptDecrypt(HCRYPTKEY hKey, HCRYPTHASH hHash,
BOOL Final, DWORD dwFlags, BYTE *pbData, DWORD
*pdwDataLen); /* расшифрование на сеансовом ключе с
дескриптором hKey порции данных из буфера *pbData
(*pdwDataLen – длина порции данных, после выполнения
функции в эту переменную записывается фактическая дли-
на расшифрованных данных); hHash=0, dwFlags=0, Final –
признак последней порции расшифровываемых данных */
```

Для изменения режима симметричного шифрования (ECB, CBC, CFB или OFB, см. подразд. 4.4) используется функция CryptSetKeyParam:

```
BOOL CryptSetKeyParam(HCRYPTKEY hKey, DWORD dwParam,
BYTE *pbData, DWORD dwFlags); /* установка режима шиф-
рования для сеансового ключа с дескриптором hKey:
dwParam=KP_MODE, pbData указывает на переменную типа
```

```
unsigned long, в которой записан код устанавливаемого
режима (CRYPT_MODE_ECB, CRYPT_MODE_CBC, CRYPT_MODE_ CFB
или CRYPT_MODE_OFB), dwFlags=0 */
```

По умолчанию режим шифрования устанавливается в CRYPT_MODE_ CBC. Функция CryptSetKeyParam может также применяться и для установки других параметров шифрования:

- использование вектора инициализации (синхропосылки, см. подразд. 4.4) — dwParam=KP_IV и буфер *pbData содержит значение вектора инициализации;
- использование случайных значений (salt values) для модификации ключа шифрования (см. подразд. 4.4) — dwParam=KP_SALT и буфер *pbData содержит случайное значение;
- размер порции данных для режимов шифрования с обратной связью CFB или OFB (см. подразд. 4.4) — dwParam=KP_MODE_BITS и буфер *pbData содержит размер порции данных (для встроенных в Windows криптопровайдеров Microsoft это значение по умолчанию равно 8);
- дополнительные ограничения на использование сеансового ключа — dwParam=KP_PERMISSIONS и буфер *pbData содержит комбинацию флагов CRYPT_ENCRYPT (разрешено шифрование на этом ключе), CRYPT_DECRYPT (разрешено расшифрование с помощью этого ключа), CRYPT_EXPORT (ключ может экспортироваться из CSP), CRYPT_READ (параметры ключа могут быть прочитаны), CRYPT_WRITE (параметры ключа могут быть изменены), CRYPT_MAC (ключ может использоваться для вычисления кода аутентификации сообщений, см. подразд. 4.4).

Приведем пример использования функций CryptoAPI для шифрования и расшифрования файла, чье имя содержится в константе SECFILE (константа TMPFILE содержит имя файла с зашифрованными данными):

```
#include <fstream.h>
BYTE Buf[128]; /* буфер для ввода-вывода данных, длина
которого должна быть кратна длине блока при использовании блочного шифрования (обычно 64 бита, см. подразд. 4.3) */
fstream TmpFile1,TmpFile2; // файловые переменные
DWORD Len; /* длина порции зашифрованных (расшифрованных) данных */
/* получение дескриптора контейнера ключей криптопровайдера и создание сеансового ключа с дескриптором
hKey из введенной пользователем ключевой фразы (см. подразд. 5.2) */
...
// открытие исходного файла
```

```
TmpFile2.open(SECFILE, ios::in | ios::binary);
// создание файла для шифротекста
TmpFile1.open(TMPFILE, ios::out | ios::binary);
// цикл шифрования
do
{ // чтение блока данных из исходного файла
TmpFile2.read((char*)Buf, sizeof(Buf));
// получение фактической длины прочитанных данных
Len=TmpFile2.gcount();
// шифрование блока данных в буфере
CryptEncrypt(hKey, 0, TmpFile2.eof(), 0, Buf, &Len,
sizeof(Buf));
// запись зашифрованного блока в файл с шифротекстом
TmpFile1.write((const char*)Buf, Len); }
while(!TmpFile2.eof());
// закрытие файлов с открытым текстом и шифротекстом
TmpFile1.close();
TmpFile2.close();
// удаление файла с открытым текстом
remove(SECFILE);
// разрушение сеансового ключа
CryptDestroyKey(hKey);
/* генерация сеансового ключа с дескриптором hKey для
расшифрования файла путем генерации его из введенной
пользователем ключевой фразы, см. подразд. 5.2 */
...
// открытие файла с шифротекстом
TmpFile1.open(TMPFILE, ios::in | ios::binary);
// создание файла для расшифрованного текста
TmpFile2.open(SECFILE, ios::out | ios::binary);
// цикл расшифрования
do
{ // чтение блока данных с шифротекстом
TmpFile1.read((char*)Buf, sizeof(Buf));
// получение фактической длины прочитанных данных
Len=TmpFile1.gcount();
// расшифрование блока шифротекста
CryptDecrypt(hKey, 0, TmpFile1.eof(), 0, Buf, &Len);
// запись блока расшифрованных данных в файл
TmpFile2.write((const char*)Buf, Len); }
while(!TmpFile1.eof());
// закрытие файлов
TmpFile1.close();
TmpFile2.close();
// удаление файла с шифротекстом
remove(TMPFILE);
```

```
// разрушение сеансового ключа
CryptDestroyKey(hKey);
// продолжение работы программы (см. подразд. 5.2)
...
```

Если при шифровании используются вектор инициализации и (или) случайные значения, то эти данные могут быть записаны в конец (начало) файла с шифротекстом для использования при расшифровании информации из этого файла. Если сеансовый ключ не генерируется из ключевой фразы, а создается случайно, то в файл с шифротекстом должны быть также записаны блобы с этим ключом, зашифрованным открытыми ключами обмена всех пользователей, которые должны иметь доступ к этому файлу.

Если к сеансовому ключу шифрования файла имеет доступ только один пользователь, то потеря (например, в результате программной или аппаратной ошибки) пары ключей обмена этого пользователя может привести к невозможности восстановления зашифрованных данных. Для предотвращения подобной угрозы может использоваться процедура резервной авторизации:

1) после шифрования файла и экспортирования сеансового ключа обычным образом сеансовый ключ экспортируется повторно с помощью открытого ключа резервной авторизации, а полученный блоб посылается сервису резервной авторизации вместе с описанием сеансового ключа и другой идентифицирующей информации;

2) при необходимости восстановления сеансового ключа и расшифрования файла потребуется сначала пройти аутентификацию у сервиса резервной авторизации.

Сервис резервной авторизации должен выполняться на защищенном (с помощью организационных и инженерно-технических методов, см. подразд. 1.3 и 1.4) компьютере и предоставлять память для хранения сеансовых ключей своих клиентов.

Для получения кода алгоритма шифрования и длины блока шифра для сеансового ключа с дескриптором hKey может использоваться функция CryptGetKeyParam:

```
BOOL CryptGetKeyParam(HCRYPTKEY hKey, DWORD dwParam,
BYTE *pbData, DWORD *pdwDataLen, DWORD dwFlags); /*
если dwParam= KP_ALGID, то в буфер *pbData длины
*pdwDataLen записывается код алгоритма симметричного
шифрования; если dwParam= KP_BLOCKLEN, то в буфер за-
писывается значение длины блока в битах (для потоковых
шифров это значение всегда равно нулю); для параметра
dwParam допустимы также значения, определенные для
функции CryptSetKeyParam; значение параметра dwFlags
не используется и должно быть равно нулю */
```

5.4. Использование функций CryptoAPI для получения и проверки электронной цифровой подписи

После создания пары ключей ЭЦП (см. подразд. 5.2) возможен вызов функций CryptoAPI для получения и проверки электронной цифровой подписи под электронными документами. Перед получением ЭЦП необходимо получить хеш-значение для подписываемых данных. Для этого должны использоваться функции CryptCreateHash, CryptHashData и CryptHashSessionKey (см. подразд. 5.2). Затем вызывается функция CryptoAPI для получения ЭЦП:

```
BOOL CryptSignHash(HCRYPTHASH hHash, DWORD dwKeySpec,
LPCTSTR sDescription, DWORD dwFlags, BYTE *pbSignature,
DWORD *pdwSigLen); /* получение для хеш-значения с дес-
криптором hHash ЭЦП в буфере *pbSignature длины *pdwSigLen
(после выполнения функции в эту переменную записывается
фактическая длина ЭЦП); dwKeySpec=AT_SIGNATURE,
sDescription=NULL, dwFlags=0 */
```

Параметр dwKeySpec может также принимать значение AT_KEYEXCHANGE. В этом случае для получения ЭЦП будет использован секретный ключ обмена сеансовыми ключами, что может потребоваться для обеспечения только целостности данных без прямого указания на их владельца (например, сеансового ключа).

Параметр sDescription может содержать строку с описанием подписываемых данных, которая будет добавлена к хеш-значению перед его подписанием. При последующей проверке подписи будет необходимо задать ту же строку, что обеспечит подтверждение знания проверяющей стороной дополнительной информации о подписанном документе.

Для проверки электронной цифровой подписи вначале также необходимо вычислить хеш-значение для электронного документа, чьи аутентичность и целостность проверяются. После этого вызывается функция CryptoAPI, выполняющая проверку ЭЦП:

```
BOOL CryptVerifySignature(HCRYPTHASH hHash, BYTE
*pbSignature, DWORD dwSigLen, HCRYPTKEY hPubKey, LPCTSTR
sDescription, DWORD dwFlags); /* проверка ЭЦП из буфе-
ра *pbSignature длины dwSigLen для хеш-значения с дес-
криптором hHash с помощью открытого ключа hPubKey
(sDescription=NULL или строка с описанием проверяемого
документа, dwFlags=0) */
```

Приведем пример получения и проверки ЭЦП для файла, чье имя содержится в константе SECFILE (константа TMPFILE содержит имя файла с ЭЦП):

```
#include <fstream.h>
fstream TmpFile1, TmpFile2; // файловые переменные
HCRYPTPROV hProv=0; // дескриптор криптопровайдера
HCRYPTHASH hHash; // дескриптор хеш-значения
HCRYPTKEY hPubKey=0; /* дескриптор открытого ключа ЭЦП
BYTE Buf[128]; /* буфер для чтения из хешируемого файла*/
DWORD Len; // длина прочитанных из файла данных
BYTE Sign[MAX_PATH]; // буфер для ЭЦП
DWORD SigLen=sizeof(Sign); // длина ЭЦП
/* функция преобразования сообщений на русском языке
для вывода в консольном приложении Windows */
char * MessageOut(char * str)
{ static char OutString[MAX_PATH]; /* буфер для преоб-
разованного сообщения */
CharToOem(str, OutString);
return OutString; }
/* получение дескриптора контейнера ключей и создание
пары ключей ЭЦП (см. подразд. 5.2) */
...
// создание пустого хеш-значения
CryptCreateHash(hProv, CALG_SHA, 0, 0, &hHash);
// открытие подписываемого файла
TmpFile1.open(SECFILE, ios::in | ios::binary);
// создание файла с ЭЦП
TmpFile2.open(TMPFILE, ios::out | ios::binary);
// цикл хеширования файла
do
{ // чтение данных из файла в буфер
TmpFile1.read((char*)Buf, sizeof(Buf));
// получение фактической длины прочитанных данных
Len=TmpFile1.gcount();
// хеширование данных из буфера
CryptHashData(hHash, Buf, Len, 0); }
while(!TmpFile1.eof());
// получение ЭЦП
CryptSignHash(hHash, AT_SIGNATURE,NULL, 0, Sign,
&SigLen);
// запись ЭЦП в файл
TmpFile2.write((const char*)Sign, SigLen);
// закрытие файлов
TmpFile1.close();
TmpFile2.close();
// разрушение хеш-значения
CryptDestroyHash(hHash);
/* получение дескриптора открытого ключа ЭЦП hPubKey
(см. подразд. 5.2) */
```

```
...
// создание пустого хеш-значения
CryptCreateHash(hProv, CALG_SHA, 0, 0, &hHash);
// открытие проверяемого файла
TmpFile1.open(SECFILE, ios::in | ios::binary);
// открытие файла с ЭЦП
TmpFile2.open(TMPFILE, ios::in | ios::binary);
// цикл хеширования файла
do
{ // чтение данных из проверяемого файла в буфер
TmpFile1.read((char*)Buf, sizeof(Buf));
// получение фактической длины прочитанных данных
Len=TmpFile1.gcount();
// хеширование данных из буфера
CryptHashData(hHash,Buf,Len,0); }
while(!TmpFile1.eof());
// чтение ЭЦП из файла
TmpFile2.read(Sign, SigLen);
// проверка ЭЦП
if(!CryptVerifySignature(hHash, Sign, SigLen, hPubKey,
NULL, 0))
cout<<MessageOut ("Подпись неверна!");
else cout<<MessageOut("Подпись верна!");
// закрытие файлов
TmpFile1.close();
TmpFile2.close();
// разрушение хеш-значения
CryptDestroyHash(hHash);
// разрушение дескриптора открытого ключа ЭЦП
CryptDestroyKey(hPubKey);
// продолжение работы программы (см. подразд. 5.2)
...
```

Возможны одновременное выполнение шифрования данных и получения ЭЦП для них, а также расшифрования и проверки ЭЦП. При этом перед шифрованием данных в приложении должно быть создано пустое хеш-значение с помощью функции CryptCreateHash и полученный дескриптор должен быть указан в качестве значения параметра hHash при вызове функции CryptEncrypt (см. подразд. 5.3). Хеширование данных в этом случае производится перед их шифрованием. После завершения процесса шифрования полученное хеш-значение может быть подписано с помощью функции CryptSignHash.

При выполнении одновременного расшифрования и хеширования данных первоначально также должно быть создано пустое хеш-значение с помощью функции CryptCreateHash, после чего

полученный дескриптор должен быть указан в качестве значения параметра hHash при вызове функции CryptDecrypt (см. подразд. 5.3). Данные вначале будут расшифровываться, а затем хешироваться. Полученное хеш-значение может быть затем использовано для проверки ЭЦП с помощью функции CryptVerifySignature.

Существует возможность получения фактического хеш-значения до или вместо получения ЭЦП. Для этого предназначена следующая функция CryptoAPI:

```
BOOL CryptGetHashParam(HCRYPTHASH hHash, DWORD dwParam,
BYTE *pbData, DWORD *pdwDataLen, DWORD dwFlags); /*
получение в буфере *pbData длиной *pdwDataLen фактического хеш-значения с дескриптором hHash
(dwParam=HP_HASHVAL); после завершения в *pdwDataLen
записывается фактическая длина переданных данных; значение параметра dwFlags не используется и должно быть
равно нулю */
```

Параметр dwParam может принимать еще два значения: HP_HASHSIZE (в *pbData будет содержаться размер в байтах хеш-значения) и HP_ALGID (в буфер *pbData будет помещен код используемого алгоритма хеширования). После получения фактического хеш-значения добавление новых данных в него с помощью функций CryptHashData и CryptHashSessionKey невозможно.

С помощью еще одной функции CryptoAPI может быть получена ЭЦП для фактического хеш-значения. Для этого вначале создается пустое хеш-значение с помощью функции CryptCreateHash, а затем вызывается функция CryptSetHashParam:

```
BOOL CryptSetHashParam(HCRYPTHASH hHash, DWORD dwParam,
BYTE *pbData, DWORD dwFlags); /* установка в хеш-значении с дескриптором hHash новых данных из буфера *pbData
(dwParam= HP_HASHVAL); значение параметра dwFlags не
используется и должно быть равно нулю */
```

Размер буфера *pbData должен быть равен длине хеш-значения для используемой функции хеширования. После копирования данных в хеш-значение для него может быть получена ЭЦП с помощью функции CryptVerifySgnature.

5.5. Защита документов Microsoft Office от несанкционированного доступа

Защита документов Microsoft Office от несанкционированного доступа основана на их шифровании с помощью вызова соответствующих функций CryptoAPI. При установке защиты пользователю предлагается ввести пароль доступа к защищаемому докумен-

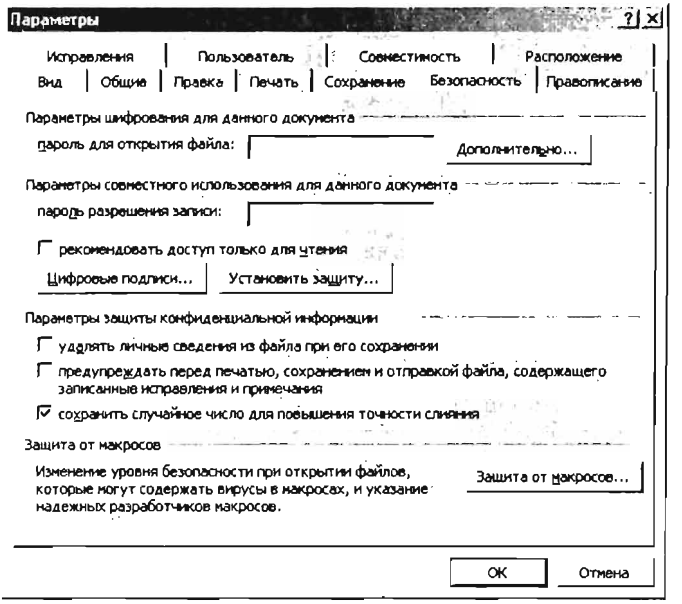

Рис. 5.3. Установка защиты на документ Microsoft Word

ту, из которого будет сгенерирован сеансовый ключ шифрования этого документа (см. подразд. 5.2 и 5.3). При попытке в дальнейшем открыть защищаемый документ потребуется ввод пароля доступа, на основании которого произойдут генерация сеансового ключа и расшифрование документа.

В текстовом процессоре Microsoft Word (версия Microsoft Office XP и старше) установка защиты от несанкционированного доступа к редактируемому документу выполняется с помощью команды меню Сервис | Параметры | Безопасность (рис. 5.3). Кнопка

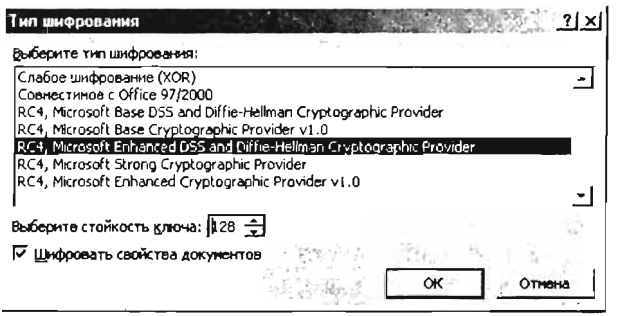

Рис. 5.4. Выбор параметров шифрования в Microsoft Word

«Дополнительно» позволяет установить параметры шифрования документа (рис. 5.4):

- тип шифрования (на основе выбора одного из установленных в системе криптопровайдеров и алгоритма потокового шифрования RC4);
- стойкость (длину) сеансового ключа шифрования в битах;
- необходимость шифрования свойств документа (возможно только при использовании шифрования с помощью CryptoAPI).

При выборе типа шифрования нецелесообразно выбирать варианты «Слабое шифрование (XOR)» и «Совместимое с Office 97/2000», поскольку в этом случае для защиты документа будет применено ненадежное шифрование, не использующее возможностей CryptoAPI. Существует немало программных средств, позволяющих расшифровывать защищенные таким образом документы путем простого перебора возможных паролей доступа.

При выборе типа шифрования, основанного на использовании одного из установленных в системе криптопровайдеров, необходимо установить максимально возможную длину ключа шифрования (обычно 128 бит).

Стойкость шифрования документа зависит также от длины пароля (фактически ключевой фразы для генерации сеансового ключа). Максимальная длина пароля доступа равна 255 знакам. При выборе пароля доступа к документу необходимо руководствоваться теми же соображениями, что и при выборе пароля пользователя для входа в КС (см. подразд. 2.2): выбирать пароли достаточной длины и сложности, не использовать один пароль для защиты различных документов, не использовать легко угадываемые пароли, совпадающие с логическим именем пользователя или названием документа, и т. п.

Для защиты от несанкционированного внесения изменений в документ Microsoft Word можно установить пароль разрешения записи в него. В этом случае перед открытием документа также будет предложено ввести пароль доступа. При вводе неправильного пароля или отказе от ввода пароля документ будет открыт только для чтения. Однако эту защиту нельзя считать достаточно надежной, поскольку измененная версия документа может быть сохранена под другим именем, после чего файл с оригинальным документом может быть удален, а вновь созданный файл соответствующим образом переименован. Кроме того, пароль разрешения записи содержится непосредственно в тексте документа и поэтому может быть просто удален из него с помощью специальных программных средств.

Для защиты документов Word от несанкционированного изменения необходимо применять средства разграничения доступа к папкам и файлам, имеющиеся в защищенных версиях операционной системы Windows и файловой системе NTFS (см. подразд. 3.3).

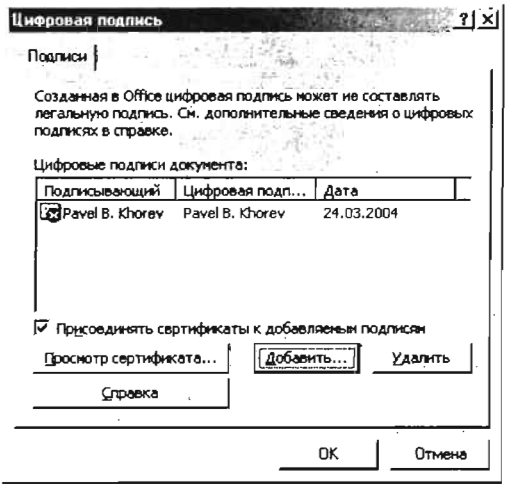

Рис. 5.5. Создание ЭЦП для документа Microsoft Word

Документ Microsoft Word (версия Microsoft Office XP и старше) может быть снабжен электронной цифровой подписью для обеспечения его аутентичности и целостности. Добавление ЭЦП к файлу документа возможно с помощью кнопки «Цифровые подписи» в окне настроек параметров безопасности (см. рис. 5.3). Для добавления ЭЦП к документу необходимо в окне «Цифровые подписи» (рис. 5.5) выбрать соответствующий сертификат ключа ЭЦП (см. подразд. 4.8). При получении первой подписи для документа следует с помощью кнопки «Добавить» выбрать сертификат для добавляемой к документу ЭЦП (рис. 5.6).

Сертификат открытого ключа ЭЦП может быть получен одним из следующих способов:

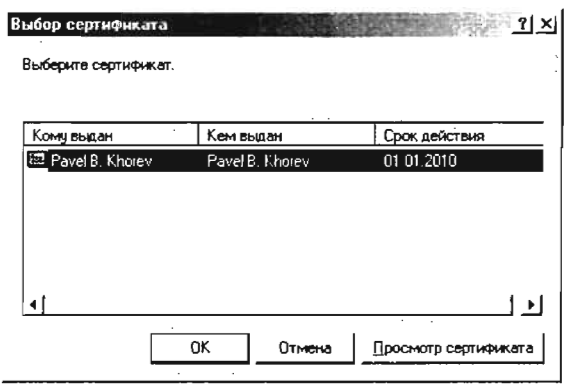

Рис. 5.6. Выбор сертификата для получения или проверки ЭЦП

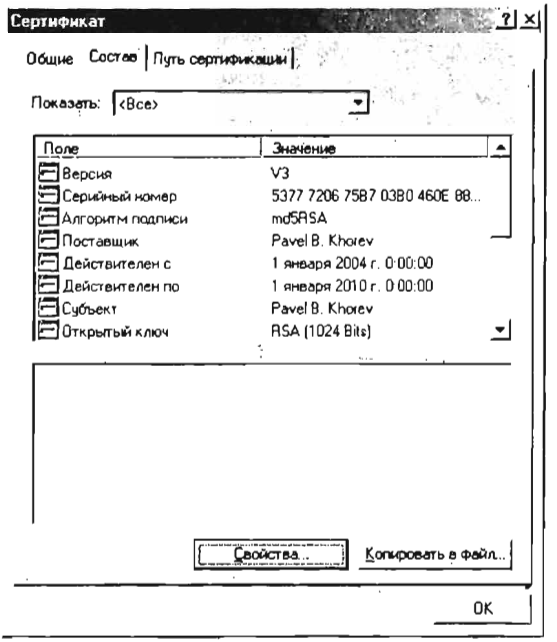

Рис. 5.7. Просмотр сертификата

• в удостоверяющем центре корпоративной КС, использующей, например, инфраструктуру открытых ключей Microsoft Windows (см. подразд. 4.8);

• в коммерческом удостоверяющем центре (например, в удостоверяющем центре компании VeriSign, Inc.);

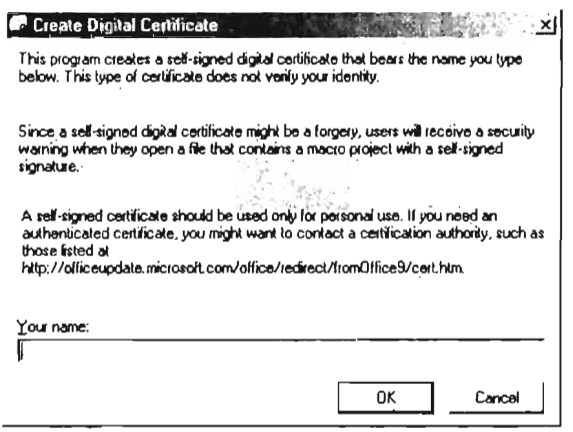

Рис. 5.8. Создание сертификата с помощью программы Selfcert

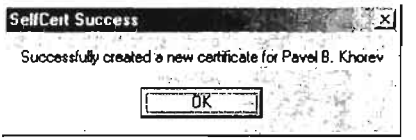

Рис. 5.9. Подтверждение создания сертификата

• самостоятельно с помощью программы Selfcert.exe, входящей в стандартную поставку пакета Microsoft Office.

Перед добавлением ЭЦП к защищаемому документу сертификат можно просмотреть с помощью кнопки «Просмотр сертификата» (рис. 5.7). При самостоятельном создании сертификата (вместе с соответствующим ему секретным ключом ЭЦП) программа Selfcert.exe запросит имя владельца сертификата (рис. 5.8), а после успешного завершения процедуры создания выдаст соответствующее сообщение (рис. 5.9). Самостоятельно созданный сертификат предназначен только для персонального использования владельцем защищаемого документа.

При попытке сохранения измененного документа, снабженного ЭЦП, Microsoft Word выдаст предупреждение о том, что все ЭЦП будут удалены из документа (рис. 5.10). Если файл с защищенным ЭЦП документом будет изменен с помощью других программных средств (например, с помощью Блокнота Windows), то при последующей попытке открытия документа Microsoft Word выдаст соответствующее сообщение (рис. 5.11) и файл с документом открыт не будет.

Защита от несанкционированного доступа к электронным таблицам Microsoft Excel и презентациям Microsoft PowerPoint в пакете Microsoft Office XP производится полностью аналогично за-

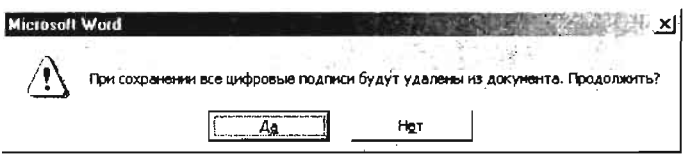

Рис. 5.10. Предупреждение об удалении ЭЦП при сохранении измененного документа

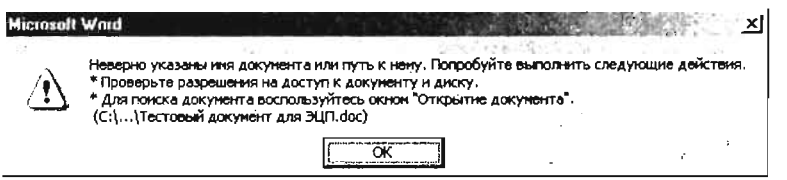

Рис. 5.11. Сообщение при попытке открытия измененного документа

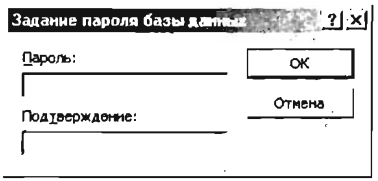

Рис. 5.12. Задание пароля на доступ к базе данных Microsoft Access

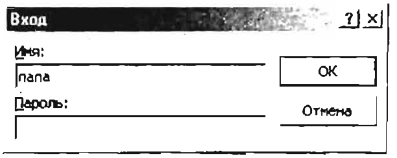

Рис. 5.13. Открытие базы данных с разграничением доступа

щите документов Microsoft Word. Установка защиты от несанкционированного доступа к базам данных Microsoft Access выполняется следующим образом.

1. Файл базы данных (с расширением «.mdb») открывается в монопольном режиме (с помощью раскрывающегося списка справа от кнопки «Открыть»).

2. Выполняется команда Сервис | Защита | Задать пароль базы данных и дважды вводится пароль доступа (рис. 5.12).

3. При последующем открытии базы данных потребуется ввести пароль доступа.

Пароль доступа к базе данных Microsoft Access сохраняется в файле базы данных в открытом виде.

Вместо использования одного пароля доступа к базе данных Microsoft Access можно организовать разграничение доступа к ее объектам (таблицам, формам, запросам и отчетам) на уровне отдельных пользователей.

Вначале удобно воспользоваться услугами Мастера защиты Microsoft Access (команда меню Сервис | Защита | Мастер).

В диалоге с Мастером после открытия файла базы данных в монопольном режиме потребуется указать:

1) необходимость создания нового или изменения существующего файла рабочей группы для базы данных;

2) имя файла и код рабочей группы, а также имя владельца базы данных и название организации;

3) объекты разграничения доступа в базе данных (по умолчанию все таблицы);

4) предопределенные группы пользователей с заранее определенными правами доступа (например, все права или только чтение);

5) разрешенные права доступа для группы Users, в которую будут входить все зарегистрированные пользователи базы данных;

6) имена и пароли (возможно, первоначально пустые) всех регистрируемых Мастером пользователей базы данных;

7) группы, в которые будут входить регистрируемые Мастером пользователи.

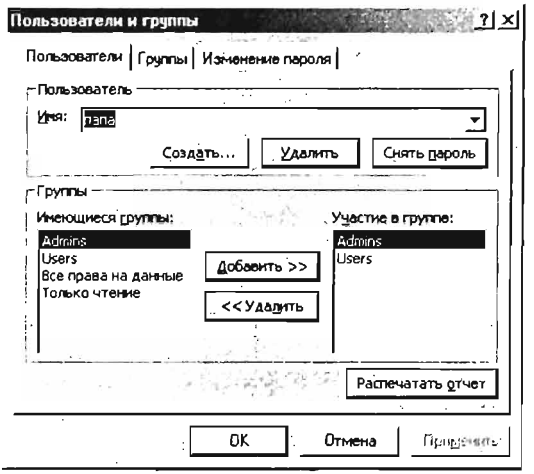

Рис. 5.14. Управление списками пользователей и групп для базы данных

После завершения работы Мастера для получения доступа к базе данных пользователю необходимо будет пройти процедуру входа, указав свои логическое имя и пароль доступа к базе данных (рис. 5.13). Для дальнейшего добавления новых пользователей базы данных и установки им прав доступа к ней необходимо использовать соответственно команды меню Сервис | Защита | Пользователи и группы и Сервис | Защита | Разрешения (рис. 5.14 и 5.15). Изменения в списке пользователей и групп, а также в их правах

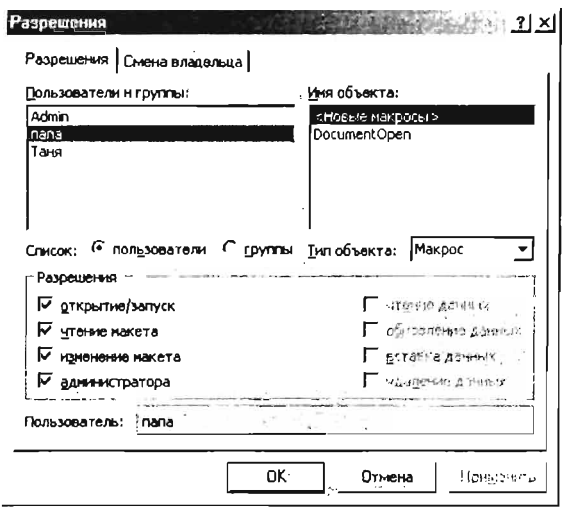

Рис. 5.15. Установка прав доступа к базе данных

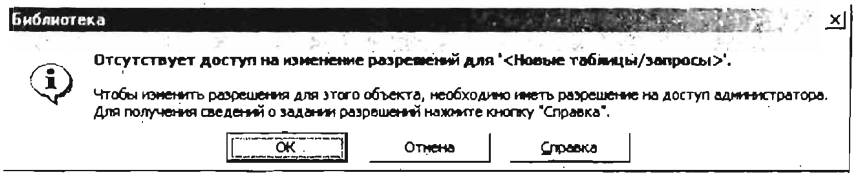

Рис. 5.16. Сообщение Microsoft Access об отказе в доступе

доступа к объектам базы данных могут быть произведены только владельцем базы данных или пользователем, входящим в группу Admins, в противном случае при попытке выполнения привилегированной операции Microsoft Access выдаст соответствующее сообщение об отказе в доступе (рис. 5.16).

Наиболее простым средством защиты базы данных Microsoft Access от несанкционированного доступа является ее шифрование, при котором она сжимается и становится недоступной для просмотра и редактирования отличными от Microsoft Access программными средствами. Но если в зашифрованной базе данных не используется разграничение доступа на уровне ее пользователей, то любой из них сможет открыть такую базу данных и получить полный доступ ко всем ее объектам. Поэтому шифрование должно применяться вместе с разграничением доступа на уровне пользователей или с применением пароля доступа к базе данных либо в целях экономии памяти при сохранении базы данных на дискете или компакт-диске.

Шифрование базы данных с разграничением доступа к ее объектам на уровне пользователей возможно только для владельца базы данных или члена группы Admins. Для шифрования базы данных Microsoft Access используется команда меню Сервис | Защита | Шифровать/расшифровать.

К достоинствам рассмотренных средств защиты от несанкционированного доступа относится то, что они могут применяться в программах пакета Microsoft Office, работающих под управлением как открытых, так и защищенных версий операционной системы Windows.

5.6. Шифрующая файловая система в защищенных версиях операционной системы Windows

В защищенных версиях операционной системы Windows, начиная с Windows 2000, для дополнительной защиты от несанкционированного доступа к папкам и файлам пользователей могут применяться средства шифрующей файловой системы (Encrypted File System, EFS), которые базируются на криптографическом

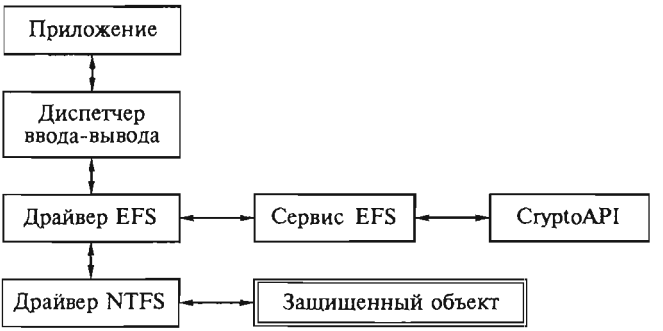

Рис. 5.17. Компоненты шифрующей файловой системы Windows

интерфейсе приложений Windows CryptoAPI. На рис. 5.17 показана структурная схема взаимодействия компонентов операционной системы при использовании EFS. Как видно из схемы, возможности шифрующей файловой системы доступны только на дисках под управлением файловой системы NTFS.

Шифрующая файловая система разработана с учетом следующих принципов:

- автоматическая генерация криптопровайдером асимметрических ключей обмена (см. подразд. 5.2) при первом обращении пользователя к услугам EFS (шифровании первой папки или файла);
- автоматическая генерация случайного сеансового ключа перед шифрованием файла, указанного пользователем;
- автоматическое шифрование и расшифрование в прозрачном режиме на уровне файла или папки (гарантируется, что для за-

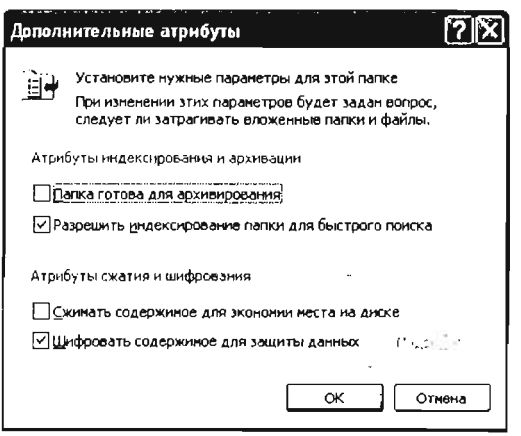

Рис. 5.18. Установка атрибута шифрования папки

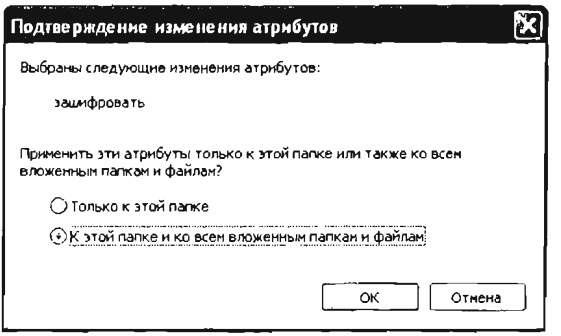

Рис. 5.19. Подтверждение изменения атрибута шифрования папки

шифрованного файла все созданные на его основе временные файлы также будут зашифрованы);

- возможность вызова функций шифрования и расшифрования с помощью контекстного меню Проводника Windows и программы командной строки cipher;
- возможность доступа к открытому ключу обмена пользователя со стороны операционной системы только во время сеанса его работы в системе.

Пользователь сообщает шифрующей файловой системе Windows о необходимости шифрования папки или файла с помощью дополнительных атрибутов этих объектов, доступных по команде контекстного меню Свойства | Другие (рис. 5.18). Состояние выключателя «Шифровать содержимое для защиты данных» определяет необходимость шифрования информации в данном объекте. При изменении этого состояния EFS запрашивает пользователя о подтверждении изменения атрибута шифрования, а также о том,

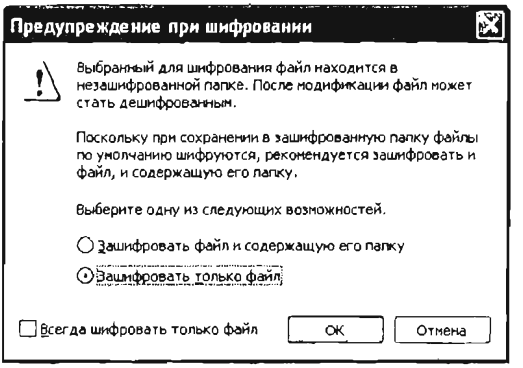

Рис. 5.20. Предупреждение при шифровании отдельного файла

распространяется это изменение только на папку или также на все вложенные в нее файлы и папки (рис. 5.19).

При изменении атрибута шифрования для отдельного файла шифрующая файловая система также запрашивает пользователя о подтверждении этого изменения и предлагает выбрать один из двух вариантов: зашифровать (расшифровать) только один файл или применить новое значение атрибута шифрования ко всей содержащий файл папке (рис. 5.20). Если выбранный для шифрования файл находится в незашифрованной папке (и наоборот), то после модификации этого файла значение его атрибута шифрования может автоматически измениться. Поэтому рекомендуется изменять значение атрибута шифрования для всей папки.

Перед шифрованием файла EFS обеспечивает генерацию случайного сеансового ключа (см. подразд. 5.2), после чего зашифровывает файл по алгоритму DESX (см. подразд. 4.4) экспортирует из криптопровайдера сеансовый ключ в блобе, зашифрованном с помощью открытого ключа обмена пользователя, и записывает этот блоб вместе с самим зашифрованным файлом в качестве одного из его атрибутов.

Перед расшифрованием зашифрованного файла EFS обеспечивает импорт блоба с сеансовым ключом шифрования файла в криптопровайдер, используя при этом секретный ключ обмена пользователя. После этого выполняется расшифрование файла.

В операционной системе Windows 2000 не обеспечивается возможность совместного доступа к зашифрованному файлу со стороны нескольких пользователей, а также передача по сети зашифрованных файлов. Поэтому для исключения угрозы потери зашифрованных пользователем данных, например из-за невозможности его входа в систему, администратор должен применять политику обязательного использования агента восстановления данных (Data Recovery Agent, DRA).

Политика восстановления зашифрованных данных определяется в рамках домена. Пара ключей обмена для агента восстановления создается после включения соответствующего параметра политики безопасности, после чего открытый ключ из этой пары реплицируется на все компьютеры домена, а секретный ключ сохраняется у администратора или на специально выделенном для этого компьютере. При использовании политики восстановления зашифрованных данных сеансовый ключ, на котором был зашифрован файл, экспортируется из CSP еще один раз — теперь в блобе, зашифрованном на открытом ключе агента восстановления, и это блоб записывается в качестве еще одного атрибута зашифрованного файла. При необходимости восстановления зашифрованного файла сеансовый ключ импортируется в криптопровайдер с использованием секретного ключа агента восстановления.

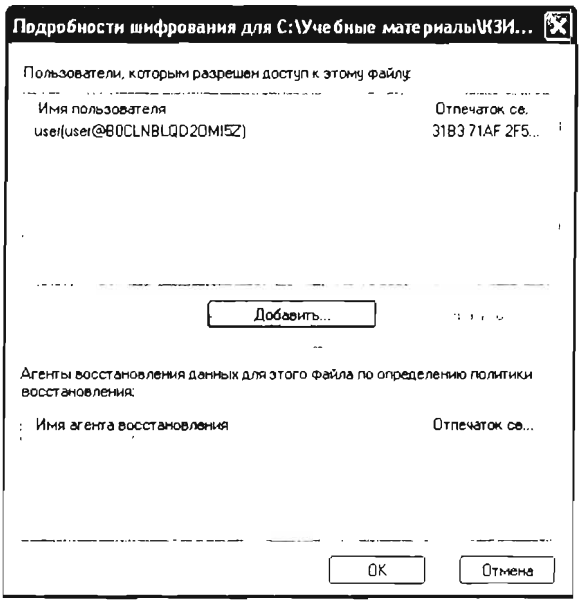

Рис. 5.21. Просмотр списка пользователей, имеющих доступ к зашифрованным объектам

В операционной системе Windows XP Professional и более поздних защищенных версиях этой системы реализована поддержка общего доступа к зашифрованным файлам. С помощью кнопки «Подробно» в окне установки дополнительных атрибутов зашифрованного файла (см. рис. 5.18) открывается окно просмотра списка пользователей, которым разрешен доступ к этому файлу (рис. 5.21). Кнопка «Добавить» в этом окне предназначена для добавления

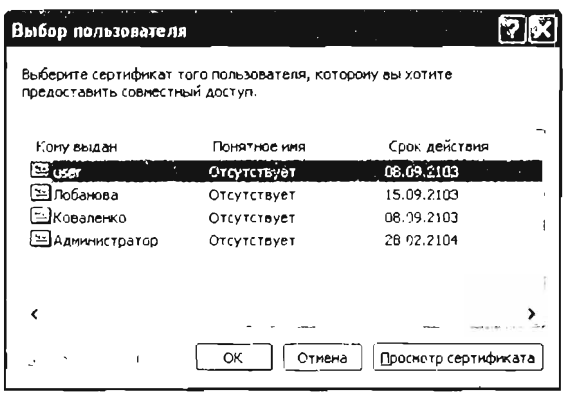

Рис. 5.22. Предоставление другому пользователю права доступа к зашифрованному объекту

пользователей к этому списку (рис. 5.22). Выбор возможен среди пользователей, уже обращавшихся за услугами шифрующей файловой системы (уже имеющих пары асимметричных ключей обмена) или имеющих сертификаты своих открытых ключей, заверенные используемым в КС удостоверяющим центром.

Разрешение доступа к зашифрованным файлам со стороны других пользователей может избавить от необходимости использовать в КС политику агента восстановления.

Недостатки реализованного механизма совместного доступа к зашифрованным файлам в Windows XP Professional:

- при выборе пользователей, которым разрешен доступ к зашифрованному файлу, нельзя использовать определенные в системе группы;
- нет возможности предоставления совместного доступа к зашифрованной папке.

В качестве дополнительной услуги в Windows XP Professional поддерживается выделение имен зашифрованных файлов и папок зеленым цветом, хотя это может рассматриваться и как дополнительное указание на объекты, содержащие конфиденциальную информацию. Помимо шифрования по алгоритму DESX поддерживается также и шифрование по алгоритму 3-DES (изменить алгоритм шифрования можно с помощью изменения значения соответствующего параметра безопасности).

Начиная с версии Windows XP Professional, возможно использование средств шифрующей файловой системы при работе с кэшированными на клиентском компьютере автономными файлами из общих для пользователей сети папок. Благодаря кэшированию пользователи могут продолжать просматривать и редактировать файлы из общих папок в ситуации, когда их мобильный компьютер отключен от сети. При последующем подключении к серверу сети операционная система синхронизирует произведенные изменения в файлах с их более ранними версиями, размещенными в общих папках. Возможность автоматического шифрования и расшифрования автономных файлов повышает их защищенность от несанкционированного доступа при работе с ними на мобильных компьютерах (например, во время командировок пользователей).

На криптографическом интерфейсе приложений Windows и технологии многокомпонентных объектов (COM-технологии) базируется объект CAPICOM, доступный в версиях операционной системы Windows, начиная с Windows XP Professional. Объект CAPICOM может использоваться для выполнения основных криптографических операций в приложениях, созданных как на универсальных языках программирования (Visual Basic for Application, C++ и др.), так и на языках сценариев (например, Visual Basic Scripting Edition).Объекты CAPICOM поддерживают получение и

проверку ЭЦП, обмен сеансовыми ключами, симметричное шифрование и расшифрование данных.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что называется провайдером криптографического обслуживания?
2. На каких принципах строится взаимодействие прикладной программы и криптопровайдера?
3. Как обеспечиваются в операционной системе аутентичность и целостность криптопровайдера?
4. Что хранится в контейнере ключей пользователя при работе с криптопровайдером?
5. Как создаются в CryptoAPI сеансовые ключи шифрования?
6. Как может быть осуществлен обмен ключами шифрования в CryptoAPI?
7. Какие функции CryptoAPI предназначены для шифрования и расшифрования данных?
8. Что должно сохраняться вместе с зашифрованным файлом?
9. Как изменить режим шифрования при использовании функций CryptoAPI?
10. Какие функции CryptoAPI используются для получения и проверки электронной цифровой подписи?
11. Каков порядок вызова функций CryptoAPI при получении ЭЦП?
12. Как могут быть совмещены процессы шифрования (расшифрования) и хеширования при использовании CryptoAPI?
13. В чем заключается защита от несанкционированного доступа к документам Microsoft Word, Microsoft Excel и Microsoft PowerPoint?
14. Как обеспечить надежную защиту от несанкционированного доступа к документам Microsoft Office XP?
15. Как получить электронную цифровую подпись для документа Microsoft Office XP? Как может быть получен сертификат открытого ключа автора документа?
16. В чем особенности защиты от несанкционированного доступа к базам данных Microsoft Access?
17. Как функционирует шифрующая файловая система операционной системы Windows?
18. Для чего предназначен агент восстановления данных шифрующей файловой системы? Как обеспечивается возможность восстановления зашифрованных файлов?
19. Кому и как может быть предоставлен доступ к зашифрованным файлам в операционной системе Windows XP Professional?

ГЛАВА 6

ЗАЩИТА КОМПЬЮТЕРНЫХ СИСТЕМ ОТ ВРЕДОНОСНЫХ ПРОГРАММ

6.1. Вредоносные программы и их классификация

К *вредоносным программам* (иначе называемым *разрушающими программными воздействиями*) относятся компьютерные вирусы и программные закладки. Впервые термин компьютерный вирус ввел в употребление специалист из США Ф. Коэн в 1984 г. *Компьютерным вирусом* называют автономно функционирующую программу, обладающую одновременно тремя свойствами:

- способностью к включению своего кода в тела других файлов и системных областей памяти компьютера;
- последующему самостоятельному выполнению;
- самостоятельному распространению в компьютерных системах.

Программной закладкой называют внешнюю или внутреннюю по отношению к атакуемой компьютерной системе программу, обладающую определенными разрушительными функциями по отношению к этой системе:

- уничтожение или внесение изменений в функционирование программного обеспечение КС, уничтожение или изменение обрабатываемых в ней данных после выполнения некоторого условия или получения некоторого сообщения извне КС («логические бомбы»);
- превышение полномочий пользователя с целью несанкционированного копирования конфиденциальной информации других пользователей КС или создания условий для такого копирования («троянские» программы);
- подмена отдельных функций подсистемы защиты КС или создание люков в ней для реализации угроз безопасности информации в КС (например, подмена средств шифрования путем эмуляции работы установленной в КС платы аппаратного шифрования);
- перехват паролей пользователей КС с помощью имитации приглашения к его вводу или перехват всего ввода пользователей с клавиатуры;
- перехват потока информации, передаваемой между объектами распределенной КС (мониторы);
- распространение в распределенных КС с целью реализации той или иной угрозы безопасности информации (компьютерные

черви, которые в отличие от компьютерных вирусов не должны обладать свойством включения своего кода в тела других файлов) и др.

Компьютерные вирусы классифицируются по следующим признакам.

1. По способу распространения в КС:

а) файловые вирусы, заражающие файлы одного или нескольких типов;

b) загрузочные вирусы, заражающие загрузочные сектора жестких дисков и дискет;

с) комбинированные вирусы, способные заражать и файлы, и загрузочные сектора дисков.

2. По способу заражения других объектов КС:

а) резидентные вирусы, часть кода которых постоянно находится в оперативной памяти компьютера и заражает другие объекты КС;

b) нерезидентные вирусы, которые заражают другие объекты КС в момент открытия уже зараженных ими объектов.

3. По деструктивным возможностям:

а) безвредные вирусы, созданные в целях обучения, однако снижающие эффективность работы КС за счет потребления ее ресурсов (времени работы центрального процессора, оперативной и внешней памяти и др.);

b) неопасные вирусы, создающие различные звуковые и видеоэффекты;

с) опасные и очень опасные вирусы, вызывающие сбои в работе программного и (или) аппаратного обеспечения компьютера, потерю программ и данных, а потенциально — вывод из строя аппаратуры КС и нанесение вреда здоровью пользователей (с помощью, например, эффекта двадцать пятого кадра).

4. По особенностям реализуемого алгоритма:

а) вирусы-спутники, создающие для заражаемых файлов одноименные файлы с кодом вируса и переименовывающие исходные файлы (при открытии зараженного файла фактически открывается файл с кодом вируса, в котором после выполнения предусмотренных автором действий открывается исходный файл);

b) паразитические вирусы, которые обязательно изменяют содержимое заражаемых объектов;

с) вирусы-невидимки («стелс»-вирусы), в которых путем перехвата обращений операционной системы к зараженным объектам и возврата вместо них оригинальных незараженных данных скрывается факт присутствия вируса в КС (при собственном обращении к дисковой памяти вирусы-невидимки также используют нестандартные средства для обхода средств антивирусной защиты);

d) вирусы-призраки (полиморфные вирусы), каждая следующая копия которых в зараженных объектах отличается от преды-

дущих (не содержит одинаковых цепочек команд за счет применения шифрования на различных ключах базового кода вируса).

5. По наличию дополнительных возможностей:

a) по обработке атрибута «только чтение» заражаемых файлов;

b) сохранению времени последнего изменения зараженного файла;

c) обработке прерывания, вызванного неисправимой ошибкой устройства ввода-вывода (например, для подавления сообщения операционной системы об ошибке при попытке заражения объекта на защищенном от записи устройстве или объекта, доступ к которому по записи для текущего пользователя запрещен; вывод подобного сообщения может обнаружить присутствие вируса в КС);

d) распространению в КС не только при открытии уже зараженного объекта, но и при выполнении любой операции с ним.

Кто и почему пишет компьютерные вирусы? Одной из главных причин, безусловно, является наличие достаточного количества квалифицированных программистов, обладающих избытком свободного времени и не отягощенных морально-этическими нормами. Развитие рынка информационных технологий в конкретном государстве или регионе, очевидно, должно способствовать сокращению в нем количества потенциальных авторов вирусов, поскольку облегчает получение ими хорошо оплачиваемой работы и уменьшает их время досуга.

Другая причина появления компьютерных вирусов и вредоносных программ вообще является гораздо более серьезной и объективно постоянной. Разработка, внедрение и распространение вредоносных программ стало неотъемлемой частью современного информационного оружия, которое, в свою очередь, явилось порождением различных информационных войн между конкурирующими государствами, коммерческими организациями, общественно-политическими группами и т. п. Очевидно, что эти причины появления вредоносных программ будут иметь место на протяжении весьма длительного периода времени.

Для отнесения конкретной программы к разряду программных закладок достаточно, чтобы данная программа обладала хотя бы одним из следующих свойств:

- скрытие признаков своего присутствия в КС («зачем Вы прячетесь — ведь за Вами никто не гонится»);
- реализация самодублирования, ассоциации себя с другими объектами КС, перенос своего кода в не занимаемые ранее области оперативной или внешней памяти;
- искажение кода других программ в оперативной памяти компьютера;
- сохранение данных, размещенных в оперативной памяти, в других ее областях;

• искажение, блокировка, подмена сохраняемых (передаваемых) данных, полученных в результате работы других программ или уже находящихся во внешней памяти.

В соответствии с методами внедрения программных закладок в КС и возможным местам их размещения в системе закладки могут быть разделены на следующие группы:

• программные закладки, ассоциированные с BIOS;

• закладки, ассоциированные с программами начальной загрузки и загрузки операционной системы;

• закладки, ассоциированные с драйверами операционной системы и другими системными модулями;

• закладки, ассоциированные с прикладным программным обеспечением общего назначения (например, архиваторами);

• программные файлы, содержащие только код закладки и внедряемые с помощью пакетных командных файлов;

• закладки, маскируемые под прикладное программное обеспечение общего назначения;

• закладки, маскируемые под игровое и образовательное программное обеспечение (для облегчения их первоначального внедрения в КС).

6.2. Загрузочные и файловые вирусы

Загрузочные вирусы заражают главный загрузочный сектор жесткого диска (Master Boot record, MBR) или загрузочный сектор раздела жесткого диска, системной дискеты или загрузочного компакт-диска (Boot Record, BR), подменяя находящиеся в них программы начальной загрузки и загрузки операционной системы своим кодом. Исходное содержимое этих секторов при этом сохраняется в одном из свободных секторов диска или непосредственно в теле вируса.

После заражения MBR, являющегося первым сектором нулевой головки нулевого цилиндра жесткого диска, вирус получает управление сразу по завершении работы процедуры проверки оборудования (POST), программы BIOS Setup (если она была вызвана пользователем), процедур BIOS и его расширений. Получив управление, загрузочный вирус выполняет следующие действия:

1) копирование своего кода в конец оперативной памяти компьютера, уменьшая тем самым размер ее свободной части;

2) переопределение «на себя» нескольких прерываний BIOS, в основном связанных с обращением к дискам;

3) загрузка в оперативную память компьютера истинной программы начальной загрузки, в функции которой входит просмотр таблицы разделов жесткого диска, определение активного разде-

ла, загрузка и передача управления программе загрузки операционной системы активного раздела;

4) передача управления истинной программе начальной загрузки.

Аналогичным образом работает и загрузочный вирус в BR, замещая программу загрузки операционной системы.

Обычной формой заражения компьютера загрузочным вирусом является случайная попытка загрузки с несистемной дискеты, загрузочный сектор которой заражен вирусом. Эта ситуация возникает, когда зараженная дискета остается в дисководе при выполнении перезагрузки операционной системы. После заражения главного загрузочного сектора жесткого диска вирус распространяется при первом обращении к любой незараженной дискете.

В связи с тем что загрузочные сектора магнитных дисков имеют небольшой размер (512 байт), при их заражении сложным вирусом необходимо найти дополнительные места размещения той части его кода, которая не поместилась в зараженных объектах. Для решения этой задачи могут использоваться:

- занесение кода вируса в свободные сектора диска с пометкой их как дефектных для защиты от перезаписи;
- помещение кода вируса в редко занимаемые свободные сектора в конце одного из разделов жесткого диска, которые остаются помеченными как свободные;
- перенос части кода вируса в зарезервированные для последующего применения операционной системой сектора диска.

Загрузочные вирусы, как правило, относятся к группе резидентных вирусов (см. подразд. 6.1).

Изменение принятого по умолчанию в BIOS порядка использования дисковых устройств при загрузке (см. подразд. 3.1) обеспечит не только защиту от несанкционированной загрузки незащищенной операционной системы, но и устранит один из каналов заражения компьютера загрузочными вирусами (но останутся другие, связанные с распространением комбинированных вирусов).

Файловые вирусы заражают файлы различных типов:

- программные файлы с расширениями «.exe» или «.com»;
- файлы документов Microsoft Word (расширение «.doc»), электронных таблиц Microsoft Excel (расширение «.xls»), баз данных Microsoft Access (расширение «.mdb») и презентаций Microsoft Power Point (расширения «.ppt» или «.pps»);
- файлы шаблонов (на языке HTML) отображения объектов в папке (расширение «.htt»);
- файлы драйверов реального режима (расширение «.sys»);
- пакетные командные файлы (расширение «.bat»);
- файлы оверлеев приложений операционной системы MS-DOS (расширения «.ovr» или «.ovl») и др.

При заражении файла вирус записывает свой код в начало, середину или конец файла либо сразу в несколько мест. Исходный файл изменяется таким образом, что после открытия файла управление немедленно передается коду вируса. После получения управления код вируса выполняет следующую последовательность действий:

1) заражение других файлов и (комбинированные вирусы) системных областей дисковой памяти;

2) установка в оперативной памяти собственных резидентных модулей (резидентные вирусы);

3) выполнение других действий, зависящих от реализуемого вирусом алгоритма;

4) продолжение обычной процедуры открытия файла (например, передача управления исходному коду зараженной программы).

Вирусы в программных файлах при заражении изменяют их заголовок таким образом, что после загрузки программы в оперативную память управление передается коду вируса. Например, переносимый формат исполнимого файла операционных систем Windows и OS/2 (Portable Executable, PE) имеет следующую структуру:

1) заголовок в формате операционной системы MS-DOS;

2) код программы реального режима процессора, которая получает управление при попытке запуска приложения Windows в среде операционной системы MS-DOS;

3) заголовок PE-файла;

4) дополнительный (опциональный) заголовок PE-файла;

5) заголовки и тела всех сегментов приложения (кода программы, ее статических данных, экспортируемых программой данных, импортируемых программой данных, отладочной информации и др.).

Раздел, содержащий опциональный заголовок PE-файла, включает в себя поле, содержащее адрес точки входа приложения. Непосредственно перед точкой входа в сегменте кода приложения размещается таблица адресов импортируемых из других модулей функций (Import Address Table, IAT), которая заполняется действительными адресами во время загрузки исполнимого кода в адресное пространство процесса (см. подразд. 3.3).

При заражении вирусом программного файла адрес точки входа приложения изменяется таким образом, что указывает на начало кода вируса и обеспечивает автоматическое получение им управления при загрузке программного файла. Возможна также модификация модулей ядра операционной системы (например, kernel32.dll) для перехвата вызовов некоторых системных функций (например, CreateProcess, CreateFile, ReadFile, WriteFile, CloseHandle) для заражения других файлов.

При заражении файлов драйверов реального режима код вируса дописывается к коду драйвера, изменяются адреса процедур стратегии и (или) прерывания. После этого при инициализации зараженного драйвера вирус перехватывает соответствующий запрос операционной системы, передает его коду драйвера, дожидается ответа драйвера и корректирует его, чтобы остаться в оперативной памяти вместе с основной частью драйвера в одном с ней блоке.

При заражении пакетных командных файлов вирусы вставляют в их начало специальный текст, например такой:

```
@ECHO OFF
REM ...
copy %0 b.com >nul
b.com
del b.com
rem ...
```

Вместо многоточия размещаются данные или код вируса. Исполнимый файл b.com, хотя и содержит исходный текст пакетного командного файла, но является уже последовательностью машинных команд, так как строка символов "@ECHO OFF\r\nREM" является фактически корректной последовательностью процессорных команд, а после нее располагается настоящий код вируса. Резидентная часть данного вируса может отслеживать запись данных в различные файлы и, если первой записываемой строкой является команда @echo, то вирус определяет данный файл как пакетный командный и заражает его.

При заражении файлов шаблонов отображения файлов в папке вирус размещает в тексте на языке HTML оператор (тэг) object, содержащий Active-X компонент с кодом вируса, который будет получать управление сразу после открытия зараженной папки.

Разновидностью файловых вирусов являются вирусы в кластерах зараженного логического диска или дискеты. При заражении код вируса копируется в один из свободных кластеров диска, который помечается в таблице размещения файлов (File Allocation Table, FAT) как последний кластер файла. Затем изменяются описания программных файлов в каталоге — вместо номера первого выделенного файлу кластера помещается номер кластера, содержащего код вируса. При этом истинный номер первого кластера зараженного файла шифруется и сохраняется, например в неиспользуемой части описания файла в каталоге.

При запуске зараженного файла управление получает код вируса, который:

1) устанавливает в оперативной памяти свой резидентный модуль, который в дальнейшем будет перехватывать все обращения к зараженному диску;

2) загружает исходный программный файл и передает ему управление.

При последующем обращении к каталогу с зараженными файлами резидентная часть вируса передает операционной системе истинные значения номеров первых кластеров, выделенных зараженным файлам.

Вирусы в файлах документов, созданных программами пакета Microsoft Office, распространяются с помощью включенных в них макросов (процедур на языке программирования Visual Basic for Applications, VBA, или WordBasic, WB). Поэтому такие вирусы иногда называют вирусами в макросах, или *макровирусами*.

Языки программирования макросов, особенно VBA, являются универсальными языками, поддерживающими технологию объектно-ориентированного программирования, имеющими большую библиотеку стандартных макрокоманд и позволяющими создавать достаточно сложные процедуры. Кроме того, поддерживаются автоматически выполняемые макросы, связанные с определенными событиями (например, открытием документа) или определенными действиями пользователя (например, при вызове команды сохранения документа в файле).

Примерами автоматически выполняющихся макросов, связанных с определенными событиями обработки документа Microsoft Word, являются:

- AutoExec (автоматически выполняется при запуске текстового процессора Microsoft Word, если находится в файле шаблонов normal.dot или в файле подпапки Startup папки Microsoft Office);
- AutoNew (автоматически получает управление при создании нового документа);
- AutoOpen (автоматически выполняется при открытии документа);
- AutoClose (автоматически выполняется при закрытии документа);
- AutoExit (автоматически получает управление при завершении работы текстового процессора Microsoft Word).

В табличном процессоре Microsoft Excel поддерживается только часть из автоматически выполняемых макросов, причем имена этих макросов слегка изменены — Auto_open и Auto_close.

В текстовом процессоре Microsoft Word определены также макросы, которые автоматически получают управление при вызове пользователем одной из стандартных команд — FileSave (Файл | Сохранить), FileSaveAs (Файл | Сохранить как), ToolsMacro (Сервис | Макрос | Макросы), ToolsCustomize (Сервис | Настройка) и т.д.

Документ Microsoft Office может также содержать макросы, автоматически получающие управление при нажатии пользователем определенной комбинации клавиш на клавиатуре или достижении некоторого момента времени (даты, времени суток).

Любой макрос (в том числе и автоматически выполняемый) из отдельного документа может быть записан в файл шаблонов normal.dot (и наоборот) и, тем самым, стать доступным при редактировании любого документа Microsoft Word. Запись макроса в файл normal.dot может быть осуществлена с помощью стандартной макрокоманды MacroCopy (WordBasic), метода OrganizerCopy объекта Application или методов Copy стандартных объектов Organizer (Microsoft Word) и Sheets (Microsoft Excel).

Для манипулирования файлами, размещенными во внешней памяти компьютера, в макросах могут использоваться стандартные макрокоманды Open (открытие существующего или создание нового файла), SetAttr (изменение атрибутов файла), Name (переименование файла или папки), Get (чтение данных из открытого файла), Put (запись данных в открытый файл), Seek (изменение текущей позиции записи или чтения из файла), Close (закрытие файла), Kill (удаление файла), RmDir (удаление папки), MkDir (создание новой папки), ChDir (изменение текущей папки) и др.

Стандартная макрокоманда Shell позволяет выполнить любую из установленных на компьютере программ или системных команд.

Таким образом, язык программирования VBA вполне может быть использован авторами макровирусов для создания весьма опасного кода. Первый макровирус появился в 1995 г., а в настоящее время их насчитывается десятки тысяч. Кроме того, известно немало вирусов, заражающих как программные файлы, так и файлы документов.

Простейший макровирус в документе Microsoft Word заражает остальные файлы документов следующим образом:

1) при открытии зараженного документа управление получает содержащийся в нем макрос с кодом вируса;

2) вирус помещает в файл шаблонов normal.dot другие макросы со своим кодом (например, FileOpen, FileSaveAs и FileSave);

3) вирус устанавливает в реестре Windows и (или) в инициализационном файле Microsoft Word соответствующий флаг о произведенном заражении;

4) при последующем запуске Microsoft Word первым открываемым файлом фактически является уже зараженный файл шаблонов normal.dot, что позволяет коду вируса автоматически получать управление, а заражение других файлов документов может происходить при их сохранении с помощью стандартных команд Microsoft Word.

Применительно к рассмотренной в подразд. 6.1 классификации компьютерных вирусов можно сказать, что большинство макровирусов относятся к группе резидентных вирусов, так как часть их кода постоянно присутствует в оперативной памяти компьютера во все время работы программы из пакета Microsoft Office.

Размещение кода макровируса внутри документа Microsoft Office можно указать достаточно схематично, так как формат файлов документов весьма сложен и содержит последовательность блоков данных различных форматов, объединяемых между собой с помощью большого числа служебных данных. Особенностью макровирусов является и то, что они могут заражать файлы документов на компьютерах различных платформ, а не только IBM PC. Заражение будет возможно, если на компьютере будут установлены офисные программы, полностью совместимые с программами из пакета Microsoft Office.

При сохранении файлов документов в их состав включаются и случайные данные, не связанные с содержанием документа, а содержащиеся в выделенных, но до конца не заполненных при редактировании документа блоках оперативной памяти. Поэтому при добавлении новых данных в документ его размер может измениться непредсказуемым образом, в том числе и уменьшиться. Это не позволяет судить о заражении файла документа макровирусами, так как его размер после заражения также изменится непредсказуемо. Отметим также, что в случайно сохраненной вместе с файлом общедоступного документа информацией может оказаться и конфиденциальная.

Большинство известных макровирусов размещают свой код только в макросах. Однако существуют и такие разновидности вирусов в файлах документов, в которых код вируса хранится не только в макросах. Эти вирусы включают в себя небольшой по объєму макрос-загрузчик основного кода вируса, который вызывает встроенный в Microsoft Office редактор макросов, создает новый макрос с кодом вируса, выполняет его, после чего для скрытия следов своего присутствия удаляет созданный макрос. Основной код вируса в этом случае присутствует в виде массива строк либо в теле макроса-загрузчика, либо в области переменных зараженного документа.

Заражение файла шаблонов normal.dot — не единственный способ распространения макровирусов на компьютере пользователя. Возможно также заражение файлов дополнительных шаблонов, размещенных в папке Startup внутри папки Microsoft Office. Еще один способ заражения файлов документов пользователя макровирусами состоит в их внедрении в файлы надстроек над Microsoft Word, размещающихся в папке AddIns папки Microsoft Office. Макровирусы, не помещающие свой код в общий шаблон normal.dot, можно отнести к группе нерезидентных вирусов (см. подразд. 6.1). Эти макровирусы для заражения других файлов либо применяют стандартные макрокоманды для работы с файлами и папками языка VBA, либо используют список последних редактированных пользователем файлов, который содержится в подменю «Файл» программы Microsoft Word и других программ пакета Microsoft Office.

В табличном процессоре Microsoft Excel не используется файл шаблонов normal.dot, поэтому для заражения других файлов документов пользователей используются файлы из папки Startup. Особенностью макровирусов, заражающих файлы электронных таблиц Excel, является то, что для их написания может использоваться не только язык программирования VBA, но и язык макрокоманд старых версий Microsoft Excel, поддержка которого обеспечена и в более поздних версиях этого табличного процессора.

В системе управления базами данных Microsoft Access для автоматического получения управления при возникновении некоторого события (например, открытия базы данных) предназначены макросы, написанные на специальном языке сценариев, имеющем весьма ограниченные возможности. Но в этих автоматически выполняемых макросах-сценариях (например, в макросе AutoExec, автоматически получающем управление при запуске Microsoft Access) могут вызываться полноценные макросы, написанные на языке VBA. Поэтому для заражения базы данных Microsoft Access вирусу необходимо создать или заменить автоматически выполняемый макрос-сценарий и скопировать в заражаемую базу модуль с макросами, содержащими основную часть кода вируса.

Известны комбинированные вирусы, которые могут заражать и базы данных Microsoft Access, и документы Microsoft Word. Подобный вирус состоит из двух основных частей, каждая из которых заражает файлы документов своего типа («.doc» или «.mdb»). Но обе части такого вируса способны переносить свой код из одного приложения Microsoft Office в другое. При переносе кода вируса из Microsoft Access в папке Startup создается зараженный файл дополнительных шаблонов («.dot»-файл), а при переносе кода вируса из Microsoft Word создается зараженный файл базы данных Access, который передается в качестве параметра запускаемому кодом вируса приложению Microsoft Access (msaccess.exe).

Макровирус Melissa заражает файлы документов и шаблонов Microsoft Word, а также рассылает свои копии в сообщениях электронной почты (при наличии такой возможности на компьютере пользователя). Данный вирус также редактирует содержимое реестра Windows и отключает антивирусную защиту Microsoft Word (см. подразд. 6.3). Для рассылки своих копий через электронную почту вирус Melissa использует стандартные средства VBA для запуска других приложений (в частности, Microsoft Outlook) и вызова их процедур (например, для работы с адресной книгой, создания и отправки электронного сообщения). К отправленным вирусом сообщениям присоединяется зараженный файл с редактируемым в данный момент пользователем документом Microsoft Word (при этом возможна и утечка конфиденциальной информации пользователя).

6.3. Методы обнаружения и удаления вирусов

Прежде чем рассмотреть методы обнаружения и удаления компьютерных вирусов, остановимся на организационных мерах, предназначенных для профилактики заражения ими объектов КС. Основными каналами распространения компьютерных вирусов в настоящее время являются:

- электронная почта, сообщения которой могут содержать зараженные присоединенные файлы;
- телеконференции и электронные доски объявлений в сети Интернет;
- свободное и условно свободное программное обеспечение, размещенное на общедоступных узлах сети Интернет и случайно или намеренно зараженное вирусами;
- размещенные на общедоступных узлах сети Интернет информационные ресурсы, содержащие ссылки на зараженные файлы с элементами управления Active-X;
- локальные компьютерные сети организаций, создающие удобную среду для заражения вирусами объектов на других рабочих станциях и серверах;
- обмен зараженными файлами на дискетах или записываемых компакт-дисках между пользователями КС;
- использование нелицензионных компакт-дисков с программным обеспечением и другими информационными ресурсами.

Для предупреждения вирусного заражения локальной сети организации или компьютера отдельного пользователя необходимо максимально перекрыть возможность проникновения вирусов с использованием перечисленных каналов.

В частности, могут быть использованы следующие профилактические меры:

- физическое или логическое (для отдельных учетных записей) отключение накопителей на гибких магнитных дисках и компакт-дисках;
- разграничение прав отдельных пользователей и групп на доступ к папкам и файлам операционной системы и других пользователей (см. подразд. 3.3 и 3.5);
- ограничение времени работы в КС привилегированных пользователей (для выполнения действий в КС, не требующих дополнительных полномочий, администраторы должны использовать вторую учетную запись с обычными привилегиями),
- использование, как правило, только лицензионного программного обеспечения, приобретенного у официальных представителей фирм-правообладателей;
- выделение не подсоединенного к локальной сети компьютера для тестирования полученного из ненадежных источников программного обеспечения и т.д.

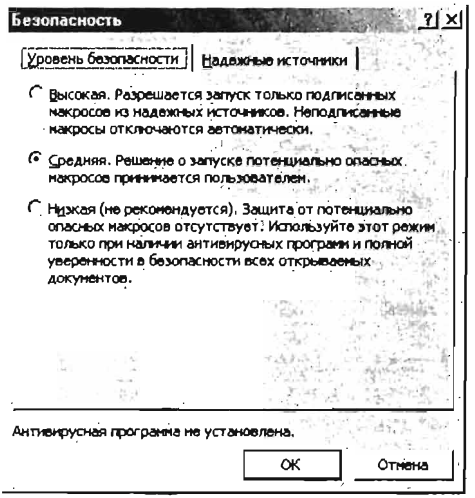

Рис. 6.1. Установка защиты от макросов, которые могут содержать вирусы

В качестве профилактической меры предупреждения заражения файлов пользователей макровирусами в программах пакета Microsoft Office предусмотрена встроенная защита от потенциально опасных макросов. Эта защита устанавливается (в пакете Microsoft Office XP) с помощью команды меню Сервис | Параметры | Безопасность и кнопки «Защита от макросов» (см. рис. 5.3) или команды меню Сервис | Макрос | Безопасность.

Возможен выбор одного из трех уровней защиты (рис. 6.1):

- высокой безопасности, при установке которого будет разрешено выполнение только макросов, снабженных ЭЦП и полученных из надежных источников, список которых содержится на вкладке «Надежные источники» окна выбора уровня безопасности (макросы без ЭЦП будут автоматически отключаться);
- средней безопасности, при выборе которого при открытии содержащего макросы документа решение об отключении этих макросов будет приниматься самим пользователем (рис. 6.2);
- низкой безопасности, при установке которого все макросы в открываемом документе будут выполняться (корпорация Microsoft рекомендует устанавливать данный уровень безопасности только при наличии антивирусных программ на компьютере пользователя и полной уверенности в безопасности открываемых документов).

Для получения подписи под макросами документа Microsoft Office (проектом макросов) необходимо открыть окно системы программирования Microsoft Visual Basic с помощью команды меню любой из программ этого пакета Сервис | Макрос | Макросы, вы-

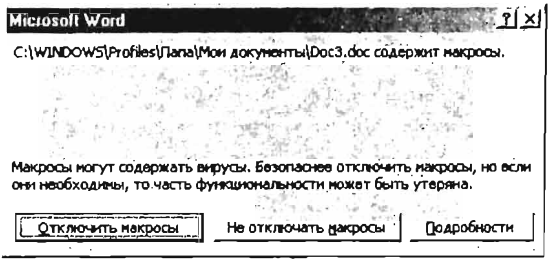

Рис. 6.2. Окно принятия решения об отключении макросов

бора имени макроса и нажатия кнопки «Изменить». В окне системы программирования затем выполняется команда Tools | Digital Signature и в появившемся окне цифровой подписи (рис. 6.3) выбирается сертификат открытого ключа ЭЦП, который в дальнейшем будет использован для проверки подписи (см. подразд. 5.5).

Отключенные при открытии документа Microsoft Office XP макросы могут быть просмотрены с помощью команды меню Сервис | Макрос | Макросы, но не могут быть выполнены (в предыдущих версиях Microsoft Office просмотр отключенных макросов был также невозможен).

В качестве дополнительной меры защиты можно отменить автоматическое выполнение макросов, полученных из надежных источников (рис. 6.4). Если снять флажок «Доверять всем установленным надстройкам и шаблонам», то вывод предупреждения о наличии макросов в открываемых документах будет производиться и для макросов, находящихся в уже установленных на компьютере пользователя шаблонах и надстройках Microsoft Office.

Установка защиты от потенциально опасных макросов не позволяет отделить макросы, расширяющие функциональность приложений Microsoft Office, от макросов, содержащих вирусы. Кро-

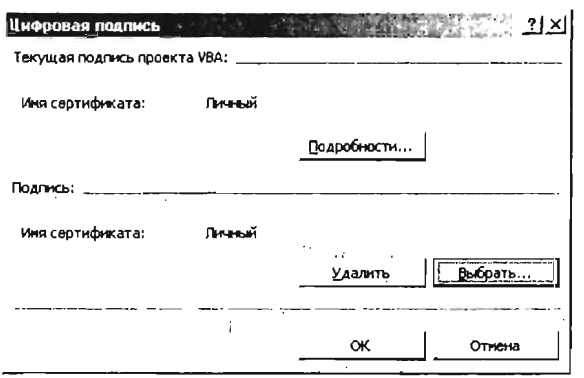

Рис. 6.3. Получение ЭЦП для макросов документа Microsoft Office

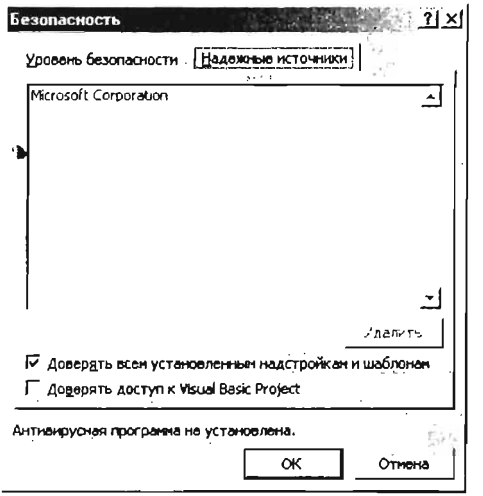

Рис. 6.4. Управление доверием к шаблонам и надстройкам Microsoft Office

ме того, некоторые из макровирусов, получив однажды управление, могут понизить уровень безопасности до самого низкого и тем самым блокировать встроенную защиту от макросов.

При открытии документа Microsoft Office существует возможность отключения автоматически выполняющихся макросов, которую можно использовать с помощью одного из следующих способов:

- удерживание нажатой клавиши Shift на клавиатуре при запуске приложения или открытии документа Microsoft Office;
- добавление параметра /m в командную строку запуска приложения Microsoft Office (например, "C:\Program Files\Microsoft Office\Office10\WINWORD.EXE /m") в ярлыке на рабочем столе пользователя и в значение параметра command раздела реестра HKEY_CLASSES_ROOT\Word.Document\shell\Open\command;
- создание макроса с именем AutoExec, который будет автоматически получать управление при запуске программы Microsoft Word, и запись в это макрос единственной строки

```
WordBasic. DisableAutoMacros
```

Созданный макрос AutoExec должен быть сохранен в файле общих шаблонов normal.dot, после чего автоматическое выполнение макросов в редактируемых документах будет запрещено.

Недостатками способов блокирования автоматического выполнения макросов является то, что предупреждается распространение только тех вирусов, которые используют автоматически выполняемые макросы, связанные с определенными событиями обработки документа, а также то, что блокируется выполнение и полезных макросов.

Для защиты от несанкционированного изменения файла общих шаблонов normal.dot, что требуется для распространения в КС многих макровирусов, доступ к этому файлу может быть защищен с помощью специального пароля. Для его установки необходимо:

1) открыть окно системы программирования Visual Basic for Applications с помощью команды меню программы пакета Microsoft Office Сервис | Макрос | Редактор Visual Basic;

2) в окне структуры проекта выделить узел Normal и выполнить команду его контекстного меню Normal Properties;

3) открыть вкладку Protection, установить флажок Lock project for viewing и ввести в поле Password (с подтверждением в поле Confirm password) пароль для доступа к файлу общих шаблонов.

Для снижения риска заражения вирусами при просмотре информационных ресурсов сети Интернет могут быть использованы свойства обозревателя Microsoft Internet Explorer (вкладка «Безопасность» окна свойств). Всем узлам зоны Интернет, в которые не войдут явно отнесенные к другим зонам узлы, целесообразно назначить высокий уровень безопасности, в соответствии с которым будут отключены:

- возможность загрузки файлов с этих узлов;
- доступ к источникам данных за пределами домена;
- возможность запуска программ в плавающих фреймах загруженного документа на языке HTML;
- возможность автоматической загрузки по каналам распространения программного обеспечения (будет выполняться только извещение пользователя);
- возможность запроса отсутствующего сертификата при попытке загрузки файлов на компьютер пользователя (само данное действие будет автоматически отменено);
- возможность перехода между фреймами документа на HTML через разные домены;
- возможность установки элементов рабочего стола пользователя;
- поддержка языка Java;
- поддержка сценариев;
- загрузка, запуск и использование элементов Active-X, кроме помеченных как безопасные.

К программно-аппаратным методам защиты от заражения загрузочными вирусами можно отнести защиту, устанавливаемую с помощью программы BIOS Setup (параметр Anti-Virus Protection или аналогичный функции Advanced BIOS Features или аналогичной). Включение этой защиты (задание значения Enable указанному параметру) обеспечит выдачу предупреждающего сообщения при попытке записи в загрузочные сектора дисковой памяти. К недостаткам подобной защиты от заражения вирусами отно-

сится то, что она может быть отключена кодом вируса прямым редактированием содержимого энергонезависимой CMOS-памяти, хранящей настройки, которые были установлены программой BIOS Setup.

Другим способом программно-аппаратной защиты от заражения компьютерными вирусами может быть использование специального контроллера, вставляющегося в один из разъемов для расширений аппаратного обеспечения компьютера, и драйвера для управления работой контроллера. Поскольку контроллер подключается к системной шине компьютера, он получает полный контроль всех обращений к его дисковой памяти. С помощью драйвера контроллера могут быть указаны недоступные для изменения области дисковой памяти (загрузочные сектора, области установленного на компьютере системного и прикладного программного обеспечения и т. п.). В этом случае заражение указанных областей любыми вирусами будет невозможно. К недостаткам подобной защиты относится то, что в указанные области дисковой памяти будет невозможна и легальная запись данных.

Обязательным средством антивирусной защиты является использование специальных программ для обнаружения и удаления вирусов в различных объектах КС. Рассмотрим методы обнаружения компьютерных вирусов.

1. Просмотр (сканирование) проверяемых объектов (системных областей дисковой и оперативной памяти, а также файлов заданных типов) в поиске сигнатур (уникальных последовательностей байтов) известных вирусов. Соответствующие программные средства называют сканерами, а при наличии дополнительной функции удаления обнаруженных вирусов — полифагами. Обычно сканеры запускаются при загрузке операционной системы или после обнаружения признаков вирусного заражения другими средствами.

К недостаткам программ-сканеров относятся:

- необходимость постоянного обновления баз данных сигнатур известных вирусов, которые используются при поиске;
- неспособность обнаружения новых компьютерных вирусов;
- недостаточная способность обнаружения сложных полиморфных вирусов.

2. Обнаружение изменений в объектах КС путем сравнения их вычисленных при проверке хеш-значений с эталонными (или проверки ЭЦП для этих объектов). При вычислении хеш-значений объектов могут учитываться и характеристики (атрибуты) проверяемых файлов. Подобные программные средства называют ревизорами, или инспекторами. Потенциально они могут обнаружить и новые вирусы. Однако не все изменения проверяемых объектов вызываются вирусным заражением: обновление отдельных компонентов операционной системы, легальное изменение фай-

лов документов и пакетных командных файлов и т. п. Программы-ревизоры не могут помочь при записи на жесткий диск компьютера пользователя уже зараженного файла, но могут обнаружить заражение вирусом новых объектов. Обычно программы-ревизоры выполняются при загрузке операционной системы.

3. Эвристический анализ — проверка системных областей памяти и файлов с целью обнаружения фрагментов исполнимого кода, характерного для компьютерных вирусов (например, установка резидентной части кода вируса). Потенциально эвристические анализаторы способны обнаружить (с определенной вероятностью) любые новые разновидности компьютерных вирусов. В Российской Федерации наиболее известным программным средством такого рода является программа DrWeb И.Данилова.

4. Постоянное присутствие в оперативной памяти компьютера с целью контроля всех подозрительных действий других программ — попыток изменения загрузочных секторов дисков, установки резидентного модуля и т. п. Подобные программы получили название мониторов. Мониторы также автоматически проверяют на наличие известных вирусов все устанавливаемые дискеты и компакт-диски, открываемые файлы и т. п. Обычно мониторы используют общую со сканерами базу сигнатур вирусов и загружаются в оперативную память в процессе загрузки операционной системы.

К недостаткам программ-мониторов относятся:

- выполнение контролируемых мониторами действий незараженными вирусами программами (например, выполнение команды операционной системы MS-DOS label изменяет метку тома в загрузочном секторе раздела жесткого диска или дискеты);
- снижение эффективности работы КС за счет потребления процессорного времени и уменьшения размера свободной оперативной памяти.

5. Вакцинирование — присоединение к защищаемому файлу специального модуля контроля, следящего за целостностью данного файла с помощью вычисления его хеш-значения и сравнения с эталоном. После заражения файла вирусом его целостность, естественно, будет нарушена. Однако вирусы-невидимки (см. подразд. 6.1) способны обнаруживать присоединенный код программы-вакцины и обходить реализуемую им проверку. Кроме того, данный метод плохо применим для защиты файлов документов Microsoft Office.

Большинство современных комплексов антивирусных программ включают в свой состав сканеры (с дополнительной функцией избыточного сканирования или эвристического анализа), мониторы и, реже, инспекторы.

При заражении файлов документов новыми разновидностями вирусов могут иметь место следующие внешние проявления, которые могут быть обнаружены вручную:

• невозможность сохранения документа Microsoft Word в другом формате (соответствующие элементы списка возможных форматов в диалоговой панели «Сохранить как» заблокированы);

• создание новых документов Microsoft Word в формате файла шаблонов;

• невозможность сохранения документа Microsoft Word в другой папке;

• наличие неизвестных файлов в папках Startup Microsoft Word и Microsoft Excel;

• наличие в рабочей книге Microsoft Excel посторонних (в том числе скрытых) листов;

• блокировка команд меню, предназначенных для работы с макросами в документах Microsoft Office;

• выдача сообщений об ошибках в макросах при их заведомом отсутствии в редактируемом документе Microsoft Office;

• установка пароля доступа к документу Microsoft Office;

• изменения в конфигурационных файлах программ пакета Microsoft Office и в реестре Windows;

• выдача пользователям странных сообщений.

Для обнаружения присутствия в документе Microsoft Office посторонних макросов можно использовать просмотр и редактирование их списка с помощью команды меню Сервис | Макрос | Макросы. Если эта команда заблокирована кодом вируса, то обнаружить присутствие в файле документа макросов с предопределенными именами (AutoOpen, FileSave и т.п.) можно с использованием программы Finder.exe, входящей в состав пакета Microsoft Office. С помощью этой программы организуется поиск любых текстовых строк (например, содержащих имена предопределенных макросов) в файлах документов Microsoft Office, расположенных в заданных перед началом поиска папках.

При автоматическом удалении обнаруженных антивирусными программами вирусов могут применяться два основных метода:

• удаление уже известных вирусов с помощью заранее разработанного алгоритма лечения зараженных данным типом вируса файлов;

• попытка удаления неизвестных до этого времени вирусов на основе сведений об общих принципах работы вирусов и (или) предварительно сохраненной информации о незараженном файле.

Рекомендуется перед удалением обнаруженных вирусов выполнить копирование зараженных файлов на резервный носитель информации, чтобы не потерять ценных данных.

Сложные разновидности даже известных вирусов не всегда могут быть удалены, а зараженные ими файлы восстановлены. Поэтому для обязательной подготовки к возможному заражению объектов КС вирусами необходимо:

• подготовить защищенную от записи системную дискету или загрузочный компакт-диск, записав на них последние версии антивирусных программ и баз сигнатур известных вирусов;

• постоянно обновлять версии установленного в КС антивирусного программного обеспечения;

• регулярно проверять объекты КС всеми имеющимися антивирусными программами (в том числе и инспекторами);

• обязательно проверять на наличие вирусов все входящие сообщения электронной почты и присоединенные к ним файлы;

• регулярно выполнять резервное копирование наиболее важных системных и прикладных файлов;

• отключить максимально возможное число каналов распространения вирусов.

Если лечение зараженных файлов с помощью антивирусных программ оказывается невозможным, то можно воспользоваться ручным восстановлением зараженных объектов, требующим наличия у исполнителя этой работы высокой квалификации, специальных знаний структуры файловой системы и различных форматов файлов, а также привлечения специального программного обеспечения. Иногда результатом ручного восстановления может стать создание большого числа новых файлов, которые затем надо просмотреть, чтобы найти фрагменты восстанавливаемых данных.

Для ручного удаления кода макровируса из файла общих шаблонов Microsoft Word (normal.dot) может использоваться удаление данного файла (при следующем запуске текстового процессора этот файл будет восстановлен в первоначальном виде). Недостатком этого способа является потеря стилевых настроек пользователей, поэтому зараженный файл normal.dot можно переименовать для ручного удаления из него макросов с вирусами.

После запуска Microsoft Word с восстановленным файлом общих шаблонов нужно выполнить команду меню Сервис | Шаблоны и надстройки, нажать кнопку «Организатор» и выбрать вкладку «Макросы».

Затем необходимо удалить из зараженного файла общих шаблонов и других файлов документов все или только подозрительные макросы. Последним шагом в этом случае будет удаление заново созданного первоначального файла normal.dot и переименование восстановленного файла общих шаблонов с настройками пользователей.

Для ручного удаления макровирусов из зараженных файлов документов (после лечения файла общих шаблонов) можно:

1) включить встроенную защиту от макровирусов (уровень средней безопасности);

2) открыть зараженный документ с отключением имеющихся в нем макросов;

3) выделить все содержимое документа с помощью соответствующей команды меню «Правка»;

4) скопировать выделенное содержимое в буфер обмена;

5) создать новый документ и вставить в него содержимое буфера обмена;

6) сохранить созданную копию (уже без макросов) открытого документа;

7) закрыть открытые документы и, если есть еще зараженные макровирусами файлы, перейти к п. 2.

В крайнем случае, при неудаче всех других попыток восстановления нормальной работоспособности КС после заражения ее объектов компьютерными вирусами, может потребоваться полная переустановка системы, включая форматирование дисковой памяти, восстановление главного загрузочного сектора, установку операционной системы и прикладного программного обеспечения, восстановление файлов данных с резервных носителей информации.

6.4. Программные закладки и методы защиты от них

Как уже отмечалось в гл. 3, в операционных системах с разграничением прав субъектов процессы получают полномочия того пользователя, который их породил. Когда пользователь выполняет программу, полученную из ненадежного источника (например, установленную с нелицензионного компакт-диска), то возможно выполнение частью этой программы недокументированных действий, связанных, например, с чтением и сохранением конфиденциальных данных пользователя. Причем возможно, что запущенная программа (например, игровая или обучающая) вообще не должна была иметь доступа к этим данным. Наиболее опасны в этом случае программы, используемые привилегированными пользователями. Угроза внедрения подобных и более опасных программных закладок актуальна практически для любых многопользовательских КС.

Одним из возможных методов защиты от программных закладок является использование принципа минимальных полномочий, в соответствии с которым субъекту (процессу или пользователю) всегда предоставляются в системе минимальные права. В этом случае необходимо также реализовать:

- возможность выполнения программ с большим, чем у запустившего их пользователя, объемом полномочий (например, программа для чтения конфиденциальных данных должна будет выдавать только их открытую часть);
- механизм динамического предоставления полномочий, в соответствии с которым текстовому процессору, например, будут

назначены полномочия пользователя на время доступа к его документам.

В операционной системе Windows всем программам предоставляется полный объем полномочий запустивших их пользователей (несущественным исключением является механизм олицетворения, см. подразд. 3.3). Имеющаяся в операционной системе Unix возможность запуска программ с правами их владельцев, а не запустивших их пользователей, может создать дополнительную угрозу безопасности КС (см. подразд. 3.5).

В подразд. 6.1 была приведена классификация закладок по используемым ими методам внедрения в защищенную КС. Применение каждого из этих методов требует наличия в КС бреши (возможно, временной). Эта брешь может стать следствием ошибок используемого в КС программного обеспечения или ошибок администрирования КС. После своего внедрения закладка создает собственную брешь в системе, поэтому наличие ранее использованной уязвимости уже не требуется.

При соблюдении правильной политики безопасности в КС, использующих защищенные версии операционных систем Windows или операционные системы семейства Unix, внедрение в них программных закладок практически невозможно (мы не рассматриваем здесь возможность наличия закладок в самом оригинальном коде этих операционных систем). Однако соблюдение адекватной угрозам политики безопасности сколько-нибудь продолжительное время также практически невозможно. В частности, наиболее вероятными причинами ослабления политики безопасности могут быть:

- разрешение доступа обычному пользователю к критичному системному ресурсу;
- сохранение в КС инсталлируемой по умолчанию уязвимой сетевой службы (например, telnet);
- оставление незаблокированной консоли администратора при его временном отсутствии;
- требования, предъявляемые пользователями, недостаточно квалифицированными в вопросах информационной безопасности, но занимающими ответственное положение в организации (например, предоставление им возможности удаленного доступа к КС) и т.п.

Для внедрения закладки в атакуемой КС нарушителю требуется записать на один из ее компьютеров и запустить инсталлятор закладки (программу для ее установки). Одной из задач инсталлятора закладки является обеспечение ее автоматического запуска при загрузке операционной системы, что обеспечит возможность установки виртуального канала связи с нарушителем по протоколу TCP/IP (закладка при этом будет исполнять роль сервера).

Для того чтобы обеспечить запуск инсталлятора на компьютере пользователя, нарушитель, как правило, маскирует его под привлекательную прикладную программу (например, расчета биоритмов или подбора паролей) либо включает код инсталлятора в полезную для пользователя утилиту (например, архиватор). Код инсталлятора закладки может находиться в автоматически выполняющемся макросе, включенном в присоединенный к электронному почтовому сообщению файл (см. разд. 6.2), или в самом почтовом сообщении («продвинутые» почтовые программы позволяют выполнять программный код, содержащийся в самом письме). Существуют и другие способы запуска инсталлятора закладки на компьютере пользователя.

Большинство известных программных закладок для операционной системы Windows характеризуются следующими особенностями:

- ссылка на исполнимый файл программной закладки содержится в разделе реестра HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run или (реже) в разделе реестра HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run;
- для организации виртуального канала связи с нарушителем используется протокол TCP/IP с очень большим номером порта, не используемым другими сетевыми программами;
- после запуска программной закладки создается отдельный процесс, отображаемый в списке системных процессов;
- после своего запуска программная закладка не выполняет никаких активных действий (в отличие от компьютерных вирусов), скрывая свое присутствие в системе.

После своего запуска программная закладка предоставляет нарушителю возможность удаленного доступа:

- для управления объектами файловой системы и системного реестра;
- управления процессами в КС;
- перехвата вводимых с клавиатуры данных (например, паролей пользователей);
- получения шифротекстов или хеш-значений ранее введенных паролей других пользователей и т. п.

В защищенных версиях операционной системы Windows, поддерживающих ограничение прав пользователей в КС и разграничение их доступа к объектам КС, инсталляция и использование программных закладок гораздо сложнее, так как целью нарушителя является инсталляция закладки привилегированным пользователем и ее выполнение от лица администратора или псевдопользователя SYSTEM. При неправильном назначении прав доступа к разделам реестра (например, предоставления права записи в раздел HKEY_LOCAL_MACHINE обычным пользователям) ин-

сталляция закладки может быть выполнена и пользователем с обычными полномочиями (будет обеспечена возможность автоматического запуска закладки при любой загрузке операционной системы). При правильном определении политики аудита (см. подразд. 3.4) действия закладки могут оставлять «следы» в файле аудита, которые могут быть обнаружены аудитором.

Для усложнения обнаружения присутствия закладки в КС ее инсталлятор может использовать следующие приемы:

- подмена модулей операционной системы или популярных прикладных программ (например, программ из пакета Microsoft Office) — требуется полная реализация в закладке всех функций подменяемого модуля, что возможно только при наличии полной документации на подменяемый модуль, а также (при использовании разграничения доступа к объектам КС) и полномочия администратора;
- прямое ассоциирование (внедрение) с уже установленными модулями системных или прикладных программ — не требуется полная реализация в закладке их функций, но может потребоваться наличие полномочий администратора КС;
- косвенное ассоциирование (внедрение) с кодом модулей, уже загруженных в оперативную память, — требуется присутствие в оперативной памяти инсталлирующей части закладки, выполняющейся, возможно, в привилегированном режиме, для получения доступа к модулям ядра операционной системы;
- применение методов компьютерной стеганографии (см. подразд. 4.9) для скрытия программной закладки в файлах-контейнерах (например, с графическими изображениями) — может использоваться в комбинации с другими приемами.

Рассмотрим возможные модели взаимодействия нарушителя с внедренной в КС программной закладкой:

- сохранение всей вводимой или выводимой в КС информации («перехват»);
- создание условий для ознакомления с ранее накопленной закладкой конфиденциальной информацией или отключения средств защиты информации в КС («троянский конь»);
- перехват всей передаваемой по сети информации («наблюдатель»);
- создание условий для облегчения несанкционированного доступа к конфиденциальной информации по различным каналам утечки, например путем постоянного обращения к конфиденциальным данным для последующего их перехвата по каналам ПЭМИН («компрометация»);
- искажение входных или выходных потоков данных при работе прикладного или системного программного обеспечения, инициация ошибок при его работе («искажение», или «инициатор ошибок»);

• изучение остаточной информации после работы программ, максимизация числа фрагментов с конфиденциальной информацией («уборка мусора»).

Примером использования модели «перехват» является замена провайдера аутентификации защищенных версий операционной системы Windows (см. подразд. 3.3). Прототипы функций, экспортируемых этим модулем операционной системы, содержатся в файле winwlx.h, входящем в состав любой системы программирования на языке С для операционной системы Windows.

Примером использования модели «искажение» является ограничение длины хешируемых данных в программах получения и проверки электронной цифровой подписи для создания возможности подмены защищаемого сообщения или файла.

Примерами наиболее известных программных закладок для удаленного управления объектами атакуемой КС являются Back Orifice, Net Bus, DIRT (Data Interception by Remote Transmission), Paparazzi. Особенностью программы Paparazzi является сохранение в специальном формате, использующем сжатие данных, получаемых с задаваемой периодичностью копий экрана компьютера пользователя. Заметим, что часть из этих программ могут называться их авторами не закладками, а средствами для удаленного доступа, контроля несанкционированного использования ресурсов КС и других благовидных действий.

Для обнаружения присутствия в системе программной закладки могут применяться следующие способы:

• просмотр списка активных процессов с помощью диспетчера задач операционной системы;

• просмотр состояния IP-портов с помощью системной программы netstat;

• просмотр разделов реестра для обнаружения дополнительно установленных программ, которые автоматически выполняются при загрузке операционной системы;

• просмотр файла аудита для поиска попыток доступа неизвестных процессов к критичным объектам КС или объектам с конфиденциальной информацией;

• контроль обращений процессов к объектам файловой системы, разделам реестра и используемым сетевыми программами портам (например, с помощью известных программ FileMon, RegMon и PortMon) и др.

Могут быть разработаны программы для автоматизации действий по обнаружению программных закладок с использованием всех или части указанных способов. В частности, некоторые антивирусные программы (сканеры и мониторы) могут обнаруживать инсталляторы закладок и сами закладки. Однако новые разновидности закладок могут преодолевать защиту, обеспечиваемую данными автоматизированными средствами. Например, для обеспе-

чения автоматического запуска закладок могут использоваться другие разделы реестра или конфигурационные файлы операционной системы win.ini и system.ini.

Поэтому наиболее эффективным методом защиты от программных закладок будет использование организационных мер, к которым, в частности, можно отнести следующие:

- минимизация времени работы в КС с полномочиями администратора;
- создание специальной учетной записи пользователя КС для выхода в сеть Интернет с минимальными полномочиями (запуск обозревателя и сохранение файлов в специальной папке);
- аккуратное использование почтовых и офисных программ привилегированными пользователями (например, запрет доступа администратора к отдельным папкам и файлам).

Помимо организационных мер для выявления программных закладок эффективными могут оказаться методы семантического анализа исполнимого кода системных и прикладных модулей с целью поиска небезопасных для КС участков и (или) недокументированных функций. Для этого должны применяться дисассемблирование кода и эмуляция его выполнения с помощью специальных отладчиков.

Эффективным методом защиты от вредоносных программ является создание *изолированной программной среды*, обладающей следующими свойствами:

- на компьютере с проверенной BIOS установлена проверенная операционная система;
- достоверно установлена целостность модулей операционной системы и BIOS для данного сеанса работы пользователя;
- исключен запуск любых программ в данной программно-аппаратной среде, кроме проверенных;
- исключен запуск проверенных программ вне проверенной среды их выполнения (т.е. в обход контролируемых проверенной средой событий).

Но в соответствии с комплексным подходом к обеспечению информационной безопасности (см. гл. 1) универсальных приемов, сохраняющих постоянную эффективность, быть не может. Требуется тщательный анализ новой информации о типах программных закладок и способах их внедрения в КС для выбора адекватных методов защиты.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие программы относят к разряду вредоносных?
2. Что такое компьютерный вирус?
3. Какие существуют виды компьютерных вирусов?
4. В чем разница между загрузочными и файловыми вирусами?

5. Как происходит заражение и функционирование загрузочных вирусов?

6. Какие типы файлов могут заражаться файловыми вирусами?

7. Как происходит заражение программных файлов?

8. Почему файлы документов могут содержать вирусы?

9. Как обеспечивается автоматическое получение управления макровирусами?

10. В чем особенность макровирусов в базах данных Microsoft Access?

11. Как включить встроенную защиту от вирусов в макросах в программах Microsoft Office? В чем недостатки этой защиты?

12. Какие существуют основные каналы заражения вирусами объектов компьютерной системы?

13. Какие существуют методы автоматического обнаружения и удаления вирусов? В чем их достоинства и недостатки?

14. В чем заключается профилактика заражения компьютерными вирусами?

15. Какие виды программных закладок существуют?

16. Как может происходить проникновение программной закладки в компьютерную систему?

17. Как осуществляется взаимодействие внедренной в КС программной закладки и нарушителя?

18. Какие существуют методы защиты от программных закладок? Что такое изолированная программная среда?

ГЛАВА 7

ЗАЩИТА ПРОГРАММНЫХ СРЕДСТВ ОТ НЕСАНКЦИОНИРОВАННОГО ИСПОЛЬЗОВАНИЯ И КОПИРОВАНИЯ

7.1. Принципы построения систем защиты от копирования

Дальнейшее развитие информационных технологий невозможно без создания новых программных средств различного назначения, баз данных, компьютерных средств обучения и других продуктов, предназначенных для корпоративного или персонального использования. При этом возникает проблема защиты авторских прав создателей и владельцев продуктов информационных технологий. Отсутствие такой защиты может привести к оттоку из сферы производства программного обеспечения части способных к творческой деятельности специалистов, снижению качества создаваемых информационных ресурсов и другим негативным социальным последствиям.

К сожалению, в настоящее время попытки нарушения авторских прав на объекты интеллектуальной собственности становятся достаточно регулярным и повсеместным явлением. Недостаточная эффективность правовых методов защиты интересов создателей и владельцев информационных ресурсов (см. подразд. 1.3) приводит к необходимости создания программных средств их защиты.

Под системой защиты от несанкционированного использования и копирования (защиты авторских прав, или просто защиты, от копирования) понимается комплекс программных или программно-аппаратных средств, предназначенных для усложнения или запрещения нелегального распространения, использования и (или) изменения программных продуктов и иных информационных ресурсов. Термин «нелегальное» здесь понимается как производимое без согласия правообладателя. Нелегальное изменение информационного ресурса может потребоваться нарушителю для того, чтобы измененный им продукт не подпадал под действие законодательства о защите авторских прав.

Под надежностью системы защиты от несанкционированного копирования понимается ее способность противостоять попыткам изучения алгоритма ее работы и обхода реализованных в нем методов защиты. Очевидно, что любая программная или программно-аппаратная система защиты от копирования может быть преодолена за конечное время, так как процессорные команды сис-

темы защиты в момент своего исполнения присутствуют в оперативной памяти компьютера в открытом виде. Также очевидно, что надежность системы защиты равна надежности наименее защищенного из ее модулей.

Выделим принципы создания и использования систем защиты от копирования.

1. Учет условий распространения программных продуктов:

• распространение дистрибутивных файлов на магнитных носителях через сеть торговых агентов или через сеть Интернет с последующей установкой самим пользователем, который при этом может пытаться копировать дистрибутивные магнитные диски, исследовать алгоритм работы системы защиты при помощи специальных программных средств (отладчиков и дисассемблеров), пытаться нарушить условия лицензионного соглашения и установить продукт на большем числе компьютеров, пытаться смоделировать алгоритм работы системы защиты для изготовления аналогичного варианта дистрибутивных файлов и распространения их от своего имени;

• установка программного продукта официальным представителем правообладателя, при котором пользователь может пытаться нарушить условия лицензионного соглашения или исследовать алгоритм работы системы защиты;

• приобретение и использование программного продукта лицами или организациями, не заинтересованными в его нелегальном распространении среди их коммерческих конкурентов — в этом случае возможны только попытки несанкционированного использования продукта другими лицами;

• приобретение программного продукта только для снятия с него системы защиты.

2. Учет возможностей пользователей программного продукта по снятию с него системы защиты (наличие достаточных материальных ресурсов, возможность привлечения необходимых специалистов и т. п.).

3. Учет свойств распространяемого программного продукта (предполагаемого тиража, оптовой и розничной цены, частоты обновления, специализированности и сложности продукта, уровня послепродажного сервиса для легальных пользователей, возможности применения правовых санкций к нарушителю и др.).

4. Оценка возможных потерь при снятии защиты и нелегальном использовании.

5. Учет особенностей уровня знаний и квалификации лиц, снимающих систему защиты.

6. Постоянное обновление использованных в системе защиты средств.

При добавлении к программному продукту системы его защиты от копирования возможен выбор уже имеющейся системы,

что минимизирует издержки на установку системы защиты. Однако имеющаяся система защиты от копирования будет более легко сниматься с программного продукта (в силу ее пристыкованности к нему, см. подразд. 1.5), а также может оказаться несовместимой с защищаемой программой и имеющимся у пользователя программно-аппаратным обеспечением. Поэтому более целесообразной является разработка специализированной системы защиты от копирования конкретного программного продукта, что, однако, более заметно увеличит затраты на его производство.

Основные требования, предъявляемые к системе защиты от копирования:

- обеспечение некопируемости дистрибутивных дисков стандартными средствами (для такого копирования нарушителю потребуется тщательное изучение структуры диска с помощью специализированных программных или программно-аппаратных средств);
- обеспечение невозможности применения стандартных отладчиков без дополнительных действий над машинным кодом программы или без применения специализированных программно-аппаратных средств (нарушитель должен быть специалистом высокой квалификации);
- обеспечение некорректного дисассемблирования машинного кода программы стандартными средствами (нарушителю потребуется использование или разработка специализированных дисассемблеров);
- обеспечение сложности изучения алгоритма распознавания индивидуальных параметров компьютера, на котором установлен программный продукт, и его пользователя или анализа применяемых аппаратных средств защиты (нарушителю будет сложно эмулировать легальную среду запуска защищаемой программы).

Выделим основные компоненты системы защиты программных продуктов от несанкционированного копирования:

- модуль проверки ключевой информации (некопируемой метки на дистрибутивном диске, уникального набора характеристик компьютера, идентифицирующей информации для легального пользователя) — может быть добавлен к исполнимому коду защищаемой программы по технологии компьютерного вируса (см. подразд. 6.2), в виде отдельного программного модуля или в виде отдельной функции проверки внутри защищаемой программы;
- модуль защиты от изучения алгоритма работы системы защиты;
- модуль согласования с работой функций защищаемой программы в случае ее санкционированного использования;
- модуль ответной реакции в случае попытки несанкционированного использования (как правило, включение такого модуля в состав системы защиты нецелесообразно по морально-этическим соображениям).

7.2. Методы защиты инсталляционных дисков от копирования

Основным методом защиты инсталляционных дисков, содержащих дистрибутивные файлы, которые используются для установки программы на компьютере пользователя, является нанесение на инсталляционный диск некопируемой метки.

Некопируемая метка — совокупность информационных характеристик магнитного носителя, существенно изменяющаяся при его копировании. Возможно нанесение магнитной или физической некопируемой метки.

К основным способам нанесения магнитной некопируемой метки относятся:

- вынос метки за пределы стандартного поля копирования информации на магнитном носителе;
- нестандартная разметка одной или нескольких дорожек диска;
- привязка к временным характеристикам чтения и записи информации с магнитного носителя;
- комбинирование нескольких способов.

Рассмотрим способ, связанный с выносом магнитной метки за пределы стандартного поля копирования, на примере магнитной дискеты емкостью 1,44 мегабайт. Каждая из двух сторон такой дискеты содержит 80 дорожек по 18 секторов каждая (размер одного сектора равен 512 байт). Нумерация сторон (головок) дискеты ведется от 0 до 1, нумерация дорожек (цилиндров) — от 0 до 79 и нумерация секторов — от 1 до 18. Сущность данного способа состоит в форматировании дорожки с номером, большим 79, и записи в один из ее секторов ключевой информации (например, максимально возможного числа инсталляций, определяемого лицензией на использование программного продукта).

В операционной системе Windows для выполнения форматирования дорожек дискеты может использоваться следующая функция:

```
BOOL DeviceIoControl(HANDLE hDevice, DWORD dwIoControlCode,
LPVOID lpInBuffer, DWORD nInBufferSize, LPVOID
lpOutBuffer, DWORD nOutBufferSize, LPDWORD
lpBytesReturned, LPOVERLAPPED lpOverlapped); /* форматирование дорожек на устройстве hDevice,
dwIoControlCode=IOCTL_DISK_FORMAT_TRACKS, буфер
lpInBuffer длиной nInBufferSize должен содержать параметры форматирования (см. далее), lpOutBuffer=NULL,
nOutBufferSize=0, lpBytesReturned =NULL, lpOverlapped=
NULL при ожидании функцией DeviceIoControl завершения
форматирования */
```

Дескриптор устройства (точнее, его драйвера) hDevice должен быть получен с помощью функции CreateFile. Параметры форматирования должны быть представлены структурой типа FORMAT_PARAMETERS, имеющей следующие поля:

- MediaType — тип дискеты (например, F3_1Pt44_512);
- StartCylinderNumber — адрес начального форматируемого цилиндра (дорожки);
- EndCylinderNumber — адрес конечного форматируемого цилиндра (дорожки);
- StartHeadNumber — адрес начальной головки (стороны);
- EndHeadNumber — адрес конечной головки (стороны).

В защищенных версиях операционной системы Windows для получения дескриптора драйвера дисковода на гибких дисках первый параметр при вызове функции CreateFile должен быть равен "\\\\.\\A:". В открытых версиях Windows для получения дескриптора драйвера необходимо вызвать функцию CreateFile с первым параметром, равным "\\\\.\\VWIN32.VXD". В открытых версиях Windows для форматирования на дискете дополнительной дорожки, записи на нее ключевой информации и чтения ключевой информации может также использоваться функция DeviceIoControl с кодом подфункции (значением параметра dwIoControlCode), равным 1. При этом параметры lpInBuffer и lpOutBuffer должны указывать на структуру DIOC_REGISTERS, содержащую значения регистров процессора до и после выполнения операции с диском (подфункции 0Dh функции 44h прерывания 21h).

Перед выполнением операции регистр ah должен содержать 44h, al — 0Dh, bl — 1 (дисковод a:), ch — 08h (дисковое устройство), cl — код операции с диском (см. далее), ds:dx — адрес используемого при выполнении операции блока параметров. Порядок выполнения операций с диском при этом должен быть следующим:

1) получение текущих параметров диска (код операции 60h);
2) изменение полученных параметров и установка новых значений параметров диска (код операции 40h);
3) проверка возможности форматирования дополнительной дорожки с заданным номером (код операции 42h);
4) форматирование дорожки (код операции 42h);
5) вместо пп. 3 и 4 возможно выполнение записи произвольных данных в указанные сектора заданной дорожки (код операции 41h) или чтения данных (код операции 61h).

Основной изменяемой перед форматированием частью блока параметров диска является его таблица разметки, размещающаяся в блоке параметров со смещением 38 байт и содержащая в себе общее число секторов на дорожке (поле длиной 2 байта) и для каждого сектора на дорожке — его номер (2 байта) и длину (2 байта).

Если перед выполнением записи или чтения дорожки диска текущие значения его параметров не изменяются, то пп. 1 и 2 могут также не выполняться.

При использовании способа создания некопируемой метки, связанного с нестандартным форматированием дорожки дискеты, выполняются следующие действия:

• перед записью некопируемой метки для выбранной дорожки устанавливается новая таблица разметки (например, предусматривающая на дорожке один сектор длиной 8192 байта) и дорожка форматируется;

• перед чтением некопируемой метки также устанавливается новая таблица разметки, соответствующая использованной при записи ключевой информации.

Способ создания некопируемой метки, опирающийся на временные характеристики чтения и записи с диска, использует нестандартную нумерацию секторов на выбранной дорожке (не 1, 2, 3, ..., 18, а 18, 17, 16, ..., 1).

Перед форматированием ключевой дорожки в этом случае также устанавливается блок параметров диска с новой таблицей разметки. При проверке ключевой дискеты выполняются следующие действия:

1) чтение первого сектора ключевой дорожки (для сохранения значения системного таймера);

2) многократное (50—100 раз) чтение секторов с номерами 1 и 2 (на оригинальной ключевой дискете они находятся на максимальном удалении друг от друга);

3) сохранение значения системного таймера;

4) повторение пп. 1...3 для обычной дорожки;

5) сравнение двух временных интервалов, полученных при чтении ключевой и обычной дорожек диска (для оригинальной ключевой дискеты эти времена существенно различаются).

Способ получения физической некопируемой метки основан на повреждении (например, лазерным лучом) небольшой части поверхности диска. После этого попытка чтения или записи поврежденных участков будет приводить к ошибке устройства только на оригинальном ключевом диске.

Порядок действий при использовании физической метки может быть следующим:

1) форматирование ключевого диска;

2) нанесение на нем физической метки;

3) определение (путем последовательного выполнения операций записи и чтения в каждый сектор диска) конкретного адреса поврежденного участка и его сохранение;

4) при проверке легальности установки программного продукта в сектора диска с сохраненными адресами делается попытка записи и чтения произвольных данных.

К недостаткам рассмотренных способов получения некопируемой метки относится их широкая известность, что облегчает нарушителям поиск противодействия (например, при использовании физической метки возможно моделирование повреждения участка дисковой поверхности с проверяемым программой установки адресом).

Обычно при установке защищенного от копирования программного продукта выполняются следующие действия:

1) запуск программы установки с инсталляционного диска;

2) попытка чтения некопируемой метки с инсталляционного диска и, возможно, ее изменение (например, уменьшение на единицу числа разрешенных инсталляций) или проверка введенной пользователем ключевой информации (например, серийного номера диска и кода аутентификации установки);

3) если метка с ключевой информацией не может быть прочитана, или число возможных установок равно нулю, или введенная пользователем ключевая информация некорректна, то завершение работы программы установки;

4) если среда установки программного продукта является легальной, то копирование (с возможной распаковкой) файлов программного продукта на жесткий диск компьютера пользователя, внесение изменений в системный реестр и выполнение других необходимых при инсталляции продукта действий;

5) настройка устанавливаемого программного продукта на характеристики компьютера пользователя для невозможности его несанкционированного использования на другом компьютере и (или) другим пользователем.

7.3. Методы настройки устанавливаемого программного обеспечения на характеристики компьютера

Настройка устанавливаемого программного обеспечения на характеристики компьютера пользователя заключается в определении максимально полного набора таких характеристик, их хешировании, получении электронной цифровой подписи с помощью секретного ключа пользователя (см. подразд. 4.7) и записи ЭЦП в реестр операционной системы в раздел с настройками текущего пользователя.

Характеристиками компьютера, на которые может быть выполнена настройка устанавливаемого программного обеспечения, являются:

- имя компьютера;
- имя пользователя;
- версия операционной системы;
- параметры центрального процессора;

- параметры оперативной памяти;
- тип используемой клавиатуры;
- параметры используемой мыши;
- ширина и высота экрана монитора;
- информация о дисковых устройствах компьютера;
- параметры диска, на котором выполняется установка программного продукты (емкость, тип файловой системы, серийный номер, метка тома);
- путь к папкам с файлами операционной системы и др.

Для получения значений указанных характеристик могут использоваться следующие функции из набора Windows API:

- `BOOL GetUserName(LPTSTR lpBuffer, LPDWORD nSize);` `/* получение в буфере lpBuffer длины nSize имени пользователя текущего сеанса */`
- `BOOL GetComputerName(LPTSTR lpBuffer, LPDWORD nSize);` `/* получение имени компьютера в буфере lpBuffer длины nSize>= MAX_COMPUTERNAME_LENGTH+1 */`
- `UINT GetWindowsDirectory(LPTSTR lpBuffer,UINT uSize);` `/* получение в буфере lpBuffer длины uSize>=MAX_PATH пути к каталогу с ОС Windows */`
- `UINT GetSystemDirectory(LPTSTR lpBuffer, UINT uSize);` `/* получение в буфере lpBuffer длины uSize>=MAX_PATH пути к системному каталогу Windows */`
- `int GetKeyboardType(int nTypeFlag);` `/* получение типа (nTypeFlag=0) или подтипа (nTypeFlag=1) клавиатуры */`
- `int GetSystemMetrics(int nIndex);` `/* получение количества кнопок мыши (nIndex=SM_CMOUSEBUTTONS), ширины (nIndex=SM_CXSCREEN) или высоты (nIndex=SM_CYSCREEN) экрана */`
- `DWORD GetLogicalDriveStrings(DWORD nBufferLength, LPTSTR lpBuffer);` `/* получение в буфере lpBuffer длины nBufferLength строки с корневыми каталогами всех дисков, разделенных 0-символами; результат – длина полученной строки без заключительного 0-символа */`
- `VOID GlobalMemoryStatus(LPMEMORYSTATUS lpBuffer);` `/* получение в буфере *lpBuffer структуры типа MEMORYSTATUS с характеристиками памяти компьютера (поле dwTotalPhys содержит целое число, равное общему объему физической памяти в байтах) */`
- `BOOL GetDiskFreeSpace(LPCTSTR lpRootPathName, LPDWORD lpSectorsPerCluster, LPDWORD lpBytesPerSector, LPDWORD lpNumberOfFreeClusters, LPDWORD lpTotalNumberOfClusters);` `/* получение информации об объеме текущего диска (lpRootPathName=NULL): количестве секторов в кластере`

(lpSectorsPerCluster), *размере сектора* (lpBytesPerSector), *общем количестве кластеров* (lpTotalNumberOfClusters), lpNumberOfFreeClusters=NULL */

- BOOL GetVolumeInformation(LPCTSTR lpRootPathName, LPTSTR lpVolumeNameBuffer, DWORD nVolumeNameSize, LPDWORD lpVolumeSerialNumber, LPDWORD lpMaximum ComponentLength, LPDWORD lpFileSystemFlags, LPTSTR lpFileSystemNameBuffer, DWORD nFileSystemNameSize); /* *получение информации о текущем диске* (lpRootPathName=NULL): *метке тома* (*в буфере* lpVolumeNameBuffer *длины* nVolumeNameSize), *серийном номере* (*в переменной* *lpVolumeSerialNumber), *файловой системе* (*в буфере* lpFileSystemNameBuffer *длины* nFileSystemNameSize), lpMaximumComponentLength=NULL, lpFileSystemFlags=NULL */

- BOOL SystemParametersInfo(UINT uiAction, UINT uiParam, PVOID pvParam, UINT fWinIni); /* *получение в буфере* pvParam *значений системных параметров длиной* uiParam; *набор запрашиваемых параметров определяется в* uiAction; fWinIni=0 */

Приведем пример фрагмента программы, вычисляющей ЭЦП для значений набора характеристик используемого для установки программного продукта компьютера и записывающего ЭЦП в реестр Windows. Для вычисления ЭЦП используются функции криптографического интерфейса приложений Windows (см. гл. 5), а для работы с реестром — классы библиотеки Visual Component Library системы программирования Borland C++ Builder.

```
/* подключение заголовочного файла с определением классов
для работы с системным реестром */
#include <vcl\registry.hpp>
/* глобальная переменная для имени раздела реестра с
ЭЦП */
AnsiString RegName;
/* функция вычисления хеш-значения характеристик ком-
пьютера с помощью криптопровайдера hProv */
HCRYPTHASH HashCompParams(HCRYPTPROV hProv)
{HCRYPTHASH hHash; // дескриптор хеш-значения
// буфер для имени компьютера и пользователя
char Buf1[MAX_COMPUTERNAME_LENGTH+1];
// буфер для пути к папкам операционной системы
char Buf2 [MAX_PATH];
int Data; // запрашиваемые данные
unsigned long Len; // длина запрашиваемых данных
// создание пустого хеш-значения
CryptCreateHash(hProv, CALG_SHA, 0, 0, &hHash);
```

```
// хеширование имени пользователя
Len=sizeof(Buf1);
if(GetUserName(Buf1, &Len))
CryptHashData(hHash, Buf1, lstrlen(Buf1), 0);
// хеширование имени компьютера
Len=sizeof(Buf1);
if(GetComputerName(Buf1, &Len))
CryptHashData(hHash, Buf1, lstrlen(Buf1), 0);
/* хеширование пути к папке с файлами операционной
системы */
if(GetWindowsDirectory(Buf2, sizeof(Buf2)))
CryptHashData(hHash, Buf2, lstrlen(Buf2), 0);
// хеширование пути к папке с системными файлами
if(GetSystemDirectory(Buf2,sizeof(Buf2)))
CryptHashData(hHash,Buf2,lstrlen(Buf2),0);
/* хеширование значений параметров мыши и экрана мони-
тора */
Data=GetSystemMetrics(SM_CMOUSEBUTTONS);
if(Data) CryptHashData(hHash, (BYTE*)&Data,
sizeof(Data), 0);
Data=GetSystemMetrics(SM_CYSCREEN);
if(Data) CryptHashData(hHash, (BYTE*)&Data,
sizeof(Data), 0);
// хеширование информации о дисковых устройствах
Data=GetLogicalDriveStrings(sizeof(Buf2), Buf2);
if(Data) CryptHashData(hHash, Buf2, Data, 0);
if(GetVolumeInformation(NULL, NULL, 0, NULL, NULL, NULL,
Buf2,
sizeof(Buf2))) CryptHashData(hHash, Buf2, lstrlen(Buf2),
0);
return hHash; }
...

TRegistry *RegKey=new TRegistry; /* объект класса раз-
дела реестра */
HCRYPTHASH hHash; // дескриптор хеш-значения
HCRYPTPROV hProv; // дескриптор криптопровайдера
HCRYPTKEY hSignKey; // дескриптор ключа ЭЦП
BYTE Sign[MAX_PATH]; // буфер для ЭЦП
DWORD SigLen=sizeof(Sign); // длина ЭЦП
// инициализация криптопровайдера
if(!CryptAcquireContext(&hProv, NULL, NULL,
PROV_RSA_FULL, 0))
if((unsigned)GetLastError()==NTE_BAD_KEYSET)
CryptAcquireContext(&hProv, NULL, NULL, PROV_RSA_FULL,
CRYPT_NEWKEYSET);
```

```
else throw Exception("Ошибка при инициализации
CryptoAPI!");
// создание раздела реестра
RegKey→OpenKey(RegKey→CurrentPath+"\\SOFTWARE\\"+RegName,
true);
// вычисление хеш-значения характеристик компьютера
hHash=HashCompParams(hProv);
// создание ключа ЭЦП
if(!CryptGetUserKey(hProv, AT_SIGNATURE, &hSignKey))
CryptGenKey(hProv,AT_SIGNATURE,0,&hSignKey);
// получение ЭЦП
CryptSignHash(hHash, AT_SIGNATURE, NULL, 0, Sign,
&SigLen);
// запись ЭЦП в реестр как значения параметра Signature
RegKey→WriteBinaryData("Signature", (void*)Sign,
SigLen);
RegKey→CloseKey();
// разрушение объектов криптопровайдера
CryptDestroyHash(hHash);
CryptDestroyKey(hSignKey);
CryptReleaseContext(hProv, 0);
// уничтожение объекта класса раздела реестра
delete RegKey;
// продолжение работы программы установки
...
```

Теперь приведем фрагмент защищаемой от несанкционированного использования и копирования программы, проверяющий легальность среды запуска программы:

```
/* подключение заголовочного файла с определением классов
для работы с системным реестром */
#include <vcl\registry.hpp>
/* глобальная переменная для имени раздела реестра с
ЭЦП */
AnsiString RegName;
/* функция вычисления хеш-значения характеристик ком-
пьютера с помощью криптопровайдера hProv */
HCRYPTHASH HashCompParams(HCRYPTPROV hProv)
{ ... }
HCRYPTPROV hProv=0; // дескриптор криптопровайдера
HCRYPTHASH hHashComp=0; // дескриптор хеш-значения
HCRYPTKEY hPubKey=0; // дескриптор открытого ключа ЭЦП
BYTE Sign[MAX_PATH]; // буфер для ЭЦП
TRegistry *RegKey=new TRegistry; /* объект класса раз-
дела реестра*/
```

```
try
{ // проверка существования раздела реестра с ЭЦП
if(!RegKey→OpenKey(RegKey→CurrentPath +"SOFTWARE\\"+
RegName, false))
throw Exception("Несанкционированный запуск програм-
мы!");
// чтение ЭЦП с получением ее длины в Res
int Res=RegKey→ReadBinaryData("Signature", Sign,
sizeof(Sign));
// проверка успешности чтения ЭЦП из реестра
if(!Res) throw Exception("Несанкционированный запуск
программы!");
/* получение дескриптора криптопровайдера с проверкой
регистрации в нем пользователя текущего сеанса */
if(!CryptAcquireContext(&hProv, NULL, NULL,
PROV_RSA_FULL, 0))
throw Exception("Несанкционированный запуск програм-
мы!");
/* проверка создания ключей ЭЦП для текущего пользова-
теля */
if(!CryptGetUserKey(hProv, AT_SIGNATURE, &hPubKey))
throw Exception("Несанкционированный запуск програм-
мы!");
/* вычисление хеш-значения характеристик компьютера,
на котором производится запуск защищенной программы */
hHashComp=HashCompParams(hProv);
// проверка ЭЦП
if(!CryptVerifySignature(hHashComp, Sign, Res, hPubKey,
NULL, 0))
throw Exception("Несанкционированный запуск програм-
мы!");
/* продолжение работы программы в случае правильности
ЭЦП */
...
```

К недостаткам рассмотренных в данном разделе методов настройки устанавливаемого программного продукта на характеристики компьютера относится возможность изменения части указанных характеристик при обновлении программно-аппаратного обеспечения компьютера, на котором был установлен продукт. В этом случае могут потребоваться удаление и повторная установка защищенной программы.

Если при ее инсталляции с защищенного дистрибутивного диска изменялось значение счетчика возможных установок, то потребуется также применение специальной программы деинсталляции защищенного программного продукта.

7.4. Методы противодействия исследованию алгоритма работы системы защиты

Для противодействия динамическому исследованию алгоритма работы системы защиты программного обеспечения от копирования с помощью отладчиков и программ мониторинга состояния вычислительной системы типа FileMon и RegMon (см. подразд. 6.4) прежде всего необходимо обнаружить присутствие в системе данных программ. Для этого могут применяться различные способы:

- использование функции из набора Windows API для перебора всех окон активных процессов

```
BOOL EnumWindows(WNDENUMPROC lpEnumFunc, LPARAM lParam);
/* вызов функции lpEnumFunc с передачей ей параметра
lParam для всех окон верхнего уровня активных процессов */
```

- использование функции из набора Windows API для защищенных версий операционной системы Windows для обнаружения факта запуска текущего процесса под отладчиком

```
BOOL IsDebuggerPresent(VOID);
```

- измерение времени выполнения критических для системы защиты участков кода (при работе под отладчиком это время будет существенно большим);
- проверка установки точек прерывания на функции из набора Windows API или функции системы защиты (первый байт кода функции в этом случае будет содержать значение 0xCC);
- изучение системного окружения с помощью функции из набора Windows API для получения значений возможно установленных отладчиком переменных среды

```
DWORD GetEnvironmentVariable(LPCTSTR lpName, LPTSTR
lpBuffer, DWORD nSize); /* получение в буфере lpBuffer
длины nSize значения переменной среды с именем lpName;
при успешном завершении возвращается длина полученного
значения */
```

При обнаружении факта работы защищаемой программы под управлением отладчика возможны следующие реакции системы защиты:

- искажение информации в стеке или вызов системной функции DestroyWindow для главного окна обнаруженного отладчика;
- некорректное взаимодействие с отладчиком по протоколам DDE (Dynamic Data Exchange, динамический обмен данными между процессами) или COM (Component Object Model, модель взаимодействия многокомпонентных объектов) для вывода отладчика из

строя (передача «мусора» или преждевременный вызов метода Release);

• отключение средств защиты для скрытия их присутствия в защищаемой программе и усыпления бдительности нарушителя (работа средств защиты будет производиться только при использовании защищаемой программы без отладчика).

Для защиты от статического изучения алгоритма получения и проверки ключевой информации о среде запуска защищаемой программы с помощью дисассемблеров обычно применяется самомодифицирующийся код, что требует использования при разработке системы защиты языка ассемблер. Если это невозможно, то применяются методы изощренного программирования, направленные на максимальное усложнение понимания алгоритма работы системы защиты путем максимального запутывания ее кода.

Для усложнения понимания алгоритма преобразования полученной ключевой информации при ее проверке могут применяться следующие способы:

• использование асимметричного шифрования (см. подразд. 4.6):

1) пользователь сообщает правообладателю персональные данные;

2) правообладатель шифрует их с помощью своего секретного ключа и возвращает пользователю вместе с программой и своим открытым ключом;

3) при запуске программы система защиты расшифровывает с помощью полученного открытого ключа данные пользователя и сравнивает их с введенными им данными;

• использование при проверке ключевой информации функции хеширования (см. подразд. 4.9);

• использование нейронной сети обратного распространения, обученной на аппроксимацию функции преобразования ключевых данных;

• использование при преобразовании ключевых данных функции F, являющейся суперпозицией функций H_1 и H_2, и аппроксимация в системе защиты функций

$$Z_1 = H_1(X) \text{ и } Z_2 = H_2^{-1}(Y),$$

где X — введенные пользователем данные; Y — значение ключа аутентификации (совпадение значений Z_1 и Z_2 означает легальность среды запуска защищаемой программы).

Перечисленные способы основаны на особенностях квалификации потенциального нарушителя, который, как правило, хорошо знает системное программирование, но гораздо хуже — математику и криптографию. При использовании последнего из перечисленных способов защиты затрудняется и простой перебор ключей аутентификации при снятии системы защиты нарушителем.

Для защиты от изучения алгоритма работы фрагмента кода демонстрационных версий защищаемой программы, в котором происходит сравнение текущей даты с датой окончания использования демонстрационной версии, можно дополнительно сохранять дату установки программы или ее последнего запуска в реестре операционной системы в открытом или зашифрованном виде. При этом возможно использование следующей функции из набора Windows API:

```
LONG RegQueryInfoKey (HKEY hKey, LPTSTR lpClass, LPDWORD
lpcbClass, LPDWORD lpReserved, LPDWORD lpcSubKeys,
LPDWORD lpcbMaxSubKeyLen, LPDWORD lpcbMaxClassLen,
LPDWORD lpcValues, LPDWORD lpcbMaxValueNameLen, LPDWORD
lpcbMaxValueLen, LPDWORD lpcbSecurityDescriptor,
PFILETIME lpftLastWriteTime); /* получение информации
о разделе реестра hKey: имени класса раздела в буфере
lpClass длины lpcbClass, количестве подразделов
lpcSubKeys, максимальной длине имени подраздела
lpcbMaxSubKeyLen и его класса lpcbMaxClassLen, количе-
стве параметров lpcValues и максимальной длине имени
параметра lpcbMaxValueNameLen, максимальной длине зна-
чения параметра lpcbMaxValueLen, длине дескриптора бе-
зопасности (см. подразд. 3.3) lpcbSecurityDescriptor и
последнего времени записи в раздел lpftLastWriteTime
(только для защищенных версий Windows); значение пара-
метра lpReserved не используется и должно быть равно
нулю */
```

Для невозможности модификации нарушителем кода системы защиты с целью обхода блока проверки ключевой информации применяется подсчет и проверка хеш-значений критически важных фрагментов кода.

В заключение назовем приемы изощренного программирования, направленные на максимальное усложнение понимания алгоритма работы системы защиты:

- использование управляемого событиями алгоритма проверки (разбиение ее на несколько частей);
- использование системных ресурсов (памяти окна приложения, атомов и т. п.) для хранения проверяемых данных;
- нестандартная проверка (например, при получении правильных данных получается отрицательное контрольное значение, из которого извлекается квадратный корень, а сама проверка выносится в функцию обработки ошибок выполнения математических операций);
- имитация проверки ключевых данных в различных местах программы;
- включение в состав системы защиты нескольких различных функций с одинаковым действием;

• создание в коде системы защиты нескольких различных переменных, получающих различные значения для обозначения одного и того же промежуточного результата проверки;

• использование косвенной адресации при доступе к переменным с проверочной информацией;

• вынесение процедур проверки в обработку стандартных сообщений Windows, например типа WM_PAINT (подмена идентификатора сообщения может производиться непосредственно в цикле обработки сообщений);

• при возможности проверка базы легальных пользователей защищаемого программного продукта по сети Интернет на узле правообладателя.

К наиболее надежным методам защиты от несанкционированного копирования программных продуктов относится использование специальных электронных ключей, присоединяемых к компьютеру пользователя при помощи USB-порта. Подобная программно-аппаратная система защиты работает следующим образом:

• приложение настраивается правообладателем на характеристики электронного ключа при помощи специального программного обеспечения;

• при работе защищаемой программы она обменивается с электронным ключом аутентифицирующей информацией, подтверждающей подлинность ключа;

• при отсутствии ключа или наличия у него иных характеристик защищаемое приложение не может быть использовано.

Недостатком данного метода защиты от копирования является увеличение стоимости для пользователя защищенной программы, что может оказаться неприемлемым для отдельных классов программных средств (например, игровых программ, учебных программ и т. п.).

Применение даже всей совокупности рассмотренных в данной главе методов и средств не позволит создать совершенную систему защиты от несанкционированного использования и копирования программ. Любая подобная система будет снята, если нарушитель обладает неограниченными временными и материальными ресурсами.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что называется защитой программных продуктов от несанкционированного копирования?

2. На каких принципах должна основываться разработка системы защиты от копирования?

3. Какие требования предъявляются к системам защиты от копирования?

4. Из каких основных компонентов состоит типовая система защиты от копирования?

5. Какие методы могут применяться для защиты инсталляционных дисков? Что называется некопируемой меткой?

6. В чем заключаются недостатки методов защиты инсталляционных дисков?

7. Какие действия выполняются программой установки продукта, защищенного от копирования?

8. Для чего производится настройка устанавливаемого программного продукта на характеристики используемого компьютера?

9. Какие характеристики компьютера могут использоваться для настройки защищаемого программного продукта?

10. Какие инструментальные средства используются для изучения алгоритма работы системы защиты от копирования?

11. Какие методы могут использоваться для защиты программных средств от изучения алгоритмов их работы?

12. Для чего предназначены методы изощренного программирования и почему они применяются в системах защиты от копирования?

13. В чем заключаются достоинства и недостатки программно-аппаратной защиты от копирования на основе электронных ключей?

14. Почему невозможно создание абсолютно надежной системы защиты от копирования?

СПИСОК ЛИТЕРАТУРЫ

1. *Анин Б. Ю.* Защита компьютерной информации. — СПб.: БХВ-Санкт-Петербург, 2000.

2. *Баричев С. Г., Гончаров В. В., Серов Р. Е.* Основы современной криптографии. — М.: Горячая линия-Телеком, 2001.

3. *Барсуков В. С., Водолазский В. В.* Современные технологии безопасности. — М.: Нолидж, 2000.

4. Безопасность компьютерных сетей на основе Windows NT / В. С. Люцарев, К. В. Ермаков, Е. Б. Рудный, И. В. Ермаков. — М.: Издательский отдел «Русская редакция» ТОО «Channel Trading Ltd.», 1998.

5. Государственная техническая комиссия при Президенте Российской Федерации. Сборник руководящих документов по защите информации от несанкционированного доступа. — М.: СИП РИА, 1997.

6. *Грушо А. А., Тимонина Е. Е.* Теоретические основы защиты информации. — М.: Издательство агентства «Яхтсмен», 1996.

7. *Зензин О. С., Иванов М. А.* Стандарт криптографической защиты — AES. Конечные поля. — М.: КУДИЦ-ОБРАЗ, 2002.

8. *Зима В. М., Молдовян А. А., Молдовян Н. А.* Безопасность глобальных сетевых технологий. — СПб.: БХВ-Петербург, 2000.

9. *Касперский Е.* Компьютерные вирусы в MS-DOS. — М.: Издательство «ЭДЭЛЬ», 1992.

10. *Курушин В. Д., Минаев В. А.* Компьютерные преступления и информационная безопасность. — М.: Новый Юрист, 1998.

11. *Малюк А. А., Пазизин С. В., Погожин Н. С.* Введение в защиту информации в автоматизированных системах. — М.: Горячая линия-Телеком, 2001.

12. *Мафтик С.* Механизмы защиты в сетях ЭВМ. — М.: Мир, 1993.

13. *Медведовский И. Д., Семьянов П. В., Леонов Д. Г.* Атака на Internet. — М.: ДМК, 1999.

14. Программирование алгоритмов защиты информации / А. В. Домашев, В. О. Попов, Д. И. Правиков и др. — М.: Нолидж, 2000.

15. Программно-аппаратные средства обеспечения информационной безопасности. Защита программ и данных / П. Ю. Белкин, О. О. Михальский, А. С. Першаков и др. — М.: Радио и связь, 1999.

16. *Проскурин В. Г., Крутов С. В., Мацкевич И. В.* Программно-аппаратные средства обеспечения информационной безопасности. Защита в операционных системах. — М.: Радио и связь, 2000.

17. *Расторгуев С. П.* Программные методы защиты информации в компьютерах и сетях. — М.: Издательство агентства «Яхтсмен», 1993.

18. *Романец Ю. В.*, *Тимофеев П. А.*, *Шаньгин В. Ф.* Защита информации в компьютерных системах и сетях. — М.: Радио и связь, 1999.

19. *Смит Р. Э.* Аутентификация: от паролей до открытых ключей. — М.: Издательский дом «Вильямс», 2002.

20. *Станек У. Р.* Microsoft Windows XP Professional. Справочник администратора. — М.: Издательско-торговый дом «Русская редакция», 2003.

21. *Столингс В.* Основы защиты сетей. Приложения и стандарты. — М.: Издательский дом «Вильямс», 2002.

22. *Такет (мл.) Дж.*, *Барнет С.* Использование Linux. Специальное издание. — М.: Издательский дом «Вильямс», 2000.

23. Теоретические основы компьютерной безопасности / П. Н. Девянин, О. О. Михальский, Д. И. Правиков, А. Ю. Щербаков. — М.: Радио и связь, 2000.

24. *Фролов А. В.*, *Фролов Г. В.* Осторожно: компьютерные вирусы. — М.: ДИАЛОГ-МИФИ, 1996.

25. *Халяпин Д. Б.* Защита информации. Вас подслушивают? Защищайтесь! — М.: НОУ ШО «Баярд», 2004.

26. *Щербаков А. Ю.* Защита от копирования. — М.: Издательство «ЭДЭЛЬ», 1992.

27. *Щербаков А. Ю.* Разрушающие программные воздействия. — М.: Издательство «ЭДЭЛЬ», 1993.

28. *Щербаков А. Ю.*, *Домашев А. В.* Прикладная криптография. Использование и синтез криптографических интерфейсов. — М.: Издательско-торговый дом «Русская редакция», 2003.

ОГЛАВЛЕНИЕ

Учебное издание

Хорев Павел Борисович

Методы и средства защиты информации в компьютерных системах

Учебное пособие

4-е издание, стереотипное

Редакторы *И. В. Могилевец, И. В. Мочалова*
Технический редактор *Н. И. Горбачева*
Компьютерная верстка: *О. В. Пешкетова*
Корректор *Т. В. Кузьмина*

Изд. № 104106748. Подписано в печать 25.03.2008. Формат 60×90/16.
Гарнитура «Таймс». Печать офсетная. Бумага офсетная № 1. Усл. печ. л. 16,0.
Тираж 2000 экз. Заказ № 26272.

Издательский центр «Академия». www.academia-moscow.ru
Санитарно-эпидемиологическое заключение № 77.99.02.953.Д.004796.07.04 от 20.07.2004.
117342, Москва, ул. Бутлерова, 17-Б, к. 360. Тел./факс: (495)330-1092, 334-8337.

Отпечатано в ОАО «Саратовский полиграфкомбинат».
410004, г. Саратов, ул. Чернышевского, 59. www.sarpk.ru

МЕТОДЫ И СРЕДСТВА ЗАЩИТЫ ИНФОРМАЦИИ В КОМПЬЮТЕРНЫХ СИСТЕМАХ

ISBN 5-7695-1839-1

Издательский центр «Академия»