МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

КУРГАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра “Безопасность информационных и автоматизированных систем”

ИСПОЛЬЗОВАНИЕ ФУНКЦИЙ КРИПТОГРАФИЧЕСКОГО ИНТЕРФЕЙСА WINDOWS ДЛЯ ЗАЩИТЫ ИНФОРМАЦИИ

Методические указания
к выполнению лабораторной работы
по дисциплине «Теоретические основы компьютерной безопасности»
для студентов специальностей 090105, 090303

Курган 2012

Кафедра: «Безопасности информационных и автоматизированных систем»

Дисциплина: «Теоретические основы компьютерной безопасности» (специальности 090105, 090303)

Составил ст. преподаватель В.В.Москвин

Утверждены на заседании кафедры « 23 »декабря 2011 г.

Рекомендованы методическим советом университета « 28 » декабря 2011 г.

ЦЕЛЬ РАБОТЫ

Изучить системы защиты с использованием криптографического интерфейса операционной системы Windows CryptoAPI с использованием сеансового ключа, генерируемого на основе вводимой администратором (пользователем) парольной фразы.

ПРИБОРЫ, ОБОРУДОВАНИЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

1 Операционная система (по выбору).
2 Используемая среда программирования (по выбору).

ТЕОРЕТИЧЕСКОЕ ВВЕДЕНИЕ

В настоящее время многие приложения используют для обмена данными открытые каналы связи, и прежде всего Internet. Однако в ряде случаев, например, в банковской и финансовой сферах, требуется ограничить доступ к конфиденциальной информации, передаваемой по таким каналам. При этом важно иметь возможность проверить, от кого исходят принятые получателем конфиденциальные данные и не были ли они искажены при пересылке. Для безопасной передачи конфиденциальных данных по открытым каналам используется криптография. В сфере защиты компьютерной информации криптография применяется в основном для: шифрования и дешифровки данных; а также создания и проверки цифровых подписей. Шифрование данных позволяет ограничить доступ к конфиденциальной информации, сделать ее нечитаемой и непонятной для посторонних. Применение цифровых подписей оставляет данные открытыми, но дает возможность верифицировать отправителя и проверять целостность полученных данных.

Криптографическая защита информации

Криптографическая защита информации на прикладном уровне является наиболее предпочтительным вариантом защиты информации с точки зрения гибкости защиты, но наиболее сложным по программно-аппаратной реализации.

Криптографическая защита информации на прикладном уровне (криптографическая защита прикладного уровня) - это такой порядок проектирования, реализации и использования криптографических средств, при котором входная и выходная информация и, возможно, ключевые параметры принадлежат потокам и объектам прикладного уровня (в модели ISO взаимодействия открытых систем).

Информация, находящаяся на нижестоящих иерархических уровнях модели ISO (далее используется термин «нижестоящие уровни») относительно объекта прикладного уровня представляет собой **подобъекты данного объекта**, **рассматриваемые,** как правило, **изолированно друг от друга**. В связи с этим на нижестоящих уровнях (сетевом и ниже) невозможно достоверно

распознавать, а, следовательно, и защищать криптографическими методами объекты сложной структуры типа «электронный документ» или поле базы данных. На нижестоящих уровнях данные объекты представляются последовательностью вмещающих подобъектов типа «пакет».

Кроме того, только на прикладном уровне возможна **персонализация** объекта, т.е. однозначное сопоставление созданного объекта породившему его субъекту (субъектом прикладного уровня является, как правило, прикладная программа, управляемая человеком-пользователем). Субъекты нижестоящих уровней снабжают атрибутами порождаемую ими последовательность объектов (подобъектов объекта прикладного уровня) в основном адресом (информацией, характеризующей субъекта нижестоящего уровня - компьютер), который является лишь опосредованной характеристикой породившего информацию субъекта прикладного уровня.

В то же время необходимо отметить свойство **наследования логической защиты вышестоящего уровня для нижестоящих**.

Поясним данное свойство примером. Предположим, на прикладном уровне зашифровано поле базы данных, при передаче информации по сети происходит ее преобразование на нижестоящий сетевой уровень. При этом поле будет передано в зашифрованном виде, разобщено (в смысле выполнения операции декомпозиции объекта на подобъекты) на последовательность пакетов, информационное поле каждого из которых также зашифровано, и затем передано по транспортной системе локальной или глобальной телекоммуникационной сети в виде датаграмм с зашифрованным информационным полем (адрес не будет закрыт, поскольку субъект нижестоящего уровня, который произвел декомпозицию, не имеет информации о функции преобразования объекта).

Следовательно, криптографическая защита объекта прикладного уровня действительна и для всех нижестоящих уровней.

Из указанного свойства следует следующее.

Утверждение 1. При защите информации на прикладном уровне процедуры передачи, разборки на пакеты, маршрутизации и обратной сборки не могут нанести ущерба конфиденциальности информации.

Упомянув о понятии конфиденциальности, нельзя не отметить, что **две классические задачи криптографической защиты -** защита конфиденциальности и защита целостности - инвариантны относительно любого уровня модели ISO (с учетом свойства наследования). Защита прикладного уровня также в основном решает две указанные задачи отдельно или в совокупности.

Особенностью существования субъектов-программ прикладного уровня, а также порождаемых ими объектов является отсутствие стандартизованных форматов представления объектов. Более того, можно утверждать, что такая стандартизация возможна лишь для отдельных структурных компонентов субъектов и объектов прикладного уровня (например, типизация данных в транслируемых и интерпретируемых языках программирования, форматы результирующего хранения для текстовых процессоров и др.). Субъекты и

объекты прикладных систем создаются пользователем, и априорно задать их структуру не представляется возможным.

Можно рассмотреть два подхода к построению СКЗИ на прикладном уровне.

Первый подход (наложенные СКЗИ) связан с реализацией функций криптографической защиты целиком в отдельном субъекте - программе (например, после подготовки электронного документа в файле активизируется программа цифровой подписи для подписания данного файла). Данный подход получил также название **абонентской защиты** (поскольку активизация программы производится оконечным пользователем - абонентом и локализуется в пределах рабочего места пользователя).

Второй подход (встраивание СКЗИ) связан с вызовом функций субъекта СКЗИ непосредственно из программы порождения защищаемых объектов и встраиванием криптографических функций в прикладную программу.

Первый подход отличается простотой реализации и применения, но требует учета **двух важных факторов**.

Во-первых, реализация субъект СКЗИ должна быть в той же операционной среде либо операционной среде, связанной потоками информации с той, в которой существует прикладной субъект.

Во-вторых, и прикладной субъект, и СКЗИ должны воспринимать объекты АС (т.е., декомпозиция компьютерной системы на объекты должна быть общей для обоих субъектов).

Два вышеуказанных фактора предполагают **раздельную реализацию СКЗИ в операционной среде и связь по данным**.

С другой стороны, в современных системах обработки и передачи информации достаточно сложно произвести пространственно-временную локализацию порождения конечного объекта, который должен подвергаться защите. В связи с этим современные информационные технологии предполагают более широкое использование второго подхода (встраивание СКЗИ), используя для этого различные технические решения.

Сравним оба рассмотренных подхода построения СКЗИ на прикладном уровне в таблице 1.

Таблица 1

Свойства СКЗИ	**Первый подход**	**Второй подход**
Сопряжение с прикладной подсистемой	На этапе эксплуатации	На этапе проектирования и разработки
Зависимость от прикладной системы	Низкая	Высокая
Локализация защищаемого объекта	Внешняя (относительно защитного модуля и прикладной программы)	Внутренняя (защита внутреннего объекта прикладной программы)
Операционная зависимость	Полная	Низкая

Можно выделить несколько способов реализации криптографической защиты в отдельном субъекте.

1 Локальная реализация в виде выделенной прикладной программы.

1.1 Локальная реализация в базовой АС.

1.2 Локальная реализация в «гостевой» АС.

1.3 Локальная реализация по принципу «копирование в защищенный объект хранения».

2 Распределенная реализация по технологии создание и запись в защищенной области.

Подходы 1.3 и 2, как правило, называют **реализацией в виде локального или распределенного прикладного криптосервера**. Сущность их реализации была рассмотрена выше.

Встраивание криптографических функций в прикладную систему может осуществляться:

1) по технологии «открытый интерфейс»;

2) по технологии «криптографический сервер»;

3) на основе интерпретируемого языка прикладного средства.

Основной проблемой встраивания является корректное использование вызываемых функций.

Субъекты АС, связанные с выполнением защитных функция (например, субъекты порождения изолированной программной среды могут использовать некоторое общее подмножество криптографических функций логического преобразования объектов (в частности, алгоритмы контроля целостности объектов).

При проектировании АС исторически сложившийся подход относительно **распределения общего ресурса** связан с использованием **разделяемых субъектов, выполняющих общие для других субъектов функции**. Распространим данный подход на функции реализации логической защиты.

Разделяемая технология применения функций логической защиты – это такой порядок использования СКЗИ в защищенной АС, при которой:

- не требуются изменения в программном обеспечении при изменении криптографических алгоритмов;
- система защиты однозначно разделяема на две части: прикладная компонента и модуль реализации криптографических функций (МРКФ).

Открытым интерфейсом (**ОИ**) **МКРФ** назовем детальное специфицирование функций, реализованных в МРКФ. Относительно некоторого множества субъектов, использующих МРКФ, можно говорить **о полноте функций МРКФ**. Удобнее оперировать с формально описанными функциями ОИ, следовательно, далее будем говорить о полноте функций ОИ. Полнота функций ОИ может быть **функциональной** и **параметрической**.

Функциональная полнота ОИ - свойство, заключающееся в реализации всех функций класса функций защиты, инициируемых фиксированным набором субъектов АС. Из данного определения следует, что **функциональная полнота понимается относительно заданного множества программ, использующих функции ОИ**.

Параметрическая полнота ОИ - свойство, заключающееся в возможности инициирования всех функций ОИ со стороны фиксированного множества субъектов с некоторым набором параметров, не приводящих к отказу в выполнении запрошенной функции.

В настоящее время подход встраивания СКЗИ для защиты информации по технологии «открытого интерфейса» применен в операционных средах MS Windows в виде так называемого криптопровайдера (CryptoAPI 1.0 и 2.0), который позволяет создавать приложения, использующие криптографические методы.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1 Выбрать операционную систему.

2 Выбрать язык и соответствующую среду программирования.

3 Получить вариант индивидуального задания у преподавателя.

4 На базе программы разграничения полномочий пользователей на основе парольной аутентификации (см. методические указания к выполнению лабораторной работы «Разработка программы разграничения полномочий пользователей на основе парольной аутентификации») разработать программу, удовлетворяющую нижеперечисленным условиям:

а) программа должна обеспечивать работу режима администратора (пользователя с фиксированным именем ADMIN) и обычного пользователя.

б) в режиме администратора программа должна поддерживать следующие функции (при правильном вводе пароля):

- смена пароля администратора (при правильном вводе старого пароля);
- просмотр списка имен зарегистрированных пользователей и установленных для них параметров (принадлежность к группам, блокировка учетной записи, включение ограничений на выбираемые пароли) – всего списка целиком в одном окне или по одному элементу списка с возможностью перемещения к его началу или концу;
- создание нового пользователя с уникальным именем, паролем и привилегиями;
- блокирование возможности работы пользователя с заданным именем;
- включение или отключение ограничений на выбираемые пользователем пароли (в соответствии с индивидуальным заданием, определяемым номером варианта);
- удаление пользователя с заданным именем;
- завершение работы с программой;

в) в режиме обычного пользователя изменять его пароль и завершать

работу программы, остальные ее функции должны быть заблокированы;

г) средства защиты от несанкционированного доступа к файлу с учетными данными зарегистрированных пользователей.

5 После своего запуска программа должна вывести интерфейс с окном для ввода имени пользователя, пароля пользователя. При вводе пароля его символы на экране всегда должны заменяться символом «*» (при создании или изменении пароля данная функция может быть опциональной).

6 При отсутствии введенного в окне входа имени пользователя в списке зарегистрированных администратором пользователей программа должна выдавать соответствующее сообщение и предоставлять пользователю возможность повторного ввода имени или завершения работы с программой.

7 При неправильном вводе пароля в окне входа программа должна выдавать соответствующее сообщение и предоставлять пользователю возможность повторного ввода. При трехкратном вводе неверного пароля отказе от его ввода работа программы должна завершаться с выдачей соответствующего сообщения.

8 При первоначальном вводе пароля (обязательном при первом входе администратора или пользователя с зарегистрированным ранее администратором именем) и при дальнейшей замене пароля программа должна просить пользователя подтвердить введенный пароль путем его повторного ввода.

9 При попытке создания уже имеющегося в списке имени пользователя программа должна выдавать соответствующее сообщение и предоставлять пользователю возможность повторного ввода имени или завершения работы с программой.

10 Если выбранный пользователем пароль не соответствует требуемым ограничениям (в соответствии с инд. заданием лаб.1) при установке соответствующего параметра учетной записи пользователя, то программа должна выдавать соответствующее сообщение и предоставлять пользователю возможность ввода другого пароля, завершения работы с программой или отказа от смены пароля.

11 Информация о зарегистрированных пользователях, их паролях, отсутствии блокировки их работы с программой, а также включении или отключении ограничений на выбираемые пароли должна сохраняться в специальном файле. При первом запуске программы этот файл должен создаваться автоматически и содержать информацию только об администраторе, имеющем пустой пароль.

12 Файл с учетными данными должен быть зашифрован при помощи функций криптографического интерфейса операционной системы Windows (CryptoAPI) с использованием сеансового ключа, генерируемого на основе вводимой администратором (пользователем) парольной фразы.

13 При запуске программы файл с учетными данными должен расшифровываться во временный файл (или в файл в оперативной памяти), который после завершения работы программы должен быть снова зашифрован для отражения возможных изменений в учетных записях пользователей. «Старое» содержимое файла учетных записей при этом стирается.

14 После ввода парольной фразы при запуске программы, генерации ключа расшифрования и расшифрования файла с учетными данными зарегистрированных пользователей правильность введенной парольной фразы определяется по наличию в расшифрованном файле учетной записи администратора программы.

15 Временный файл на диске с расшифрованными учетными данными после завершения работы программы удаляется.

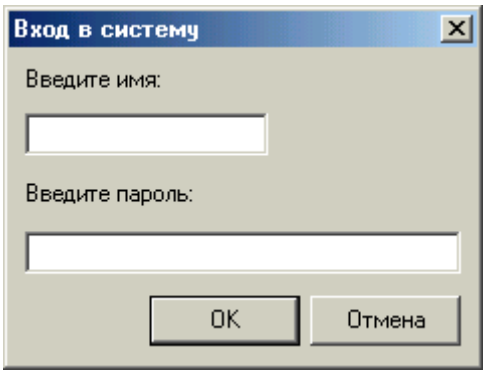
16 Варианты использования алгоритмов шифрования и хеширования выбираются в соответствии с выданным преподавателем заданием.

17 Для доступа к функциям CryptoAPI из программ на Паскале следует использовать интерфейсный модуль wincrypt.pas с указанного преподавателем сетевого диска.

18 Интерфейс с программой должен быть организован на основе меню, обязательной частью которого должно являться подменю «Справка» с командой «О программе». При выборе этой команды должна выдаваться информация об авторе программы и выданном индивидуальном задании. Интерфейс пользователя программы может также включать панель управления с дублирующими команды меню графическими кнопками и строку состояния.

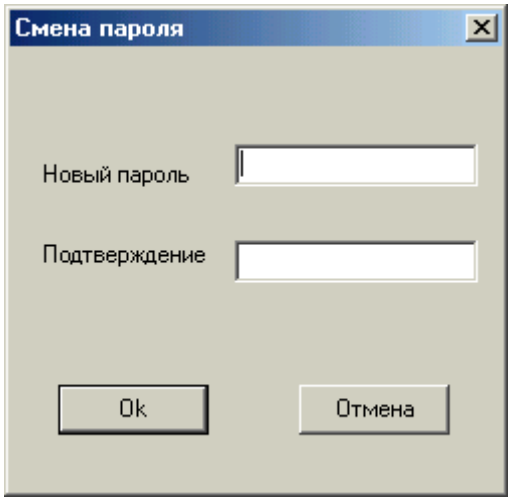
19 Для реализации указанных в пункте 2 функций в программе должны использоваться специальные диалоговые формы, позволяющие администратору вводить необходимую информацию.
Возможный вид диалоговых форм программы:

Окно входа в программу



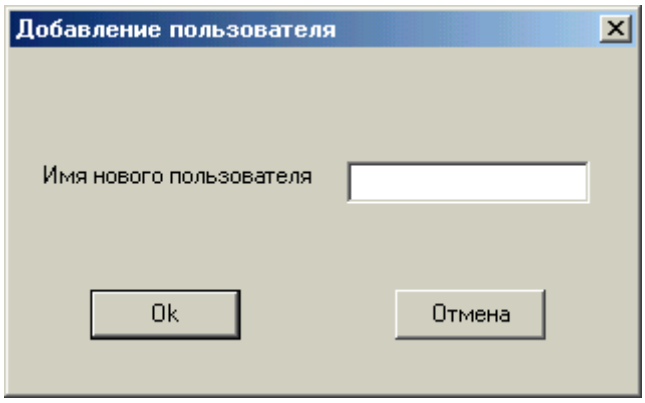
Может быть создано на основе шаблона Password Dialog, выбираемого с помощью команды File | New | Dialogs систем программирования Borland Delphi или Borland C++ Builder.

Окно смены пароля



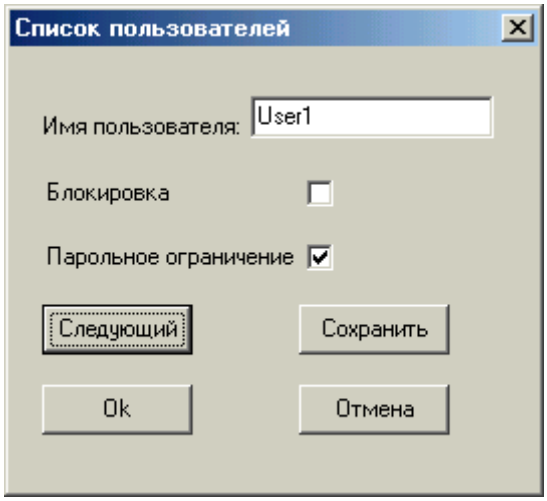
Возможно добавление на форму надписи «Старый пароль» и редактируемой строки для ввода действующего пароля.

Окно добавление нового пользователя



Возможно добавление на форму элементов управления для отображения и изменения значений параметров, устанавливаемых администратором для новой учетной записи (блокировка, ограничение на выбираемые пароли).

Окно просмотра (редактирования) учетных записей



Возможно добавление кнопки «Предыдущий» для перехода к предыдущей учетной записи или отображение списка учетных записей пользователей и их параметров в одном окне с помощью компонента StringGrid (группа Additional) систем программирования Borland Delphi или Borland C++ Builder.

20 Ограничения на выбираемые пользователями программы пароли и методы шифрования зависят от указанного преподавателем номера индивидуального варианта.

21 При выполнении лабораторной работы рекомендуется использовать указанные ниже средства языков и систем программирования и системные функции из набора Windows API.

22 Представить отчет по выполненной работе с блок-схемой программы и описанием использованных в ней алгоритмов. Приложить zip-архив с исходным и исполняемым кодом разработанной программы. Согласно таблице 2.

Таблица 2

Индивидуальные варианты заданий

№	Тип симметричного шифрования	Используемый режим шифрования	Добавление к ключу случайного значения	Используемый алгоритм хеширования
1	2	3	4	5
1	Блочный	Электронная кодовая книга	Да	MD2
2	Потоковый	-	Да	MD2
3	Блочный	Сцепление блоков шифра	Да	MD2
4	Потоковый	-	Да	MD5
5	Блочный	Обратная связь по шифротексту	Да	MD2
6	Потоковый	-	Да	SHA
7	Блочный	Электронная кодовая книга	Да	MD4
8	Потоковый	-	Нет	MD2
9	Блочный	Сцепление блоков шифра	Да	MD4
10	Потоковый	-	Нет	MD5
11	Блочный	Обратная связь по шифротексту	Да	MD4
12	Потоковый	-	Нет	SHA
13	Блочный	Электронная кодовая книга	Да	MD5
14	Блочный	Сцепление блоков шифра	Да	MD5

Продолжение таблицы 2

15	Блочный	Обратная связь по шифротексту	Да	MD5
16	Блочный	Электронная кодовая книга	Да	SHA
17	Блочный	Сцепление блоков шифра	Да	SHA
18	Блочный	Обратная связь по шифротексту	Да	SHA
19	Блочный	Электронная кодовая книга	Нет	MD2
20	Блочный	Сцепление блоков шифра	Нет	MD2
21	Блочный	Обратная связь по шифротексту	Нет	MD2
22	Блочный	Электронная кодовая книга	Нет	MD4
23	Блочный	Сцепление блоков шифра	Нет	MD4
24	Блочный	Обратная связь по шифротексту	Нет	MD4
25	Блочный	Электронная кодовая книга	Нет	MD5
26	Блочный	Сцепление блоков шифра	Нет	MD5
27	Блочный	Обратная связь по шифротексту	Нет	MD5
28	Блочный	Электронная кодовая книга	Нет	SHA
29	Блочный	Сцепление блоков шифра	Нет	SHA
30	Блочный	Обратная связь по шифротексту	Нет	SHA

Рекомендуемые для разработки программы средства языка Object Pascal

1 Нетипизированный файл для операций шифрования (расшифрования) файла учетных записей:

Var *Имя_файловой_переменной*: File;

Для повышения безопасности расшифрованный файл учетных записей может временно сохраняться в объекте класса TMemoryStream.

2 Работа с нетипизированным файлом:

procedure AssignFile(var F: File; FileName: string); { «связывание» файловой переменной F с файлом под именем FileName }

procedure Reset(var F : File; RecSize: Word); { открытие существующего файла F с длиной записи RecSize (рекомендуется 1) }
procedure Rewrite(var F: File ; RecSize: Word); { создание нового файла F с длиной записи RecSize (рекомендуется 1) }
procedure BlockRead(var F: File; var Buf; Count: Integer; var AmtTransferred: Integer); { чтение Count записей из файла F в переменную Buf (в переменную AmtTransferred помещается фактическое количество прочитанных записей) }
procedure BlockWrite(var f: File; var Buf; Count: Integer); { запись Count записей из переменной Buf в файл F }
function DeleteFile(const FileName: string): Boolean; { удаление файла с именем FileName }
procedure CloseFile(var F: File); // закрытие файла
function Eof(var F: File): Boolean; // проверка достижения конца файла

3 Работа с файлом в оперативной памяти (свойства и методы класса TMemoryStream):

Position:Integer // смещение текущей позиции в байтах
Size:Integer // размер в байтах
function Read(Buffer:Pointer; Count:Integer):Integer; { чтение Count байт в буфер Buffer (результат – фактическое количество прочитанных байт) }
function Write(const Buffer: Pointer; Count: Integer):Integer; { запись Count байт из буфера Buffer }
function Seek(Offset:Integer; Origin: Word):Integer; { смещение текущей позиции на Offset байт относительно Origin: начала потока soFromBeginning, текущей позиции soFromCurrent, конца потока soFromEnd }

4 Шифрование (расшифрование) файла:

THandle – тип данных для дескрипторов криптопровайдера, криптографического ключа, хеш-объекта
ALG_ID – тип данных для кодов криптографических алгоритмов
function CryptAcquireContext(var hProv: THandle; pszContainer, pszProvider: PChar;
dwProvType, dwFlags: Longint): Longbool; { инициализация криптопровайдера (в hProv записывается его дескриптор, pszContainer=Nil, pszProvider=Nil, dwProvType= PROV_RSA_FULL, dwFlags=0) или (когда при первом запуске программы CryptAcquireContext возвращает false) регистрация нового пользователя в криптопровайдере (dwFlags= CRYPT_NEWKEYSET) }
function CryptCreateHash(hProv: THandle; Algid: ALG_ID; hKey: THandle; dwFlags: Longint; var hHash: THandle): Longbool; { создание пустого хеш-объекта (Algid – код алгоритма хеширования, hKey=0, dwFlags=0, в hHash записывается дескриптор хеш-объекта) }
function CryptHashData(hHash: THandle; pbData: Pointer; dwDataLen, dwFlags: Longint): Longbool; { хеширование парольной фразы (или произвольных данных) pbData длины dwDataLen (dwFlags=0) }
function CryptDestroyHash(hHash: THandle): Longbool; // разрушение хеш-объекта

function CryptDeriveKey(hProv: THandle; Algid: ALG_ID; hBaseData: THandle; dwFlags: Longint; var hKey: THandle): Longbool; { создание ключа шифрования из хеш-объекта с парольной фразой hBaseData (Algid – код алгоритма шифрования, dwFlags= CRYPT_EXPORTABLE с возможным объединением через or с признаком добавления к ключу случайного значения CRYPT_CREATE_SALT, в hKey записывается дескриптор ключа) }
function CryptDestroyKey(hKey: THandle): Longbool; { разрушение ключа шифрования }
function CryptReleaseContext(hProv: THandle; dwFlags: Longint): Longbool; { освобождение криптопровайдера }
function CryptEncrypt(hKey, hHash: THandle; Final: Longbool; dwFlags: Longint; pbData: Pointer; var dwDataLen: Longint; dwBufLen: Longint): Longbool; { шифрование порции данных из буфера pbData длины dwBufLen, которая для блочных шифров должна быть кратной 8 (dwDataLen – длина порции данных, после выполнения функции в эту переменную записывается фактическая длина зашифрованных данных; hHash=0, dwFlags=0, Final – признак последней порции данных) }
function CryptDecrypt (hKey, hHash: THandle; Final: Longbool; dwFlags: Longint; pbData: Pointer; var dwDataLen: Longint):Longbool; { расшифрование порции данных из буфера pbData (dwDataLen – длина порции данных, после выполнения функции в эту переменную записывается фактическая длина расшифрованных данных; hHash=0, dwFlags=0, Final – признак последней порции данных) }
function CryptSetKeyParam(hKey: THandle; dwParam: Longint; pbData: Pointer; dwFlags: Longint): Longbool; { установка режима шифрования для ключа hKey (dwParam= KP_MODE, pbData указывает на переменную типа Longint, в которой записан код устанавливаемого режима, dwFlags=0) }

Рекомендуемые для разработки программы средства языка Си++

1 Класс fstream для операций шифрования (расшифрования) файла учетных записей пользователей:

fstream *имя_файловой_переменной*;

Для повышения безопасности расшифрованный файл учетных записей может временно сохраняться в объекте класса TMemoryStream (система Borland C++ Builder) или CMemFile (система Microsoft Visual C++).

2 Работа с файлом:

void open(const char *FileName, ios::in | ios::binary); /* открытие существующего файла под именем FileName для чтения */
void open(const char *FileName,ios::out|ios::binary); /* создание нового файла с именем FileName */
istream& read(char *buf, int n); // чтение данных в буфер buf длины n
int gcount(); /* количество байт, фактически прочитанных во время последней операции */
ostream& write(const char *buf,int n); // запись данных из буфера buf длины n
void close(); // закрытие файла

bool eof(); // проверка достижения конца файла
int remove(const char *filename); // удаление файла с именем filename

3 Работа с файлом в оперативной памяти:
Свойства и методы класса TMemoryStream
int Position // смещение текущей позиции в байтах
int Size // размер в байтах
int Read(void *Buffer, int Count); /* чтение Count байт в буфер Buffer (результат – фактическое количество прочитанных байт) */
int Write(const void *Buffer, int Count); // запись Count байт из буфера Buffer
int Seek(int Offset, Word Origin); /* смещение текущей позиции на Offset байт относительно Origin: начала потока soFromBeginning, текущей позиции soFromCurrent, конца потока soFromEnd */
Методы класса CMemFile
UINT Read(void *lpBuf,UINT nCount); /* чтение nCount байт в буфер lpBuf (результат – фактическое количество прочитанных байт) */
void Write(const void *lpBuf,UINT nCount); // запись nCount байт из буфера lpBuf
LONG Seek(Long lOff,UINT nFrom); /* смещение текущей позиции на lOff байт относительно nFrom: начала файла CFile::begin, текущей позиции CFile::current, конца файла CFile::end */
DWORD GetLength(); // длина файла в байтах
DWORD GetPosition(); // текущая позиция файла

4 Шифрование (расшифрование) файла:
HCRYPTPROV, HCRYPTKEY, HCRYPTHASH – типы данных для дескрипторов криптопровайдера, криптографического ключа, хеш-объекта
ALG_ID – тип данных для кодов криптографических алгоритмов
BOOL CryptAcquireContext(HCRYPTPROV *phProv, LPCSTR pszContainer,
LPCSTR pszProvider, DWORD dwProvType, DWORD dwFlags); /* инициализация криптопровайдера (в *phProv записывается его дескриптор, pszContainer=NULL, pszProvider=NULL, dwProvType= PROV_RSA_FULL, dwFlags=0) или (когда при первом запуске программы CryptAcquireContext возвращает FALSE) регистрация нового пользователя в криптопровайдере (dwFlags= CRYPT_NEWKEYSET) */
BOOL CryptCreateHash(HCRYPTPROV hProv, ALG_ID Algid, HCRYPTKEY hKey,
DWORD dwFlags, HCRYPTHASH *phHash); /* создание пустого хеш-объекта (Algid – код алгоритма хеширования, hKey=0, dwFlags=0, в *phHash записывается дескриптор хеш-объекта) */
BOOL CryptHashData(HCRYPTHASH hHash, CONST BYTE *pbData, DWORD dwDataLen, DWORD dwFlags); /* хеширование парольной фразы (или любых других данных) pbData длины dwDataLen (dwFlags=0) */
BOOL CryptDestroyHash(HCRYPTHASH hHash); // разрушение хеш-объекта
BOOL CryptDeriveKey(HCRYPTPROV hProv, ALG_ID Algid, HCRYPTHASH hBaseData, DWORD dwFlags, HCRYPTKEY *phKey); /* создание ключа шиф-

рования из хеш-объекта с парольной фразой hBaseData (Algid – код алгоритма шифрования, dwFlags=CRYPT_EXPORTABLE с возможным объединением через or с признаком добавления к ключу случайного значения CRYPT_CREATE_SALT, в *phKey записывается дескриптор ключа) */

BOOL CryptDestroyKey(HCRYPTKEY hKey); // разрушение ключа шифрования

BOOL CryptReleaseContext(HCRYPTPROV hProv, DWORD dwFlags); /* освобождение криптопровайдера */

BOOL CryptEncrypt(HCRYPTKEY hKey, HCRYPTHASH hHash, BOOL Final, DWORD dwFlags, BYTE *pbData, DWORD *pdwDataLen, DWORD dwBufLen); /* шифрование порции данных из буфера pbData длины dwBufLen, которая для блочных шифров должна быть кратной 8 (dwDataLen – длина порции данных, после выполнения функции в эту переменную записывается фактическая длина зашифрованных данных; hHash=0, dwFlags=0, Final – признак последней порции данных) */

BOOL CryptDecrypt(HCRYPTKEY hKey, HCRYPTHASH hHash, BOOL Final, DWORD dwFlags,BYTE *pbData,DWORD *pdwDataLen); /* расшифрование порции данных из буфера pbData (dwDataLen – длина порции данных, после выполнения функции в эту переменную записывается фактическая длина расшифрованных данных; hHash=0, dwFlags=0, Final – признак последней порции данных) */

BOOL CryptSetKeyParam(HCRYPTKEY hKey,DWORD dwParam,BYTE *pbData, DWORD dwFlags); /* установка режима шифрования для ключа hKey (dwParam= KP_MODE, pbData указывает на переменную типа unsigned long, в которой записан код устанавливаемого режима, dwFlags=0) */

Контрольные вопросы

1 Способы аутентификации пользователей в компьютерных системах.
2 Организация базы учетных записей пользователей и доступа к ней.
3 Хранение идентифицирующей информации в операционной системе Unix.
4 Хранение идентифицирующей информации в операционной системе Windows.
5 Аутентификация пользователей на основе паролей.
6 Использование одноразовых паролей.
7 Аутентификация пользователей на основе модели «рукопожатия».
8 Аутентификация пользователей по их биометрическим характеристикам.
9 Разграничение прав пользователей в незащищенных версиях ОС Windows.
10 Разграничение прав субъектов в защищенных версиях ОС Windows.

Список литературы

1 Хатч Б. Ли Д. Курц Д. Секреты хакеров. Безопасность Linux – готовые решения./Пер. с англ. – М.: Вильямс, 2004.

2 Скембрей Д., Мак-Клар С. Секреты хакеров. Безопасность Windows 2000 – готовые решения./Пер. с англ. – М.: Вильямс, 2002.

3 Грушо А.А., Применко Э.А., Тимонина Е.Е. Теоретические основы компьютерной безопасности. – М.: «Академия», 2009.

Москвин Владимир Викторович

ИСПОЛЬЗОВАНИЕ ФУНКЦИЙ КРИПТОГРАФИЧЕСКОГО ИНТЕРФЕЙСА WINDOWS ДЛЯ ЗАЩИТЫ ИНФОРМАЦИИ

Методические указания
к выполнению лабораторной работы
по дисциплине «Теоретические основы компьютерной безопасности»
для студентов направления 075500
специальности 090105

Редактор Е.А. Устюгова

Подписано к печати	Формат 60×84 1/16	Бумага тип. №1
Печать трафаретная	Усл. печ.л. 1,25	Уч.-изд.л.1,25
Заказ	Тираж э/в	Цена свободная

РИЦ Курганского государственного университета.
640669, г. Курган, ул. Гоголя, 25.
Курганский государственный университет.