Лабораторная работа №2

Использование функций криптографического интерфейса Windows для защиты информации

Содержание задания

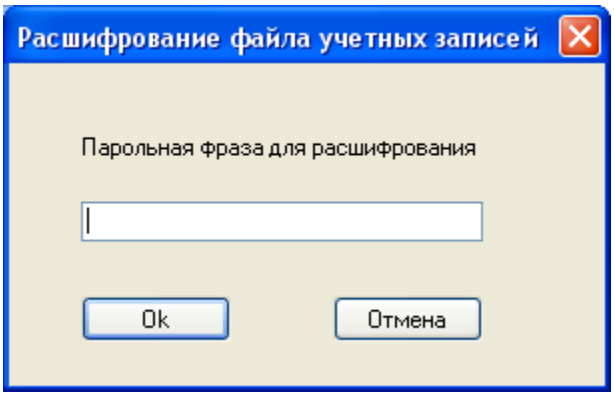
1. В программу, разработанную при выполнении лабораторной работы № 1, добавить средства защиты от несанкционированного доступа к файлу с учетными данными зарегистрированных пользователей.
2. Файл с учетными данными должен быть зашифрован при помощи функций криптографического интерфейса операционной системы Windows (CryptoAPI) с использованием сеансового ключа, генерируемого на основе вводимой администратором (пользователем) парольной фразы.
3. При запуске программы файл с учетными данными должен расшифровываться во временный файл (или в файл в оперативной памяти, что безопаснее), 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.
4. 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.
5. При вводе неправильной парольной фразы или отказе от ее ввода работа программы должна завершаться с выдачей соответствующего сообщения.
6. Временный файл на диске с расшифрованными учетными данными после завершения работы программы удаляется.
7. Варианты использования алгоритмов шифрования и хеширования выбираются в соответствии с выданным преподавателем заданием.
8. Для доступа к функциям CryptoAPI из программ на Паскале следует использовать интерфейсный модуль wincrypt.pas с указанного преподавателем сетевого диска.

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

1	2	3	4	5
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
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
31	Потоковый	-	Да	MD4
32	Потоковый	-	Нет	MD4
33	DES	Электронная кодовая книга	Нет	SHA
34	DES	Сцепление блоков шифра	Нет	MD5
35	DES	Обратная связь по шифротексту	Нет	MD4

Возможный вид дополнительной диалоговой формы программы



Для повышения безопасности эта форма должна создаваться (уничтожаться) в программе явным образом.

Рекомендуемые средства языка программирования C#

1. Классы для алгоритмов симметричного блочного шифрования и расшифрования (требуется пространство имен System.Security.Cryptography).

RC2CryptoServiceProvider, AesCryptoServiceProvider, AesManaged, DESCryptoServiceProvider, TripleDESCryptoServiceProvider, RijndaelManaged.

Свойства

CipherMode Mode; // режим шифрования.
byte[] Key; // криптографический ключ.
int KeySize; // длина ключа в битах.
byte[] IV; // начальный вектор.
int BlockSize; // длина блока в битах.

Методы:

ICryptoTransform CreateEncryptor(byte[] rgbKey, byte[] rgbIV); /* создание объекта для шифрования с ключом rgbKey и начальным вектором rgbIV */
ICryptoTransform CreateDecryptor(byte[] rgbKey, byte[] rgbIV); /* создание объекта для расшифрования с ключом rgbKey и начальным вектором rgbIV */
void GenerateIV(); // генерация случайного начального вектора.

2. Классы для алгоритма потокового симметричного шифрования RC4CryptoServiceProvider и ARCFourManaged.

Требуется подключение к проекту ссылки на библиотеку Org.Mentalis.Security.dll и использования пространства имен Org.Mentalis.Security.Cryptography.

3. Класс для потока данных для шифрования и расшифрования. CryptoStream.

Конструктор CryptoStream(Stream* *stream*, ICryptoTransform* *transform*, CryptoStreamMode *mode*); /* создание криптографического потока для потока *stream* и криптографического объекта *transform* в режиме чтения (*mode*=CryptoStreamMode.Read) или записи (*mode*=CryptoStreamMode.Write) */

Методы

void Write(byte[] buffer, int offset, int count); /* запись данных длиной *count* из буфера *buffer* со смещением *offset* */

int Read(byte[] buffer, int offset, int count); /* чтение данных в буфер *buffer* максимальной длины *count* со смещением *offset* с возвращением числа фактически прочитанных байт */

4. Класс для потока расшифрованных учетных записей в оперативной памяти.
MemoryStream.

Конструктор без параметров для автоматически расширямого потока, свойства Length и Position, методы Close, Read, Write и Seek аналогичны одноименным свойствам и методам класса FileStream (см. описание лабораторной работы 1).

5. Класс для криптографически стойкого генератора псевдослучайных чисел (используется при создании примеси).
NGCryptoServiceProvider.

Метод void GetBytes(byte[] data); /* получение случайного числа в буфере data */

6. Классы для вывода криптографического ключа, основанного на парольной фразе.

6.1. PasswordDeriveBytes.

Конструктор PasswordDeriveBytes(byte[] password, byte[] salt); /* создание экземпляра класса на основе парольной фразы password и случайного значения (примеси) salt */

Метод byte[] CryptDeriveKey(string algname, string alghashname, int keySize, byte[] rgbIV); /* получение ключа длиной keySize бит (если указан 0, то используется длина ключа по умолчанию для заданного алгоритма) для алгоритма algname с помощью функции хеширования alghashname и начального вектора rgbIV */

6.2. Rfc2898DeriveBytes (парольная фраза хешируется по алгоритму SHA1).

Конструктор Rfc2898DeriveBytes(string password, byte[] salt); /* создание экземпляра класса на основе парольной фразы password и случайного значения (примеси) salt длиной 8 байт или более */

Метод byte[] GetBytes(int cb); // получение ключа длиной cb байт.

7. Классы для функций хеширования MD2 и MD4.
MD2CryptoServiceProvider и MD4CryptoServiceProvider.
Требуется подключение к проекту ссылки на библиотеку Org.Mentalis.Security.dll и использования пространства имен Org.Mentalis.Security.Cryptography.

Свойство int HashSize: // длина хеш-значения в битах.

Метод byte[] ComputeHash (byte[] buffer); /* хеширование данных из буфера *buffer* */

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 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 *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) */
Microsoft Visual C++.
Требуется подключение файла wincrypt.h (например, в автоматически создаваемом файле stdafx.h при использовании конструктора приложений AppWizard).

Криптографические многокомпонентные объекты библиотеки CAPICOM (для использования в языках интерпретируемого типа)

Для использования объектов CAPICOM на компьютере должна быть установлена библиотека CAPICOM.dll, а COM-объекты из этой библиотеке должны быть зарегистрированы в системном реестре Windows. Библиотека CAPICOM.dll поставляется вместе с операционной системой Windows, начиная с версии Windows XP. Для использования этой библиотеки с другими версиями Windows она может быть загружена с Web-сайта корпорации Microsoft (URL http://www.microsoft.com/msdownload/ platformsdk/sdkupdate/psdkredist.htm).

После загрузки библиотеки CAPICOM.dll находящиеся в ней COM-объекты необходимо зарегистрировать в реестре, введя в командной строке следующую команду (после изменения текущего каталога на тот, в котором записан файл CAPICOM.dll):

regsvr32 CAPICOM.dll

Если библиотека объектов CAPICOM установлена на компьютере и успешно загружена, то перед использованием в программе (сценарии) любого из объектов этой библиотеки необходимо получить ссылку на объект. Для этого потребуется определить соответствующую переменную и вызвать функцию для задания этой переменной нужного значения, например:

´ язык сценариев Visual Basic Scripting Edition,

```
´ язык программирования Visual Basic
´ определение переменной
dim hash
´ получение ссылки на объект
set hash=CreateObject("CAPICOM.HashedData.1")
// язык сценариев Java Script
// определение переменной и получение ссылки на объект
var MyStore = new ActiveXObject("CAPICOM.Store");
```

При получении ссылки на объект указываются (разделенные точкой) имя приложения или библиотеки, поставляющей этот объект (CAPICOM), имя объекта (в примерах HashedData и Store) и, возможно, номер версии библиотеки CAPICOM.

Объект EncryptedData предназначен для шифрования и расшифрования данных на сеансовом ключе, генерируемом из секретной парольной фразы или других секретных данных. Этот объект имеет методы SetSecret (установка секретной парольной фразы), Encrypt (шифрование и кодирование данных) и Decrypt (декодирование и расшифрование данных), а также свойства Content (строка с шифруемым открытым текстом) и Algorithm (ссылка на объект с характеристиками используемого алгоритма симметричного шифрования).

Метод SetSecret может принимать до двух параметров – строку с секретными данными, которые будут использоваться для генерации сеансового ключа, и код вида устанавливаемого секрета (по умолчанию 0, т.е. секретом является парольная фраза). Этот метод должен вызываться как перед шифрованием, так и перед расшифрованием, чтобы установить один и тот же секрет.

Метод Encrypt создает сеансовый ключ из ранее установленного секрета, зашифровывает с помощью этого ключа содержимое свойства Content и возвращает строку с закодированным шифротекстом. Единственным параметром этого метода может быть тип кодировки шифротекста (по умолчанию используется значение 0, т.е. CRYPT_STRING_BASE64, а другим возможным значением является 1, при котором шифротекст сохраняется в двоичном виде).

Метод Decrypt имеет единственный параметр – строку с шифротекстом – и после своего успешного выполнения записывает в свойство Content строку с расшифрованными данными. Для расшифрования используется сеансовый ключ, генерируемый из ранее установленного секрета. Объект Algorithm, ссылка на который может быть получена с помощью одноименного свойства объекта EncryptedData, имеет свойства KeyLength (код длины ключа симметричного шифрования: 0 – максимально возможная для данного алгоритма, 1 – 40 бит, 2 – 56 бит, 3 – 128 бит, 4 – 192 бита, 5 – 256 бит) и Name (код алгоритма шифрования: 0 – RC2, 1 – RC4, 2 – DES, 3 – 3-DES, 4 – AES). Для алгоритмов DES и 3-DES длина ключа является стандартной и не может быть изменена. При изменении значения свойства Name состояние объекта Algorithm сбрасывается.