Министерство науки и высшего образования РФ

Томский государственный университет
систем управления и радиоэлектроники

**И.А. Рахманенко, А.Ю. Якимук**

# РАЗРАБОТКА КОМПОНЕНТОВ СРЕДСТВ ЗАЩИТЫ ИНФОРМАЦИИ

Учебно-методическое пособие
для студентов направлений подготовки
10.00.00 Информационная безопасность

Томск
2022

**УДК** 004.056
**ББК** 32.973.26-018.2
Р 64

**Рахманенко Иван Андреевич**
Р 64 Разработка компонентов средств защиты информации: учебно-методическое пособие/ И.А. Рахманенко, А.Ю. Якимук. – Томск: Томск. гос. ун-т систем упр. и радиоэлектроники, 2022. – 78 с.

Настоящее учебно-методическое пособие содержит описания лабораторных и самостоятельных работ по дисциплине «Разработка компонентов средств защиты информации» для направлений подготовки, входящих в укрупненную группу специальностей и направлений 10.00.00 Информационная безопасность.

УДК 004.056
ББК 32.973.26-018.2

## СОДЕРЖАНИЕ

## ВВЕДЕНИЕ

Целью преподавания дисциплины является формирование глубокого понимания принципов функционирования компонентов средств защиты информации и получение практических навыков создания и тестирования компонентов средств защиты информации.

Задачи изучения дисциплины:

– Ознакомление с типовыми компонентами средств защиты информации;

– Изучение принципов функционирования компонентов средств защиты информации;

– Получение практических навыков создания компонентов средств защиты информации;

– Получение практических навыков тестирования компонентов средств защиты информации.

# ЛАБОРАТОРНАЯ РАБОТА №1
# Аутентификация с использованием пароля, внешних аутентификаторов

## 1. Цель работы

Целью работы является ознакомление с основами работы с eToken API, изучение базовых функций, связанных с определением наличия eToken в системе, приобретение навыков получения информацию о подключенном eToken.

## 2. Краткие теоретические сведения

Для использования eToken в разрабатываемом ПО существует 3 вида API:

- PKCS#11
- CAPI
- SAPI

### PKCS#11

PKCS#11 (Public-Key Cryptography Standard - Стандарт криптографии с открытым ключом) используется для работы с асимметричными криптографическими алгоритмами, был разработан RSA Laboratories и включает в себя как зависимые от алгоритма, так и не зависимые стандарты разработки. Это промышленный стандарт, который задает интерфейс для криптографических устройств, таких как смарт-карты и карты PCMCIA.

Этот стандарт определяет API (application programming interface - интерфейс программирования приложений), называемый Cryptoki (Cryptographic Token Interface), для устройств, физических или виртуальных, содержащих криптографическую информацию (ключи или данные) и выполняющих криптографические функции. Этот API используется на многих платформах и обладает достаточными возможностями для большинства приложений, связанных с безопасностью. Компания Аладдин рекомендует использовать PKCS#11 в качестве основного API для программирования eToken.

### CAPI

CAPI (CryptoAPI) разработан компанией Microsoft в качестве части операционных систем Microsoft Windows. Предполагалось, что он

будет использоваться для разработки приложений на платформе MS Windows. CAPI позволяет одновременно использовать несколько криптопровайдеров (cryptographic service providers, CSP) на одном компьютере или в одном приложении. Также он позволяет ассоциировать конкретный криптопровайдер с конкретной смарт-картой, так чтобы приложения вызывали корректный криптопровайдер при работе с криптографией. ОС Windows содержит много вспомогательных функций, позволяющих упростить код при работе с криптографией или сложными структурами данных (например сертификатами).

**SAPI**

SAPI (Supplementary API – дополнительный API) был реализован в версии eToken RTE 3.60 для того, чтобы убрать необходимость в использовании низкоуровневых функций при работе с eToken. Он давал доступ к специфичным для eToken возможностям, не рассматриваемым в стандарте PKCS#11. Этот функционал теперь доступен в PKCS#11 API.

Выбор API зависит от конкретных нужд разработчика. Вот несколько ключевых отличий:

– PKCS#11 позволяет управлять несколькими eToken одновременно. В CAPI нет понятия физического токена (необходимо использовать специальные техники).

– PKCS#11 позволяет ожидать уведомлений о подключении/удалении eToken. CAPI не обладает такой возможностью (возможность есть через Win32 API).

– PKCS#11 позволяет хранить ключи RSA, сертификаты и данные на eToken. CAPI позволяет хранить только ключи RSA и соответствующие им сертификаты.

– В PKCS#11 нет вспомогательных функций для работы с сертификатами. Обработка и проверка сертификатов X.509 может быть довольно сложной задачей. Однако, работая в ОС Windows, имеется возможность использовать функции Win32 даже работая с PKCS#11.

– PKCS#11 - это API, а не архитектура. Если приложению требуется работать с несколькими провайдерами, оно само должно определить как взаимодействовать с ними. CAPI является частью ОС Windows и поэтому, как только новый провайдер установлен в систему, он автоматически становится доступен для всех приложений.

– В CAPI имеется множество вспомогательных функций. Они могут помочь программисту сконцентрироваться на логике работы приложения, не сталкиваясь с низкоуровневыми проблемами.

## 3. Ход работы

Загрузите проект для Visual Studio под названием Lab1. В данном проекте содержится один файл, содержащий только код, необходимый для запуска программы. Добавьте заголовочный файл “eTPkcs11.h” в проект (`#include "include\eTPkcs11.h"`). Данный файл содержит функции, необходимые для работы с eToken. Добавьте следующий код в проект:

```
void init();
void leave(const char*);
void displayLibraryInfo();
void displayTokenInfo(DWORD slotId);

//Глобальные переменные
CK_FUNCTION_LIST_PTR   pFunctionList=NULL;
CK_C_GetFunctionList   pGFL    = 0;
bool                   wasInit = false;
static HANDLE          hThread = 0;
```

Здесь описаны заголовки функций, которые в дальнейшем будут использоваться для работы с eToken. Функция `init()` обеспечивает инициализацию библиотеки PKCS#11, функция `leave(const char*)` обеспечивает выгрузку библиотки PKCS#11, функция `displayLibraryInfo()` предназначен для вывода информации о библиотеке PKCS#11, функция предназначен для вывода информации о подключенном eToken.

Глобальные переменные pFunctionList, pGFL предназначены для получения указателя на функции библиотеки eTpkcs11.dll, wasInit указывает, была ли уже загружена динамическая библиотека eTpkcs11.dll, hThread служит для получения указателя на поток, в котором будет выполняться поиск событий подключения или отключения eToken.

Для того чтобы начать использовать функции библиотеки PKCS#11, ее необходимо сначала загрузить и инициализировать. Добавьте следующий код в проект:

```
void init()
{
  // Загружаем dll
  HINSTANCE hLib = LoadLibraryA("etpkcs11.DLL");
```

```
  if (hLib == NULL)
  {
    leave ("Cannot load DLL.");
  }

  // Ищем точку входа для C_GetFunctionList

  (FARPROC&)pGFL= GetProcAddress(hLib,
"C_GetFunctionList");

  if (pGFL == NULL)
  {
    leave ("Cannot find GetFunctionList().");
  }

  //Берем список функций
  if (CKR_OK != pGFL(&pFunctionList))
  {
    leave ("Can't get function list. \n");
  }

  // Инициализируем библиотеку PKCS#11

  if (CKR_OK != pFunctionList->C_Initialize (0))
  {
    leave ("C_Initialize failed...\n");
  }
  wasInit = true;
}
```

Данная функция проводит загрузку и инициализацию библиотеки PKCS#11, а также получает указатель на список функций данной библиотеки (pFunctionList). В случае, если не удается загрузить библиотеку, или по какой-либо другой причине, выдается сообщение о соответствующей ошибке и производится завершение работы программы (функция leave). Добавьте код функции leave:

```
static void leave(const char * message)
{
  if (message) printf("%s ", message);

       if(wasInit)
```

```
  {
                // Закрываем библиотеку PKCS#11
                if (CKR_OK != pFunctionList->C_Finalize(0))
                {
                        printf ("C_Finalize failed...\n");
                }

                //ждем завершения работы потока, иначе убиваем
его.
                WaitForSingleObject(hThread, 5000);
                if (hThread) { TerminateThread(hThread,0);
CloseHandle(hThread); }

    hThread = 0;
    wasInit = false;
  }

       exit(message ? -1 : 0 );
}
```

Данную функцию будем вызывать в случае ошибки, задавая в качестве параметра сообщение об ошибке, а также в случае корректного завершения работы, передав параметр NULL.

Обратите внимание на условие `if (CKR_OK != pFunctionList->C_Finalize(0))`. С помощью сравнения со значением успешного выполнения операции `CKR_OK` можно убедиться, что любая из функций из списка pFunctionList выполнена успешно.

Рассмотрим каким образом получить информацию о библиотеке PKCS#11. За это отвечает функция C_GetInfo из списка функций pFunctionList. В качестве параметра этой функции передается ссылка на переменную типа `CK_INFO`, например `C_GetInfo (&info)`. Данная структура определена так:

```
typedef struct CK_INFO {

  CK_VERSION    cryptokiVersion;     /* Версия интерфейса
PKCS#11 */
  CK_UTF8CHAR   manufacturerID[32];  /* Информация о
производителе */
  CK_UTF8CHAR   libraryDescription[32];  /* Описание
библиотеки */
```

```
  CK_VERSION     libraryVersion;          /* Версия
библиотеки */
} CK_INFO;
```

Выведите в консоль информацию о библиотеке PKCS#11, написав функцию displayLibraryInfo() с использованием функции C_GetInfo. **Примечание:** структуры типа `CK_VERSION` включают в себя две целочисленных переменных – major и minor.

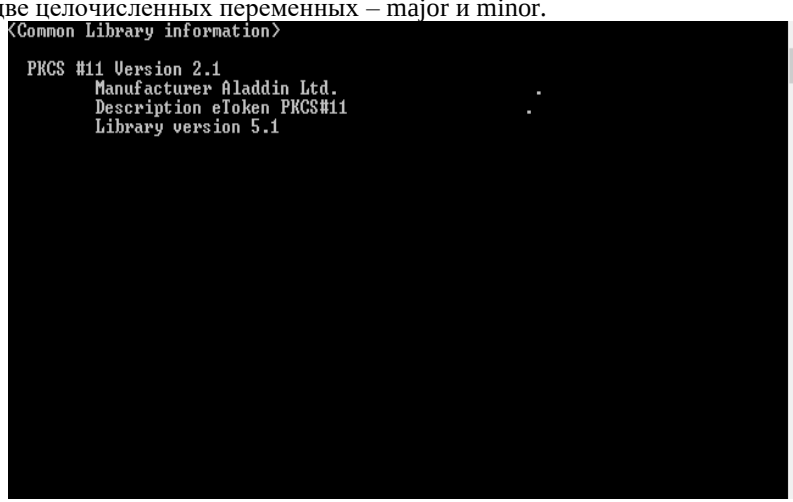



Рис. 1 – Пример вывода информации о библиотеке PKCS#11

*Получение информации о eToken.*

Для того, чтобы получить информацию о подключенном eToken, необходимо знать идентификатор виртуального слота, к которому подключен eToken. Идентификатор слота будет представлен в качестве параметра функции `displayTokenInfo(DWORD slotId)`.

Получить информацию, присутствует ли в данном слоте eToken можно по флагам переменной типа `CK_SLOT_INFO`. Получить данные флаги можно с помощью функции `C_GetSlotInfo (slotId, &slot_info)`. Всего имеется 3 флага:

```
#define CKF_TOKEN_PRESENT      0x00000001  /* eToken
присутствует в данном слоте */
#define CKF_REMOVABLE_DEVICE   0x00000002  /* К данному
слоту подключено извлекаемое устройство*/
```

```
#define  CKF_HW_SLOT                    0x00000004    /*  слот
аппаратный, иначе виртуальный */
```

Выведите информацию о данном слоте (`slot_info.slotDescription`), проверьте наличие eToken в слоте (`slot_info.flags & CKF_TOKEN_PRESENT`). В случае отсутствия eToken выведите соответствующее сообщение.

В случае наличия eToken в слоте, соберем информацию о нем. Для этого используются переменные типа `CK_TOKEN_INFO`. Для заполнения этой информации используйте функцию `C_GetTokenInfo (slotId, &token_info)`. Переменная типа `CK_TOKEN_INFO` содержит следующие поля:

```
typedef struct CK_TOKEN_INFO {

  CK_UTF8CHAR   label[32];           /* Имя eToken */
  CK_UTF8CHAR   manufacturerID[32];  /* Идентификатор
производителя */
  CK_UTF8CHAR   model[16];           /* Название модели */
  CK_CHAR       serialNumber[16];    /* Серийный номер */
  CK_FLAGS      flags;               /* флаги */
  CK_ULONG      ulMaxSessionCount;     /* max открытых
сессий */
  CK_ULONG      ulSessionCount;        /* Число открытых
сейчас сессий */
  CK_ULONG      ulMaxRwSessionCount;   /* max сессий
чтения/записи */
  CK_ULONG      ulRwSessionCount;      /* Число сессий
чтения/записи  сейчас*/
  CK_ULONG      ulMaxPinLen;           /* max длина Pin
кода */
  CK_ULONG      ulMinPinLen;           /* min длина Pin
кода */
  CK_ULONG      ulTotalPublicMemory;   /* Всего открытой
области памяти */
  CK_ULONG      ulFreePublicMemory;    /* Свободно в
открытой области памяти */
  CK_ULONG      ulTotalPrivateMemory;  /* Всего закрытой
области памяти */
  CK_ULONG      ulFreePrivateMemory;   /* Свободно в
закрытой области памяти */
  CK_VERSION    hardwareVersion;       /* Версия hardware
*/
```

```
  CK_VERSION     firmwareVersion;        /* Версия
встроенного ПО */
  CK_CHAR        utcTime[16];            /* Время (если есть
часы) */
} CK_TOKEN_INFO;
```

Перечень доступных наборов флагов представлен в табл. 1.

Таблица 1

Доступный набор флагов

| Имя флага | Маска | Значение |
|---|---|---|
| CKF_RNG | 0x00000001 | Присутствует генератор случайных чисел |
| CKF_WRITE_PROTECTED | 0x00000002 | eToken защищен от записи |
| CKF_LOGIN_REQUIRED | 0x00000004 | Требуется вход пользователя для совершения операций |
| CKF_USER_PIN_INITIALIZED | 0x00000008 | Установлен пользовательский Pin |
| CKF_RESTORE_KEY_NOT_NEEDED | 0x00000020 | Если установлен, значит не требуется ключ для восстановления криптографических операций |
| CKF_CLOCK_ON_TOKEN | 0x00000040 | Если установлен, значит на токене присутствуют внутренние часы. |
| CKF_PROTECTED_AUTHENTICATION_PATH | 0x00000100 | Если установлен, значит имеется возможность произвести аутентификацию без отправки Pin посредством библиотеки |
| CKF_DUAL_CRYPTO_OPERATIONS | 0x00000200 | Если установлен, значит имеется возможность одновременного выполнения двух криптографических операций (хеширование и шифрование, расшифровывание и формирование цифровой подписи, и т.д.) |

Продолжение табл. 1

| Имя флага | Маска | Значение |
|---|---|---|
| CKF_TOKEN_INITIALIZED | 0x00000400 | Если установлен, значит токен был инициализирован с помощью функции C_InitializeToken или эквивалентного механизма. Вызов функции C_InitializeToken с данным флагом приведет к повторной инициализации токена |
| CKF_SECONDARY_AUTHENTICATION | 0x00000800 | Если установлен, значит токен поддерживает вторичную аутентификацию для доступа к закрытым ключам |
| CKF_USER_PIN_COUNT_LOW | 0x00010000 | Если установлен, значит неправильный Pin был введен хотя бы раз с момента последней успешной аутентификации |
| CKF_USER_PIN_FINAL_TRY | 0x00020000 | Если установлен, значит при вводе некорректного Pin токен будет заблокирован |
| CKF_USER_PIN_LOCKED | 0x00040000 | Если установлен, значит токен заблокирован для входа пользователя |
| CKF_USER_PIN_TO_BE_CHANGED | 0x00080000 | Если установлен, значит установлен стандартный Pin или истек срок действия текущего Pin |
| CKF_SO_PIN_COUNT_LOW | 0x00100000 | Если установлен, значит неправильный Pin администратора был введен хотя бы раз с момента последней успешной аутентификации |
| CKF_SO_PIN_FINAL_TRY | 0x00200000 | Если установлен, значит при вводе некорректного Pin администратора токен будет заблокирован |

Продолжение табл. 1

| Имя флага | Маска | Значение |
|---|---|---|
| CKF_SO_PIN_LOCKED | 0x00400000 | Если установлен, значит вход администратора токена заблокирован |
| CKF_SO_PIN_TO_BE_CHANGED | 0x00800000 | Если установлен, значит установлен стандартный Pin администратора или истек срок действия текущего Pin |

Выведите в консоль необходимые поля в соответствии с заданным вариантом.

Осталось добавить функцию, которая будет в отдельном потоке проверять возникновение событий подключения и отключения eToken к компьютеру. За ожидание возникновения данных событий отвечает функция `C_WaitForSlotEvent()`. Вставьте данный код в проект:

```
static DWORD __stdcall TokenNotifyThread(void*)
{
       while (true)
       {
              DWORD slotId;
              int res = pFunctionList->C_WaitForSlotEvent(0,
&slotId, 0);

              if (res==CKR_OK) displayTokenInfo(slotId);
    else break;
       }
       return 0;
}
```

Данная функция в бесконечном цикле при возникновении одного из событий вызывает функцию `displayTokenInfo`, которая либо выводит информацию о подключенном eToken, либо сообщает что eToken был извлечен. Для того, чтобы запустить в отдельном потоке данную функцию, необходимо добавить в основную функцию main следующий код:

```
hThread=CreateThread(NULL,0,TokenNotifyThread,NULL,0,N
ULL);
```

Дополните функцию main, чтобы в ней производилась инициализация библиотеки, выводилась информация о библиотеке, запускался поток ожидания событий и завершалась работа библиотеки.

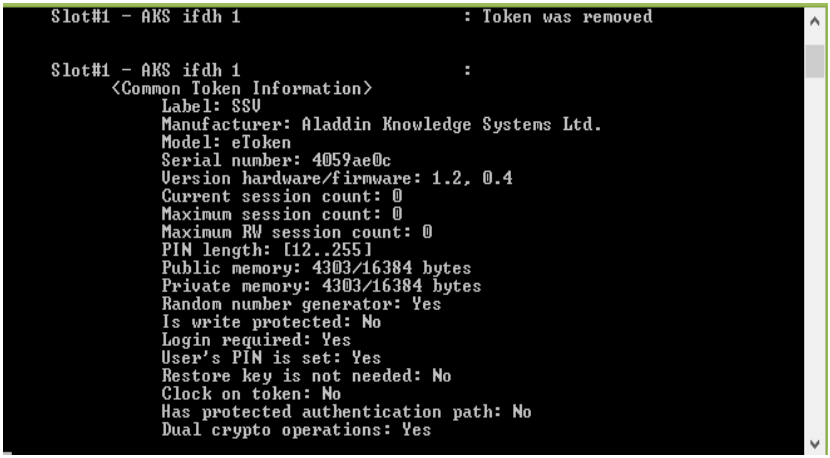



Рис. 2 – Пример вывода информации о eToken

## 4. Задание на лабораторную работу

Необходимо написать программу, которая будет производить инициализацию библиотеки PKCS#11 для eToken, выводить информацию о данной библиотеке, в отдельном потоке получать информацию о событиях подключения/отключения eToken. В случае возникновения события подключения eToken должен производится вывод информации о подключенном eToken. Для выполнения работы необходимо использовать среду разработки Microsoft Visual Studio 2012 (и выше) и eToken PKI Client версии не менее, чем 5.0. Тип выводимой информации должен соответствовать варианту (Табл. 2).

Таблица 2

Варианты заданий работы с пользователями

| № варианта | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| label | + | | + | | + | | + | | + | | + | | + | | + | | + | | + | |
| manufacturerID | | + | | + | | + | | + | | + | | + | | + | | + | | + | | + |
| model | + | | | + | | | + | | | + | | | + | | | + | | | + | |
| serialNumber | + | + | + | + | + | + | + | + | + | + | + | + | + | + | + | + | + | + | + | + |
| ulMaxSessionCount | | + | | | + | | | + | | | + | | | + | | | + | | | + |
| ulSessionCount | | | + | | | + | | | + | | | + | | | + | | | + | | |
| ulMaxRwSessionCount | | | + | | | | + | | | | + | | | | + | | | | + | |
| ulRwSessionCount | | | | + | | | | + | | | | + | | | | + | | | | + |
| ulMaxPinLen | + | | | | + | | | | + | | | | + | | | | + | | | |
| ulMinPinLen | | + | | | | + | | | | + | | | | + | | | | + | | |

Продолжение табл. 2

| № варианта | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ulTotalPublicMemory | + | | + | | + | | + | | + | | + | | + | | + | | + | | + | |
| ulFreePublicMemory | + | | + | | + | | + | | + | | + | | + | | + | | + | | + | |
| ulTotalPrivateMemory | | + | | + | | + | | + | | + | | + | | + | | + | | + | | + |
| ulFreePrivateMemory | | + | | + | | + | | + | | + | | + | | + | | + | | + | | + |
| CKF_RNG | + | | + | | + | | + | | + | | + | | + | | + | | + | | + | |
| CKF_WRITE_PROTECTED | | + | | + | | + | | + | | + | | + | | + | | + | | + | | + |
| CKF_LOGIN_REQUIRED | | | + | | | | + | | | | + | | | | + | | | | + | |
| CKF_USER_PIN_INITIALIZED | | | | + | | | | + | | | | + | | | | + | | | | + |
| CKF_USER_PIN_COUNT_LOW | + | | | | + | | | | + | | | | + | | | | + | | | |
| CKF_USER_PIN_LOCKED | | + | | | | + | | | | + | | | | + | | | | + | | |

## 5. Контрольные вопросы

1. Назовите основные отличия между API, используемыми для работы с eToken.
2. Каким образом производится инициализация PKCS#11?
3. Каким образом можно посмотреть список функций, доступных в библиотеке PKCS#11?
4. Какую информацию можно получить с помощью функции C_GetSlotInfo?
5. Какую информацию можно получить с помощью функции C_GetTokenInfo?
6. Для чего нужна функция C_WaitForSlotEvent?
7. Каким образом можно определить наличие флага?

# ЛАБОРАТОРНАЯ РАБОТА №2
# Разработка подсистемы криптографической защиты

## 1. Цель работы

Целью работы является ознакомление с сертификатами X.509, изучение функций eToken API, связанные с импортом сертификатов на eToken, получить навыки производить операцию входа пользователя на eToken.

## 2. Краткие теоретические сведения

*X.509* – стандарт, определяющий форматы данных и процедуры распределения открытых ключей с помощью сертификатов с цифровыми подписями, которые предоставляются сертификационными органами (CA). RFC 1422 создает основу для PKI на базе X.509.

Для технологии открытых ключей необходимо, чтобы пользователь открытого ключа был уверен, что этот ключ принадлежит именно тому удаленному субъекту (пользователю или системе), который будет использовать средства шифрования или цифровой подписи. Такую уверенность дают сертификаты открытых ключей, то есть структуры данных, которые связывают величины открытых ключей с субъектами. Эта связь достигается цифровой подписью доверенного CA под каждым сертификатом. Сертификат имеет ограниченный срок действия, указанный в его подписанном содержании. Поскольку пользователь сертификата может самостоятельно проверить его подпись и срок действия, сертификаты могут распространяться через незащищенные каналы связи и серверные системы, а также храниться в кэш-памяти незащищённых пользовательских систем. Содержание сертификата должно быть одинаковым в пределах всего PKI. В настоящее время в этой области предлагается общий стандарт для Интернет с использованием формата X.509 v3:

– Номер версии.
– Серийный номер.
– Эмитент.
– Субъект.
– Открытый ключ субъекта (алгоритм, ключ).
– Период действия.
– Дополнительные (необязательные) значения.
– Алгоритм подписи сертификата.
– Значение подписи сертификата.

Пример структуры сертификата:

```
SEQUENCE(3 elem)
  SEQUENCE(7 elem)
    [0](1 elem)
      INTEGER 2
    INTEGER 1
    SEQUENCE(2 elem)
      OBJECT IDENTIFIER 1.2.840.113549.1.1.5
      NULL
    SEQUENCE(1 elem)
      SET(1 elem)
        SEQUENCE(2 elem)
          OBJECT IDENTIFIER 2.5.4.3
          UTF8String CA
    SEQUENCE(2 elem)
      UTCTime 13-09-15 15:35:02 UTC
      GeneralizedTime 2113-09-22 15:35:02 UTC
    SEQUENCE(1 elem)
      SET(1 elem)
        SEQUENCE(2 elem)
          OBJECT IDENTIFIER 2.5.4.3
          UTF8String CA
    SEQUENCE(2 elem)
      SEQUENCE(2 elem)
        OBJECT IDENTIFIER 1.2.840.113549.1.1.1
        NULL
      BIT STRING(1 elem)
        SEQUENCE(2 elem)
          INTEGER 00: 8D 80 B5 8E 80 8E 94 D1 04 03 6A 45 1A 54 5E 7E
                EE 6D 0C CB 0B 82 03 F1 7D C9 6F ED 52 02 B2 08
                C3 48 D1 24 70 C3 50 C2 1C 40 BC B5 9D F8 E8 A8
                41 16 7B 0B 34 1F 27 8D 32 2D 38 BA 18 A5 31 A9
                E3 15 20 3D E4 0A DC D8 CD 42 B0 E3 66 53 85 21
                7C 90 13 E9 F9 C9 26 5A F3 FF 8C A8 92 25 CD 23
                08 69 F4 A2 F8 7B BF CD 45 E8 19 33 F1 AA E0 2B
                92 31 22 34 60 27 2E D7 56 04 8B 1B 59 64 77 5F
          INTEGER 65537
    SEQUENCE(2 elem)
      OBJECT IDENTIFIER 1.2.840.113549.1.1.5
      NULL
  BIT STRING 00: 0A 1C ED 77 F4 79 D5 EC 73 51 32 25 09 61 F7 00
        C4 64 74 29 86 5B 67 F2 3D A9 39 34 6B 3C A9 92
        B8 BF 07 13 0B A0 9B DF 41 E2 8A F6 D3 17 53 E1
        BA 7F C0 D0 BC 10 B7 9B 63 4F 06 D0 7B AC C6 FB
        CE 95 F7 8A 72 AA 10 EA B0 D1 6D 74 69 5E 20 68
        5D 1A 66 28 C5 59 33 43 DB EE DA 00 80 99 5E DD
        17 AC 43 36 1E D0 5B 06 0F 8C 6C 82 D3 BB 3E 2B
        A5 F1 94 FB 53 7B B0 54 22 6F F6 4C 18 1B 72 1C
```

X.509-сертификаты хранятся, как правило, в виде DER (стандартное расширение .cer) или PEM-файлов. Файл *.crt хранит информацию о сертификате в закодированном виде с использованием кодировки ASN.1. ASN.1 — стандарт записи, описывающий структуры данных для представления, кодирования, передачи и декодирования данных.

## 3. Ход работы

Для начала, необходимо создать сертификат, который будет импортирован на eToken. Данный сертификат необходимо создать с использованием утилиты *makecert.exe***,** которая входит в пакет средств обеспечения безопасности платформы .Net Framework (http://msdn.microsoft.com/ru-ru/library/bfsktky3(v=vs.110).aspx ). Также эта утилита может быть запущена через консоль разработчика Visual Studio (Developer Command Prompt) в меню Пуск. Данный сертификат должен быть издан от имени центра сертификации, сертификат и закрытый ключ которого будут выданы преподавателем. Предварительно необходимо импортировать данный сертификат в ваше хранилище сертификатов (файл teacher.pfx, пароль 123).

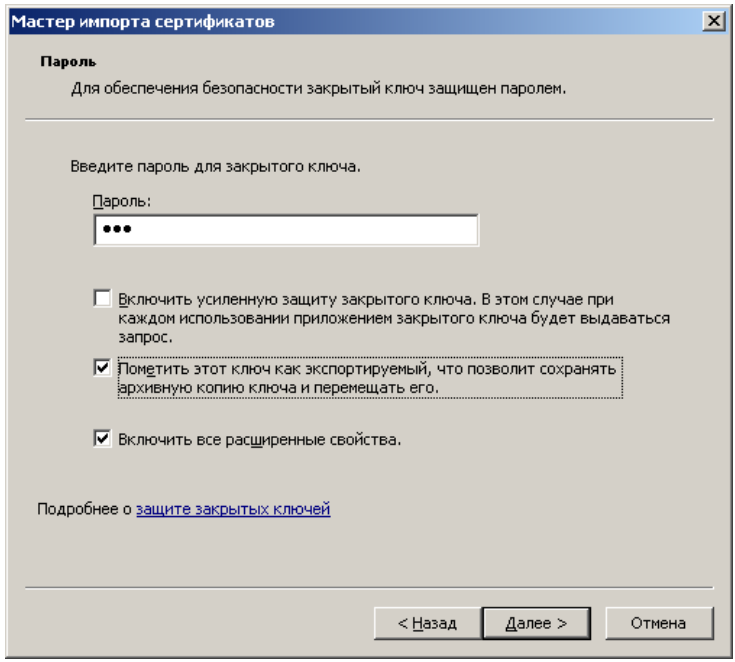



Рисунок 1 – Импорт сертификата преподавателя

При создании сертификата используйте следующие параметры командной строки, желтым выделены параметры, которые вам нужно будет изменить:

-ic “Путь сертификата поставщика (преподавателя)”;

-ik t2 (Имя контейнера закрытого ключа поставщика (преподавателя));

-n “CN=ФИО и № группы”;

-sk “Имя контейнера с закрытым ключом”;

-pe;

-ss My;

-len 1024;

“Путь к файлу создаваемого сертификата”.

Созданный сертификат будет находиться в хранилище личных сертификатов, которое можно открыть через командную строку – certmgr.msc, а также в файле по указанному вами пути.

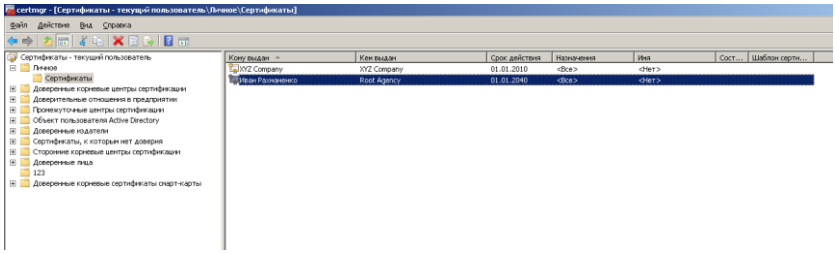



Рисунок 2 – Созданный сертификат в хранилище личных сертификатов

Созданный таким образом сертификат студента будет использоваться далее для импорта на eToken.

Загрузите проект для Visual Studio под названием Lab2. В данном проекте содержится код, производящий инициализацию библиотеки PKCS#11. Добавьте заголовки функций, которые необходимо будет запрограммировать:

```
static void ImportCertificate(const char* fileName, const
char* password);
static bool ReadCertFromFile(const char* fileName,
CK_BYTE_PTR* cert, DWORD* certSize);
static void GetX509Subject(CK_BYTE_PTR cert, int certSize,
CK_BYTE_PTR* subject, int* subjectSize);
static bool CreateCertFromBlob(CK_SESSION_HANDLE hSession,
CK_BYTE_PTR cert, int certSize, CK_BYTE_PTR subject, int
subjSize);
```

Здесь описаны заголовки функций, которые в дальнейшем будут использоваться для считывания и импорта сертификата. Функция ImportCertificate(const char* fileName, const char* password) обеспечивает импорт сертификата по пути *filename* на eToken с ПИН-кодом *password*, функция ReadCertFromFile(const char* fileName, CK_BYTE_PTR* cert, DWORD* certSize) обеспечивает чтение файла с сертификатом, функция GetX509Subject(CK_BYTE_PTR cert, int certSize, CK_BYTE_PTR* subject, int* subjectSize) предназначена для получения темы сертификата, функция CreateCertFromBlob(CK_SESSION_HANDLE hSession, CK_BYTE_PTR cert, int certSize, CK_BYTE_PTR subject, int subjSize) предназначена для создания файла с сертификатом на eToken.

Добавьте в функцию main код, позволяющий считывать путь к сертификату и ПИН-код с консоли (можно использовать функцию **gets(char* buffer)**).

Объявите следующие переменные в теле функции ImportCertificate:

```
CK_BYTE_PTR          cert = NULL;
DWORD                              certSize;
CK_BYTE_PTR          subject = NULL;
int                                      subjSize;
```

Функция `ReadCertFromFile(const char* fileName, CK_BYTE_PTR* cert, DWORD* certSize)` считывает сертификат, расположенный по пути fileName, массив считанных данных – cert, размер считанного сертификата – certSize. Один из способов, с помощью которого в C++ можно считать файл – создание дескриптора файла с помощью функции WinAPI CreateFileA(), получение размера файла с помощью функции GetFileSize(), чтение самого файла с помощью функции ReadFile(). Найдите дополнительную информацию о данных функциях. Запрограммируйте функцию ReadCertFromFile.

Рассмотрим функцию `GetX509Subject(CK_BYTE_PTR cert, int certSize, CK_BYTE_PTR* subject, int* subjectSize)`. Данная функция извлекает информацию о субъекте, которому принадлежит сертификат. Т.к. сертификат x.509 имеет сложную структуру (см. теоретические сведения), задача извлечения данной информации является непростой, поэтому предлагается использовать следующий код:

```
static void GetX509Subject(CK_BYTE_PTR cert     //
Сертификат в двоичном представлении
, int certSize                                  // Размер
сертификата
, CK_BYTE_PTR* subject                    // Указатель на
массив информации о субъекте
, int* subjectSize                        // Размер
информации о субъекте
) {
       unsigned int i;
       unsigned char* current = cert;
       unsigned char* prev;
       unsigned char* end = cert + certSize;
       unsigned char tags[] = {0x30, 0,             //
основная последовательность
                               0x30, 0,          //
последовательность, подписываемая ЭЦП
                               0xa0, 1,       // версия
```

```
                                0x02, 1,        // серийный
номер
                                0x30, 1,        // подпись
                                0x30, 1,        // издатель
                                0x30, 1,        // период
действия
                                0x30, 1,        // информация
о субъекте
                                };
        *subject = NULL;
        *subjectSize = 0;
        for (i=0; i<sizeof(tags); i+=2)
        {
                unsigned char v;
                unsigned int length = 0;
                prev = current;
                if (current+2>end) return;
                if (*current++ != tags[i]) return;
                v = *current++;

                if ((v & 0x80) == 0) length = v;
                else
                {
                   unsigned char lenlen = v & 0x7f;
                   if (current + lenlen > end) return;
                   for (v=0; v<lenlen; v++) length =
(length<<8) + *current++;
                }
                if (tags[i+1]) current+=length;
        }
        *subject = prev;
        *subjectSize = (int)(current-prev);
}
```

Рассмотрим функцию `CreateCertFromBlob(CK_SESSION_HANDLE hSession, CK_BYTE_PTR cert, int certSize, CK_BYTE_PTR subject, int subjSize)`. Данная функция предназначена для записи сертификата на eToken. Обратите внимание, что одним из параметров данной функции является переменная типа `CK_SESSION_HANDLE hSession`. Данная переменная является дескриптором сессии работы с токеном. Для того, чтобы получить данный дескриптор, используйте функцию `C_OpenSession()` из списка функций `pFunctionList`. В качестве параметров к данной функции передаются идентификатор

слота, к котрому подключен eToken, флаги CKF_SERIAL_SESSION и CKF_RW_SESSION, 2 параметра NULL, так как не используется механизм обратного вызова и уведомления, и ссылка на дескриптор &hSession. Кроме того, необходимо произвести вход под ролью пользователя на токен, с использованием функцию C_Login() из списка функций pFunctionList. В качестве параметров к данной функции передаются дескриптор сессии hSession, роль пользователя CKU_USER, ПИН-код в формате BYTE (LPBYTE)password, длина ПИН-кода strlen(password).

Рассмотрим код данной функции CreateCertFromBlob:

```
static bool CreateCertFromBlob(CK_SESSION_HANDLE hSession
	// ID сессии
				, CK_BYTE_PTR cert
		// сертификат
				, int certSize
		// размер сертификата
				, CK_BYTE_PTR subject
	// субъект сертификата
				, int subjSize
		// размер субъекта
				)
{
	CK_OBJECT_CLASS classAttr = CKO_CERTIFICATE;
	CK_CERTIFICATE_TYPE certType = CKC_X_509;
	unsigned long certCategory = 2;
	CK_BBOOL trueVal = CK_TRUE;
	CK_OBJECT_HANDLE hObject;

	CK_ATTRIBUTE templateArray [] =
	{
		{CKA_CLASS, &classAttr, sizeof(classAttr)},
		{CKA_CERTIFICATE_TYPE, &certType,
sizeof(certType)},
		{CKA_TOKEN, &trueVal, sizeof(trueVal)},
		{CKA_SUBJECT, subject, subjSize },
		{CKA_VALUE, (void *)cert, certSize },
		{CKA_CERTIFICATE_CATEGORY, (void
*)&certCategory, sizeof (certCategory) },
	};
	int sizeOfTemplate = sizeof (templateArray) /
sizeof(CK_ATTRIBUTE);
```

```
	CK_RV rv = pFunctionList->C_CreateObject ( hSession,
templateArray, sizeOfTemplate, &hObject );
	if (rv) {
		return false;
	}
	return true;
	}
```

Здесь создается объект класса CKO_CERTIFICATE, выбирается тип сертификата CKC_X_509, категория сертификата 2, дескриптор создаваемого объекта hObject, затем создается набор атрибутов, которые присущи объекту “Сертификат x.509” на eToken, в состав которого входят указанные ранее значения, а также сам сертификат и субъект сертификата в двоичном представлении. Создается сертификат с помощью функции C_CreateObject, которая создает объект на токене с указанным набором атрибутов templateArray.

Также здесь используется переменная rv – она отображается результат выполнения запроса к библиотеке eToken API. Если rv = 0, значит запрос был выполнен корректно, в противном случае значение этой переменной будет соответствовать коду ошибки. Все коды ошибок находятся в файле pkcs11t.h, начинаясь с CKR.

Для получения идентификатора слота, к которому подключен eToken используйте следующую функцию:

```
static CK_ULONG GetFirstSlotId() {
	CK_ULONG slotID = -1;
	CK_ULONG ulCount = 0;
	CK_SLOT_ID_PTR pSlotIDs	= NULL_PTR;
	CK_ULONG i;
	if (pFunctionList-
>C_GetSlotList(TRUE,NULL_PTR,&ulCount) == CKR_OK) {
	if (ulCount > 0) {
		pSlotIDs = new CK_SLOT_ID[ulCount];

		if ((pFunctionList-
>C_GetSlotList(TRUE,pSlotIDs,&ulCount)) == CKR_OK) {
			for (i=0;i < ulCount;i++){
			CK_SLOT_INFO info;
			if ((pFunctionList-
>C_GetSlotInfo(pSlotIDs[i],&info)) == CKR_OK) {
			if (info.flags & (CKF_HW_SLOT |
CKF_TOKEN_PRESENT)) {
```

```
                                                        slotID =
pSlotIDs[i];
                                                        break;
                                                }
                                        }
                                }
                        }
                }
        }

        if (pSlotIDs) {
                delete[] pSlotIDs;
                pSlotIDs = NULL_PTR;
        }
        return slotID;
        }
```

Дополните функцию `ImportCertificate` процедурами закрытия сессии:

```
pFunctionList->C_Logout( hSession );
pFunctionList->C_CloseSession( hSession );
```

Проверьте работоспособность программы. Загрузите с помощью нее созданный сертификат на eToken. Проверьте наличие сертификата с помощью eToken PKI Client (Значок в панели задач –> Open eToken Properties -> Advanced View).

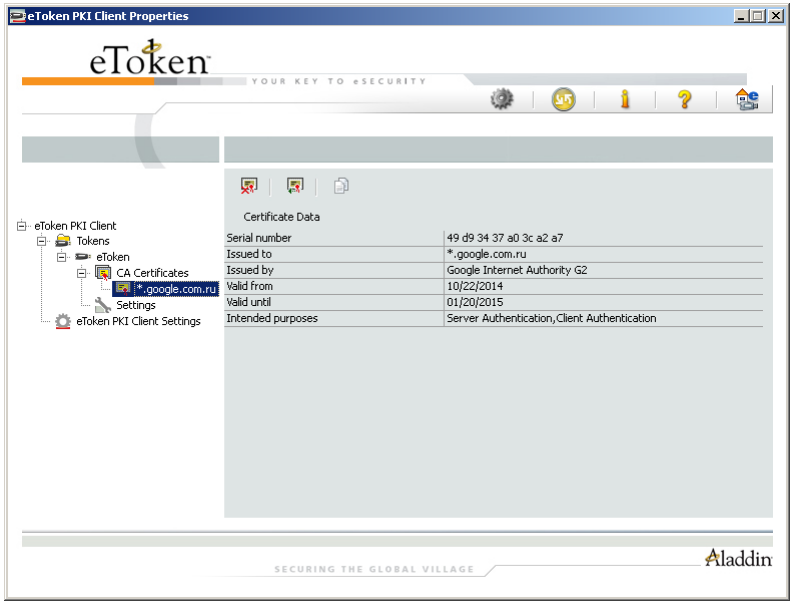



Рисунок 3 – Проверка наличия сертификата на eToken

## 4. Задание на лабораторную работу

Необходимо создать сертификат студента, подписанный с помощью сертификата преподавателя. Требуется написать программу, которая импортирует сертификат x.509 в DER кодировке на eToken. В данной программе необходимо: произвести инициализацию библиотеки PKCS#11 для eToken, запросить путь к сертификату на жестком диске, ПИН для подключения к eToken, произвести операцию Login к eToken, считать сертификат с жестокого диска, скопировать сертификат на eToken. Проверить наличие сертификата с помощью eToken PKI Client.

## 5. Контрольные вопросы

1. Что такое x.509?
2. Опишите структуру сертификата x.509.
3. Опишите формат использования утилиты makecert.exe?
4. Каким образом производится подключение пользователя к eToken?
5. Как создается сертификат x.509 на eToken?

6. Каким образом можно проверить наличие сертификата на eToken?

# ЛАБОРАТОРНАЯ РАБОТА №3
# Разработка подсистемы обеспечения целостности

## 1. Цель работы

Целью работы является ознакомление с типами объектов, которые позволяет хранить и использовать eToken, приобретение навыков осуществлять операции, связанные с чтением, записью, удалением и поиском объектов на eToken.

## 2. Краткие теоретические сведения

*Закрытый ключ RSA.*

В алгоритме шифрования RSA, к отрытому ключу относят открытую экспоненту и модуль {e,n}, к закрытому ключу закрытую экспоненту и модуль {d,n} соответственно https://ru.wikipedia.org/wiki/RSA. eToken для ускорения вычислений, использует другие компоненты – сами простые числа p и q, функцию эйлера и две дополнительные экспоненты, которые также относят к закрытому ключу. Можно рассмотреть примерную структуру PEM файла, в соответствии с которой компоненты закрытого ключа расположены. В соответствии с данной структурой, перед каждым из компонентов ключа, предварительно указывается длина компонента, затем идет сам компонент.

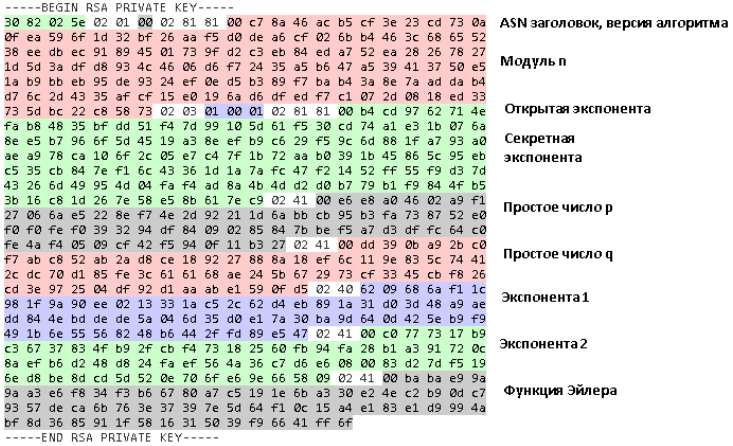



Рисунок 1 – Структура PEM файла

В случае отсутствия или некорректности одного из компонентов закрытого ключа, при создании ключа на токене, будет выдана ошибка, а сам закрытый ключ на токен не будет записан.

*Функции eToken API.*

Для работы с объектами на токене, имеются функции eToken API представленные в табл. 1.

Таблица 1

Функции для работы с объектами eToken

| Название функции | Параметры | Описание функции |
|---|---|---|
| C_CreateObject | CK_SESSION_HANDLE hSession, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount, CK_OBJECT_HANDLE_PTR phObject | С помощью данной функции на токене создается объект, соответствующий шаблону pTemplate, с количеством атрибутов шаблоне ulCount. Идентификатор созданного объекта возвращается в phObject. |
| C_GetAttributeValue | CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hObject, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount | С помощью данной функции можно получить значение атрибута или набора атрибутов шаблона pTemplate,соответствующих объекту hObject на токене. Количество атрибутов в шаблоне – ulCount. При этом фукнцию приходится вызывать 2 раза – сначала для получения длин атрибутов в байтах, затем, после выделения памяти под атрибуты, вторым вызовом функции они передаются с токена. |

Продолжение табл. 1

| Название функции | Параметры | Описание функции |
|---|---|---|
| C_FindObjectsInit | CK_SESSION_HANDLE hSession, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount | Производит инициализацию поиска, задав шаблон pTemplate, по которому будет производится поиск объектов на токене. ulCount – количество атрибутов в шаблоне. |
| C_FindObjects | CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE_PTR phObject CK_ULONG maxObjectCount, CK_ULONG_PTR pulObjectCount | Производит поиск идентификаторов объектов, соответствующих заданному ранее шаблону, максимальное количество возвращаемых функцией идентификаторов – maxObjectCount, реальное число возвращенных объектов – pulObjectCount. |
| C_FindObjectsFinal | CK_SESSION_HANDLE hSession | Завершает поиск объектов. |
| C_DestroyObject | CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hObject | Удаляет объект с идентификатором hObject. |

Продолжение табл. 1

| Название функции | Параметры | Описание функции |
|---|---|---|
| C_CopyObject | CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hObject, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount, CK_OBJECT_HANDLE_PTR phNewObject | Копирует объект с идентификатором hObject, с шаблоном для нового объекта pTemplate, число атрибутов в шаблоне-ulCount, идентификатор копии – phNewObject. |
| C_SetAttributeValue | CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hObject, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount | Задает значение атрибута(-ов) объекта с идентификатором hObject, заданных в шаблоне pTemplate, количество атрибутов ulCount. |

*Объекты eToken*

Существует несколько классов объектов, которые могут храниться и с которыми может работать eToken:

- CKO_DATA. Объект типа данные, может содержать любую информацию, которая может использоваться сторонними приложениями. В случае наличия атрибута CKA_PRIVATE, объект будет скрыт до момента ввода ПИН-кода пользователя.

- CKO_CERTIFICATE. Объект – сертификат открытого ключа (поддерживаются только сертификаты X.509). Бывают двух категорий, обычный сертификат (CKA_CERTIFICATE_CATEGORY=0) и сертификат Центра Сертификации (CKA_CERTIFICATE_CATEGORY=2).

- CKO_PRIVATE_KEY, CKO_PUBLIC_KEY. Объекты – закрытый и открытый ключи шифрования. eToken PRO поддерживает только алгоритм и ключи RSA. Экспорт закрытого ключа с токена в целях обеспечения безопасности не поддерживается.

- CKO_SECRET_KEY. Объект – секретный ключ симметричного алгоритма шифрования. Поддерживаются алгоритмы DES, DES2, DES3, HOTP, AES, RC4. Можно задать в атрибутах тип алгоритма CKK_GENERIC_SECRET, в случае наличия требований дополнительной защиты для ключа шифрования, реализация которого не поддерживается токеном.

*Шаблоны объектов eToken*

Для работы с объектами используются шаблоны – объекты типа CK_ATTRIBUTE, содержащие набор атрибутов, свойственных либо конкретному типу объектов, либо общие для всех объектов. Шаблоны имеют следующую структуру:

```
typedef struct CK_ATTRIBUTE
{
CK_ATTRIBUTE_TYPE type; /* тип атрибута */
CK_VOID_PTR       pValue; /* значение атрибута */
CK_ULONG                ulValueLen;   /* длина pValue
(количество байт)*/
}
```

.В качестве значений могут использоваться как массивы, так и одиночные переменные. Следующие атрибуты могут быть заданы для всех типов объектов:

CKA_CLASS – класс объекта;

CKA_TOKEN – объект находится на токене, если истина (всегда истина для аппаратных токенов);

CKA_VALUE – значение, содержит хранимую объектом информацию (необязательный атрибут);

CKA_LABEL – метка объекта (название);

CKA_APPLICATION – идентификатор приложения, которое работает с объектом;

CKA_PRIVATE – объект приватный, если истина, он не виден до момента аутентификации пользователя.

Атрибуты, необходимые для создания сертификата на токене:

CKA_CLASS – класс объекта (CKO_CERTIFICATE);

CKA_TOKEN – объект находится на токене, если истина;

CKA_VALUE – сертификат в формате x.509;

CKA_SUBJECT– информация о субъекте;

CKA_CERTIFICATE_CATEGORY – категория сертификата (обычный или сертификат ЦС);

CKA_CERTIFICATE_TYPE – тип сертификата (поддерживается сертификат RSA - CKC_X_509).

Объект класса закрытый ключ RSA, должен иметь набор атрибутов:

CKA_CLASS – класс объекта (CKO_PRIVATE_KEY);

CKA_TOKEN – объект находится на токене, если истина;

CKA_KEY_TYPE – алгоритм ключа (CKK_RSA);

CKA_MODULUS – модуль;
CKA_PUBLIC_EXPONENT – открытая экспонента;
CKA_PRIVATE_EXPONENT – закрытая экспонента;
CKA_PRIME_1 – простое число 1;
CKA_PRIME_2 – простое число 2;
CKA_EXPONENT_1 – экспонента 1;
CKA_EXPONENT_2 – экспонента 2;
CKA_COEFFICIENT – значение функции Эйлера.

## 3. Ход работы

*Подготовка закрытых ключей и данных.*

Первым шагом необходимо получить закрытый ключ, соответствующий созданному в прошлой лабораторной работе сертификату. Данный закрытый ключ можно извлечь вместе с сертификатом из хранилища личных сертификатов, которое можно открыть набрав в консоли *certmgr.msc* (Рис. 2).

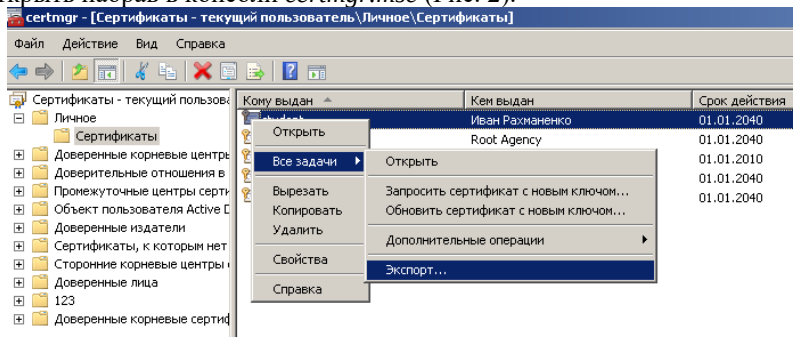



Рисунок 2 – Экспорт личного сертификата

При экспорте сертификата необходимо выбрать экспорт закрытого ключа вместе с сертификатом, выбрать формат pfx и задать пароль на контейнер.

Далее, извлечем закрытый ключ в DER-кодировке с помощью программы *OpenSSL*. Для этого в командной строке следующую строку, подставив пути к контейнеру и выходному файлу:

openssl pkcs12 -in student.pfx -out student.pem -nodes -nocerts

Также потребуется ввести пароль от контейнера (он не отображается). Данная команда из контейнера формата PKCS#12 извлекает закрытый ключ без сертификата открытого ключа (-nocerts) в незашифрованном виде (-nodes). Дополнительную информацию о параметрах можно найти по адресу

https://www.openssl.org/docs/apps/pkcs12.html. В реальных условиях хранить ключи в незашифрованном виде не рекомендуется.

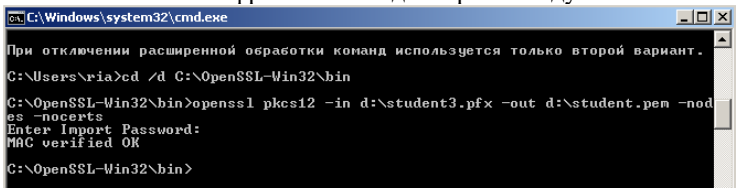



Рисунок 3 – Экспорт закрытого ключа

Создайте два текстовых файла, содержащих конфиденциальные сведения. Информация, содержащаяся в данных файлах будет записана на eToken.

*Импорт закрытого ключа на eToken.*

Создайте функцию `static void ImportPrivateKey(const char* fileName, const char* password)`, в которой будет производиться импорт закрытого ключа, находящегося в pem файле, находящемуся по пути Filename, с ПИН-ом токена password. В качестве образца можно использовать функцию импорта сертификата на токен.

Закрытый ключ, находящийся в pem файле находится в BASE64 кодировке, используемой для передачи двоичной информации в текстовых сообщениях. Для того, чтобы разобрать закрытый ключ на составляющие, его необходимо сначала раскодировать в бинарный формат.

```
Bag Attributes
    friendlyName: s3
    localKeyID: 01 00 00 00
    Microsoft CSP Name: Microsoft Strong Cryptographic Provider
Key Attributes
    X509v3 Key Usage: 80
-----BEGIN PRIVATE KEY-----
MIICdgIBADANBgkqhkiG9w0BAQEFAASCAmAwggJcAgEAAoGBAO/70uFHMXP449gX
dZhZRy99nTLf4gi/jyQyq6GC0EqcGkqJHqLxI6S1loH99QT/+2NiYhKEXbCknoxZ
YegmrYrsDG6ruST/rBgxP+Qnemyt+tlEb67VC7zexuOUGXWNSbih6Ukxt46A6doY
tdVQwBBnxI3+Cs/a8ERyW8u8/mxTAgMBAAECgYEAk0nsiMVoLbSBKpxjWUjaLuSa
uaG9Gf9v/Bh5TFx8OH6MiFiWDA2SAcJNiKT0X8g4Yyv74z+IPSUcBinJtHEgquWG
ysTSDHriLk5OL05qyWrcPVOL6H+PPf9zJvIDoAP+5l5Wp94HGB2bJF42aKo1W5V5
T1/i1KS+PicN46jeT/ECQQD/ejjt1Y/NrMydiBksxqmimp7JF31JiWr3miC0j7Hi
v9MMPvAXzMSmJsfWlQSDKOQ8JxUPvAc92kAZzaiwkw8pAkEA8Hl8/rgZNwVg5SLX
Fk+V3w/bKwxaiSzInA9So+YjhKn4vyYxowp5/PT4JDbvSQLXqDHjNfIJWY8nTEcd
+J6bGwJAFgFkF/BWQ6QHt+X6LSYrhGsHFFEmpIfmo88tO/+wldRqDFr7eUFVCu66
NeIBUn7aMeg34zQR95zt+osBZbw1CQJARLrEUdz2P/aGkhbaGKNYd2+FuZEHU0t8
brYmbXe6URFvadqVtz6oTPcK8PbOfzSTxJu1j9aEe5dhqPhuhX93XwJABdX52NVd
VHwZ2BamORI3Hxt821thPDKFAaoOJDp/+YhhkcSlFwMRCgWri9wOPbK40pLSWYOx
a7s5wzuEv9PmYA==
-----END PRIVATE KEY-----
```

Рисунок 4 – Содержимое pem файла с закрытым ключом

Функция `static void GetPKBlob(CK_BYTE_PTR pk, int pkSize, CK_BYTE_PTR* subject, int* subjectSize)` раскодирует данные между блоками Begin private key и End Private key в двоичное представление. Эти данные находятся в DER кодировке, которую тоже необходимо раскодировать.

Используйте следующий код для извлечения компонент закрытого ключа RSA из массива байт key:

```
if (key[0]!=0x30) return false; //неверный формат
	BYTE *modulus, *pubE, *privE,*p1, *p2, *e1, *e2,*coeff;
//компоненты закр. ключа
	int modulusL, pubEL, privEL, p1L, p2L, e1L, e2L, coeffL;
//длины компонентов

//напишите более красивый парсер, если сможете
	int cur=0;
	while
((key[cur]!=0x02)||((key[cur+1]!=0x81)&&(key[cur+1]!=0x82)))
	{
		cur++;
	}
	cur++;

	if (key[cur] ==0x81) {modulusL = key[cur+1];
modulus=key+cur+2; if (key[cur+2]==0) {modulusL--; modulus++;
cur++;} cur = cur+2+modulusL;}
	if (key[cur] ==0x82) {modulusL =
key[cur+1]<<8|key[cur+2]; modulus=key+cur+3; if (key[cur+3]==0)
{modulusL--; modulus++; cur++;} cur = cur+3+modulusL;}

	if (key[cur] !=0x02) return false; cur++;
	pubEL = key[cur]; pubE=key+cur+1; cur = cur+1+pubEL;

	if (key[cur] !=0x02) return false; cur++;

	if (key[cur] ==0x81) {privEL = key[cur+1];
privE=key+cur+2; if (key[cur+2]==0) {privEL--; privE++; cur++;}
cur = cur+2+privEL;}
	if (key[cur] ==0x82) {privEL = key[cur+1]<<8|key[cur+2];
privE=key+cur+3; if (key[cur+3]==0) {privEL--; privE++; cur++;}
cur = cur+3+privEL;}

	if (key[cur] !=0x02) return false; cur++;
```

```
        if ((key[cur] ==0x81)||(key[cur] ==0x80))  cur++;
        p1L = key[cur]; p1=key+cur+1; if (key[cur+1]==0) {p1L--;
p1++; cur++;} cur = cur+1+p1L;

        if (key[cur] !=0x02) return false; cur++;
        if ((key[cur] ==0x81)||(key[cur] ==0x80))  cur++;
        p2L = key[cur]; p2=key+cur+1; if (key[cur+1]==0) {p2L--;
p2++; cur++;} cur = cur+1+p2L;

        if (key[cur] !=0x02) return false; cur++;
        if ((key[cur] ==0x81)||(key[cur] ==0x80))  cur++;
        e1L = key[cur]; e1=key+cur+1; if (key[cur+1]==0) {e1L--;
e1++; cur++;} cur = cur+1+e1L;

        if (key[cur] !=0x02) return false; cur++;
        if ((key[cur] ==0x81)||(key[cur] ==0x80))  cur++;
        e2L = key[cur]; e2=key+cur+1; if (key[cur+1]==0) {e2L--;
e2++; cur++;} cur = cur+1+e2L;

        if (key[cur] !=0x02) return false; cur++;
        if ((key[cur] ==0x81)||(key[cur] ==0x80))  cur++;
        coeffL = key[cur]; coeff=key+cur+1; if (key[cur+1]==0)
{coeffL--; coeff++;}
```

Для создания объектов на eToken используется функция `C_CreateObject(hSession, templateArray, sizeOfTemplate, &hObject)`, в качестве параметров которой передаются идентификатор сессии, массив атрибутов, количество атрибутов в массиве, идентификатор создаваемого объекта. Напишите функцию `static bool CreatePKFromBlob (CK_SESSION_HANDLE hSession, CK_BYTE_PTR key, int keySize)`, которая будет создавать закрытый ключ на токене (необходимые атрибуты для закрытого ключа см. в теоретических сведениях).

*Запись объектов данных на eToken.*

Для записи произвольных данных на eToken, также как и для создания сертификатов и закрытых ключей используется функция `C_CreateObject(hSession, templateArray, sizeOfTemplate, &hObject)`, отличия заключаются в наборе атрибутов создаваемого объекта. Для создания объекта данных используйте атрибуты Class, Token, Value, Label, Application, Private (см. теоретические сведения).

Запрограммируйте функцию `ImportFile(const char* fileName, const char* label, const char* password)`, которая

будет импортировать файл, находящийся по пути Filename, с меткой label, ПИН-ом токена password.

*Чтение объектов с eToken*

Для того чтобы считать объекты определенного типа с токена, необходимо получить идентификатор объекта, который необходимо считать, а затем получить значение атрибута Value для этого объекта. Создайте функцию `static void ExportFile(const char* fileName,const char* password)`, в которой будет производится считывание объекта с токена и запись его в файл с именем fileName.

Чтобы получить идентификаторы всех объектов с данными на токене, необходимо задать шаблон, по которому будет производиться поиск объектов, и задать массив, в который будут записаны ссылки на найденные объекты. Для поиска объектов данных на токене используйте шаблон:

```
CK_OBJECT_CLASS classAttr = CKO_DATA;
CK_BBOOL trueVal = CK_TRUE;
CK_ATTRIBUTE Template[]  =
{
              {CKA_CLASS, &classAttr, sizeof(classAttr)},
              {CKA_TOKEN, &trueVal, sizeof(trueVal)},
};
```

Для того, чтобы определить, имеется ли хоть один объект соответствующий шаблону на токене, производится инициализация поиска `C_FindObjectsInit(hSession, Template, sizeOfTemplate)`:

```
int rv = pFunctionList->C_FindObjectsInit(hSession,
Template, sizeOfTemplate);
if (rv) leave("Cannot find data");
```

Если функция возвращает значение отличное от 0, значит объектов данного типа на токене нет, либо произошла ошибка. Далее нужно произвести получение идентификаторов (`CK_OBJECT_HANDLE`) найденных объектов с помощью функции `C_FindObjects(hSession, ObjList, nObjMax, &nObjCount)`. К параметрам данной функции относят идентификатор сессии hSession, массив идентификаторов найденных объектов ObjList(не забудьте выделить память), размер массива объектов nObjMax, количество найденных на токене объектов nObjCount.

Теперь, когда известны идентификаторы объектов, нужно считать их метки Label и выбрать, какой из объектов сохранять на диск (выведите в консоль метки с предложением выбора номера одной из

них). Создайте функцию `static void ReadLabel(CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hLabel, CK_BYTE_PTR* label, DWORD* labelSize)`,которая будет считывать метку объекта с заданным идентификатором hLabel.

Чтение атрибута, в данном случае Label, осуществляется с помощью функции `C_GetAttributeValue(hSession, hLabel, &ValueTemplate, 1)` в 2 этапа. Задается шаблон этого атрибута и извлекается его размер в байтах:

```
CK_ATTRIBUTE ValueTemplate  = {CKA_LABEL, NULL, 0};
  int rv = pFunctionList->C_GetAttributeValue(hSession,
hLabel, &ValueTemplate, 1);
  if (rv) leave( "Cannot read data from the eToken");
```

Затем выделяется память и считывается сама метка:

```
  *labelSize = ValueTemplate.ulValueLen;
  *label = new BYTE[*labelSize];
  ValueTemplate.pValue = *label;
  rv = pFunctionList->C_GetAttributeValue(hSession, hLabel,
&ValueTemplate, 1);
  if (rv) leave( "Cannot read data from the eToken");
```

Вернемся к основной функции, после того, как был выбран номер метки, необходимо считать сами данные с токена. Для этого используйте функцию `C_GetAttributeValue()` аналогичным образом, создав шаблон с атрибутом `CKA_VALUE`.

Доделайте основную функцию, добавив код для записи считанных с токена данных в файл.

*Удаление объектов с eToken.*

Создайте функцию `static void DeleteData(const char* password)`, которая будет удалять объект, выбранный из списка найденных на токене. Для вывода меток доступных объектов на токене, используйте код, запрограммированный для предыдущей функции. Удалением объектов занимается функция `C_DestroyObject(hSession, hObject)`, в параметрах которой указывается идентификаторы сессии и удаляемого объекта. Запрограммируйте данную функцию.

*Проверка правильности работы функций*

С помощью меню программы импортируйте свой закрытый ключ на eToken; Загрузите 2 файла с конфиденциальными данными на eToken; Экспортируйте один из файлов на жесткий диск; Удалите данный объект с токена. Продемонстрируйте корректность работы данных функций преподавателю. Продемонстрировать наличие

объектов на eToken можно с помощью подробного вида в Safenet Authentication Client или eToken PKI Client.

## 4. Задание на лабораторную работу

1. Требуется написать программу, которая:
   - позволяет считывать закрытый ключ в формате PEM, записывать его на eToken;
   - позволяет записывать произвольные файлы на eToken;
   - позволяет искать данные на eToken и сохранять выбранные данные на жесткий диск;
   - позволяет удалять данные с eToken.
2. Необходимо извлечь закрытый ключ из хранилища Windows, записать его на eToken.
3. Создать несколько файлов с секретными сведениями и записать их на eToken.
4. Произвести поиск данных на eToken, сохранить выбранные данные на жесткий диск.
5. Удалить один из файлов с eToken.

## 5. Контрольные вопросы

1. Назовите основные компоненты ключей RSA.
2. Назовите и опишите основные функции eToken API для работы с объектами eToken.
3. Опишите основные классы объектов eToken. Для чего они предназначены?
4. Что такое шаблон объекта eToken?
5. Перечислите основные атрибуты объектов eToken.
6. Для чего предназначен атрибут CKA_PRIVATE?
7. Каким образом организован поиск объектов на токене?
8. Каким образом производится считывание объектов с токена?
9. Найдите функцию WinAPI, с помощью которой можно считать закрытый ключ сертификата (который либо находится в хранилище сертификатов, либо на жестком диске в формате pfx). Запрограммируйте запись закрытого ключа на токен с применением данной функции (+1 балл на экзамене).

# ЛАБОРАТОРНАЯ РАБОТА №4
# Проведение испытаний компонентов средств защиты информации

## 1. Цель работы

Целью работы является ознакомление с типами шифрования и механизмами, обеспечивающими генерацию ключей eToken, приобретение навыков осуществлять шифрование и расшифровывание данных с помощью eToken.

## 2. Краткие теоретические сведения

*Механизмы eToken*

Механизмы eToken (Mechanisms) позволяют выполнять и определяют каким образом должны выполняться криптографические операции. Следующие операции доступны с помощью механизмов eToken: шифрование и расшифровывание, цифровая подпись и проверка подписи, хэширование, генерация ключевых пар и ключей шифрования и др. По умолчанию нет гарантии, что конкретный токен поддерживает тот или иной механизм, поэтому перед использованием механизма необходимо убедиться, что он поддерживается токеном. Например, некоторые токены не поддерживают шифрование RSA. Кроме того, механизмы позволяют использовать разные режимы работы криптографических алгоритмов. Например, для алгоритма RSA в eToken API доступны следующие механизмы:

```
#define CKM_AES_KEY_GEN                 0x00001080
#define CKM_AES_ECB                     0x00001081
#define CKM_AES_CBC                     0x00001082
#define CKM_AES_MAC                     0x00001083
#define CKM_AES_MAC_GENERAL             0x00001084
#define CKM_AES_CBC_PAD                 0x00001085
```

Необходимо помнить, что для работы различных механизмов могут требоваться различные длины ключей шифрования и предъявляться различные требования к длине входных данных.

*Функции eToken API.*

Для шифрования и расшифровывания данных с помощью токена, имеются функции eToken API представленные в табл. 1.

Таблица 1

Функции eTokenсвязанные с шифрованием

| Название функции | Параметры | Описание функции |
|---|---|---|
| C_GenerateKey | CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount, CK_OBJECT_HANDLE_PTR phKey. | С помощью данной функции на токене создается ключ шифрования, использующий механизм pMechanism, соответствующий шаблону pTemplate, с количеством атрибутов шаблоне ulCount. Идентификатор созданного ключа шифрования возвращается в phKey. |

| | | |
|---|---|---|
| C_GenerateKeyPair | CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_ATTRIBUTE_PTR pPublicKeyTemplate, CK_ULONG ulPublicKeyAttributeCount, CK_ATTRIBUTE_PTR pPrivateKeyTemplate, CK_ULONG ulPrivateKeyAttributeCount, CK_OBJECT_HANDLE_PTR phPublicKey, CK_OBJECT_HANDLE_PTR phPrivateKey | С помощью данной функции на токене создается открытый и закрытый ключи шифрования, использующие механизм pMechanism, соответствующие шаблонам pPublicKeyTemplate и pPrivateKeyTemplate, с количеством атрибутов в шаблоне ulPublicKeyAttrivuteCount и ulPrivateKeyAttrivuteCount. Идентификаторы созданных ключей шифрования возвращаются в phPublicKey и phPrivateKey. |

Продолжение табл. 1

| Название функции | Параметры | Описание функции |
|---|---|---|
| C_EncryptInit | CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE_PTR phKey. | С помощью данной функции инициализируется начало процедуры шифрования данных с использованием механизма pMechanism и ключа шифрования phKey. Используется идентификатор сессии hSession. |

| | | |
|---|---|---|
| C_EncryptUpdate | K_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulDataLen, CK_BYTE_PTR pEncryptedData, CK_ULONG_PTR pulEncryptedDataLen. | С помощью данной функции производится шифрование блока данных pData длиной ulDataLen. Шифртекст записывается в массив pEncryptedData, количество записанных в данный массив байт - pulEncryptedDataLen. Используется идентификатор сессии hSession. |
| C_DecryptInit | CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE_PTR phKey. | С помощью данной функции инициализируется начало процедуры расшифровывания данных с использованием механизма pMechanism и ключа шифрования phKey. Используется идентификатор сессии hSession. |

Продолжение табл. 1

| Название функции | Параметры | Описание функции |
|---|---|---|

| | | |
|---|---|---|
| C_DecryptUpdate | K_SESSION_HANDLE hSession, CK_BYTE_PTR pEncryptedData, CK_ULONG ulEncryptedDataLen, CK_BYTE_PTR pData, CK_ULONG_PTR pulDataLen, | С помощью данной функции производится расшифровывание блока данных pEncryptedData длиной ulEncryptedDataLen. Открытый текст записывается в массив pData, количество записанных в данный массив байт - pulDataLen. Используется идентификатор сессии hSession. |
| C_DigestInit | CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism. | С помощью данной функции инициализируется начало процедуры хэширования данных с использованием механизма pMechanism. Используется идентификатор сессии hSession. |
| C_DigestUpdate | K_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulDataLen, CK_BYTE_PTR pDigest, CK_ULONG_PTR pulDigestLen. | С помощью данной функции производится хэширование блока данных pData длиной ulDataLen. Хэш записывается в массив pDigest, количество записанных в данный массив байт - pulDigestLen. Используется идентификатор сессии hSession. |

*Используемые объекты eToken.*

В данной лабораторной работе будет производиться работа только с одним объектом – ключом шифрования. Рассмотрим шаблон данного объекта:

```
CK_ATTRIBUTE SessionKeyTemplate[] =
  {
        { CKA_CLASS,     &cko_SecretKey,
sizeof(cko_SecretKey)},
        { CKA_KEY_TYPE,  &ckk_DES3,
sizeof(ckk_DES3)},
        { CKA_TOKEN,     &ck_False,     sizeof(ck_False)}
  };
```

В данном случае класс создаваемого объекта - CKO_SECRET_KEY, тип генерируемого ключа – ключ алгоритма DES3, CKA_TOKEN = False означает, что создается сессионный ключ, который не будет сохранен на токене.

## 3. Ход работы

Для того, чтобы создать сессионный ключ и получить его идентификатор используется функция CreateSessionKey:

```
CK_OBJECT_HANDLE
CreateSessionKey(CK_SESSION_HANDLE hSession)
{
 CK_ATTRIBUTE SessionKeyTemplate[] =
  {
        { CKA_CLASS,     &cko_SecretKey,
sizeof(cko_SecretKey)},
        { CKA_KEY_TYPE,  &ckk_DES3,
sizeof(ckk_DES3)},
        { CKA_TOKEN,     &ck_False,     sizeof(ck_False)}
  };

 CK_OBJECT_HANDLE hSessionKey = NULL;

 int rv = pFunctionList->C_GenerateKey(
  hSession,
  &ckm_DES3_KEY_GEN,
  SessionKeyTemplate,
  sizeofarray(SessionKeyTemplate),
  &hSessionKey);
 if (rv!=CKR_OK) leave("Failed to create session key");

 return hSessionKey;
}
```

Данная функция задает шаблон для генерации ключа шифрования SessionKeyTemplate, вызывает функцию C_GenerateKey, которая возвращает идентификатор созданного временного ключа.

Для того, чтобы произвести шифрование данных используйте функции C_EncryptInit и C_EncryptUpdate. Для расшифровывания

данных используйте функции C_DecryptInit и C_DecryptUpdate. В качестве механизма используйте алгоритм шифрования CKM_DES3_CBC, переменная ckm_DES3.

Используя перечисленные функции, выполните задание.

## 4. Задание на лабораторную работу

1. Требуется написать программу, которая:

- генерирует сессионный ключ шифрования;
- считывает данные из файла и производит их шифрование, выводит зашифрованные данные в консоль;
- расшифровывает данные и выводит их на консоль.

2. Для проверки используйте текстовый файл с содержимым, включающим номер группы и ФИО студента.
3. Необходимо, чтобы программа выводила исходный текст после расшифровывания.
4. Учитывайте, что необходимо использовать буфер длиной 128 байт и производить шифрование данных кусками по 128 байт. Убедитесь, что программа работает с файлами больше 128 байт.

## 5. Контрольные вопросы

1. Опишите назначение механизмов eToken.
2. Назовите и опишите основные функции eToken API для работы с криптографией.
3. Опишите каким образом производится генерация ключей шифрования.
4. Каким образом задается, что создаваемый ключ шифрования будет сессионным и не будет сохранен на токене?
5. Каким образом задается алгоритм шифрования при использовании функций шифрования eToken API?

# ЛАБОРАТОРНАЯ РАБОТА №5
# Защита автоматизированных систем от вредоносного программного обеспечения

## 1. Цель работы

Целью работы является практическое изучение принципов работы вредоносных программ.

## 2. Краткие теоретические сведения

Вредоносная программа — любое программное обеспечение, предназначенное для получения несанкционированного доступа к вычислительным ресурсам самой ЭВМ или к информации, хранимой на ЭВМ, с целью несанкционированного использования ресурсов ЭВМ или причинения вреда (нанесения ущерба) владельцу информации, и/или владельцу ЭВМ, и/или владельцу сети ЭВМ, путём копирования, искажения, удаления или подмены информации.

Все вредоносные программы в соответствии со способами распространения и вредоносной нагрузкой можно разделить на четыре основные типа:

– компьютерные вирусы;

– черви;

– трояны;

– другие программы.

Компьютерный вирус - это программа, способная создавать свои дубликаты (не обязательно совпадающие с оригиналом) и внедрять их в вычислительные сети и/или файлы, системные области компьютера и прочие выполняемые объекты. При этом дубликаты сохраняют способность к дальнейшему распространению.

Условно жизненный цикл любого компьютерного вируса можно разделить на пять стадий:

– Проникновение на компьютер.

– Активация

– Поиск объектов для заражения

– Подготовка копий

– Внедрение копий

Червь - это вредоносная программа, распространяющаяся по сетевым каналам и способная к самостоятельному преодолению систем защиты компьютерных сетей, а также к созданию и дальнейшему

распространению своих копий, не обязательно совпадающих с оригиналом.

Жизненный цикл червей состоит из таких стадий:

– Проникновение в систему (через сеть)

– Активация (самостоятельно или пользователем)

– Поиск объектов для заражения

– Подготовка копий

– Распространение копий

Троян - программа, основной целью которой является вредоносное воздействие по отношению к компьютерной системе.

Следовательно, жизненный цикл троянов состоит всего из трех стадий:

– Проникновение в систему.

– Активация.

– Выполнение вредоносных действий

Признаки заражения компьютера:

– автоматическое открытие окон с незнакомым содержимым при запуске компьютера;

– блокировка доступа к официальным сайтам антивирусных компаний, или же к сайтам, оказывающим услуги по «лечению» компьютеров от вредоносных программ;

– появление новых неизвестных процессов в выводе диспетчера задач (например, окне «Процессы» диспетчера задач Windows);

– появление в ветках реестра, отвечающих за автозапуск, новых записей;

– запрет на изменение настроек компьютера в учётной записи администратора;

– невозможность запустить исполняемый файл (выдаётся сообщение об ошибке);

– появление всплывающих окон или системных сообщений с непривычным текстом, в том числе содержащих неизвестные веб-адреса и названия;

– перезапуск компьютера во время старта какой-либо программы;

– случайное и/или беспорядочное отключение компьютера;

– случайное аварийное завершение программ;

– снижение производительности при достаточном объёме памяти, вплоть до «зависаний» вкупе с аномальным перегреванием системного блока;

– случайное появление BSoD при запуске компьютера;

– появление неизвестных файлов и директорий в проводнике файловой системы ОС, которые обычно выдают ошибку удаления;

– шифрование или повреждение пользовательских файлов;

– неизвестные изменения в содержимом системных файлов при открытии их в текстовом редакторе;

– быстрая утечка памяти на жёстком диске.

Способы защиты от вредоносных программ:

– использовать операционные системы, не дающие изменять важные файлы без ведома пользователя;

– своевременно устанавливать обновления;

– помимо антивирусных продуктов, использующих сигнатурные методы поиска вредоносных программ, использовать программное обеспечение, обеспечивающее проактивную защиту от угроз (необходимость использования проактивной защиты обуславливается тем, что сигнатурный антивирус не замечает новые угрозы, ещё не внесенные в антивирусные базы).

– постоянно работать на персональном компьютере исключительно под правами пользователя, а не администратора, что не позволит некоторым вредоносным программам инсталлироваться на персональном компьютере и изменить системные настройки;

– ограничить физический доступ к компьютеру посторонних лиц;

– использовать внешние носители информации только от проверенных источников на рабочем компьютере;

– не открывать компьютерные файлы, полученные от ненадёжных источников, на рабочем компьютере;

– использовать межсетевой экран (аппаратный или программный), контролирующий выход в сеть Интернет с персонального компьютера на основании политик, которые устанавливает сам пользователь;

– использовать второй компьютер (не для работы) для запуска программ из малонадежных источников, на котором нет ценной информации, представляющей интерес для третьих лиц и злоумышленников;

– делать резервное копирование важной информации на внешние носители и отключать их от компьютера;

– хранить важную информацию, которая может представлять интерес для третьих лиц и злоумышленников, в зашифрованных архивах.

## 3. Требования для выполнения работы

Для начала работы необходимо иметь установленную систему виртуализации VMware Player не ниже 12 версии или VMware Workstation не ниже 12 версии.

Необходимыми условиями для работы виртуальной машиной Windows XP SP3 являются:

– в настройках виртуальной машины выбран тип сетевого адаптера «Только для узла» («Host-only»);

– объем ОЗУ для нормального функционирования виртуальной машины - 256 МБ (при необходимости можно увеличить объем).

## 4. Ход работы

### 4.1. Знакомство с инструментом Process Monitor

После старта программы Process Monitor, выводится окно с фильтрами для исключения из процесса наблюдения событий стандартной активности системы и самого монитора (рис.1).

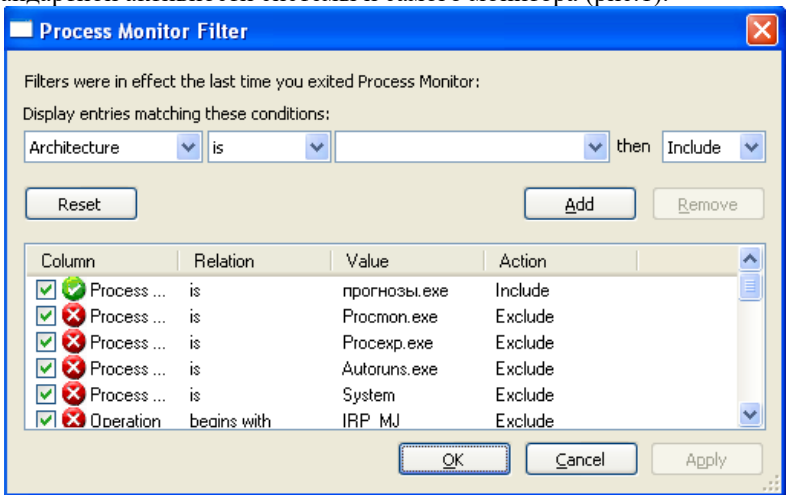



Рисунок 1 – Окно с фильтрами программы Process Monitor

После запуска исполняемого файла procmon.exe начинается сбор, обработка и вывод данных об отслеживаемых событиях в основном окне программы (рис.2).

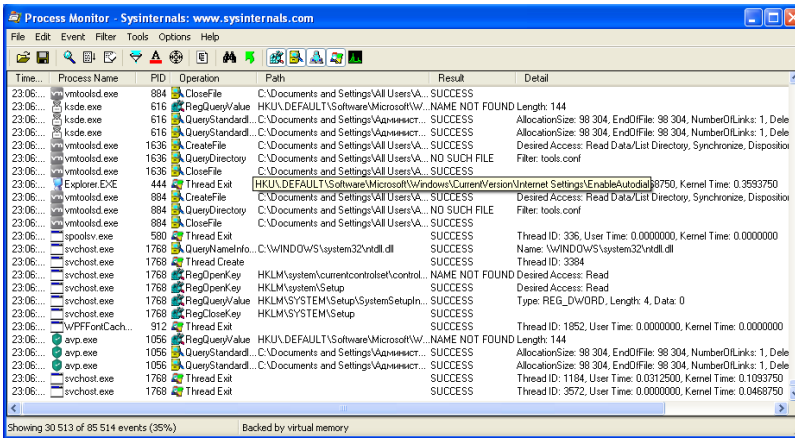



Рисунок 2 – Основное окно программы Process Monitor

Интерфейс программы состоит из 3-х частей - строка меню, панель инструментов и область вывода данных в виде списка. Программа перехватывает отслеживаемые события, связанные с активностью процессов и выдает данные в соответствии с заданными критериями фильтрации и пользовательскими настройками отображаемых колонок. Для остановки мониторинга нужно щелкнуть мышкой по кнопке с лупой на панели инструментов, так, чтобы ее изображение стало перечеркнутым красной линией. Повторный щелчок вернет режим перехвата.

Каждому событию, перехваченному программой Process Monitor, соответствует одна строка в окне вывода данных. Двойной щелчок на отдельной строке вызовет окно просмотра свойств события. Порядок следования строк соответствует последовательности выполнения операций. Информация в окне вывода данных разделена на несколько столбцов, состав которых можно выбрать с помощью контекстного меню, вызываемого правой кнопкой мышки на поле описания колонок или через главное меню (рис.3).

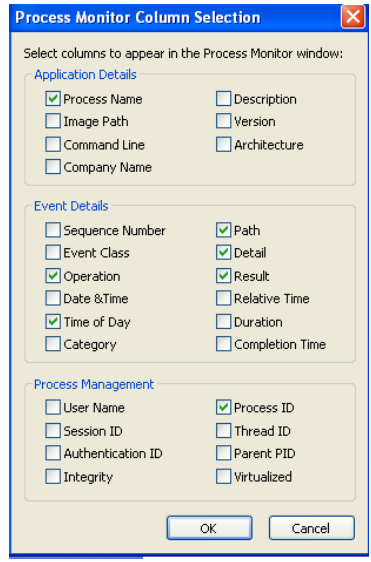


Рисунок 3 – Окно вывода данных

Возможен вывод колонок, разбитых на 3 категории:

– Application Details - сведения о процессе;

– Event Details - сведения о событии;

– Process Management - данные о родительском процессе, порождаемых потоках и контексте учетной записи безопасности исследуемого процесса.

При первом запуске выводятся колонки, наиболее подходящие для быстрого анализа информации и дающие представление о том, какой процесс, какую операцию выполнил, и с каким результатом:

– PID – идентификатор процесса.

– Process Name - имя процесса, вызвавшего событие.

– Operation - выполняемая операция. Значение зависит от типа обращения и представляет собой краткое описание, как , например, открытие ключа реестра RegOpenKey или отправка TCP пакета TCP Send

– Path - путь, связанный с используемым ресурсом. Это может быть файл, ключ реестра, данные TCP соединения и т.п.

– Result - результат выполнения запроса:

– END OF FILE - обнаружен признак конца файла (EOF)

– NAME NOT FOUND - файл, каталог или данные реестра не найдены

– NAME COLLISION - была попытка создать новый файл, но файл с таким именем уже существует.

– FILE LOCKED -файл открыт для монопольного доступа.

– SUCCESS - операция выполнена успешно.

– INVALID DEVICE REQUEST - неверный запрос к устройству.

– FAST I/O DISALLOWED - операция ввода/вывода с использованием устаревшего запроса к драйверу запрещена (интерфейс "fast I/O" в большинстве современных драйверов не поддерживается и заменен на интерфейс IRP - I/o Request Packet - пакет запроса на ввод/вывод).

– Detail - дополнительная информация о событии, описывающая тип запроса, права доступа, свойства файла или каталога, тип данных, значение ключа реестра и т.п.

Панель инструментов позволяет быстро выполнить наиболее необходимые и часто используемые в практической работе с программой, действия (рис.4).

Рисунок 4 – Панель инструментов

 - Open - открыть ранее сохраненные данные мониторинга. Комбинация клавиш Ctrl+O.

 - Save - сохранить текущие данные мониторинга в файл. Комбинация клавиш Ctrl+S.

 - Capture - Включение / выключение режима перехвата событий. Комбинация клавиш Ctrl+E.

 - Autoscroll - Включение / выключение режима автоматической прокрутки экрана данных. Комбинация клавиш Ctrl+E.

 - Clear - очистить текущие данные перехвата событий активности процессов. Комбинация клавиш Ctrl+X.

 - Filter - вызвать окно настройки фильтров. Комбинация клавиш Ctrl+L.

- Highlight - вызвать окно настройки подсвечиваемых событий. Комбинация клавиш Ctrl+H.

- Include Process From Window - Позволяет включить в мониторинг процесс, связанный с определенным окном.

- Show Process Tree - отобразить окно с деревом процессов (рис.5).

Информация отображается в виде иерархической структуры, отображающей зависимости между родительскими и порожденными процессами. Процессы, имеющие одного и того же родителя, отображаются в порядке, соответствующем времени запуска.

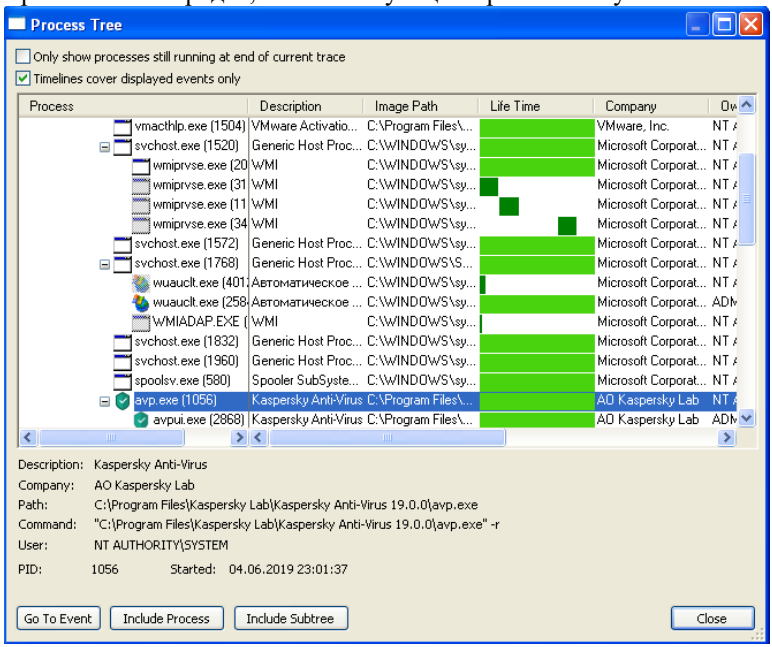



Рисунок 5 – Окно с деревом процессов

При отображении дерева процессов для каждого из них выводится информация с именем, описанием, путем исполняемого файла, владельцем, командной строкой запуска и временем старта. При установке указателя мыши на строку конкретного процесса, в нижней части окна будет выведена подробная информация о нем.

 Find - стандартный диалог поиска строки Windows. Комбинация клавиш Ctrl+F.

 Jump To Object - возможность быстрого перехода к исследуемому объекту - ключу или разделу в редакторе реестра, папке или файлу в проводнике Windows. Комбинация клавиш Ctrl+J.

Следующая группа кнопок панели инструментов задает тип отображаемых событий (класс событий). Если кнопка "утоплена" - события данного типа будут отображаться:

 Show Registry Activity - отображать информацию об обращениях к реестру Windows.

 Show File System Activity - отображать информацию об обращениях к файловой системе Windows.

 Show Network Activity - отображать информацию о сетевой активности процессов.

 Show Process and Thread Activity - отображать информацию об активности процессов, связанной с загрузкой библиотек, созданием и завершением других процессов или потоков.

 Show Profiling Events - Отображать события класса Profiling. Применяется для определения степени использования центрального процессора (CPU) отдельными процессами и их компонентами.

### 4.2. Создание фильтров инструмента Process Monitor

Для начала необходимо знать имя исследуемого процесса, в нашем примере это calc.exe. Используя контекстное меню программы (Filter) или сочетание клавиш (Ctrl+L) открываем Process Monitor Filter (рис.6).

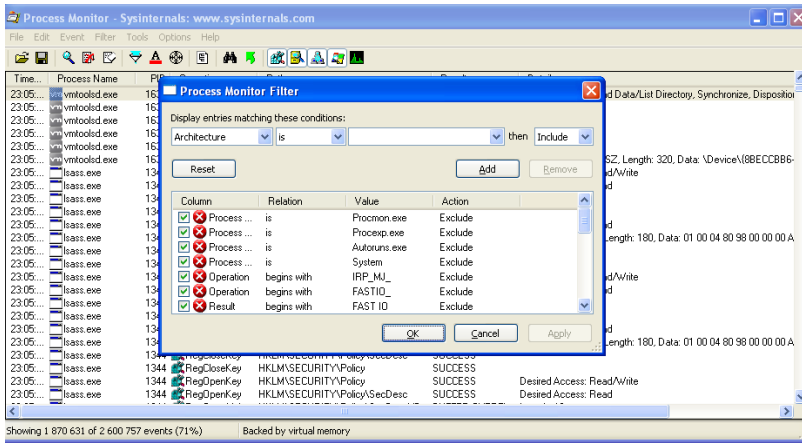



Рисунок 6 - Process Monitor Filter

По умолчанию в программе уже задействован ряд фильтров, исключающих из общего списка различного рода системные процессы, в которых нет необходимости для анализа априори.

В примере мы создадим свой собственный фильтр (по имени процесса) с целью найти и отследить только то, что касается программы калькулятор (файлы, записи в системном реестре, сетевая активность и т.д.).

В первом поле выбираем «Process Name» (по умолчанию там «Architecture») (рис.7).

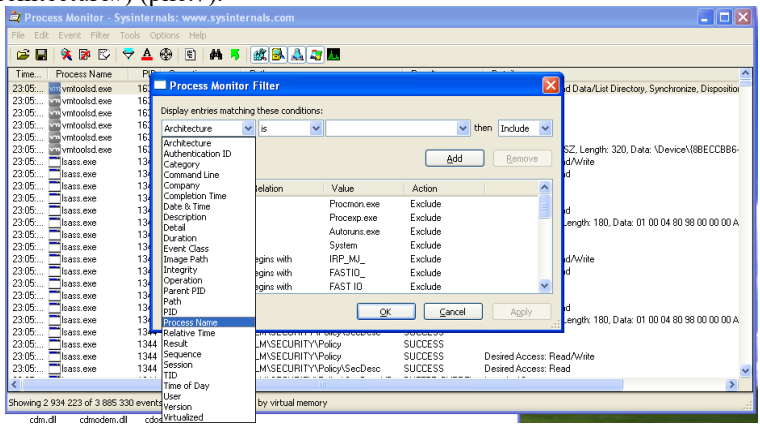



Рисунок 7 – Выбор колонки Process Name

Во втором поле выбираем «is», т.е. мы включаем значение в поиск, в третьем поле выбираем сам процесс «calc.exe», при условии, что программа уже запущена (если мы ходим отследить, что делает та или иная программа до запуска, то название процесса в это поле необходимо вводить вручную) (рис.8).

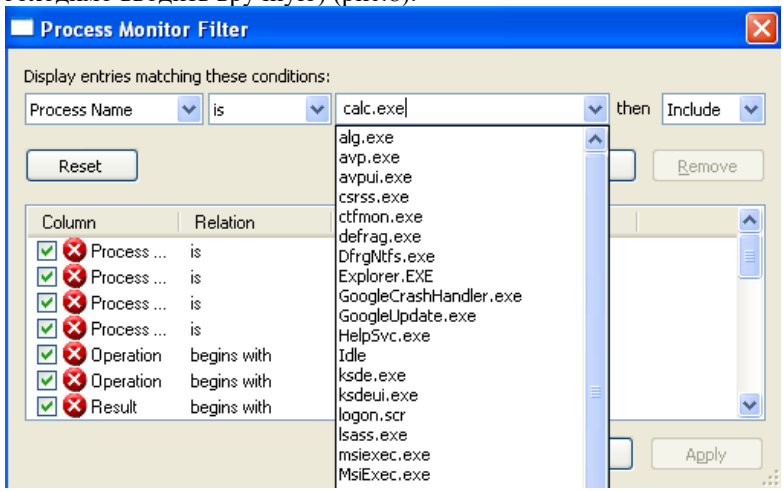



Рисунок 8 – Заполнение полей для создания фильтра

В четвёртом поле оставляем значение по умолчанию «Include» (это позволит отображать в общем потоке, только интересующий нас процесс). Для добавления фильтра нажимаем «Add» (рис.9).

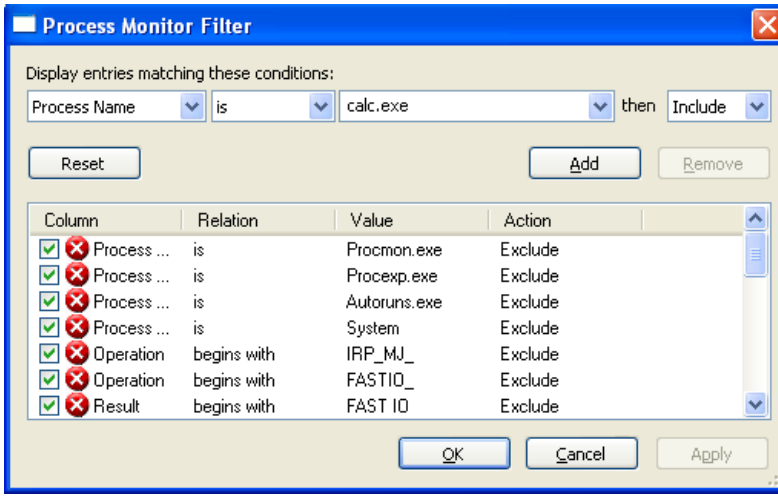



Рисунок 9 – Добавление фильтра

Нажимаем «Apply» и «OK» (рис.10).

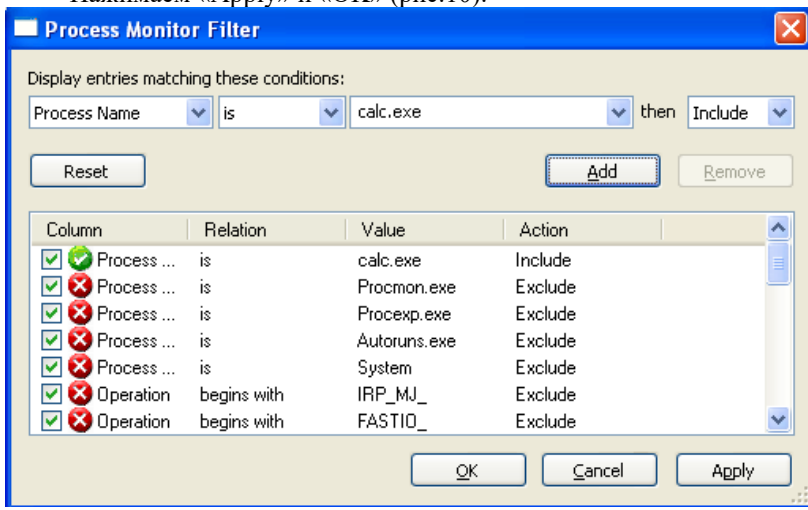



Рисунок 10 – Применение созданного фильтра

После нажатия на кнопку «OK» появится детальная информация по процессу «calc.exe»: ID процесса, потока, обращения к файлам, системному реестру Windows и т.д (рис.11).

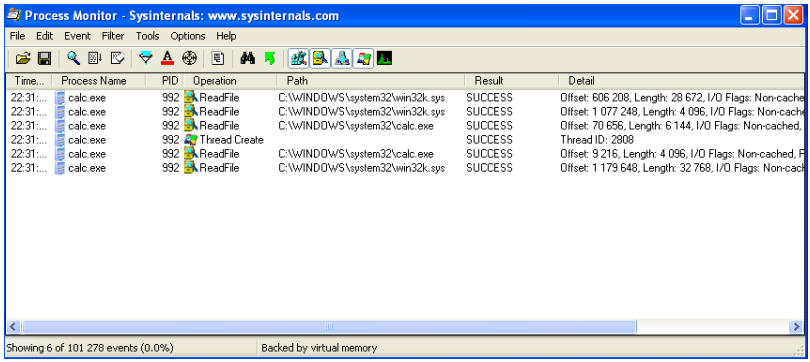



Рисунок 11 – Информация о процессе

### 4.3. Анализ системы на наличие вредоносных программ

На рабочем столе находится архив «14_44_100_vir» (рис.12).


Рисунок 12 – Архив с вредоносными программами

Данный архив содержит 100 вредоносных программ в различных форматах (например, exe, html, doc и т.д.) (рис.13).

Вредоносные программы разделены на несколько видов: трояны, вирусы, черви, кейлогеры, меломаны.

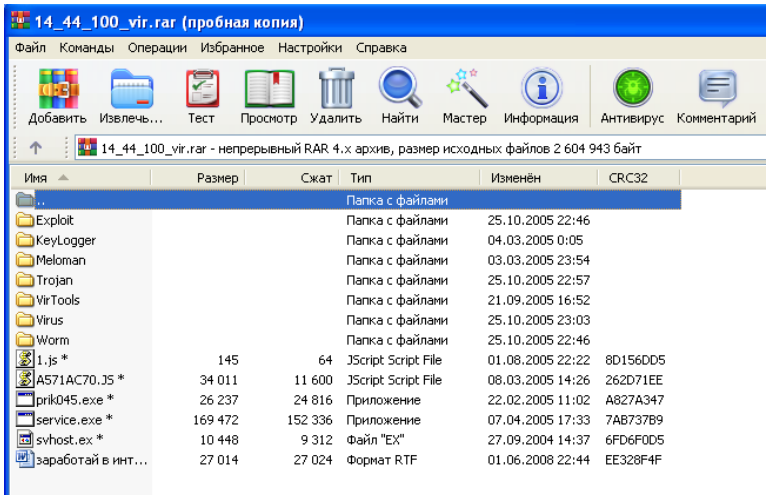



Рисунок 13 – Перечень вредоносных программ

Для того чтобы разархивировать вредоносные программы целиком или же по отдельности, необходимо ввести пароль «virus» (рис.14).

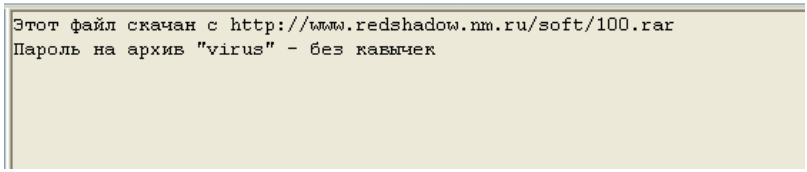
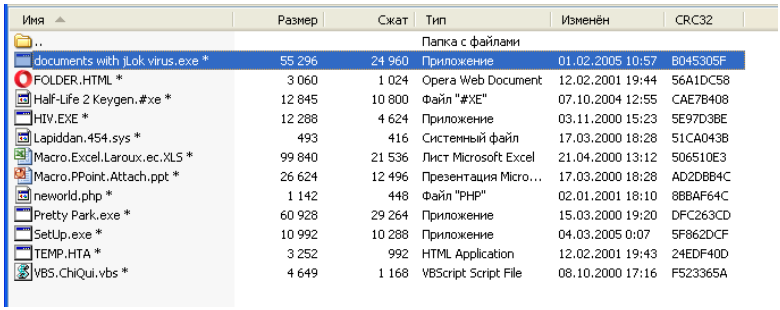



Рисунок 14 – Пароль для разархивации вредоносных программ

Для демонстрации полного разбора (т.е. определение типа, особенностей, функций и уничтожения) вредоносной программы будут использоваться следующие вредоносные программы:

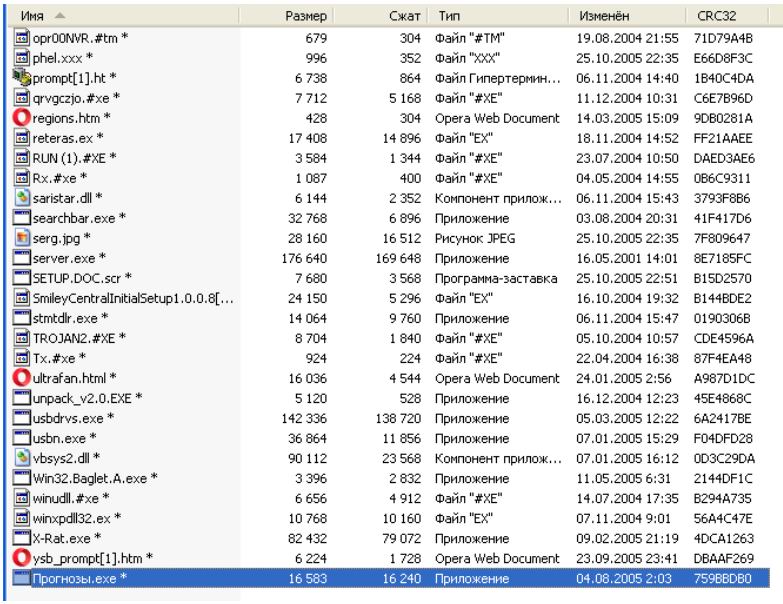
– documents with jLok virus (рис.15);

– Прогнозы (рис.16).



| Имя | Размер | Сжат | Тип | Изменён | CRC32 |
|---|---|---|---|---|---|
| .. | | | Папка с файлами | | |
| documents with jLok virus.exe * | 55 296 | 24 960 | Приложение | 01.02.2005 10:57 | B045305F |
| FOLDER.HTML * | 3 060 | 1 024 | Opera Web Document | 12.02.2001 19:44 | 56A1DC58 |
| Half-Life 2 Keygen.#xe * | 12 845 | 10 800 | Файл "#XE" | 07.10.2004 12:55 | CAE7B408 |
| HIV.EXE * | 12 288 | 4 624 | Приложение | 03.11.2000 15:23 | 5E97D3BE |
| Lapiddan.454.sys * | 493 | 416 | Системный файл | 17.03.2000 18:28 | 51CA043B |
| Macro.Excel.Laroux.ec.XLS * | 99 840 | 21 536 | Лист Microsoft Excel | 21.04.2000 13:12 | 506510E3 |
| Macro.PPoint.Attach.ppt * | 26 624 | 12 496 | Презентация Micro... | 17.03.2000 18:28 | AD2DBB4C |
| neworld.php * | 1 142 | 448 | Файл "PHP" | 02.01.2001 18:10 | 8BBAF64C |
| Pretty Park.exe * | 60 928 | 29 264 | Приложение | 15.03.2000 19:20 | DFC263CD |
| SetUp.exe * | 10 992 | 10 288 | Приложение | 04.03.2005 0:07 | 5F862DCF |
| TEMP.HTA * | 3 252 | 992 | HTML Application | 12.02.2001 19:43 | 24EDF40D |
| VBS.ChiQui.vbs * | 4 649 | 1 168 | VBScript Script File | 08.10.2000 17:16 | F523365A |

Рисунок 15 – Вредоносная программа 1



| Имя | Размер | Сжат | Тип | Изменён | CRC32 |
|---|---|---|---|---|---|
| opr00NVR..#tm * | 679 | 304 | Файл "#TM" | 19.08.2004 21:55 | 71D79A4B |
| phel.xxx * | 996 | 352 | Файл "XXX" | 25.10.2005 22:35 | E66D8F3C |
| prompt[1].ht * | 6 738 | 864 | Файл Гипертермин... | 06.11.2004 14:40 | 1B40C4DA |
| qrvgczjo.#xe * | 7 712 | 5 168 | Файл "#XE" | 11.12.2004 10:31 | C6E7B96D |
| regions.htm * | 428 | 304 | Opera Web Document | 14.03.2005 15:09 | 9DB0281A |
| reteras.ex * | 17 408 | 14 896 | Файл "EX" | 18.11.2004 14:52 | FF21AAEE |
| RUN (1).#XE * | 3 584 | 1 344 | Файл "#XE" | 23.07.2004 10:50 | DAED3AE6 |
| Rx.#xe * | 1 087 | 400 | Файл "#XE" | 04.05.2004 14:55 | 0B6C9311 |
| saristar.dll * | 6 144 | 2 352 | Компонент прилож... | 06.11.2004 15:43 | 3793F8B6 |
| searchbar.exe * | 32 768 | 6 896 | Приложение | 03.08.2004 20:31 | 41F417D6 |
| serg.jpg * | 28 160 | 16 512 | Рисунок JPEG | 25.10.2005 22:35 | 7F809647 |
| server.exe * | 176 640 | 169 648 | Приложение | 16.05.2001 14:01 | 8E7185FC |
| SETUP.DOC.scr * | 7 680 | 3 568 | Программа-заставка | 25.10.2005 22:51 | B15D2570 |
| SmileyCentralInitialSetup1.0.0.8[... | 24 150 | 5 296 | Файл "EX" | 16.10.2004 19:32 | B144BDE2 |
| stmtdlr.exe * | 14 064 | 9 760 | Приложение | 06.11.2004 15:47 | 0190306B |
| TROJAN2.#XE * | 8 704 | 1 840 | Файл "#XE" | 05.10.2004 10:57 | CDE4596A |
| Tx.#xe * | 924 | 224 | Файл "#XE" | 22.04.2004 16:38 | 87F4EA48 |
| ultrafan.html * | 16 036 | 4 544 | Opera Web Document | 24.01.2005 2:56 | A987D1DC |
| unpack_v2.0.EXE * | 5 120 | 528 | Приложение | 16.12.2004 12:23 | 45E4868C |
| usbdrvs.exe * | 142 336 | 138 720 | Приложение | 05.03.2005 12:22 | 6A2417BE |
| usbn.exe * | 36 864 | 11 856 | Приложение | 07.01.2005 15:29 | F04DFD28 |
| vbsys2.dll * | 90 112 | 23 568 | Компонент прилож... | 07.01.2005 16:12 | 0D3C29DA |
| Win32.Baglet.A.exe * | 3 396 | 2 832 | Приложение | 11.05.2005 6:31 | 2144DF1C |
| winudll.#xe * | 6 656 | 4 912 | Файл "#XE" | 14.07.2004 17:35 | B294A735 |
| winxpdll32.ex * | 10 768 | 10 160 | Файл "EX" | 07.11.2004 9:01 | 56A4C47E |
| X-Rat.exe * | 82 432 | 79 072 | Приложение | 09.02.2005 21:19 | 4DCA1263 |
| ysb_prompt[1].htm * | 6 224 | 1 728 | Opera Web Document | 23.09.2005 23:41 | DBAAF269 |
| Прогнозы.exe * | 16 583 | 16 240 | Приложение | 04.08.2005 2:03 | 759BBDB0 |

Рисунок 16 – Вредоносная программа 2

После разархивирования файлов из вирусной базы, запустим вредоносную программу 1 и для просмотра информации о процессе создадим и применим фильтр с помощью инструмента Process Monitor (рис.17).

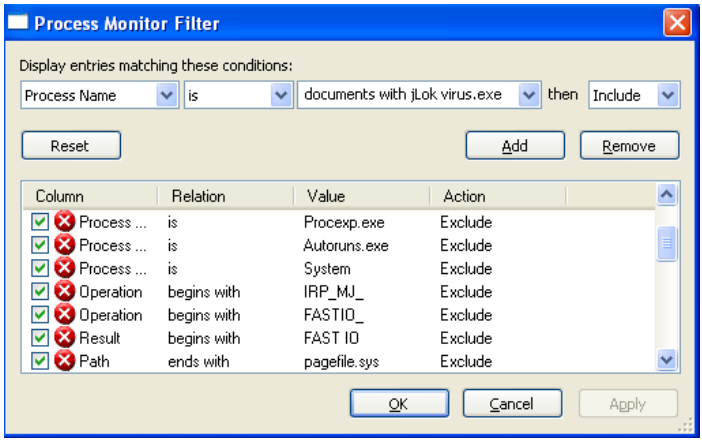



Рисунок 17 – Создание фильтра для вредоносной программы 1

В результате применения созданного фильтра просмотрим и проанализируем детальную информацию о процессе (рис.18).

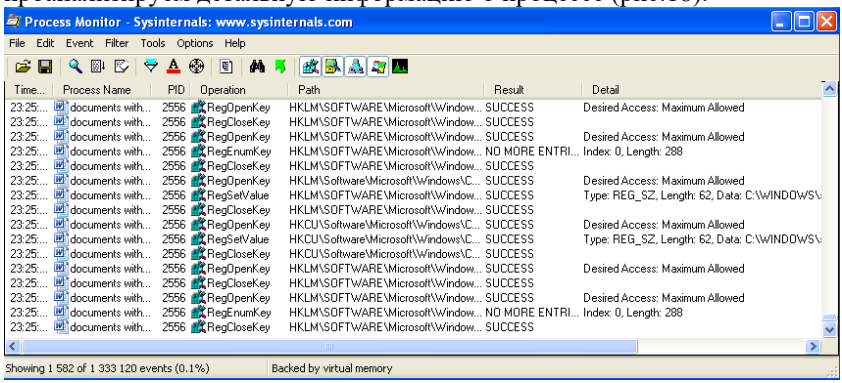



Рисунок 18 – Детальная информация о процессе

Вредоносная программа 1 пытается достучаться до следующих ключей реестра:

– HKLM

Раздел содержит информацию об установленном программном обеспечении, его настройках, драйверах. Здесь же – информация, относящаяся к операционной системе и оборудованию, например, тип шины компьютера, общий объем доступной памяти, список загруженных в данный момент времени драйверов устройств, а также

сведения о загрузке Windows. Данная ветвь включает наибольшее количество информации в системном реестре и нередко используется для тонкой настройки аппаратной конфигурации компьютера. Хранящиеся в этой ветви данные справедливы для всех профилей, зарегистрированных в системе пользователей.

– HKCU

Эта ветвь реестра хранит настройки персональной оболочки пользователя, совершающего вход в операционную систему (меню «Пуск», рабочий стол и т. д.). В ее подразделах находится информация о переменных окружения, группах программ данного пользователя, настройках Рабочего стола, цветах экрана, сетевых соединениях, принтерах и дополнительных настройках приложений. Эта информация берется из подраздела Security ID (SID) ветви HKEY_USERS для текущего пользователя. Фактически, в данной ветви собраны все сведения, относящиеся к профилю пользователя, работающего с Windows в настоящий момент.

Далее вредоносная программа проходится по файловой системе.

После просмотра информации о процессе выведем дерево процессов (рис.19).

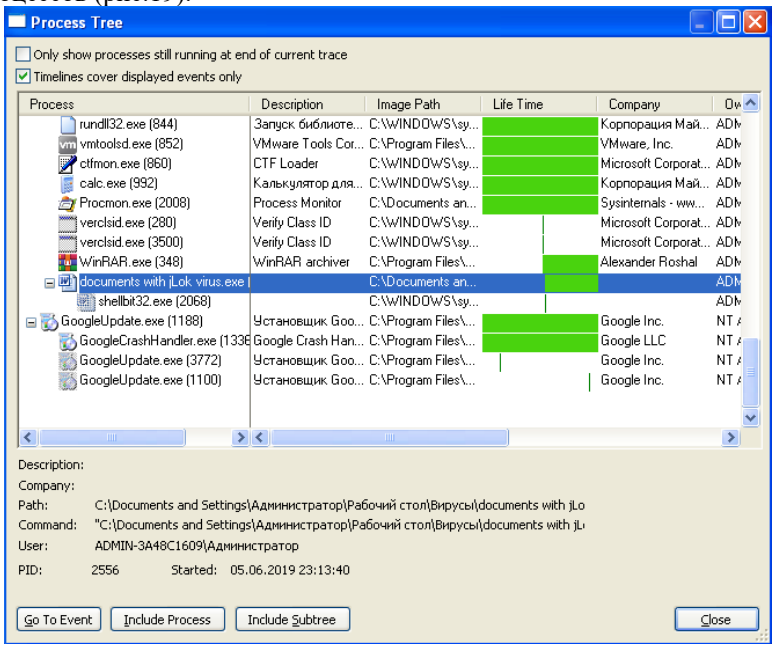



Рисунок 19 – Окно с деревом процессов

В результате мы видим, что от исследуемого процесса запустился дочерний процесс.

После изучения процесса можно сделать вывод, что исследуемый процесс ведет подозрительную активность в системе.

Далее проведем исследование вредоносной программы 2. Также разархивируем ее (рис.20) и запустим. После чего создадим и применим фильтр для процесса (рис.21).

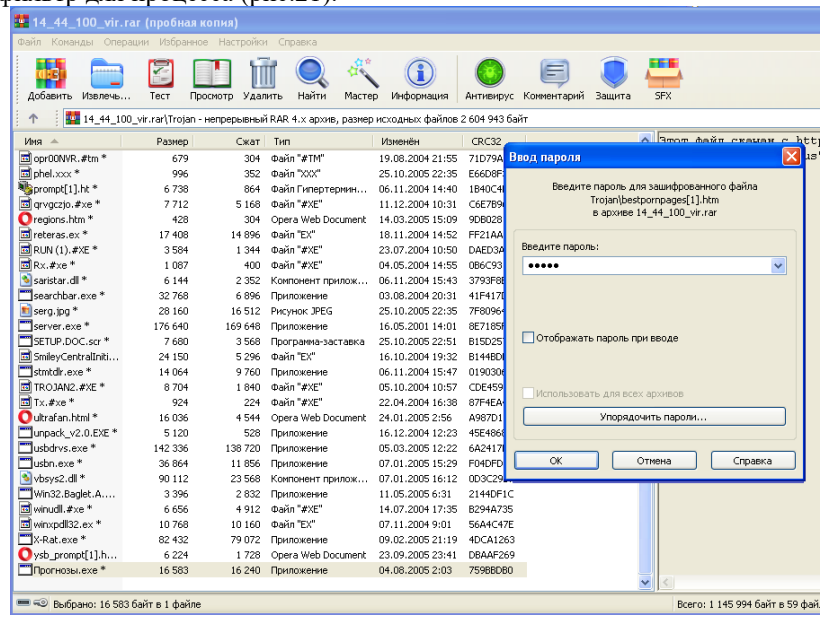



Рисунок 20 – Разархивирование вредоносной программы 2

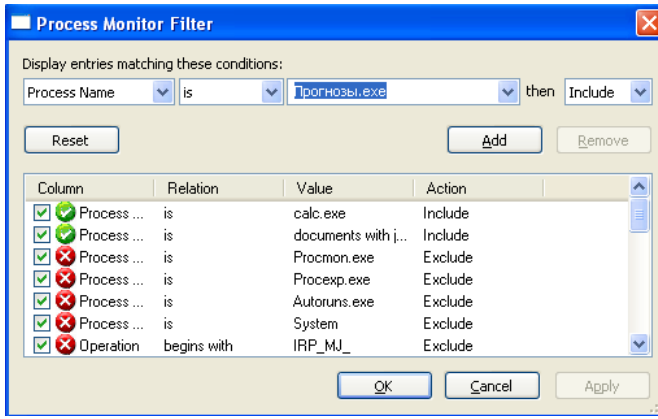



Рисунок 21 – Создание фильтра для исследуемого процесса

Далее рассмотрим детальную информацию о процессе в результате применения фильтра (рис.22).

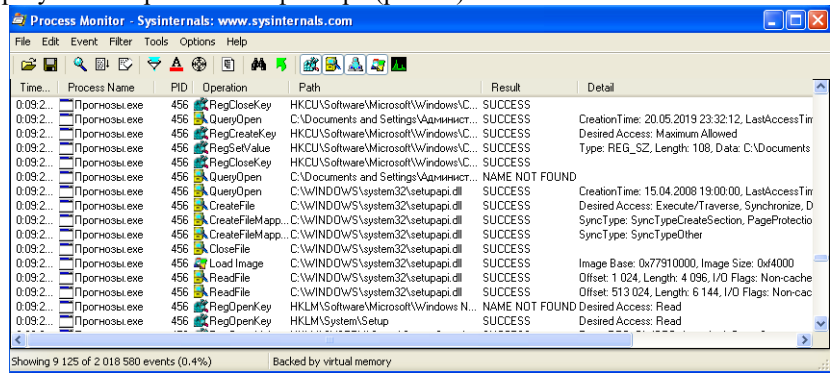



Рисунок 22 – Детальная информация о вредоносной программе 2

Вредоносная программа 2 проходится по тем же ключам реестра, что и вредоносная программа 1, за исключением – HKCR.

Этот раздел включает в себя ряд подразделов, в которых содержатся сведения о расширениях всех зарегистрированных в системе типов файлов и данные о COM-серверах, зарегистрированных на компьютере. Данные этого раздела нужны при открытии файлов по двойному щелчку мыши или операций drag-and-drop. Кроме того, раздел HKEY_CLASSES_ROOT предоставляет объединенные данные программам, написанным под ранние версии Windows.

Далее вредоносная программа заходит в файловую систему.

Посмотрим окно с деревом процессов (рис.23).

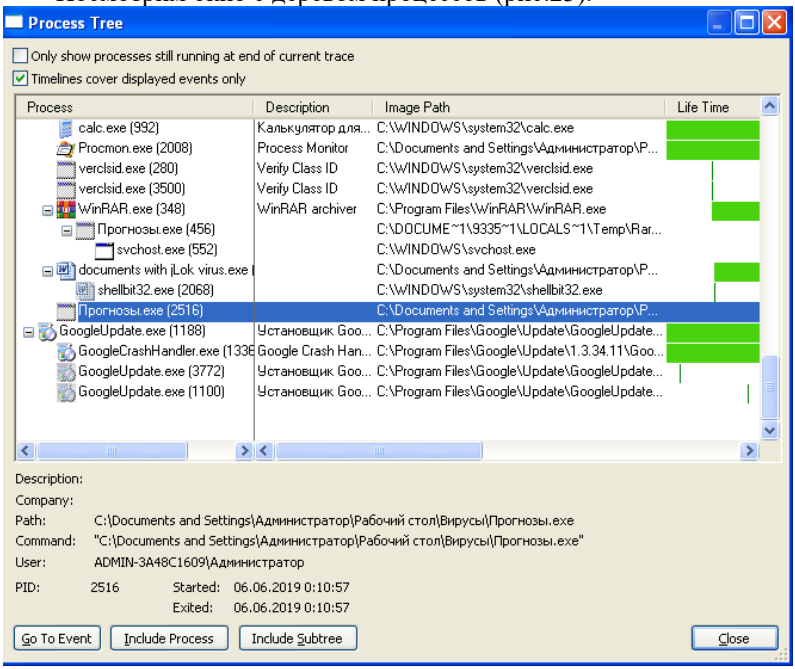



Рисунок 23 – Окно с деревом процессов

Просмотрев данные о исследуемом процессе можно сделать вывод, что данная программа ведет подозрительную активность в системе.

Далее выводы о исследуемых процессах можно подтвердить средством антивирусной защиты.

### 4.5. Лечение системы после ее заражения

После такого как были запущены два вируса необходимо привести систему в порядок. Для этого установим средство антивирусной защиты Kaspersky Anti-Virus.

Перейдем по ссылке https://www.kaspersky.ru/downloads (рис.24).

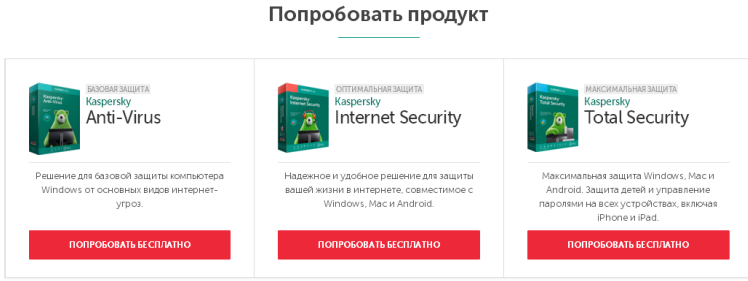



Рисунок 24 – Выбор средства антивирусной защиты

Выберем первое средство и нажмем на кнопку «попробовать бесплатно». Далее запустим скачанный файл (рис.25).

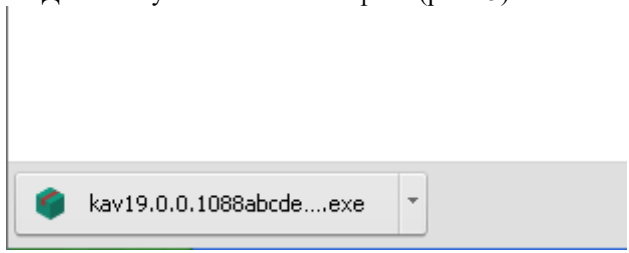



Рисунок 25 – Скачанное средство антивирусной защиты

Появится стартовое окно для установки продукта (рис.26).

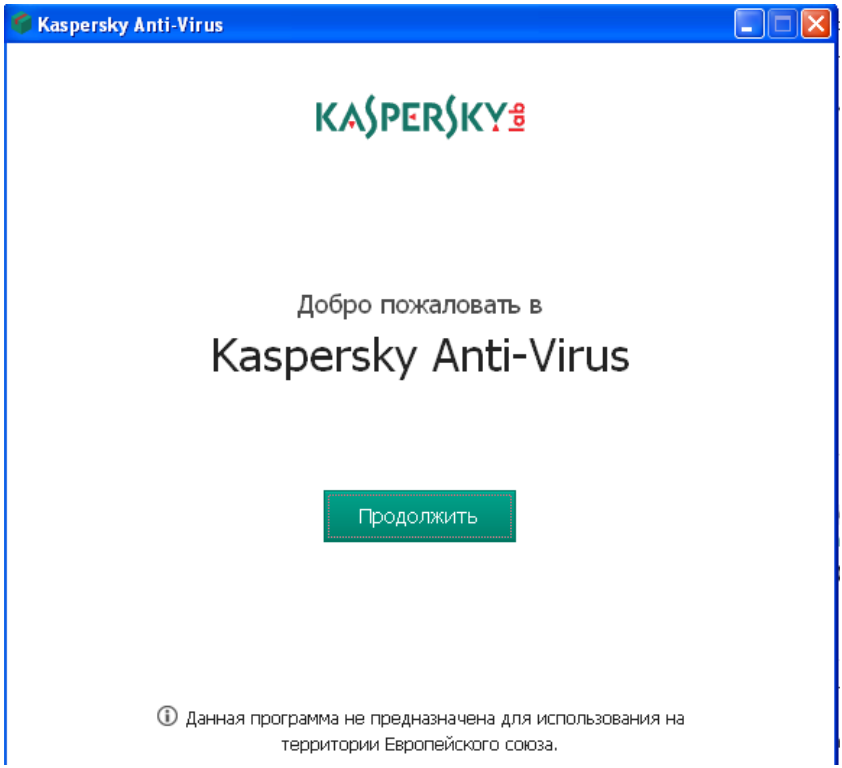



Рисунок 26 – Стартовое окно

Нажмем на кнопку «Продолжить». И далее необходимо согласиться с условиями лицензии (рис.27).

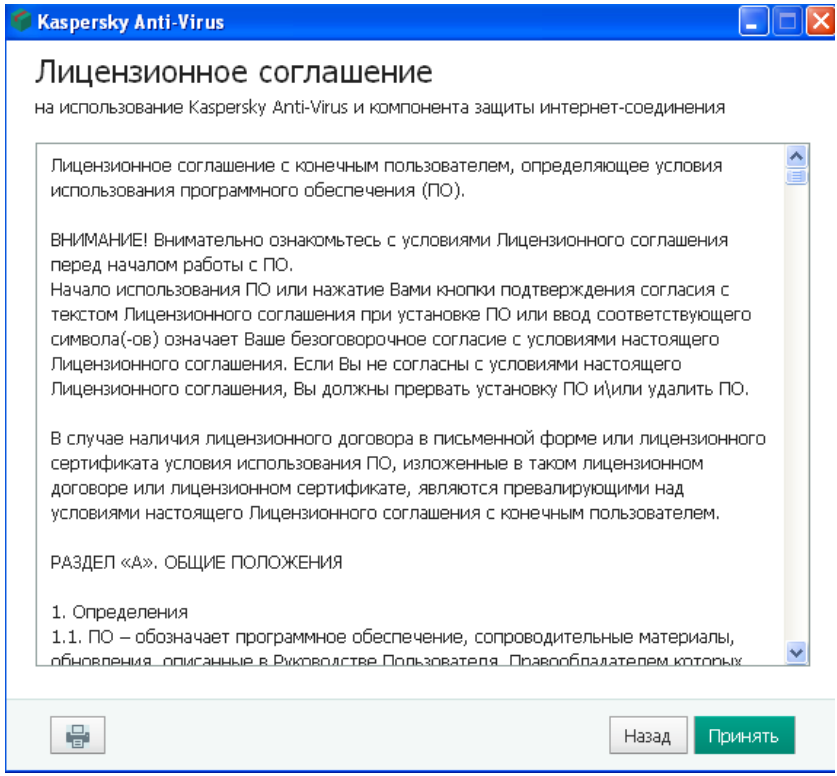



Рисунок 27 – Лицензионное соглашение

После примем положения о компонентах защиты интернет-соединения (рис.28).

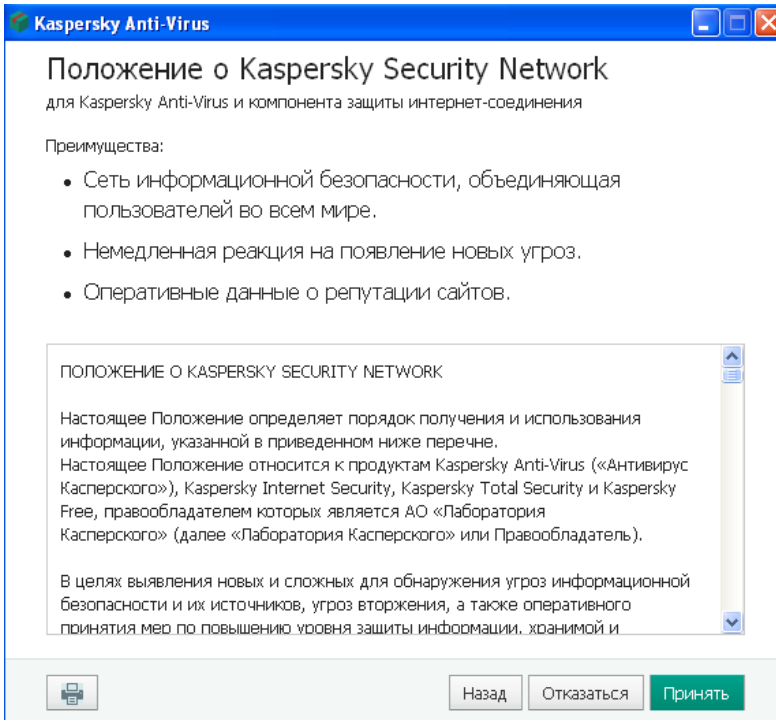


Рисунок 28 – Положение о компонентах защиты Интернет-соединения

После принятия всех положений появится окно для установки программы (рис.29).

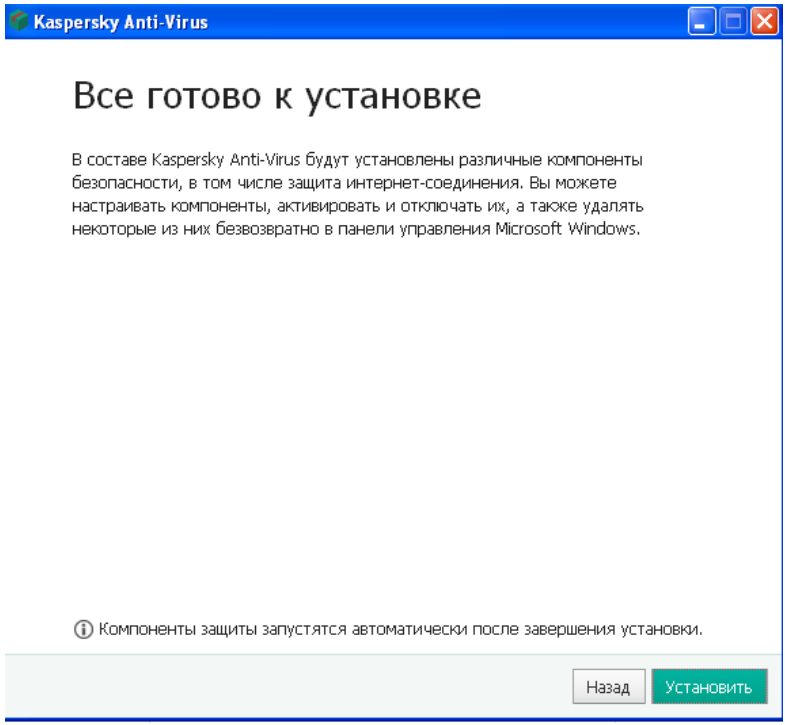



Рисунок 29 – Готовность к установке

Перед запуском программы можно выбрать рекомендуемые параметры (рис.30).

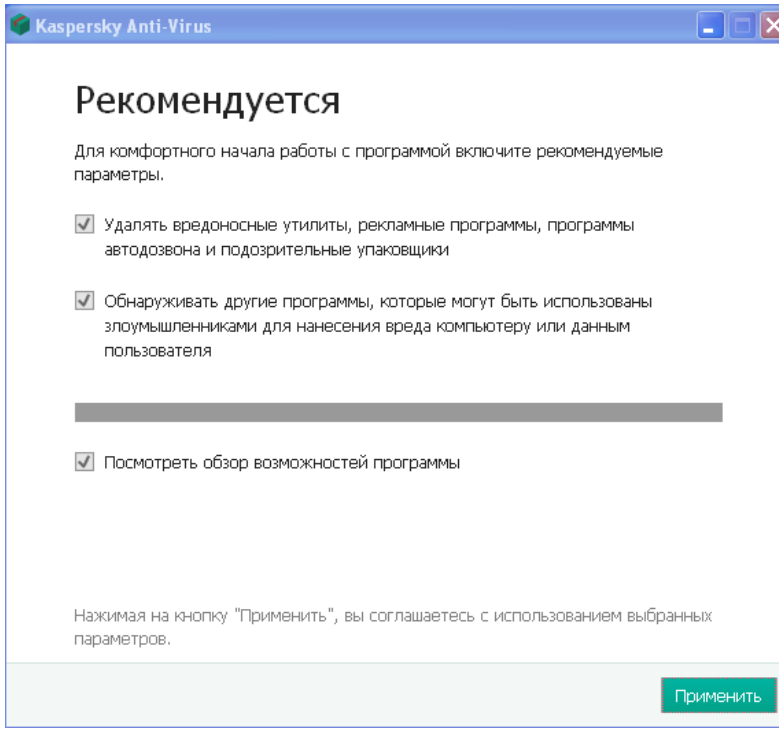



Рисунок 30 – Рекомендуемые параметры

После нажатия на кнопку «Применить» появится окно об успешной установке средства антивирусной защиты (рис.31).

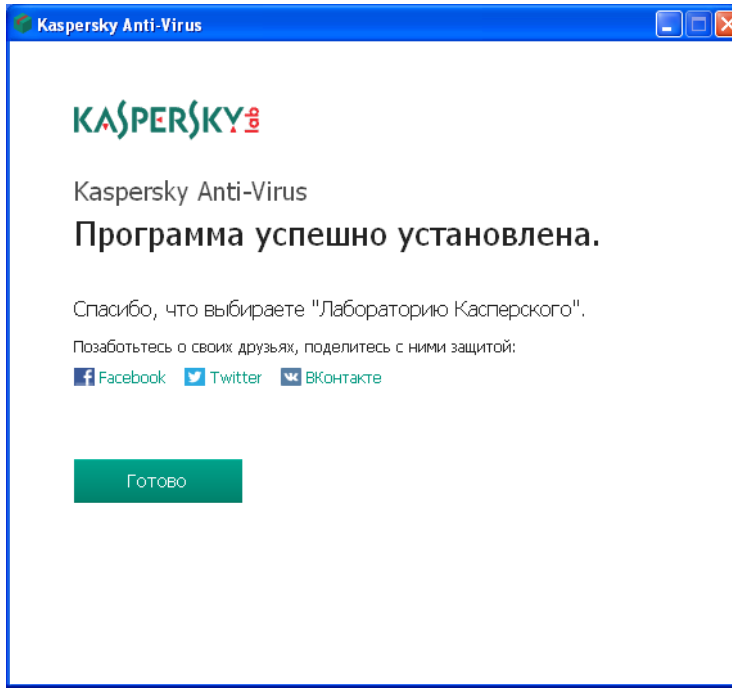



Рисунок 31 – Успешная установка

Сразу после запуска антивирусного средства, началось лечение системы (рис.32).

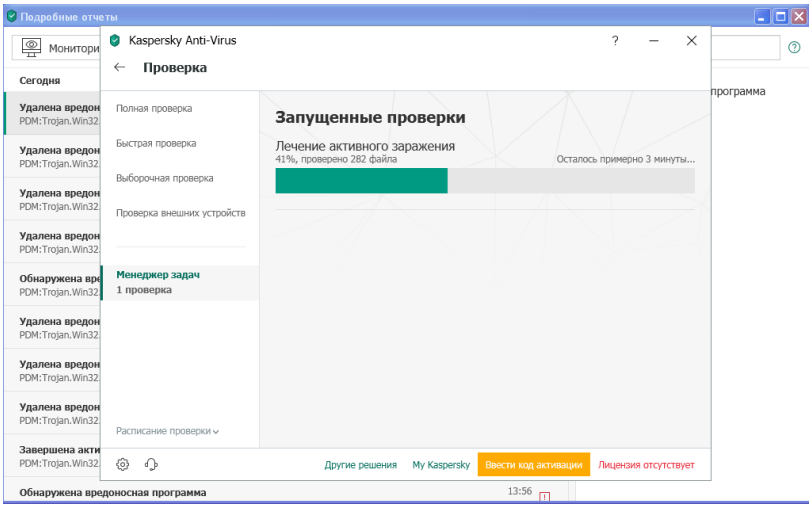



Рисунок 32 – Лечение системы после заражения

## 4. Задание на лабораторную работу

1. Ознакомиться с инструментом Process Monitor. Создайте и примените три фильтра к различным программам.
2. Разархивируйте две вредоносные программы согласно варианту (табл. 1).
3. Запустите разархивированные вредоносные программы, создайте и примените фильтры для данных программ.
4. Проанализируйте детальную информацию, выведенную при помощи фильтров, и сделайте выводы о особенностях вредоносных программ.
5. Проведите лечение системы после ее заражения.
6. Зафиксируйте проделанные пункты в отчете и защитите лабораторную работу у преподавателя.

Таблица 1

Индивидуальное задание

| Вариант | Вредоносное ПО |
|---|---|
| 1 | documents with jLok virus и prik045 |
| 2 | Прогнозы и заработай в интернете без денежных вложений |
| 3 | Marco.Excell.Laroux.ec и Win32.Bagle.CY |
| 4 | Marco.Ppoint. Attach и I-Worm.Blebla.H |

Продолжение табл. 1

| Вариант | Вредоносное ПО |
|---|---|
| 5 | X-Rat и document.txt |
| 6 | eied_s7 и SetUp |
| 7 | Pretty Park и Marco.PPoint.Attach |
| 8 | stuff и Marco.Excell.Laroux.ec |
| 9 | service и HIV |
| 10 | FOLDER и patch |

## 5. Контрольные вопросы

1. Дайте определение вредоносной программе.
2. Какие типы вредоносных программ существуют?
3. Перечислите признаки заражения системы.
4. Перечислите методы преодоления заражения системы.
5. Опишите жизненный цикл вирусов.
6. Опишите жизненный цикл троянов.
7. Опишите жизненный цикл червей.
8. Опишите основные возможности средства антивирусной защиты Kaspersky Anti-Virus.
9. Опишите функционал инструмента мониторинга Process Monitor.
10. Опишите процесс вывода дерева процессов с помощью инструмента мониторинга Process Monitor.

## Литература

1. Рябко, Б. Я. Криптографические методы защиты информации [Электронный ресурс]: учебное пособие / Б. Я. Рябко, А. Н. Фионов. — 2-е изд., стер. — Москва : Горячая линия-Телеком, 2017. — 230 с

2. Бабенко, Людмила Климентьевна. Защита информации с использованием смарт-карт и электронных брелоков. - М. : "Гелиос АРВ" , 2003. - 352 с.