ЛАБОРАТОРНАЯ РАБОТА № 2 «ИЗУЧЕНИЕ СИСТЕМЫ КРИПТОГРАФИЧЕСКОЙ ЗАЩИТЫ В ОПЕРАЦИОННЫХ СИСТЕМАХ MS WINDOWS. ПРОТОКОЛ CRYPTO API 1.0, CRYPTO SPI.»

СОДЕРЖАНИЕ

Цель работы

Научиться работать со стандартным криптографическим интерфейсом ОС Microsoft Windows CryptoApi. Изучить работу криптопровайдеров в ОС Microsoft Windows. Реализовать базовые криптографические функции с использованием криптопровайдера.

Теоретический минимум

Поскольку реализация алгоритмов шифрования в каждом новом проекте достаточно трудоемкий процесс, требующий зачастую не только программной реализации, но и аппаратной, получил распространение подход работы со стандартизированными криптографическими функциями. Для этой цели в ОС Windows используется специальный интерфейс CryptoApi. Данный интерфейс предоставляет единый механизм доступа к уже готовым коиптографическим функциям, реализованным в рамках специальной программной либо программно-аппаратной системы – криптопровайдера.

Криптопровайдер, или Cryptographic Service Provider (кратко - CSP), в ОС Windows – это библиотека криптографических алгоритмов, доступных прикладным программистам посредством интерфейса CryptoAPI (Cryptographic Application Programming Interface). При взаимодействии с CSP приложения вызывают функции CryptoAPI, которые обращаются к библиотекам Crypt32.dll и Advapi32.dll операционной системы. Последняя фильтрует вызовы этих функций и передаёт их далее соответствующим функциям CSP через CryptoSPI (Cryptographic System Program Interface).

CryptoAPI 1.0 это так называемый уровень криптографических примитивов - определенный набор функций реализующих основные криптографические операции — шифрование/расшифрование данных, хеширование данных, генерация и проверка электронной цифровой подписи (ЭЦП) и работу с ключевой информацией.

С точки зрения операционной системы криптопровайдер представляется как библиотека динамической линковки (dll). Набор экспортируемых функций этой библиотеки должен соответствовать интерфейсу криптопровайдера (cryptographic service provider interface — CryptoSPI), который приведен ниже в таблицах.

Таблица 1. Группы функций интерфейс криптопровайдера CryptoSPI

Группа функций	Описание
Инициализация и параметры криптопровайдера	Функции инициализации и параметры. Предназначены для установления сессии прикладного ПО с криптопровайдером и получения (установки) параметров сессии.

Функции генерации и работы с ключами	Функции генерации и работы с ключами предназначены для генерации парных и обработки различных типов ключей, используемых в криптопровайдере.
Функции шифрования/расшифрования данных	Функции шифрования/расшифрования данных предназначены для выполнения операций, шифрования и расшифрования данных.
Функции хэширования и ЭЦП	Функции хэширования и ЭЦП предназначены для вычисления значения хэш-функции и формирования/проверки ЭЦП данных.

Интерфейс криптопровайдера стандартизирован с целью абстрагирования конечного приложения от конкретной реализации криптографических функций. Основной архитектурной особенностью криптографических ядер, разработанноых по принципу криптопровайдера, является то, что прикладное программное обеспечение не имеет непосредственного доступа к ключевой и криптографически-опасной информации. Все операции с сессионными и долговременными закрытыми и симметричными ключами, незавершёнными значениями хэш-функций и т. п. осуществляется через дескрипторы соответствующих объектов. Это позволяет обеспечить корректность операций над этими объектами, так как дескриптор объекта непосредственно не содержит его адреса.
Использующее криптопровайдер прикладное программное обеспечение вызывает «абстрактные» функции CryptoAPI 1.0, которые экспортируются системной библиотекой advapi32.dll. На рисунке (Рисунок 6.3) показана схема вызова функций CryptoAPI 1.0, и последующий вызов криптопровайдера.

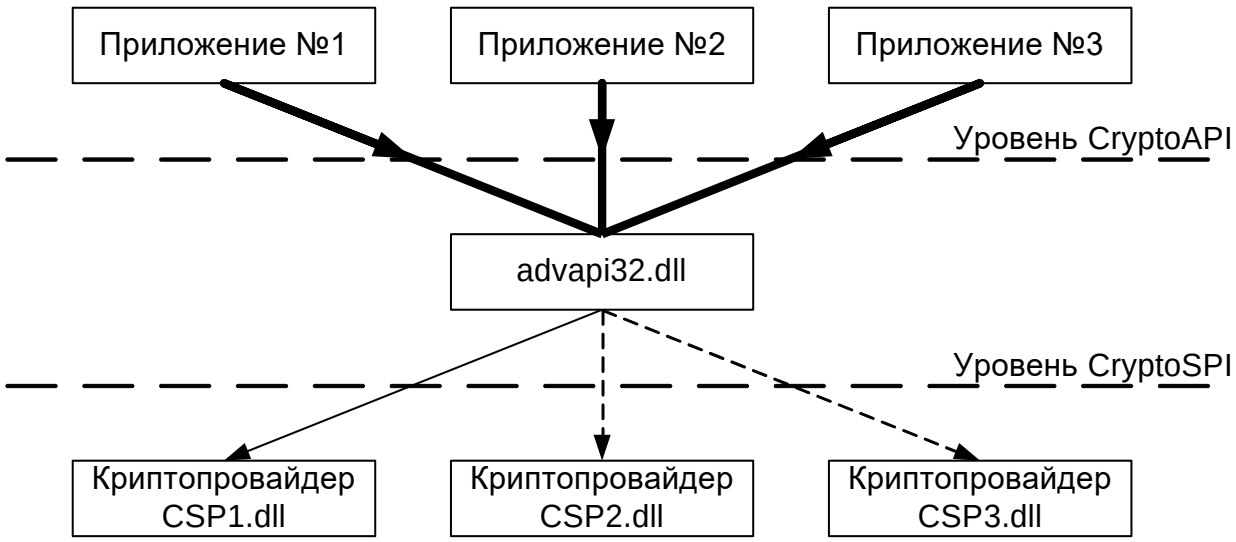

Рисунок 1. Схема вызова криптопровайдера

Функции, экспортируемые библиотекой advapi32.dll по параметрам фактически полностью совпадают с функциями экспортируемыми криптопровайдером. Параметрами, главным образом отличаются функция CryptoAPI 1.0 CryptAcquireContext() и функция CryptoSPI CPAcquireContext().

```
BOOL WINAPI CryptAcquireContext(
 HCRYPTPROV *phProv,
```

```
  LPCTSTR pszContainer,
  LPCTSTR pszProvider,
  DWORD dwProvType,
  DWORD dwFlags
);

BOOL CPAcquireContext(
  HCRYPTPROV* phProv,
  CHAR* pszContainer,
  DWORD dwFlags,
  PVTableProvStruc pVTable
);
```

В параметре pszProvider предается имя криптопровайдера. Имена всех зарегистрированных в системе криптопровайдеров находятся в разделе системного реестра:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography\Defaults\Provider

Имя папок в этом разделе имеет особый смысл. При вызове CryptAcquireContext() в этом разделе ищется папка имя которой совпадает со строкой преданно в параметре pszProvider. В этой папке находится ключ Image Path, в котором содержится путь библиотеки криптопровайдера.

Функция CryptAcquireContext() возвращает дескриптор контекста (сессии) по указателю phProv. При дальнейших вызовах всех Crypt* функций дескриптор контекста будет входным параметром. Дескриптор контекста определяет при вызовах других функций CrytoAPI 1.0 тот криптопровайдер, который был указан при инициализации контекста. Таким образом, вызов функции Crypt* перенаправляется в библиотеку найденную по имени при вызове CryptAcquireContext(), в самом начале работы с CrytoAPI 1.0. Собственно, все вызовы криптографических функций CrytoAPI 1.0 «обрамляются» вызовами двух функций — CryptAcquireContext() и CryptReleaseContext(), — открытие и закрытие сессии работы с криптопровайдером.

CryptoSPI регламентирует состав параметров вызова функций криптопровайдра, а также регламентирует некоторый стандартный (закрепленный) набор флагов и значений параметров вызова. Также регламентируется набор возвращаемых ошибок. Однако разработчикам разрешается добавлять собственные параметры и флаги вызовов. Есть также набор необязательных параметров и флагов, которые разрабатываемый криптопровайдер может или игнорировать или не поддерживать.

Для завершения описания функциональности криптопровайдера ниже приведены таблицы, описывающие внутренний интерфейс криптопровайдера CryptoSPI.

Таблица 2. Функции инициализации и параметров криптопровайдера

Название	Описание
CPAcquireContext()	Используется для создания дескриптора криптопровайдера с именем ключевого контейнера.
CPReleaseContext()	Используется для удаления дескриптора криптопровайдера, созданного функцией CPAcquireContext().

CPGetProvParam()	Производит возвращение параметров криптопровайдера.
CPSetProvParam()	Устанавливает параметры криптопровайдера.

Таблица 3. Функции генерации и работы с ключами

Название	Описание
CPGenKey()	Производит случайные криптографические ключи или парные (секретный/открытый) ключи.
CPDestroyKey()	Удаляет ключ. После удаления ключ (дескриптор ключа) не может использоваться.
CPDeriveKey()	Производит криптографические ключи сессии на основе значения хэш-функции, вычисленной по другим ключам, паролям или любым другим данным пользователя.
CPDuplicateKey()	Создает копию заданного ключа, включая все его переменные, определяющие внутреннее состояние ключа (например, синхропосылки).
CPExportKey()	Используется для экспорта криптографических ключей из ключевого контейнера криптопровайдера, сохраняя их в защищённом виде.
CPGenRandom()	Используется для заполнения буфера случайными байтами.
CPGetKeyParam()	Возвращает параметры ключа.
CPGetUserKey()	Возвращает дескриптор одной из постоянных ключевых пар в ключевом контейнере.
CPImportKey()	Используется для импорта криптографического ключа из ключевого блоба в контейнер криптопровайдера.
CPSetKeyParam()	Устанавливает параметры ключа.

Таблица 4. Функции шифрования/расшифрования данных

Название	Описание
CPEncrypt()	Производит зашифрование данных. Одновременно может быть произведено их хэширование.
CPDecrypt()	Расшифровывает данные, предварительно зашифрованные функцией CPEncrypt(). Одновременно может быть произведено хэширование данных.

Таблица 5. Функции хэширования и ЭЦП

Название	Описание
CPCreateHash()	Инициализирует дескриптор нового объекта функции хэширования потока данных.
CPDestroyHash()	Разрушает объект функции хэширования.
CPDuplicateHash()	Создает точную копию объекта функции хэширования, включая все его переменные, определяющие внутреннее

	состояние объекта функции хэширования.
CPGetHashParam()	Возвращает параметры объекта функции хэширования и значение функции хэширования.
CPHashData()	Передаёт данные указанному объекту функции хэширования.
CPHashSessionKey()	Передаёт криптографический ключ указанному объекту функции хэширования. Это позволяет хэшировать ключи сессии без передачи ключа приложению.
CPSetHashParam()	Устанавливает параметры объекта хэширования.
CPSignHash()	Возвращает значение электронной цифровой подписи от значения функции хэширования.
CPVerifySignature()	Осуществляет проверку цифровой подписи, соответствующей объекту функции хэширования.

Далее приведен листинг программы, выполняющей инициализацию базового криптопровайдера Microsoft и генрирующей сессионный ключ (требует библиотеку Advapi32.dll).

```
//--------------------------------------------------------------------
#include “Wincrypt.h”

// Объявление и инициализация переменных.

HCRYPTPROV hCryptProv = NULL;        // дескриптор криптопровайдера
LPCSTR UserName = "MyKeyContainer";  // название ключевого контейнера
HCRYPTKEY hKey;                      // дескриптор ключа


//--------------------------------------------------------------------
// Инициализация криптопровайдера, получение дескриптора криптопровайдера


if(CryptAcquireContext(
  &hCryptProv,             // дескриптор криптопровайдера
  UserName,                // название ключевого контейнера
  NULL,                    // используем криптопровайдер по-умолчанию (Microsoft)
  PROV_RSA_FULL,           // тип провайдера
  0))                      // значение флага (выставляется в 0, чтобы предоставить
                           // возможность открывать существующий ключевой контейнер)
{
   printf("A cryptographic context with the %s key container \n",
 UserName);
```

```
    printf("has been acquired.\n\n");
}
else
{
//-------------------------------------------------------------------
// Возникла ошибка при инициализации криптопровайдера. Это может
// означать, что ключевой контейнер не был открыт, либо не существует.
// В этом случае функция получения дескриптора криптопровайдера может быть
// вызвана повторно, с измененным значением флага, что позволит создать
// новый ключевой контейнер.Коды ошибок определены в Winerror.h.
 if (GetLastError() == NTE_BAD_KEYSET)
 {
  if(CryptAcquireContext(
    &hCryptProv,
    UserName,
    NULL,
    PROV_RSA_FULL,
    CRYPT_NEWKEYSET))
   {
    printf("A new key container has been created.\n");
   }
   else
   {
    printf("Could not create a new key container.\n");
    exit(1);
   }
 }
 else
 {
    printf("A cryptographic service handle could not be "
       "acquired.\n");
    exit(1);
  }

} // Конец если.
```

```
//--------------------------------------------------------------------
//  Создание случайного сессионного ключа

 if(CryptGenKey(
        hCryptProv,
        CALG_RC4,
        CRYPT_EXPORTABLE,
        &hKey))
 {
       printf("A session key has been created.\n");
 }
 else
 {
        printf("Error during CryptGenKey.\n");
        exit(1);
 }



//--------------------------------------------------------------------
// По окончании работы все дескрипторы должны быть удалены.
if (!CryptDestroyKey(hKey))         // удаление дескриптора ключа
{
        printf("Error during CryptDestroyKey.\n");
        exit(1);
}

if (CryptReleaseContext(hCryptProv,0))    // удаление дескриптора криптопровайдера
{
  printf("The handle has been released.\n");
}
else
{
  printf("The handle could not be released.\n");
}
```

Рисунок 2 и Рисунок 3 демонстрируют обмен ключами по схемам Эль-Гамаля и Диффи-Хеллмана соответственно.

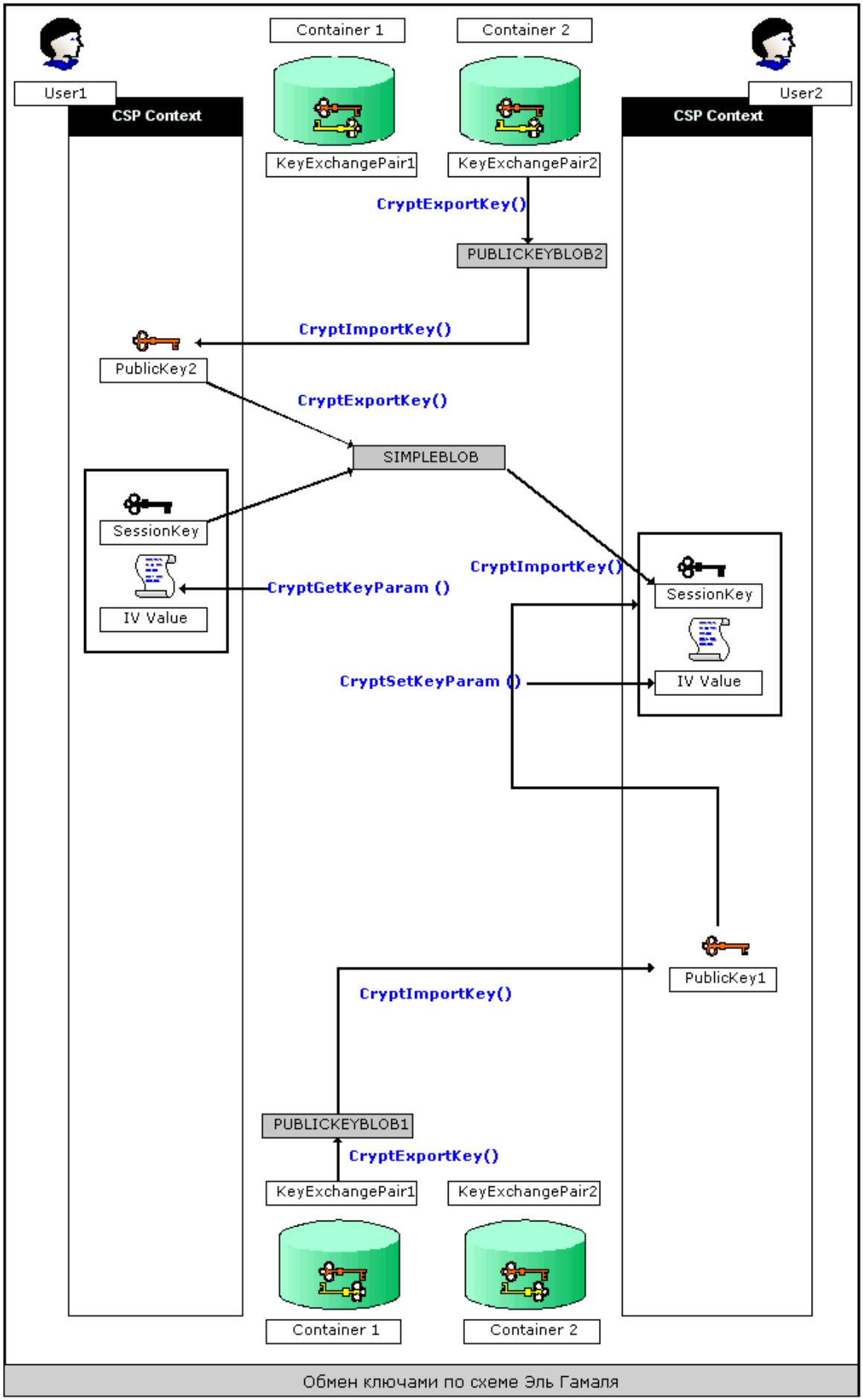

Рисунок 2. Обмен ключами по схеме Эль-Гамаля

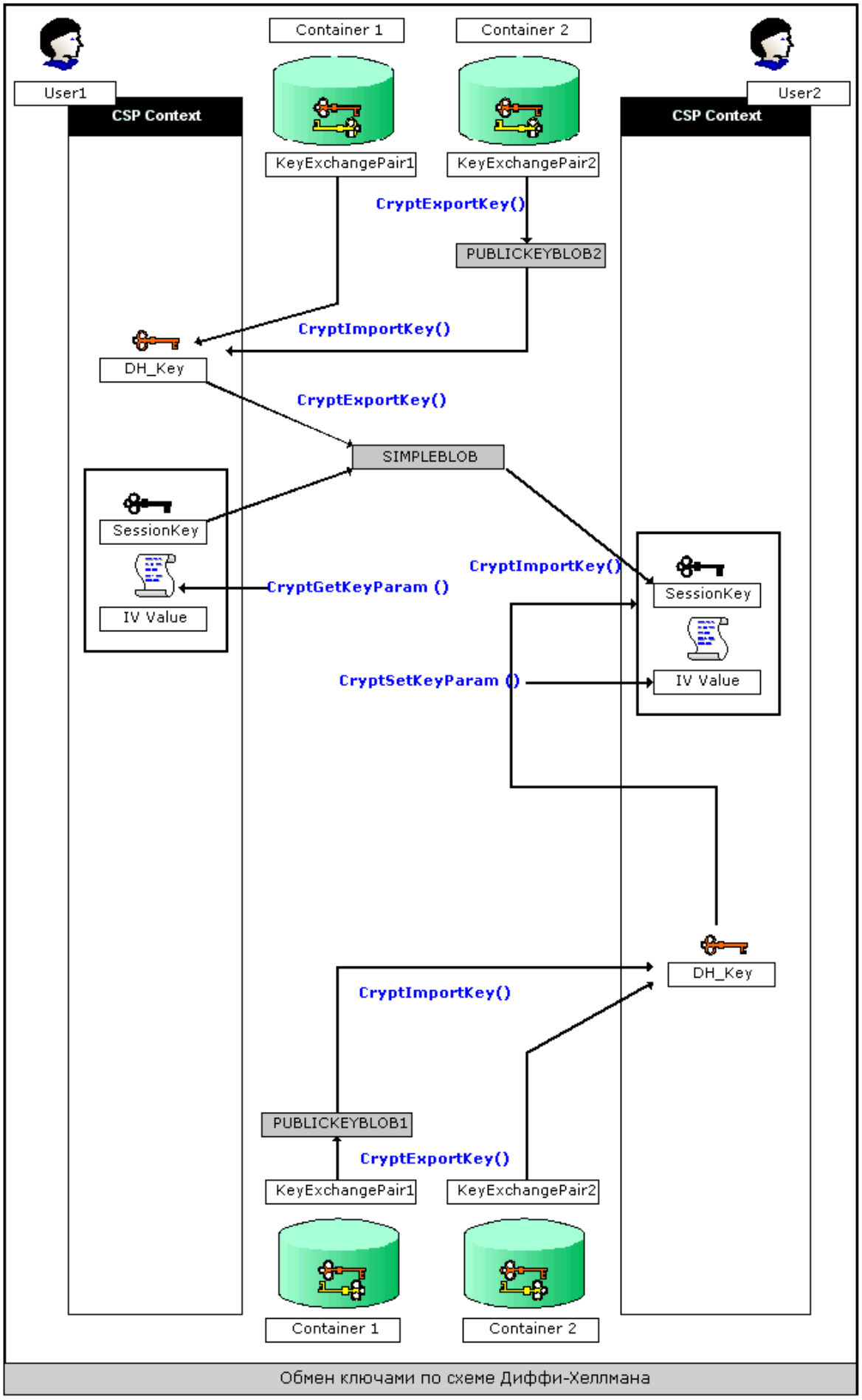

Рисунок 3. Обмен ключами по схеме Диффи-Хеллмана

ЭЦП

Отправитель:

1. Алиса формирует сообщение;
2. Алиса вычисляет хэш-функцию от своего сообщения;
3. Алиса шифрует значение полученной хэш-функции своим закрытым ключом;
4. Алиса отправляет бобу открытое сообщение и зашифрованное значение хэш-функции.

Получатель:

1. Боб вычисляет хэш-функцию от открытого сообщения Алисы;
2. Боб дешифрует полученное от Алисы значение хэш-функции открытым ключом Алисы;
3. Если значения полученные на этапах 1 и 2 одинаковы – ЭЦП верна.

Верба ДМ

Для выполнения рада заданий к лабораторной работе необходимо провести установку криптопровайдера Верба ДМ, разработанного силами студентов и сотрудников кафедры ЭВМ и систем ВолгГТУ. Данный криптопровайдер кроме стандартных функций CryptoAPI реализует и отечественный криптоалгоритмы.

Для установки криптопровайдера необходимо запустить файл VerbaDM setup.exe. Для работы с криптопровайдером необходим внешний ключевой носитель, в качестве которого может выступать ГМД 3,5", flash-накопитель, eToken-брелок или смарт-карта. **Обязательное условие** – скопировать на ключевой носитель папку RND. Кроме того, к Вашему проекту необходимо подключить заголовочные файлы mycspdef.h и myOIDs.h.

Задания

Вариант № 1.

Вывести список установленных криптопровайдеров на экран.
Написать тест скорости шифрования базового криптопровайдера Microsoft.

Вариант № 2.

Шифрование и дешифрование выбранного файла. Сессионный ключ экспортируется и хранится в заголовке зашифрованного файла. При дешифровании считывается и импортируется сессионный ключ. Разработать визуальный интерфейс.

Вариант № 3.

Подпись файла ЭЦП-2001. Проверка ЭЦП. Протестировать для нескольких ключевых контейнеров. Разработать визуальный интерфейс.

Вариант № 4.

На основе используемого в базовом криптопровайдере Microsoft генератора случайных чисел реализовать функцию формирования случайных массивов данных размером 40 байт. Каждый массив снабжается контрольной суммой (2 байта) и результат записывается в файл. В итоге работы функции – некоторое количество

таких файлов. Задаваемые параметры – путь сохранения файлов, количество файлов.

Вариант № 5.

Реализовать обмен сессионными ключами по схеме Диффи-Хелмана.

Вариант № 6.

Реализовать обмен сессионными ключами по схеме Эль-Гамаля.

Вариант № 7.

Написать программу, тестирующую работу инициализационных функций базового криптопровайдера Microsoft. Должны быть использованы различные значения параметров функций.

Вариант № 8.

Написать программу, тестирующую работу функций работы с ключами базового криптопровайдера Microsoft. Должны быть использованы различные значения параметров функций. Осуществить возможность формирования, экспорта, импорта сессионных ключей и ключевых пар.

Указания по выполнению, оформлению и отчету заданий

- Задания выполняются на любом языке программирования высокого уровня в среде ОС Windows XP;
- Необходим минимальный пользовательский интерфейс, позволяющий ввести произвольные входные данные и просмотреть результат;
- В качестве отчета выступают:
    - Титульный лист с информацией об исполнителе (ФИО, группа) и лабораторной работе (№, вариант, дата выполнения)
    - Исходный код программы;
    - Экранные формы и/или пользовательские диалоги;
    - Комментарии к алгоритму (по необходимости);
- На отчете могут быть заданы вопросы по соответствующему лекционному материалу + тестирование на программном стенде: