Тема: **Введение в средства криптографической защиты информации**

Санкт-Петербург 2022

Учебные цели:

1. Изучить типы, варианты построения СКЗИ и классы.
2. Изучить архитектуру и криптоинтерфейсы программных СКЗИ.
3. Изучить назначение и возможности криптопровайдеров.

Учебные вопросы:

1. Типы, варианты и классы СКЗИ.
2. Архитектура и криптоинтерфейсы программных СКЗИ.
3. Криптопровайдеры.

1. Типы, виды и классы СКЗИ

Типы шифровальных (криптографических) средств (СКЗИ)

а) средства шифрования - аппаратные, программные и программно-аппаратные шифровальные (криптографические) средства, реализующие алгоритмы криптографического преобразования информации для ограничения доступа к ней, в том числе при ее хранении, обработке и передаче;

б) средства имитозащиты - аппаратные, программные и программно-аппаратные шифровальные (криптографические) средства (за исключением средств шифрования), реализующие алгоритмы криптографического преобразования информации для ее защиты от навязывания ложной информации, в том числе защиты от модифицирования, для обеспечения ее достоверности и некорректируемости, а также обеспечения возможности выявления изменений, имитации, фальсификации или модифицирования информации;

в) средства электронной подписи;

г) средства кодирования - средства шифрования, в которых часть криптографических преобразований информации осуществляется с использованием ручных операций или с использованием автоматизированных средств, предназначенных для выполнения таких операций;

д) средства изготовления ключевых документов - аппаратные, программные, программно-аппаратные шифровальные (криптографические) средства, обеспечивающие возможность изготовления ключевых документов для шифровальных (криптографических) средств, не входящие в состав этих шифровальных (криптографических) средств;

Основные документы, связанные с разработкой СКЗИ

ТЕХНИЧЕСКИЙ КОМИТЕТ ПО СТАНДАРТИЗАЦИИ

«КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА ИНФОРМАЦИИ»

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ТК 26

МР 26.2.003-2012

Информационная технология

Криптографическая защита информации

Ключевой контейнер (дополнение к PKCS#15)

Р 50.1.110-2016 Информационная технология. Криптографическая защита информации. Контейнер хранения ключей

Р 50.1.110-2016

Группа П85

РЕКОМЕНДАЦИИ ПО СТАНДАРТИЗАЦИИ

Информационная технология

КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА ИНФОРМАЦИИ

Контейнер хранения ключей

Information technology. Cryptographic data security. Key storage container

ОКС 35.040

ОКСТУ 5002

Дата введения 2017-06-01

Виды шифровальных (криптографических) средств (СКЗИ)

а) аппаратные шифровальные (криптографические) средства - устройства и их компоненты, в том числе содержащие ключевую информацию, обеспечивающие возможность преобразования информации в соответствии с алгоритмами криптографического преобразования информации без использования программ для ЭВМ;

б) программные шифровальные (криптографические) средства - программы для ЭВМ и их части, в том числе содержащие ключевую информацию, обеспечивающие возможность преобразования информации в соответствии с алгоритмами криптографического преобразования информации в программно-аппаратных шифровальных (криптографических) средствах, информационных системах и телекоммуникационных системах, защищенных с использованием шифровальных (криптографических) средств;

в) программно-аппаратные шифровальные (криптографические) средства - устройства и их компоненты (за исключением информационных систем и телекоммуникационных систем), в том числе содержащие ключевую информацию, обеспечивающие возможность преобразования информации в соответствии с алгоритмами криптографического преобразования информации с использованием программ для электронных вычислительных машин, предназначенных для осуществления этих преобразований информации или их части.

Основные варианты криптографической защиты в программных и программно-аппаратных средствах

1. Использование инструкций криптопреобразования главного процессора.

2. Применяя возможности криптопроцессоров или криптосопрпроцессоров.

3. Посредством применения библиотек криптофункций и библиотек работы с большими числами (для несимметричных криптоалгоритмов).

4. Задействуя интерфейсы безопасности и криптоинтерфейсы.

5. Применяя криптопровайдеры.

6. Используя драйверы безопасности и криптопреобразований.

7. Используя ПЛИС, специализированные языки программирования , SDK и программаторы.

Основные документы, связанные с разработкой и применением программных СКЗИ

ТЕХНИЧЕСКИЙ КОМИТЕТ ПО СТАНДАРТИЗАЦИИ
«КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА ИНФОРМАЦИИ»

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ТК 26

МР 26.2.001-2013

Информационная технология

Криптографическая защита информации

Использование наборов алгоритмов шифрования на основе ГОСТ 28147-89 для протокола безопасности транспортного уровня (TLS)

ФЕДЕРАЛЬНОЕ АГЕНТСТВО
ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

РЕКОМЕНДАЦИИ ПО СТАНДАРТИЗАЦИИ

Р 123565.1.030 - 2020

Информационная технология

КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА ИНФОРМАЦИИ

Использование российских криптографических алгоритмов в протоколе безопасности транспортного уровня (TLS 1.3)

- Какие сайты запрашивают шифрование ГОСТ TLS
- Примеры сайтов, запрашивающих шифрование: www.gosuslugi.ru, сайты на домене .gov.ru, .kamgov.ru, .nalog.ru.
- Если сайт запрашивает шифрование ГОСТ TLS, Браузер проверяет, установлена ли программа КриптоПро CSP.

Классы СКЗИ

• Различают пять уровней **КС1, КС2, КС3, КВ, КА** криптографической защиты информации, не содержащей сведений, составляющих государственную тайну, определенных в порядке возрастания количества и жесткости предъявляемых к криптосредствам требований,

• и, соответственно, **пять классов криптосредств**, также обозначаемых через **КС1, КС2, КС3, КВ, КА**.

• Уровень криптографической защиты информации, не содержащей сведений, составляющих государственную тайну, обеспечиваемой криптосредством, определяется заказчиком этого криптосредства в ТЗ путем отнесения нарушителя, действиям которого должно противостоять криптосредство, к конкретному типу.

• При отнесении заказчиком нарушителя к типу Н1 криптосредство должно обеспечить криптографическую защиту по уровню КС1, к типу Н2 – КС2, к типу Н3 – КС3, к типу Н4 – КВ, к типу Н5 – КА1.

Начальные классы СКЗИ

- Класс КС1 выбирается в предположении, что атака на систему осуществляется с территорий, находящихся за пределами защищаемой области. Считается, что в число лиц осуществляющих атаку не входят профессионалы по анализу уязвимостей ПО и технических средств (ТС), дополнительно предполагается, что злоумышленники об атакуемой системе имеют информацию только из открытых источников.
- Класс КС2 должен противостоять таким же угрозам, что и КС1, но предполагается, что лица осуществляющие атаку могут иметь доступ в защищаемую зону и располагают документацией о технических способах защиты атакуемых ИС.
- Класс КС3, блокируя угрозы по классу КС2, должен противодействовать и атакам со стороны лиц, имеющих доступ и к оборудованию, обеспечивающему КЗИ.

Старшие классы СКЗИ

- КВ — класс, выдерживающий все атаки КС3, но уровень защищённости должен быть выше, поскольку в осуществлении и планировании атак могут участвовать разработчики и аналитики используемых на объекте ПО и ТС. Предполагается, что эти специалисты имели возможность проводить исследования СКЗИ защищаемого объекта.
- КА — это класс, способный противостоять всем угрозам КВ, но и дополнительно отражать атаки с участием лиц, знающих незадекларированные возможности системного ПО и ТС защищаемой системы, имевшие опыт по исследованию атак на системы с таким же оборудованием и ПО.

Примеры СКЗИ и их классов

НАУЧНОЕ ПРОИЗВОДСТВЕННОЕ ПРЕДПРИЯТИЕ «ФАКТОР-ТС»

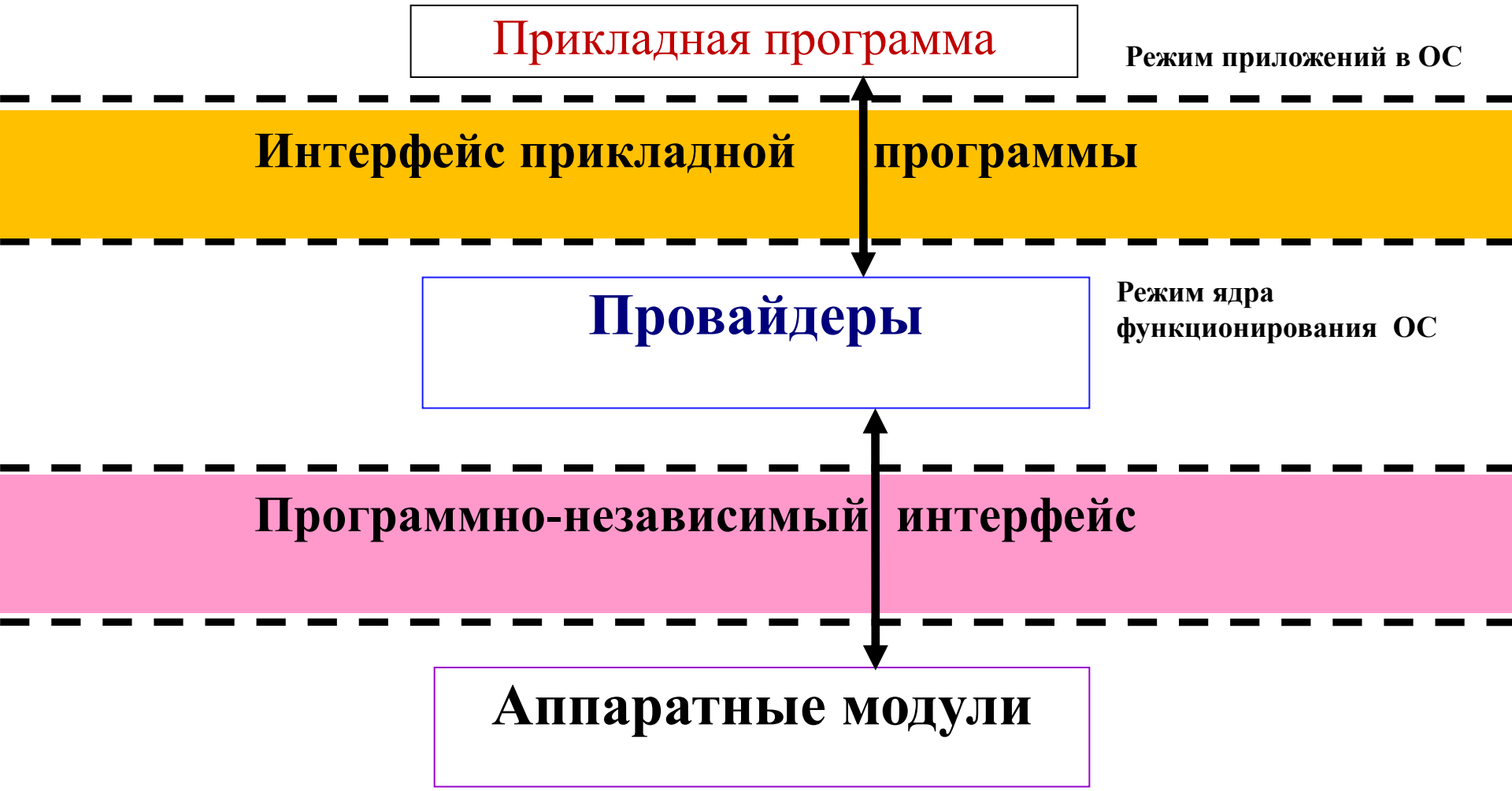
РЕШЕНИЯ • ДИОНИС • СЕРТИФИКАТЫ • ЛИЦЕНЗИИ • ДОКУМЕНТАЦИЯ • ПОДДЕРЖКА

Перечень сертификатов

- Сертификат соответствия требованиям ГОСТ Р ИСО 9001-2001.
- Сертификат ФСБ России на соответствие СКЗИ «DSignCA» требованиям по классу КС1 и КС2.
- Сертификат ФСБ России на соответствие СКЗИ «DPostCA» требованиям по классу КС1 и КС2.
- Сертификат ФСБ России на соответствие СКЗИ «МГК» требованиям по классу КС1 и КС2.
- Сертификат ФСБ России на соответствие СКЗИ «DSec» требованиям по классу КС3.
- Сертификат ФСБ России на соответствие СКЗИ «Donis TS/FW 16000/КВ2» требованиям по классу КВ2.
- Сертификат ФСБ России на соответствие СКЗИ «Donis TS/FW 16000/КС3» требованиям по классу КС3.
- Сертификат ФСБ России на соответствие СКЗИ «Donis-LX» требованиям по классу КС2.
- Сертификат ФСБ России на соответствие ПАК «Маршрутизатор DioNIS TS/FW 16000R» требованиям по защите мультипротокольного оборудования.
- Сертификат ФСБ России на соответствие СКЗИ «М-479К» требованиям к шифровальным средствам класса КДС-1.02.
- Сертификат ФСБ России на соответствие СКЗИ «Автоматизированное рабочее место генерации ключей АРМ ГК/КВ2» требованиям по классу КВ2.
- Сертификат ФСТЭК России на соответствие межсетевого экрана «DioNIS Firewall» показателям защищенности по 2-му классу и по уровню контроля НДВ по 2-му классу.
- Сертификат в области связи на соответствие IP-маршрутизатора «DioNIS TS/FW» техническим требованиям к аппаратуре маршрутизации пакетов IP.

2. Архитектура и криптоинтерфейсы программных СКЗИ

Архитектура построения прикладных программ в ОС Windows - Windows Open Services Architecture (WOSA)



ИНТЕРФЕЙС (англ. interface – сопрягать, согласовывать) – набор правил, определяющий структуру данных и способ (алгоритм) обмена данными между взаимодействующими объектами (соседними уровнями взаимодействия).

ИНТЕРФЕЙС ПРИКЛАДНОЙ ПРОГРАММЫ или **ИНТЕРФЕЙС ПРИКЛАДНОГО ПРОГРАММИРОВАНИЯ** (application program (programming) interface, **API**) – интерфейс, определяющий **порядок взаимодействия между прикладными программами и операционной системой (процессами пользовательского режима операционной системы)**

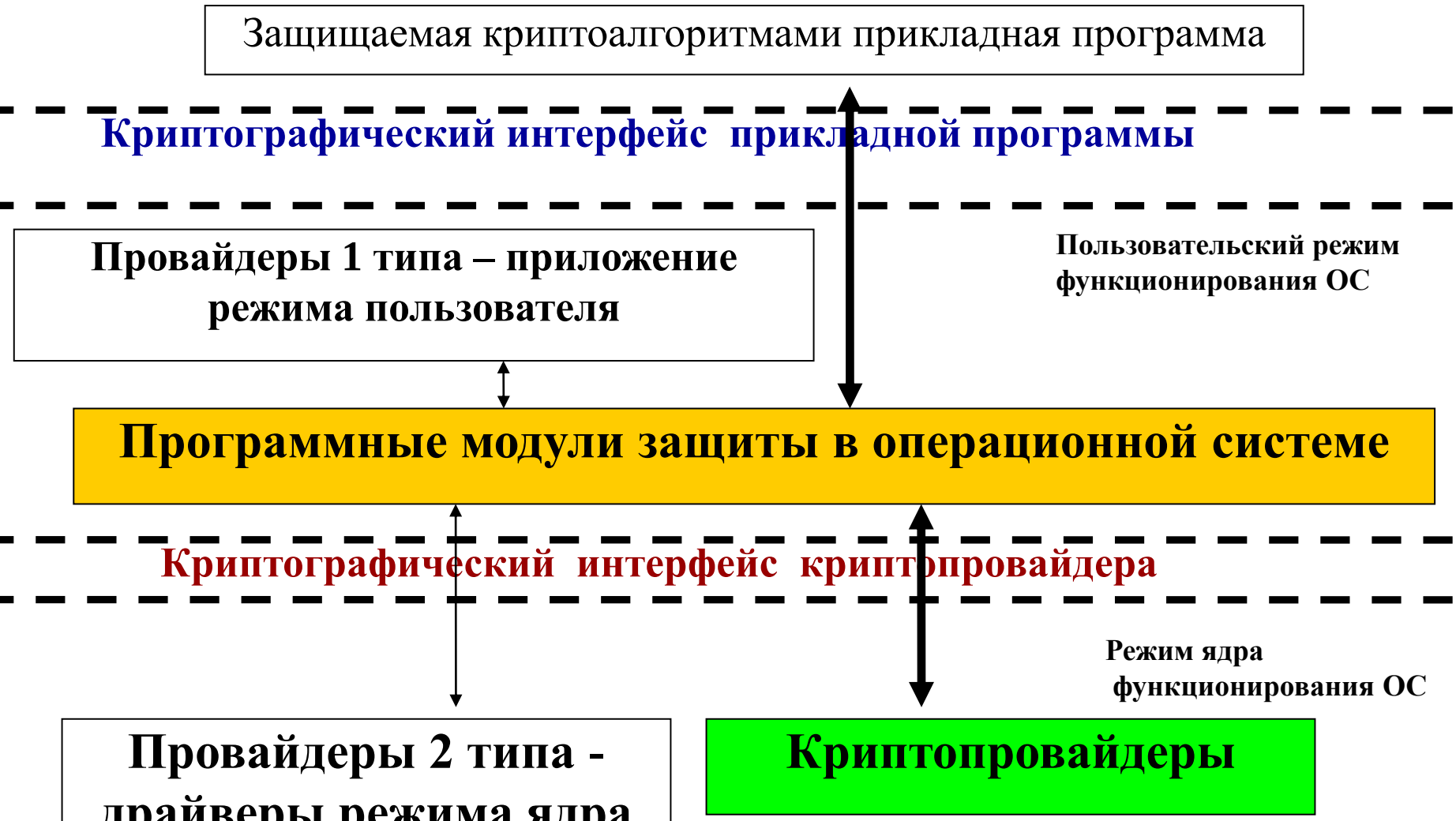
ПРИМЕРЫ: **Win32 API**, **WinSockAPI**, **ISAPI**, **MAPI**, **TSAPI**, **VirusScanningAPI**, **Signal Computing System Architecture API**

Основное требование: данные прикладной программы должны быть криптографически защищены

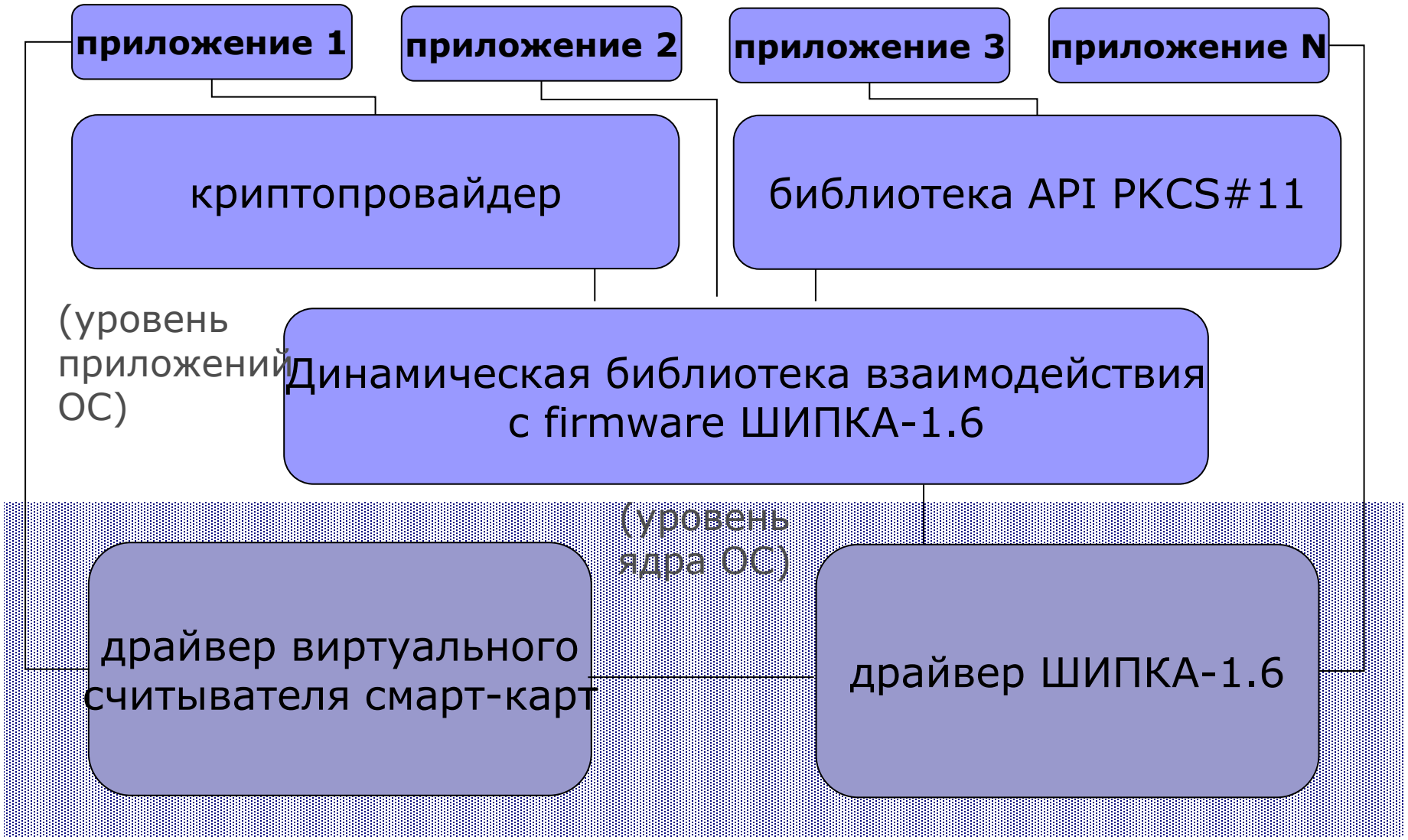
ВОПРОС: На каком уровне архитектуры целесообразно “размещать” криптоалгоритмы ?

ВОПРОС: Как проектировать криптоалгоритмы для защищенных программ работающих в различных ОС ?

Архитектура построения защищенных прикладных программ



Пример архитектуры СКЗИ ШИПКА-1.6 (ОКБ САПР)



ОСНОВНЫЕ ЭЛЕМЕНТЫ АХИТЕКТУРЫ

КРИПТОГРАФИЧЕСКИЙ ИНТЕРФЕЙС ПРИКЛАДНОЙ ПРОГРАММЫ (КИПП) – **интерфейс для организации *защищенного* взаимодействия прикладных программ с операционной системой.**

КРИПТОПРОВАЙДЕР – **библиотека динамической компоновки (файлы с расширением .DLL), содержащая программные реализации криптоалгоритмов, и позволяющая прикладной программе применять криптографические функции над ее данными без получения непосредственного доступа к ключевым данным**

Вывод: КИПП определяет **порядок обращения прикладных программ к библиотеке программ, реализующих криптографические функции**

ОСНОВНЫЕ КРИПТОГРАФИЧЕСКИЕ ИНТЕРФЕЙСЫ

Основные криптографические интерфейсы прикладных программ

«Верхний» уровень ОС

Криптографический интерфейс прикладного программирования (Crypto API)

Интерфейс прикладных программ общих служб безопасности Generic Security Service (GSS API)

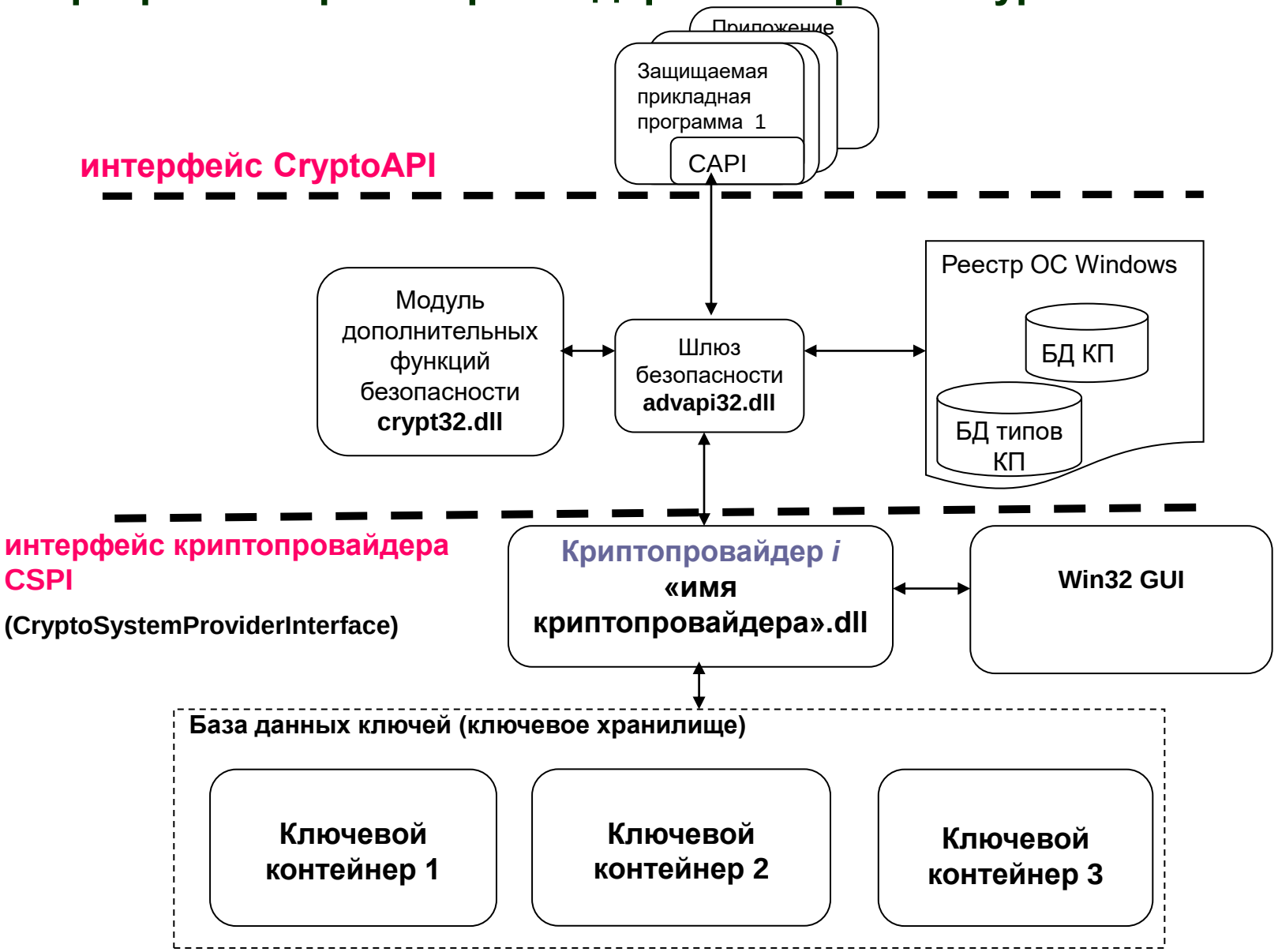
Основные криптографические интерфейсы криптопровайдеров

«Нижний» уровень ОС

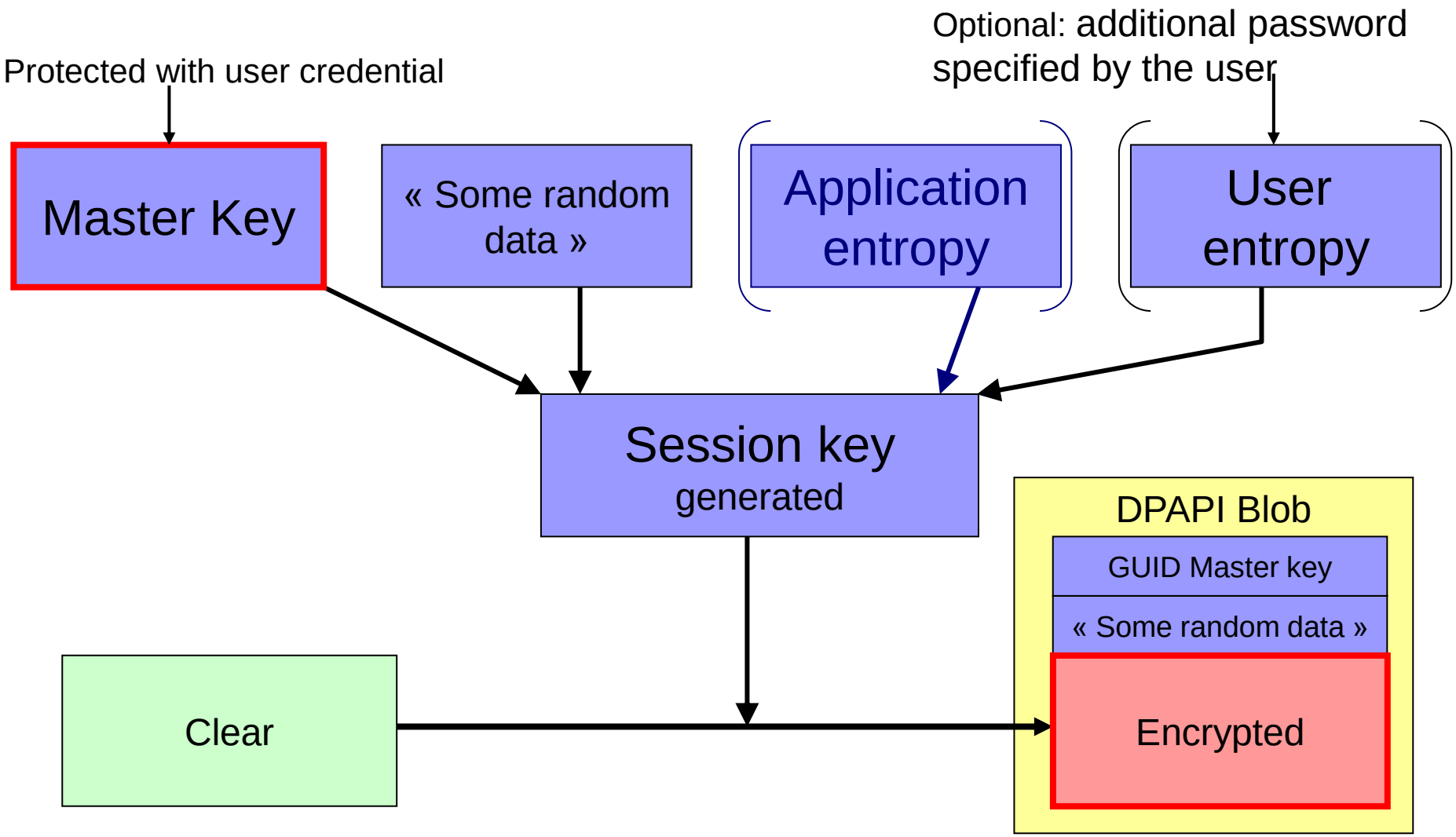
Интерфейс провайдера криптографических сервисных служб (CryptoServiceProviderInterface (Crypto SPI))

Интерфейс провайдера поддержки (поставщика) безопасности (Security Support Provider Interface (SSPI)

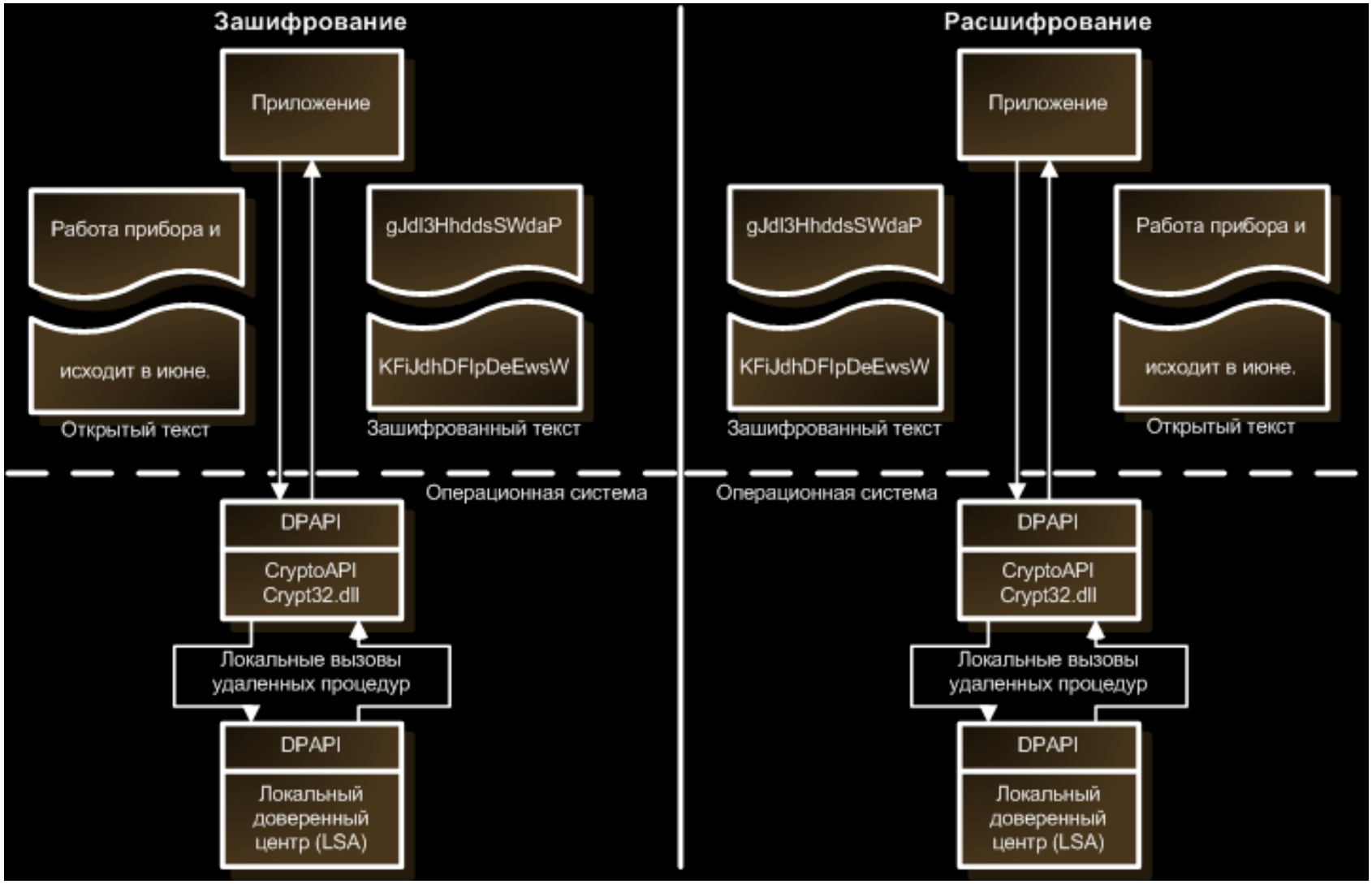
модули процесса взаимодействия защищаемых прикладных программ с криптопровайдерами в архитектуре ОС Windows



DPAPI - Data Protection API



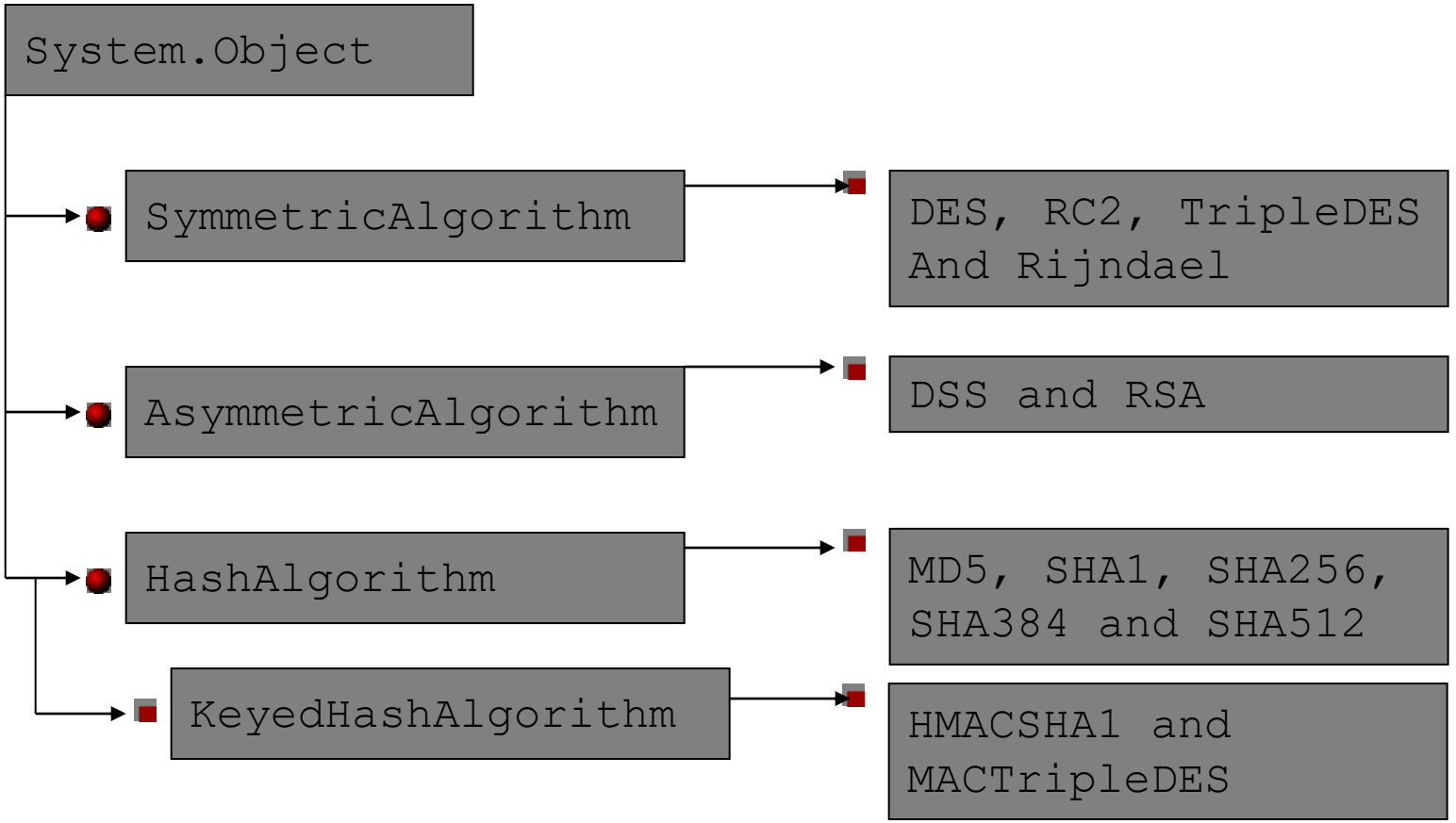
Data Protection API (DPAPI)



Криптографические классы в .NET (библиотека .NET Framework - язык C#)

- Большая часть криптографических классов (не абстрактных) .NET базируется на криптопровайдерах CryptoAPI. Однако имеются и исключения (например, SHA256Managed). Иерархия классов .NET позволяет абстрагироваться от конкретной реализации алгоритма, что может обеспечить простой переход на не привязанные к CryptoAPI классы в будущем.
- Симметричные блочные шифры представлены в .NET классами **DESCryptoServiceProvider**, **TripleDESCryptoServiceProvider**, **RijndaelManaged**. Все эти классы являются потомками абстрактного класса **SymmetricAlgorithm,** описывающего все семейство блочных алгоритмов с симметричными ключами. Класс описывает свойства, позволяющие манипулировать основными параметрами алгоритма: размером блока, режимом работы, инициализационным вектором, ключом и другими.
- Методы **CreateEncryptor** и **CreateDecryptor**, возвращающие контексты (интерфейс **ICryptoTransform**) предназначены для криптографических трансформаций данных.
- Методы **GenerateKey** и **GenerateIV** для генерации ключей и инициализационных векторов. Конкретные реализации наследуются от этого класса (возможно, через другие абстрактные классы – например, **DESCryptoServiceProvider** наследуется от класса **DES**, унаследованного от **SymmetricAlgorithm**).

System.Security.Cryptography



Возможности CryptoAPI

- изолирует приложение от модулей CSP и позволяет использовать различные CSP без модификации кода приложения.
- позволяет выбирать CSP по своему усмотрению.
- доступен в операционных системах Windows, Macintosh, Unix, Linux.

Функции CryptoAPI 1.0: генерация ключей и обмен ими, шифрование/дешифрование данных, хеширование, формирование цифровых подписей и их верификация

Функции CryptoAPI 2.0:

поддержка стандартных форматов сертификатов X.509 версии 3, ASN.1 и DER.

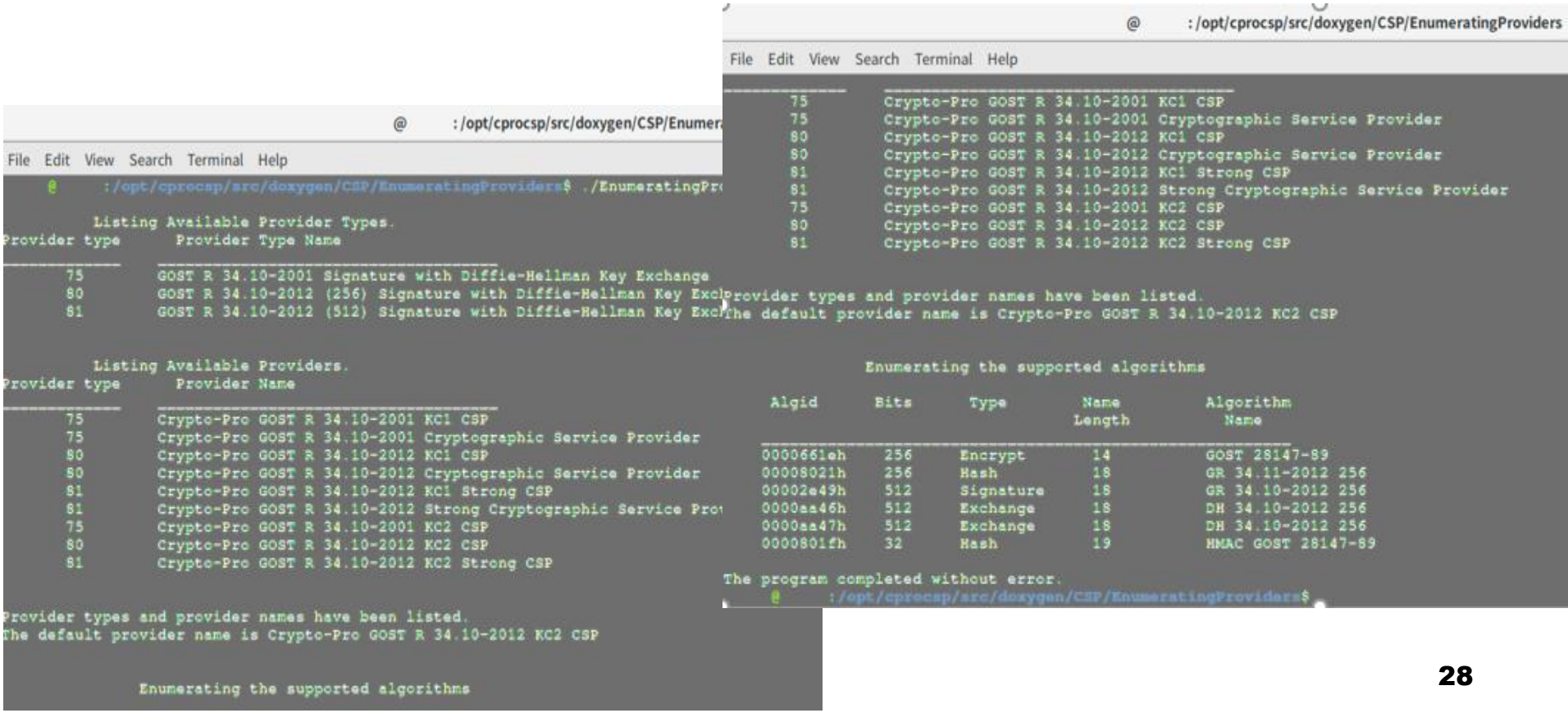
CAPILite - Crypto-Pro GOST R 34.10-2012 KC2 CSP

CAPILite – представляет частичный аналог Microsoft CryptoAPI и может выполнять почти все те же, используя такой же программный интерфейс.

Дополнительно устанавливается пакет со средствами разработки «lsb-cprocsp-devel», также находящийся внутри.

После выполнения вышеописанных шагов должна появиться директория «/opt/cprocsp», содержащая необходимые для работы файлы.

Чтобы убедиться в работоспособности установленного криптопровайдера, а также получить список доступных для использования типов провайдеров и алгоритмов, необходимо выполнить сборку и запустить пример, находящийся в папке «/opt/cprocsp/src/doxygen/CSP/EnumeratingProviders».



Необходимым условием применения CryptoAPI подключение файла

wincrypt.h **или** *WinCryptEx.h* (для КриптоПРО CSP)

Фрагмент wincrypt.h

```
// Some RSA sub-ids
#define ALG_SID_RSA_ANY             0
#define ALG_SID_RSA_PKCS            1
#define ALG_SID_RSA_MSATWORK        2
#define ALG_SID_RSA_ENTRUST         3
#define ALG_SID_RSA_PGP             4

// Some DSS sub-ids
//
#define ALG_SID_DSS_ANY             0
#define ALG_SID_DSS_PKCS            1
#define ALG_SID_DSS_DMS             2

// Block cipher sub ids
// DES sub_ids
#define ALG_SID_DES                 1
#define ALG_SID_3DES                3
#define ALG_SID_DESX                4
#define ALG_SID_IDEA                5
#define ALG_SID_CAST                6
#define ALG_SID_SAFERSK64           7
#define ALG_SID_SAFERSK128          8
#define ALG_SID_3DES_112            9
#define ALG_SID_CYLINK_MEK          12
#define ALG_SID_RC5                 13
```

WinCryptEx.h

```
#define CP_GR3410_2001_PROV_A "Crypto-Pro GOST R 34.10-2001
Cryptographic Service Provider"
#define CP_GR3410_2001_PROV_W L"Crypto-Pro GOST R 34.10-2001
Cryptographic Service Provider"
#endif

#define CP_SMARTCARD_PROV_A "Crypto-Pro SmartCard CSP"
#define CP_SMARTCARD_PROV_W L"Crypto-Pro SmartCard CSP"

#define CP_KC1_GR3410_94_PROV_A "Crypto-Pro GOST R 34.10-94 KC1
CSP"
#define CP_KC1_GR3410_94_PROV_W L"Crypto-Pro GOST R 34.10-94
KC1 CSP"

#define CP_KC1_GR3410_2001_PROV_A "Crypto-Pro GOST R 34.10-2001
KC1 CSP"
#define CP_KC1_GR3410_2001_PROV_W L"Crypto-Pro GOST R 34.10-
2001 KC1 CSP"

#define CP_KC2_GR3410_94_PROV_A "Crypto-Pro GOST R 34.10-94 KC2
CSP"
#define CP_KC2_GR3410_94_PROV_W L"Crypto-Pro GOST R 34.10-94
KC2 CSP"

#define CP_KC2_GR3410_2001_PROV_A "Crypto-Pro GOST R 34.10-2001
KC2 CSP"
#define CP_KC2_GR3410_2001_PROV_W L"Crypto-Pro GOST R 34.10-
2001 KC2 CSP"
```

Подключение к CAPILite

Для подключения СКЗИ к прикладному ПО необходимо пользоваться интерфейсом CAPILite:

- можно либо использовать библиотеку libcapi10 из пакета rdr,
- либо libcapi20 из пакета capilite.

libcapi10 обладает ограниченным функционалом: по объему он соответствует интерфейсу Microsoft CSP.

libcapi20 является частичным эквивалентом Microsoft CryptoAPI 2.0 и рекомендуется к использованию.

Для подключения требуемых библиотек к проекту Qt, в файл с расширением «*.pro» следует добавить следующие строки:

```
INCLUDEPATH += /opt/cprocsp/include /opt/cprocsp/include/cpcsp
LIBS += -L/opt/cprocsp/lib/amd64 -lcapi10 -lcapi20 -lrdrsup –lssp
DEFINES += UNIX SIZEOF_VOID_P=8 QT_STATIC_BUILD
```

После этого в заголовочных файлах можно подключать необходимые библитеки, например, «WinCryptEx.h» и пр.

CryptoAPI System Architecture - MSDN Library - Visual Studio 2008 - Microsoft Document Explorer

File Edit View Tools Window Help

Back How Do I Search Index Contents Help Favorites MSDN Forums

Contents

Filtered by:

(unfiltered)

- Graphics and Multimedia
- Messaging and Collaboration
- Mobile PC
- Networking
- Security
  - Active Directory Rights Management Scripting API
  - Active Directory Rights Management Services SDK
  - Parental Controls for Windows Vista
  - Best Practices for the Security APIs
  - Authentication
  - Authorization
  - Software Licensing API
  - Cryptography
    - Cryptography
      - About Cryptography
        - Cryptography Essentials
          - Cryptography, CryptoAPI, and CAPICOM
            - Cryptography Concepts
            - Additional Documentation on Cryptography
            - CryptoAPI System Architecture
            - CSPs and the Cryptography Process
            - Cryptographic Service Provider Contexts
          - Cryptographic Keys
          - Encoded and Decoded Data
          - Data Encryption and Decryption
          - Hashes and Digital Signatures
          - Digital Certificates
          - Digital Messages
        - Cryptography Tools
        - Cryptographic Service Providers
        - Certificate Services
        - Certificate Enrollment Control
        - CAPICOM Versions
        - Crypt32.dll Versions
      - Using Cryptography
      - Cryptography Reference

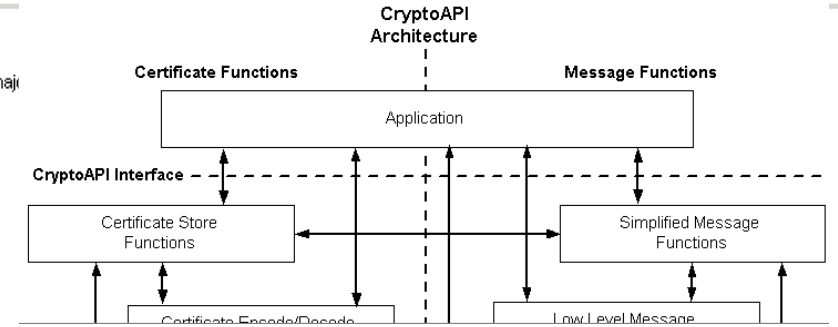
CryptoAPI System Architecture

URL: ms-help://MS.MSDNQTR.v90.en/seccrypto/security/cryptoapi_system_architecture.htm#certificate_encode/decode_functions

CryptoAPI System Architecture

The CryptoAPI system architecture is composed of five maj

- Base Cryptographic Functions
- Certificate Encode/Decode Functions
- Certificate Store Functions
- Simplified Message Functions
- Low-level Message Functions



Base Cryptographic Functions

- *Context functions* used to connect to a CSP. These functions enable applications to choose a specific CSP by name or to choose a specific CSP that can provide a needed class of functionality.
- *Key generation functions* used to generate and store *cryptographic keys*. Full support is included for changing *chaining modes*, *initialization vectors*, and other encryption features. For more information, see Key Generation and Exchange Functions.
- *Key exchange functions* used to exchange or transmit keys. For more information, see Cryptographic Key Storage and Exchange and Key Generation and Exchange Functions.

Certificate Encode/Decode Functions

- Functions used to encrypt or decrypt data. Support is also included for *hashing* data. For more information, see Data Encryption and Decryption Functions and Data Encryption and Decryption.

Certificate Store Functions

- Functions used to manage collections of digital certificates. For more information, see Digital Certificates and Certificate Store Functions.

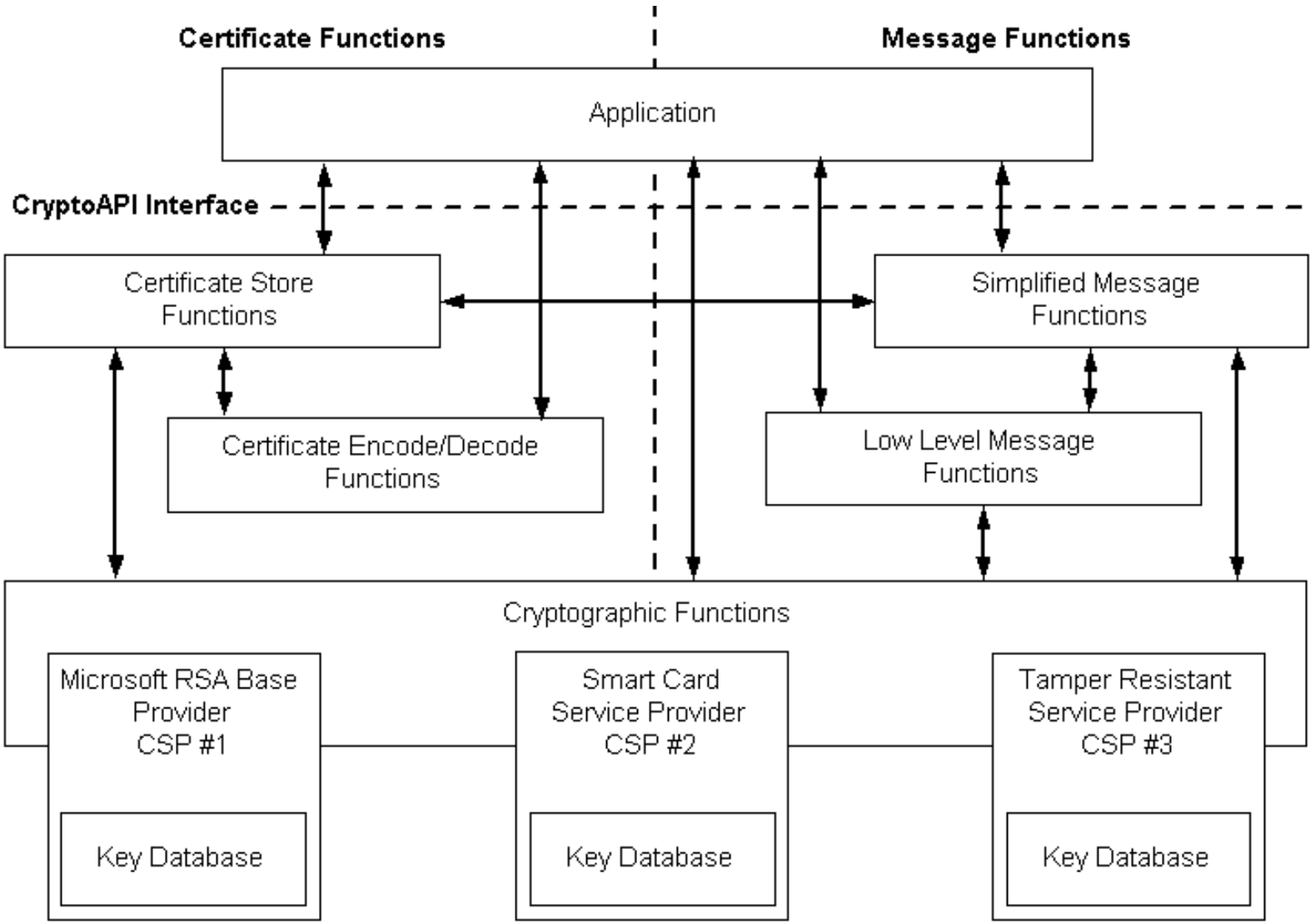
Simplified Message Functions

- Functions used to encrypt and decrypt messages and data.
- Functions used to sign messages and data.
- Functions used to verify the authenticity of signatures on received messages and related data.

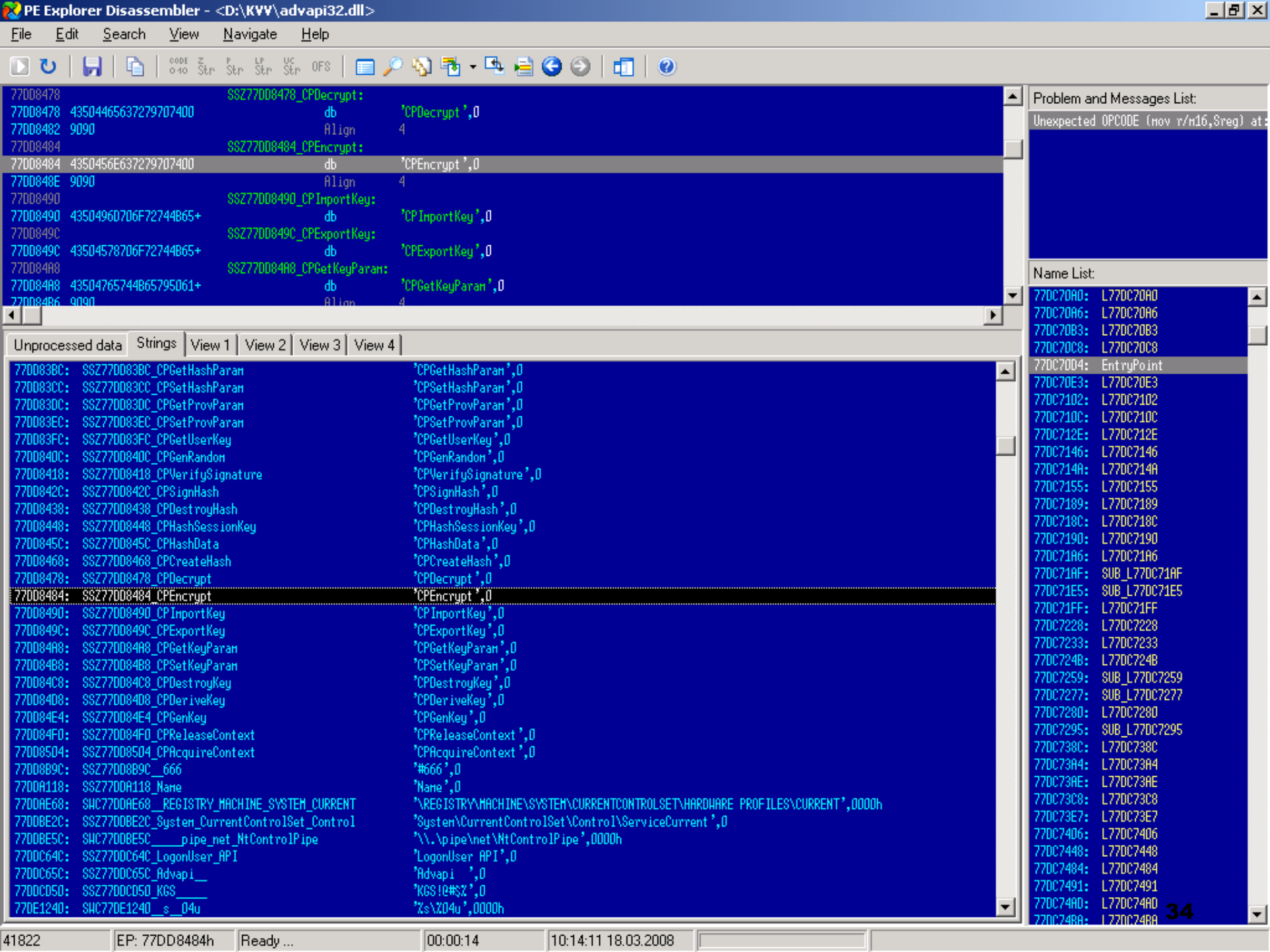
For more information, see Simplified Messages and Simplified Message Functions.

Low-level Message Functions

- Functions used to perform all of the tasks performed by the simplified message functions. The low-level message functions provide more flexibility than the simplified message functions but require more function calls. For more information, see Low-level Messages and Low-level Message Functions.


Certificate Functions
Message Functions
Application
CryptoAPI Interface
Certificate Store Functions
Simplified Message Functions
Certificate Encode/Decode Functions
Low Level Message Functions
Cryptographic Functions
Microsoft RSA Base Provider CSP #1
Key Database
Smart Card Service Provider CSP #2
Key Database
Tamper Resistant Service Provider CSP #3
Key Database

Однотипные криптофункции интерфейсов верхнего и нижних уровней являются совместимыми и различаются префиксами.


PE Explorer Disassembler - <D:\KVV\advapi32.dll>
File Edit Search View Navigate Help
77DD8478 SSZ77DD8478_CPDecrypt:
77DD8478 4350446563727970740O db 'CPDecrypt',0
77DD8482 9090 Align 4
77DD8484 SSZ77DD8484_CPEncrypt:
77DD8484 4350456E637279707400 db 'CPEncrypt',0
77DD848E 9090 Align 4
77DD8490 SSZ77DD8490_CPImportKey:
77DD8490 4350496D706F72744B65+ db 'CPImportKey',0
77DD849C SSZ77DD849C_CPExportKey:
77DD849C 4350457870GF72744B65+ db 'CPExportKey',0
77DD84A8 SSZ77DD84A8_CPGetKeyParam:
77DD84A8 4350476574B65795O61+ db 'CPGetKeyParan',0
77DD84B6 9090 Align 4
Problem and Messages List:
Unexpected OPCODE (mov r/m16,Sreg) at:
Name List:
77DC70AD: L77DC70AD
77DC70A6: L77DC70A6
77DC70B3: L77DC70B3
77DC70C8: L77DC70C8
77DC70D4: EntryPoint
77DC70E3: L77DC70E3
77DC7102: L77DC7102
77DC710C: L77DC710C
77DC712E: L77DC712E
77DC7146: L77DC7146
77DC714A: L77DC714A
77DC7155: L77DC7155
77DC7189: L77DC7189
77DC718C: L77DC718C
77DC719D: L77DC719D
77DC71A6: L77DC71A6
77DC71AF: SUB_L77DC71AF
77DC71E5: SUB_L77DC71E5
77DC71FF: L77DC71FF
77DC7228: L77DC7228
77DC7233: L77DC7233
77DC724B: L77DC724B
77DC7259: SUB_L77DC7259
77DC7277: SUB_L77DC7277
77DC728D: L77DC728D
77DC7295: SUB_L77DC7295
77DC738C: L77DC738C
77DC73A4: L77DC73A4
77DC73AE: L77DC73AE
77DC73C8: L77DC73C8
77DC73E7: L77DC73E7
77DC74D6: L77DC74D6
77DC7448: L77DC7448
77DC7484: L77DC7484
77DC7491: L77DC7491
77DC74AD: L77DC74AD
77DC74BA: L77DC74BA
Unprocessed data Strings View 1 View 2 View 3 View 4
77DD83BC: SSZ77DD83BC_CPGetHashParam 'CPGetHashParan',0
77DD83CC: SSZ77DD83CC_CPSetHashParan 'CPSetHashParan',0
77DD83DC: SSZ77DD83DC_CPGetProvParan 'CPGetProvParan',0
77DD83EC: SSZ77DD83EC_CPSetProvParan 'CPSetProvParan',0
77DD83FC: SSZ77DD83FC_CPGetUserKey 'CPGetUserKey',0
77DD840C: SSZ77DD840C_CPGenRandom 'CPGenRandom',0
77DD8418: SSZ77DD8418_CPVerifySignature 'CPVerifySignature',0
77DD842C: SSZ77DD842C_CPSignHash 'CPSignHash',0
77DD8438: SSZ77DD8438_CPDestroyHash 'CPDestroyHash',0
77DD8448: SSZ77DD8448_CPHashSessionKey 'CPHashSessionKey',0
77DD845C: SSZ77DD845C_CPHashData 'CPHashData',0
77DD8468: SSZ77DD8468_CPCreateHash 'CPCreateHash',0
77DD8478: SSZ77DD8478_CPDecrypt 'CPDecrypt',0
77DD8484: SSZ77DD8484_CPEncrypt 'CPEncrypt',0
77DD8490: SSZ77DD8490_CPImportKey 'CPImportKey',0
77DD849C: SSZ77DD849C_CPExportKey 'CPExportKey',0
77DD84A8: SSZ77DD84A8_CPGetKeyParam 'CPGetKeyParam',0
77DD84B8: SSZ77DD84B8_CPSetKeyParam 'CPSetKeyParam',0
77DD84C8: SSZ77DD84C8_CPDestroyKey 'CPDestroyKey',0
77DD84D8: SSZ77DD84D8_CPDeriveKey 'CPDeriveKey',0
77DD84E4: SSZ77DD84E4_CPGenKey 'CPGenKey',0
77DD84F0: SSZ77DD84F0_CPReleaseContext 'CPReleaseContext',0
77DD8504: SSZ77DD8504_CPAcquireContext 'CPAcquireContext',0
77DD8B9C: SSZ77DD8B9C__666 '#666',0
77DDA118: SSZ77DDA118_Name 'Name',0
77DDAE68: SWC77DDAE68__REGISTRY_MACHINE_SYSTEM_CURRENT '\REGISTRY\MACHINE\SYSTEM\CURRENTCONTROLSET\HARDWARE PROFILES\CURRENT',0000h
77DDBE2C: SSZ77DDBE2C_System_CurrentControlSet_Control 'System\CurrentControlSet\Control\ServiceCurrent',0
77DDBE5C: SWC77DDBE5C____pipe_net_NtControlPipe '\\.\pipe\net\NtControlPipe',0000h
77DDC64C: SSZ77DDC64C_LogonUser_API 'LogonUser API',0
77DDC65C: SSZ77DDC65C_Advapi__ 'Advapi ',0
77DDCD5D: SSZ77DDCD5D_KGS____ 'KGS!@#$%',0
77DE1240: SWC77DE1240__s__04u '%s\%04u',0000h
41822
EP: 77DD8484h
Ready ...
00:00:14
10:14:11 18.03.2008
34

Выводы по вопросу 2:

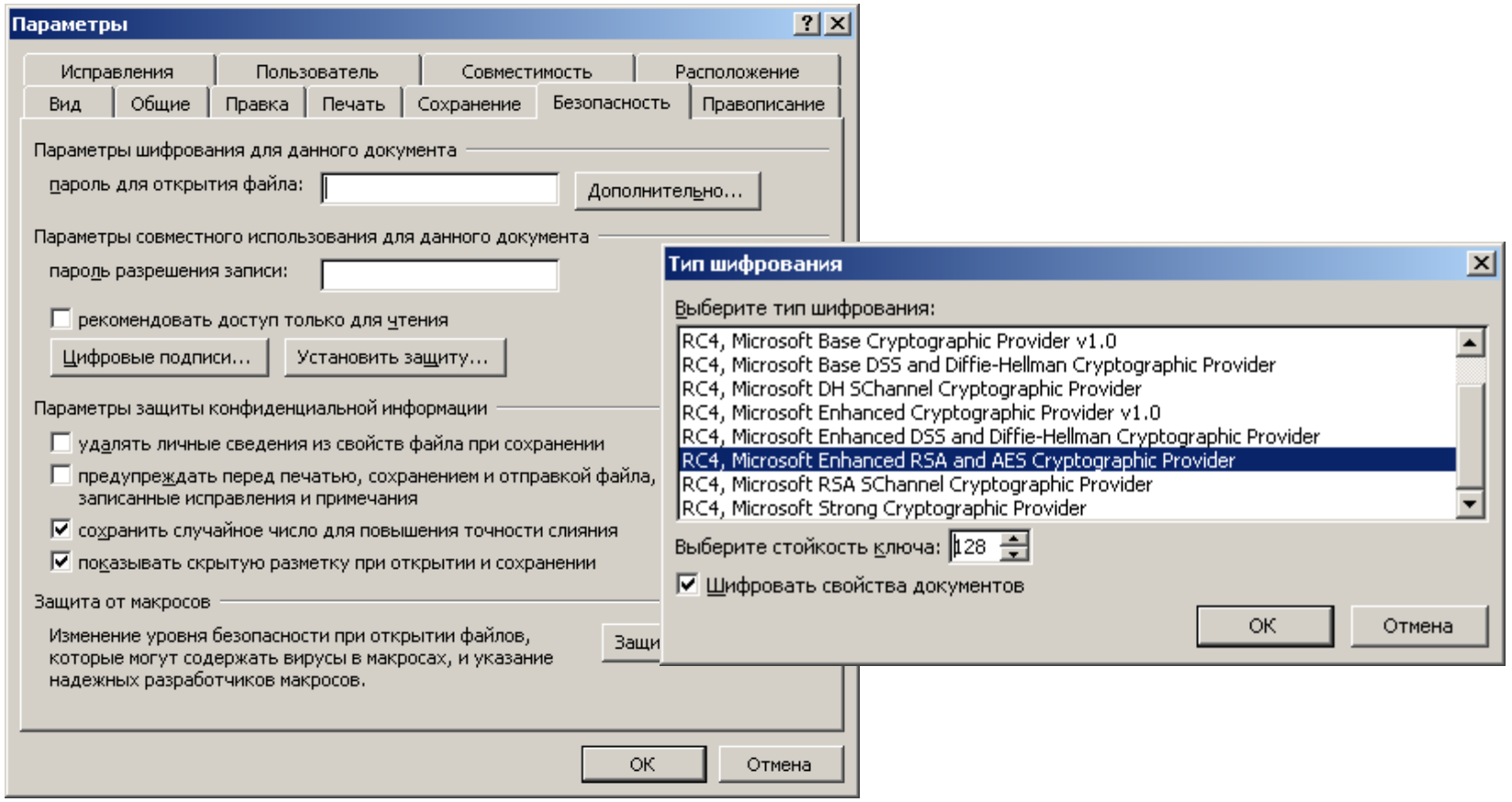
1. **При разработке защищенных прикладных программ применяется архитектура с двумя криптоинтерфейсами: верхнего уровня (аппаратно-независимым интерфейсом прикладной программы) и нижнего (программно-независимым интерфейсом – интерфейсом криптопровайдера) уровней.**

2. **В целях обеспечения безопасности ключевых данных создана двухконтурная система, исключающая возможность прямого обращения прикладной программы (разработчика/пользователя) через криптоинтерфейс нижнего уровня к обрабатываемым данным и, соответственно, и к ключевым данным. Прикладная программа с криптопровайдером связаны через шлюз-посредник (программный модуль ОС).**

3. **КИПП позволяет разработчику облегчить и упростить работу по криптозащите данных прикладной программы, посредством обращения к различным установленным в ОС криптопровайдерам и вызова уже реализованных в них криптографических функций.**

3. Криптопровайдеры.

Шифрование документа Microsoft Word посредством вызова и обращения к КП



Основная начальные публикации по теме:

1. А.Щербаков, А.Домашев. Прикладная криптография. Использование и синтез криптографических интерфейсов. – М.: Издательско-торговый дом «Русская редакция», 2003. – 416 с.
2. П.Б. Хореев. Методы и средства защиты информации в компьютерных системах. – М.: Издательский центр «Академия», 2005. – 256 с.
3. А.В.Домашев, В.О.Попов, Д.И.Правиков, И.В.Прокофьев, А.Ю.Щербаков.
   Программирование алгоритмов защиты информации. – М.: 1999. – 235 с.
4. А. Фридман, Л. Кландер, М. Михаэлис, Ч. Шилдт. C/C++: Архив программ. – М.: Издательство Бином, 2001. – 640 с.

КРИПТОПРОВАЙДЕР (КП) – **программный модуль ОС** - Cryptographic Service Provider (CSP) - библиотека динамической компоновки - в ОС Windows - это **файлы с расширением .DLL**

Назначение КП: 1) для взаимодействия через криптоинтерфейс ПП с защищаемой прикладной программой и

2) выполнения различных функций защиты, таких как: генерация ключей и обмен ими, шифрование/дешифрование данных, хеширование, формирование цифровых подписей и их верификация.

Основную работу по криптопреобразованию выполняет криптопровайдер.

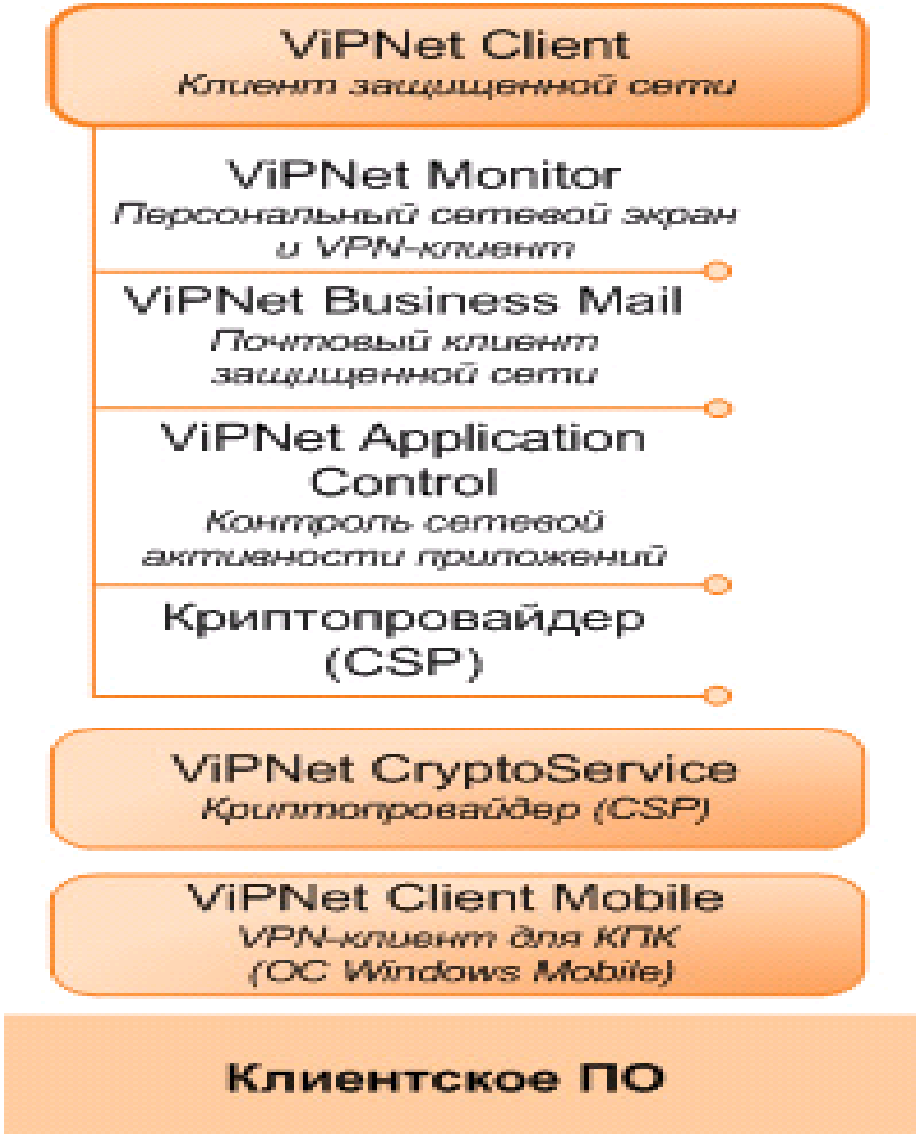
Примеры зарубежных КП

1. **Microsoft Enhanced RSA and AES**
2. **Gemplus**

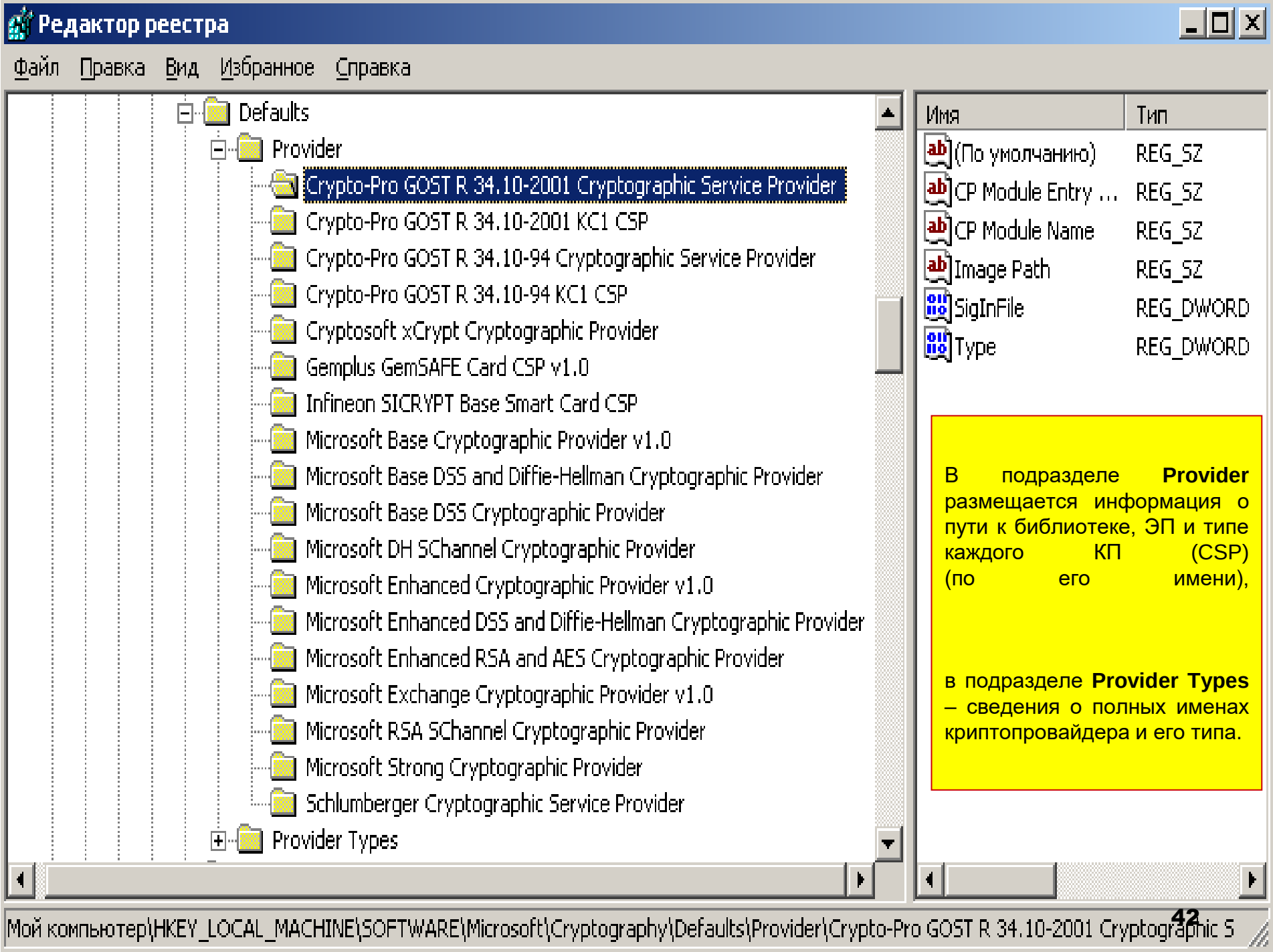
Примеры отечественных КП:

1. Крипто ПРО;
2. Сигнал Ком;
3. Инфотекс;
4. Криптософт.

Криптопровайдеры ViPNet (Инфотекс)




КриптоПро CSP 3.0
Ключевое слово в защите информации
КРИПТО ПРО
English
Установить CSP
Обзор CD
WEB сайт
Наши координаты
Документация
Выход



В подразделе **Provider** размещается информация о пути к библиотеке, ЭП и типе каждого КП (CSP) (по его имени),

в подразделе **Provider Types** – сведения о полных именах криптопровайдера и его типа.

Характеристика криптопровайдеров в ОС MS Windows

Название	Тип	Имя типа	файл	Поддерживаемые криптоалгоритмы			
				Защита обмена ключами	Шифрование	Хэширование	ЭЦП
Microsoft Base Cryptographic Provider v1.0	1	RSA FULL	**rsaenh.dll**	**RSA**	**RC4, DES, 3DES**	**MD5 SHA**	**RSA**
Microsoft Strong Cryptographic Provider	1	RSA FULL	**rsaenh.dll**				
Microsoft Enhanced Cryptographic Provider v1.0	1	RSA FULL	**rsaenh.dll**				
Microsoft RSA SChannel Cryptographic Provider	12	RSA SChannel	**rsaenh.dll**	**RSA**	**RC4, DES, 3DES**	**MD5 SHA**	**RSA**
Microsoft Exchange Cryptographic Provider v1.0	5	Microsoft Office Outlook/Exchange Cryptographic Service Provider	**exchcsp.dll**	**RSA**	**CAST**	**MD5**	**RSA**

Центр Защиты Ресурсов Информационных Систем

ЦЕЗАРИС

Инфраструктура открытых ключей

Руководство программиста

Редакция 1.0

АННОТАЦИЯ

Данный документ содержит методику использования криптопровайдера ***"CESARIS Cryptographic Service Provider ©"***, который входит в состав проекта **Центр Защиты Ресурсов Информационных Систем (ЦЕЗАРИС)**, для встраивания инструментов цифровой подписи в разрабатываемые приложения с применением ***Microsoft Cryptographic Application Programming Interface (CryptoAPI).*** Приведены примеры.

Документ предназначен для прикладных программистов, разрабатывающих собственные приложения.

СОДЕРЖАНИЕ

Криптографический сервис-провайдер
Java LirPKCS11
Руководство программиста

ООО "ЛИССИ-Крипто"

Платформа Java определяет набор программных интерфейсов для выполнения криптографических операций. Эти интерфейсы известны как Java Cryptography Architecture (JCA) и Java Cryptography Extension (JCE).

Стандарт интерфейса криптографических токенов, PKCS#11, был создан RSA Security и определяет программные интерфейсы для криптографических токенов. Для обеспечения интеграции родных токенов PKCS#11, поддерживающих российские криптографические алгоритмы, на платформе Java разработан криптографический сервис-провайдер LirPKCS11. Этот новый провайдер позволяет существующим приложениям, написанным для JCA и JCE API, обращаться к токенам PKCS#11. Не требуется никаких модификаций в приложениях. Единственным требованием является соответствующая конфигурация провайдера в среде выполнения Java.

Основные требования к КП

- В ОС должны содержаться определенные записи для регистрации криптопровайдера. Для MS Windows – записи регистрируются в реестре Windows
- Криптопровайдер должен быть подписан ЭЦП. Для MS Windows – ЭЦП формирует Microsoft.
- Криптопровайдер должен экспортировать обязательный набор функций – большее число которых – функции защиты. В частности для CAPI их количество 23.

Основные параметры КП :

- алгоритм генерации сеансового ключа из хеш-значения;
- длины сеансовых ключей (в зависимости от алгоритма);
- формат блока сеансового ключа при его экспорте из КП;
- поддерживаемый алгоритм обмена сеансовыми ключами симметричного шифрования;
- поддерживаемые алгоритмы симметричного шифрования (конкретный КП может поддерживать только часть таких алгоритмов, определенных для соответствующего типа криптопровайдера);
- режимы симметричного шифрования, принятые по умолчанию (например, режим CBC);
- длина ключей несимметричного шифрования;
- обязательно поддерживаемый алгоритм ЭЦП;
- формат ЭЦП;
- форматы блоков экспорта из КП ключей несимметричного шифрования.

Четыре основные группы функций криптопровайдера

1) **Функции обеспечения функционирования криптопровайдера.**
Включают создание и управление криптопровайдерами, включая настройку их операций

(***CSPAcquireContext, CSPGetProvParam***)

2) **Функции управления и передачи ключевой информации**.
конфигурирование и уничтожение криптографических ключей. Обеспечивают создание , а также обмен ключами с другими пользователями

(***CSPGenKey, CSPDeriveKey***)

3) **Функции шифрования данных.**
Обеспечивают операции шифрования и расшифрования

(***CSPEncrypt, CSPDecrypt***)

4) **Функции хеширования и электронной цифровой подписи**.
Обеспечивают вычисление значения функции хеширования от сообщения, а также формирование и проверку цифровой подписи сообщения

(***CSPHashData, CSPSignHash***)

Типы блобов

1) **PUBLICKEYBLOB** – для передачи (экспорта) открытого ключа. Это общедоступные ключевые блобы; Используется для экспорта открытых ключей ключевых пар подписи или обмена ключами. PUBLICKEYBLOBEX – используется для обмена значениями p, g, и Y=(g^X) mod p в обмене ключами по алгоритму Диффи-Хеллмана.

2) **PRIVATEKEYBLOB** – для передачи (экспорта) пары ключей (открытого и секретного ключа). Это секретные (защищенные) ключевые блобы;

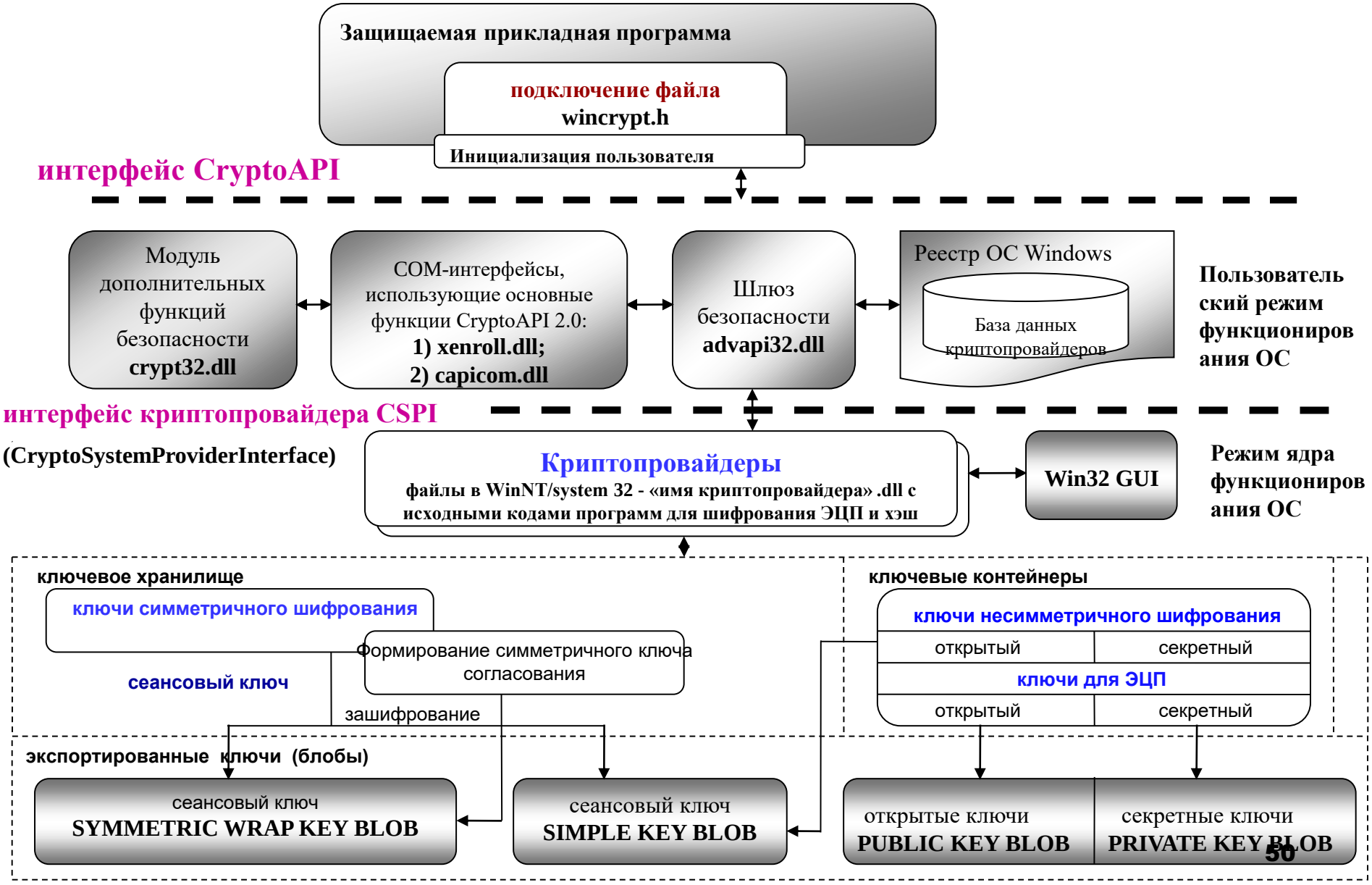
3) **SIMPLEKEYBLOB** – для защищенной передачи сеансового ключа симметричного шифрования (шифрование выполняется на ОК получателя, как правило, с применением RSA); Используется для экспорта сессионных ключей.

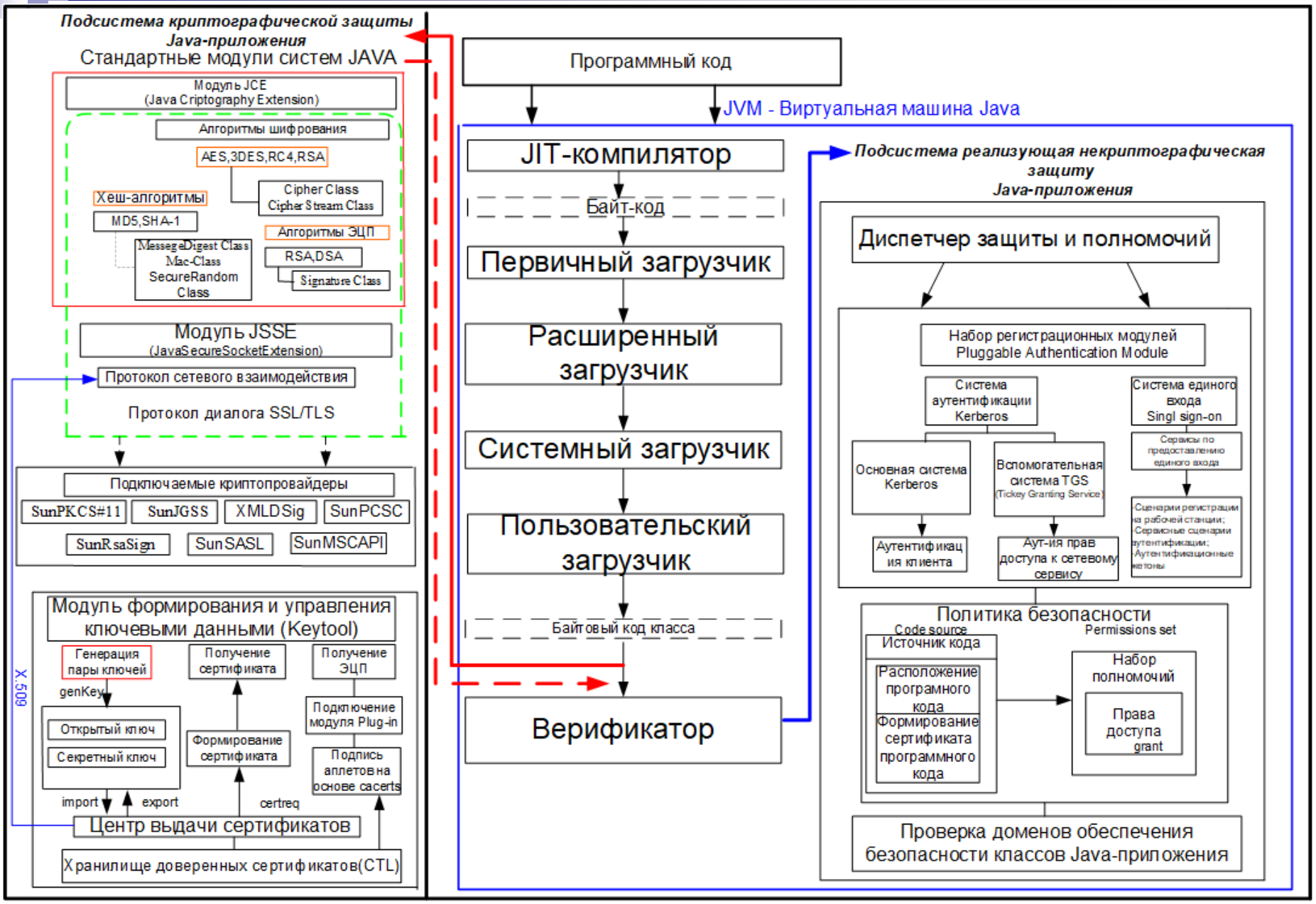
4) **SYMMETRICWRAPKEYBLOB** – для экспортирования ключей при обмене ключами для симметричных алгоритмов.

Типы ключей

Типы ключей	Назначение ключей	Ключевой контейнер/ имя контейнера/имя переменной	Блок импорта – имя ключевого блоба
Ключ обмена – ключевая пара обмена	Для защиты обмена ключами	Exchange/ AT_KEYEXCHANGE/ hXchgKey	**PUBLIC KEY BLOB**
Сеансовый секретный ключ	Для шифрования	Symmetric/-/hKey	
			SIMPLE KEY BLOB
			SYMMETRIC WRAP KEY BLOB
Ключ подписи – ключевая пара подписи	Для ЭЦП	Sign/AT_SIGNATURE/ hPubKey и hPrivateKey	**PUBLIC KEY BLOB**
			PRIVATE KEY BLOB

Криптопровайдеры в ОС Windows



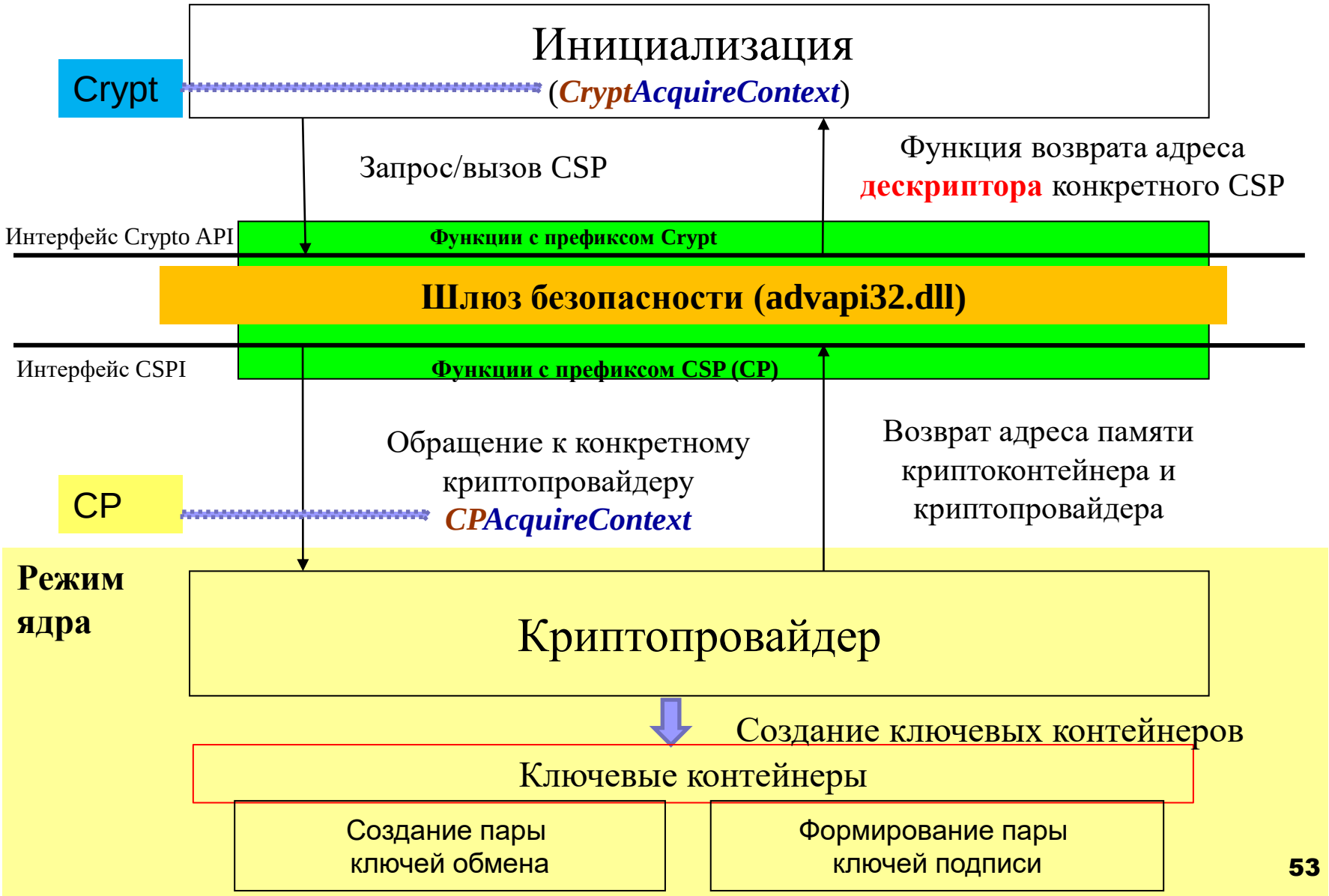

Подсистема криптографической защиты Java-приложения
Стандартные модули систем JAVA
Модуль JCE (Java Criptography Extension)
Алгоритмы шифрования
AES,3DES,RC4,RSA
Cipher Class
Cipher Stream Class
Хеш-алгоритмы
MD5,SHA-1
Алгоритмы ЭЦП
RSA,DSA
MessegeDigest Class
Mac-Class
SecureRandom Class
Signature Class
Модуль JSSE (JavaSecureSocketExtension)
Протокол сетевого взаимодействия
Протокол диалога SSL/TLS
Подключаемые криптопровайдеры
SunPKCS#11
SunJGSS
XMLDSig
SunPCSC
SunRsaSign
SunSASL
SunMSCAPI
Модуль формирования и управления ключевыми данными (Keytool)
Генерация пары ключей
genKey
Получение сертификата
Получение ЭЦП
Открытый ключ
Секретный ключ
Формирование сертификата
Подключение модуля Plug-in
Подпись аплетовна основе cacerts
import
export
certreq
X.509
Центр выдачи сертификатов
Хранилище доверенных сертификатов(CTL)
Программный код
JVM - Виртуальная машина Java
JIT-компилятор
Байт-код
Первичный загрузчик
Расширенный загрузчик
Системный загрузчик
Пользовательский загрузчик
Байтовый код класса
Верификатор
Подсистема реализующая некриптографическая защиту Java-приложения
Диспетчер защиты и полномочий
Набор регистрационных модулей Pluggable Authentication Module
Система аутентификации Kerberos
Система единого входа Singl sign-on
Основная система Kerberos
Вспомогательная система TGS (Tickey Granting Service)
Сервисы по предоставлению единого входа
Сценарии регистрации на рабочей станции; Сервисные сценарии аутентификации; Аутентификационные ключи
Аутентификация клиента
Аут-ия прав доступа к сетевому сервису
Политика безопасности
Code source
Permissions set
Источник кода
Расположение программного кода
Формирование сертификата программного кода
Набор полномочий
Права доступа grant
Проверка доменов обеспечения безопасности классов Java-приложения

КОНТЕКСТ (**context**) представляет собой установленный сеанс взаимодействия между защищаемой прикладной программой-клиентом и криптопровайдером-сервером.

ДЕСКРИПТОР (handle) – числовой идентификатор ресурса, который открывается и обрабатывается операционной системой.

Отличие дескриптора от указателя:
дескриптор является просто числом, по которому операционная система идентифицирует объект данного типа,
указатель содержит действительный адрес объекта в оперативной памяти.
Дескрипторы позволяют реализовать доступ из ПП к ресурсам не напрямую, а через операционную систему.

Формирование контекста



Выводы по вопросу 3:

1. КП позволяет прикладной программе применять криптографические функции над ее данными без получения непосредственного доступа к ключевым данным.

2. Каждый КП имеет свою ассоциированную базу данных для ключевых контейнеров, которая содержит все секретные личные и открытые ключи всех пользователей, имеющих доступ к ПЭВМ.

3. Без ключевой базы, а так же без инициализации конкретного ключевого контейнера, любые вызовы CryptoAPI приняты не будут.

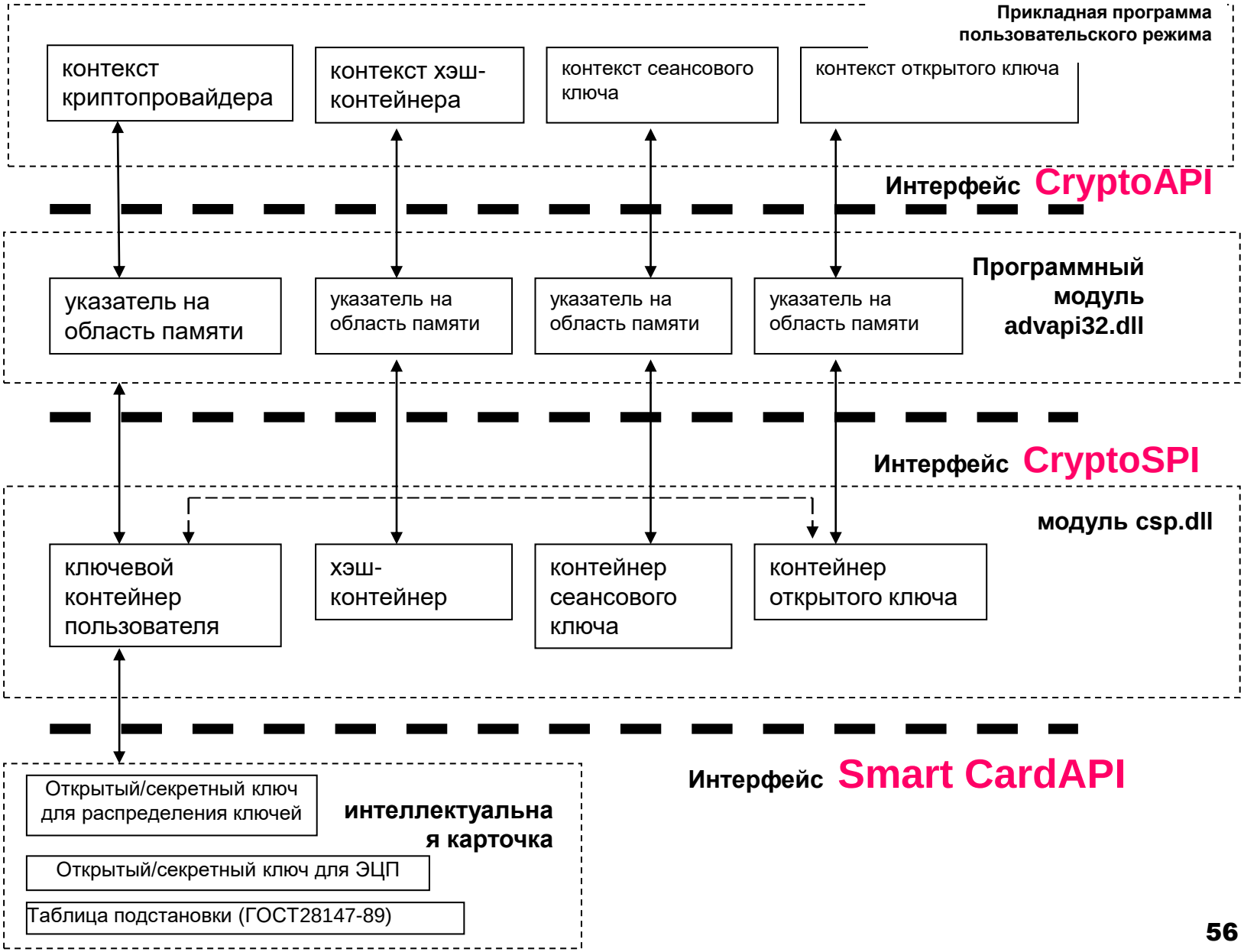
4. Для взаимодействия защищенных прикладных программ друг с другом должен использоваться одноименный КП.

Тема: Средства криптографической защиты информации

Учебные вопросы:

1. Типы, варианты и классы СКЗИ.
2. Архитектура и криптоинтерфейсы программных СКЗИ.
3. Криптопровайдеры.

Функциональная зависимость контекстов и структур



ИНИЦИАЛИЗАЦИЯ: Пример вызова функции CryptAcquireContext

BOOL WINAPI CryptAcquireContext
(HCRYPTPROV *phProv,
LCTSTR pszContainer, LPCTSTR pszProvider,
DWORD dwProvType, DWORD dwFlags)

Параметр	Значение
phProv	Указатель на дескриптор криптопровайдера (CSP)
pszContainer	Имя ключевого контейнера
pszProvider	Указатель на строку, содержащую имя криптопровайдера
dwProvType	Значение типа запрашиваемого криптопровайдера
dwFlags	Флаги

Описание функции шифрования

BOOL *CryptEncrypt*

(HCRYPTKEY *hKey*, HCRYPTHASH *hHash*,
BOOL *Final*, DWORD *dwFlags*, BYTE **pbData*,
DWORD **pdwDataLen*, DWORD *dwBufLen*)

Параметр	Значение
hKey	Дескриптор ключа сессии, используемого для зашифрования данных
hHash	Дескриптор объекта функции хеширования
Final	Булева величина. Определяет, является ли переданный функции блок последним зашифрованным блоком данных.
dwFlags	Флаги
pbData	Буфер, содержащий данные для за(рас) -шифрования
pdwDataLen	Адрес длины входных (выходных) данных
dwBufLen	Размер буфера *pbData* (в байтах)