Development of the Software Cryptographic Service Provider on the Basis of National Standards

Rakhmatillo Djuraevich Aloev[1], Mirkhon Mukhammadovich Nurullaev[2]

[1]*Department of Mathematical modeling & cryptanalysis, National University of Uzbekistan named M. Ulugbek. Tashkent, 100000, Uzbekistan*

[2]*Department of Information communication technology, Bukhara Engineering Technological Institute Bukhara, 201100, Uzbekistan*

[1]aloevr@mail.ru, [2]mirxon@mail.ru

Abstract

The article provides a brief description of the cryptography service provider software developed by the authors of this article, which is designed to create encryption keys, private and public keys of electronic digital signature, create and confirm authenticity of digital signature, hashing, encrypting and simulating data using the algorithms described in State Standards Of Uzbekistan. It can be used in telecommunications networks, public information systems, government corporate information systems by embedding into application applications that store, process and transmit information that does not contain information related to state secrets, as well as in the exchange of information and ensuring the legal significance of electronic documents.

The cryptography service provider includes the following functional components: a dynamically loadable library that implements a biophysical random number sensor; dynamic library that implements cryptographic algorithms in accordance with the State Standards of Uzbekistan; module supporting the work with external devices; installation module that provides the installation of cryptography service provider in the appropriate environment of operation (environment).

Keywords: *tools of cryptographic protection of information, data encryption algorithm, cryptographic provider, hash function, encryption key.*

1. Introduction

The cryptography service provider (CSP) provides the creation of private and public electronic digital signature keys and encryption keys; creation and confirmation of authenticity of electronic digital signature according to the algorithms described in ("GOST R 34.11-94," 1994; "O`z DSt 1106," 2009); The formation of derived encryption keys used by data encryption algorithms described in ("PKCS #5 v2.0," 1999, March 25; "Expansion of PKCS #11 for the use of Russian cryptographic algorithms," 2008); Work with key information

stored on external media; hashing of memory areas and other data according to the algorithms described in ("GOST 28147-89," 1989; "O`z DSt 1092," 2009); Encryption of memory areas and other data in accordance with the data encryption algorithms described in ("PKCS #5 v2.0," 1999, March 25; "Expansion of PKCS #11 for the use of Russian cryptographic algorithms," 2008).
The cryptography service provider provides support for identifiers of algorithms and parameters for compatibility with third-party crypto-providers in terms of the ability to work with public-key certificates issued by third-party registration centers, provided they use the cryptographic algorithms described in ("GOST 28147-89," 1989; "GOST R 34.11-94," 1994; "PKCS #5 v2.0," 1999, March 25; "Expansion of PKCS #11 for the use of Russian cryptographic algorithms," 2008; "O`z DSt 1106," 2009; "O`z DSt 1092," 2009). The cryptography service provider provides the ability to work with digital certificates of public keys, which are structured binary in ASN.1 format, conforming to ITU-T X.509 v.3 standard and IETF RFC 5280, RFC 3739 Recommendations. The cryptography service provider provides work with external key carriers such as USB-flash, eToken Aladdin (eToken PRO 72K (JAVA)). As part of CIPF[1] – CSP, modules are provided that provide for calling cryptographic functions through the Microsoft CryptoAPI 2.0 interface when running under Microsoft operating systems.

In accordance with the functional purpose of a cryptography service provider, it generates public and private electronic digital signature keys, hash keys, functional keys and encryption keys for use, respectively, in the algorithms described in ("GOST 28147-89," 1989; "GOST R 34.11-94," 1994; "PKCS #5 v2.0," 1999, March 25; "Expansion of PKCS #11 for the use of Russian cryptographic algorithms," 2008; "O`z DSt 1106," 2009; "O`z DSt 1092," 2009). In applications where the cryptography service provider will be integrated, an appropriate key manufacturing and distribution system is provided, which will be based on the key generation functions of the cryptography service provider. Cryptography service provider allows the use of a multi-level key protection model using random and derivative keys of key protection. Protection of keys is carried out on the basis of cryptographic transformations in accordance with the PKCS #5 standard ("RFC 4357," 2006) using the State Standards of Uzbekistan ("O`z DSt 1092," 2009), ("PKCS #5 v2.0," 1999, March 25) or the interstate standard ("Expansion of PKCS #11 for the use of Russian cryptographic algorithms," 2008). A cryptography service provider provides storage of key information on a hard magnetic disk drive (HDD) and/or external key storage devices, such as USB-flash, eToken Aladdin (eToken PRO 72K (JAVA)). A cryptography service provider provides work with key containers containing: signature keys, encryption keys and additional information necessary to ensure the cryptographic protection of keys and ensure control of their integrity. To protect key information from substitution and/or distortion during its storage on HDD and external key carriers, as well as during distribution, key information is supplied with a checksum. In order to ensure the safe use of the cryptography service provider installed on a PC, organizational measures are provided, as well as software and hardware methods and means of protecting information are used to ensure the secrecy of secret keys located in the PC's memory during the operation of cryptography service provider, as well as service cryptography service provider parameters stored on the hard disk. Cryptography service provider contains a

[1] CIPF - Cryptographic Information Protection Facility

component that allows you to verify the operability of the cryptographic algorithms implemented in it. Functioning is carried out on the basis of test examples. To ensure the safe use of an application with a built-in ICS-CSP, mechanisms are provided to control the integrity of cryptography service provider libraries. In cryptography service provider, a biophysical sensor of random numbers is used to generate random binary sequences, which implements the mechanism for generating secret digital signature keys, encryption keys, initialization vectors using various algorithms.

The paper briefly describes the software necessary for the operation of the tools of cryptographic protection of information (TCPI) - data encryption cryptography service provider which is developed by the authors based on the national standards. Furthermore, here is given the purpose, the capabilities of the TCPI - CSP software and its main characteristics, the limitation of the using area of this software.

One of the following operating systems is necessary in order to perform the TCPI - CSP: Microsoft Windows XP (32 bit) Professional SP3; Microsoft Windows Vista (32 bit) Ultimate SP2; Microsoft Windows 7 (32 bit) Ultimate; Microsoft Windows Server 2003 (32 bit) Enterprise Edition R2 SP2; Microsoft Windows Server 2008 (32 bit) Enterprise Edition SP2.

The original programming languages for the TCPI - CSP are C and C ++.

TCPI - CSP performs the following main functions:

- generation of encryption keys for the data encryption algorithm O`z DSt[2] 1105: 2009 ("O`z DSt 1105," 2009) and the algorithm GOST[3] 28147-89 ("GOST 28147-89," 1989);
- encryption of RAM[4] areas and other data in accordance with the data encryption algorithm O`z DSt 1105: 2009 ("O`z DSt 1105," 2009) and the algorithm GOST 28147-89 ("GOST 28147-89," 1989);
- generation of keys for implementing and verifying the electronic digital signature (EDS) using algorithms 1 and 2 of O`z DSt 1092: 2009 ("O`z DSt 1092," 2009) and the algorithm of GOST R 34.10-2001 ("GOST R 34.10-2001," 2001);
- hashing of memory areas and other data according to algorithm 1 with the parameter p = 256 O`z DSt 1106: 2009 ("O`z DSt 1106," 2009) and the algorithm GOST 34.11-94 ("GOST R 34.11-94," 1994);
- formation and verification of the EDS result in accordance with O`z DSt 1092: 2009 ("O`z DSt 1092," 2009) algorithms 1 and 2 and GOST R 34.10-2001 ("GOST R 34.10-2001," 2001);

work with key information stored on external media.

TCPI - CSP can be used as a default crypto-provider for the Windows operating system. TCPI - CSP supports the cryptographic algorithms of the Republic of Uzbekistan and Russia as well as some of the common cryptographic algorithms used in Windows OS, such as RSA, 3DES, SHA-1 etc.

[2] O`z DSt - State standard of Uzbekistan

[3] GOST - Interstate standard

[4] RAM - Random Access Memory

2. Functions of working with key information

TCPI - CSP product works with key information in a key container - storages (Key Container).
Since TCPI - CSP built in accordance with Microsoft technology, the container contains the following keys:

- ***AT_KEYEXCHANGE*** key used to encrypt and exchange session keys;
- ***AT_SIGNATURE*** - keys used to create and verify a digital signature.

Note. Private keys (encryption keys and secret signature keys) contained in the container are protected using a security key, which is derived from the value of the user's PIN-code of the token.
Cryptographic procedures are invoked in TCPI - CSP using the PKCS #11 interface.

The underlying concepts of the PKCS #11 interface are slot and token. The token is a repository of some personal information (various keys, certificates, private data, etc.), and the slot acts as a link between a computer and a token that allows different tokens to be connected at different times.
For both ***AT_KEYEXCHANGE*** and ***AT_SIGNATURE***, the same and different PKCS #11 slots can be used.

TCPI - CSP supports work with containers located both on the hard disk of the computer and on removable media such as Flash Memory and Smart Card.
Each container has a unique name consisting of a prefix or several prefixes and the name itself. Prefixes in the container name are separated from each other by the "\" symbol. The container name can contain from zero to three prefixes:

Container Name = [pref1 \] [pref2 \] [pref3 \] Name

The location of the media is determined by the first prefix in the container name, depending on the presence of the ***CRYPT_MACHINE_KEYSET*** flag in the ***CPAcquireContext*** function.

In the container name, the second prefix is a reference to the slot for ***AT_KEYEXCHANGE***, and the third is for ***AT_SIGNATURE***. If the third prefix is absent, then ***AT_KEYEXCHANGE*** and ***AT_SIGNATURE*** are stored in the same slot.

Protection of token's private objects is carried out using the PKCS #5 cryptographic interface. This algorithm solves two problems at once: encrypting private data and protecting it from accidental or intentional distortion.

3. Encryption function

The encryption function is a cryptographic algorithm, which is a bijective mapping from a finite set of plaintext to a finite set of encrypted texts, in which the mapping function depends on a secret parameter called a key.
The encryption function uses to encrypt and decrypt information.
The encryption function in accordance with ("O`z DSt 1105," 2009) can use cryptographic keys of length **256** or **512 bits** for encrypting and decrypting data blocks of length **256 bits**.

The encryption function is used for cryptographic protection of data storing and transmitting in computer networks, telecommunications, in separate computing systems or in computers of enterprises, organizations, and institutions.

In symmetric cryptosystems, data exchange takes place in three stages:

1) the sender of the message sends the encryption key (or/and functional key) to the recipient via a secure channel that is known only to them;
2) the sender, using the encryption key and the function key, converts the original data into encrypted data and sends them to the recipient via the communication channel;
3) the recipient, having received the encrypted data, decrypts it with the help of an encryption key and a function key. Both sides may use these keys several times.

In addition to data protection, the encryption function can be used to protect the symmetric keys themselves as they are transmitted over unprotected communication channels. In this case, the transmitted symmetric key is encrypted with some other key, called the security key.

The encryption function ("O`z DSt 1105," 2009) contains two modes:

- electronic codebook mode (ECM);
- block chaining mode (BCM).

The electronic codebook mode is an encryption mode in which all plaintext blocks are encrypted independently of one another on the same key, in accordance with the data encryption algorithm.

ECM mode is usually used when encrypting symmetric keys.

The block chaining mode of encryption is a mode in which each encrypted (decrypted) depends on the previous block of an encrypted (decrypted) block. For the first block, the initialization vector is used as the previous block. If the last block of text is not complete, it is supplemented to the required length. This procedure is called padding. BCM mode is usually used when encrypting data.

The purpose of these functions, functionality of operation algorithms are presented in the document "O`z DSt 1105: 2009. State standard of Uzbekistan. Information technology. CRYPTOGRAPHIC PROTECTION OF INFORMATION. Data encryption algorithm.

4. Hash function

The hash function is designed to implement a unidirectional compressing mapping **f** from the set **A** to the set **B**, the input of which is a message of arbitrary length **M**, and the output is a string of fixed length **h(M)**. Using a hashing transform allows you to reduce the input text redundancy.

The hashing function is used in cryptographic methods for processing and protecting information, including for the implementation of electronic digital signature procedures (hereinafter referred to as EDS[5]) when transferring, processing and storing information in automated systems.

Basic requirements for a cryptographic hash function:

- the input of the function must be a message of any length;

[5] EDS - electronic digital signature

- at the output of the function, a message of fixed length is obtained;
- the hash function is simply calculated for any message;
- hash function - unidirectional function;
- knowing the message **M**, it is almost impossible to find another message **M'** for which **h(M) = h(M')**.

In the TCPI - CSP hash function, the output sequence and the hash key have fixed lengths of **256 bits**.

The composition and purpose of this function, functionality, and functioning algorithm are presented in the document "O`z DSt 1106: 2009. State standard of Uzbekistan. Information technology. CRYPTOGRAPHIC PROTECTION OF INFORMATION. A hash function".

5. Signature function

The electronic digital signature function is used to generate and confirm the authenticity of an electronic digital signature (EDS) under a given message (electronic document) transmitted over unprotected public telecommunications channels.

Upon receipt of the message, the recipient can verify the integrity of the message transmitted by the sender and verify the authenticity of the sender's authorship.

EDS is an electronic analog of a written signature and therefore an EDS can be used by the recipient or a third party to verify that the message was actually signed by the sender.

To describe the formation and confirmation of the authenticity of a digital signature, two algorithms are used (Algorithm 1, Algorithm 2)[6]. Algorithm 1 is considered in two basic modes:

- without session key[7];
- with a session key.

Algorithm 2 is used in the classical (without session key) mode. Algorithm 1 provides a backup path for detecting a fake digital signature by introducing a session key procedure used in the process of authenticating the authenticity of a digital signature to the EDS generation process.

The composition and purpose of this function, functionality, and functioning algorithm are presented in the document "O`z DSt 1092: 2009. State standard of Uzbekistan. Information technology. CRYPTOGRAPHIC PROTECTION OF INFORMATION. Processes of formation and verification of electronic digital signature".

[6] Algorithm 1, Algorithm 2 - A description of these algorithms is given in the document "O`z DSt 1092: 2009. State standard of Uzbekistan. Information technology CRYPTOGRAPHIC PROTECTION OF INFORMATION. Processes of formation and verification of electronic digital signature".

[7] - a session key is a single-use symmetric key used for encrypting all messages in one communication session.

Functional restrictions on the use of TCPI – CSP: Cryptographic interface TCPI - CSP is implemented in accordance with the CSP standard, which is applicable only in the Windows operating system and is not applicable in other operating systems such as Linux, Mac, Unix, etc.

Since cryptographic algorithms of the Republic of Uzbekistan are implemented in TCPI - CSP, which are not recognized by standard Windows tools, for embedding TCPI in typical Windows applications (MS Outlook, Internet Explorer, VPN, etc.) requires changes to the standard OS software (advapi32.dll, cryptsp.dll, crypt32.dll, inetcomm.dll, schannel.dll, secur32.dll, mailcomm.dll, etc.). Making such changes to Windows can be done in various ways.

6. Description of the logical structure

The used algorithms and methods: When implementing cryptographic algorithms of the Republic of Uzbekistan, including the implementation of operations with big numbers, the source texts of programs from the OpenSSL[8] the software package was used.

When implementing cryptographic algorithms of the Republic of Uzbekistan, all simple numbers that are part of the parameters of the algorithms are checked for simplicity using the standard procedures contained in the package OpenSSL.

Data Encryption Algorithm: Data encryption in TCPI - CSP supports both cryptographic algorithms of the Republic of Uzbekistan ("O`z DSt 1105," 2009) and cryptographic standards of the Russian Federation (GOST 28147-89) ("GOST 28147-89," 1989).

Data encryption: Cryptographic provider TCPI - CSP supports various algorithms for symmetric data encryption (SDE), including the data encryption algorithm of the Republic of Uzbekistan ("O`z DSt 1105," 2009). According to section 6.4 ("O`z DSt 1105," 2009), an SDE in TCPI - CSP is implemented in three different ways:

- DEA[9] with key 256-bit;
- DEA with key 512-bit;
- DEA with the function key update.

Here is the implementation of all three of these methods using the TCPI - CSP.

DEA with key 256-bit*:* To implement a DEA with a key of 256 bits, you need:

1) get the key for the algorithm **CALG_SYMM**[10]. This key can be obtained in the following ways, in which **Algid = CALG_SYMM**:
   - CryptGenKey,
   - CryptDeriveKey,

   Either through CryptImportKey from **SIMPLEBLOB**[11] or **SYMMETRICWRAPKEYBLOB**[12], created previously via CryptExportKey;

[8] OpenSSL - OpenSSL is a robust, commercial-grade, and full-featured toolkit for the Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols. It is also a general-purpose cryptography library.
[9] DEA - Data Encryption Algorithm
[10] ***CALG_SYMM*** - The description of this parameter is given in the document ("O`z DSt 1105," 2009)
[11] ***SIMPLEBLOB*** - The description of this parameter is given in the document ("O`z DSt 1105," 2009)

2) call function CryptEncrypt or CryptDecrypt depending on the operation performed.

DEA with key 512-bit: To implement a DEA with a key of 512 bits, you need:
1) get the key of the DEA algorithm with the key 256 (in accordance with paragraph 1 of the DEA with the key of 256 bits);
2) perform a function CryptSetKeyParam with parameter **KP_FUNC_KEY** (#define KP_FUNC_KEY 200). The value **pbData** will be the value of the function key that can be generated, in particular, using the function CryptGenRandom or by other means;
3) perform a function CryptEncrypt or CryptDecrypt depending on the operation performed.

DEA with the function key update: To implement a DEA with a function key update, you need:
1) get the key of the DEA algorithm with the key 256 (in accordance with paragraph 1 of the DEA with the key of 256 bits);
2) if necessary, install a function key (see p. 2 of the DEA with a key of 512 bits);
3) perform a function CryptSetKeyParam with parameter **KP_OID** (#define KP_OID 102). As a value for **pbData** fed to the input value of the function **szOID_SYMM_B**[13];
4) call function CryptEncrypt or CryptDecrypt depending on the performed operation.

In the encryption function, the length of the input and output blocks, as well as the length of the elements of the array **Holat**[14] are equal **256 bit**. The length of the encryption key and the function key are also equal **256 bit**. The number of steps for the encryption feature is set **e**=8.

In both the encryption mode and the decryption mode, the algorithm uses a one-time conversion - forming an array of the session key and the next four bytes - oriented and one bit - oriented conversion at each stage. These transformations include:
- forming arrays of step keys;
- mixing data based on the session key array;
- cyclic shifts of rows and columns of the array **Holat** for various values of displacement;
- byte-wise replacement of bytes of the **Holat** array based on linear array arrays;
- addition operation modulo 2 **Holat** arrays and an array of the stage key $\mathbf{K_e}$;
- cyclic shifts of a linear array of a session-stage key by the same value of bits at each stage.

When encrypting a cryptographic module is initialized, the encryption key **k** and functional key $\mathbf{k_f}$, number of stages **e**, and also initialization vector **IV** for mode **m=ShBil** is first loaded into the cryptographic module. Also, when encrypting a cryptographic module into the **Holat** array, the plaintext is loaded; when decrypted, the ciphertext is loaded. At the beginning of the encryption

[12] ***SYMMETRICWRAPKEYBLOB*** - The description of this parameter is given in the document ("O`z DSt 1105," 2009)
[13] ***szOID_SYMM_B*** - The description of this parameter is given in the document ("O`z DSt 1105," 2009)
[14] ***Holat*** - array containing one block of information

procedure, **ShaklSeansKalitBayt(k,k_f)**[15], **ShaklSeansKalit(K_{st})**[16] and **ShaklBosqichKalit(k_{se})** crypto-transformations are initialized. At the outputs of crypto transformations **ShaklSeansKalitBayt(k,k_f), ShaklSeansKalit(K_{st})**, byte-level arrays of substitutions and a session key consisting of diamatrix parts are formed at the byte level. These arrays are used in the following sessions as long as **k, k_f** remain constant. At the output of the crypto-transformation **ShaklBosqichKalit(k_{se})** the initial key and the set of stage keys formed for each stage are formed.

Encryption of symmetric keys using symmetric keys: To encrypt symmetric keys using symmetric keys, an DEA is used in ECM mode.

To produce imitation protection, a hash function from cryptographic standards of Uzbekistan in 256 bit mode is used. The first 4 bytes of the key's hash function is the simulated prefix for the encrypted key.

Encryption of symmetric keys using asymmetric keys: When encrypting symmetric keys using asymmetric keys, the following algorithm is used.

Using the Diffie-Hellman method, a common key of the form $\mathbf{a^{xy}}$ is formed, where **x, y** are private keys, the common key size is 256 and 64 bytes for algorithms 1 and 2, respectively. This shared key is used instead of a password to generate a shared symmetric key and initialization vector in accordance with PKCS #5. The original symmetric key is encrypted in BCM mode using a common symmetric key and an initialization vector.

The size of the encrypted key is 64 bytes. Imitation protection provides padding size of 32 bytes.

An algorithm of generation and verification of EDS: Processes of formation and verification of electronic digital signatures in TCPI-CSP supports both cryptographic algorithms of the Republic of Uzbekistan ("O`z DSt 1092," 2009) and the cryptographic standard of the Russian Federation (GOST R 34.10-2001) ("GOST R 34.10-2001," 2001).

The function of EDS (formation and verification) is designed to ensure the reliability of the transmitted and received information and confirm its authorship.

Program structure: This subsection presents the general structure of the TCPI - CSP, as well as a description of the functions of each module of the system.

General structure: TCPI - CSP is implemented in the form of the following dynamic libraries:

- **CSP.DLL** - loading the CSP interface using the Crypto API;
- **CSPFUNC.DLL** - loading the CSP interface directly;
- **PKCS11.DLL** - loading the PKCS #11[17] interface (PKCS #11 interface for TCPI - CSP, hereinafter PKCS #11);

[15] ***ShaklSeansKalitBayt(k,k_f)*** - is a function that is used to generate the key for each session and to perform the BaytAlmash() conversion when encrypting and decrypting.

[16] ***ShaklSeansKalit(K_{st})*** - is a function that is used to generate the key for each session and to perform the Aralash() transformation when encrypting and decrypting.

[17] PKCS #11 - is one of the standards of the Public-Key Cryptography Standards (PKCS) family. It defines a platform-independent software interface for accessing cryptographic devices (smartcards, tokens, cryptographic accelerators, key servers and other means of cryptographic information protection).

– **SCTOKEN.DLL** - functions for working with a smart card through the interface PKCS #11;
– **VTOKEN.DLL** - functions for working with virtual slots and tokens through the interface PKCS #11;
– **CRYPTOSP.DLL** - a library of cryptographic procedures.

The above libraries are located in the WINDOWS \ SYSTEM32 folder. Also, the following modules are included in the scope of delivery of TCPI - CSP:

– **GUI.DLL** - interface for entering the password to the key, random number generation using the electronic roulette mechanism;
– **CSP_INTEGRAL_TEST.EXE** integral test for TCPI - CSP;
– **PKCS11INI.EXE** initialization of virtual slots and tokens for the PKCS #11 interface;
– **CRYPTOTEST.EXE** - tests of cryptographic algorithms;
– **KEYMANAGER.EXE** - test program for obtaining and viewing certificates;
– **KM.DLL** - dynamic library for supporting the work of the test program **KEYMANAGER.EXE**;
– **CSPCON.DLL** is a dynamic library for supporting the import and export of private keys in the PFX format.

Note. The **KEYMANAGER.EXE** program and two dynamic libraries supporting it are intended only for testing the operation of CSP with certificates and private keys.

7. CSP interface

The TCPI - CSP interface consists of two dynamic libraries **CSP.DLL, CSPFUNC.DLL**, as well as an auxiliary test program **CSP_INTEGRAL_TEST.EXE**.

The cryptographic interface TCPI - CSP was created in accordance with the requirements of the Microsoft Cryptography Service Provider (CSP) standard. A description of the CSP standard (with a detailed description of all functions) can be found on the Microsoft website (http://msdn.microsoft.com/en-us/library/aa380245(v=VS.85).aspx). The TCPI - CSP interface was developed for the implementation of cryptographic algorithms of the Republic of Uzbekistan ("O`z DSt 1105," 2009; "O`z DSt 1106," 2009; "O`z DSt 1092," 2009), Russian encryption algorithms and electronic signatures ("GOST 28147-89," 1989; "GOST R 34.11-94," 1994; "GOST R 34.10-2001," 2001) using PKCS #11 ("Expansion of PKCS #11 for the use of Russian cryptographic algorithms," 2008) interface.

When developing the software implementation of the cryptographic interface of TCPI - CSP, the source texts of programs from the world - famous and freely distributed software package OpenSSL were widely used.

Using TCPI - CSP can be done directly, by loading the **CSPFUNC.DLL** library using the LoadLibrary mechanism and obtaining addresses of cryptographic functions using the GetProcAddress[18] command, or via the CryptoAPI[19] interface.

[18] GetProcAddress - The GetProcAddress function retrieves the address of the exported function or variable from the specified dynamic link library (DLL).

[19] CryptoAPI - CryptoAPI is an application programming interface that provides Windows developers with a standard set of functions for working with a cryptographic provider. Included in the Microsoft operating

Input and output data: The cryptographic interface of TCPI-CSP was created in accordance with the requirements of the Microsoft Cryptography Service Provider (CSP). A description of the CSP standard can be found on the Microsoft website. A description of the input and output data for the CSP standard can also be found on the Microsoft website at the links below.

CSP connection functions:
Function & Description
CPAcquireContext *– Associates a key container with a CSP pointer.*
CPGetProvParam *– Displays CSP parameters.*
CPReleaseContext *– Frees pointer received by* ***CPAcquireContext****.*
CPSetProvParam *– Sets specific CSP parameters.*

Key generation and CSP key exchange functions:
Function & Description
CPDeriveKey – Creates a key from a password.
CPDestroyKey – Removes a key from memory.
CPDuplicateKey – Creates a copy of the key.
CPExportKey – Export key.
CPGenKey *– Generating a random key.*
CPGenRandom *– Generating random numbers.*
CPGetKeyParam *– Getting key parameters.*
CPGetUserKey *– Getting a user key pointer.*
CPImportKey *– Import key.*
CPSetKeyParam *– Setting key parameters.*

Encryption and decryption functions:
Function & Description
CPDecrypt – Decrypts encrypted text with a key for encryption.
CPEncrypt – Encrypts plaintext with an encryption key.

8. Conclusion

So, the work is devoted to a brief description of the CSP software, which is designed to create encryption keys, private and public keys of an electronic digital signature, creating and authenticating EDS, hashing, encrypting and simulating data using the algorithms described in the State Standards of Uzbekistan. It can be used in telecommunications networks, public information systems, government corporate information systems by embedding into application applications that store, process and transmit information that does not contain information related to

systems. Most CryptoAPI features are supported starting from Windows 2000. CryptoAPI supports asymmetric and symmetric keys, that is, allows you to encrypt and decrypt data, as well as work with electronic certificates. The set of supported cryptographic algorithms depends on the specific cryptographic provider.

state secrets, as well as in the exchange of information and ensuring the legal significance of electronic documents.

CSP includes the following functional components: a dynamically loadable library that implements a biophysical sensor of random numbers; dynamic library that implements cryptographic algorithms in accordance with the State Standards of Uzbekistan; module supporting the work with external devices; installation module that provides the installation of CSP in the appropriate environment of operation (environment).

CSP provides the creation of private and public EDS keys and encryption keys; creation and confirmation of authenticity of EDS according to the algorithms described in ("GOST R 34.11-94," 1994; "O`z DSt 1106," 2009); The formation of derived encryption keys used by data encryption algorithms described in ("PKCS #5 v2.0," 1999, March 25; "Expansion of PKCS #11 for the use of Russian cryptographic algorithms," 2008); Work with key information stored on external media; hashing of memory areas and other data according to the algorithms described in ("GOST 28147-89," 1989; "O`z DSt 1092," 2009); Encryption of memory areas and other data in accordance with the data encryption algorithms described in ("PKCS #5 v2.0," 1999, March 25; "Expansion of PKCS #11 for the use of Russian cryptographic algorithms," 2008).

CSP provides support for identifiers of algorithms and parameters for the implementation of compatibility with third-party cryptographic providers in terms of the ability to work with public-key certificates issued by third-party registration centers, provided they use the cryptographic algorithms described in ("GOST 28147-89," 1989; "GOST R 34.11-94," 1994; "PKCS #5 v2.0," 1999, March 25; "Expansion of PKCS #11 for the use of Russian cryptographic algorithms," 2008; "O`z DSt 1106," 2009; "O`z DSt 1092," 2009). The cryptography service provider provides the ability to work with digital certificates of public keys, which are structured binary in ASN.1 format, conforming to ITU-T X.509 v.3 standard and IETF RFC 5280, RFC 3739 Recommendations. CSP provides work with external key carriers such as USB-flash, eToken Aladdin (eToken PRO 72K (JAVA)). As part of CSP, modules are provided that provide for calling cryptographic functions through the Microsoft CryptoAPI 2.0 interface when running under Microsoft operating systems.

In accordance with the functional purpose, CSP provides the generation of public and private EDS keys, hashing keys, functional keys and encryption keys for use, respectively, in the algorithms described in ("GOST 28147-89," 1989; "GOST R 34.11-94," 1994; "PKCS #5 v2.0," 1999, March 25; "Expansion of PKCS #11 for the use of Russian cryptographic algorithms," 2008; "O`z DSt 1106," 2009; "O`z DSt 1092," 2009). In the application where the CSP will be integrated, an appropriate system for the manufacture and distribution of keys will be provided, which will be based on the functions of generation of CSP keys. CSP allows the use of a multi-level key protection model using random and derivative keys of key protection. Protection of keys is carried out on the basis of cryptographic transformations in accordance with the PKCS #5 standard ("RFC 4357," 2006) using the State Standards of Uzbekistan ("O`z DSt 1092," 2009; "PKCS #5 v2.0," 1999, March 25) or the interstate standard ("Expansion of PKCS #11 for the use of Russian cryptographic algorithms," 2008). CSP provides storage of key information on a hard disk drive (HDD) and/or external key storage devices, such as USB-flash, eToken Aladdin (eToken PRO 72K (JAVA)). CSP provides work with key containers containing: signature keys, encryption keys and

additional information necessary to ensure the cryptographic protection of keys and ensure control of their integrity. To protect key information from substitution and/or distortion during its storage on HDD and external key carriers, as well as during distribution, key information is supplied with a checksum. In order to ensure the safe use of the CSP installed on the PC, organizational measures are provided, as well as software and hardware methods and means of protecting information are used to ensure that secret keys stored in the PC's memory during operation of the CSP are kept secret, as well as the CSP service parameters stored on the hard drive. CSP contains a component that allows you to verify the operation of cryptographic algorithms implemented in it. Functioning is carried out on the basis of test examples. To ensure the safe use of an application with a built-in CSP, mechanisms are provided to control the integrity of the CSP libraries. In CSP, a biophysical random number sensor is used to generate random binary sequences that implement the mechanism for generating secret digital signature keys, encryption keys, initialization vectors using various algorithms.

References

GOST 28147-89: 1989. *Information processing systems. Cryptographic protection. An algorithm of cryptographic transformation.*

GOST R 34.11-94: 1994. *Information technology. Cryptographic protection of information. A Hash function.*

PKCS #5 v2.0 (1999, March 25). *Password-Based Cryptography Standard. RSA Laboratories.* [Electronic resource]. Date of treatment 12.11.2016. Retrieved from http://www.rsa.com/rsalabs/node.asp?id=2127.

GOST R 34.10-2001: 2001. *Information technology. Cryptographic protection of information. Processes of formation and verification of electronic digital signature.*

RFC 4357 (2006). *Additional Cryptographic Algorithms for Use with GOST 28147-89, GOST R 34.10-94, GOST R 34.10-2001, and GOST R 34.11-94 Algorithms.*

Expansion of PKCS #11 for the use of Russian cryptographic algorithms," (2008).

O`z DSt 1105: 2009. *State standard of Uzbekistan. Information technology CRYPTOGRAPHIC PROTECTION OF INFORMATION. Data encryption algorithm.*

O`z DSt 1106: 2009. *State standard of Uzbekistan. Information technology CRYPTOGRAPHIC PROTECTION OF INFORMATION. Hash function.*

O`z DSt 1092: 2009. *State standard of Uzbekistan. Information technology CRYPTOGRAPHIC PROTECTION OF INFORMATION. Processes of formation and verification of electronic digital signature.*