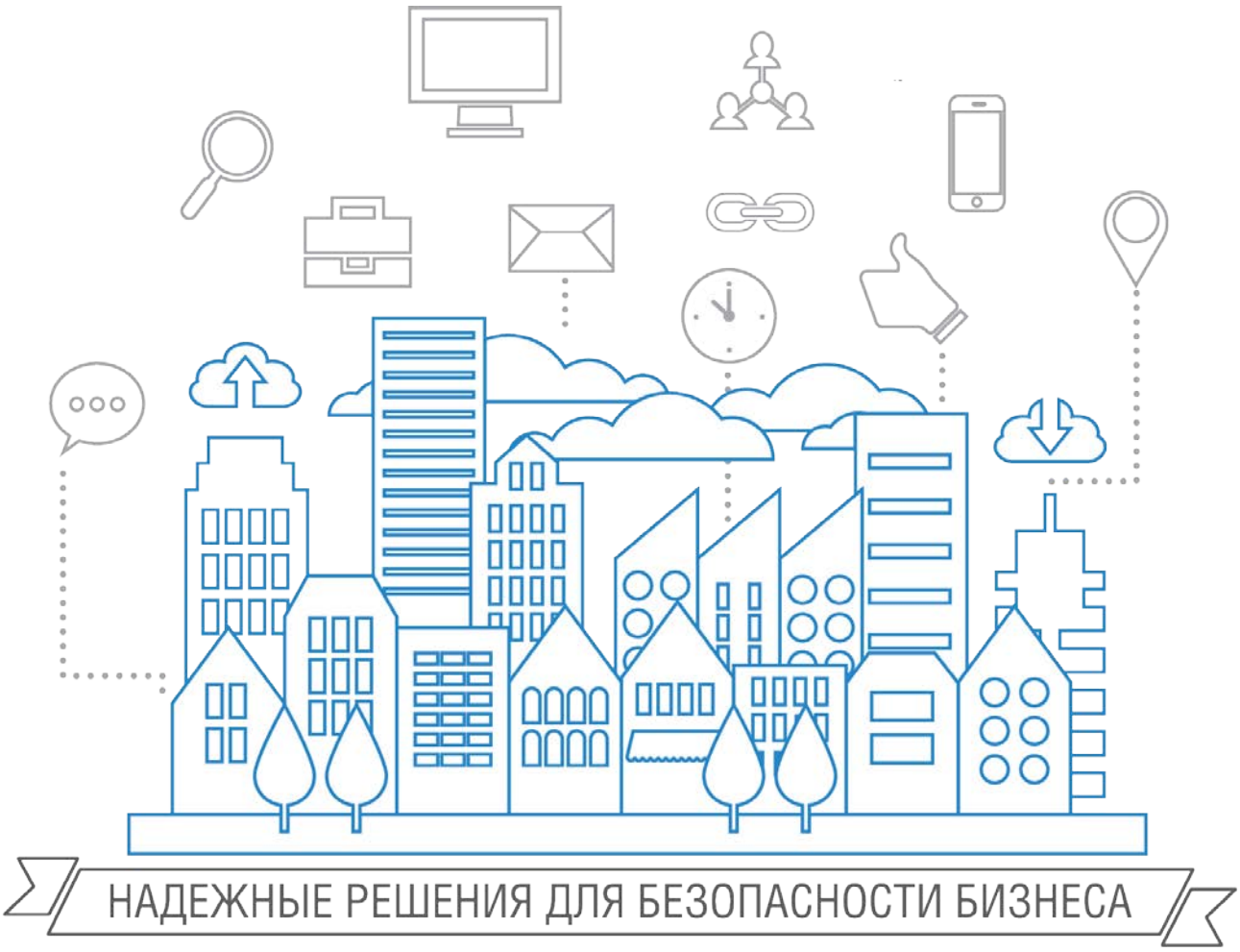
ГАЗИНФОРМСЕРВИС

198096, г. Санкт-Петербург, ул. Кронштадтская, д.10, лит. А, тел.: (812) 677-20-50, факс: (812) 677-20-51
Почтовый адрес: 198096, г. Санкт-Петербург, а/я 59, e-mail: resp@gaz-is.ru, www.gaz-is.ru
р/с 40702810800000001703 Ф-л Банка ГПБ (АО) в г. Санкт-Петербурге БИК 044030827,
к/с 30101810200000000827, ОКПО 72410666, ОГРН 1047833006099, ИНН/КПП 7838017968/783450001

Библиотека «GiSCryptoC»

Описание интерфейсов



Санкт-Петербург, 2017

Аннотация

В документе приводится описание программных интерфейсов библиотеки «GisCryptoC», реализованных на языке C, с помощью которых вызываются основные функции криптографической платформы «Litoria Crypto Platform».

В разделе «Общие сведения» описано назначение библиотеки «GisCryptoC» и указаны функции, реализованные в ней.

В разделе «Описание программных интерфейсов» приведена классификация программных интерфейсов, а также указаны заголовки объявления, входные параметры, возвращаемые значения и примеры вызовов интерфейсов.

В разделе «Обработка исключительных ситуаций» описан метод обработки исключительных ситуаций.

В Приложении 1 приведен текст приложения – примера вызова всех интерфейсов библиотеки «GisCryptoC», для реализации криптографических функций.

В конце документа приведен список использованных терминов и сокращений.

Содержание

1 Общие сведения

Библиотека «GisCryptoC» содержит программные интерфейсы, реализованные на языке C, и предназначена для вызова основных функций криптографической платформы «Litoria Crypto Platform».

В библиотеку «GisCryptoC» включены интерфейсы для вызова следующих функций криптографической платформы «Litoria Crypto Platform»:

- получение коллекции сертификатов из хранилища;
- создание электронной подписи (ЭП);
- поточное создание ЭП;
- проверка ЭП;
- проверка ЭП с извлечением данных;
- проверка отделенной ЭП;
- поточная проверка ЭП;
- поточная проверка отделенной ЭП;
- шифрование данных;
- поточное шифрование данных;
- расшифровывание данных;
- поточное расшифровывание данных.

2 Описание программных интерфейсов

При использовании программных интерфейсов библиотеки «GisCryptoC» и наличии на рабочей станции криптографической платформы «Litoria Crypto Platform» можно реализовать клиентское программное обеспечение, в котором пользователю предоставляются перечисленные выше функции криптографической платформы «Litoria Crypto Platform».

Классификация программных интерфейсов представлена в таблице 2.1.

Таблица 2.1. Классификация программных интерфейсов

Тип интерфейсов	Интерфейс	Назначение
Интерфейсы для работы с сертификатами	getCertFromStore	Получение сертификата из хранилища
	getAllCertFromStore	Получение коллекции сертификатов из хранилища
Интерфейсы для создания ЭП	signData	Создание ЭП
	signDataStream	Поточное создание ЭП
Интерфейсы для проверки ЭП	verifyEds	Проверка ЭП
	verifyEdsExractData	Проверка ЭП с извлечением данных
	verifyEdsDetached	Проверка отделенной ЭП
	verifyEdsStream	Поточная проверка ЭП
	verifyEdsDetachedStream	Поточная проверка отделенной ЭП
Интерфейсы для шифрования данных	encryptData	Шифрование данных
	encryptDataStream	Поточное шифрование данных
Интерфейсы для расшифровывания данных	decryptData	Расшифровывание данных
	decryptDataStream	Поточное расшифровывание данных
Интерфейсы для очистки структуры, массива структур и памяти	freeGisCryptoError	Очистка структуры, представляющей ошибку библиотеки
	freeSignatureInfos	Очистка массива структур, представляющих информацию о подписи
	freeMemory	Очистка памяти, выделенной в библиотеке

2.1 Получение сертификата из хранилища. Интерфейс getCertFromStore

Заголовок объявления интерфейса:

```
GISCRYPTOC_API bool getCertFromStore(
    const wchar_t* storeName,
    unsigned char*& outCertData,
    unsigned long& outCertDataSize,
    GisCryptoError& outCryptoError);
```

Входные параметры:

storeName – имя хранилища

Примеры имен хранилищ:

- «MY» – хранилище личных сертификатов
- «CA» – сертификаты удостоверяющего центра
- «ROOT» – корневые сертификаты
- «AddressBook» – сертификаты других пользователей

outCertData – массив байт сертификата

outCertDataSize – размер массива байт сертификата

outCryptoError – ошибки операции

Возвращаемое значение:

Результат операции

Пример вызова:

```
unsigned char* signerCert = 0;
unsigned long signerCertSize = 0;
GisCryptoC::GisCryptoError error;
if (!GisCryptoC::getCertFromStore(L"MY", signerCert, signerCertSize, error)) {
        GisCryptoC::freeGisCryptoError(error);
        }
```

2.2 Получение коллекции сертификатов из хранилища. Интерфейс getAllCertFromStore

Заголовок объявления интерфейса:

```
GISCRYPTOC_API bool getAllCertFromStore(
        const wchar_t* storeName,
        unsigned char**& outCerts,
        unsigned long*& outCertsSizes,
        unsigned long& outCertsCount,
        GisCryptoError& outCryptoError);
```

Входные параметры:

StoreName – имя хранилища

Примеры имен хранилищ:

- «MY» – хранилище личных сертификатов
- «CA» – сертификаты удостоверяющего центра
- «ROOT» – корневые сертификаты
- «AddressBook» – сертификаты других пользователей

outCerts – массив сертификатов

outCertsSize – массив размеров сертификатов

outCertsCount – размер массива сертификатов

outCryptoError – ошибки операции

Возвращаемое значение:

Результат операции

Пример вызова:

```
unsigned char** receviersCerts = 0;
unsigned long* receiversCertsSizes = 0;
unsigned long receviersCertsCount = 0;
if (!GisCryptoC:: getAllCertFromStore(L"AddressBook", receviersCerts, receiversCertsSizes,
receviersCertsCount, error)) {
GisCryptoC::freeGisCryptoError(error);
}
```

2.3 Создание ЭП. Интерфейс signData

Заголовок объявления интерфейса:

```
GISCRYPTOC_API bool signData(
        const unsigned char* data,
        unsigned long dataSize,
        const CreateSignInfo& createSignInfo,
        unsigned char*& outData,
        unsigned long& outDataSize,
        GisCryptoError& outCryptoError);
```

Входные параметры:

data – массив данных для подписи
dataSize – размер массива данных для подписи
createSignInfo – параметры подписи
outData – массив выходных данных
outDataSize – размер массива выходных данных
outCryptoError – ошибки операции

Пример вызова:

```
GisCryptoC::CreateSignInfo createSignInfo;
memset(&createSignInfo, 0, sizeof(createSignInfo));
createSignInfo.cert = signerCert;
createSignInfo.certSize = signerCertSize;
createSignInfo.tspAddress = tspAddress;
createSignInfo.edsFlag = edsFlag;
unsigned char* signedData= 0;
unsigned long signedDataSize = 0;
if (GisCryptoC::signData(inputData, inputDataSize, createSignInfo, signedData, signedDataSize, error)) {
        FILE* fSigned = fopen(".\\Test\\1.txt.p7s", "wb");
        if (fSigned) {
                if(signedData && signedDataSize) {
                        fwrite(signedData, 1, signedDataSize, fSigned);
                }
                fclose(fSigned);
        }
}
```

2.3.1 Структура CreateSignInfo, описывающая параметры подписи

Описание полей структуры, описывающей параметры подписи, представлено в таблице 2.2.

Таблица 2.2. Поля структуры, описывающей параметры подписи

Поле структуры	Значение
const unsigned char* cert;	Массив байтов сертификата
unsigned long certSize;	Размер массива байтов сертификатов
const wchar_t* comment;	Комментарий

EdsFlag edsFlag;	Тип управляющих флагов операций typedef unsigned int EdsFlag:
	• флаг отделенной ЭП:
	static const EdsFlag DetachedEds = 0x01;
	• флаг включения штампа времени:
	static const EdsFlag IncludeTimeStamp = 0x02;
	• флаг создания усовершенствованной ЭП:
	static const EdsFlag CreateCades = 0x08;

const wchar_t* tspAddress;	Адрес службы штампов времени

2.4 Поточное создание ЭП. Интерфейс signDataStream

Заголовок объявления интерфейса:

```
GISCRYPTOC_API bool signDataStream(
        const CreateSignInfo& createSignInfo,
        bool (*dataCallback) ( unsigned char*& data, unsigned long& dataSize),
        void (*outputDataCallback) (const unsigned char* data, unsigned long dataSize, bool lastCall),
        GisCryptoError& outCryptoError);
```

Входные параметры:

createSignInfo – параметры подписи
dataCallback – указатель на функцию обратного вызова передачи данных для подписи
outputDataCallback – указатель на функцию обратного вызова для получения подписанных данных
outCryptoError – ошибки операции

Пример вызова:

```
GisCryptoC::CreateSignInfo createSignInfo;
memset(&createSignInfo, 0, sizeof(createSignInfo));
createSignInfo.cert = signerCert;
createSignInfo.certSize = signerCertSize;
createSignInfo.tspAddress = tspAddress;
createSignInfo.edsFlag = edsFlag;
GisCryptoC::GisCryptoError error;
if (GisCryptoC::signDataStream(createSignInfo, singDataCallback, outputSignedDataCallback, error))
{
}
```

2.5 Проверка ЭП. Интерфейс verifyEds

Заголовок объявления интерфейса:

```
GISCRYPTOC_API bool verifyEds(
    const unsigned char* signedData,
    unsigned long signedDataSize,
    SignatureInfo*& signatureInfos,
    unsigned long& signatureInfosSize,
    GisCryptoError& outCryptoError);
```

Входные параметры:

signedData – массив подписанных данных
signedDataSize – размер массива подписанных данных
signatureInfos – массив структур информации о подписи
signatureInfosSize – размер массива структур
outCryptoError – ошибки операции

Возвращаемое значение:

Результат операции

Пример вызова:

```
unsigned char* inputSignedData = 0;
unsigned long inputSignedDataSize = 0;
FILE* fInForVerify = fopen(".\\Test\\1.txt.p7s", "rb");
if (fInForVerify) {
        fseek(fInForVerify, 0, SEEK_END);
        inputSignedDataSize = ftell(fInForVerify);
        fseek(fInForVerify, 0, SEEK_SET);
        if(inputSignedDataSize) {
                inputSignedData = new unsigned char[inputSignedDataSize];
                fread(inputSignedData, 1, inputSignedDataSize, fInForVerify);
        }
        fclose(fInForVerify);
}
GisCryptoC::SignatureInfo* signInfos = 0;
unsigned long signInfosSize = 0;
```

```
if (GisCryptoC::verifyEds(inputSignedData, inputSignedDataSize, signInfos, signInfosSize, error)) {
        if (signInfos) {
                for (int i = 0; i < signInfosSize; i++) {
                        GisCryptoC::SignatureInfo signInfo = signInfos[i];
                }
                GisCryptoC::freeSignatureInfos(signInfos, signInfosSize);
        }
} else {
        GisCryptoC::freeGisCryptoError(error);
}
```

2.5.1 Структура SignatureInfo, предоставляющая информацию о подписи

Описание полей структуры, предоставляющей информацию о подписи, приведено в таблице 2.3.

Таблица 2.3. Поля структуры, предоставляющей информацию о подписи

Поле структуры	Значение
unsigned long index;	Индекс подписи
bool verifyResult;	Результат проверки ЭП
bool verifyCertificateResult;	Результат проверки сертификата
unsigned char* cert;	Массив байт сертификата подписчика
unsigned long certSize;	Размер массива байт сертификата подписчика
time_t signatureTime;	Время создания ЭП
unsigned char* rawSignature;	Массив байт подписи
unsigned long rawSignatureSize;	Размер массива байт подписи
wchar_t* comment;	Комментарий
bool advanced;	Флаг усовершенствованной ЭП
bool timeStampIncluded;	Флаг наличия штампа времени
bool counterSignature;	Флаг заверяющей подписи
GiSCryptoError signatureError;	Ошибки, возникшие при проверке подписи

2.6 Проверка ЭП с извлечением данных. Интерфейс verifyEdsExractData

Заголовок объявления интерфейса:

```
GISCRYPTOC_API bool verifyEdsExractData(
        const unsigned char* signedData,
        unsigned long signedDataSize,
        SignatureInfo*& signatureInfos,
        unsigned long& signatureInfosSize,
        unsigned char*& outData,
        unsigned long& outDataSize,
        GisCryptoError& outCryptoError);
```

Входные параметры:

signedData – массив подписанных данных
signedDataSize – размер массива подписанных данных
signatureInfos – массив структур информации о подписи
signatureInfosSize – размер массива структур
outData – массив выходных данных
outDataSize – размер массива выходных данных
outCryptoError – ошибки операции

Возвращаемое значение:

Результат операции

Пример вызова:

```
signInfos = 0;
signInfosSize = 0;
unsigned char* inputExtractedData = 0;
unsigned long inputExtractedDataSize = 0;
if (GisCryptoC::verifyEdsExractData(inputSignedData, inputSignedDataSize, signInfos, signInfosSize,
inputExtractedData, inputExtractedDataSize, error)) {
        if (signInfos) {
                for (int i = 0; i < signInfosSize; i++) {
                        GisCryptoC::SignatureInfo signInfo = signInfos[i];
                }
                FILE* fExtracted = fopen(".\\Test\\1_extracted.txt", "wb");
                if (fExtracted) {
                        if(inputExtractedData && inputExtractedDataSize) {
                                fwrite(inputExtractedData, 1, inputExtractedDataSize, fExtracted);
                                GisCryptoC::freeMemory(inputExtractedData);
                        }
                        fclose(fExtracted);
                }
                GisCryptoC::freeSignatureInfos(signInfos, signInfosSize);
        }
}
```

2.7 Проверка отделенной ЭП. Интерфейс verifyEdsDetached

Заголовок объявления интерфейса:

```
GISCRYPTOC_API bool verifyEdsDetached(
        const unsigned char* signedData,
        unsigned long signedDataSize,
        const unsigned char* detachedData,
        unsigned long detachedDataSize,
        SignatureInfo*& signatureInfos,
        unsigned long& signatureInfosSize,
        GisCryptoError& outCryptoError);
```

Входные параметры:

signedData – массив отсоединенной подписи данных
signedDataSize – размер массива отсоединенной подписи данных
detachedData – массив данных
detachedDataSize – размер массива данных
signatureInfos – массив структур информации о подписи
signatureInfosSize – размер массива структур
outCryptoError – ошибки операции

Возвращаемое значение:

Результат операции

Пример вызова:

```
signInfos = 0;
signInfosSize = 0;
unsigned char* inputSignedData = 0;
unsigned long inputSignedSize = 0;
unsigned char* inputDetachedData = 0;
unsigned long inputDetachedSize = 0;
if (GisCryptoC::verifyEdsDetached(inputSignedData, inputSignedSize, inputDetachedData,
inputDetachedSize, signInfos, signInfosSize, error)) {
        if (signInfos) {
                for (int i = 0; i < signInfosSize; i++) {
                        GisCryptoC::SignatureInfo signInfo = signInfos[i];
                }
                fclose(fExtracted);
                }
                GisCryptoC::freeSignatureInfos(signInfos, signInfosSize);
        }
}
```

2.8 Поточная проверка ЭП. Интерфейс verifyEdsStream

Заголовок объявления интерфейса:

```
GISCRYPTOC_API bool verifyEdsStream(
        SignatureInfo*& signatureInfos,
        unsigned long& signatureInfosSize,
        bool (*dataCallback) (unsigned char*& data, unsigned long& dataSize),
        void (*outputDataCallback) (const unsigned char* data, unsigned long dataSize, bool lastCall),
        GisCryptoError& outCryptoError);
```

Входные параметры:

signatureInfos – массив структур информации о подписи
signatureInfosSize – размер массива структур
dataCallback – указатель на функцию обратного вызова передачи подписанных данных
outputDataCallback – указатель на функцию обратного вызова для получения исходных данных
outCryptoError – ошибки операции

Возвращаемое значение:

Результат операции

Пример вызова:

```
fSignedStreamToVerify = fopen(".\\Test\\1.txt.p7s","rb");
fVerifiedStream = fopen(".\\Test\\1_verified.txt","wb");
GisCryptoC::SignatureInfo* signInfos = 0;
unsigned long signInfosSize = 0;
if (!GisCryptoC::verifyEdsStream(signInfos, signInfosSize, verifyDataCallback, outputVerifyDataCallback,
error)) {
} else {
        GisCryptoC::freeGisCryptoError(error);
}
```

2.9 Поточная проверка отделенной ЭП. Интерфейс verifyEdsDetachedStream

Заголовок объявления интерфейса:

```
GISCRYPTOC_API bool verifyEdsDetachedStream(
    SignatureInfo*& signatureInfos,
    unsigned long& signatureInfosSize,
    bool (*dataCallback) (unsigned char*& data, unsigned long& dataSize),
    bool (*detachedDataCallback) (unsigned char*& data, unsigned long& dataSize),
    GisCryptoError& outCryptoError);
```

Входные параметры:

signatureInfos – массив структур информации о подписи
signatureInfosSize – размер массива структур
dataCallback – указатель на функцию обратного вызова передачи исходных данных (документ)
detachedDataCallback – указатель на функцию обратного вызова передачи данных отделенной подписи
outCryptoError – ошибки операции

Возвращаемое значение:

Результат операции

Пример вызова:

```
fSignedStreamToVerifyDetached = fopen(".\\Test\\1_detached.txt.p7s","rb");
fSignedStreamToVerifyDetachedData = fopen(".\\Test\\1.txt","rb");
signInfos = 0;
signInfosSize = 0;
if (GisCryptoC::verifyEdsDetachedStream(signInfos, signInfosSize, verifyDetachedEdsDataCallback, verifyDetachedDataCallback, error)) {
} else {
    GisCryptoC::freeGisCryptoError(error);
}
```

2.10 Шифрование данных. Интерфейс encryptDat

Заголовок объявления интерфейса:

```
GISCRYPTOC_API bool encryptData(
    const unsigned char* data,
    unsigned long dataSize,
    const unsigned char** receiversCerts,
    const unsigned long*& receiversCertsSizes,
    unsigned long receiversCertsCount,
    unsigned char*& outData,
    unsigned long& outDataSize,
    GisCryptoError& outCryptoError);
```

Входные параметры:

data – массив данных для шифрования
dataSize – размер массива данных для шифрования
receiversCerts – массив сертификатов получателей
receiversCertsSizes – массив размеров сертификатов получателей
receiversCertsCount – размер массива сертификатов получателей
outData – массив зашифрованных данных
outDataSize – размер массива зашифрованных данных
outCryptoError – ошибки операции

Возвращаемое значение:

Результат операции

Пример вызова:

```
unsigned char* encyptedData= 0;
unsigned long encyptedDataSize = 0;
if (GisCryptoC::encryptData(inputData, inputDataSize, const_cast<const unsigned char**>(receviersCerts), const_cast<const unsigned long*&>(receiversCertsSizes), receviersCertsCount, encyptedData, encyptedDataSize, error)) {
	FILE* fEncrypted = fopen(".\\Test\\1.txt.p7m", "wb");
	if (fEncrypted) {
		if(encyptedData && encyptedDataSize) {
			fwrite(encyptedData, 1, encyptedDataSize, fEncrypted);
		}
		fclose(fEncrypted);
	}
}
```

2.11 Поточное шифрование данных. Интерфейс encryptDataStream

Заголовок объявления интерфейса:

```
GISCRYPTOC_API bool encryptDataStream(
	const unsigned char** receiversCerts,
	const unsigned long*& receiversCertsSizes,
	unsigned long receiversCertsCount,
	bool (*dataCallback) (unsigned char*& data, unsigned long& dataSize),
	void (*outputDataCallback) (const unsigned char* data, unsigned long dataSize, bool lastCall),
	GisCryptoError& outCryptoError);
```

Входные параметры:

receiversCerts – массив сертификатов получателей
receiversCertsSizes – массив размеров сертификатов получателей
receiversCertsCount – размер массива сертификатов получателей
dataCallback – указатель на функцию обратного вызова передачи данных для шифрования
outputDataCallback – указатель на функцию обратного вызова для получения зашифрованных данных
outCryptoError – ошибки операции

Возвращаемое значение:

Результат операции

Пример вызова:

```
fInStreamToCrypt = fopen(".\\Test\\1.txt", "rb");
fCryptedStream = fopen(".\\Test\\1.txt.p7m","wb");
unsigned char** receviersCerts = 0;
unsigned long* receiversCertsSizes = 0;
unsigned long receviersCertsCount = 0;
if (!GisCryptoC::getAllCertFromStore(L"AddressBook", receviersCerts, receiversCertsSizes, receviersCertsCount, error)) {
	GisCryptoC::freeGisCryptoError(error);
}
if (!GisCryptoC::encryptDataStream(const_cast<const unsigned char**>(receviersCerts), const_cast<const unsigned long*&>(receiversCertsSizes), receviersCertsCount, cryptDataCallback, outputCryptDataCallback, error)) {
	GisCryptoC::freeGisCryptoError(error);
}
```

2.12 Расшифровывание данных. Интерфейс decryptData

Заголовок объявления интерфейса:

```
GISCRYPTOC_API bool decryptData(
        const unsigned char* encryptedData,
        unsigned long encryptedDataSize,
        unsigned char*& receiverCert,
        unsigned long& receiverCertSize,
        unsigned char*& outData,
        unsigned long& outDataSize,
        GisCryptoError& outCryptoError);
```

Входные параметры:

encryptedData – массив зашифрованных данных
encryptedDataSize – размер массива зашифрованных данных
receiverCert – массив сертификата получателя
receiverCertSize – размер массива сертификата получателя
outData – массив выходных данных
outDataSize – размер массива выходных данных
outCryptoError – ошибки операции

Выходные параметры:

Результат операции

Пример вызова:

```
unsigned char* receiverCert = 0;
unsigned long receiverCertSize = 0;
unsigned char* decryptedData = 0;
unsigned long decryptedDataSize = 0;
if (GisCryptoC::decryptData(inputEncryptedData, inputEncryptedDataSize, receiverCert, receiverCertSize,
decryptedData, decryptedDataSize, error)) {
        FILE* fDecrypted = fopen(".\\Test\\1_decrypted.txt", "wb");
        if (fDecrypted) {
                if(decryptedData && decryptedDataSize) {
                        fwrite(decryptedData, 1, decryptedDataSize, fDecrypted);
                }
                fclose(fDecrypted);
        }
}
```

2.13 Поточное расшифровывание данных. Интерфейс decryptDataStream

Заголовок объявления интерфейса:

```
GISCRYPTOC_API bool decryptDataStream(
        unsigned char*& receiverCert,
        unsigned long& receiverCertSize,
        bool (*dataCallback) (unsigned char*& data, unsigned long& dataSize),
        void (*outputDataCallback) (const unsigned char* data, unsigned long dataSize, bool lastCall),
        GisCryptoError& outCryptoError);
```

Входные параметры:

receiverCert – массив сертификата получателя
receiverCertSize – размер массива сертификата получателя
dataCallback – указатель на функцию обратного вызова передачи зашифрованных данных
outputDataCallback – указатель на функцию обратного вызова для получения расшифрованных данных
outCryptoError – ошибки операции

Выходные параметры:

Результат операции

Пример вызова:

```
fCryptedStreamToDecrypt = fopen(".\\Test\\1.txt.p7m","rb");
fDecryptedStream = fopen(".\\Test\\1_decr.txt","wb");
unsigned char* receiverCert = 0;
unsigned long receiverCertSize = 0;
if (!GisCryptoC::decryptDataStream(receiverCert, receiverCertSize, decryptDataCallback, outputDecryptDataCallback, error)) {
    GisCryptoC::freeGisCryptoError(error);
}
```

2.14 Очистка структуры, представляющей ошибку библиотеки. Интерфейс freeGisCryptoError

Заголовок объявления интерфейса:

```
GISCRYPTOC_API void freeGisCryptoError(
    GisCryptoError& cryptoError);
```

Входные параметры:

cryptoError – ошибка

Пример вызова:

```
GisCryptoC::GisCryptoError error;
    if (!GisCryptoC::getCertFromStore(L"MY", signerCert, signerCertSize, error)) {
        GisCryptoC::freeGisCryptoError(error);
    }
```

2.15 Очистка массива структур, представляющих информацию о подписи. Интерфейс freeSignatureInfos

Заголовок объявления интерфейса:

```
GISCRYPTOC_API void freeSignatureInfos(
    SignatureInfo* signatureInfos,
    unsigned long signatureInfosSize);
```

Входные параметры:

signatureInfos – массив структур
signatureInfosSize – размер массива структур

Пример вызова:

```
GisCryptoC::SignatureInfo* signInfos = 0;
unsigned long signInfosSize = 0;
if (!GisCryptoC::verifyEdsStream(signInfos, signInfosSize, verifyDataCallback, outputVerifyDataCallback,
error)) {
} else {
        GisCryptoC::freeGisCryptoError(error);
}
if (signInfos) {
        GisCryptoC::freeSignatureInfos(signInfos, signInfosSize);
}
```

2.16 Очистка памяти, выделенной в памяти. Интерфейс freeMemory

Заголовок объявления интерфейса:

```
GISCRYPTOC_API void freeMemory(
    void* memoryPtr);
```

Входные параметры:

memoryPtr – указатель на память

Пример вызова:

```
delete inputDataToSignDetached;
GisCryptoC::freeMemory(signerCert);
```

3 Обработка исключительных ситуаций

Функции обработки исключений помогают обрабатывать любые непредвиденные или исключительные ситуации, происходящие при выполнении основных операций.

Пример обработки исключения при выполнении операции создания ЭП:

```
GisCryptoC::GisCryptoError error;

if (GisCryptoC::signDataStream(createSignInfo, singDataCallback, outputSignedDataCallback,
error)) {
        } else {

                int* bhCodes = error.bhGiSErrorCodes;
                for (int i = 0; i < error.bhGiSErrorCodesSize; i++) {
                        int bhCode = bhCodes[i];
                }
                wchar_t** bhStrings = error.bhGiSErrorStrings;
                for (int i = 0; i < error.bhGiSErrorStringsSize; i++) {
                        wchar_t* bhString = error.bhGiSErrorStrings[i];
                }
                unsigned long sysCode = error.systemErrorCode;
                wchar_t* sysString = error.systemErrorText;

                GisCryptoC::freeGisCryptoError(error);
}
```

3.1 Структура GiSCryptoError, описывающая ошибки библиотеки

Описание полей структуры, описывающей ошибки библиотеки, представлено в таблице 3.1.

Таблица 3.1. Поля структуры, описывающей ошибки библиотеки

Поле структуры	Значение
unsigned long systemErrorCode;	Код системной ошибки
wchar_t* systemErrorText;	Описание системной ошибки
int* bhGiSErrorCodes;	Массив ошибок криптографической библиотеки
unsigned long bhGiSErrorCodesSize;	Размер массива ошибок криптографической библиотеки
wchar_t** bhGiSErrorStrings;	Массив описаний ошибок криптографической библиотеки
unsigned long bhGiSErrorStringsSize;	Размер массива описаний ошибок криптографической библиотеки

Приложение 1. Пример вызова всех интерфейсов библиотеки «GisCryptoC» для реализации криптографических функций

```
#define TRUE FALSE

#include "giscrypto.h"

#include <Windows.h>

#include <string.h>
#include <stdio.h>
#include <errno.h>

void testSimpleFunctions();
void testStreamedFunctions();

FILE* fInStreamToSign;
FILE* fSignedStream;

FILE* fSignedStreamToVerify;
FILE* fVerifiedStream;

FILE* fInStreamToCrypt;
FILE* fCryptedStream;

FILE* fCryptedStreamToDecrypt;
FILE* fDecryptedStream;

FILE* fInStreamToSignDetached;
FILE* fSignedStreamDetached;
FILE* fSignedStreamToVerifyDetached;
FILE* fSignedStreamToVerifyDetachedData;

unsigned char* inputDataToSign;
unsigned char* inputDataToCrypt;
unsigned char* inputDataToDecrypt;
unsigned char* inputDataToVerify;
unsigned char* inputDataToSignDetached;
unsigned char* inputDataToVerifyDetachedData;
unsigned char* inputDataToVerifyDetached;

bool singDataCallback(unsigned char*& data, unsigned long& dataSize);
void outputSignedDataCallback (const unsigned char* data, unsigned long dataSize, bool lastCall);

bool cryptDataCallback(unsigned char*& data, unsigned long& dataSize);
void outputCryptDataCallback (const unsigned char* data, unsigned long dataSize, bool lastCall);

bool decryptDataCallback(unsigned char*& data, unsigned long& dataSize);
void outputDecryptDataCallback (const unsigned char* data, unsigned long dataSize, bool lastCall);

bool verifyDataCallback(unsigned char*& data, unsigned long& dataSize);
void outputVerifyDataCallback (const unsigned char* data, unsigned long dataSize, bool lastCall);

bool singDetachedDataCallback(unsigned char*& data, unsigned long& dataSize);
void outputDetachedSignedDataCallback(const unsigned char* data, unsigned long dataSize, bool lastCall);

bool detachedDataCallback(unsigned char*& data, unsigned long& dataSize);
bool verifyDetachedDataCallback(unsigned char*& data, unsigned long& dataSize);
bool verifyDetachedEdsDataCallback(unsigned char*& data, unsigned long& dataSize);

int main()
```

```
{
	testStreamedFunctions();

	return 0;
}

void testStreamedFunctions()
{
	wchar_t* tspAddress = L"http://tsp.gaz-is.ru/tsp00/tsp.srf";
	GisCryptoC::EdsFlag edsFlag = GisCryptoC::IncludeTimeStamp;

	fInStreamToSign = fopen(".\\Test\\1.txt", "rb");
	fSignedStream = fopen(".\\Test\\1.txt.p7s", "wb");

	//Получаем сертификат из хранилища

	unsigned char* signerCert = 0;
	unsigned long signerCertSize = 0;

	GisCryptoC::GisCryptoError error;
	if (!GisCryptoC::getCertFromStore(L"MY", signerCert, signerCertSize, error)) {
		GisCryptoC::freeGisCryptoError(error);
	}

	//Создаем ЭП

	GisCryptoC::CreateSignInfo createSignInfo;
	memset(&createSignInfo, 0, sizeof(createSignInfo));

	createSignInfo.cert = signerCert;
	createSignInfo.certSize = signerCertSize;
	createSignInfo.tspAddress = tspAddress;
	createSignInfo.edsFlag = edsFlag;

	if (GisCryptoC::signDataStream(createSignInfo, singDataCallback, outputSignedDataCallback, error)) {

	} else {

		//Обработка ошибки

		int* bhCodes = error.bhGiSErrorCodes;
		for (int i = 0; i < error.bhGiSErrorCodesSize; i++) {
			int bhCode = bhCodes[i];
		}
		wchar_t** bhStrings = error.bhGiSErrorStrings;
		for (int i = 0; i < error.bhGiSErrorStringsSize; i++) {
			wchar_t* bhString = error.bhGiSErrorStrings[i];
		}
		unsigned long sysCode = error.systemErrorCode;
		wchar_t* sysString = error.systemErrorText;

		GisCryptoC::freeGisCryptoError(error);
	}

	delete inputDataToSign;

	if (fInStreamToSign) {
		fclose (fInStreamToSign);
	}
	if (fSignedStream) {
		fclose (fSignedStream);
	}
```

```
	fInStreamToCrypt = fopen(".\\Test\\1.txt", "rb");
	fCryptedStream = fopen(".\\Test\\1.txt.p7m","wb");

	unsigned char** receviersCerts = 0;
	unsigned long* receiversCertsSizes = 0;
	unsigned long receviersCertsCount = 0;

	if (!GisCryptoC::getAllCertFromStore(L"AddressBook", receviersCerts, receiversCertsSizes, receviersCertsCount,
error)) {
		GisCryptoC::freeGisCryptoError(error);
	}

	if (!GisCryptoC::encryptDataStream(const_cast<const unsigned char**>(receviersCerts), const_cast<const unsigned
long*&>(receiversCertsSizes), receviersCertsCount, cryptDataCallback, outputCryptDataCallback, error)) {
		GisCryptoC::freeGisCryptoError(error);
	}

	delete inputDataToCrypt;

	if (fInStreamToCrypt) {
		fclose(fInStreamToCrypt);
	}
	if (fCryptedStream) {
		fclose(fCryptedStream);
	}

	fCryptedStreamToDecrypt = fopen(".\\Test\\1.txt.p7m","rb");
	fDecryptedStream = fopen(".\\Test\\1_decr.txt","wb");

	unsigned char* receiverCert = 0;
	unsigned long receiverCertSize = 0;

	if (!GisCryptoC::decryptDataStream(receiverCert, receiverCertSize, decryptDataCallback, outputDecryptDataCallback,
error)) {
		GisCryptoC::freeGisCryptoError(error);
	}
	if (receiverCert) {
		GisCryptoC::freeMemory(receiverCert);
	}

	delete inputDataToDecrypt;

	if (fCryptedStreamToDecrypt) {
		fclose(fCryptedStreamToDecrypt);
	}
	if (fDecryptedStream) {
		fclose(fDecryptedStream);
	}

	//Проверка ЭП

	fSignedStreamToVerify = fopen(".\\Test\\1.txt.p7s","rb");
	fVerifiedStream = fopen(".\\Test\\1_verified.txt","wb");

	GisCryptoC::SignatureInfo* signInfos = 0;
	unsigned long signInfosSize = 0;

	if (!GisCryptoC::verifyEdsStream(signInfos, signInfosSize, verifyDataCallback, outputVerifyDataCallback, error)) {
	} else {
		GisCryptoC::freeGisCryptoError(error);
	}
```

```
	if (signInfos) {
		GisCryptoC::freeSignatureInfos(signInfos, signInfosSize);
	}

	delete inputDataToVerify;

	if (fSignedStreamToVerify) {
		fclose(fSignedStreamToVerify);
	}
	if (fVerifiedStream) {
		fclose(fVerifiedStream);
	}

	// Создание отделенной ЭП

	fInStreamToSignDetached  = fopen(".\\Test\\1.txt","rb");
	fSignedStreamDetached = fopen(".\\Test\\1_detached.txt.p7s","wb");

	edsFlag = GisCryptoC::IncludeTimeStamp | GisCryptoC::DetachedEds;
	createSignInfo.edsFlag = edsFlag;

	if (GisCryptoC::signDataStream(createSignInfo, singDetachedDataCallback, outputDetachedSignedDataCallback, error))
{

	} else {
		GisCryptoC::freeGisCryptoError(error);
	}

	delete inputDataToSignDetached;
	GisCryptoC::freeMemory(signerCert);

	if (fInStreamToSignDetached) {
		fclose(fInStreamToSignDetached);
	}
	if (fSignedStreamDetached) {
		fclose(fSignedStreamDetached);
	}

	// Проверка отделенной ЭП

	fSignedStreamToVerifyDetached = fopen(".\\Test\\1_detached.txt.p7s","rb");
	fSignedStreamToVerifyDetachedData = fopen(".\\Test\\1.txt","rb");

	signInfos = 0;
	signInfosSize = 0;

	if (GisCryptoC::verifyEdsDetachedStream(signInfos, signInfosSize, verifyDetachedEdsDataCallback,
verifyDetachedDataCallback, error)) {
	} else {
		GisCryptoC::freeGisCryptoError(error);
	}

	delete inputDataToVerifyDetached;
	delete inputDataToVerifyDetachedData;

	if (fSignedStreamToVerifyDetached) {
		fclose(fSignedStreamToVerifyDetached);
	}
	if (fSignedStreamToVerifyDetachedData) {
		fclose(fSignedStreamToVerifyDetachedData);
	}
}
```

```
bool singDataCallback(unsigned char*& data, unsigned long& dataSize)
{
	bool lastCall = false;

	if (inputDataToSign) {
		delete inputDataToSign;
	}
	unsigned long datatoRead = 128;
	unsigned long dataRead = 0;
	inputDataToSign = new  unsigned char[datatoRead];
	memset(inputDataToSign, 0, datatoRead);

	if (fInStreamToSign) {
		dataRead = fread(inputDataToSign, 1, datatoRead, fInStreamToSign);
		if (feof(fInStreamToSign)) {
			fseek(fInStreamToSign, 0, SEEK_SET);
			lastCall = true;
		}
	}
	dataSize = dataRead;
	data = inputDataToSign;

	return lastCall;
}

void outputSignedDataCallback (const unsigned char* data, unsigned long dataSize, bool lastCall)
{
	if (fSignedStream) {
		if(data && dataSize) {
			fwrite(data, 1, dataSize, fSignedStream);
		}
		if (lastCall) {
			fclose(fSignedStream);
		}
	}
}

bool cryptDataCallback(unsigned char*& data, unsigned long& dataSize)
{
	bool lastCall = false;

	if (inputDataToCrypt) {
		delete inputDataToCrypt;
	}
	unsigned long datatoRead = 128;
	unsigned long dataRead = 0;
	inputDataToCrypt = new  unsigned char[datatoRead];
	memset(inputDataToCrypt, 0, datatoRead);

	if (fInStreamToCrypt) {
		dataRead = fread(inputDataToCrypt, 1, datatoRead, fInStreamToCrypt);
		if (feof(fInStreamToCrypt)) {
			fseek(fInStreamToCrypt, 0, SEEK_SET);
			lastCall = true;
		}
	}
	dataSize = dataRead;
	data = inputDataToCrypt;

	return lastCall;
}
```

```
void outputCryptDataCallback (const unsigned char* data, unsigned long dataSize, bool lastCall)
{
	if (fCryptedStream) {
		if(data && dataSize) {
			fwrite(data, 1, dataSize, fCryptedStream);
		}
		if (lastCall) {
			fclose(fCryptedStream);
		}
	}
}

bool decryptDataCallback(unsigned char*& data, unsigned long& dataSize)
{
	bool lastCall = false;

	if (inputDataToDecrypt) {
		delete inputDataToDecrypt;
	}
	unsigned long datatoRead = 128;
	unsigned long dataRead = 0;
	inputDataToDecrypt = new  unsigned char[datatoRead];
	memset(inputDataToDecrypt, 0, datatoRead);

	if (fCryptedStreamToDecrypt) {
		dataRead = fread(inputDataToDecrypt, 1, datatoRead, fCryptedStreamToDecrypt);
		if (feof(fCryptedStreamToDecrypt)) {
			fseek(fCryptedStreamToDecrypt, 0, SEEK_SET);
			lastCall = true;
		}
	}
	dataSize = dataRead;
	data = inputDataToDecrypt;

	return lastCall;
}

void outputDecryptDataCallback(const unsigned char* data, unsigned long dataSize, bool lastCall)
{
	if (fDecryptedStream) {
		if(data && dataSize) {
			fwrite(data, 1, dataSize, fDecryptedStream);
		}
		if (lastCall) {
			fclose(fDecryptedStream);
		}
	}
}

bool verifyDataCallback(unsigned char*& data, unsigned long& dataSize)
{
	bool lastCall = false;

	if (inputDataToVerify) {
		delete inputDataToVerify;
	}
	unsigned long datatoRead = 128;
	unsigned long dataRead = 0;
	inputDataToVerify = new  unsigned char[datatoRead];
	memset(inputDataToVerify, 0, datatoRead);
```

```
	if (fSignedStreamToVerify) {
		dataRead = fread(inputDataToVerify, 1, datatoRead, fSignedStreamToVerify);
		if (feof(fSignedStreamToVerify)) {
			fseek(fSignedStreamToVerify, 0, SEEK_SET);
			lastCall = true;
		}
	}
	dataSize = dataRead;
	data = inputDataToVerify;

	return lastCall;
}

bool singDetachedDataCallback(unsigned char*& data, unsigned long& dataSize)
{
	bool lastCall = false;

	if (inputDataToSignDetached) {
		delete inputDataToSignDetached;
	}
	unsigned long datatoRead = 128;
	unsigned long dataRead = 0;
	inputDataToSignDetached = new  unsigned char[datatoRead];
	memset(inputDataToSignDetached, 0, datatoRead);

	if (fInStreamToSignDetached) {
		dataRead = fread(inputDataToSignDetached, 1, datatoRead, fInStreamToSignDetached);
		if (feof(fInStreamToSignDetached)) {
			fseek(fInStreamToSignDetached, 0, SEEK_SET);
			lastCall = true;
		}
	}
	dataSize = dataRead;
	data = inputDataToSignDetached;

	return lastCall;
}

void outputVerifyDataCallback (const unsigned char* data, unsigned long dataSize, bool lastCall)
{
	if (fVerifiedStream) {
		if(data && dataSize) {
			fwrite(data, 1, dataSize, fVerifiedStream);
		}
		if (lastCall) {
			fclose(fVerifiedStream);
		}
	}
}

bool verifyDetachedDataCallback(unsigned char*& data, unsigned long& dataSize)
{
	bool lastCall = false;

	if (inputDataToVerifyDetachedData) {
		delete inputDataToVerifyDetachedData;
	}
	unsigned long datatoRead = 128;
	unsigned long dataRead = 0;
	inputDataToVerifyDetachedData = new  unsigned char[datatoRead];
	memset(inputDataToVerifyDetachedData, 0, datatoRead);
```

```
	if (fSignedStreamToVerifyDetached) {
		dataRead = fread(inputDataToVerifyDetachedData, 1, datatoRead, fSignedStreamToVerifyDetachedData);
		if (feof(fSignedStreamToVerifyDetachedData)) {
			fseek(fSignedStreamToVerifyDetachedData, 0, SEEK_SET);
			lastCall = true;
		}
	}
	dataSize = dataRead;
	data = inputDataToVerifyDetachedData;

	return lastCall;
}

bool verifyDetachedEdsDataCallback(unsigned char*& data, unsigned long& dataSize)
{
	bool lastCall = false;

	if (inputDataToVerifyDetached) {
		delete inputDataToVerifyDetached;
	}
	unsigned long datatoRead = 128;
	unsigned long dataRead = 0;
	inputDataToVerifyDetached = new  unsigned char[datatoRead];
	memset(inputDataToVerifyDetached, 0, datatoRead);

	if (fSignedStreamToVerifyDetached) {
		dataRead = fread(inputDataToVerifyDetached, 1, datatoRead, fSignedStreamToVerifyDetached);
		if (feof(fSignedStreamToVerifyDetached)) {
			fseek(fSignedStreamToVerifyDetached, 0, SEEK_SET);
			lastCall = true;
		}
	}
	dataSize = dataRead;
	data = inputDataToVerifyDetached;

	return lastCall;
}

void outputDetachedSignedDataCallback(const unsigned char* data, unsigned long dataSize, bool lastCall)
{
	if (fSignedStreamDetached) {
		if(data && dataSize) {
			fwrite(data, 1, dataSize, fSignedStreamDetached);
		}
		if (lastCall) {
			fclose(fSignedStreamDetached);
		}
	}
}

void testSimpleFunctions()
{
	wchar_t* tspAddress = L"http://tsp.gaz-is.ru/tsp/tsp.srf";
	GisCryptoC::EdsFlag edsFlag = GisCryptoC::IncludeTimeStamp;

	unsigned char* inputData = 0;
	unsigned long inputDataSize = 0;
	FILE* fIn = fopen(".\\Test\\1.txt", "rb");
	if (fIn) {
		fseek(fIn, 0, SEEK_END);
		inputDataSize = ftell(fIn);
		fseek(fIn, 0, SEEK_SET);
```

```
	if(inputDataSize) {
		inputData = new unsigned char[inputDataSize];
		fread(inputData, 1, inputDataSize, fIn);
	}
	fclose(fIn);
}

//Получаем сертификат из хранилища

unsigned char* signerCert = 0;
unsigned long signerCertSize = 0;

GisCryptoC::GisCryptoError error;
if (!GisCryptoC::getCertFromStore(L"MY", signerCert, signerCertSize, error)) {
	GisCryptoC::freeGisCryptoError(error);
}

//Создаем ЭП

GisCryptoC::CreateSignInfo createSignInfo;
memset(&createSignInfo, 0, sizeof(createSignInfo));

createSignInfo.cert = signerCert;
createSignInfo.certSize = signerCertSize;
createSignInfo.tspAddress = tspAddress;
createSignInfo.edsFlag = edsFlag;

unsigned char* signedData= 0;
unsigned long signedDataSize = 0;

if (GisCryptoC::signData(inputData, inputDataSize, createSignInfo, signedData, signedDataSize, error)) {
	FILE* fSigned = fopen(".\\Test\\1.txt.p7s", "wb");
	if (fSigned) {
		if(signedData && signedDataSize) {
			fwrite(signedData, 1, signedDataSize, fSigned);
		}
		fclose(fSigned);
	}
} else {

	//Обработка ошибки

	int* bhCodes = error.bhGiSErrorCodes;
	for (int i = 0; i < error.bhGiSErrorCodesSize; i++) {
		int bhCode = bhCodes[i];
	}
	wchar_t** bhStrings = error.bhGiSErrorStrings;
	for (int i = 0; i < error.bhGiSErrorStringsSize; i++) {
		wchar_t* bhString = error.bhGiSErrorStrings[i];
	}
	unsigned long sysCode = error.systemErrorCode;
	wchar_t* sysString = error.systemErrorText;

	GisCryptoC::freeGisCryptoError(error);
}

GisCryptoC::freeMemory(signerCert);
GisCryptoC::freeMemory(signedData);

//Проверка ЭП
```

```
	unsigned char* inputSignedData = 0;
	unsigned long inputSignedDataSize = 0;

	FILE* fInForVerify = fopen(".\\Test\\1.txt.p7s", "rb");
	if (fInForVerify) {
		fseek(fInForVerify, 0, SEEK_END);
		inputSignedDataSize = ftell(fInForVerify);
		fseek(fInForVerify, 0, SEEK_SET);

		if(inputSignedDataSize) {
			inputSignedData = new unsigned char[inputSignedDataSize];
			fread(inputSignedData, 1, inputSignedDataSize, fInForVerify);
		}
		fclose(fInForVerify);
	}

	GisCryptoC::SignatureInfo* signInfos = 0;
	unsigned long signInfosSize = 0;

	if (GisCryptoC::verifyEds(inputSignedData, inputSignedDataSize, signInfos, signInfosSize, error)) {
		if (signInfos) {
			for (int i = 0; i < signInfosSize; i++) {
				GisCryptoC::SignatureInfo signInfo = signInfos[i];
			}
			GisCryptoC::freeSignatureInfos(signInfos, signInfosSize);
		}
	} else {
		GisCryptoC::freeGisCryptoError(error);
	}

	delete inputSignedData;

	FILE* fInForVerifyExtract = fopen(".\\Test\\1.txt.p7s", "rb");
	if (fInForVerifyExtract) {
		fseek(fInForVerifyExtract, 0, SEEK_END);
		inputSignedDataSize = ftell(fInForVerify);
		fseek(fInForVerifyExtract, 0, SEEK_SET);

		if(inputSignedDataSize) {
			inputSignedData = new unsigned char[inputSignedDataSize];
			fread(inputSignedData, 1, inputSignedDataSize, fInForVerifyExtract);
		}
		fclose(fInForVerifyExtract);
	}

	signInfos = 0;
	signInfosSize = 0;

	unsigned char* inputExtractedData = 0;
	unsigned long inputExtractedDataSize = 0;

	if (GisCryptoC::verifyEdsExractData(inputSignedData, inputSignedDataSize, signInfos, signInfosSize,
inputExtractedData, inputExtractedDataSize, error)) {
		if (signInfos) {
			for (int i = 0; i < signInfosSize; i++) {
				GisCryptoC::SignatureInfo signInfo = signInfos[i];
			}
			FILE* fExtracted = fopen(".\\Test\\1_extracted.txt", "wb");
			if (fExtracted) {
				if(inputExtractedData && inputExtractedDataSize) {
					fwrite(inputExtractedData, 1, inputExtractedDataSize, fExtracted);
					GisCryptoC::freeMemory(inputExtractedData);
```

```
                        }
                        fclose(fExtracted);
                }
                GisCryptoC::freeSignatureInfos(signInfos, signInfosSize);
        }
} else {
        GisCryptoC::freeGisCryptoError(error);
}

delete inputSignedData;

//Шифрование данных

//Получение сертификата из хранилища

unsigned char** receviersCerts = 0;
unsigned long* receiversCertsSizes = 0;
unsigned long receviersCertsCount = 0;

if (!GisCryptoC:: getAllCertFromStore(L"AddressBook", receviersCerts, receiversCertsSizes, receviersCertsCount,
error)) {
        GisCryptoC::freeGisCryptoError(error);
}

//Шифрование

unsigned char* encyptedData= 0;
unsigned long encyptedDataSize = 0;

if (GisCryptoC::encryptData(inputData, inputDataSize, const_cast<const unsigned char**>(receviersCerts),
const_cast<const unsigned long*&>(receiversCertsSizes), receviersCertsCount, encyptedData, encyptedDataSize, error)) {
        FILE* fEncrypted = fopen(".\\Test\\1.txt.p7m", "wb");
        if (fEncrypted) {
                if(encyptedData && encyptedDataSize) {
                        fwrite(encyptedData, 1, encyptedDataSize, fEncrypted);
                }
                fclose(fEncrypted);
        }
} else {
        GisCryptoC::freeGisCryptoError(error);
}

for (int i = 0; i < receviersCertsCount; i++) {
        GisCryptoC::freeMemory(receviersCerts[i]);
}
GisCryptoC::freeMemory(receviersCerts);
GisCryptoC::freeMemory(receiversCertsSizes);
GisCryptoC::freeMemory(encyptedData);
delete inputData;

//Расшифровывание

unsigned char* inputEncryptedData = 0;
unsigned long inputEncryptedDataSize = 0;

FILE* fInForDecrypt = fopen(".\\Test\\1.txt.p7m", "rb");
if (fInForDecrypt) {
        fseek(fInForDecrypt, 0, SEEK_END);
        inputEncryptedDataSize = ftell(fInForDecrypt);
        fseek(fInForDecrypt, 0, SEEK_SET);

        if(inputEncryptedDataSize) {
```

```
                inputEncryptedData = new unsigned char[inputEncryptedDataSize];
                fread(inputEncryptedData, 1, inputEncryptedDataSize, fInForDecrypt);
            }
            fclose(fInForDecrypt);
        }

        unsigned char* receiverCert = 0;
        unsigned long receiverCertSize = 0;

        unsigned char* decryptedData = 0;
        unsigned long decryptedDataSize = 0;

        if (GisCryptoC::decryptData(inputEncryptedData, inputEncryptedDataSize, receiverCert, receiverCertSize,
decryptedData, decryptedDataSize, error)) {
            FILE* fDecrypted = fopen(".\\Test\\1_decrypted.txt", "wb");
            if (fDecrypted) {
                if(decryptedData && decryptedDataSize) {
                    fwrite(decryptedData, 1, decryptedDataSize, fDecrypted);
                }
                fclose(fDecrypted);
            }
        } else {
            GisCryptoC::freeGisCryptoError(error);
        }

        GisCryptoC::freeMemory(receiverCert);
        GisCryptoC::freeMemory(decryptedData);
        delete inputEncryptedData;

}
```

Термины и сокращения

CA	–	Certifying Authority (Сертифицирующая организация, Удостоверяющий центр)
Отделенная ЭП	–	электронная подпись, сформированная в виде отдельного файла.
Сертификат	–	документ на бумажном носителе или электронный документ с ЭП уполномоченного лица удостоверяющего центра, которые включают в себя открытый ключ ЭП и которые выдаются удостоверяющим центром участнику информационной системы для подтверждения подлинности ЭП и идентификации владельца сертификата ключа подписи.
Усовершенствованная ЭП	–	электронная подпись, усовершенствованная, в качестве меры борьбы с общепризнанными угрозами безопасности, добавлением (как необходимое требование) признаков ее (подписи) регламента и доказательств подлинности – таких, как штамп времени, данные об отзыве сертификата и др.
Электронная Подпись (ЭП)	–	реквизит электронного документа, предназначенный для защиты данного электронного документа от подделки, полученный в результате криптографического преобразования информации с использованием закрытого ключа электронной подписи и позволяющий идентифицировать владельца сертификата ключа подписи, а также установить отсутствие искажения информации в электронном документе.