

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

БАКАЛАВРСКАЯ РАБОТА
на тему
**Программные средства поддержки онлайн-
проверки статуса сертификата для
операционной системы QPOS**

Студент	_____	Подлеснов Ярослав Евгеньевич
	(подпись, дата)	(ФИО полностью)
Руководитель	_____	Князев В.Н.
	(подпись, дата)	(фамилия, инициалы)
Нормоконтролёр	_____	Такташкин Д.В.
	(подпись, дата)	(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой	_____	Макарычев П.П.
	(подпись)	(фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК	_____	Попова Н.А.
	(подпись)	(фамилия, инициалы)

Пенза, 2020

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

Кафедра

Вычислительной техники

МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
П.П. Макарычев
«30» декабря 2019 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА**

1. Студент Подлеснов Ярослав Евгеньевич гр. 16ВП1 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы Программные средства поддержки онлайн-проверки статуса сертификата для операционной системы QPOS

Тема утверждена приказом ПГУ № 1488/о от 30 " декабря 2019 г.

3. Руководитель работы к.т.н., доцент кафедры «МОиПЭВМ» Князев В.Н.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

- 1) Разработать программные средства сервера онлайн-проверки статуса сертификата, реализующие протокол состояния сетевого сертификата в соответствии со стандартом RFC 2560 в рамках инфраструктуры открытых ключей.
- 2) Среда выполнения: операционная система QPOS,
 Языковые средства разработки: C++, UML.
 Инструментальные средства: Microsoft Visual Studio, nginx, Software Ideas Modeler, Microsoft Project
- 3) Технологии разработки RUP, ООП.
 Основные функции сервера онлайн-проверки:
 - а) ответы на запросы о состоянии сертификатов;
 - б) установка, проверка и обновление списков отозванных сертификатов;
 - в) установка собственного и корневого сертификата;
 - г) настройка средств электронной подписи ответов.

5. Перечень подлежащих разработке вопросов:

- 1) *Анализ предметной области и постановка задачи;*
- 2) *Проектирование;*
- 3) *Реализация;*
- 4) *Контроль качества программного обеспечения;*
- 5) *Оформление пояснительной записки.*

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	<i>Анализ предметной области и постановка задачи</i>	15%	13.04.2020– 21.04.2020	
2	<i>Проектирование</i>	25%	22.04.2020– 05.05.2020	
3	<i>Реализация</i>	30%	06.05.2020– 30.05.2020	
4	<i>Контроль качества программного обеспечения</i>	20%	31.05.2020– 02.06.2020	
5	<i>Оформление пояснительной записки</i>	10%	03.06.2020– 13.06.2020	

Дата выдачи задания " 30 " декабря 2019

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан факультета _____
(подпись, дата)

Реферат

Пояснительная записка содержит 151 лист, 31 рисунок, 9 таблиц, 21 листинг, 30 использованных источников, 6 приложений.

C++, СЛУЖБА, QPOS, UML, ООП, СЕРВЕР, ЦИФРОВОЙ СЕРТИФИКАТ, УДОСТОВЕРЯЮЩИЙ ЦЕНТР, РОССИЙСКИЕ КРИПТОГРАФИЧЕСКИЕ АЛГОРИТМЫ, OCSP, ASN, PKI

Объектом разработки являются программные поддержки онлайн-проверки статуса сертификата для операционной системы QPOS.

Цель работы – разработка средств поддержки онлайн-проверки статуса сертификата для операционной системы QPOS, реализующих протокол состояния сетевого сертификата.

Результатом разработки является исполняемый файл приложения «OCSPService.qrx», предназначенный для работы в режиме, аналогичном службе Windows, под управлением операционной системы QPOS.

Проектирование проводилось с помощью языка моделирования UML с использованием среды разработки диаграмм «Software Ideas Modeler».

Разработка проводилась на языке программирования C++. В качестве среды разработки использовалась IDE Visual Studio 2019 Enterprise. Разработка проводилась с использованием операционной системы Windows 10. Среда исполнения – операционная система QPOS.

Был получен акт о внедрении данного продукта в состав программного комплекса, разрабатываемого ООО НТП «Криптософт».

					ПГУ 09.03.04 – 08БР161.22 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Программные средства поддержки онлайн-проверки статуса сертификата для операционной системы QPOS. Пояснительная записка	Лит.	Лист	Листов
Разраб.		Подлеснов Я.Е.						
Провер.		Князев В.Н					4	151
						Группа 16ВП1		
Н. Контр.		Такташкин Д.В.						
Утверд.		Макарычев П.П						

Содержание

Введение.....	7
1 Анализ предметной области и постановка задачи.....	9
1.1 Анализ предметной области	9
1.2 Анализ программных продуктов аналогичного назначения	13
1.2.1 Анализ компонента «КриптоПро OCSP Server»	13
1.2.2 Анализ средства «OCSP Responder»	14
1.2.3 Анализ средства «EJBCA OCSP»	15
1.2.4 Результат анализа программ аналогов	16
1.3 Анализ требований.....	16
1.3.1 Функциональные требования.....	16
1.3.2 Нефункциональные требования	21
1.4 Планирование разработки и оценка бюджета.....	22
2 Проектирование средств поддержки онлайн-проверки статуса сертификата для операционной системы QPOS	30
2.1 Проектирование архитектуры.....	30
2.2 Проектирование внешних интерфейсов	32
2.3 Проектирование структур данных	35
2.4 Проектирование алгоритмов.....	37
3 Реализация средств поддержки онлайн-проверки статуса сертификата для операционной системы QPOS	40
4 Тестирование программных средств.....	46
4.1 Функциональное тестирование.....	46
4.2 Модульное тестирование.....	48
4.3 Нагрузочное тестирование	50
4.4 Метрики кода.....	52
4.4.1 Цикломатическая метрика Мак-Кейба.	52
4.4.2 Метрика Холстеда.	55
4.4.3 Метрика Чепина	57
Заключение	59

Список использованных источников	60
Приложение А. Исходный код.....	64
Приложение Б. Конфигурация сервера nginx.....	107
Приложение В. Результаты тестирования	109
Приложение Г. Исходный код функций тестирования.....	127
Приложение Д. Результаты нагрузочного тестирования.....	147
Приложение Е. Акт о внедрении	150

Введение

В настоящее время растет потребность к различным информационным сервисам. Это связано с тем, что информационные технологии занимают все большее место в жизни человека. Применение электронных документов помогает перевести общественные отношения на качественно другой уровень, однако вместе с тем поднимает вопросы информационной безопасности.

Значимость и приоритетность развития информационных технологий в Российской Федерации подтверждается принятием в 2017 году Государственной программы под названием «Цифровая экономика Российской Федерации».

Для разработки и реализацию мероприятий программа определяет такие основополагающие принципы информационной безопасности, как: «использование российских технологий обеспечения целостности, конфиденциальности, аутентификации и доступности передаваемой информации и процессов ее обработки», «преимущественное использование отечественного программного обеспечения и оборудования» и «применение технологий защиты информации с использованием российских криптографических стандартов» [1, стр. 13].

Помимо Государственной программы следует упомянуть Указ Президента Российской Федерации от 9 мая 2017 г. № 203 «О стратегии развития информационного общества в Российской Федерации на 2017 – 2030 годы», в котором сказано, что «Для устойчивого функционирования информационной инфраструктуры Российской Федерации необходимо ... обеспечить использование российских криптоалгоритмов и средств шифрования при электронном взаимодействии федеральных органов» [2, стр. 12].

Важными инструментами информационной безопасности являются механизмы электронной подписи и механизмы шифрования данных [3]. Для обеспечения обоих механизмов широко применяются удостоверяющие центры.

Главный потребитель услуг удостоверяющих центров – юридические лица, использующие электронные подписи для нужд своей организации. Рынок электронных подписей отличается устойчивый рост в течение нескольких лет [4]. Динамику роста можно увидеть на диаграмме, представленной на рисунке 1.

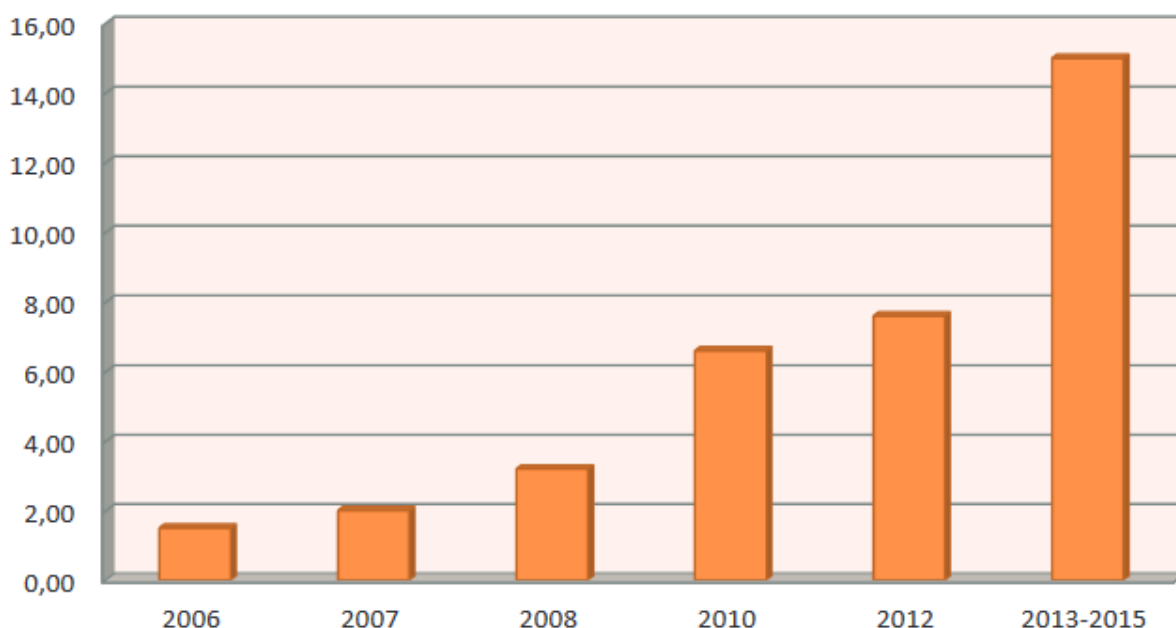


Рисунок 1 – Рост рынка электронных подписей в России, млрд руб.

Из представленных данных следует, что разработка отечественного программного обеспечения для удостоверяющих центров, которые являются ключевым звеном в современных криптографических системах [5], является актуальной и перспективной задачей.

Удостоверяющие центры поддерживают ряд дополнительных протоколов, которые позволяют получать расширенную информацию о цифровых сертификатах. Одним из важнейших таких протоколов является протокол состояния сетевого сертификата [5].

С учетом вышесказанного можно сделать вывод, что цель данной работы, заключающаяся в разработке средств поддержки онлайн-проверки статуса сертификата для операционной системы QPOS, является актуальной задачей.

1 Анализ предметной области и постановка задачи

1.1 Анализ предметной области

Конфиденциальность является одним из важнейших условий для множества сценариев работы в электронных сетях. На сегодняшний день существуют индустриальные решения обеспечения конфиденциальности на основе практического использования криптографических средств [6].

На данный момент крайне популярна криптографическая система с открытым ключом (разновидность асимметричного шифрования) [3]. В данной системе открытый ключ передаётся по открытому (то есть незащищённому, доступному для наблюдения) каналу и используется для проверки электронной подписи и для шифрования сообщения. Для генерации электронной подписи и для расшифровки сообщения используется закрытый ключ [6].

Приведём определения открытого и закрытого ключа.

Открытый ключ – криптографический ключ, который связан с секретным с помощью особого математического соотношения. Открытый ключ известен всем другим пользователям системы и предназначен для проверки электронной цифровой подписи и расшифрования, позволяет определить автора подписи и достоверность электронного документа, но не позволяет вычислить секретный ключ [5].

Секретный (закрытый) ключ – ключ асимметричной ключевой пары, который доступен только одному пользователю системы и хранится им в тайне. В системе цифровой подписи определяет преобразование подписи, в асимметричной системе шифрования определяет преобразование расшифрования.

Пара ключей (ключевая пара) – открытый ключ, используемый в криптосистеме с открытыми ключами, и соответствующий ему секретный ключ [5].

После того как шифрование с открытыми ключами приобрело популярность, появилась потребность в цифровых сертификатах [7].

Цифровой сертификат (сертификат, сетевой сертификат, сертификат открытого ключа) – цифровой документ, содержащий открытый ключ, информацию о владельце ключа, области применения ключа, подписанный выдавшим его удостоверяющим центром и подтверждающий принадлежность открытого ключа владельцу [5].

Сертификат и соответствующий ему секретный ключ позволяют идентифицировать их владельца всем, кому это необходимо. Универсальное применение сертификатов обеспечивает стандарт Международного Союза по телекоммуникациям X.509, который является базовым и поддерживается целым рядом протоколов безопасности [8].

Стандарт X.509 задает формат цифрового сертификата. Основными атрибутами сертификата являются имя и идентификатор субъекта, информация об открытом ключе субъекта, имя, идентификатор и цифровая подпись удостоверяющего центра, серийный номер, версия и срок действия сертификата, информация об алгоритме подписи и др. [9].

Выдачу сертификатов производит удостоверяющий центр.

Удостоверяющий центр (Certificate Authority (CA)) – это доверенное лицо (физическое или юридическое), которое выпускает, публикует, аннулирует сертификаты, приостанавливает их действие [10].

Важно, что цифровой сертификат содержит цифровую подпись на основе секретного ключа удостоверяющего центра. Подлинность сертификата можно проверить с помощью открытого ключа удостоверяющего центра.

Однако одного наличия сертификата недостаточно. Действительно защищенный обмен сообщениями, надежная идентификация и электронная коммерция невозможны без инфраструктуры открытых ключей – Public Key Infrastructure (PKI) [5].

PKI представляет собой совокупность сервисов для управления ключами и цифровыми сертификатами пользователей, программ и систем. В них входят сервисы выпуска (издания), отзыва (аннулирования) и другие операции.

Удостоверяющий центр является одним из основных компонентов инфраструктуры открытых ключей. Как выпуск, так и отзыв сертификата выполняется именно с его помощью.

Аннулирование сертификата – признание сертификата недействительным в период его действия в случаях компрометации секретного ключа или изменения атрибутов сертификата с момента его выпуска (например, при изменении имени пользователя) [5].

Удостоверяющий центр производит аннулирование сертификата, заноса его серийный номер в список аннулированных сертификатов.

Список аннулированных сертификатов (САС, список отозванных сертификатов, Certificate Revocation List (CRL)) – список сертификатов, которые удостоверяющий центр пометил как отозванные. Публикуется удостоверяющим центром на регулярной основе в публичных источниках.

Списки отозванных сертификатов обновляются удостоверяющими центрами с заданной частотой. К сожалению, так как списки отзывов содержат серийные номера всех отозванных сертификатов, их использование оказывается весьма ресурсоемким. А так как выпускаются списки аннулированных сертификатов с заданной частотой, они могут содержать устаревшую, неактуальную информацию. Попыткой решения данной проблемы является введение разностных списков, которые представляют собой САС, содержащий только те данные, которые не успели войти в основной список. Однако, разностные списки только частично решают проблему [10].

В связи с обозначенными недостатками применения САС, был разработан протокол онлайн-проверки статуса сертификата.

Протокол состояния сетевого сертификата (Online Certificate Status Protocol (OCSP)) – протокол, позволяющий проверить статус сертификатов в режиме реального времени [10]. Программные средства сервера, реализующего данный протокол, являются объектом данной разработки.

Сервер онлайн-проверки статуса сертификата обычно является одним из компонентов удостоверяющего центра. Сервер возвращает подписанный ответ со статусом сертификата, указанного в запросе. Если он не может обработать запрос, он может вернуть код ошибки [11].

Особенности OCSP описаны в стандарте RFC 2560 «X.509 Internet Public Key Infrastructure Certificate Status Protocol – OCSP» [11]. В данном и в зависимых от данного стандартах указаны особенности работы протокола и применяемые структуры данных.

Структуры данных, описанные в данных стандартах, являются важными сущностями, необходимыми для работы данной системы.

На основе полученных данных построим модель предметной области в виде диаграммы классов, приведенную на рисунке 2.

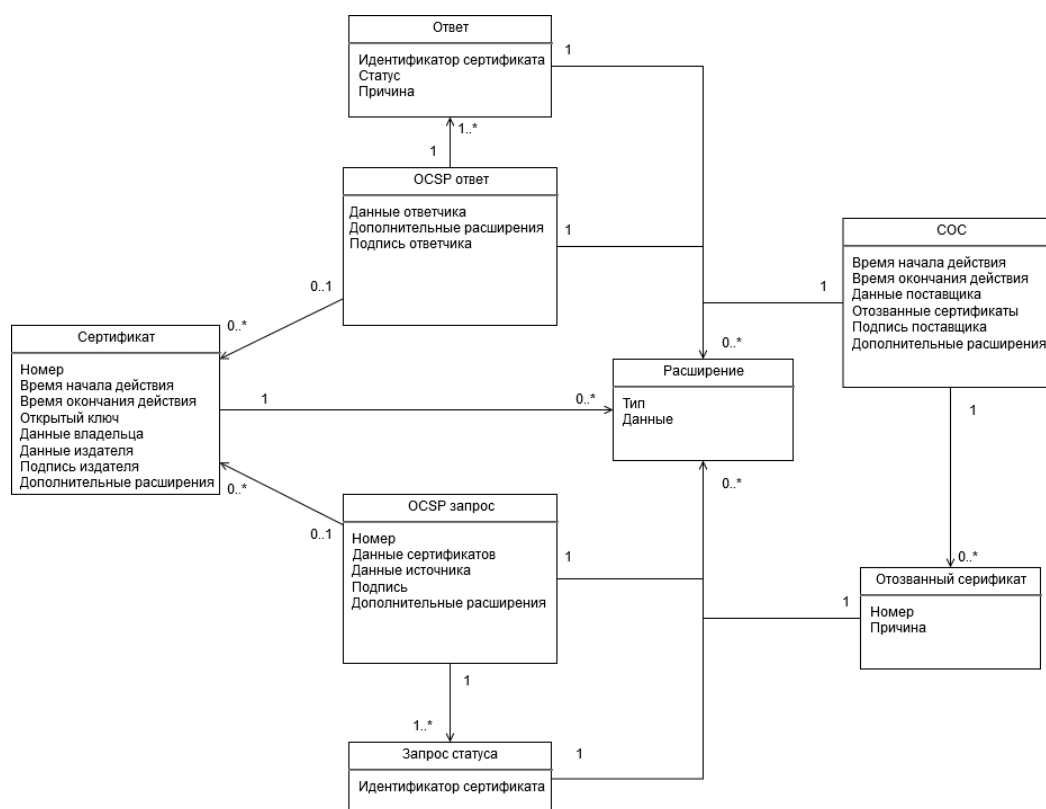


Рисунок 2 – Модель предметной области

На модели представлено укрупненное представление структур, описанных в стандарте.

1.2 Анализ программных продуктов аналогичного назначения

Разрабатываемые программные средства не являются уникальной разработкой на данный момент времени. Существует ряд аналогов данного сервера онлайн-проверки статуса цифрового сертификата. Одним из аналогов является компонент программно-аппаратного комплекса «КриптоПро OCSP» «КриптоПро OCSP Server». Также рассмотрим «OCSP Responder» и Enterprise Java Beans Certificate Authority (EJBICA), в составе которого имеется OCSP Responder.

1.2.1 Анализ компонента «КриптоПро OCSP Server»

«КриптоПро OCSP Server» входит в состав программно-аппаратного комплекса (ПАК) «КриптоПро OCSP» разработанного компанией КриптоПро. Данный комплекс в свою очередь входит в состав ПАК «Службы УЦ», осуществляющего по запросам пользователей предоставление информации о статусе сертификатов ключей подписей, посредством реализации протокола Online Certificate Status Protocol (OCSP) и предоставление меток времени, посредством реализации протокола Time-Stamp Protocol (TSP). ПАК «КриптоПро OCSP», является первым в Российской Федерации специализированным программно-аппаратным комплексом, успешно прошедшим сертификационные испытания и получившим сертификат соответствия ФСБ России для такого рода продуктов [13].

Перечислим основные особенности «КриптоПро OCSP Server»:

- функционирует под управлением операционных систем семейств Windows Server 2003 (x86 или x64), Windows Server 2008 (x86 или x64), Windows Server 2008 R2;
- реализует протокол OCSP поверх HTTP в соответствии с RFC 2560 с учётом использования российских криптографических алгоритмов;

- использует встроенный веб-сервер Microsoft Internet Information Services (IIS), поддерживающий различные методы аутентификации и протокол TLS (SSL);
- поддерживает развёртывание нескольких экземпляров службы на одном компьютере;
- реализует разграничение доступа по списку контроля доступа, что позволяет задать ролевое разграничение доступа;
- поддерживает одновременное использование нескольких криптопровайдеров (cryptography service provider, CSP) на разных экземплярах службы;
- устанавливается с помощью Windows Installer [13].

Следует отметить следующие недостатки данной системы: «КриптоПро OCSP Server» является платной системой, имеет закрытый исходный код, поэтому не может быть адаптирована для использования под управлением операционной системы QPOS.

1.2.2 Анализ средства «OCSP Responder»

«OCSP Responder» – это компонент, разработанный в компании «Криптософт». Данный компонент реализует протокол OCSP в соответствии с RFC 2560 и поддерживает использование российских криптографических алгоритмов.

Данный компонент является продуктом с закрытым исходным кодом, подробная информация о нем отсутствует в открытом доступе.

«OCSP Responder» может функционировать под управлением операционной системы QPOS, однако его пропускная способность недостаточна, что и послужило основанием разработки данных программных средств.

1.2.3 Анализ средства «EJBCA OCSF»

EJBCA (Enterprise Java Beans Certificate Authority) – это программные средства удостоверяющего центра для инфраструктуры PKI с открытым исходным кодом. Они распространяются по лицензии GNU GPL. Написаны на Java и могут быть установлены на любую платформу, в которой имеются необходимые компоненты. В составе данного инструмента также находится реализация протокола OCSF [14].

Перечислим основные особенности компонента OCSF в составе данного продукта:

- вместо списка отозванных сертификатов, выданного центром сертификации, используется собственная база данных с информацией о состоянии сертификата;
- реализует RFC 2560, RFC 6960 и RFC 5019;
- не зависит от списков, отозванных сертификатов, информация о состоянии может быть обновлена в режиме реального времени;
- поддерживает PKCS#11;
- встроенная проверка работоспособности используется балансировщиками нагрузки и для мониторинга.
- может отвечать на хорошие или неизвестные несуществующие сертификаты с различной конфигурацией на основе URI запроса.
- линейная масштабируемость для повышения производительности и высокой доступности за счет добавления нескольких узлов.
- высокая производительность, более пятисот запросов в секунду на одном сервере [14].

Важными недостатками данной системы является отсутствие поддержки российских криптографических алгоритмов и невозможность функционирования под управлением операционной системы QPOS.

1.2.4 Результат анализа программ аналогов

На основании проведенного анализа можно сделать вывод, что рассмотренные аналоги не соответствуют обозначенным требованиям, либо имеют иные значимые недостатки.

Компонент «КриптоПро OSCP Server» удовлетворяет часть требований, однако он не может быть адаптирован для использования под управлением операционной системы QPOS.

Средство «OCSP Responder» удовлетворяет всем основным требованиям, кроме языка разработки, однако его производительность недостаточна.

Средство «EJBCA OSCP» удовлетворяет часть требований, однако оно не может быть адаптирован для использования под управлением операционной системы QPOS, а также не имеет поддержки российских криптографических алгоритмов.

Таким образом, было принято решение в ходе данной работы реализовать собственные программные средства онлайн-проверки статуса сертификата.

1.3 Анализ требований

1.3.1 Функциональные требования

На основе данных, полученных при анализе предметной области, была построена диаграмма вариантов использования. Она представлена на рисунке 3.

Разработка данной диаграммы и других UML диаграмм будем проводить с помощью среды разработки диаграмм «Software Ideas Modeler».

На диаграмме представлены внешние сущности, взаимодействующие с системой. Опишем их подробнее.

«Клиент» представляет собой произвольное программное средство, соответствующее стандарту OSCP, в роли клиента.



Рисунок 3 – Диаграмма вариантов использования сервера онлайн-проверки статуса сертификата

«Администратор» представляет собой программное средство, производящее управление данной системой с помощью предлагаемого интерфейса.

«Хранилище сертификатов» представляет собой систему, предоставляющую сертификаты и списки отозванных сертификатов с помощью специального интерфейса.

Спецификация вариантов использования «Остановить сервер» и «Запустить сервер» приведена на рисунках 4 и 5.

Прецедент: Остановить сервер
ID: 1
Краткое описание: Администратор останавливает Сервер
Актеры: Администратор
Предусловия: Нет
Основной поток: <ol style="list-style-type: none"> 1. Администратор запрашивает остановку сервера. 2. Сервер останавливает работу. 3. Сервер сообщает об остановке.
Постусловия: Сервер остановлен

Рисунок 4 – Спецификация варианта использования «Остановить сервер»

Прецедент: Запустить сервер
ID: 2
Краткое описание: Администратор запускает Сервер
Актеры: Администратор
Предусловия: Нет
Основной поток: <ul style="list-style-type: none"> 1. Include (Установить сертификат) 2. Include (Установить средство электронной подписи) 3. Include (Установить список отозванных сертификатов) 4. Сервер сообщает о готовности
Постусловия: Сервер запущен

Рисунок 5 – Спецификация варианта использования «Запустить сервер»

Ключевой функциональностью данных программных средств является вариант использования «Проверить статус сертификата». Приведем спецификацию данного варианта использования на рисунке 6.

Прецедент: Проверить статус сертификата
ID: 3
Краткое описание: Клиент запрашивает статус сертификата
Актеры: Клиент, Хранилище сертификатов
Предусловия: Сервер запущен и настроен
Основной поток: <ul style="list-style-type: none"> 1. Прецедент начинается, когда Клиент делает запрос. 2. Если запрос правильный. <ul style="list-style-type: none"> 2.1. Сервер запрашивает сертификат у Хранилища сертификатов. 2.2. Если сертификат найден <ul style="list-style-type: none"> 2.2.1. Сервер проверяет сертификат 2.2.2. Сервер отправляет Клиенту статус сертификата 2.3. Иначе Сервер отправляет Клиенту неизвестный статус сертификата 3. Иначе <ul style="list-style-type: none"> 3.1. Сервер отправляет Клиенту сообщение об ошибке
Постусловия: Клиенту отправлен ответ

Рисунок 6 – Спецификация варианта использования «Проверить статус сертификата»

Диаграмма последовательности, поясняющая вариант использования «Проверить статус сертификата», приведена на рисунке 7.

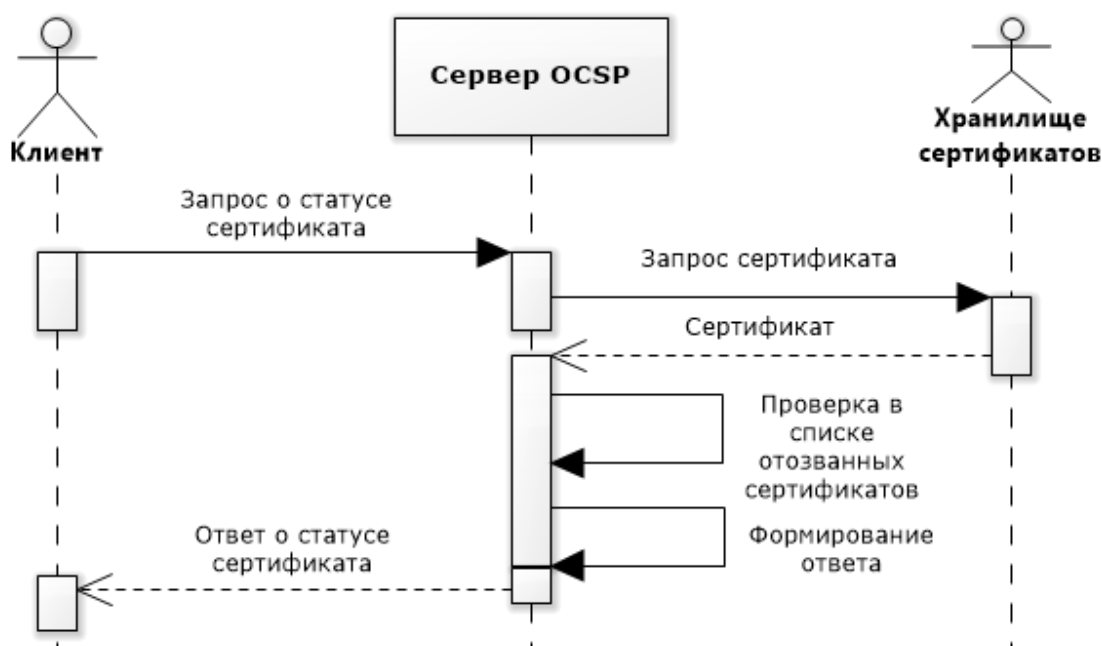


Рисунок 7 – Диаграмма последовательности для варианта использования «Проверить статус сертификата»

Спецификации оставшихся прецедентов приведены на рисунках 8-10.

Прецедент: Установить сертификат
ID: 4
Краткое описание: Администратор устанавливает в Сервер новый сертификат сервера онлайн-проверки статуса сертификата
Актеры: Администратор, Хранилище сертификатов
Предусловия: Нет
Основной поток: <ol style="list-style-type: none"> 1. Администратор отправляет Серверу номер устанавливаемого сертификата. 2. Сервер запрашивает сертификат у Хранилища сертификатов. 3. Сервер устанавливает сертификат. 4. Сервер оповещает Администратора об успехе операции.
Постусловия: Сертификат установлен

Рисунок 8 – Спецификация варианта использования «Установить сертификат»

Прецедент: Установить список отозванных сертификатов
ID: 5
Краткое описание: Администратор заносит в Сервер список отозванных сертификатов
Актеры: Администратор
Предусловия: Настройка средства электронной подписи завершена
Основной поток: <ol style="list-style-type: none"> 1. Администратор отправляет Серверу список отозванных сертификатов. 2. Сервер проверяет и устанавливает список отозванных сертификатов 3. Сервер оповещает Администратора об успехе операции 4. Если есть разностный список отозванных сертификатов. <ol style="list-style-type: none"> 4.1. Администратор отправляет Серверу разностный список отозванных сертификатов. 4.2. Сервер проверяет и устанавливает разностный список отозванных сертификатов 4.3. Сервер оповещает Администратора об успехе операции
Постусловия: СОС установлен

Рисунок 9 – Спецификация варианта использования «Установить список отозванных сертификатов»

Прецедент: Установить средство электронной подписи
ID: 6
Краткое описание: Администратор устанавливает в Сервер и настраивает средства, позволяющие производить электронную подпись
Актеры: Администратор
Предусловия: Сертификат установлен
Основной поток: <ol style="list-style-type: none"> 1. Администратор запрашивает список доступных носителей. 2. Сервер предоставляет список доступных носителей. 3. Администратор выбирает носитель. 4. Администратор авторизуется в носителе. 5. Сервер ищет в носителе ключевую пару. 6. Сервер сообщает Администратору об успехе операции.
Постусловия: Настройка средства электронной подписи завершена

Рисунок 10 – Спецификация варианта использования «Установить средство электронной подписи»

В данном разделе были выделены и представлены в виде диаграммы вариантов использования основные функции, разрабатываемых программных средств, а также представлена их спецификация.

1.3.2 Нефункциональные требования

Помимо обозначенных ранее функциональных требований, к разрабатываемым программным средствам было предъявлено ряд нефункциональных.

Программное средство должно работать под управлением операционной системы QPOS. Это требование является ключевым фактором разработки.

Программное средство должно быть разработано на языке программирования C++ в среде Visual Studio, что обусловлено высокой скоростью работы данного языка, а также наличием компонентов, пригодных для использования при разработке данных программных средств.

Необходимо обеспечить высокую пропускную способность системы.

Необходимо обеспечить взаимодействие с существующими компонентами и сервисами, используя специальный протокол взаимодействия. Для выполнения данного требования следует применить библиотеку «NetCoreNative» для сетевого взаимодействия с сущностями «Администратор» и «Хранилище сертификатов», а также предоставить заданный интерфейс. Подробности требуемого интерфейса и применения библиотеки опишем позднее.

Необходимо обеспечить исполнения серверной части протокола OSCP, описанного в стандарте RFC 2560 [12], используя протокол HTTP в качестве транспорта.

Алгоритмы формирования и проверки электронной подписи должны соответствовать ГОСТ Р 34.10-2012 [15]. Для выполнения данного требования необходимо применить существующую библиотеку

шифрования, исполняющую данное требование, реализованную в соответствии со стандартом PKCS#11 [16, 17].

Необходимо обеспечить надежность функционирования системы.

1.4 Планирование разработки и оценка бюджета

Для определения трудоемкости и стоимости проекта необходимо провести его грамотное планирование. Для этого были определены список задач, решение которых необходимо для создания приложения, и перечень ресурсов, с помощью которых эти задачи будут решаться.

В таблице 1 представлены задачи, представляющие из себя этапы создания проекта.

Таблица 1 – Перечень задач проекта

№	Название задачи	Длительность	Предшественники
1	Начало проекта	-	-
2	Анализ требований	-	-
3	Поиск информации о предметной области	2 дня	1
4	Изучение и анализ информации о предметной области	6 дней	3
5	Изучение и анализ программ аналогов	3 дня	3
6	Формирование уточненных требований к проекту	4 дня	4; 5
7	Анализ требований завершен	-	6
8	Проектирование проекта онлайн-проверки статуса сертификата для операционной системы QPOS	-	-
9	Проектирование общей архитектуры проекта	6 дней	7

Продолжение таблицы 1

№	Название задачи	Длительность	Предшественники
10	Анализ возможных технологий	5 дней	7
11	Выбор технологий для реализации	2 дня	10
12	Проектирование архитектуры компонентов	5 дней	9;11
13	Проектирование проекта завершено	-	12
14	Реализация проекта онлайн-проверки статуса сертификата для операционной системы QPOS	-	-
15	Реализация библиотеки компонентов	11 дней	13
16	Реализация и настройка окружения	9 дней	13
17	Реализация автоматических тестов	7 дней	13
18	Реализация системы проверки статуса сертификата	12 дней	15
19	Реализация проекта завершена	-	16;17;18
20	Отладка и тестирование	-	-
21	Тестирование библиотеки компонентов	5 дней	15; 17
22	Отладка работы системы	9 дней	18
23	Исправление ошибок	5 дней	22
24	Тестирование системы в целом	6 дней	19;21;23
25	Отладка и тестирование завершены	-	24
26	Внедрение программного средства	-	-
27	Включение системы в инфраструктуру	3 дня	25

Окончание таблицы 1

28	Отладка работы со внешними компонентами	4 дня	27
29	Поиск и исправление ошибок при внедрении	4 дня	27
30	Внедрение завершено	-	28;29
31	Конец проекта	-	-

Из данной таблицы можно сделать вывод, что для реализации программного проекта нужно решить 31 задачу.

В таблице 2 представлен список затрачиваемых ресурсов.

Таблица 2 – Список ресурсов проекта

Название ресурса	Тип	Стандартная ставка	Ставка сверхурочных
Программист1	Трудовой	80 Р\ч	100 Р\час
Программист2	Трудовой	80 Р\ч	100 Р\час
Стенд с операционной системой QPOS	Трудовой	10 Р\ч	10 Р\ч

Распределение ресурсов по задачам приведено в таблице 3.

Таблица 3 – Назначения ресурсов

№	Название задачи	Ресурс	Единицы
1	Начало проекта		
2	Анализ требований		
3	Поиск информации о предметной области	Программист1	100%
4	Изучение и анализ информации о предметной области	Программист2	100%

Продолжение таблицы 3

№	Название задачи	Ресурс	Единицы
5	Изучение и анализ программ аналогов	Программист1	100%
6	Формирование уточненных требований к проекту	Программист1	100%
7	Анализ требований завершен		
8	Проектирование проекта онлайн-проверки статуса сертификата для операционной системы QPOS		
9	Проектирование общей архитектуры проекта	Программист1	100%
10	Анализ возможных технологий	Программист2	100%
11	Выбор технологий для реализации	Программист2	100%
12	Проектирование архитектуры компонентов	Программист1	100%
13	Проектирование проекта завершено		
14	Реализация проекта онлайн-проверки статуса сертификата для операционной системы QPOS		
15	Реализация библиотеки компонентов	Программист2	100%
16	Реализация и настройка окружения	Программист1	100%
17	Реализация автоматических тестов	Программист1	100%
18	Реализация системы проверки статуса сертификата	Программист2	100%
19	Реализация проекта завершена		
20	Отладка и тестирование		

Окончание таблицы 3

№	Название задачи	Ресурс	Единицы
21	Тестирование библиотеки компонентов	Программист1 Стенд с операционной системой QPOS	100% 50%
22	Отладка работы системы	Программист2 Стенд с операционной системой QPOS	100% 50%
23	Исправление ошибок	Программист2 Стенд с операционной системой QPOS	100% 50%
24	Тестирование системы в целом	Программист2 Стенд с операционной системой QPOS	100% 50%
25	Отладка и тестирование завершены	Программист2 Стенд с операционной системой QPOS	100% 50%
26	Внедрение программного средства		
27	Включение системы в инфраструктуру	Программист1 Стенд с операционной системой QPOS	100% 50%
28	Отладка работы со внешними компонентами	Программист2 Стенд с операционной системой QPOS	100% 50%
29	Поиск и исправление ошибок при внедрении	Программист2 Стенд с операционной системой QPOS	100% 50%
30	Внедрение завершено		
31	Конец проекта		

На представленных таблицах можно увидеть, что решением задач будут заниматься 2 программиста, при этом, так как выполнение одной задачи несколькими программистами вызывает сложности, каждой задачей занимается только один программист. Также на последних этапах потребуется стенд с операционной системой QPOS, который будет использоваться программистами совместно.

На рисунке 11 изображено введенное в систему управления проектами Microsoft Project содержание таблицы 1 с вычисленными датами начала и завершения каждой задачи.

За начало проекта была принята дата 16.04.2020.

	Режим задачи	Название задачи	Длительность	Начало	Окончание	Предшественники
0		Проект1	74 дней	Чт 16.04.20	Вт 28.07.20	
1		Начало проекта	0 дней	Чт 16.04.20	Чт 16.04.20	
2		Анализ требований	12 дней	Чт 16.04.20	Пн 04.05.20	
3		Поиск информации о предметной области	2 дней	Чт 16.04.20	Пт 17.04.20	
4		Изучение и анализ информации о предметной области	6 дней	Пн 20.04.20	Пн 27.04.20	3
5		Изучение и анализ программ аналогов	3 дней	Пн 20.04.20	Ср 22.04.20	3
6		Формирование уточненных требований к проекту	4 дней	Вт 28.04.20	Пт 01.05.20	4;5
7		Анализ требований завершен	0 дней	Пн 04.05.20	Пн 04.05.20	6
8		Проектирование проекта	12 дней	Пн 04.05.20	Вт 19.05.20	
9		Проектирование общей архитектуры проекта	6 дней	Пн 04.05.20	Пн 11.05.20	7
10		Анализ возможных технологий	5 дней	Пн 04.05.20	Пт 08.05.20	7
11		Выбор технологий для реализации	2 дней	Пн 11.05.20	Вт 12.05.20	10
12		Проектирование архитектуры компонентов	5 дней	Ср 13.05.20	Вт 19.05.20	11;9
13		Проектирование проекта завершено	0 дней	Вт 19.05.20	Вт 19.05.20	12
14		Реализация проекта	23 дней	Ср 20.05.20	Пт 19.06.20	
15		Реализация библиотеки компонентов	11 дней	Ср 20.05.20	Ср 03.06.20	13
16		Реализация и настройка окружения	9 дней	Пт 29.05.20	Ср 10.06.20	13
17		Реализация автоматических тестов	7 дней	Ср 20.05.20	Чт 28.05.20	13
18		Реализация системы проверки статуса сертификата	12 дней	Чт 04.06.20	Пт 19.06.20	15
19		Реализация проекта завершена	0 дней	Пт 19.06.20	Пт 19.06.20	16;17;18
20		Отладка и тестирование	27 дней	Чт 11.06.20	Пт 17.07.20	
21		Тестирование библиотеки компонентов	5 дней	Чт 11.06.20	Ср 17.06.20	15;17
22		Отладка работы системы	9 дней	Пн 22.06.20	Чт 02.07.20	18
23		Исправление ошибок	5 дней	Пт 03.07.20	Чт 09.07.20	22
24		Тестирование системы в целом	6 дней	Пт 10.07.20	Пт 17.07.20	19;21;23
25		Отладка и тестирование завершены	0 дней	Пт 17.07.20	Пт 17.07.20	24
26		Внедрение программного средства	7 дней	Пн 20.07.20	Вт 28.07.20	
27		Включение системы в инфраструктуру	3 дней	Пн 20.07.20	Ср 22.07.20	25
28		Отладка работы со внешними компонентами	4 дней	Чт 23.07.20	Вт 28.07.20	27
29		Поиск и исправление ошибок при внедрении	4 дней	Чт 23.07.20	Вт 28.07.20	27
30		Внедрение завершено	0 дней	Вт 28.07.20	Вт 28.07.20	28;29
31		Конец проекта	0 дней	Чт 16.04.20	Чт 16.04.20	

Рисунок 11 – Задачи проекта, введенные в среду Microsoft Project

На рисунке 12 представлена построенная средой Microsoft Project диаграмма Ганта.

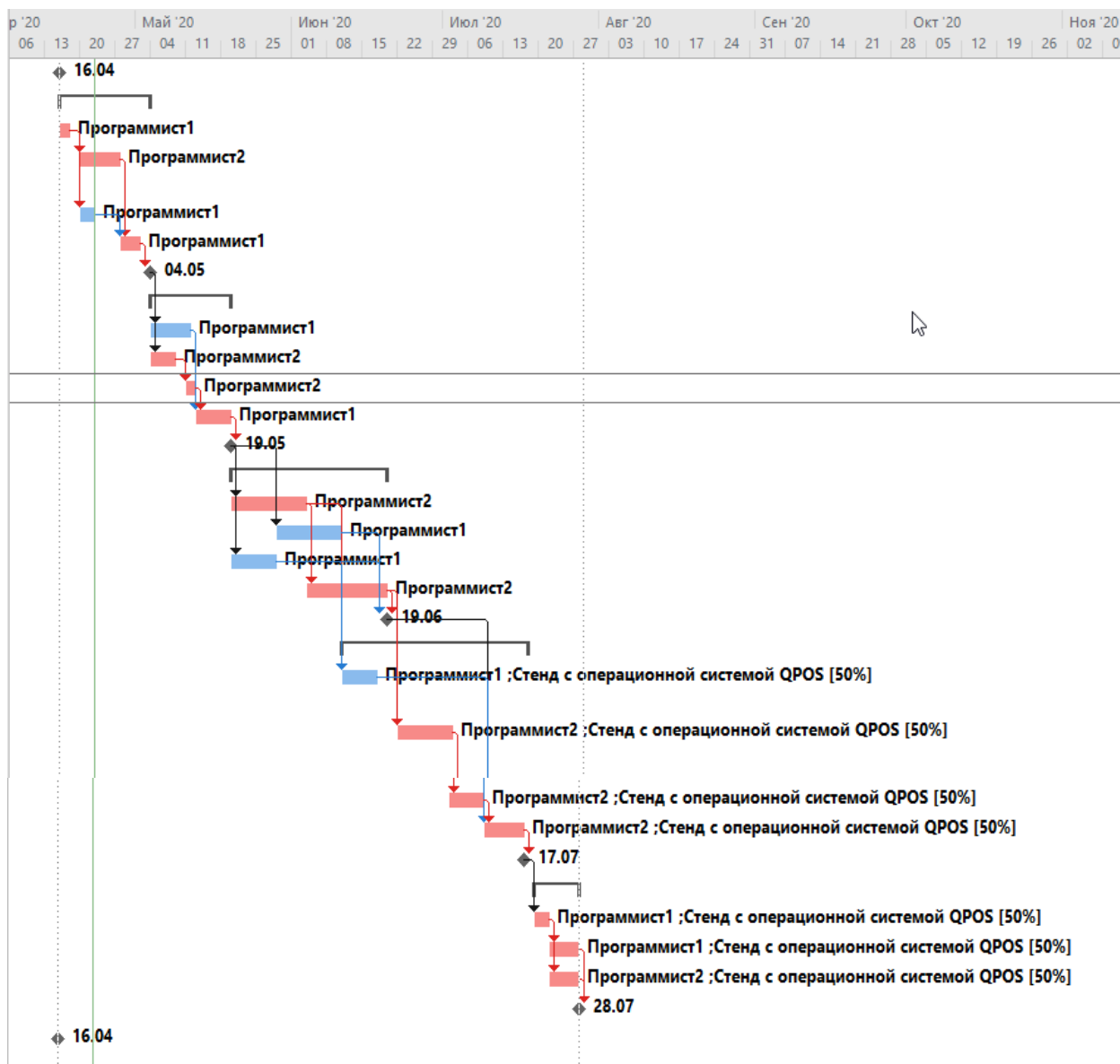


Рисунок 12 – Диаграмма Ганта

Из диаграммы видно, что вычисленный программой критический путь проходит по задачам 1, 3, 4, 6, 7, 10, 11, 12, 13, 15, 18, 22, 23, 24, 25, 27, 28, 29, 30, 31.

Отчет о статистике проекта, сгенерированный средством Microsoft Project можно увидеть на рисунке 13.

	Начало	Окончание
Текущее	Чт 16.04.20	Вт 28.07.20
Базовое	НД	НД
Фактическое	НД	НД
Отклонение	0д	0д

	Длительность	Трудозатраты	Затраты
Текущие	74д	1 008ч	70 560,00 Р
Базовые	0д	0ч	0,00 Р
Фактические	0д	0ч	0,00 Р
Оставшиеся	74д	1 008ч	70 560,00 Р

Рисунок 13 – Статистика проекта

В статистике проекта можно увидеть подсчитанную Microsoft Project длительность реализации проекта – 74 дня, и предполагаемые затраты – 70 560 руб., что свидетельствует о экономической целесообразности данного проекта.

2 Проектирование средств поддержки онлайн-проверки статуса сертификата для операционной системы QPOS

2.1 Проектирование архитектуры

Согласно поставленным требованиям средства поддержки онлайн-проверки статуса сертификата для операционной системы QPOS представляет собой службу, не имеющую прямого пользовательского интерфейса. Взаимодействие с программными средствами происходит по нескольким каналам.

Клиенты службы взаимодействуют с ней по протоколу HTTP.

Служба записывает историю произошедших событий и ошибок в заданный текстовый файл.

Настройки службы хранятся в реестре и могут быть изменены там.

Взаимодействие службы с такими сущностями как «Хранилище сертификатов» и «Администратор» происходит по специальному протоколу удаленных вызовов «NetCore», доступ к которому предоставляет библиотека «NetCoreNative».

Было принято решение отказаться от собственной реализации HTTP сервера и использовать в качестве сервера HTTP nginx. От Nginx запросы будут получены по протоколу FastCGI, с помощью библиотеки «FastCGI».

На верхнем уровне абстракции служба представляет собой машину состояний, имеющую заданное число состояний и схему переходов между ними. Причиной выбора подобной архитектуры является то, что управление службой производится удаленно, причем управлять службой может несколько администраторов одновременно, а наличие состояний позволяет каждому администратору определить статус службы и доступные операции.

Для описания машины состояний лучше всего подходит диаграмма состояний [18]. Состояния и переходы службы описаны в диаграмме состояний, представленной на рисунке 14.

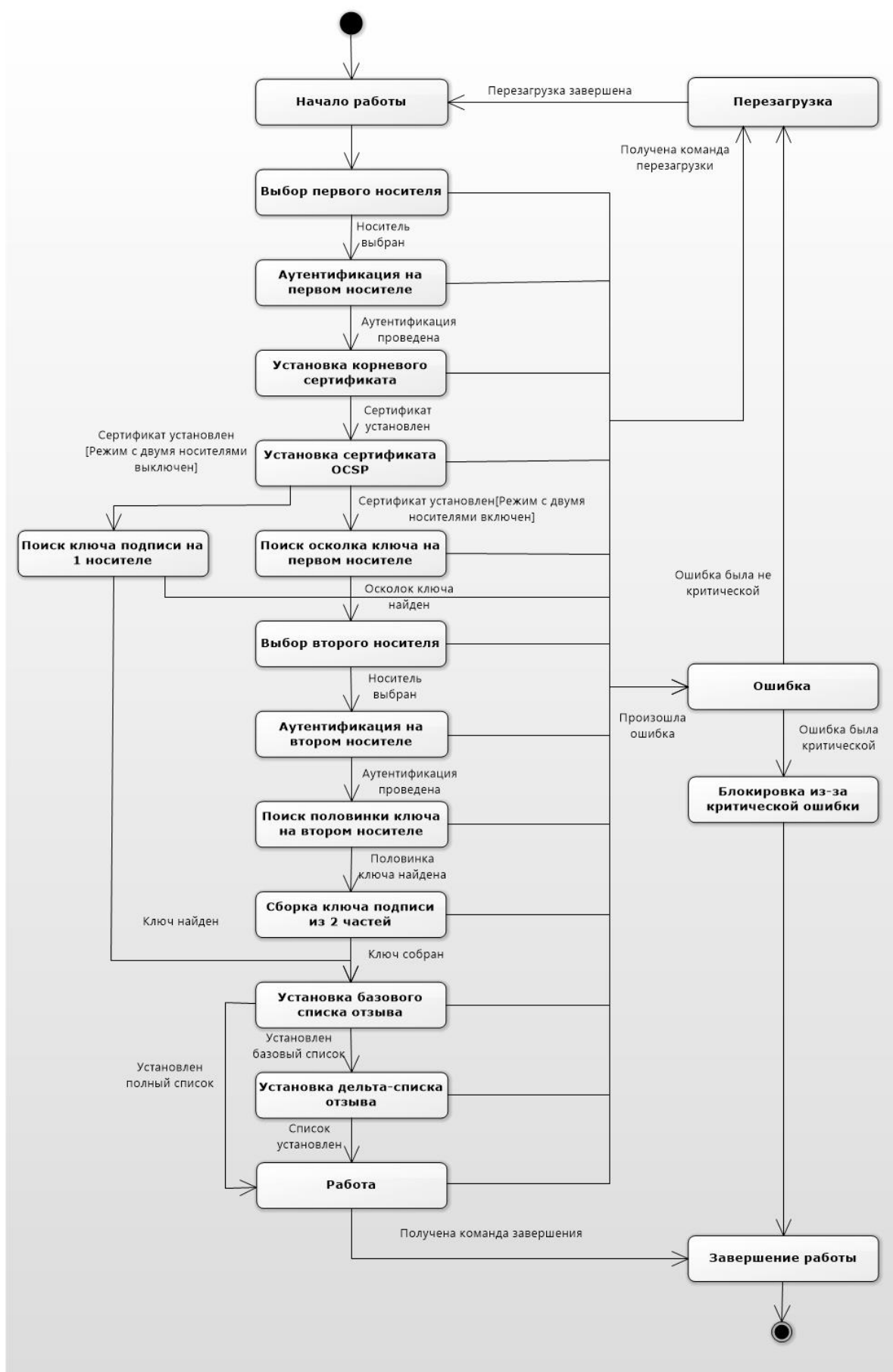


Рисунок 14 – Диаграмма состояний службы

Большинство переходов между состояниями инициируется сущностью «Администратор», внешним по отношению к данной системе программным средством, содержащим (вероятно), пользовательский интерфейс. Оставшиеся переходы представляют собой обработку ошибок.

Данная диаграмма, а также терминология, применённая при наименовании состояний, нуждается в более подробном описании.

После начала работы система переходит в состояние выбора первого носителя. Под словом «носитель» здесь понимается аппаратное (как правило) средство хранения ключей электронной подписи.

Для получения доступа к носителю ключей, необходимо аутентифицироваться в нем посредством передачи ему верного имени пользователя и пароля. В состоянии аутентификация службы ожидает получения этих данных.

После получения доступа к носителю служба ожидает установку двух сертификатов.

Дальнейший переход зависит от режима работы службы, определяемого настройками. Если режим работы с двумя носителями включен, то на первом носителе ожидается поиск осколка ключа подписи, после чего выполняется выбор второго носителя, аутентификация и поиск на нем половинки ключа. Затем выполняется сборка ключа подписи. Если режим работы с двумя носителями отключен, служба ожидает поиск цельного ключа подписи на первом носителе.

После установки САС служба начинает работу согласно протоколу OCSP.

2.2 Проектирование внешних интерфейсов

Протокол «NetCore» представляет собой протокол удаленных вызовов функций. Вызов функций производится по их названию.

Для взаимодействия с сущностью «Администратор» разрабатываемая система должна предоставлять для удаленного вызова набор функций, приведенных в таблице 4.

Таблица 4 – Функции для взаимодействия с сущностью «Администратор»

Название функции	Описание
RegisterClient	Подключение администратора к системе
UnregisterClient	Отключение администратора от системы
GetCurrentState	Получение текущего состояния системы
Reset	Перезапуск системы
GetSlotList	Получение списка доступных носителей
SelectSlot	Выбор одного носителя из доступных
LoginOnSlot	Аутентификация в выбранном носителе
FindKeyShard	Поиск осколка ключа подписи
FindKeyPart	Поиск половинки ключа подписи
FindSignKeyFromOneToken	Поиск ключа подписи службы
MakeOcspSignKeyFromParts	Сборка ключа подписи службы OCSP из двух частей
SetRootCertificate	Установка корневого сертификата с заданным серийным номером
SetOcspCertificate	Установка сертификата службы OCSP с заданным серийным номером
SetBaseCrl	Установка актуального основного списка отзыва
SetDeltaCrl	Установка актуального разностного списка отзыва
GetCurrentBaseCrl	Получение текущего установленного основного списка

Продолжение таблицы 4

Название функции	Описание
GetCurrentDeltaCrl	Получение текущего установленного разностного списка отозванных сертификатов
GetCurrentBaseCrlInstallTime	Получение времени установки основного САС
GetCurrentDeltaCrlInstallTime	Получение времени установки разностного САС
UpdateCrls	Принудительное обновления САС
Beacon	Проверка доступности службы

«Администратор» также предоставляет доступ к удаленному вызову набора функций, приведенных в таблице 5.

Таблица 5 – Функции, предоставляемые «Администратор»

Название функции	Описание
SetLastState	Оповещение о смене состояния службы
ShowError	Запрос на отображение текста ошибки
LogMessage	Оповещение о произошедшем событии
LogError	Оповещение о произошедшей ошибке
LogWarning	Оповещение о полученном предупреждении
NewCrlInstalled	Оповещение об установленном САС
Beacon	Проверка доступности администратора

Взаимодействие с хранилищем сертификатов происходит при помощи функции удаленного вызова «GetElement», предоставляемой хранилищем. С помощью данной функции можно получить как сертификат, так и САС. В параметрах данной функции задается тип получаемого элемента, а также дополнительные данные, идентифицирующие необходимый объект.

2.3 Проектирование структур данных

При анализе предметной области были выделены основные структуры данных предметной области. Данные структуры достаточно четко описаны в стандарте в терминах ASN.1 [19] и не нуждаются в более подробном описании.

В диаграмме классов на рисунке 15 приведем только наиболее значимые для проектирования классы.

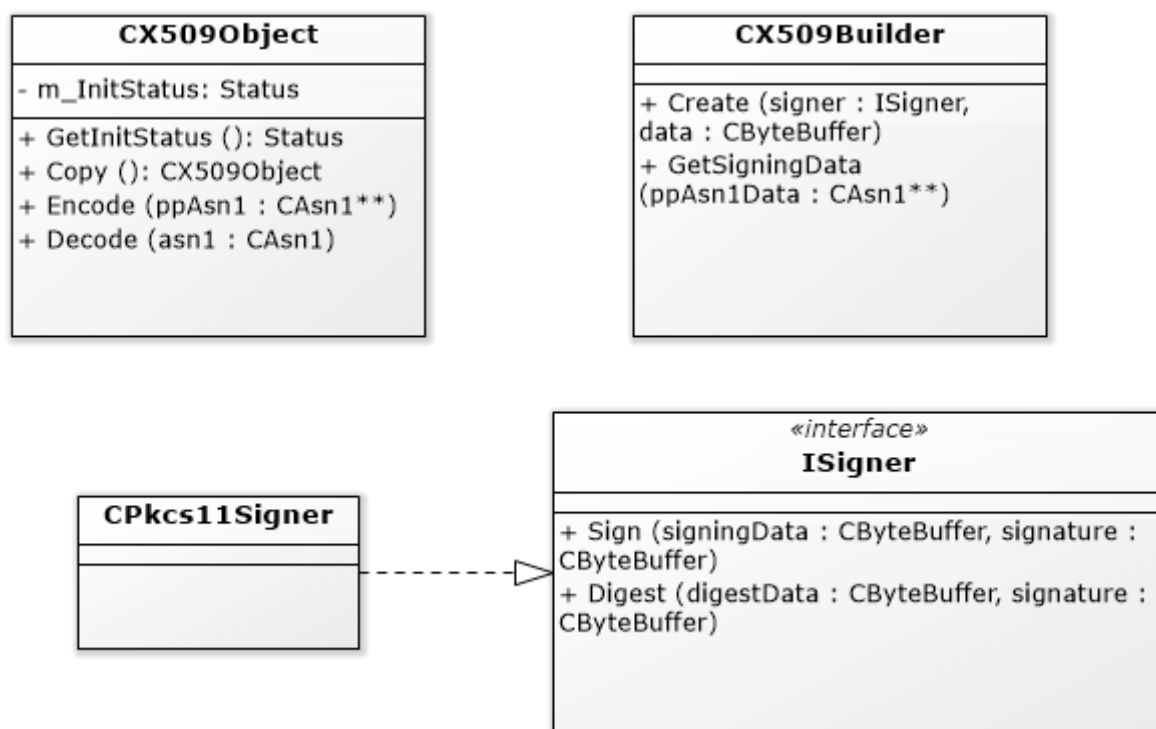


Рисунок 15 – Диаграмма классов, связанных с предметной областью

Классы соответствующих структурам данных, описанным в стандарте должны быть наследниками класса `CX509Object`, реализуя методы кодирования и декодирования в нотацию ASN.1.

Объекты, для формирования которых необходимо провести электронную подпись должны иметь отдельные классы создания, являющиеся наследниками класса CX509Builder. Интерфейс ISigner обеспечивает универсальный способ электронной подписи.

Помимо данных классов необходимо также выделить классы инкапсулирующие функциональность OSCP [20].

Упрощенная диаграмма классов, включающая в себя основные классы, участвующие в функционировании OSCP службы приведены на рисунке 16.

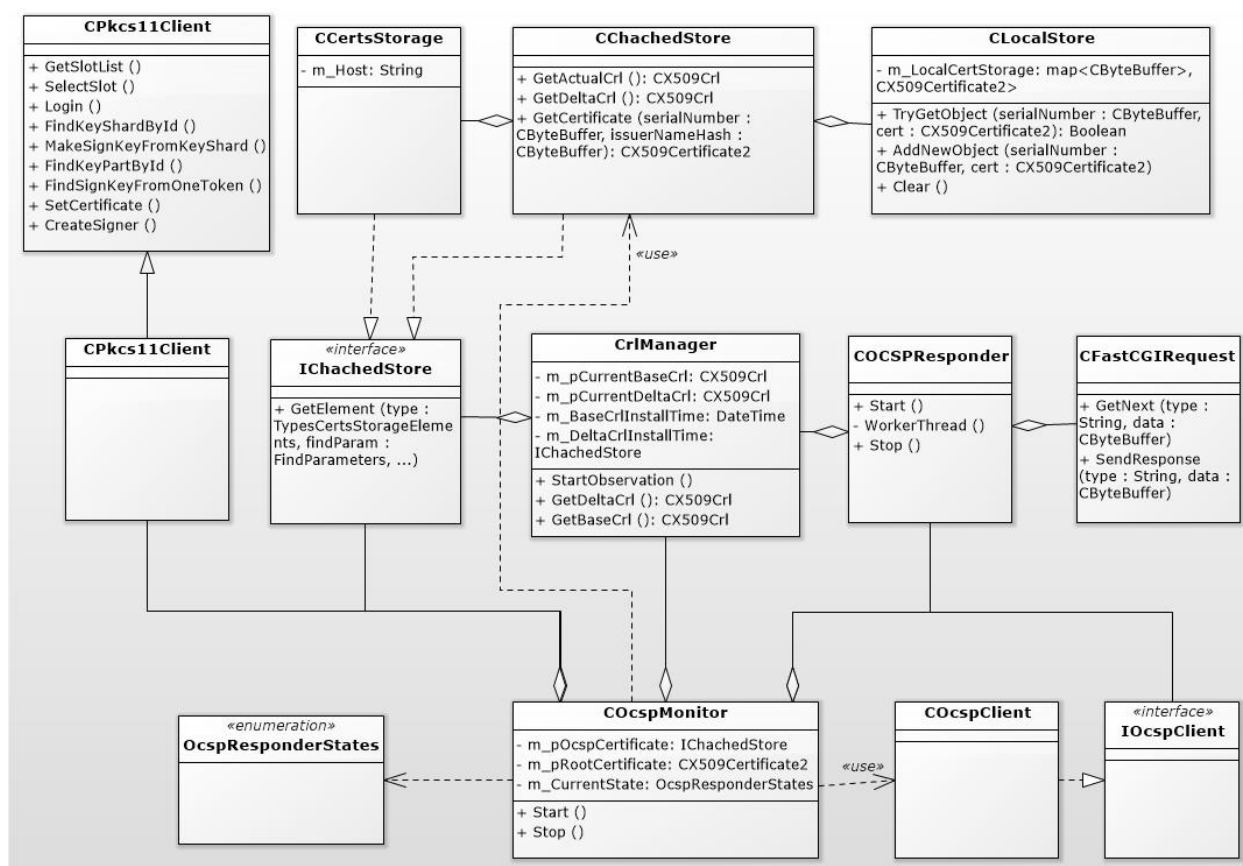


Рисунок 16 – Диаграмма классов OSCP службы

На диаграмме можно увидеть, что центральным является класс **COcspMonitor**. Именно он содержит в себе состояния «**OcspResponderStates**», описываемые при проектировании архитектуры. Помимо этого, данный класс содержит методы удаленного вызова, предоставляемые службой для администратора (они не указаны на диаграмме). Также данный класс содержит сертификаты, описанные при анализе предметной области и обозначенные здесь как «**CX509Certificate2**».

Интерфейс IOcspClient содержит набор функций, предоставляемых для удаленного вызова администратором. Класс COcspClient реализует удаленный вызов функций интерфейса.

Интерфейс IChachedStorage содержит набор функций, предоставляемых для удаленного вызова хранилищем сертификатов. Класс CCertsStorage реализует удаленный вызов функций интерфейса.

Классы CChachedStore, обеспечивает кеширование сертификатов, полученных от хранилища, в целях повышения быстродействия. Кеш хранится с помощью класса CLocalStore.

Класс CFastCGIRequest инкапсулирует в себе работу с библиотекой FastCGI и получение запросов о статусе сертификатов.

Класс CPkcs11Client инкапсулирует в себе работу со средствами подписи, предоставляя весь необходимый интерфейс.

Класс COcspPkcs11Client переопределяет ряд методов класса CPkcs11Client, устанавливая параметры подписи для OCSP.

Класс CCrlManager хранит в себе списки отозванных сертификатов, а также следит за их актуальностью, обновляя при необходимости.

Следует отметить класс COCSPResponder, так как именно он реализует основной функционал данной системы, а именно обработку запросов о статусе сертификатов. Данный класс запускает и останавливает пул потоков-обработчиков, которые исполняют метод WorkerThread с помощью методов Start и Stop.

2.4 Проектирование алгоритмов

Представим алгоритм инициализации службы, происходящей в рамках варианта использования «Запустить сервер», в виде диаграммы последовательности на рисунке 17.

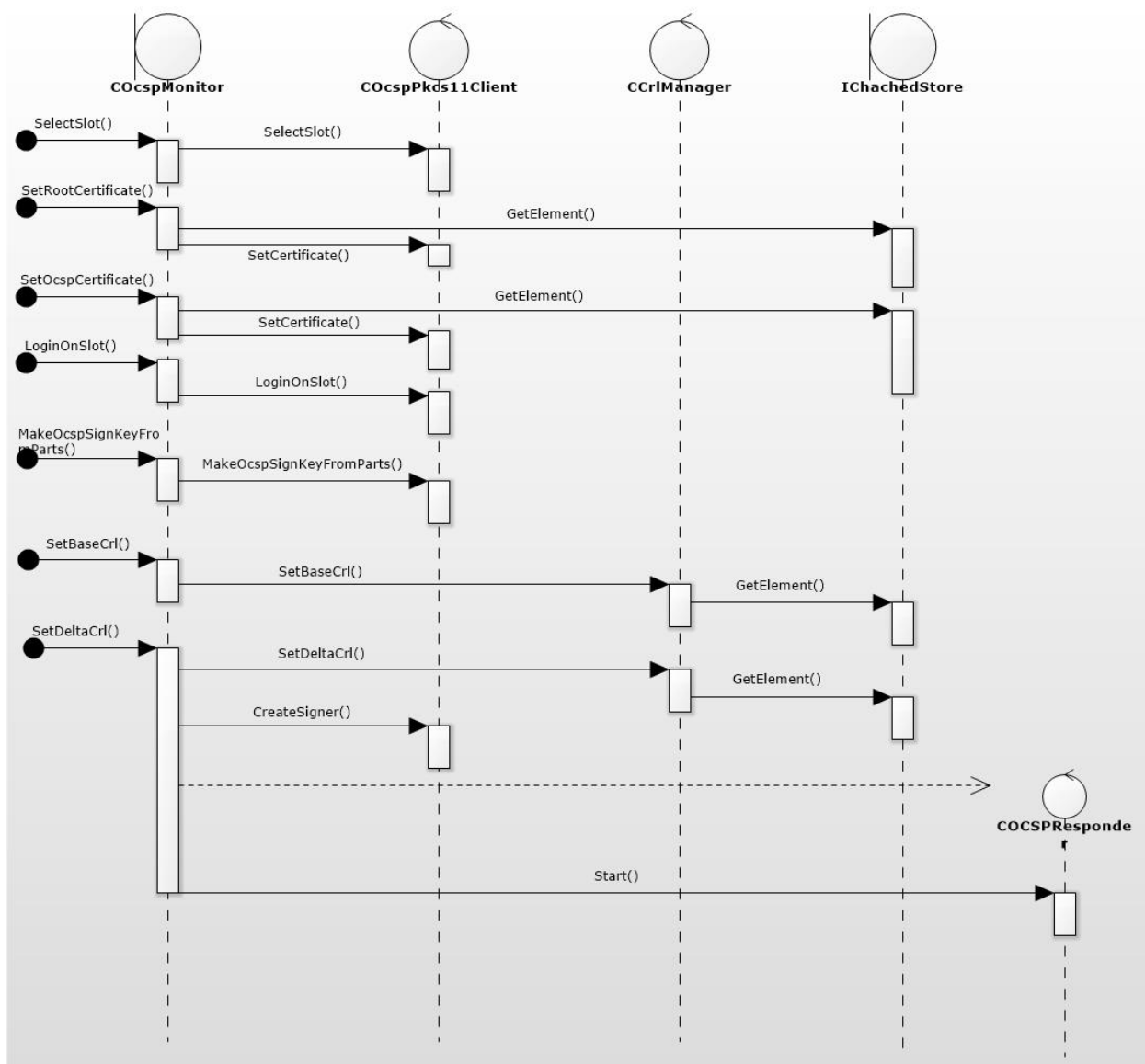


Рисунок 17 – Диаграмма последовательности инициализации службы

Инициализации производится при отключенном режиме работы с двумя носителями.

Из диаграммы видно, что в результате удаленных вызовов методов класса COCspMonitor производится настройка классов COCspPkcs11Client и CCrlManager, сопровождаемая получением сертификатов и САС от хранилища сертификатов через IChachedStore. После установки списков аннулированных сертификатов COCspMonitor переходит в режим работы, что сопровождается инициализацией класса COCSPResponde.

Рассмотрим алгоритм обработки запроса о статусе сертификата. Обработка запросов производится параллельно несколькими потоками, число которых устанавливается в настройках.

Потоки запускаются методом Start класса COCSPResponder, а останавливаются методом Stop, при этом код, исполняемый параллельно, находится в методе WorkerThread. Представим алгоритм метода WorkerThread в виде диаграммы деятельности на рисунке 18.

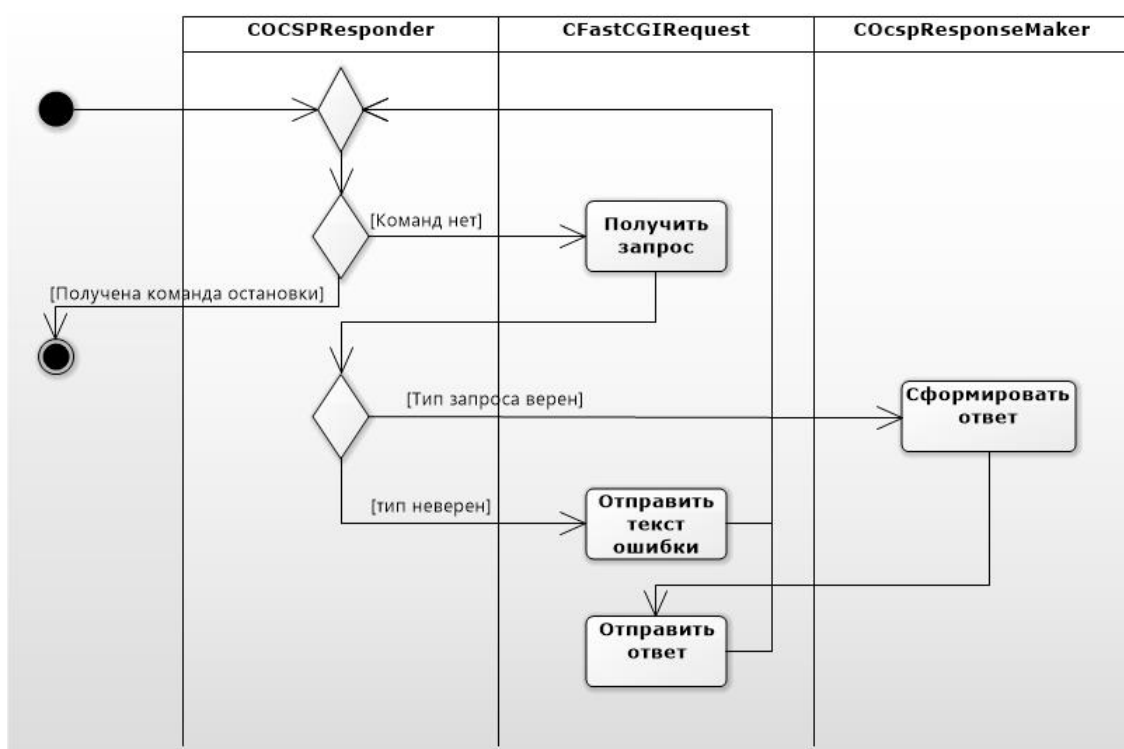


Рисунок 18 – Диаграмма деятельности метода WorkerThread

COcspResponseMaker представляет собой класс, в котором собраны функции, управляющие формированием OCSP ответа.

3 Реализация средств поддержки онлайн-проверки статуса сертификата для операционной системы QPOS

Результатом разработки является исполняемый файл приложения «OCSPService.qrx», предназначенный для работы в режиме, аналогичном службе Windows [21], под управлением операционной системы QPOS.

Разработка велась на языке программирования C++ в среде программирования Visual Studio 2019 Enterprise [22, 23]. Исходный код наиболее значимых классов службы приведен в приложении А.

Запуск и остановка службы осуществляется через меню управления службами.

Служба получает настройки из системной базы данных, по структуре аналогичной реестру Windows [24].

После запуска и настройки служба ожидает получения OCSP запросов на заданном в настройках порту по протоколу FastCGI [25, 26]. Для обработки HTTP запросов клиентов и их перенаправления в виде FastCGI запросов предполагается использование сервера Nginx [27]. Текст конфигурации сервера, применяемый для работы службы приведен в приложении Б.

Для работы службы необходимо наличие доступа к хранилищу сертификатов, адрес которого задается в настройках.

Служба разделена на несколько компонентов, собираемых независимо друг от друга. Часть компонентов, хотя и необходимы для функционирования службы, поставляются извне в виде библиотеки и заголовочных файлов на языке С, в связи с чем рассмотреть их внутренне строение не представляется возможным.

Компоненты и их зависимости друг от друга представлены на диаграмме компонентов на рисунке 19.

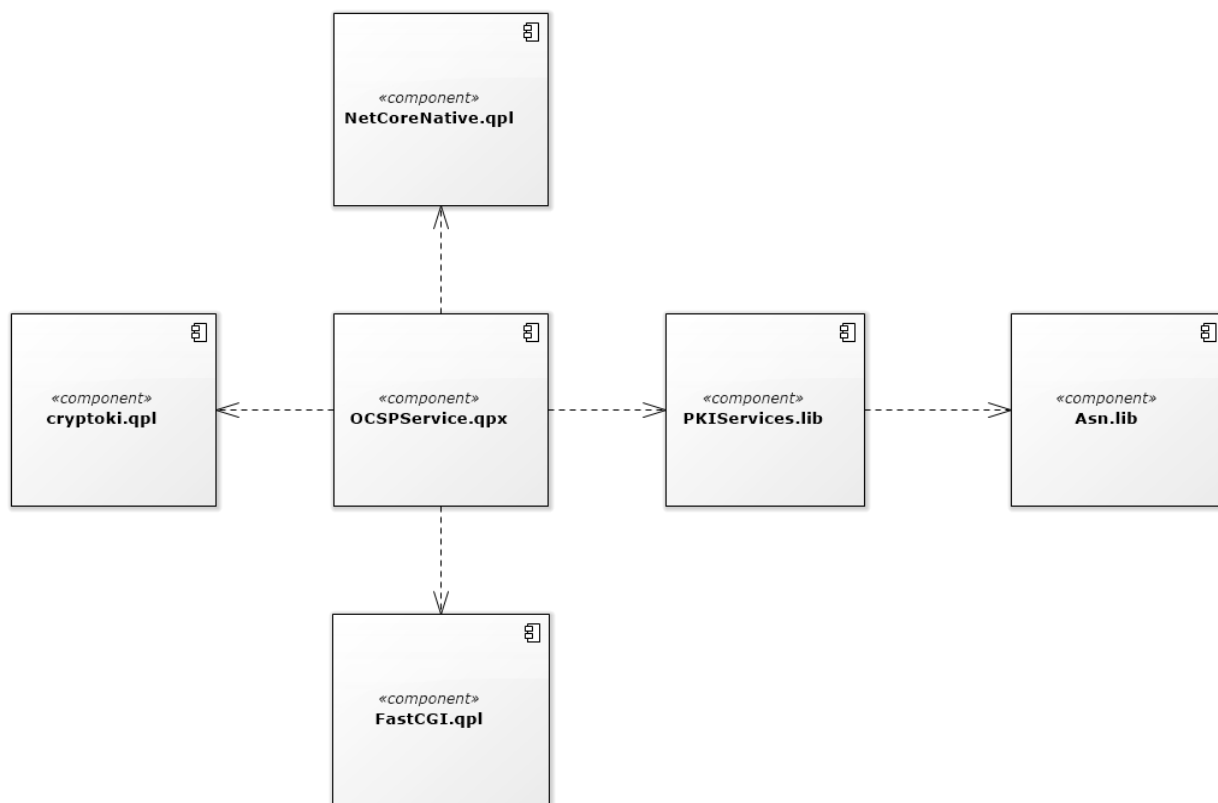


Рисунок 19 – Диаграмма компонентов службы

Компонент NetCoreNative.qpl является внешней библиотекой, реализующей протокол удаленных вызовов «NetCore».

Компонент FastCGI.qpl является внешней библиотекой, реализующей протокол «FastCGI».

Компонент cryptoki.qpl является внешней библиотекой, реализующей программный интерфейс доступа к криптографическим устройствам в соответствии со стандартом PKCS #11.

Оставшиеся компоненты разработаны в составе службы и отделены с целью возможности повторного использования.

Компонент Asn.lib включает в себя код работы с данными в нотации ASN.1. Состав данного компонента приведен на рисунке 20 в виде диаграммы классов, построенной средой Visual Studio.

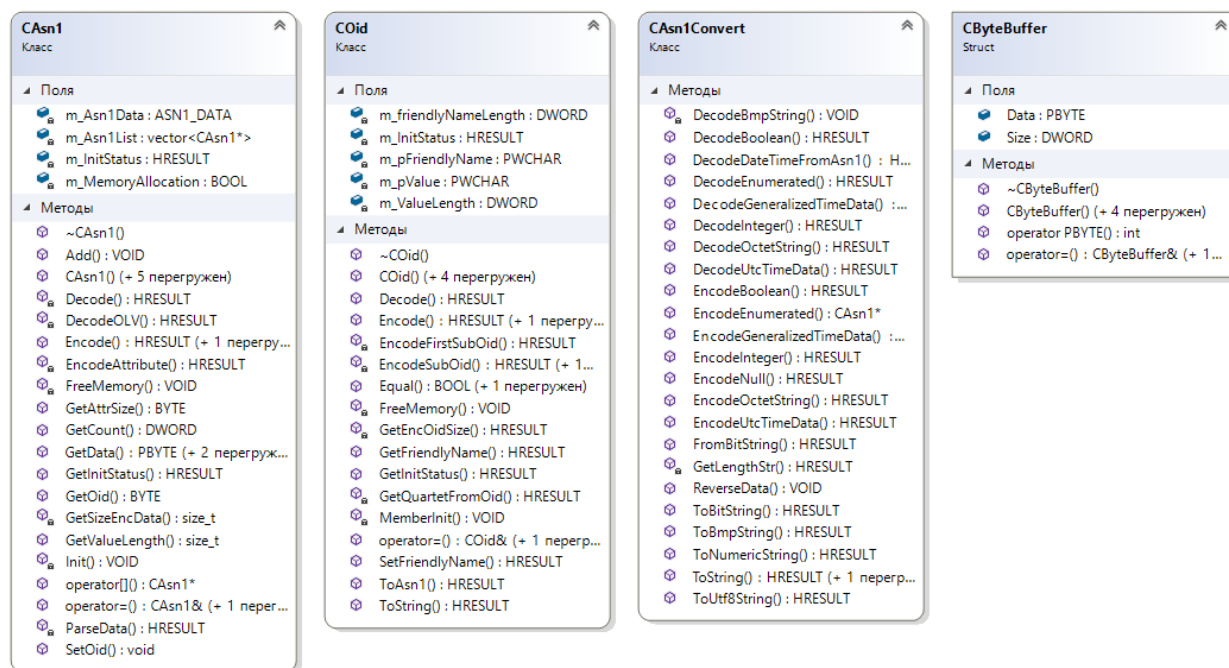


Рисунок 20 – Диаграмма классов компонента Asn.lib

Класс CAsn1 представляет собой элемент в нотации ASN.1, и предоставляет методы его кодирования и декодирования из последовательности байтов.

Класс COid является представлением объектного идентификатора – строки специального формата, используемого в нотации ASN.1.

Класс CAsnConvert предоставляет набор статических методов преобразования объектов CAsn1 в типы свойственные языку программирования и наоборот.

Класс CByteBuffer представляет собой реализацию буфера байтов заданной длины.

Компонент PKIServices.lib включает в себя ряд классов, соответствующих структурам данных, описанным в стандарте.

Диаграмма классов данного компонента, построенная средой Visual Studio, приведена на рисунке 21.

Рисунок 21 – Диаграмма классов компонента PKIServices.lib

Классы соответствующих структурам данных, описанным в стандарте являются наследниками класса CX509Object. Данные классы поддерживают кодирование и декодирование в нотацию ASN.1.

Объекты, для формирования которых необходимо провести электронную подпись имеют отдельные классы создания, являющиеся наследниками класса CX509Builder. Данные классы зависят от интерфейса ISigner.

Классы, реализующие интерфейс ISigner, производят электронную подпись объектов.

Классы, имеющие префикс «CPkcs11» в названии, производят взаимодействие с библиотекой Cryptoki, предоставляя функционал классам электронной подписи.

Не упомянутые здесь классы обеспечивают функциональность описанных классов.

Компонент OCSPService.qrx включает в себя классы, обеспечивающие функциональность сервера OCSP.

Диаграмма классов данного компонента, построенная средой Visual Studio, приведена на рисунке 22.

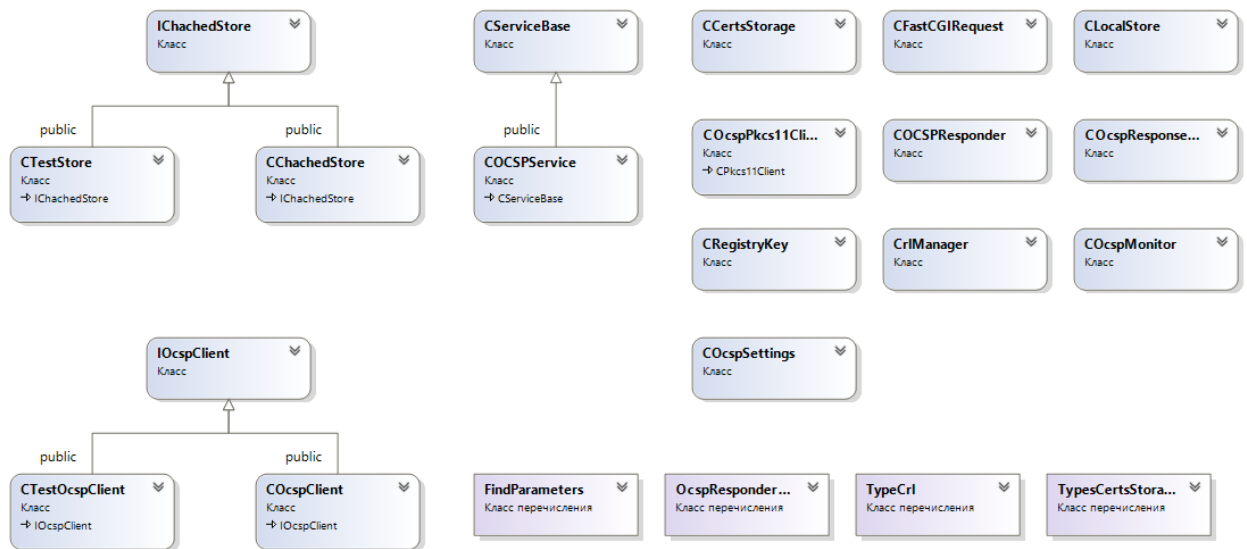


Рисунок 22 – Диаграмма классов компонента OCSPService.qrx

Большая часть классов данного компонента были описаны на этапе проектирования системы, в связи с чем не нуждаются в повторном описании.

Заметим, что интерфейсы ICachedStore и IOcspClient имеют по две реализации, одна из которых имеет префикс «CTest» и предназначена для тестирования.

Класс COcspSettings обеспечивает получение настроек из реестра, используя класс CRegistryKey.

Класс CServiceBase и его наследник обеспечивает функционирование данных средств как службы.

Стоит обратить внимание на перечисление OcspResponderStates, элементы которого обозначают состояния службы, описанные при проектировании архитектуры.

Особенности развертывания службы OCSP представлены с помощью диаграммы развертывания на рисунке 23.

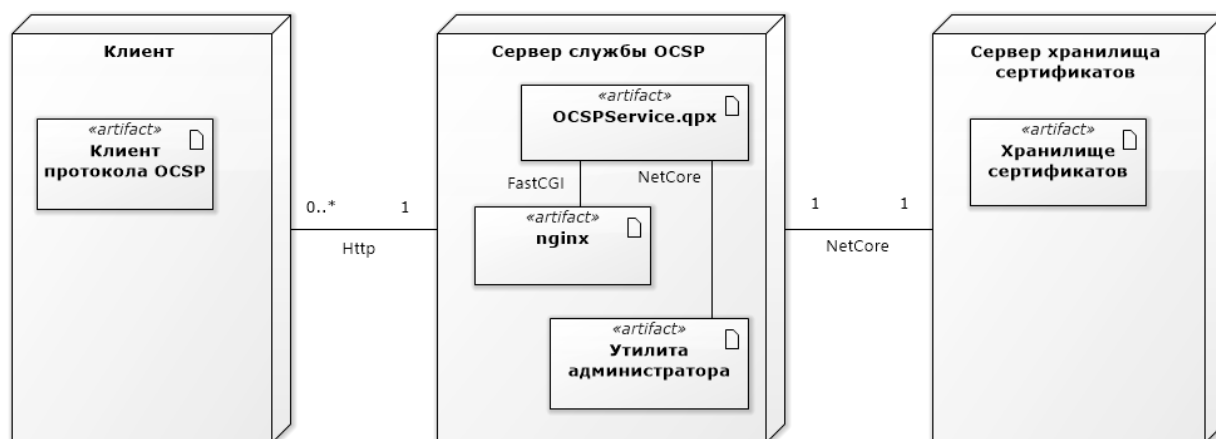


Рисунок 23 – Диаграмма развертывания OSCP службы

Следует отметить, что протокол NetCore не накладывает ограничений на местонахождение утилиты администратора, однако в целях безопасности, было принято решение ограничить доступ к управлению службой в пределах одной машины.

4 Тестирование программных средств

4.1 Функциональное тестирование

Функциональное тестирование относится к категории тестирования «черного ящика». Каждая функция программы тестируется вводом её исходных данных и анализа выходных. При этом внутренняя структура программы учитывается очень редко [28].

Функциональное тестирование предполагает составление плана тестирования, который строится на основе вариантов использования и включает в себя описание тестов и их результаты.

План функционального тестирования программных средств приведен в таблице 6, результаты приведены в приложении В.

Таблица 6 – План тестирования

	Вариант использования	Тест	Результат
1	Запустить сервер	Производится запуск и настройка сервера с корректными данными в корректном режиме	Тест пройден. Управляющая утилита получила отчет об успешном запуске службы (рисунок В.1). Фрагменты лога службы подтверждают это (листинг В.1).
2	Запустить сервер	Производится запуск и настройка службы, при этом введен неверный пароль.	Тест пройден. Управляющая утилита получила отчет о неудачной инициализации токена (Рисунки В.2 и В.3). Фрагмент лога службы подтверждает это (Листинг В.2).

Продолжение таблицы 6

	Вариант использования	Тест	Результат
3	Остановить сервер	Производится запуск и настройка сервера с корректными данными в корректном режиме, после чего состояние сервера сбрасывается	Тест пройден. Фрагмент лога службы подтверждает, успешный сброс состояния сервера (листинг В.3).
4	Проверить статус сертификата	Корректный запрос на действительный сертификат, присутствующий в хранилище	Тест пройден. Получен ответ о действительности сертификата. (листинги В.4 и В.5)
5	Проверить статус сертификата	Некорректный запрос	Тест пройден. Отправлен ответ, содержащий код ошибки некорректного запроса (листинг В.6).
6	Проверить статус сертификата	Корректный запрос на сертификат, отсутствующий в хранилище	Тест пройден. Статус сертификата неизвестен (листинг В.7).
7	Проверить статус сертификата	Корректный запрос на отозванный сертификат, присутствующий в хранилище	Тест пройден. Сертификат не действителен (листинг В.8).

На основе анализа результатов тестирования можно сделать вывод, что основные функции разработанного программного обеспечения онлайн-проверки статуса сертификатов работают корректно.

4.2 Модульное тестирование

Модульное тестирование – это процесс тестирования отдельных блоков, подпрограмм, классов и процедур, образующих крупную программу. Модульное тестирование в основном ориентировано на использование метода «белого ящика». Объясняется это прежде всего тем, что при последующем переходе к тестированию более крупных программных единиц, например, программ в целом, применимость метода «белого ящика» снижается [29].

Значительная часть кода в данной работе представляет собой реализацию ASN.1 кодирования и декодирования объектов, определенных стандартом. Данные процессы хорошо поддаются автоматизированному модульному тестированию. Для тестирования выберем классы CX509Certificate, CX509Crl, CX509OCSPRequest и CX509OCSPResponse, так как они включают в себя все остальные классы.

Проведем модульное тестирование с использованием платформы модульного тестирования Майкрософт для C++.

Непосредственная реализация тестов приведена в листинге 1.

Исходный код функций тестирования, используемых в модульном тестировании приведен в приложении Г.

Листинг 1 – Исходный код модульного теста

```
#include "pch.h"
#include "CppUnitTest.h"
#include "Windows.h"
#include "..\..\NativePKI\PKIServices\PKIServices.h"
#include "..\..\NativePKI\PKIServices\TestFunctions.h"
#define path "D:\\Files\\"
#define Count 1

using namespace Microsoft::VisualStudio::CppUnitTestFixture;

void Test(const char* pathToFile, const char* name, PVOID func)
{
    PBYTE pData = NULL;
    DWORD Size = 0;
    PBYTE pNewData = NULL;
    DWORD NewSize = 0;
```

```

PBYTE pResultData = NULL;
DWORD ResultDataSize = NULL;
string Str = pathToFile;
Str += name;
ReadDataFromFile(Str.c_str(), &pData, &Size);
for (int i1 = 0; i1 < Count; i1++)
{
    delete pResultData;
    delete pNewData;
    pResultData = NULL;
    ResultDataSize = NULL;
    pNewData = NULL;
    NewSize = 0;
    auto Status = ((HRESULT(*) (BYTE*, DWORD, PBYTE*,
DWORD*))func)(pData, Size, &pResultData, &ResultDataSize);
    Assert::IsTrue(Status == S_OK);
    Status = ((HRESULT(*) (BYTE*, DWORD, PBYTE*,
DWORD*))func)(pResultData, ResultDataSize, &pNewData,
&NewSize);
    Assert::IsTrue(Status == S_OK);
    Assert::AreEqual(NewSize, ResultDataSize);
    Assert::IsTrue((memcmp(pNewData, pResultData,
NewSize)) == 0);
}
delete pData;
delete pResultData;
delete pNewData;
}

namespace PKIServicesUnitTest
{
    TEST_CLASS(PKIServicesUnitTest)
    {
    public:

        TEST_METHOD(TestOcspRequest)
        {
            Test(path, "OcspRequest.orq", &OCSPRequestTest);
        }
        TEST_METHOD(TestOcspResponse)
        {
            Test(path, "OcspResponse.ors", &OCSPRequestTest);
        }
        TEST_METHOD(TestCertificate)
        {
            Test(path, "Certificate.cer", &CertificateTest);
        }
        TEST_METHOD(TestCrl)
        {
            Test(path, "Crl.crl", &CrlTest);
        }
    };
}

```

Скриншот, подтверждающий проведение модульного тестирования приведен на рисунке 24.

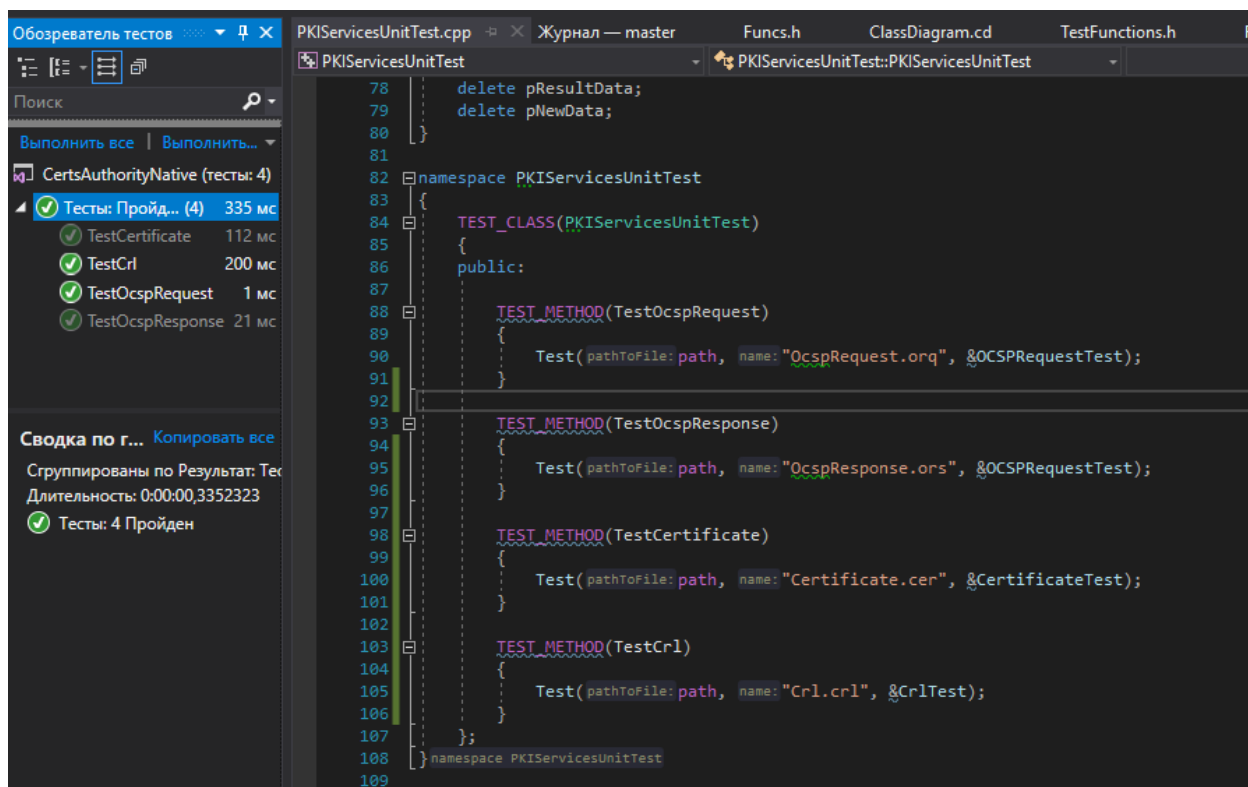


Рисунок 24 – Результат модульного тестирования

Все unit-тесты были успешно пройдены, что свидетельствует о корректности работы программы.

4.3 Нагрузочное тестирование

Главной функциональностью тестируемого программного обеспечения является ответы на запросы о статусе цифровых сертификатов. Согласно требованиям, программное средство должно эффективно обрабатывать большое число запросов. Для проверки соответствия данным требованиям было решено провести нагрузочное тестирование.

Нагрузочное тестирование — подвид тестирования производительности, сбор показателей и определение производительности и времени отклика программно-технической системы или устройства в ответ на внешний запрос с целью установления соответствия требованиям, предъявляемым к данной системе (устройству).

Для нагрузочного тестирования используется утилита «Tester», разработанная для проверки работы сервера онлайн-проверки статуса сертификатов.

Утилита применяется для проведения нагрузочного тестирования следующим способом: выбирается образец для запросов, http адрес сервера и количество соединений и потоков, отправляющих запросы.

В процессе работы утилита предоставляет следующие данные: среднее число запросов в секунду, общее число обработанных запросов, общее число необработанных запросов, а также подробные сведения о каждом запросе.

Успешным прохождением базового нагрузочного тестирования будем считать нулевое число необработанных запросов.

Представим применяемые сценарии нагрузочного тестирования в таблице 7.

Таблица 7 – Сценарии нагрузочного тестирования

Название	Описание	Ожидаемые параметры	Результат
Тестирование в однопоточном режиме	Число потоков равно одному, стандартные параметры утилиты тестирования (число потоков равно десяти, число соединений равно 10)	Общее число необработанных запросов равно нулю, при среднем числе запросов в секунду более 10	Число необработанных запросов: 0; Среднее число запросов в секунду: 208 (рисунок Д.1).
Тестирование с четырьмя рабочими потоками	Число потоков равно четырем, стандартные параметры утилиты тестирования (число потоков равно десяти, число соединений равно 10)	Общее число необработанных запросов равно нулю, при среднем числе запросов в секунду более 10	Число необработанных запросов: 0; Среднее число запросов в секунду: 362 (рисунок Д.2).
Тестирование с десятью рабочими потоками	Число потоков равно десяти, стандартные параметры утилиты тестирования (число потоков равно десяти, число соединений равно 10)	Общее число необработанных запросов равно нулю, при среднем числе запросов в секунду более 10	Число необработанных запросов: 0; Среднее число запросов в секунду: 356 (рисунок Д.3).

Следует отметить, что утилита тестирования и программа должны по возможности выполняться на различных компьютерах, чтобы убрать взаимное влияние.

Результаты нагрузочного тестирования показали, что приложение стабильно работает на заданных нагрузках, не имеет утечек памяти, а его производительность соответствует требованиям.

4.4 Метрики кода

4.4.1 Цикломатическая метрика Мак-Кейба.

При вычислении цикломатической сложности используется граф потока управления программы. Узлы графа соответствуют неделимым группам команд программы, они соединены ориентированными рёбрами, если группа команд, соответствующая второму узлу, может быть выполнена непосредственно после группы команд первого узла.

Проведем подсчет данной метрики для функции WorkerThread, обслуживающей приходящие запросы. Функция приведена в листинге 2.

Листинг 2 – Функция WorkerThread

```

/// <summary>
/// Рабочий поток. Обслуживает запросы (можно создать несколько)
/// </summary>
/// <param name="lpParam">Указатель на объект
COCSPPResponder.</param>
uintptr_t WINAPI COCSPPResponder::WorkerThread(LPVOID lpParam)
{
    LOG_PROC_ENTRY;
    HRESULT          Status      = E_FAIL;
    COCSPPResponder* pThis       = (COCSPPResponder*)lpParam;
    LPCSTR           TypeRequest = NULL;
    CSupportOfCryptX509* CryptX509 = NULL;
    CFastCGIRequest   Request;
    CByteBuffer       RequestData;

    __try
    {
        CryptX509 = pThis->m_pPkcs11->CreateSigner();

        CHECK_FOR_INTERNAL_ERROR("Не удалось создать рабочий
        криптографический объект", CryptX509 != NULL);
    }
}

```

```

COcspResponseMaker ResponseMaker(
    CryptX509,
    pThis->m_pOcspCertificate,
    pThis->m_pChachedStore,
    pThis->m_pCrlManager);

LOG_WRITE("Ожидание запроса");

while (!pThis->m_Stopping) {

    Status = Request.GetNext(TypeRequest,
RequestData);

    // Пришла команда остановки
    if (pThis->m_Stopping && (FAILED(Status)))
        break;

    CHECK_HR_STATUS_LEAVE("Не удалось получить
следующий запрос", Status);

    LOG_WRITE("Получен запрос");

    if (strcmp(TypeRequest, "application/ocsp-
request") != 0)
    {
        ERROR_WRITE("Не правильный CONTENT_TYPE");
        continue;
    }
    CByteBuffer ResponseData;
    Status = ResponseMaker.MakeResponse(RequestData,
RequestData.Size, ResponseData);
    LogStatus("При формировании ответа произошла
ошибка", Status);
// Ответ нужно отправить в любом случае
    Status = Request.SendResponse("application/ocsp-
response", ResponseData);
    CHECK_HR_STATUS_LEAVE("Не удалось отправить
ответ", Status);
    LOG_WRITE("Отправлен ответ");

    } /* while */

    Status = S_OK;
}
__finally
{
    LOG_WRITE("Рабочий поток завершен");
    if (FAILED(Status))
    {
        pThis->m_Stopping = TRUE;
    }
}
return Status;
}

```

Граф потока управления программы для рассматриваемой функции приведен на рисунке 25.

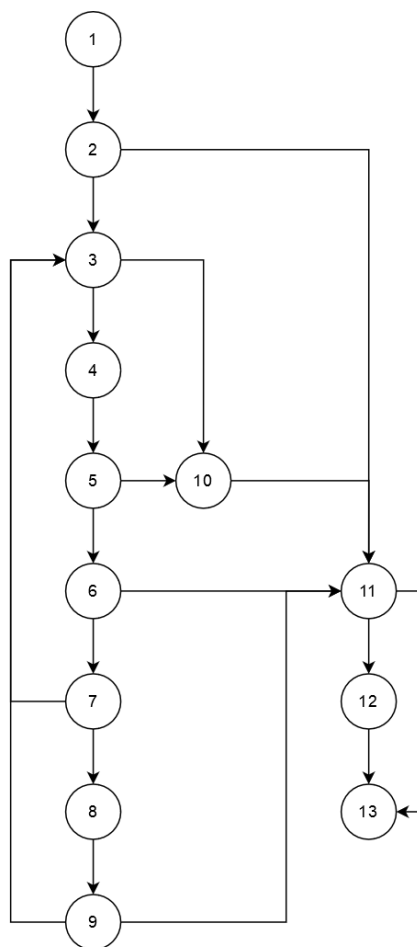


Рисунок 25 – Граф потока управления программы

Для вычисления цикломатической сложности $V(G)$ применяется следующая формула:

$$V(G) = e - n + 2p,$$

где e – количество дуг;

n – количество вершин;

p – число компонент связности.

Граф содержит 13 вершин, 19 дуг и один компонент связности.

Подставляем полученные значения и получаем:

$$V(G) = 19 - 13 + 2 = 8.$$

Таким образом значение цикломатической сложности равно 8. Это значение не превышает 10, следовательно, функция не является структурно сложной.

4.4.2 Метрика Холстеда

Метрика Холстеда относится к метрикам, вычисляемым на основании анализа числа строк и синтаксических элементов исходного кода программы.

Основу метрики Холстеда составляют четыре измеряемые характеристики программы [30]:

$n1$ – число уникальных операторов программы, включая символы-разделители, имена процедур и знаки операций (словарь операторов);

$n2$ – число уникальных операндов программы, включая константы и имена переменных (словарь операндов);

$N1$ – общее число операторов в программе;

$N2$ – общее число операндов в программе.

Подсчет операторов и операндов для функции WorkerThread приведен в таблицах 8 и 9 соответственно.

Таблица 8 – Операторы

Операторы	Число вхождений
=	9
!=	2
!	1
.	5
->	8
*	5
__try	1
__finally	1
<i>вызов функции ()</i>	11
<i>явное приведение ()</i>	1
If	3
while	1
break	1
continue	1
&&	1
LOG_PROC_ENTRY	1
CreateSigner	1
COcspResponseMaker	1
GetNext	1
CHECK_FOR_INTERNAL_ERROR	1

Продолжение таблицы 8

Операторы	Число вхождений
CHECK_HR_STATUS_LEAVE	2
strcmp	1
MakeResponse	1
SendResponse	1
;	27
,	12
{}	6
COCSPResponder::WorkerThread	1
n1 = 29	N1 = 109

Таблица 9 – Операнды

Операнды	Число вхождений
Status	14
pThis	8
TypeRequest	3
CryptX509	5
Request	12
RequestData	4
E_FAIL	1
lpParam	2
NULL	3
COCSPResponder*	2
m_pOcsCertificate	1
m_pChachedStore	1
m_pCrlManager	1
0	1
RequestData.Size	1
S_OK	1
TRUE	1
HRESULT	1
LPCSTR	1
CSupportOfCryptX509*	1
CFastCGIRequest	1
CByteBuffer	1
n2 = 22	N2 = 67

Сложность программы рассчитывается по формуле:

$$HD = (1 / 2) * (n2 / N2) = (29 / 2) * (67 / 22) = 44.16.$$

Длина программы рассчитывается по формуле:

$$N = N1 + N2 = 109 + 67 = 176.$$

Словарь программы рассчитывается по формуле:

$$n = n1 + n2 = 7 + 12 = 51.$$

Объем программы рассчитывается по формуле:

$$NV = N * \log_2(n) = 176 * 5.67 = 998.$$

Потенциальный объем программы рассчитывается по формуле:

$$V^* = (2 + n') * \log_2(2 + n') = 5 * \log_2(5) = 11.6.$$

Теорический уровень программы рассчитывается по формуле:

$$L' = V^* / NV = 11.6 / 998 = 0.0116.$$

Значение уровня программы согласно метрике Холстеда равняется 0.0116. Уровень программы имеет значение ниже среднего по данным измерений из книги Холстеда, однако остается в пределах допустимого [29].

4.4.3 Метрика Чепина

Суть метода состоит в оценке информационной прочности отдельно взятого программного модуля с помощью анализа характера использования переменных из списка ввода-вывода.

Все множество переменных, составляющих список ввода-вывода, разбивается на 4 функциональные группы:

Р – вводимые переменные для расчетов и для обеспечения вывода;

М – модифицируемые, или создаваемые внутри программы переменные;

С – переменные, участвующие в управлении работой программного модуля (управляющие переменные);

Т – не используемые в программе («паразитные») переменные.

Поскольку каждая переменная может выполнять одновременно несколько функций, необходимо учитывать ее в каждой соответствующей функциональной группе.

Метрика Чепина:

$$Q = aP + bM + cC + dT = 15.$$

где a, b, c, d – весовые коэффициенты.

Весовые коэффициенты использованы для отражения различного влияния на сложность программы каждой функциональной группы. По мнению автора метрики, наибольший вес, равный 3, имеет функциональная группа C , так как она влияет на поток управления программы. Весовые коэффициенты остальных групп распределяются следующим образом: $a = 1$, $b = 2$, $d = 0.5$. Весовой коэффициент группы T не равен 0, поскольку «паразитные» переменные не увеличивают сложность потока данных программы, но иногда затрудняют ее понимание. С учетом весовых коэффициентов.

В представленной функции характеристики равны:

$$P = 1,$$

$$M = 6,$$

$$C = 3,$$

$$T = 0.$$

Подставим полученные значения в формулу:

$$Q = 1 + 2 \times 6 + 3 \times 3 + 0.5 \times 0 = 22.$$

Значение метрики Чепина равняется 22. Если в результате изменения рассматриваемой функции значение метрики значительно увеличится, это будет означать низкое качество изменений. Сравнение значения метрики Чепина для различных программ обычно неэффективно, обычно она используется для анализа изменений одного и того же участка кода.

Заключение

В рамках выполнения выпускной квалификационной работы были разработаны программные средства поддержки онлайн-проверки статуса сертификата для операционной системы QPOS.

Был проведен анализ предметной области и произведена постановка задачи, в ходе чего было проанализировано программное обеспечение аналогичного назначения, рассмотрены их особенности и выделены недостатки, анализ существующих разработок показал необходимость разработки собственного аналога. Также были разработаны функциональные и нефункциональные требования.

Было проведено планирование разработки и оценка бюджета. Длительность реализации проекта составляет 74 дня, а предполагаемые затраты – 70 560 руб.

На этапе проектирования данных программных средств применялся язык моделирования UML с помощью среды разработки диаграмм «Software Ideas Modeler».

Для реализации программных средств использовался язык программирования C++, код на котором писался в среде программирования Visual Studio 2019.

Было проведено функциональное, модульное и нагрузочное тестирование разработанного приложения. На основе результатов тестирования можно сделать вывод, что разработанное приложение соответствует поставленным требованиям и работает корректно.

В приложении Е представлен акт о внедрении данного продукта в состав программного комплекса, разрабатываемого ООО НТП «Криптософт».

Таким образом, в результате выполнения выпускной квалификационной работы были разработаны программные средства, отвечающее всем требованиям технического задания.

Список использованных источников

1. Программа «Цифровая экономика Российской Федерации». Утверждена распоряжением Правительства Российской Федерации от 28 июля 2017 г. № 1632-р. [Электронный ресурс]. – URL: <http://static.government.ru/media/files/9gFM4FHj4PsB79I5v7yLVuPgu4bvR7M0.pdf> (дата обращения: 23.04.2020).
2. Распоряжение Президента Российской Федерации от 9 мая 2017 г. № 203 «Стратегия развития информационного общества в Российской Федерации на 2017 – 2030 годы». [Электронный ресурс]. – URL: http://zmedu.hostedu.ru/files/ykaz_7668.pdf (дата обращения: 23.04.2020).
3. Мао, В. Современная криптография: теория и практика / В. Мао.: Пер. с англ. – М.: Издательский дом Вильямс, 2005. – 768 с.
4. Царев, Е. Исследование «Рынок информационной безопасности Российской федерации + обзор рынка информационной безопасности Украины» [Электронный ресурс]. URL: https://tsarev.biz/wp-content/uploads/Pervoe%20expertnoe%20issledovanie%20rinka%20IB%20Rossii_2013.pdf (дата обращения: 12.05.2020).
5. Горбатов, В.С. Основы технологии PKI / В.С. Горбатов, О.Ю. Полянская. – М.: Горячая линия – Телеком, 2004. – 248 с.
6. Блинов, А.М. Информационная безопасность: Учебное пособие. Часть 1 / А.М. Блинов. – СПб.: Изд-во СПбГУЭФ, 2010. – 96 с.
7. Клерк, Ж. Проверка подлинности цифровых сертификатов в инфраструктуре Windows PKI / Ж. Клерк // Windows IT Pro/RE. 2007. № 02 [Электронный ресурс]. URL: <https://www.osp.ru/winitpro/2007/02/4102619/> (дата обращения: 23.04.2020)
8. Родичев, Ю.А. Информационная безопасность. Национальные стандарты Российской Федерации / Ю.А. Родичев. – Москва [и др.]: Питер, 2008. – 271 с.

9. Маккинли, Б. Основы PKI / Б. Маккинли // Computerworld Россия. 2000. № 38 [Электронный ресурс]. URL: <https://www.osp.ru/cw/2000/38/7235/> (дата обращения: 25.04.2020)
10. Мерзликин, Н.Ю. Особенности реализации протоколов ocsp и tsp / Н.Ю. Мерзликин, В.Ю. Платонов, В.С. Лукьянов, Д.В. Быков // Современные проблемы науки и образования. Выпуск журнала № 3 за 2013 год [Электронный ресурс]. URL: <https://science-education.ru/ru/article/view?id=9175> (дата обращения: 25.04.2020)
11. Стардеван К. PKI заступает на дежурство / К. Стардеван // PCWeek/RE, Netweek. 2001. No 6 (276) [Электронный ресурс]. URL: <http://www.pcweek.ru/Year2001/N6/CP1251/NetWeek/chaptl.htm> (дата обращения: 23.04.2020)
12. X.509 Internet Public Key Infrastructure Certificate Status Protocol – OCSP // RFC [Электронный ресурс]. – URL: <http://www.ietf.org/rfc/rfc2560.txt> (дата обращения: 23.04.2020)
13. КриптоПро OCSP // КриптоПро [Электронный ресурс]. URL: <https://www.cryptopro.ru/products/pki/ocsp> (дата обращения: 23.04.2020)
14. EJBCA product documentation // EJBCA [Электронный ресурс]. URL: <https://www.ejbca.org/documentation/> (дата обращения: 23.04.2020)
15. ГОСТ Р 34.10-2012. Национальный стандарт российской федерации. Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи.
16. ITU-T Recommendation X.509. Information Technology - Open Systems Interconnection - The Directory Public Key and Attribute Certificate Frameworks. June 2000 [Электронный ресурс]. URL: <https://www.itu.int/ITU-T/recommendations/rec.aspx?rec=X.509> (дата обращения: 23.04.2020)
17. OASIS Enhances Popular Public-Key Cryptography Standard, PKCS #11, for Mobile and Cloud // OASIS [Электронный ресурс]. URL: www.oasis-open.org (дата обращения 23.04.2020).

18. Фаулер, М. UML. Основы, 33е издание / М. Фаулер. – Пер. с англ. – СПб: СимволлПлюс, 2004. – 192 с., ил.
19. Строжевский Ю. ASN.1 простыми словами // RSDN [Электронный ресурс]. – URL: <http://rsdn.org/article/ASN/ASN.xml> (дата обращения: 23.04.2020)
20. Арлоу, Д. UML 2 и Унифицированный процесс / Д. Арлоу, А. Нейштадт. – 2-е издание, перевод с англ. Шатохиной Н. – Санкт-Петербург: «Символ-Плюс», 2007. – 624 с.
21. Служба Windows // Википедия [Электронный ресурс]. – URL: https://ru.wikipedia.org/wiki/Служба_Windows (дата обращения: 23.04.2020)
22. Иванова, Г.С. Объектно-ориентированное программирование / Г.С. Иванова, Е.К. Пугачев. – М.: Издательство МГТУ имени Н.Э. Баумана, 2001. – 319 с.
23. Павловская, Т.А. С/С++. Программирование на языке высокого уровня / Т.А. Павловская. – СПб: Питер, 2007.
24. Реестр Windows // Википедия [Электронный ресурс]. – URL: https://ru.wikipedia.org/wiki/Реестр_Windows (дата обращения: 23.04.2020)
25. FastCGI Specification // FastCGI-Archives [Электронный ресурс]. – URL: https://fastcgi-archives.github.io/FastCGI_Specification.html (дата обращения: 23.04.2020)
26. FastCGI // Википедия [Электронный ресурс]. – URL: <https://ru.wikipedia.org/wiki/FastCGI> (дата обращения: 23.04.2020)
27. nginx: документация // nginx [Электронный ресурс]. – URL: <https://nginx.org/ru/docs/> (дата обращения: 23.04.2020)
28. Котляров, В.П. Основы тестирования программного обеспечения / В.П. Котляров, Т.В. Коликова. – М.: Бином. Лаборатория знаний, 2006. – 285 с.
29. Канер, С. Тестирование программного обеспечения / С. Канер, Д. Фолк, Е.К. Нгуен. – К.: ДияСофт, 2001. – 544 с.

30. Холстед, М.Х. Начала науки о программах / М.Х. Холстед; пер. с англ. В. М. Юфы. - М. : Финансы и статистика, 1981. - 128 с., ил.

Приложение А
Исходный код
(рекомендуемое)

Листинг A.1 – OcspMonitor.h

```
#pragma once
class COcspMonitor
{
public:
    /// <summary>
    /// Запуск сервера NetCore, для связи с управляющей консолью
    /// </summary>
    /// <returns>Результат выполнения операции.</returns>
    HRESULT static Start();
    /// <summary>
    /// Запуск сервера NetCore, для связи с управляющей консолью
    /// </summary>
    /// <param name="Host">Хост сервера.</param>
    /// <param name="Port">Порт сервера.</param>
    /// <returns>Результат выполнения операции.</returns>
    HRESULT static Start(const char* Host,
                          short Port);

    /// <summary>
    /// Остановка сервера связи с управляющей консолью
    /// </summary>
    /// <returns>Результат выполнения операции. </returns>
    HRESULT static Stop();
    /// Интерфейс сервера WCF, предоставляемого службой OSCP, для
    взаимодействия и управления ею.
#pragma region IOcspServer
    /// <summary>
    /// Регистрация нового клиента.
    /// </summary>
    /// <param name="retVoid">Ничего, либо исключение.</param>
    /// <param name="clientIdInt">Уникальный идентификатор клиента.</param>
    /// <returns>Результат выполнения операции.</returns>
    HRESULT static RegisterClient(PNC_OBJECT, PVOID clientIdInt);
    /// <summary>
    /// Удаление клиента.
    /// </summary>
    /// <param name="retVoid">Ничего, либо исключение.</param>
    /// <param name="clientIdInt">Уникальный идентификатор клиента.</param>
    /// <returns>Результат выполнения операции.</returns>
    HRESULT static UnregisterClient(PNC_OBJECT, PVOID clientIdInt);
    /// <summary>
    /// Получение состояния конечного автомата у службы.
    /// </summary>
    /// <param name="retUInt">Номер текущего состояния.</param>
    /// <returns>Результат выполнения операции.</returns>
    HRESULT static GetCurrentState(PNC_OBJECT retUInt);
    /// <summary>
    /// Перезапуск процесса.
    /// </summary>
    /// <param name="retVoid">Ничего, либо исключение.</param>
    /// <returns>Результат выполнения операции.</returns>
    HRESULT static Reset(PNC_OBJECT);
    /// <summary>
    /// Получение списка слотов.
    /// </summary>
    /// <param name="retBytes">Список слотов, которые может использовать
    служба.</param>
    /// <returns>Результат выполнения операции.</returns>
    HRESULT static GetSlotList(PNC_OBJECT retBytes);
    /// <summary>
    /// Выбор текущего слота.
    /// </summary>
    /// <param name="retVoid">Ничего, либо исключение.</param>
```

```

/// <param name="slotUInt">Описание слота.</param>
/// <returns>Результат выполнения операции.</returns>
HRESULT static SelectSlot(PNC_OBJECT, PVOID slotUInt);
/// <summary>
/// Выполнить аутентификацию на выбранном слоте.
/// </summary>
/// <param name="retVoid">Ничего, либо исключение.</param>
/// <param name="passwordString">Пароль.</param>
/// <returns>Результат выполнения операции.</returns>
HRESULT static LoginOnSlot(PNC_OBJECT, PVOID passwordString);
/// <summary>
/// Поиск осколка ключа администратора.
/// </summary>
/// <param name="retVoid">Ничего, либо исключение.</param>
/// <returns>Результат выполнения операции.</returns>
HRESULT static FindKeyShard(PNC_OBJECT);
/// <summary>
/// Поиск половинки ключа администратора сертификации.
/// </summary>
/// <param name="retVoid">Ничего, либо исключение.</param>
/// <returns>Результат выполнения операции.</returns>
HRESULT static FindKeyPart(PNC_OBJECT);
/// <summary>
/// Поиск ключа подписи службы.
/// </summary>
/// <param name="retVoid">Ничего, либо исключение.</param>
/// <returns>Результат выполнения операции.</returns>
HRESULT static FindSignKeyFromOneToken(PNC_OBJECT);
/// <summary>
/// Выполнить сборку ключа подписи службы OSCP из 2-х частей.
/// </summary>
/// <param name="retVoid">Ничего, либо исключение.</param>
/// <returns>Результат выполнения операции.</returns>
HRESULT static MakeOcspsignKeyFromParts(PNC_OBJECT);
/// <summary>
/// Установка корневого сертификата в ДВУ с заданным серийным номером.
/// </summary>
/// <param name="retVoid">Ничего, либо исключение.</param>
/// <param name="serialNumber">Серийный номер устанавливаемого
сертификата.</param>
/// <returns>Результат выполнения операции.</returns>
HRESULT static SetRootCertificate(PNC_OBJECT, PVOID serialNumber);
/// <summary>
/// Установка сертификата службы OSCP в ДВУ с заданным серийным номером.
/// </summary>
/// <param name="retVoid">Ничего, либо исключение.</param>
/// <param name="serialNumber">Серийный номер устанавливаемого
сертификата.</param>
/// <returns>Результат выполнения операции.</returns>
HRESULT static SetOcspsCertificate(PNC_OBJECT, PVOID serialNumber);
/// <summary>
/// Установка актуального основного списка отзыва.
/// </summary>
/// <param name="retVoid">Ничего, либо исключение.</param>
/// <returns>Результат выполнения операции.</returns>
HRESULT static SetBaseCrl(PNC_OBJECT);
/// <summary>
/// Установка актуального разностного списка отзыва.
/// </summary>
/// <param name="retVoid">Ничего, либо исключение.</param>
/// <returns>Результат выполнения операции.</returns>
HRESULT static SetDeltaCrl(PNC_OBJECT);
/// <summary>

```

```

    /// Получение текущего установленного основного списка отозванных
    сертификатов.
    /// </summary>
    /// <param name="idInt">Закодированный в Asn.1 список отозванных
    сертификатов.</param>
    /// <returns>Результат выполнения операции.</returns>
    HRESULT static GetCurrentBaseCrl(PNC_OBJECT retBytes);
    /// <summary>
    /// Получение текущего установленного разностного списка отозванных
    сертификатов.
    /// </summary>
    /// <param name="idInt">Закодированный в Asn.1 список отозванных
    сертификатов.</param>
    /// <returns>Результат выполнения операции.</returns>
    HRESULT static GetCurrentDeltaCrl(PNC_OBJECT retBytes);
    /// <summary>
    /// Получение времени установки основного CAC.
    /// </summary>
    /// <param name="retDateTime">Время установки основного CAC.</param>
    /// <returns>Результат выполнения операции.</returns>
    HRESULT static GetCurrentBaseCrlInstallTime(PNC_OBJECT retDateTime);
    /// <summary>
    /// Получение времени установки разностного CAC.
    /// </summary>
    /// <param name="retDateTime">Время установки основного CAC.</param>
    /// <returns>Результат выполнения операции.</returns>
    HRESULT static GetCurrentDeltaCrlInstallTime(PNC_OBJECT retDateTime);
    /// <summary>
    /// Выполнить попытку принудительного обновления CAC.
    /// </summary>
    /// <param name="retVoid">Ничего, либо исключение.</param>
    /// <returns>Результат выполнения операции.</returns>
    HRESULT static UpdateCrls(PNC_OBJECT);
    /// <summary>
    /// Маячок для клиента.
    /// </summary>
    /// <param name="retBool">Зарегистрирован ли клиент.</param>
    /// <param name="idInt">Идентификатор клиента.</param>
    /// <returns>Результат выполнения операции.</returns>
    HRESULT static Beacon(PNC_OBJECT retBool, PVOID idInt);
#pragma endregion
private:
    COcspMonitor() = delete;
    /// <summary>
    /// Переход в рабочий режим
    /// </summary>
    /// <returns>Результат выполнения операции.</returns>
    HRESULT static StartWorking();
    /// <summary>
    /// Выход из рабочего режима
    /// </summary>
    /// <returns>Результат выполнения операции.</returns>
    VOID static StopWorking();
    /// <summary>
    /// Изменение состояния службы
    /// </summary>
    /// <param name="newState">Новое состояние.</param>
    void static ChangeState(OcspResponderStates newState);
    /// <summary>
    /// Обработчик состояния ошибки
    /// </summary>
    /// <param name="message">Сообщение об ошибке.</param>
    /// <param name="error">Код ошибки.</param>
    void static ErrorState(wstring message,

```

```

                                CK_RV    error = CKR_OK);
HRESULT static RegisterFuncs();
HRESULT static UnregisterFuncs();
/*/// <summary>
/// Инициализация OCSP монитора
/// </summary>
/// <param name="pCrlManager">Менеджер списка отзывает.</param>
/// <param name="pPkcs11">Криптография.</param>
/// <param name="pChachedStore">Хранилище сертификатов.</param>
/// <param name="Host">Хост сервера.</param>
/// <param name="Port">Порт сервера.</param>
COcspMonitor(CrlManager*          pCrlManager,
              CPkcs11*             pPkcs11,
              IChachedStore*       pChachedStore,
              const char*          Host,
              short                 Port);*/

static BOOL          m_IsRunning;
static CrlManager*   m_pCrlManager;
static COCSPResponder* m_pOCSPResponder;
static OcspResponderStates m_CurrentState;
static OcspResponderStates m_PrevState;
static IOcspClient*   m_pOCSPClient;
static IChachedStore* m_pChachedStore;
static CX509Certificate2* m_pOcspCertificate;
static DWORD          m_ClientId;
static CPkcs11Client* m_pPkcs11;
static const char*    m_Host;
static short          m_Port;
};

```

Листинг A.2 – OcspMonitor.cpp

```

#include "pch.h"
#ifdef WIN32 && defined(DEBUG)
    #include "TestStore.h"
#else
    #include "ChachedStore.h"
#endif
// 1970 год в DateTime.
#define NET_CORE_START_TIME_FOR_TIME_T 621355968000000000ULL
// Коэффициент для преобразования time_t (1 сек) в DateTime (100 нс)
#define NET_CORE_COEF_TIME 10000000
// 1600 год в DateTime.
#define NET_CORE_START_TIME_FOR_FILETIME 504595008000000000ULL
// Преобразования time_t в DateTime
#define TIME_T_TO_DATETIME(FILETIME) ((FILETIME) * NET_CORE_COEF_TIME +
NET_CORE_START_TIME_FOR_TIME_T)
// Преобразования DateTime в time_t
// ! DateTime начинается с 01.01.0001 00:00, а time_t с 1970. Если в DateTime
будет год раньше 1970 - будет мусор
#define DATETIME_TO_TIME_T(DateTime) (((DateTime) -
NET_CORE_START_TIME_FOR_TIME_T) / NET_CORE_COEF_TIME)
// Преобразования FILETIME в DateTime
#define FILETIME_TO_DATETIME(FILETIME) (FILETIME_TO_INT64(FILETIME) +
NET_CORE_START_TIME_FOR_FILETIME)
// Преобразования DateTime в FILETIME
// ! DateTime начинается с 01.01.0001 00:00, а FILETIME с 1600. Если в
DateTime будет год раньше 1600 - будет мусор
#define DATETIME_TO_FILETIME(DateTime) ((DateTime) -
NET_CORE_START_TIME_FOR_FILETIME)
#define MAX_ARGS_COUNT 5
// Имя сокетa
#define OCSPMONITOR_HOST "127.0.0.1"
// Порт на котором монитор ожидает вызовов NetCore

```

```

#define OCSPMONITOR_PORT 8082
BOOL COcspMonitor::m_IsRunning =
FALSE;
CrlManager* COcspMonitor::m_pCrlManager = NULL;
COCSPPResponder* COcspMonitor::m_pOCSPResponder = NULL;
IOcspClient* COcspMonitor::m_pOCSPClient = NULL;
IChachedStore* COcspMonitor::m_pChachedStore = NULL;
CPkcs11Client* COcspMonitor::m_pPkcs11 = NULL;
CX509Certificate2* COcspMonitor::m_pOcspCertificate = NULL;
OcspResponderStates COcspMonitor::m_CurrentState =
OcspResponderStates::Start;
OcspResponderStates COcspMonitor::m_PrevState =
OcspResponderStates::Start;
DWORD COcspMonitor::m_ClientId = 0;
const char* COcspMonitor::m_Host = NULL;
short COcspMonitor::m_Port = 0;
#define CHECK_CK_STATUS_NETERR(rv, mes)\
    if (CKR_OK != rv)\
    {\
        ErrorState(mes, rv);\
        return S_OK;\
    }\
    \\
/// <summary>
/// Запуск сервера NetCore, для связи с управляющей консолью
/// </summary>
/// <returns>Результат выполнения операции.</returns>
HRESULT COcspMonitor::Start()
{
    return Start(OCSPMONITOR_HOST, OCSPMONITOR_PORT);
}
/// <summary>
/// Запуск сервера NetCore, для связи с управляющей консолью
/// </summary>
/// <param name="Host">Хост сервера.</param>
/// <param name="Port">Порт сервера.</param>
/// <returns>Результат выполнения операции.</returns>
HRESULT COcspMonitor::Start(const char* Host,
                             short Port)
{
    if (m_IsRunning)
    {
        return ERROR_SERVICE_ALREADY_RUNNING;
    }
    m_pPkcs11 = new
COcspPkcs11Client(COcspSettings::GetWorkWithTwoKeys());
    if (!m_pPkcs11->IsInitialized())
    {
        ERROR_WRITE("Не удалось инициализировать csCryptoki");
        return E_FAIL;
    }
    m_pChachedStore = CChachedStore::GetInstance();
    ///#if defined(WIN32) && defined(DEBUG)
    /// m_pChachedStore = new CTestStore();
    ///#else
    /// m_pChachedStore = CChachedStore::GetInstance();
    ///#endif
    m_pCrlManager = new CrlManager(m_pChachedStore);
    HRESULT Status = RegisterFuncs();
    CHECK_HR_STATUS_RET1(Status);
    Status = NcStartServer(Host, Port);
    CHECK_HR_STATUS_RET1(Status);
    m_IsRunning = TRUE;
    ChangeState(OcspResponderStates::SelectFirstToken);
    LOG_WRITE("Начало работы OCSP монитора");
}

```

```

        return S_OK;
    }
    /// <summary>
    /// Остановка сервера связи с управляющей консолью
    /// </summary>
    /// <returns>Результат выполнения операции.</returns>
    HRESULT COcspMonitor::Stop()
    {
        if (!m_IsRunning)
            return S_OK;
        m_IsRunning = FALSE;
        HRESULT Status = S_OK;
        UnregisterFuncs();
        Status = NcStopServer();
        StopWorking();

        delete m_pCrlManager;
        delete m_pOCSPClient;
        delete m_pChachedStore;
        delete m_pPkcs11;
        delete m_pOcspCertificate;
        m_pCrlManager          = NULL;
        m_pOCSPClient          = NULL;
        m_pChachedStore        = NULL;
        m_pPkcs11              = NULL;
        m_pOcspCertificate     = NULL;
        m_CurrentState         = OcspResponderStates::Start;
        m_PrevState            = OcspResponderStates::Start;
        m_ClientId             = 0;
        m_Host                  = NULL;
        m_Port                  = 0;
        LOG_WRITE("Остановка работы OCSP монитора");
        return Status;
    }
    HRESULT COcspMonitor::RegisterFuncs()
    {
        HRESULT Status;
        BYTE TI[sizeof(NC_FUNCTION) + sizeof(GUID)*MAX_ARGS_COUNT];
        NC_FUNCTION* ti = (NC_FUNCTION*)&TI;
        ti->pwszFuncName = (PWCHAR)L"RegisterClient";
        ti->pfnFunc = RegisterClient;
        ti->dwArgCount = 1;
        ti->ArgIds[0] = NC_TYPE_INT;
        Status = NcRegisterFunc(ti);
        CHECK_HR_STATUS_RET1(Status);
        ti->pwszFuncName = (PWCHAR)L"UnregisterClient";
        ti->pfnFunc = UnregisterClient;
        ti->dwArgCount = 1;
        ti->ArgIds[0] = NC_TYPE_INT;
        Status = NcRegisterFunc(ti);
        CHECK_HR_STATUS_RET1(Status);
        ti->pwszFuncName = (PWCHAR)L"GetCurrentState";
        ti->pfnFunc = GetCurrentState;
        ti->dwArgCount = 0;
        Status = NcRegisterFunc(ti);
        CHECK_HR_STATUS_RET1(Status);
        ti->pwszFuncName = (PWCHAR)L"Reset";
        ti->pfnFunc = Reset;
        ti->dwArgCount = 0;
        Status = NcRegisterFunc(ti);
        CHECK_HR_STATUS_RET1(Status);
        ti->pwszFuncName = (PWCHAR)L"GetSlotList";
        ti->pfnFunc = GetSlotList;
        ti->dwArgCount = 0;
    }

```

```

Status = NcRegisterFunc(ti);
CHECK_HR_STATUS_RET1(Status);
ti->pwszFuncName = (PWCHAR)L"SelectSlot";
ti->pfnFunc = SelectSlot;
ti->dwArgCount = 1;
ti->ArgIds[0] = NC_TYPE_UINT;
Status = NcRegisterFunc(ti);
CHECK_HR_STATUS_RET1(Status);
ti->pwszFuncName = (PWCHAR)L"LoginOnSlot";
ti->pfnFunc = LoginOnSlot;
ti->dwArgCount = 1;
ti->ArgIds[0] = NC_TYPE_STRING;
Status = NcRegisterFunc(ti);
CHECK_HR_STATUS_RET1(Status);
ti->pwszFuncName = (PWCHAR)L"FindKeyShard";
ti->pfnFunc = FindKeyShard;
ti->dwArgCount = 0;
Status = NcRegisterFunc(ti);
CHECK_HR_STATUS_RET1(Status);
ti->pwszFuncName = (PWCHAR)L"FindKeyPart";
ti->pfnFunc = FindKeyPart;
ti->dwArgCount = 0;
Status = NcRegisterFunc(ti);
CHECK_HR_STATUS_RET1(Status);
ti->pwszFuncName = (PWCHAR)L"FindSignKeyFromOneToken";
ti->pfnFunc = FindSignKeyFromOneToken;
ti->dwArgCount = 0;
Status = NcRegisterFunc(ti);
CHECK_HR_STATUS_RET1(Status);
ti->pwszFuncName = (PWCHAR)L"MakeOcspSignKeyFromParts";
ti->pfnFunc = MakeOcspSignKeyFromParts;
ti->dwArgCount = 0;
Status = NcRegisterFunc(ti);
CHECK_HR_STATUS_RET1(Status);
ti->pwszFuncName = (PWCHAR)L"SetRootCertificate";
ti->pfnFunc = SetRootCertificate;
ti->dwArgCount = 1;
ti->ArgIds[0] = NC_TYPE_BYTE_ARRAY;
Status = NcRegisterFunc(ti);
CHECK_HR_STATUS_RET1(Status);
ti->pwszFuncName = (PWCHAR)L"SetOcspCertificate";
ti->pfnFunc = SetOcspCertificate;
ti->dwArgCount = 1;
ti->ArgIds[0] = NC_TYPE_BYTE_ARRAY;
Status = NcRegisterFunc(ti);
CHECK_HR_STATUS_RET1(Status);
ti->pwszFuncName = (PWCHAR)L"SetBaseCrl";
ti->pfnFunc = SetBaseCrl;
ti->dwArgCount = 0;
Status = NcRegisterFunc(ti);
CHECK_HR_STATUS_RET1(Status);
ti->pwszFuncName = (PWCHAR)L"SetDeltaCrl";
ti->pfnFunc = SetDeltaCrl;
ti->dwArgCount = 0;
Status = NcRegisterFunc(ti);
CHECK_HR_STATUS_RET1(Status);
ti->pwszFuncName = (PWCHAR)L"GetCurrentBaseCrl";
ti->pfnFunc = GetCurrentBaseCrl;
ti->dwArgCount = 0;
Status = NcRegisterFunc(ti);
CHECK_HR_STATUS_RET1(Status);
ti->pwszFuncName = (PWCHAR)L"GetCurrentDeltaCrl";
ti->pfnFunc = GetCurrentDeltaCrl;
ti->dwArgCount = 0;

```

```

    Status = NcRegisterFunc(ti);
    CHECK_HR_STATUS_RET1(Status);
    ti->pwszFuncName = (PWCHAR)L"GetCurrentBaseCrlInstallTime";
    ti->pfnFunc = GetCurrentBaseCrlInstallTime;
    ti->dwArgCount = 0;
    Status = NcRegisterFunc(ti);
    CHECK_HR_STATUS_RET1(Status);
    ti->pwszFuncName = (PWCHAR)L"GetCurrentDeltaCrlInstallTime";
    ti->pfnFunc = GetCurrentDeltaCrlInstallTime;
    ti->dwArgCount = 0;
    Status = NcRegisterFunc(ti);
    CHECK_HR_STATUS_RET1(Status);
    ti->pwszFuncName = (PWCHAR)L"UpdateCrls";
    ti->pfnFunc = UpdateCrls;
    ti->dwArgCount = 0;
    Status = NcRegisterFunc(ti);
    CHECK_HR_STATUS_RET1(Status);
    ti->pwszFuncName = (PWCHAR)L"Beacon";
    ti->pfnFunc = Beacon;
    ti->dwArgCount = 1;
    ti->ArgIds[0] = NC_TYPE_INT;
    Status = NcRegisterFunc(ti);
    CHECK_HR_STATUS_RET1(Status);
    return S_OK;
}
HRESULT COcspMonitor::UnregisterFuncs()
{
    HRESULT Status = E_FAIL;
    Status = NcUnregisterFunc((PWCHAR)L"RegisterClient");
    CHECK_HR_STATUS_RET1(Status);
    Status = NcUnregisterFunc((PWCHAR)L"UnregisterClient");
    CHECK_HR_STATUS_RET1(Status);
    Status = NcUnregisterFunc((PWCHAR)L"GetCurrentState");
    CHECK_HR_STATUS_RET1(Status);
    Status = NcUnregisterFunc((PWCHAR)L"Reset");
    CHECK_HR_STATUS_RET1(Status);
    Status = NcUnregisterFunc((PWCHAR)L"GetSlotList");
    CHECK_HR_STATUS_RET1(Status);
    Status = NcUnregisterFunc((PWCHAR)L"SelectSlot");
    CHECK_HR_STATUS_RET1(Status);
    Status = NcUnregisterFunc((PWCHAR)L"LoginOnSlot");
    CHECK_HR_STATUS_RET1(Status);
    Status = NcUnregisterFunc((PWCHAR)L"FindKeyShard");
    CHECK_HR_STATUS_RET1(Status);
    Status = NcUnregisterFunc((PWCHAR)L"FindKeyPart");
    CHECK_HR_STATUS_RET1(Status);
    Status = NcUnregisterFunc((PWCHAR)L"FindSignKeyFromOneToken");
    CHECK_HR_STATUS_RET1(Status);
    Status = NcUnregisterFunc((PWCHAR)L"MakeOcspSignKeyFromParts");
    CHECK_HR_STATUS_RET1(Status);
    Status = NcUnregisterFunc((PWCHAR)L"SetRootCertificate");
    CHECK_HR_STATUS_RET1(Status);
    Status = NcUnregisterFunc((PWCHAR)L"SetOcspCertificate");
    CHECK_HR_STATUS_RET1(Status);
    Status = NcUnregisterFunc((PWCHAR)L"SetBaseCrl");
    CHECK_HR_STATUS_RET1(Status);
    Status = NcUnregisterFunc((PWCHAR)L"SetDeltaCrl");
    CHECK_HR_STATUS_RET1(Status);
    Status = NcUnregisterFunc((PWCHAR)L"GetCurrentBaseCrl");
    CHECK_HR_STATUS_RET1(Status);
    Status = NcUnregisterFunc((PWCHAR)L"GetCurrentDeltaCrl");
    CHECK_HR_STATUS_RET1(Status);
    Status = NcUnregisterFunc((PWCHAR)L"GetCurrentBaseCrlInstallTime");
    CHECK_HR_STATUS_RET1(Status);

```

```

        Status = NcUnregisterFunc((PWCHAR)L"GetCurrentDeltaCrlInstallTime");
        CHECK_HR_STATUS_RET1(Status);
        Status = NcUnregisterFunc((PWCHAR)L"UpdateCrls");
        CHECK_HR_STATUS_RET1(Status);
        Status = NcUnregisterFunc((PWCHAR)L"Beacon");
        CHECK_HR_STATUS_RET1(Status);
        return S_OK;
    }

    /// <summary>
    /// Регистрация нового клиента.
    /// </summary>
    /// <param name="retVoid">Ничего, либо исключение.</param>
    /// <param name="clientIdInt">Уникальный идентификатор клиента.</param>
    /// <returns>Результат выполнения операции.</returns>
    HRESULT COcspMonitor::RegisterClient(PNC_OBJECT retVoid, PVOID clientIdInt)
    {
        LOG_PROC_ENTRY;
        CHECK_FOR_INTERNAL_ERROR_RET("Служба завершается", m_IsRunning);
        m_ClientId = (DWORD)clientIdInt;
        m_pOCSPClient = new COcspClient((int)clientIdInt);
        if (!m_pOCSPClient->Beacon((int)clientIdInt))
        {
            ERROR_WRITE("Клиент не отвечает на запрос");
            return S_OK;
        }
        LOG_WRITE("Зарегистрирована консоль службы");
        return S_OK;
    }

    /// <summary>
    /// Удаление клиента.
    /// </summary>
    /// <param name="retVoid">Ничего, либо исключение.</param>
    /// <param name="clientIdInt">Уникальный идентификатор клиента.</param>
    /// <returns>Результат выполнения операции.</returns>
    HRESULT COcspMonitor::UnregisterClient(PNC_OBJECT retVoid, PVOID clientIdInt)
    {
        LOG_PROC_ENTRY;
        delete m_pOCSPClient;
        m_pOCSPClient = NULL;
        LOG_WRITE("Консоль службы отключена");
        return S_OK;
    }

    /// <summary>
    /// Получение состояния конечного автомата у службы.
    /// </summary>
    /// <param name="retUInt">Номер текущего состояния.</param>
    /// <returns>Результат выполнения операции.</returns>
    HRESULT COcspMonitor::GetCurrentState(PNC_OBJECT retUInt)
    {
        LOG_PROC_ENTRY;
        CHECK_FOR_INTERNAL_ERROR_RET("Служба завершается", m_IsRunning);
        retUInt->TypeId = NC_TYPE_UINT;
        retUInt->pObject = (PVOID)m_CurrentState;

        return S_OK;
    }

    /// <summary>
    /// Перезапуск процесса.
    /// </summary>
    /// <param name="retVoid">Ничего, либо исключение.</param>
    /// <returns>Результат выполнения операции.</returns>
    HRESULT COcspMonitor::Reset(PNC_OBJECT retVoid)
    {
        LOG_PROC_ENTRY;

```

```

m_IsRunning = FALSE;
StopWorking();

delete m_pCrlManager;
delete m_pPkcs11;
delete m_pOCSPClient;
delete m_pOcspCertificate;
m_pChachedStore->Clear();
m_pCrlManager = new CrlManager(m_pChachedStore);
m_pPkcs11 = new COcspPkcs11Client(COcspSettings::GetWorkWithTwoKeys());
if (!m_pPkcs11->IsInitialized())
{
    ERROR_WRITE("Не удалось инициализировать csCryptoki");
    return E_FAIL;
}

m_pOCSPClient          = NULL;
m_pOcspCertificate     = NULL;
m_CurrentState         = OcspResponderStates::Start;
m_PrevState            = OcspResponderStates::Start;
m_IsRunning = TRUE;
ChangeState(OcspResponderStates::SelectFirstToken);
return S_OK;
}
#define WRITE_TO_BUFFER(Buffer, Position, Data, Count) \
{
    \
    memcpy(pBuffer + Position, Data, Count);          \
    Position += Count;                                \
}
/// <summary>
/// Получение списка слотов.
/// </summary>
/// <param name="retBytes">Список слотов, которые может использовать
служба.</param>
/// <returns>Результат выполнения операции.</returns>
HRESULT COcspMonitor::GetSlotList(PNC_OBJECT retBytes)
{
    LOG_PROC_ENTRY;
    HRESULT Status =
E_FAIL;
    TokenInfo* pSlotInfo = NULL;
    DWORD CountSlots = 0;
    PBYTE pBuffer = NULL;
    const BYTE MaxLengthSlotLabel = 32;
    CHECK_FOR_INTERNAL_ERROR_RET("Служба завершается", m_IsRunning);
    __try
    {
        Status = m_pPkcs11->GetTokensInfo(pSlotInfo, CountSlots);
        CHECK_HR_STATUS_LEAVE("Ошибка m_pPkcs11->GetTokensInfo", Status);
        // Длина буфера = Число слотов + (для каждого слота: id + длина
метки + метка, максимальная длина метки = 32).
        DWORD LengthBuffer = sizeof(DWORD) + CountSlots *
(sizeof(DWORD) + sizeof(BYTE) + MaxLengthSlotLabel);
        DWORD Position = 0;
        pBuffer = new BYTE[LengthBuffer];
        memset(pBuffer, 0, LengthBuffer);
        WRITE_TO_BUFFER(pBuffer, Position, &CountSlots, sizeof(DWORD));
        for (DWORD i = 0; i < CountSlots; i++)
        {
            WRITE_TO_BUFFER(pBuffer, Position, &pSlotInfo[i].slotId,
sizeof(DWORD));
            WRITE_TO_BUFFER(pBuffer, Position, &MaxLengthSlotLabel,
sizeof(BYTE));

```

```

        WRITE_TO_BUFFER(pBuffer, Position,
pSlotInfo[i].tokenInfo.label, MaxLengthSlotLabel);
    }
    Status = NC_OBJECT_ByteArrayCreate(pBuffer, LengthBuffer,
retBytes);
    CHECK_HR_STATUS_LEAVE("NC_OBJECT_ByteArrayCreate", Status);
    Status = S_OK;
}
__finally
{
    delete[] pSlotInfo;
    delete[] pBuffer;
    if (FAILED(Status))
    {
        HRESULT Status = NC_OBJECT_ByteArrayCreate(NULL, 0,
retBytes);
        LogStatus("NC_OBJECT_ByteArrayCreate не удалось создать
пустой буффер", Status);
    }
    return Status;
}
/// <summary>
/// Выбор текущего слота.
/// </summary>
/// <param name="retVoid">Ничего, либо исключение.</param>
/// <param name="slotUInt">Описание слота.</param>
/// <returns>Результат выполнения операции.</returns>
HRESULT COcspMonitor::SelectSlot(PNC_OBJECT retVoid, PVOID slotUInt)
{
    LOG_PROC_ENTRY;
    CHECK_FOR_INTERNAL_ERROR_RET("Служба завершается", m_IsRunning);
    CK_RV rv = E_FAIL;
    // Определяем, можем ли мы выполнить данную операцию.
    if (m_CurrentState != OcspResponderStates::SelectFirstToken &&
m_CurrentState != OcspResponderStates::SelectSecondToken)
    {
        ErrorState(L"Невозможно выполнить данную операцию из текущего
состояния.");
        return S_OK;
    }
    __try
    {
        rv = m_pPkcs11->SelectSlot((DWORD)slotUInt);
        CHECK_CK_STATUS_LEAVE("Ошибка SelectSlot", rv);
        // Определяем следующее состояние.
        if (m_CurrentState == OcspResponderStates::SelectFirstToken)
        {
            ChangeState(OcspResponderStates::LoginFirstToken);
        }
        else if (m_CurrentState == OcspResponderStates::SelectSecondToken)
        {
            ChangeState(OcspResponderStates::LoginSecondToken);
        }
        LOG_WRITE("Слот выбран");
    }
    __finally
    {
        if (CKR_OK != rv)
        {
            ErrorState(L"При выборе слота произошла ошибка", rv);
        }
    }
    return S_OK;
}

```

```

}
/// <summary>
/// Выполнить аутентификацию на выбранном слоте.
/// </summary>
/// <param name="retVoid">Ничего, либо исключение.</param>
/// <param name="passwordString">Пароль.</param>
/// <returns>Результат выполнения операции.</returns>
HRESULT COcspMonitor::LoginOnSlot(PNC_OBJECT retVoid, PVOID passwordString)
{
    LOG_PROC_ENTRY;
    CHECK_FOR_INTERNAL_ERROR_RET("Служба завершается", m_IsRunning);
    if (!passwordString)
        return E_INVALIDARG;
    // Определяем, можем ли мы выполнить данную операцию.
    if (m_CurrentState != OcspResponderStates::LoginFirstToken &&
        m_CurrentState != OcspResponderStates::LoginSecondToken)
    {
        ErrorState(L"Невозможно выполнить данную операцию из текущего
состояния.");
        return S_OK;
    }
    __try
    {
        PWCHAR Password = (PWCHAR)passwordString;
        DWORD PasswordLength = wcslen(Password);
        PBYTE pUtf8String = new BYTE[PasswordLength];
        // Преобразование в utf-8 (символы только однобайтовые)
        for (DWORD i = 0; Password[i]; ++i)
        {
            if (Password[i] > 0xFF)
            {
                ErrorState(L"В пароле встречены некорректные
символы");
                return S_OK;
            }
            else
            {
                pUtf8String[i] = (BYTE)Password[i];
            }
        }
        CK_RV rv = m_pPkcs11->Login(pUtf8String, PasswordLength);
        delete[] pUtf8String;
        CHECK_CK_STATUS_NETERR(rv, L"Не удалось инициализироваться в
токене");
        LOG_WRITE("Авторизация проведена");
        ChangeState(OcspResponderStates::SetRootCert);
    }
    __finally
    {
    }
    return S_OK;
}
/// <summary>
/// Поиск половинки ключа администратора.
/// </summary>
/// <param name="retVoid">Ничего, либо исключение.</param>
/// <returns>Результат выполнения операции.</returns>
HRESULT COcspMonitor::FindKeyShard(PNC_OBJECT retVoid)
{
    LOG_PROC_ENTRY;
    CHECK_FOR_INTERNAL_ERROR_RET("Служба завершается", m_IsRunning);
    // Определяем, можем ли мы выполнить данную операцию.
    if (m_CurrentState != OcspResponderStates::FindKeyShard)
    {

```

```

        ErrorState(L"Невозможно выполнить данную операцию из текущего
состояния.");
        return S_OK;
    }
    CK_RV rv = m_pPkcs11->FindKeyShardById();
    CHECK_CK_STATUS_NETERR(rv, L"Ошибка FindKeyShardById");
    LOG_WRITE("Найдена половинка ключа администратора");
    ChangeState(OcspResponderStates::SelectSecondToken);
    return S_OK;
}
/// <summary>
/// Поиск половинки ключа администратора сертификации.
/// </summary>
/// <param name="retVoid">Ничего, либо исключение.</param>
/// <returns>Результат выполнения операции.</returns>
HRESULT COcspMonitor::FindKeyPart(PNC_OBJECT retVoid)
{
    LOG_PROC_ENTRY;
    CHECK_FOR_INTERNAL_ERROR_RET("Служба завершается", m_IsRunning);
    // Определяем, можем ли мы выполнить данную операцию.
    if (m_CurrentState != OcspResponderStates::FindKeyPart)
    {
        ErrorState(L"Невозможно выполнить данную операцию из текущего
состояния.");
        return S_OK;
    }
    CK_RV rv = m_pPkcs11->FindKeyPartById();
    CHECK_CK_STATUS_NETERR(rv, L"Ошибка FindKeyPartById");
    LOG_WRITE("Найдена половинка ключа администратора сертификации");
    ChangeState(OcspResponderStates::MakeSignKey);
    return S_OK;
}
/// <summary>
/// Поиск ключа подписи службы.
/// </summary>
/// <param name="retVoid">Ничего, либо исключение.</param>
/// <returns>Результат выполнения операции.</returns>
HRESULT COcspMonitor::FindSignKeyFromOneToken(PNC_OBJECT retVoid)
{
    LOG_PROC_ENTRY;
    CHECK_FOR_INTERNAL_ERROR_RET("Служба завершается", m_IsRunning);
    // Определяем, можем ли мы выполнить данную операцию.
    if (m_CurrentState != OcspResponderStates::FindSignKeyFromOneToken)
    {
        ErrorState(L"Невозможно выполнить данную операцию из текущего
состояния.");
        return S_OK;
    }
    CK_RV rv = m_pPkcs11->FindSignKeyFromOneToken();
    CHECK_CK_STATUS_NETERR(rv, L"Ошибка FindSignKeyFromOneToken");
    LOG_WRITE("Найден ключ подписи службы");
    ChangeState(OcspResponderStates::SetBaseCrl);
    return S_OK;
}
/// <summary>
/// Выполнить сборку ключа подписи службы OSCP из 2-х частей.
/// </summary>
/// <param name="retVoid">Ничего, либо исключение.</param>
/// <returns>Результат выполнения операции.</returns>
HRESULT COcspMonitor::MakeOcspSignKeyFromParts(PNC_OBJECT retVoid)
{
    LOG_PROC_ENTRY;
    CHECK_FOR_INTERNAL_ERROR_RET("Служба завершается", m_IsRunning);
    // Определяем, можем ли мы выполнить данную операцию.

```

```

        if (m_CurrentState != OcspResponderStates::MakeSignKey)
        {
            ErrorState(L"Невозможно выполнить данную операцию из текущего
состояния.");
            return S_OK;
        }
        CK_RV rv = m_pPkcs11->MakeSignKeyFromKeyShard();
        CHECK_CK_STATUS_NETERR(rv, L"Не удалось найти ключ подписи");
        LOG_WRITE("Собран ключ подписи службы OCSP");
        ChangeState(OcspResponderStates::SetBaseCrl);
        return S_OK;
    }
    /// <summary>
    /// Установка корневого сертификата в ДБУ с заданным серийным номером.
    /// </summary>
    /// <param name="retVoid">Ничего, либо исключение.</param>
    /// <param name="serialNumber">Серийный номер устанавливаемого
сертификата.</param>
    /// <returns>Результат выполнения операции.</returns>
    HRESULT COcspMonitor::SetRootCertificate(PNC_OBJECT retVoid, PVOID
serialNumber)
    {
        LOG_PROC_ENTRY;
        if (!serialNumber)
            return E_INVALIDARG;
        HRESULT Status = E_FAIL;
        CK_RV rv = CKR_OK;
        CHECK_FOR_INTERNAL_ERROR_RET("Служба завершается", m_IsRunning);
        // Определяем, можем ли мы выполнить данную операцию.
        if (m_CurrentState != OcspResponderStates::SetRootCert)
        {
            ErrorState(L"Невозможно выполнить данную операцию из текущего
состояния.");
            return S_OK;
        }
        CX509Certificate2* pCertificate2 = m_pChachedStore-
>GetCertificate((CByteBuffer*)serialNumber, NULL);
        if (!pCertificate2)
        {
            ErrorState(L"Не удалось получить корневой сертификат.");
            return S_OK;
        }
        CX509SubjectKeyIdentifierExtension* pExtension = NULL;
        __try
        {
            CByteBuffer PublicKey;
            Status = pCertificate2->GetPublicKey(PublicKey);
            CHECK_HR_STATUS_LEAVE1(Status);
            Status = pCertificate2-
>GetExtension((PWCHAR)OID_X509_SUBJECT_KEY_IDENTIFIER_EXT,
wcslen(OID_X509_SUBJECT_KEY_IDENTIFIER_EXT), (CX509Extension**)&pExtension);
            CHECK_HR_STATUS_LEAVE1(Status);
            rv = m_pPkcs11->SetCertificate(pCertificate2->GetData(),
PublicKey, pExtension->GetKeyIdentifier(), TRUE);
            CHECK_CK_STATUS_LEAVE("При установке корневого сертификата
произошла ошибка", Status);
            // Сохраняем серийный номер текущего корневого сертификата.
            m_pChachedStore-
>SetCurrentRootCertificateSerialNumber(pCertificate2->GetSerialNumber());
            LOG_WRITE("Установлен корневой сертификат");
            Status = S_OK;
            ChangeState(OcspResponderStates::SetOcspCert);
        }
        __finally

```

```

    {
        delete pExtension;
        if (FAILED(Status) || rv != S_OK)
        {
            ErrorState(L"При установке корневого сертификата произошла
ошибка.", rv);
        }
        delete pCertificate2;
    }
    return S_OK;
}
/// <summary>
/// Установка сертификата службы OCSP в ДВУ с заданным серийным номером.
/// </summary>
/// <param name="retVoid">Ничего, либо исключение.</param>
/// <param name="serialNumber">Серийный номер устанавливаемого
сертификата.</param>
/// <returns>Результат выполнения операции.</returns>
HRESULT COcspMonitor::SetOcspCertificate(PNC_OBJECT retVoid, PVOID
serialNumber)
{
    LOG_PROC_ENTRY;
    if (!serialNumber)
        return E_INVALIDARG;
    HRESULT Status = E_FAIL;
    CK_RV rv = CKR_OK;
    CHECK_FOR_INTERNAL_ERROR_RET("Служба завершается", m_IsRunning);
    // Определяем, можем ли мы выполнить данную операцию.
    if (m_CurrentState != OcspResponderStates::SetOcspCert)
    {
        ErrorState(L"Невозможно выполнить данную операцию из текущего
состояния.");
        return S_OK;
    }
    CX509Certificate2* pCertificate2 = m_pChachedStore-
>GetCertificate((CByteBuffer*)serialNumber, NULL);
    if (!pCertificate2)
    {
        ErrorState(L"Не удалось получить службы OCSP.");
        return S_OK;
    }
    CX509SubjectKeyIdentifierExtension* pExtension = NULL;
    __try
    {
        CByteBuffer PublicKey;
        Status = pCertificate2->GetPublicKey(PublicKey);
        CHECK_HR_STATUS_LEAVE1(Status);
        Status = pCertificate2-
>GetExtension((PWCHAR)OID_X509_SUBJECT_KEY_IDENTIFIER_EXT,
wcslen(OID_X509_SUBJECT_KEY_IDENTIFIER_EXT), (CX509Extension**) &pExtension);
        CHECK_HR_STATUS_LEAVE1(Status);
        rv = m_pPkcs11->SetCertificate(pCertificate2->GetData(),
PublicKey, pExtension->GetKeyIdentifier(), FALSE);
        CHECK_CK_STATUS_LEAVE("При установке сертификата службы произошла
ошибка", Status);

        m_pChachedStore-
>SetCurrentOcspCertificateSerialNumber(pCertificate2->GetSerialNumber());
        m_pOcspCertificate = pCertificate2;
        LOG_WRITE("Установлен сертификат OCSP службы");
        Status = S_OK;

        // Определяем следующее состояние.
        if (COcspSettings::GetWorkWithTwoKeys())

```

```

        {
            // TODO Не проверялось.
            ChangeState(OcspResponderStates::FindKeyShard);
        }
        else
        {
            ChangeState(OcspResponderStates::FindSignKeyFromOneToken);
        }
    }
    __finally
    {
        delete pExtension;
        if (FAILED(Status) || rv != S_OK)
        {
            delete pCertificate2;
            ErrorState(L"При установке корневого сертификата произошла
ошибка.", rv);
        }
    }
    return S_OK;
}
/// <summary>
/// Установка актуального основного списка отзыва.
/// </summary>
/// <param name="retVoid">Ничего, либо исключение.</param>
/// <returns>Результат выполнения операции.</returns>
HRESULT COcspMonitor::SetBaseCrl(PNC_OBJECT retVoid)
{
    LOG_PROC_ENTRY;
    CHECK_FOR_INTERNAL_ERROR_RET("Служба завершается", m_IsRunning);
    // Определяем, можем ли мы выполнить данную операцию.
    if (m_CurrentState != OcspResponderStates::SetBaseCrl)
    {
        ErrorState(L"Невозможно выполнить данную операцию из текущего
состояния.");
        return S_OK;
    }
    CX509Crl* pCrl = m_pChachedStore->GetActualCrl();
    if (!pCrl)
    {
        ErrorState(L"Не удалось получить актуальный базовый список
отозванных сертификатов.");
        return S_OK;
    }
    if (!m_pCrlManager->SetBaseCrl(pCrl))
    {
        ErrorState(L"Не удалось установить актуальный базовый список
отозванных сертификатов.");
        return S_OK;
    }
    LOG_WRITE("Установлен базовый список отзывов");
    // Определяем следующее состояние.
    if (pCrl->IsBase())
        ChangeState(OcspResponderStates::SetDeltaCrl);
    else
        ChangeState(OcspResponderStates::Working);
    return S_OK;
}
/// <summary>
/// Установка актуального разностного списка отзыва.
/// </summary>
/// <param name="retVoid">Ничего, либо исключение.</param>
/// <returns>Результат выполнения операции.</returns>
HRESULT COcspMonitor::SetDeltaCrl(PNC_OBJECT retVoid)

```

```

{
    LOG_PROC_ENTRY;
    CHECK_FOR_INTERNAL_ERROR_RET("Служба завершается", m_IsRunning);
    // Определяем, можем ли мы выполнить данную операцию.
    if (m_CurrentState != OcspResponderStates::SetDeltaCrl)
    {
        ErrorState(L"Невозможно выполнить данную операцию из текущего
состояния.");
        return S_OK;
    }
    CX509Crl* pCrl = m_pChachedStore->GetDeltaCrl();
    if (!pCrl)
    {
        ErrorState(L"Не удалось получить актуальный разностный список
отозванных сертификатов.");
        return S_OK;
    }
    if (!m_pCrlManager->SetDeltaCrl(pCrl))
    {
        ErrorState(L"Не удалось установить актуальный разностный список
отозванных сертификатов.");
        return S_OK;
    }
    LOG_WRITE("Установлен разностный список отзывает");
    ChangeState(OcspResponderStates::Working);
    return S_OK;
}
/// <summary>
/// Получение текущего установленного основного списка отозванных
сертификатов.
/// </summary>
/// <returns>Результат выполнения операции.</returns>
HRESULT COcspMonitor::GetCurrentBaseCrl(PNC_OBJECT retBytes)
{
    LOG_PROC_ENTRY;
    CX509Crl* pCrl = m_pCrlManager->GetBaseCrl();

    __try
    {
        if (pCrl)
        {
            CByteBuffer Buffer = pCrl->GetData();
            HRESULT Status = NC_OBJECT_ByteArrayCreate(Buffer.Data,
Buffer.Size, retBytes);
            CHECK_HR_STATUS_RET1(Status);
        }
        else
        {
            retBytes->pObject = NULL;
        }
    }
    __finally
    {
        delete pCrl;
    }
    return S_OK;
}
/// <summary>
/// Получение текущего установленного разностного списка отозванных
сертификатов.
/// </summary>
/// <returns>Результат выполнения операции.</returns>
HRESULT COcspMonitor::GetCurrentDeltaCrl(PNC_OBJECT retBytes)
{

```

```

LOG_PROC_ENTRY;
CX509Crl* pCrl = m_pCrlManager->GetDeltaCrl();

__try
{
    if (pCrl)
    {
        CByteBuffer Buffer = pCrl->GetData();
        HRESULT Status = NC_OBJECT_ByteArrayCreate(Buffer.Data,
Buffer.Size, retBytes);
        CHECK_HR_STATUS_RET1(Status);
    }
    else
    {
        retBytes->pObject = NULL;
    }
}
__finally
{
    delete pCrl;
}
return S_OK;
}
/// <summary>
/// Получение времени установки основного CAC.
/// </summary>
/// <param name="retDateTime">Время установки основного CAC.</param>
/// <returns>Результат выполнения операции.</returns>
HRESULT COcspMonitor::GetCurrentBaseCrlInstallTime(PNC_OBJECT retDateTime)
{
    LOG_PROC_ENTRY;
    INT64 Time = FILETIME_TO_DATETIME(m_pCrlManager-
>GetBaseCrlInstallTime());

    return NC_OBJECT_DatetimeCreate(Time, retDateTime);
}
/// <summary>
/// Получение времени установки разностного CAC.
/// </summary>
/// <param name="retDateTime">Время установки основного CAC.</param>
/// <returns>Результат выполнения операции.</returns>
HRESULT COcspMonitor::GetCurrentDeltaCrlInstallTime(PNC_OBJECT retDateTime)
{
    LOG_PROC_ENTRY;
    INT64 Time = FILETIME_TO_DATETIME(m_pCrlManager-
>GetDeltaCrlInstallTime());

    return NC_OBJECT_DatetimeCreate(Time, retDateTime);
}
/// <summary>
/// Выполнить попытку принудительного обновления CAC.
/// </summary>
/// <param name="retVoid">Ничего, либо исключение.</param>
/// <returns>Результат выполнения операции.</returns>
HRESULT COcspMonitor::UpdateCrls(PNC_OBJECT retVoid)
{
    m_pCrlManager->UpdateCrls();
    return S_OK;
}
/// <summary>
/// Маячок для клиента.
/// </summary>
/// <param name="retBool">Зарегистрирован ли клиент.</param>
/// <param name="idInt">Идентификатор клиента.</param>

```

```

/// <returns>Результат выполнения операции.</returns>
HRESULT COcspMonitor::Beacon(PNC_OBJECT retBool, PVOID idInt)
{
    LOG_PROC_ENTRY;
    CHECK_FOR_INTERNAL_ERROR_RET("Служба завершается", m_IsRunning);
    // TODO Где это проверять
    // Чтобы не делать это в отдельном потоке, состояние OCSPRespondera
    проверяется здесь
    if (m_pOCSPResponder && !m_pOCSPResponder->GetStatus())
    {
        ErrorState(L"При обработке запросов произошла ошибка");
    }
    retBool->TypeId = NC_TYPE_BOOL;
    retBool->pObject = (PVOID) (m_ClientId == (DWORD)idInt);
    return S_OK;
}
/// <summary>
/// Изменение состояния службы
/// </summary>
/// <param name="newState">Новое состояние.</param>
void
COcspMonitor::ChangeState(OcspResponderStates newState)
{
    LOG_PROC_ENTRY;
    HRESULT Status = E_FAIL;
    m_PrevState = m_CurrentState;
    m_CurrentState = newState;
    if (newState == OcspResponderStates::Working)
    {
        Status = StartWorking();
        if (FAILED(Status))
        {
            ERROR_WRITE("Не удалось начать работу");
            // TODO Что делать, если FAILED
            ErrorState(L"Не удалось начать работу");
        }
    }
    // Оповещаем клиента.
    if (m_pOCSPClient)
    {
        // TODO Оповещать клиента можно только после возврата управления
        из метода NetCore, иначе клиент попадет в deadlock. (предположительно)
        //m_pOCSPClient->SetLastState((int)m_CurrentState);
    }
}
/// <summary>
/// Обработчик состояния ошибки
/// </summary>
/// <param name="message">Сообщение об ошибке.</param>
/// <param name="error">Код ошибки.</param>
void
COcspMonitor::ErrorState(
    __in wstring message,
    __in CK_RV error)
{
    LOG_PROC_ENTRY;
    if (!m_IsRunning)
        return;
    __try
    {
        ChangeState(OcspResponderStates::Error);
        m_IsRunning = FALSE;
        // Остановка обработки запросов
        StopWorking();
    }
}

```

```

m_pPkcs11->CloseOpenedKeys();
//LogWStr("message", message.c_str());
if (error != CKR_OK)
{
    LogCKStatus("Ошибка в COCSPResponder", error);
}
if (m_pOCSPClient)
    m_pOCSPClient->ShowError(message, L"");
BOOL IsCriticalError = FALSE;
switch (error)
{
case(CKR_D_TCU_LOCKED_CRITICAL):
case(CKR_D_KEY_LOCKED):
case(CKR_D_KEY_EXPIRED):
case(CKR_D_TIME_INVALID):
case(CKR_D_CRL_OLDER_THAN_CURRENT):
case(CKR_D_CRL_DELTA_REFERENCE_TO_ANOTHER_CRL):
case(CKR_D_CRL_CORRUPTED):
    IsCriticalError = TRUE;
    ChangeState(OcspResponderStates::Blocked);
    break;
default: ;
}
// Определяем следующее состояние.
if (!IsCriticalError)
{
    switch (m_PrevState)
    {
    case (OcspResponderStates::SelectFirstToken):
    case (OcspResponderStates::LoginFirstToken):
    case (OcspResponderStates::SetRootCert):
    case (OcspResponderStates::SetOcspCert):
    case (OcspResponderStates::FindSignKeyFromOneToken):
    case (OcspResponderStates::FindKeyShard):
    case (OcspResponderStates::SelectSecondToken):
    case (OcspResponderStates::LoginSecondToken):
    case (OcspResponderStates::FindKeyPart):
    case (OcspResponderStates::MakeSignKey):
    case (OcspResponderStates::SetBaseCrl):
    case (OcspResponderStates::SetDeltaCrl):
    case (OcspResponderStates::Working):
        ChangeState(OcspResponderStates::Start);
        break;
    default: ;
    }
}
__finally
{
    m_IsRunning = TRUE;
}
}
/// <summary>
/// Переход в рабочий режим
/// </summary>
/// <returns>Результат выполнения операции.</returns>
HRESULT
COcspMonitor::StartWorking()
{
    LOG_PROC_ENTRY;
    HRESULT Status = E_FAIL;
    __try
    {
        Status = m_pCrlManager->StartObservation();
    }
}

```

```

        CHECK_HR_STATUS_LEAVE("Не удалось начать слежение за обновлением
COC", Status);
        delete m_pOCSPResponder;
        m_pOCSPResponder = new COCSPResponder(m_pOcspCertificate,
m_pCrlManager, m_pChachedStore, m_pPkcs11);
        Status = m_pOCSPResponder->Start();
        CHECK_HR_STATUS_LEAVE("Не удалось начать слежение за OCSP
запросами", Status);
        Status = S_OK;
    }
    __finally
    {
        if (FAILED(Status))
        {
            StopWorking();
        }
    }
    return Status;
}
/// <summary>
/// Выход из рабочего режима
/// </summary>
/// <returns>Результат выполнения операции.</returns>
VOID
COCspMonitor::StopWorking()
{
    LOG_PROC_ENTRY;

    m_pCrlManager->StopObservation();
    if (m_pOCSPResponder)
    {
        m_pOCSPResponder->Stop();
        delete m_pOCSPResponder;
        m_pOCSPResponder = NULL;
    }
    return;
}

```

Листинг A.3 – OCSPResponder.h

```

#pragma once

#include "fcgi_config_x86.h"

#include "fcgi_stdio.h"

/// <summary>
/// С помощью библиотеки FastCGI обрабатывает OCSP запросы.
/// </summary>

class COCSPResponder
{
public:

    /// <summary>

```

```

    /// Инициализация

    /// </summary>

    /// <param name="pOcspCertificate">Сертификат ocsp службы.</param>

    /// <param name="pCrlManager">Менеджер списка отзывов.</param>

    /// <param name="pChachedStore">Хранилище сертификатов.</param>

    /// <param name="pPkcs11">Криптография.</param>

    COCSPResponder(__in CX509Certificate2*
    pOcspCertificate,

    __in CrlManager*
    pCrlManager,

    __in IChachedStore*
    pChachedStore,

    __in CPkcs11Client*
    pPkcs11);

    ~COCSPResponder();

    /// <summary>

    /// TRUE, если производится работа.

    /// </summary>

    BOOL GetStatus();

    /// <summary>

    /// Непосредственно работа OCSP службы.

    /// </summary>

    HRESULT Start();

    /// <summary>

    /// Остановка работы

    /// </summary>

    void Stop();

private:

    /// <summary>

    /// Рабочий поток. Обслуживает запросы (можно создать несколько)

```



```

/// <param name="pPkcs11">Криптография.</param>
COCSPPResponder::COCSPPResponder(__in CX509Certificate2*
    pOcspCertificate,
                                   __in CrlManager*
    pCrlManager,
                                   __in IChachedStore*
    pChachedStore,
                                   __in CPkcs11Client*
    pPkcs11)
{
    LOG_PROC_ENTRY;
    m_Stopping = FALSE;
    m_ListenSocket = 0;
    m_pWorkerThreads = NULL;
    m_CountRequest = 0;
    m_CountThread = 1;
    m_pOcspCertificate = new CX509Certificate2(*pOcspCertificate);
    m_pPkcs11 = pPkcs11;
    m_pCrlManager = pCrlManager;
    m_pChachedStore = pChachedStore;
}
COCSPPResponder::~COCSPPResponder()
{
    if (m_pWorkerThreads)
    {
        Stop();
    }
    delete m_pOcspCertificate;

    m_pPkcs11 = NULL;
    m_pOcspCertificate = NULL;
    m_pCrlManager = NULL;
}
/// <summary>
/// TRUE, если производится работа.
/// </summary>
BOOL
COCSPPResponder::GetStatus()
{
    return !m_Stopping;
}
/// <summary>
/// Непосредственно работа OSCP службы.
/// </summary>
HRESULT
COCSPPResponder::Start()
{
    LOG_PROC_ENTRY;
    HRESULT Status = E_FAIL;
    m_Stopping = FALSE;
    __try
    {
        Status = CFastCGIRequest::StartListen(URLADDRESS);
        CHECK_HR_STATUS_LEAVE("При запуске FastCGI произошла ошибка",
Status)
        m_CountThread = COcspSettings::GetOcspWorkerThreadsCount();
        m_pWorkerThreads = new HANDLE[m_CountThread];
        memset(m_pWorkerThreads, 0, sizeof(HANDLE) * m_CountThread);
        for (DWORD i = 0; i < m_CountThread; i++)
        {
            m_pWorkerThreads[i] = (HANDLE)_beginthreadex(NULL, 0,
&WorkerThread, this, 0, NULL);
            CHECK_FOR_INTERNAL_ERROR("Не удалось создать поток",
m_pWorkerThreads[i] != INVALID_HANDLE_VALUE);

```

```

    }
    LOG_WRITE("COCSPResponder начал работу");
    Status = S_OK;
}
__finally
{
}
return Status;
}

void
COCSPResponder::Stop()
{
    LOG_PROC_ENTRY;
    m_Stopping = TRUE;
    CFastCGIRequest::StopListen();
    // Если потоков нет, значит останавливать нечего
    if (m_pWorkerThreads)
    {
        DWORD Status = WaitForMultipleObjects(m_CountThread,
m_pWorkerThreads, TRUE, 4000);
        LogBoolStatus("Не удалось корректно завершить потоки (Они
завершатся, когда обработают заявки).", Status == WAIT_OBJECT_0);
        delete[] m_pWorkerThreads;
        m_pWorkerThreads = NULL;
    }
    LOG_WRITE("Работа COCSPResponder завершена");
}

/// <summary>
/// Рабочий поток. Обслуживает запросы (можно создать несколько)
/// </summary>
/// <param name="lpParam">Указатель на объект COCSPResponder.</param>
uintptr_t WINAPI COCSPResponder::WorkerThread(LPVOID lpParam)
{
    LOG_PROC_ENTRY;
    HRESULT Status = E_FAIL;
    COCSPResponder* pThis = (COCSPResponder*)lpParam;
    LPCSTR TypeRequest = NULL;
    CSupportOfCryptX509* CryptX509 = NULL;
    CFastCGIRequest Request;
    CByteBuffer RequestData;
    __try
    {
        CryptX509 = pThis->m_pPkcs11->CreateSigner();
        CHECK_FOR_INTERNAL_ERROR("Не удалось создать рабочий
криптографический объект", CryptX509 != NULL);

        COcspResponseMaker ResponseMaker(
            CryptX509,
            pThis->m_pOcspCertificate,
            pThis->m_pChachedStore,
            pThis->m_pCrlManager);

        LOG_WRITE("Ожидание запроса");
        while (!pThis->m_Stopping) {
            Status = Request.GetNext(TypeRequest, RequestData);

            // Пришла команда остановки
            if (pThis->m_Stopping && (FAILED(Status)))
                break;
            CHECK_HR_STATUS_LEAVE("Не удалось получить следующий
запрос", Status);
            LOG_WRITE("Получен запрос");
            if (strcmp(TypeRequest, "application/ocsp-request") != 0)
            {

```

```

        ERROR_WRITE("Не правильный CONTENT_TYPE");
        continue;
    }
    CByteBuffer ResponseData;
    Status = ResponseMaker.MakeResponse(RequestData,
RequestData.Size, ResponseData);
    LogStatus("При формировании ответа произошла ошибка",
Status);    // Ответ нужно отправить в любом случае
            // ResponseData всегда должен содержать ответ с сообщением
об ошибке. Эта проверка на всякий случай
            if (!ResponseData.Data)

        ResponseMaker.MakeFailedResponse(X509OCSPResponseStatus::INTERNAL_ERROR,
ResponseData);
        Status = Request.SendResponse("application/ocsp-response",
ResponseData);
        CHECK_HR_STATUS_LEAVE("Не удалось отправить ответ", Status);
        LOG_WRITE("Отправлен ответ");
    } /* while */
    Status = S_OK;
}
__finally
{
    LOG_WRITE("Рабочий поток завершен");
    if (FAILED(Status))
    {
        pThis->m_Stopping = TRUE;
    }
}
return Status;
}

```

Листинг A.5 – OcspResponseMaker.h

```

#pragma once
class COcspResponseMaker
{
public:
    /// <summary>
    /// Конструктор класса
    /// </summary>
    /// <param name="cryptX509">Криптографический объект.</param>
    /// <param name="pOcspCerificate">Сертификат OCSP службы.</param>
    /// <param name="pChachedStore">Хранилище сертификатов.</param>
    /// <param name="pCrlManager">Класс обновления списов отзывов.</param>
    COcspResponseMaker(
        __in CSupportOfCryptX509* cryptX509,
        __in CX509Certificate2* pOcspCerificate,
        __in IChachedStore* pChachedStore,
        __in CrlManager* pCrlManager);
    ~COcspResponseMaker();
    /// <summary>
    /// Формирование ответа на OCSP запрос
    /// </summary>
    /// <param name="pRequest">Данные OCSP запроса.</param>
    /// <param name="requestSize">Размер данных OCSP запроса.</param>
    /// <param name="Response">Данные OCSP ответа.</param>
    /// <returns>Статус выполнения операции.</returns>
    HRESULT
    MakeResponse(
        __in_bcount(RequestSize) PBYTE pRequest,
        __in DWORD requestSize,
        __inout CByteBuffer&
Response);

```

```

    /// <summary>
    /// Генерация OSCP ответа, содержащего код ошибки
    /// </summary>
    /// <param name="responseStatus">Ошибка.</param>
    /// <returns>Статус выполнения операции.</returns>
    HRESULT
    MakeFailedResponse(
        __in          X509OCSPResponseStatus  responseStatus,
        __inout       CByteBuffer&
        response);
private:
    /// <summary>
    /// Формирование ответа на простой OSCP запрос (часть общего)
    /// </summary>
    /// <param name="pSingleRequest">Простой OSCP запрос.</param>
    /// <param name="ppSingleResponse">Простой OSCP ответ.</param>
    /// <returns>Статус выполнения операции.</returns>
    HRESULT
    MakeSingleResponse(
        __in          CX509Request*              pSingleRequest,
        __out         CX509SingleResponse**      ppSingleResponse);
    /// <summary>
    /// Устанавливает состояние сертификата исходя из его наличия в Crl
    /// </summary>
    /// <param name="response">Простой OSCP ответ, в который записывается
результат.</param>
    /// <param name="serialNumber">Серийный номер сертификата.</param>
    VOID
    CheckInCrl(
        __inout       CX509SingleResponse&      response,
        __in          CByteBuffer&              serialNumber);
    /// <summary>
    /// Поиск подходящего сертификата и проверка подписи запроса.
    /// </summary>
    /// <returns>Валидность подписи.</returns>
    X509OCSPResponseStatus
    CheckRequestSignature(CX509OCSPRequest& request);
    /// <summary>
    /// Генерация OSCP ответа, содержащего код ошибки
    /// </summary>
    /// <param name="ResponseStatus">Ошибка.</param>
    /// <returns>Статус выполнения операции.</returns>
    HRESULT
    FailedResponse(
        __in          X509OCSPResponseStatus  ResponseStatus);
    /// <summary>
    /// При необходимости копирует из менеджера обновленную crl
    /// </summary>
    VOID
    UpdateCrl();
    CByteBuffer*      m_FailedResponses;
    CSupportOfCryptX509* m_pCryptX509;
    CX509Crl*         m_pBaseCrl;
    CX509Crl*         m_pDeltaCrl;
    CrlManager*       m_pCrlManager;
    IChachedStore*    m_pChachedStore;
    /// <summary>
    /// Время установки основного списка отзыва.
    /// </summary>
    FILETIME          m_BaseCrlInstallTime;
    /// <summary>
    /// Время установки разностного списка отзыва.
    /// </summary>
    FILETIME          m_DeltaCrlInstallTime;

```

```

CX509Certificate2*      m_pOcspCertificate;
FILETIME                m_CurrentTime;
};

```

Листинг А.6 – OcspResponseMaker.cpp

```

#include "pch.h"
#include "OcspResponseMaker.h"
#include "../Asn/Asn1Convert.h"
/// <summary>
/// Конструктор класса
/// </summary>
/// <param name="cryptX509">Криптографический объект.</param>
/// <param name="pOcspCerificate">Сертификат OCSP службы.</param>
/// <param name="pChachedStore">Хранилище сертификатов.</param>
/// <param name="pCrlManager">Класс обновления списов отзывов.</param>
COcspResponseMaker::COcspResponseMaker(
    __in          CSupportOfCryptX509*      cryptX509,
    __in          CX509Certificate2*        pOcspCerificate,
    __in          IChachedStore*            pChachedStore,
    __in          CrlManager*               pCrlManager)
{
    LOG_PROC_ENTRY;
    m_pCryptX509          = cryptX509;          // Принадлежит нам
    m_pCrlManager         = pCrlManager;        // Общие
    m_pChachedStore       = pChachedStore;      // Общие
    m_pOcspCertificate    = new CX509Certificate2(*pOcspCerificate);
    m_BaseCrlInstallTime  = { 0 };
    m_DeltaCrlInstallTime = { 0 };
    m_pBaseCrl            = NULL;
    m_pDeltaCrl           = NULL;
    m_CurrentTime         = { 0 };
    m_FailedResponses     = new CByteBuffer[X509OCSP_RESPONSE_STATUSES_COUNT];
    HRESULT Status = E_FAIL;
    __try
    {
        Status =
FailedResponse(X509OCSPResponseStatus::MALFORMED_REQUEST);
        CHECK_HR_STATUS_LEAVE1(Status);
        Status = FailedResponse(X509OCSPResponseStatus::INTERNAL_ERROR);
        CHECK_HR_STATUS_LEAVE1(Status);
        Status = FailedResponse(X509OCSPResponseStatus::TRY_LATER);
        CHECK_HR_STATUS_LEAVE1(Status);
        Status = FailedResponse(X509OCSPResponseStatus::SIG_REQUIRED);
        CHECK_HR_STATUS_LEAVE1(Status);
        Status = FailedResponse(X509OCSPResponseStatus::UNAUTHORIZED);
        CHECK_HR_STATUS_LEAVE1(Status);
    }
    __finally
    {
        if (FAILED(Status))
        {
            delete[] m_FailedResponses;
            m_FailedResponses = NULL;
        }
    }
}

COcspResponseMaker::~COcspResponseMaker()
{
    LOG_PROC_ENTRY;
    delete[] m_FailedResponses;
    delete m_pCryptX509;
    delete m_pBaseCrl;
    delete m_pDeltaCrl;
}

```

```

        delete m_pOcspCertificate;
    }
    /// <summary>
    /// Формирование ответа на OCSP запрос
    /// </summary>
    /// <param name="pRequest">Данные OCSP запроса.</param>
    /// <param name="requestSize">Размер данных OCSP запроса.</param>
    /// <param name="Response">Данные OCSP ответа.</param>
    /// <returns>Статус выполнения операции.</returns>
    HRESULT
    COcspResponseMaker::MakeResponse(
        __in_bcount(RequestSize) PBYTE pRequest,
        __in DWORD requestSize,
        __inout CByteBuffer& Response)
    {
        LOG_PROC_ENTRY;
        X509OCSPResponseStatus ResponseStatus =
X509OCSPResponseStatus::MALFORMED_REQUEST;
        HRESULT Status = S_OK;
        GetSystemTimeAsFileTime(&m_CurrentTime);
        if (!pRequest)
            return E_INVALIDARG;
        CX509OCSPRequest Request(pRequest, requestSize);
        CX509OCSPResponseBuilder ResponseBuilder;
        __try
        {
            UpdateCrl();
            Status = Request.GetInitStatus();
            CHECK_HR_STATUS_LEAVE("При анализе OCSP запроса произошла ошибка",
Status);
            ResponseStatus = CheckRequestSignature(Request);
            if (ResponseStatus != X509OCSPResponseStatus::SUCCESSFUL)
                __leave;
            for (int i = 0; i < Request.GetCountRequest(); ++i)
            {
                CX509Request* pSingleRequest = NULL;
                CX509SingleResponse* pSingleResponse = NULL;
                __try
                {
                    Status = Request.GetRequest(i, &pSingleRequest);
                    CHECK_HR_STATUS_LEAVE("GetRequest", Status);
                    Status = MakeSingleResponse(pSingleRequest,
&pSingleResponse);
                    CHECK_HR_STATUS_LEAVE("GetRequest", Status);
                    Status =
ResponseBuilder.AddX509Response(pSingleResponse);
                    CHECK_HR_STATUS_LEAVE("GetRequest", Status);
                }
                __finally
                {
                    delete pSingleRequest;
                }
                if (FAILED(Status))
                    __leave;
            }
            // Добавляем расширение Nonce, если он есть.
            CX509Extension* pNonceExtention = NULL;
            Status = Request.GetExtension((PWCHAR)OID_X509_PKIX_OCSP_NONCE,
wcslen(OID_X509_PKIX_OCSP_NONCE), &pNonceExtention);
            if (Status == S_OK)
            {
                ResponseBuilder.SetX509Extension(pNonceExtention);
            }
            ResponseBuilder.SetProducedAtDateTime(m_CurrentTime);

```

```

        ResponseBuilder.SetName(m_pOcspCertificate->GetSubjectName());

        ResponseBuilder.AddX509Certificate((CX509Certificate2*)m_pOcspCertificate->Copy());
        ResponseBuilder.SetStatus(X509OCSPResponseStatus::SUCCESSFUL);

        Status = m_pCryptX509->Sign(ResponseBuilder, Response);
        CHECK_HR_STATUS_LEAVE("ResponseBuilder.Create", Status);
        Status = S_OK;
    }
    __finally
    {
        if (FAILED(Status) || (ResponseStatus != X509OCSPResponseStatus::SUCCESSFUL))
        {
            if (ResponseStatus == X509OCSPResponseStatus::SUCCESSFUL)
            {
                ResponseStatus = X509OCSPResponseStatus::INTERNAL_ERROR;
            }
            // Если при генерации ошибочных ответов не было ошибок
            if (m_FailedResponses)
            {
                Response = m_FailedResponses[(DWORD)ResponseStatus];
            }
        }
        return Status;
    }
}
/// <summary>
/// Генерация OCSP ответа, содержащего код ошибки
/// </summary>
/// <param name="responseStatus">Ошибка.</param>
/// <returns>Статус выполнения операции.</returns>
HRESULT
COcspResponseMaker::MakeFailedResponse(
    __in X509OCSPResponseStatus responseStatus,
    __inout CByteBuffer& response)
{
    if (!m_FailedResponses)
        return E_FAIL;
    response = m_FailedResponses[(DWORD)responseStatus];
    return S_OK;
}
/// <summary>
/// Формирование ответа на простой OCSP запрос (часть общего)
/// </summary>
/// <param name="pSingleRequest">Простой OCSP запрос.</param>
/// <param name="ppSingleResponse">Простой OCSP ответ.</param>
/// <returns>Статус выполнения операции.</returns>
HRESULT
COcspResponseMaker::MakeSingleResponse(
    __in CX509Request* pSingleRequest,
    __out CX509SingleResponse** ppSingleResponse)
{
    LOG_PROC_ENTRY;
    HRESULT Status = S_OK;

    if (!ppSingleResponse || !pSingleRequest)
        return E_INVALIDARG;
    CX509SingleResponse* pSingleResponse = new CX509SingleResponse();
    CX509CertID CertID = pSingleRequest->GetCertID();

```

```

        CByteBuffer                SerialNumber        =
CertId.GetSerialNumber();

        __try
        {
            pSingleResponse->SetCertID(CertId);
            CX509Certificate2* pCertificate = m_pChachedStore-
>GetCertificate(&CertId.GetSerialNumber(), &CertId.GetIssuerNameHash());
            if (pCertificate != NULL &&
                (FILETIME_TO_INT64(m_CurrentTime) >=
FILETIME_TO_INT64(pCertificate->GetNotBeforeDate())) &&
                (FILETIME_TO_INT64(m_CurrentTime) <=
FILETIME_TO_INT64(pCertificate->GetNotBeforeDate())))
            {
                CheckInCrl(*pSingleResponse, SerialNumber);
                // Используем время из СОС.
                FILETIME BaseDate;
                FILETIME DeltaDate;
                m_pBaseCrl->GetThisUpdateDate(&BaseDate);
                pSingleResponse->SetThisUpdateDate(&BaseDate);
                if (m_pDeltaCrl)
                {
                    m_pDeltaCrl->GetThisUpdateDate(&DeltaDate);
                    if (*(INT64*)&DeltaDate > *(INT64*)&BaseDate)
                    {
                        pSingleResponse->SetThisUpdateDate(&DeltaDate);
                    }
                }
            }
            else
            {
                pSingleResponse-
>SetStatus(CX509SingleResponse::CERT_STATUS_UNKNOWN);
                delete pCertificate;
                // Устанавливаем время действия элемента ответа.
                if (pSingleResponse->GetStatus() !=
CX509SingleResponse::CERT_STATUS_GOOD)
                {
                    // Сертификат отозван или его статус неизвестен. В этом
случае время NextUpdate не может быть установлено.
                    // Так же, если сертификат приостановлен, NextUpdate должен
быть выставлен.
                    pSingleResponse->SetNextUpdateDate(NULL);
                }
                else
                {
                    pSingleResponse->SetNextUpdateDate(m_pBaseCrl-
>GetNextUpdateDate());
                    if (m_pDeltaCrl && m_pDeltaCrl->GetNextUpdateDate())
                    {
                        pSingleResponse->SetNextUpdateDate(m_pDeltaCrl-
>GetNextUpdateDate());
                    }
                }
                *ppSingleResponse = pSingleResponse;
            }
        }__finally
        {
        }
        return Status;
    }

    /// <summary>
    /// Устанавливает состояние сертификата исходя из его наличия в Crl

```

```

/// </summary>
/// <param name="response">Простой OCSP ответ, в который записывается
результат.</param>
/// <param name="serialNumber">Серийный номер сертификата.</param>
VOID
COcspResponseMaker::CheckInCrl(
    __inout CX509SingleResponse& response,
    __in CByteBuffer& serialNumber)
{
    LOG_PROC_ENTRY;
    HRESULT Status = S_OK;
    __try
    {
        CX509RevokedCertificate* pCertificate = NULL;
        vector<BYTE>
        SerialNumber(serialNumber.Data, serialNumber.Data + serialNumber.Size);
        Status = m_pBaseCrl->GetRevokedCertificate(SerialNumber,
&pCertificate);
        CHECK_HR_STATUS_LEAVE("GetRevokedCertificate", Status);
        if (pCertificate)
        {
            response.SetStatus(CX509SingleResponse::CERT_STATUS_REVOKED);
            X509CRLReason *Reason = pCertificate->GetReasonCode();
            response.SetReason(Reason);
            pCertificate->SetRevocationDate(m_CurrentTime);
        }
        else
        {
            response.SetStatus(CX509SingleResponse::CERT_STATUS_GOOD);
        }
        if (!m_pDeltaCrl)
            return;
        Status = m_pDeltaCrl->GetRevokedCertificate(SerialNumber,
&pCertificate);
        CHECK_HR_STATUS_LEAVE("GetRevokedCertificate", Status);
        if (pCertificate)
        {
            response.SetStatus(CX509SingleResponse::CERT_STATUS_REVOKED);
            X509CRLReason *Reason = pCertificate->GetReasonCode();
            response.SetReason(Reason);
            pCertificate->SetRevocationDate(m_CurrentTime);
        }
    }
    __finally
    {
        if (FAILED(Status))
        {
            response.SetStatus(CX509SingleResponse::CERT_STATUS_UNKNOWN);
        }
    }
}
/// <summary>
/// Поиск подходящего сертификата и проверка подписи запроса.
/// </summary>
/// <returns>Валидность подписи.</returns>
X509OCSPResponseStatus
COcspResponseMaker::CheckRequestSignature(CX509OCSPRequest& request)
{
    LOG_PROC_ENTRY;
    HRESULT Status = S_OK;

```

```

    if (!request.Signed())
    {
        if (COcspSettings::IsSigRequestNeeded())
        {
            ERROR_WRITE("Запрос не был подписан, хотя политика
безопасности требует обязательного наличия подписи запроса.");
            return X509OCSPResponseStatus::SIG_REQUIRED;
        }
    }
    else
    {
        /*for (DWORD i = 0; i < request.GetCountCertificate(); i++)
        {
            CX509Certificate2* pCert = NULL;
            Status = request.GetCertificate(i, &pCert);
            if (FAILED(Status))
                continue;
            if (request.Verify(m_pCryptX509, pCert))
            {
                CX509Certificate2* pCertsForThisUser =
m_pChachedStore->GetCertificate(&pCert->GetSerialNumber(), NULL);
                delete pCert;
                if (!pCertsForThisUser)
                {
                    LOG_WRITE("Запрос, подписан сертификатом,
издатель которого не принадлежит данному УЦ и не может быть проверен!");
                    return
X509OCSPResponseStatus::MALFORMED_REQUEST;
                }
                return X509OCSPResponseStatus::SUCCESSFUL;
            }
        }
        */
        // Если так и не удалось проверить подпись, используя переданный
сертификат,
        // пытаемся это сделать с сертификатом из хранилища.

        // TODO проверка подписи сертификатами из хранилища
    }
    return X509OCSPResponseStatus::SUCCESSFUL;
}
/// <summary>
/// Генерация OCSP ответа, содержащего код ошибки
/// </summary>
/// <param name="ResponseStatus">Ошибка.</param>
/// <returns>Статус выполнения операции.</returns>
HRESULT
COcspResponseMaker::FailedResponse(
    __in X509OCSPResponseStatus ResponseStatus)
{
    LOG_PROC_ENTRY;
    HRESULT Status = E_FAIL;
    __try
    {
        CX509OCSPResponseBuilder ResponseBuilder;
        ResponseBuilder.SetStatus(ResponseStatus);
        Status = ResponseBuilder.Create(NULL,
m_FailedResponses[(DWORD)ResponseStatus]);
        CHECK_HR_STATUS_LEAVE("Ошибка построения ответа с ошибкой",
Status);
    }
    __finally
    {
    }
    return Status;
}

```

```

}
/// <summary>
/// При необходимости копирует из менеджера обновленный СОС
/// </summary>
VOID
COcspResponseMaker::UpdateCrl()
{
    LOG_PROC_ENTRY;

    if (memcmp(&m_BaseCrlInstallTime, &m_pCrlManager-
>GetBaseCrlInstallTime(), sizeof(m_BaseCrlInstallTime) != 0))
    {
        m_pBaseCrl = m_pCrlManager->GetBaseCrl();
        m_BaseCrlInstallTime = m_pCrlManager->GetBaseCrlInstallTime();
    }
    if (memcmp(&m_BaseCrlInstallTime, &m_pCrlManager-
>GetDeltaCrlInstallTime(), sizeof(m_BaseCrlInstallTime) != 0))
    {
        m_pDeltaCrl = m_pCrlManager->GetDeltaCrl();
        m_DeltaCrlInstallTime = m_pCrlManager->GetDeltaCrlInstallTime();
    }
}

```

Листинг A.7 – CrlManager.h

```

#pragma once
/// <summary>
/// Менеджер список отзыва.
/// </summary>
class CrlManager
{
public:
    /// <summary>
    /// Инициализация менеджера списков отзыва.
    /// </summary>
    CrlManager(ICachedStore* cachedStore);
    /// <summary>
    /// Уничтожение объекта.
    /// </summary>
    ~CrlManager();
    /// <summary>
    /// Начать слежение за актуальностью САС.
    /// </summary>
    HRESULT StartObservation();
    /// <summary>
    /// Очистка ресурсов.
    /// </summary>
    VOID StopObservation();
    /// <summary>
    /// Получение копии нового разностного списка отозванных сертификатов.
    /// </summary>
    /// <returns>Список отозванных сертификатов.</returns>
    CX509Crl* GetDeltaCrl();
    /// <summary>
    /// Получение копии нового базового списка отозванных сертификатов.
    /// </summary>
    /// <returns>Список отозванных сертификатов.</returns>
    CX509Crl* GetBaseCrl();
    /// <summary>
    /// Получение времени установки разностного списка отозванных
    сертификатов.
    /// </summary>
    /// <returns>Время установки.</returns>
    FILETIME GetDeltaCrlInstallTime();

```

```

    /// <summary>
    /// Получение времени установки базового списка отозванных сертификатов.
    /// </summary>
    /// <returns>Время установки.</returns>
    FILETIME GetBaseCrlInstallTime();
    /// <summary>
    /// Проверка и установка нового базового списка отозванных сертификатов.
    /// </summary>
    /// <param name="certList">Устанавливаемый базовый список отозванных
сертификатов.</param>
    BOOL SetBaseCrl(CX509Crl* certList);
    /// <summary>
    /// Проверка и установка нового разностного списка отозванных
сертификатов.
    /// </summary>
    /// <param name="certList">Устанавливаемый разностный список отозванных
сертификатов.</param>
    BOOL SetDeltaCrl(CX509Crl* certList);
    /// <summary>
    /// Выполнить обновление CAC.
    /// </summary>
    BOOL UpdateCrls();
    /// <summary>
    /// Очистка установленных списков отзыва.
    /// </summary>
    VOID Clear();
private:
    /// <summary>
    /// Поток, который следит за актуальностью CAC.
    /// </summary>
    static uintptr_t WINAPI CrlUpdaterThreadHandler(LPVOID lpParam);
    /// <summary>
    /// Проверка, опубликован ли новый CAC.
    /// </summary>
    /// <returns>Если true, есть новый CAC. Иначе - false.</returns>
    BOOL IsNewCrlPublicated();
    /// <summary>
    /// Обработчик события установки нового CAC с записью в реестр для
оповещения криптоменеджера.
    /// </summary>
    /// <param name="UpdatedCrls">Обновленный CAC.</param>
    /// <param name="UpdateTime">Время обновления CAC.</param>
    VOID OnCrlInstalledSetInfoHandler(CX509Crl* UpdatedCrls, FILETIME
UpdateTime);
    /// <summary>
    /// Проверка обновлений остановлена
    /// </summary>
    HANDLE m_Stop;
    /// <summary>
    /// Объект синхронизации читатели-писатель
    /// </summary>
    shared_mutex m_hUsingCrl;
    /// <summary>
    /// Поток, который занимается обновлением CAC.
    /// </summary>
    HANDLE m_hCrlUpdaterThread;
    /// <summary>
    /// Время установки основного списка отзыва.
    /// </summary>
    FILETIME m_BaseCrlInstallTime;
    /// <summary>
    /// Время установки разностного списка отзыва.
    /// </summary>
    FILETIME m_DeltaCrlInstallTime;

```

```

    /// <summary>
    /// Экземпляр текущего основного списка отзыва.
    /// </summary>
    CX509Crl* m_pCurrentBaseCrl;
    /// <summary>
    /// Экземпляр текущего разностного списка отзыва.
    /// </summary>
    CX509Crl* m_pCurrentDeltaCrl;
    /// <summary>
    /// Хранилище для получение списков отзывов.
    /// </summary>
    IChachedStore* m_ChachedStore;
};

```

Листинг A.8 – CrlManager.cpp

```

#include "pch.h"
/// <summary>
/// Инициализация менеджера списков отзыва.
/// </summary>
CrlManager::CrlManager(IChachedStore* chachedStore)
{
    LOG_PROC_ENTRY;
    m_BaseCrlInstallTime = { 0 };
    m_DeltaCrlInstallTime = { 0 };
    m_pCurrentBaseCrl = NULL;
    m_pCurrentDeltaCrl = NULL;
    m_hCrlUpdaterThread = NULL;
    m_Stop = NULL;
    m_ChachedStore = chachedStore;
}
/// <summary>
/// Уничтожение объекта.
/// </summary>
CrlManager::~CrlManager()
{
    StopObservation();
    Clear();
}
/// <summary>
/// Начать слежение за актуальностью САС.
/// </summary>
HRESULT CrlManager::StartObservation()
{
    LOG_PROC_ENTRY;
    if (!m_ChachedStore)
    {
        return E_INVALIDARG;
    }
    m_Stop = CreateEvent(NULL, FALSE, FALSE, NULL);
    m_hCrlUpdaterThread = (HANDLE)_beginthreadex(NULL, 0,
&CrlUpdaterThreadHandler, this, 0, NULL);
    if (!m_hCrlUpdaterThread)
    {
        return HRESULT_FROM_WIN32(GetLastError());
    }
    return S_OK;
}
/// <summary>
/// Очистка ресурсов.
/// </summary>
VOID CrlManager::StopObservation()
{
    LOG_PROC_ENTRY;

```

```

SetEvent(m_Stop);
if (m_hCrlUpdaterThread)
{
    WaitForSingleObject(m_hCrlUpdaterThread, 1000);
    m_hCrlUpdaterThread = NULL;
}
CloseHandle(m_Stop);
m_Stop = INVALID_HANDLE_VALUE;
}
/// <summary>
/// Выполнить обновление СОС.
/// </summary>
BOOL CrlManager::UpdateCrls()
{
    LOG_PROC_ENTRY;
    BOOL Updated = FALSE;
    __try
    {
        LOG_WRITE("Проверяем СОС");
        // Получаем актуальный основной СОС.
        CX509Crl* ActualBaseCrl = m_ChachedStore->GetActualCrl();
        CHECK_FOR_INTERNAL_ERROR("Не удалось получить актуальный основной
список отозванных сертификатов", ActualBaseCrl != NULL)
        // Пробуем сразу же получить дельту.
        CX509Crl* ActualDeltaCrl = NULL;
        if (ActualBaseCrl->IsBase())
        {
            ActualDeltaCrl = m_ChachedStore->GetDeltaCrl();
            CHECK_FOR_INTERNAL_ERROR("Не удалось получить актуальный
разностный список отозванных сертификатов", ActualDeltaCrl != NULL);
            Updated = SetBaseCrl(ActualBaseCrl);
            Updated = SetDeltaCrl(ActualDeltaCrl);
        }
        else
        {
            Updated = SetBaseCrl(ActualBaseCrl);
        }
    }
    __finally
    {
    }
    return Updated;
}
/// <summary>
/// Очистка установленных списков отзыва.
/// </summary>
VOID CrlManager::Clear()
{
    LOG_PROC_ENTRY;
    unique_lock<shared_mutex> locker(m_hUsingCrl);
    delete m_pCurrentBaseCrl;
    delete m_pCurrentDeltaCrl;
    m_pCurrentBaseCrl = NULL;
    m_pCurrentDeltaCrl = NULL;
    m_BaseCrlInstallTime = { 0 };
    m_DeltaCrlInstallTime = { 0 };
    OnCrlInstalledSetInfoHandler(m_pCurrentBaseCrl, m_BaseCrlInstallTime);
    OnCrlInstalledSetInfoHandler(m_pCurrentDeltaCrl, m_DeltaCrlInstallTime);
}
/// <summary>
/// Поток, который следит за актуальностью СОС.
/// </summary>
uintptr_t WINAPI CrlManager::CrlUpdaterThreadHandler(LPVOID lpParam)
{

```

```

LOG_PROC_ENTRY;
CrlManager*      pThis      = (CrlManager*)lpParam;
time_t           LastUpdate = 0;
__try
{
    int UpdatePeriod =
COCspSettings::GetUpdateCrlPeriod();
    LOG_WRITE("Поток обновления актуальных СОС запущен.");
    while (TRUE)
    {
        DWORD Status = WaitForSingleObject(pThis->m_Stop, 10000);
        //RegNotifyChangeKeyValue()
        if (Status != WAIT_TIMEOUT) // Если не дождались
конца таймаута, значит либо ошибка, либо нужно останавливаться
            return HRESULT_FROM_WIN32(GetLastError());
        if (UpdatePeriod != COCspSettings::GetUpdateCrlPeriod())
        {
            UpdatePeriod = COCspSettings::GetUpdateCrlPeriod();
            LOG_WRITE("Интервал обновления СОС изменен");
        }

        // Проверяем, прошло ли достаточно времени. Или проверяем,
появился ли признак нового СОС в реестре.
        if (!pThis->IsNewCrlPublicated())
        {
            // Если не прошло достаточно времени с последней
попытки обновления, выходим.
            time_t TimeNow = time(NULL);

            if ((TimeNow - LastUpdate) < (60 * UpdatePeriod))
                continue;
        }
        // Время последней попытки обновления, независимо от успеха
time(&LastUpdate);
        pThis->UpdateCrls();
    }
}
__finally
{
    LOG_WRITE("Поток обновления актуальных СОС остановлен.");
}
return S_OK;
}
/// <summary>
/// Проверка, опубликован ли новый СОС.
/// </summary>
/// <returns>Если true, есть новый СОС. Иначе - false.</returns>
BOOL CrlManager::IsNewCrlPublicated()
{
    LOG_PROC_ENTRY;
    HRESULT Status
    = E_FAIL;
    CX509CertsAuthorityVersionExtension* pCaVersion = NULL;
    __try
    {
        // Для работы нам необходим базовый САС.
        if (m_pCurrentBaseCrl == NULL)
            return FALSE;
        Status = m_pCurrentBaseCrl-
>GetExtension((PWCHAR)OID_X509_CERTS_AUTHORITY_VERSION_EXT,
wcslen(OID_X509_CERTS_AUTHORITY_VERSION_EXT), (CX509Extension * *)&
pCaVersion);
        if (FAILED(Status))
            return FALSE;
    }
}

```

```

        // Получаем номер последнего установленного САС.
        int LastNumber = COcspSettings::GetUpdatedCrlInfo(pCaVersion-
>Format());
        if (-1 == LastNumber)
            return FALSE;
        // Проверяем, если есть дельта, берем номер оттуда. Иначе смотрим
базовый САС.
        if (m_pCurrentDeltaCrl != NULL)
        {
            return (int)m_pCurrentDeltaCrl->GetNumber() < LastNumber;
        }
        else
        {
            return (int)m_pCurrentBaseCrl->GetNumber() < LastNumber;
        }
    }
    __finally
    {
        delete pCaVersion;
    }
    return FALSE;
}
/// <summary>
/// Обработчик события установки нового САС с записью в реестр для оповещения
криптоменеджера.
/// </summary>
/// <param name="UpdatedCrls">Обновленный САС.</param>
/// <param name="UpdateTime">Время обновления САС.</param>
VOID
CrlManager::OnCrlInstalledSetInfoHandler(CX509Crl* UpdatedCrls, FILETIME
    UpdateTime)
{
    if (UpdatedCrls)
    {
        CX509CertsAuthorityVersionExtension* pCaVersion = NULL;
        HRESULT Status = UpdatedCrls-
>GetExtension((PWCHAR)OID_X509_CERTS_AUTHORITY_VERSION_EXT,
wcslen(OID_X509_CERTS_AUTHORITY_VERSION_EXT), (CX509Extension**)&pCaVersion);
        if (FAILED(Status))
            return;

        COcspSettings::SetOcspUpdateCrlInfo(
            pCaVersion->Format(),
            (DWORD)UpdatedCrls->GetNumber(),
            UpdateTime);
        delete pCaVersion;
    }
}
/// <summary>
/// Получение нового разностного списка отозванных сертификатов.
/// </summary>
/// <returns>Список отозванных сертификатов.</returns>
CX509Crl*
CrlManager::GetDeltaCrl()
{
    LOG_PROC_ENTRY;
    if (!m_pCurrentDeltaCrl)
        return NULL;

    __try
    {
        shared_lock<shared_mutex> locker(m_hUsingCrl);
        CX509Crl* pDeltaCrl = m_pCurrentDeltaCrl ?
(CX509Crl*)m_pCurrentDeltaCrl->Copy() : NULL;
        return pDeltaCrl;
    }
}

```

```

    }
    __finally
    {
    }
    return NULL;
}
/// <summary>
/// Получение нового базового списка отозванных сертификатов.
/// </summary>
/// <returns>Список отозванных сертификатов.</returns>
CX509Crl*
CrlManager::GetBaseCrl()
{
    LOG_PROC_ENTRY;
    if (!m_pCurrentBaseCrl)
        return NULL;

    __try
    {
        shared_lock<shared_mutex> locker(m_hUsingCrl);
        CX509Crl* pBaseCrl = (CX509Crl*)m_pCurrentBaseCrl->Copy();
        return pBaseCrl;
    }
    __finally
    {
    }
    return NULL;
}
/// <summary>
/// Получение времени установки разностного списка отозванных сертификатов.
/// </summary>
/// <returns>Время установки.</returns>
FILETIME CrlManager::GetDeltaCrlInstallTime()
{
    LOG_PROC_ENTRY;
    return m_DeltaCrlInstallTime;
}
/// <summary>
/// Получение времени установки базового списка отозванных сертификатов.
/// </summary>
/// <returns>Время установки.</returns>
FILETIME CrlManager::GetBaseCrlInstallTime()
{
    LOG_PROC_ENTRY;
    return m_BaseCrlInstallTime;
}
/// <summary>
/// Проверка и установка нового базового списка отозванных сертификатов.
/// </summary>
/// <param name="certList">Устанавливаемый базовый список отозванных
сертификатов.</param>
BOOL CrlManager::SetBaseCrl(CX509Crl* certList)
{
    LOG_PROC_ENTRY;
    HRESULT Status = E_FAIL;
    if (!certList)
        return FALSE;

    __try
    {
        unique_lock<shared_mutex> locker(m_hUsingCrl);
        if (m_pCurrentBaseCrl != NULL && certList->GetNumber() <=
m_pCurrentBaseCrl->GetNumber())
            return FALSE;
        // Если основной СОС новее, чем предыдущий, обновляем его.
    }

```

```

LOG_WRITE("Получен новый базовый список отозванных сертификатов");
delete m_pCurrentBaseCrl;
m_pCurrentBaseCrl = certList;

SYSTEMTIME SysTime = { 0 };
GetSystemTime(&SysTime);
SystemTimeToFileTime(&SysTime, &m_DeltaCrlInstallTime);
delete m_pCurrentDeltaCrl;
m_pCurrentDeltaCrl = NULL;
m_DeltaCrlInstallTime = m_BaseCrlInstallTime;
OnCrlInstalledSetInfoHandler(m_pCurrentBaseCrl,
m_BaseCrlInstallTime);

LOG_WRITE("Обновили базовый СОС");
Status = S_OK;
}
__finally
{
    if (FAILED(Status))
    {
        delete certList;
    }
}
return Status == S_OK;
}
/// <summary>
/// Проверка и установка нового разностного списка отозванных сертификатов.
/// </summary>
/// <param name="certList">Устанавливаемый разностный список отозванных
сертификатов.</param>
BOOL CrlManager::SetDeltaCrl(CX509Crl* certList)
{
    LOG_PROC_ENTRY;
    if (!certList)
        return FALSE;
    HRESULT Status = E_FAIL;
    __try
    {
        unique_lock<shared_mutex> locker(m_hUsingCrl);
        // Получаем версию и сравниваем. Устанавливаемый список не должен
        быть "младше" имеющегося.
        if (m_pCurrentDeltaCrl && certList->GetNumber() <=
m_pCurrentDeltaCrl->GetNumber())
        {
            Status = E_FAIL;
            CHECK_FOR_INTERNAL_ERROR("Нельзя установить список, номер
которого меньше, чем текущий.", FALSE);
        }
        CX509DeltaCRLExtension* MinBaseCrl = NULL;
        Status = certList-
>GetExtension((LPWSTR)OID_X509_CRL_DELTA_INDICATOR_EXTENSION,
wcslen(OID_X509_CRL_DELTA_INDICATOR_EXTENSION), (CX509Extension
**)&MinBaseCrl);
        CHECK_HR_STATUS_LEAVE("У разностного списка отзвов нет расширения
дельта индикатора", Status);
        if (m_pCurrentDeltaCrl && (MinBaseCrl->GetBaseCrlNumber() !=
m_pCurrentDeltaCrl->GetNumber()))
        {
            Status = E_FAIL;
            CHECK_FOR_INTERNAL_ERROR("Нельзя установить разностный
список, так как текущий базовый не подходит для него.", FALSE);
        }
        // В этом случае необходимо установить новый список отзыва.
        delete m_pCurrentDeltaCrl;
    }
    __except (EXCEPTION_EXECUTE_HANDLER)
    {
        Status = E_FAIL;
    }
}

```

```

        m_pCurrentDeltaCrl          = certList;

        SYSTEMTIME SysTime          = { 0 };
        GetSystemTime(&SysTime);
        SystemTimeToFileTime(&SysTime, &m_DeltaCrlInstallTime);
        OnCrlInstalledSetInfoHandler(m_pCurrentDeltaCrl,
m_DeltaCrlInstallTime);
        LOG_WRITE("Обновили разностный СОС");
        Status = S_OK;
    }
    __finally
    {
        if (FAILED(Status))
        {
            delete certList;
        }
    }
    return Status == S_OK;
}

```

Приложение Б
Конфигурация сервера nginx
(рекомендуемое)

Листинг Б.1 – Текст конфигурации

```

daemon off;
worker_processes 2;
error_log logs/error.log;

events {
    worker_connections 10000;
}

http {
    server {
        listen 80;
        server_name 0.0.0.0;

        location /ocsp {
            fastcgi_pass 127.0.0.1:9000;

            fastcgi_param SCRIPT_FILENAME
$document_root$fastcgi_script_name;
            fastcgi_param QUERY_STRING $query_string;
            fastcgi_param REQUEST_METHOD $request_method;
            fastcgi_param CONTENT_TYPE $content_type;
            fastcgi_param CONTENT_LENGTH $content_length;

            fastcgi_param SCRIPT_NAME
$fastcgi_script_name;
            fastcgi_param REQUEST_URI $request_uri;
            fastcgi_param DOCUMENT_URI $document_uri;
            fastcgi_param DOCUMENT_ROOT $document_root;
            fastcgi_param SERVER_PROTOCOL $server_protocol;
            fastcgi_param REQUEST_SCHEME $scheme;
            fastcgi_param HTTPS $https;
            if_not_empty;

            fastcgi_param GATEWAY_INTERFACE CGI/1.1;
            fastcgi_param SERVER_SOFTWARE
nginx/$nginx_version;

            fastcgi_param REMOTE_ADDR $remote_addr;
            fastcgi_param REMOTE_PORT $remote_port;
            fastcgi_param SERVER_ADDR $server_addr;
            fastcgi_param SERVER_PORT $server_port;
            fastcgi_param SERVER_NAME $server_name;
        }
    }
}

```

Приложение В
Результаты тестирования
(рекомендуемое)


```

8152:OcspService:      [CPkcs11::IsInitialized]
8152:OcspService:      {CPkcs11::GetSlotList
8152:OcspService:      [CPkcs11::IsInitialized]
8152:OcspService:      }CPkcs11::GetSlotList
8152:OcspService:      }CPkcs11::GetSlotInfo
8152:OcspService:      [CPkcs11Client::MatchSlot]
8152:OcspService:      [CPkcs11Client::MatchSlot]
8152:OcspService:      [CPkcs11Client::MatchSlot]
8152:OcspService:      }CPkcs11Client::GetMatchedTokensInfo
8152:OcspService:      }COcspPkcs11Client::GetTokensInfo
8152:OcspService: }COcspMonitor::GetSlotList
5020:OcspService: [COcspMonitor::GetCurrentState]
23076:OcspService: [COcspMonitor::GetCurrentState]
5140:OcspService: [COcspMonitor::Beacon]
6288:OcspService: [COcspMonitor::GetCurrentState]
22844:OcspService: [COcspMonitor::Beacon]
22776:OcspService: [COcspMonitor::GetCurrentState]
23064:OcspService: [COcspMonitor::Beacon]
22508:OcspService: {COcspMonitor::SelectSlot
22508:OcspService:   {CPkcs11Client::SelectSlot
22508:OcspService:   [CPkcs11::IsInitialized]
22508:OcspService:   [CPkcs11Slot::CreateSession]
22508:OcspService:   }CPkcs11Client::SelectSlot
22508:OcspService:   [COcspMonitor::ChangeState]
22508:OcspService:   Слот выбран
22508:OcspService: }COcspMonitor::SelectSlot
23268:OcspService: [COcspMonitor::GetCurrentState]
8508:OcspService: [COcspMonitor::Beacon]
22700:OcspService: {COcspMonitor::LoginOnSlot
22700:OcspService:   {CPkcs11Client::Login
22700:OcspService:   {CPkcs11Slot::Login
22700:OcspService:   [CPkcs11::IsInitialized]
22700:OcspService:   [CPkcs11Slot::Logout]
22700:OcspService:   [CPkcs11Slot::CreateSession]
22700:OcspService:   }CPkcs11Slot::Login
22700:OcspService:   }CPkcs11Client::Login
22700:OcspService:   Авторизация проведена
22700:OcspService:   [COcspMonitor::ChangeState]
22700:OcspService: }COcspMonitor::LoginOnSlot
22616:OcspService: [COcspMonitor::GetCurrentState]
20536:OcspService: {COcspMonitor::SetRootCertificate
20536:OcspService:   {CChachedStore::GetCertificate
20536:OcspService:   [CLocalStore::TryGetObject]
20536:OcspService:   {CCertsStorage::GetCertificate
20536:OcspService:   {CCertsStorage::GetElement
...
}CPkcs11Client::SetCertificate
20536:OcspService:
[CCachedStore::SetCurrentRootCertificateSerialNumber]
20536:OcspService:   Установлен корневой сертификат
20536:OcspService:   [COcspMonitor::ChangeState]
20536:OcspService: }COcspMonitor::SetRootCertificate
14416:OcspService: [COcspMonitor::Beacon]
21536:OcspService: [COcspMonitor::GetCurrentState]

```

```

22560:OcspService: {COcspMonitor::SetOcspCertificate
...
22560:OcspService:    }CPkcs11Client::SetCertificate
22560:OcspService:
[CChachedStore::SetCurrentOcspCertificateSerialNumber]
22560:OcspService:    Установлен сертификат OCSP службы
22560:OcspService:    [CRegistryKey::GetSubKey]
22560:OcspService:    [CRegistryKey::GetDWORDValue]
22560:OcspService:    [COcspMonitor::ChangeState]
22560:OcspService: }COcspMonitor::SetOcspCertificate
22904:OcspService: [COcspMonitor::Beacon]
13980:OcspService: [COcspMonitor::GetCurrentState]
20100:OcspService: [COcspMonitor::Beacon]
20284:OcspService: {COcspMonitor::FindSignKeyFromOneToken
20284:OcspService:    {CPkcs11Client::FindSignKeyFromOneToken
20284:OcspService:        Поиск ключа подписи
20284:OcspService:        {CPkcs11Client::GetKeyHandle
20284:OcspService:            [CPkcs11Session::FindObjects]
20284:OcspService:        }CPkcs11Client::GetKeyHandle
20284:OcspService:        Описатель ключа подписи найден
20284:OcspService:    }CPkcs11Client::FindSignKeyFromOneToken
20284:OcspService:    Найден ключ подписи службы
20284:OcspService:    [COcspMonitor::ChangeState]
20284:OcspService: }COcspMonitor::FindSignKeyFromOneToken
20180:OcspService: [COcspMonitor::GetCurrentState]
10964:OcspService: {COcspMonitor::SetBaseCrl
10964:OcspService:    {CChachedStore::GetActualCrl
10964:OcspService:        {CChachedStore::GetCurrentRootCertificate
10964:OcspService:            {CChachedStore::GetCertificate
10964:OcspService:                {CLocalStore::TryGetObject
10964:OcspService:                    [CX509Object::CX509Object]
10964:OcspService:
{CX509Certificate2::CX509Certificate2
10964:OcspService:                    [CX509Object::CX509Object]
10964:OcspService:
{CX500DistinguishedName::CX500DistinguishedName
10964:OcspService:                    [COidX500Name::COidX500Name]
...
10964:OcspService: }CrlManager::SetBaseCrl
10964:OcspService:    Установлен базовый список отзывов
10964:OcspService:    [CX509Decoder::GetExtension]
10964:OcspService:    {COcspMonitor::ChangeState
10964:OcspService:        {COcspMonitor::StartWorking
10964:OcspService:            [CrlManager::StartObservation]
10964:OcspService:            {COCSPResponder::COCSPResponder
10964:OcspService:                [CX509Object::CX509Object]
19508:OcspService: {CrlManager::CrlUpdaterThreadHandler
10964:OcspService:        {CX509Certificate2::CX509Certificate2
10964:OcspService:            [CX509Object::CX509Object]
10964:OcspService:
{CX500DistinguishedName::CX500DistinguishedName
19508:OcspService:    {COcspSettings::GetUpdateCrlPeriod
10964:OcspService:        [COidX500Name::COidX500Name]
19508:OcspService:    [CRegistryKey::GetSubKey]

```

```

10964:OcspService:
}CX500DistinguishedName::CX500DistinguishedName
19508:OcspService:      {CRegistryKey::GetDWORDValue
10964:OcspService:      [CX509Object::CX509Object]
19508:OcspService:      - Не удалось прочитать параметр
реестра FAILED! Line: 69 Status: 80070002
10964:OcspService:
{CX500DistinguishedName::CX500DistinguishedName
19508:OcspService:      }CRegistryKey::GetDWORDValue
10964:OcspService:      [COidX500Name::COidX500Name]
19508:OcspService:      }COcspSettings::GetUpdateCrlPeriod
10964:OcspService:
}CX500DistinguishedName::CX500DistinguishedName
19508:OcspService:      Поток обновления актуальных СОС запущен.
10964:OcspService:      [CX509Object::CX509Object]
10964:OcspService:      [CX509Object::CX509Object]
10964:OcspService:      [CPublicKey::CPublicKey]
10964:OcspService:      [CX509Object::CX509Object]
10964:OcspService:      [CX509Extension::CX509Extension]
10964:OcspService:      [CX509Extension::CX509Extension]
10964:OcspService:      [CX509KeyUsageExtension::Decode]
10964:OcspService:      [CX509Object::CX509Object]
10964:OcspService:      [CX509Extension::CX509Extension]
10964:OcspService:      [CX509Extension::CX509Extension]
10964:OcspService:
{CX509ExtKeyUsageExtension::CX509ExtKeyUsageExtension
10964:OcspService:
{CX509ExtKeyUsageExtension::Decode
10964:OcspService:      [CX509Decoder::ParseListOfX509]
10964:OcspService:
}CX509ExtKeyUsageExtension::Decode
10964:OcspService:
}CX509ExtKeyUsageExtension::CX509ExtKeyUsageExtension
10964:OcspService:      [CX509Object::CX509Object]
10964:OcspService:      [CX509Extension::CX509Extension]
10964:OcspService:      [CX509Extension::CX509Extension]
10964:OcspService:
{CX509PrivateKeyUsagePeriodExtension::CX509PrivateKeyUsagePeriod
Extension
10964:OcspService:
[CX509PrivateKeyUsagePeriodExtension::Decode]
10964:OcspService:
}CX509PrivateKeyUsagePeriodExtension::CX509PrivateKeyUsagePeriod
Extension
10964:OcspService:      [CX509Object::CX509Object]
10964:OcspService:      [CX509Extension::CX509Extension]
10964:OcspService:      [CX509Extension::CX509Extension]
10964:OcspService:
{CX509SubjectKeyIdentifierExtension::CX509SubjectKeyIdentifierEx
tension
10964:OcspService:
[CX509SubjectKeyIdentifierExtension::Decode]

```

```

10964:OcspService:
}CX509SubjectKeyIdentifierExtension::CX509SubjectKeyIdentifierEx
tension
10964:OcspService:          [CX509Object::CX509Object]
10964:OcspService:          [CX509Extension::CX509Extension]
10964:OcspService:          [CX509Extension::CX509Extension]
10964:OcspService:
[CX509BasicConstraintsExtension::Decode]
10964:OcspService:          [CX509Object::CX509Object]
10964:OcspService:          [CX509Extension::CX509Extension]
10964:OcspService:          [CX509Extension::CX509Extension]
10964:OcspService:
{CX509AuthorityKeyIdentifierExtension::CX509AuthorityKeyIdentifi
erExtension
10964:OcspService:
[CX509AuthorityKeyIdentifierExtension::Decode]
10964:OcspService:
}CX509AuthorityKeyIdentifierExtension::CX509AuthorityKeyIdentifi
erExtension
10964:OcspService:          [CX509Object::CX509Object]
10964:OcspService:          [CX509Extension::CX509Extension]
10964:OcspService:          [CX509Extension::CX509Extension]
10964:OcspService:          [CX509Object::CX509Object]
10964:OcspService:          [CX509Extension::CX509Extension]
10964:OcspService:          [CX509Extension::CX509Extension]
10964:OcspService:          }CX509Certificate2::CX509Certificate2
10964:OcspService:          }COCSPPResponder::COCSPPResponder
10964:OcspService:          {COCSPPResponder::Start
10964:OcspService:          [CRegistryKey::GetSubKey]
10964:OcspService:          {CRegistryKey::GetDWORDValue
10964:OcspService:          - Не удалось прочитать параметр
пеестра FAILED! Line: 69 Status: 80070002
10964:OcspService:          }CRegistryKey::GetDWORDValue
10964:OcspService:          COCSPPResponder начал работу
11732:OcspService: {COCSPPResponder::WorkerThread
11732:OcspService:   {COcspPkcs11Client::CreateSigner
10964:OcspService:   }COCSPPResponder::Start
10964:OcspService:   }COcspMonitor::StartWorking
10964:OcspService:   }COcspMonitor::ChangeState
10964:OcspService: }COcspMonitor::SetBaseCrl
11732:OcspService:   [CPkcs11Slot::CreateSession]
11732:OcspService:   }COcspPkcs11Client::CreateSigner
11732:OcspService:   {COcspResponseMaker::COcspResponseMaker
11732:OcspService:   [CX509Object::CX509Object]
19820:OcspService: [COcspMonitor::Beacon]
11732:OcspService:   {CX509Certificate2::CX509Certificate2
11732:OcspService:   [CX509Object::CX509Object]
11732:OcspService:
{CX500DistinguishedName::CX500DistinguishedName
11732:OcspService:          [COidX500Name::COidX500Name]
11732:OcspService:
}CX500DistinguishedName::CX500DistinguishedName
11732:OcspService:          [CX509Object::CX509Object]

```

```

11732:OcspService:
{CX500DistinguishedName::CX500DistinguishedName
11732:OcspService:      [COidX500Name::COidX500Name]
11732:OcspService:
}CX500DistinguishedName::CX500DistinguishedName
11732:OcspService:      [CX509Object::CX509Object]
11732:OcspService:      [CX509Object::CX509Object]
11732:OcspService:      [CPublicKey::CPublicKey]
11732:OcspService:      [CX509Object::CX509Object]
11732:OcspService:      [CX509Extension::CX509Extension]
11732:OcspService:      [CX509Extension::CX509Extension]
11732:OcspService:      [CX509KeyUsageExtension::Decode]
11732:OcspService:      [CX509Object::CX509Object]
11732:OcspService:      [CX509Extension::CX509Extension]
11732:OcspService:      [CX509Extension::CX509Extension]
11732:OcspService:
{CX509ExtKeyUsageExtension::CX509ExtKeyUsageExtension
11732:OcspService:      {CX509ExtKeyUsageExtension::Decode
11732:OcspService:      [CX509Decoder::ParseListOfX509]
11732:OcspService:      }CX509ExtKeyUsageExtension::Decode
11732:OcspService:
}CX509ExtKeyUsageExtension::CX509ExtKeyUsageExtension
11732:OcspService:      [CX509Object::CX509Object]
11732:OcspService:      [CX509Extension::CX509Extension]
11732:OcspService:      [CX509Extension::CX509Extension]
11732:OcspService:
{CX509PrivateKeyUsagePeriodExtension::CX509PrivateKeyUsagePeriod
Extension
11732:OcspService:
[CX509PrivateKeyUsagePeriodExtension::Decode]
11732:OcspService:
}CX509PrivateKeyUsagePeriodExtension::CX509PrivateKeyUsagePeriod
Extension
11732:OcspService:      [CX509Object::CX509Object]
11732:OcspService:      [CX509Extension::CX509Extension]
11732:OcspService:      [CX509Extension::CX509Extension]
11732:OcspService:
{CX509SubjectKeyIdentifierExtension::CX509SubjectKeyIdentifierEx
tension
11732:OcspService:
[CX509SubjectKeyIdentifierExtension::Decode]
11732:OcspService:
}CX509SubjectKeyIdentifierExtension::CX509SubjectKeyIdentifierEx
tension
11732:OcspService:      [CX509Object::CX509Object]
11732:OcspService:      [CX509Extension::CX509Extension]
11732:OcspService:      [CX509Extension::CX509Extension]
11732:OcspService:
[CX509BasicConstraintsExtension::Decode]
11732:OcspService:      [CX509Object::CX509Object]
11732:OcspService:      [CX509Extension::CX509Extension]
11732:OcspService:      [CX509Extension::CX509Extension]

```

```

11732:OcspService:
{CX509AuthorityKeyIdentifierExtension::CX509AuthorityKeyIdentifi
erExtension
11732:OcspService:
[CX509AuthorityKeyIdentifierExtension::Decode]
11732:OcspService:
}CX509AuthorityKeyIdentifierExtension::CX509AuthorityKeyIdentifi
erExtension
11732:OcspService:      [CX509Object::CX509Object]
11732:OcspService:      [CX509Extension::CX509Extension]
11732:OcspService:      [CX509Extension::CX509Extension]
11732:OcspService:      [CX509Object::CX509Object]
11732:OcspService:      [CX509Extension::CX509Extension]
11732:OcspService:      [CX509Extension::CX509Extension]
11732:OcspService:      }CX509Certificate2::CX509Certificate2
11732:OcspService:      {COcspResponseMaker::FailedResponse
11732:OcspService:      [CX509Builder::CX509Builder]
11732:OcspService:
{CX509OCSPResponseBuilder::CX509OCSPResponseBuilder
11732:OcspService:      [CX509Object::CX509Object]
11732:OcspService:
[CX500DistinguishedName::CX500DistinguishedName]
11732:OcspService:
}CX509OCSPResponseBuilder::CX509OCSPResponseBuilder
11732:OcspService:      [CX509OCSPResponseBuilder::SetStatus]
11732:OcspService:      [CX509OCSPResponseBuilder::Create]
11732:OcspService:      }COcspResponseMaker::FailedResponse
11732:OcspService:      {COcspResponseMaker::FailedResponse
11732:OcspService:      [CX509Builder::CX509Builder]
11732:OcspService:
{CX509OCSPResponseBuilder::CX509OCSPResponseBuilder
11732:OcspService:      [CX509Object::CX509Object]
11732:OcspService:
[CX500DistinguishedName::CX500DistinguishedName]
11732:OcspService:
}CX509OCSPResponseBuilder::CX509OCSPResponseBuilder
11732:OcspService:      [CX509OCSPResponseBuilder::SetStatus]
11732:OcspService:      [CX509OCSPResponseBuilder::Create]
11732:OcspService:      }COcspResponseMaker::FailedResponse
11732:OcspService:      {COcspResponseMaker::FailedResponse
11732:OcspService:      [CX509Builder::CX509Builder]
11732:OcspService:
{CX509OCSPResponseBuilder::CX509OCSPResponseBuilder
11732:OcspService:      [CX509Object::CX509Object]
11732:OcspService:
[CX500DistinguishedName::CX500DistinguishedName]
11732:OcspService:
}CX509OCSPResponseBuilder::CX509OCSPResponseBuilder
11732:OcspService:      [CX509OCSPResponseBuilder::SetStatus]
11732:OcspService:      [CX509OCSPResponseBuilder::Create]
11732:OcspService:      }COcspResponseMaker::FailedResponse
11732:OcspService:      {COcspResponseMaker::FailedResponse
11732:OcspService:      [CX509Builder::CX509Builder]

```

```

11732:OcspService:
{CX509OCSPResponseBuilder::CX509OCSPResponseBuilder
11732:OcspService:      [CX509Object::CX509Object]
11732:OcspService:
[CX500DistinguishedName::CX500DistinguishedName]
11732:OcspService:
}CX509OCSPResponseBuilder::CX509OCSPResponseBuilder
11732:OcspService:      [CX509OCSPResponseBuilder::SetStatus]
11732:OcspService:      [CX509OCSPResponseBuilder::Create]
11732:OcspService:      }COcspResponseMaker::FailedResponse
11732:OcspService:      {COcspResponseMaker::FailedResponse
11732:OcspService:      [CX509Builder::CX509Builder]
11732:OcspService:
{CX509OCSPResponseBuilder::CX509OCSPResponseBuilder
11732:OcspService:      [CX509Object::CX509Object]
11732:OcspService:
[CX500DistinguishedName::CX500DistinguishedName]
11732:OcspService:
}CX509OCSPResponseBuilder::CX509OCSPResponseBuilder
11732:OcspService:      [CX509OCSPResponseBuilder::SetStatus]
11732:OcspService:      [CX509OCSPResponseBuilder::Create]
11732:OcspService:      }COcspResponseMaker::FailedResponse
11732:OcspService:      }COcspResponseMaker::COcspResponseMaker
11732:OcspService:      Ожидание запроса

```

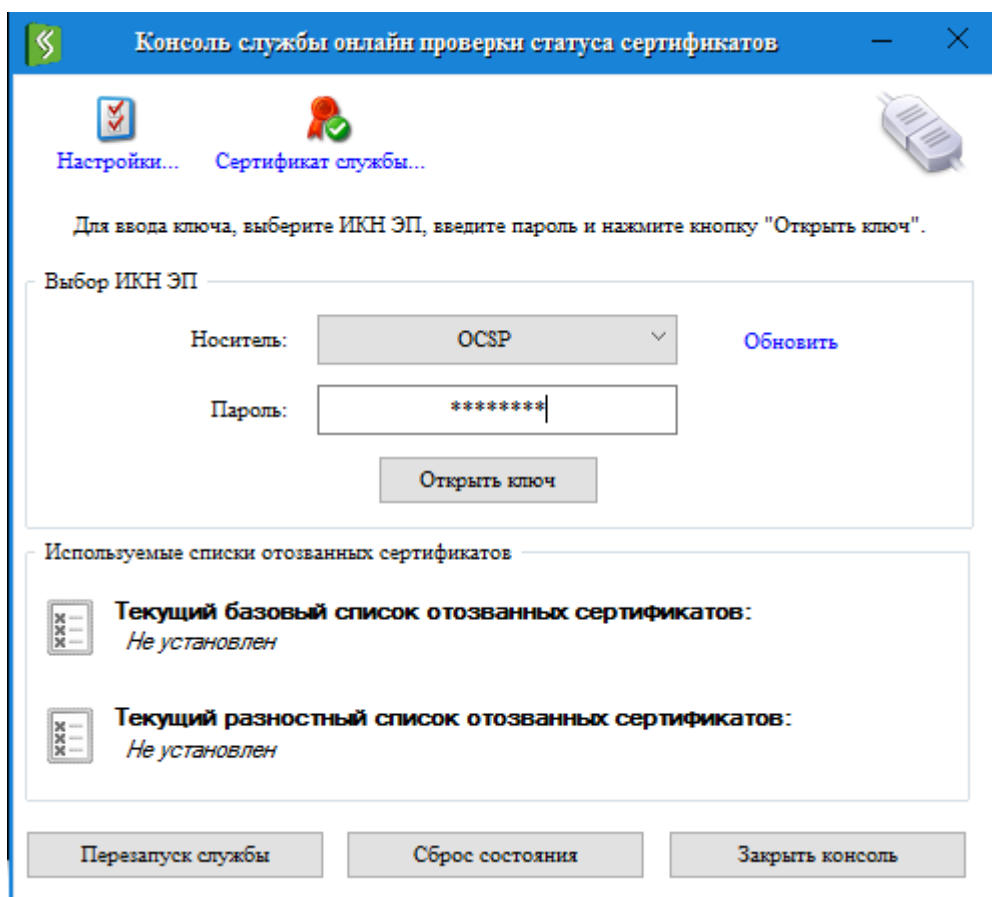


Рисунок В.2 – Отчет о неудачной инициализации токена

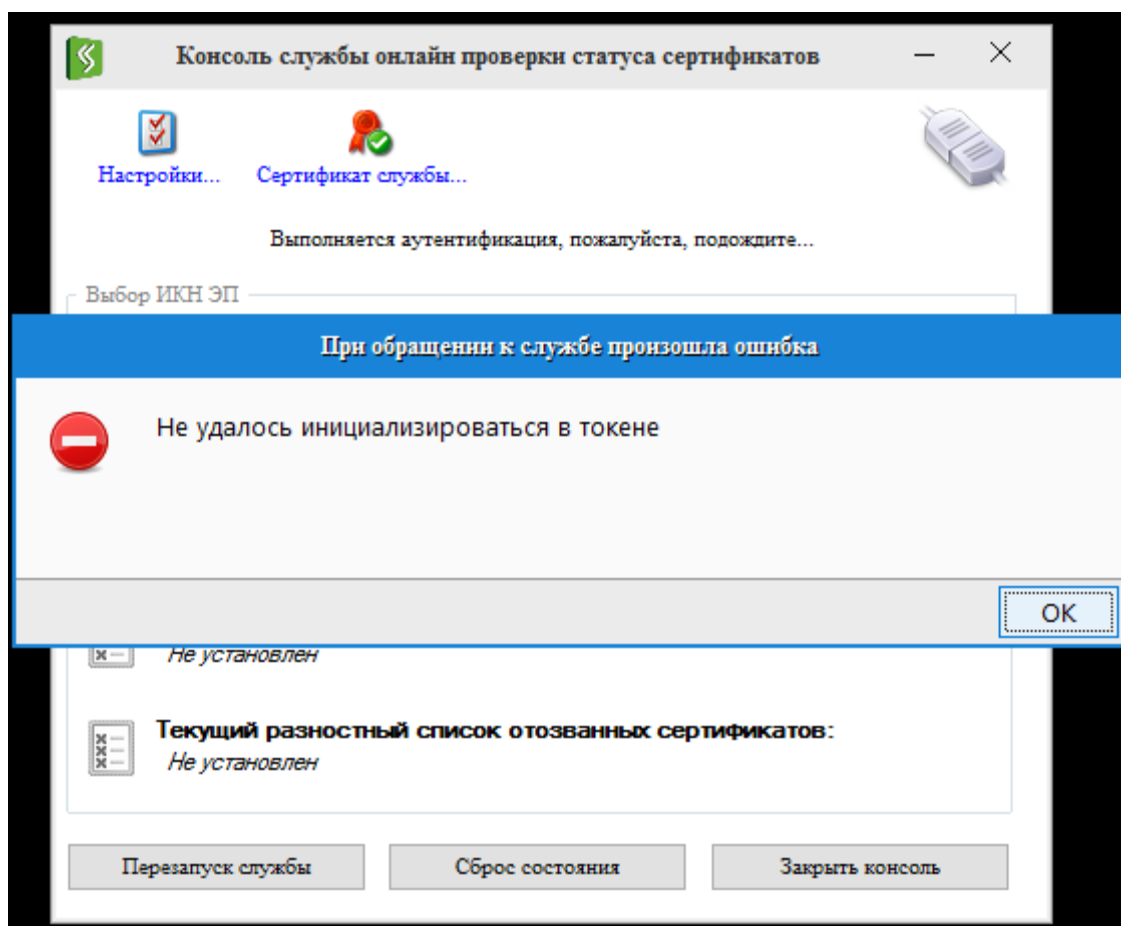


Рисунок В.3 – Отчет о неудачной инициализации токена

Листинг В.2 – Фрагмент лога, с ошибкой инициализации токена

```

16692:OcspService: {COcspMonitor::SelectSlot
16692:OcspService:   {CPkcs11Client::SelectSlot
16692:OcspService:     [CPkcs11::IsInitialized]
16692:OcspService:     [CPkcs11Slot::CreateSession]
16692:OcspService:   }CPkcs11Client::SelectSlot
16692:OcspService: [COcspMonitor::ChangeState]
16692:OcspService:   Слот выбран
16692:OcspService: }COcspMonitor::SelectSlot
3320:OcspService: [COcspMonitor::GetCurrentState]
20260:OcspService: [COcspMonitor::Beacon]
21192:OcspService: {COcspMonitor::LoginOnSlot
21192:OcspService:   {CPkcs11Client::Login
21192:OcspService:     {CPkcs11Slot::Login
21192:OcspService:       [CPkcs11::IsInitialized]
21192:OcspService:       [CPkcs11Slot::Logout]
21192:OcspService:       [CPkcs11Slot::CreateSession]
21192:OcspService:       Неверный пароль FAILED! Line: 77
21192:OcspService:       (LastError: CKR_PIN_INCORRECT)
21192:OcspService:     }CPkcs11Slot::Login
21192:OcspService:   m_pPkcs11Slot->Login FAILED! Line: 615
21192:OcspService:   (LastError: CKR_PIN_INCORRECT)
21192:OcspService: }CPkcs11Client::Login
21192:OcspService: {COcspMonitor::ErrorState

```

```

21192:Ocservice: [COcserviceMonitor::ChangeState]
21192:Ocservice: {COcserviceMonitor::StopWorking
21192:Ocservice: [CrlManager::StopObservation]
21192:Ocservice: }COcserviceMonitor::StopWorking
21192:Ocservice: [CPkcs11Client::CloseOpenedKeys]
21192:Ocservice: Ошибка в COCSPResponder FAILED! Line:
1187 (LastErr: CKR_PIN_INCORRECT)
21192:Ocservice: [COcserviceClient::ShowError]
21192:Ocservice: [COcserviceMonitor::ChangeState]
21192:Ocservice: }COcserviceMonitor::ErrorState
21192:Ocservice: }COcserviceMonitor::LoginOnSlot
20444:Ocservice: [COcserviceMonitor::GetCurrentState]
18592:Ocservice: {COcserviceMonitor::Reset
18592:Ocservice: {COcserviceMonitor::StopWorking
18592:Ocservice: [CrlManager::StopObservation]
18592:Ocservice: }COcserviceMonitor::StopWorking
18592:Ocservice: [CrlManager::StopObservation]
18592:Ocservice: [CrlManager::Clear]
18592:Ocservice: [CPkcs11Client::CloseOpenedKeys]
18592:Ocservice: [CPkcs11Slot::Logout]
18592:Ocservice: [CPkcs11::UnLoad]
18592:Ocservice: {CChachedStore::Clear
18592:Ocservice: [CLocalStore::Clear]
18592:Ocservice: }CChachedStore::Clear
18592:Ocservice: [CrlManager::CrlManager]
18592:Ocservice: [CRegistryKey::GetSubKey]
18592:Ocservice: [CRegistryKey::GetDWORDValue]
18592:Ocservice: {CPkcs11::CPkcs11
18592:Ocservice: [CPkcs11::Initialize]
18592:Ocservice: }CPkcs11::CPkcs11
18592:Ocservice: [CPkcs11::IsInitialized]
18592:Ocservice: [COcserviceMonitor::ChangeState]
18592:Ocservice: }COcserviceMonitor::Reset
20188:Ocservice: [COcserviceMonitor::Beacon]
20884:Ocservice: [COcserviceMonitor::GetCurrentState]
22928:Ocservice: [COcserviceMonitor::Beacon]
15540:Ocservice: {COcservicePkcs11Client::GetTokensInfo
15540:Ocservice: {CPkcs11Client::GetMatchedTokensInfo
15540:Ocservice: [CPkcs11::IsInitialized]
15540:Ocservice: {CPkcs11::GetSlotInfo
15540:Ocservice: [CPkcs11::IsInitialized]
15540:Ocservice: {CPkcs11::GetSlotList
15540:Ocservice: [CPkcs11::IsInitialized]
15540:Ocservice: }CPkcs11::GetSlotList
15540:Ocservice: }CPkcs11::GetSlotInfo
15540:Ocservice: [CPkcs11Client::MatchSlot]
15540:Ocservice: [CPkcs11Client::MatchSlot]
15540:Ocservice: [CPkcs11Client::MatchSlot]
15540:Ocservice: }CPkcs11Client::GetMatchedTokensInfo
15540:Ocservice: }COcservicePkcs11Client::GetTokensInfo
15540:Ocservice: }COcserviceMonitor::GetSlotList

```

Листинг В.3 – Лог, подтверждающий успешный сброс состояния сервера

```

23292:Ocservice: [COcserviceMonitor::GetCurrentState]
3688:Ocservice: [COcserviceMonitor::Beacon]
20984:Ocservice: [COcserviceMonitor::GetCurrentState]
21268:Ocservice: [COcserviceMonitor::Beacon]
21104:Ocservice: {COcserviceMonitor::Reset
21104:Ocservice:     {COcserviceMonitor::StopWorking
21104:Ocservice:         [CrlManager::StopObservation]
21104:Ocservice:     }COcserviceMonitor::StopWorking
21104:Ocservice:     [CrlManager::StopObservation]
21104:Ocservice:     [CrlManager::Clear]
21104:Ocservice:     [CPkcs11Client::CloseOpenedKeys]
21104:Ocservice:     [CPkcs11::UnLoad]
21104:Ocservice:     {CChachedStore::Clear
21104:Ocservice:         [CLocalStore::Clear]
21104:Ocservice:     }CChachedStore::Clear
21104:Ocservice:     [CrlManager::CrlManager]
21104:Ocservice:     [CRegistryKey::GetSubKey]
21104:Ocservice:     [CRegistryKey::GetDWORDValue]
21104:Ocservice:     {CPkcs11::CPkcs11
21104:Ocservice:         [CPkcs11::Initialize]
21104:Ocservice:     }CPkcs11::CPkcs11
21104:Ocservice:     [CPkcs11::IsInitialized]
21104:Ocservice:     [COcserviceMonitor::ChangeState]
21104:Ocservice: }COcserviceMonitor::Reset
17828:Ocservice: [COcserviceMonitor::GetCurrentState]
17872:Ocservice: [COcserviceMonitor::Beacon]
22008:Ocservice: [COcserviceMonitor::GetCurrentState]

```

Листинг В.4 – Ответ запроса о статусе

```

[0 ( 15.06.2020 16:40:31 ) ]
Приверяемый сертификат: 1e 5a e5 86 98 3e c0 83 4c 16 17 42 e5
00 00 a2
Отправка запроса: успешно
Статус ответа: успешно
Проверка подписи ответа: успешно (подпись действительна)
Статус сертификата: действителен
Длительность (сеть): 213
Длительность (общая): 224

```

Листинг В.5 – Фрагмент лога обработки запроса

```

22924:Ocservice: Получен запрос
22924:Ocservice: {COcserviceResponseMaker::MakeResponse
22924:Ocservice:     [CX509Object::CX509Object]
22924:Ocservice:     [CX509OCSPRequest::CX509OCSPRequest]
22924:Ocservice:     {CX509OCSPRequest::CX509OCSPRequest
22924:Ocservice:         {CX509OCSPRequest::Decode
22924:Ocservice:             {CX509OCSPRequest::ParseTBS
22924:Ocservice:                 {CX509OCSPRequest::ParseRequestList
22924:Ocservice:                     [CX509Object::CX509Object]
22924:Ocservice:                     [CX509Request::CX509Request]

```

```

22924:OcspService:          {CX509Request::CX509Request
22924:OcspService:          {CX509Request::Decode
22924:OcspService:          {CX509CertID::CX509CertID
22924:OcspService:          {CX509CertID::Decode
22924:OcspService:
[CX509Decoder::ParseAlgorithmId]
22924:OcspService:
[CX509Decoder::ParseSerialNumber]
22924:OcspService:          }CX509CertID::Decode
22924:OcspService:          }CX509CertID::CX509CertID
22924:OcspService:          }CX509Request::Decode
22924:OcspService:          }CX509Request::CX509Request
22924:OcspService:          }CX509OCSPRequest::ParseRequestList
22924:OcspService:          }CX509OCSPRequest::ParseTBS
22924:OcspService:          }CX509OCSPRequest::Decode
22924:OcspService:          }CX509OCSPRequest::CX509OCSPRequest
22924:OcspService:          [CX509Builder::CX509Builder]
22924:OcspService:
{CX509OCSPResponseBuilder::CX509OCSPResponseBuilder
22924:OcspService:          [CX509Object::CX509Object]
22924:OcspService:
[CX500DistinguishedName::CX500DistinguishedName]
22924:OcspService:
}CX509OCSPResponseBuilder::CX509OCSPResponseBuilder
22924:OcspService:          {COcspResponseMaker::UpdateCrl
22924:OcspService:          [CrlManager::GetBaseCrlInstallTime]
22924:OcspService:          [CrlManager::GetDeltaCrlInstallTime]
22924:OcspService:          }COcspResponseMaker::UpdateCrl
22924:OcspService:
{COcspResponseMaker::CheckRequestSignature
22924:OcspService:          {COcspSettings::IsSigRequestNeeded
22924:OcspService:          [CRegistryKey::GetSubKey]
22924:OcspService:          {CRegistryKey::GetDWORDValue
22924:OcspService:          - Не удалось прочитать параметр
реестра FAILED! Line: 69 Status: 80070002
22924:OcspService:          }CRegistryKey::GetDWORDValue
22924:OcspService:          }COcspSettings::IsSigRequestNeeded
22924:OcspService:
}COcspResponseMaker::CheckRequestSignature
22924:OcspService:          {CX509OCSPRequest::GetRequest
22924:OcspService:          [CX509Object::CX509Object]
22924:OcspService:          [CX509Request::CX509Request]
22924:OcspService:          [CX509Request::CX509Request]
22924:OcspService:          }CX509OCSPRequest::GetRequest
22924:OcspService:          {COcspResponseMaker::MakeSingleResponse
22924:OcspService:          [CX509Object::CX509Object]
22924:OcspService:
[CX509SingleResponse::CX509SingleResponse]
22924:OcspService:          [CX509SingleResponse::SetCertID]
22924:OcspService:          {CACHEDStore::GetCertificate
22924:OcspService:          {CLOCALStore::TryGetObject
22924:OcspService:          [CX509Object::CX509Object]
22924:OcspService:
{CX509Certificate2::CX509Certificate2

```

```

22924:OcspService:          [CX509Object::CX509Object]
22924:OcspService:
{CX500DistinguishedName::CX500DistinguishedName
22924:OcspService:          [COidX500Name::COidX500Name]
22924:OcspService:
}CX500DistinguishedName::CX500DistinguishedName
22924:OcspService:          [CX509Object::CX509Object]
22924:OcspService:
{CX500DistinguishedName::CX500DistinguishedName
22924:OcspService:          [COidX500Name::COidX500Name]
22924:OcspService:
}CX500DistinguishedName::CX500DistinguishedName
22924:OcspService:          [CX509Object::CX509Object]
22924:OcspService:          [CX509Object::CX509Object]
22924:OcspService:          [CPublicKey::CPublicKey]
22924:OcspService:          [CX509Object::CX509Object]
22924:OcspService:          [CX509Extension::CX509Extension]
22924:OcspService:          [CX509Extension::CX509Extension]
22924:OcspService:          [CX509KeyUsageExtension::Decode]
22924:OcspService:          [CX509Object::CX509Object]
22924:OcspService:          [CX509Extension::CX509Extension]
22924:OcspService:          [CX509Extension::CX509Extension]
22924:OcspService:
{CX509ExtKeyUsageExtension::CX509ExtKeyUsageExtension
22924:OcspService:
{CX509ExtKeyUsageExtension::Decode
22924:OcspService:
[CX509Decoder::ParseListOfX509]
22924:OcspService:
}CX509ExtKeyUsageExtension::Decode
22924:OcspService:
}CX509ExtKeyUsageExtension::CX509ExtKeyUsageExtension
22924:OcspService:          [CX509Object::CX509Object]
22924:OcspService:          [CX509Extension::CX509Extension]
22924:OcspService:          [CX509Extension::CX509Extension]
22924:OcspService:
{CX509PrivateKeyUsagePeriodExtension::CX509PrivateKeyUsagePeriod
Extension
22924:OcspService:
[CX509PrivateKeyUsagePeriodExtension::Decode]
22924:OcspService:
}CX509PrivateKeyUsagePeriodExtension::CX509PrivateKeyUsagePeriod
Extension
22924:OcspService:          [CX509Object::CX509Object]
22924:OcspService:          [CX509Extension::CX509Extension]
22924:OcspService:          [CX509Extension::CX509Extension]
22924:OcspService:
{CX509SubjectKeyIdentifierExtension::CX509SubjectKeyIdentifierEx
tension
22924:OcspService:
[CX509SubjectKeyIdentifierExtension::Decode]
22924:OcspService:
}CX509SubjectKeyIdentifierExtension::CX509SubjectKeyIdentifierEx
tension

```

```

22924:OcspService: [CX509Object::CX509Object]
22924:OcspService: [CX509Extension::CX509Extension]
22924:OcspService: [CX509Extension::CX509Extension]
22924:OcspService: [CX509BasicConstraintsExtension::Decode]
22924:OcspService: [CX509Object::CX509Object]
22924:OcspService: [CX509Extension::CX509Extension]
22924:OcspService: [CX509Extension::CX509Extension]
22924:OcspService: {CX509AuthorityKeyIdentifierExtension::CX509AuthorityKeyIdentifi
erExtension
22924:OcspService: [CX509AuthorityKeyIdentifierExtension::Decode]
22924:OcspService: }CX509AuthorityKeyIdentifierExtension::CX509AuthorityKeyIdentifi
erExtension
22924:OcspService: [CX509Object::CX509Object]
22924:OcspService: [CX509Extension::CX509Extension]
22924:OcspService: [CX509Extension::CX509Extension]
22924:OcspService: [CX509Object::CX509Object]
22924:OcspService: [CX509Extension::CX509Extension]
22924:OcspService: [CX509Extension::CX509Extension]
22924:OcspService: }CX509Certificate2::CX509Certificate2
22924:OcspService: }CLocalStore::TryGetObject
22924:OcspService: }CChachedStore::GetCertificate
22924:OcspService: {COcspResponseMaker::CheckInCrl
22924:OcspService: [CX509SingleResponse::SetStatus]
22924:OcspService: }COcspResponseMaker::CheckInCrl
22924:OcspService: [CX509SingleResponse::SetThisUpdateDate]
22924:OcspService: [CX509SingleResponse::SetNextUpdateDate]
22924:OcspService: }COcspResponseMaker::MakeSingleResponse
22924:OcspService: [CX509OCSPResponseBuilder::AddX509Response]
22924:OcspService: {CX509OCSPRequest::GetExtension
22924:OcspService: [CX509Decoder::GetExtension]
22924:OcspService: }CX509OCSPRequest::GetExtension
22924:OcspService: [CX509OCSPResponseBuilder::SetProducedAtDateTime]
22924:OcspService: [CX509Object::CX509Object]
22924:OcspService: {CX500DistinguishedName::CX500DistinguishedName
22924:OcspService: [COidX500Name::COidX500Name]
22924:OcspService: }CX500DistinguishedName::CX500DistinguishedName
22924:OcspService: [CX509OCSPResponseBuilder::SetName]
22924:OcspService: [CX509Object::CX509Object]
22924:OcspService: {CX509Certificate2::CX509Certificate2
22924:OcspService: [CX509Object::CX509Object]
22924:OcspService: {CX500DistinguishedName::CX500DistinguishedName
22924:OcspService: [COidX500Name::COidX500Name]

```

```

22924:OcspService:
}CX500DistinguishedName::CX500DistinguishedName
22924:OcspService:      [CX509Object::CX509Object]
22924:OcspService:
{CX500DistinguishedName::CX500DistinguishedName
22924:OcspService:      [COidX500Name::COidX500Name]
22924:OcspService:
}CX500DistinguishedName::CX500DistinguishedName
22924:OcspService:      [CX509Object::CX509Object]
22924:OcspService:      [CX509Object::CX509Object]
22924:OcspService:      [CPublicKey::CPublicKey]
22924:OcspService:      [CX509Object::CX509Object]
22924:OcspService:      [CX509Extension::CX509Extension]
22924:OcspService:      [CX509Extension::CX509Extension]
22924:OcspService:      [CX509KeyUsageExtension::Decode]
22924:OcspService:      [CX509Object::CX509Object]
22924:OcspService:      [CX509Extension::CX509Extension]
22924:OcspService:      [CX509Extension::CX509Extension]
22924:OcspService:
{CX509ExtKeyUsageExtension::CX509ExtKeyUsageExtension
22924:OcspService:      {CX509ExtKeyUsageExtension::Decode
22924:OcspService:      [CX509Decoder::ParseListOfX509]
22924:OcspService:      }CX509ExtKeyUsageExtension::Decode
22924:OcspService:
}CX509ExtKeyUsageExtension::CX509ExtKeyUsageExtension
22924:OcspService:      [CX509Object::CX509Object]
22924:OcspService:      [CX509Extension::CX509Extension]
22924:OcspService:      [CX509Extension::CX509Extension]
22924:OcspService:
{CX509PrivateKeyUsagePeriodExtension::CX509PrivateKeyUsagePeriod
Extension
22924:OcspService:
[CX509PrivateKeyUsagePeriodExtension::Decode]
22924:OcspService:
}CX509PrivateKeyUsagePeriodExtension::CX509PrivateKeyUsagePeriod
Extension
22924:OcspService:      [CX509Object::CX509Object]
22924:OcspService:      [CX509Extension::CX509Extension]
22924:OcspService:      [CX509Extension::CX509Extension]
22924:OcspService:
{CX509SubjectKeyIdentifierExtension::CX509SubjectKeyIdentifierEx
tension
22924:OcspService:
[CX509SubjectKeyIdentifierExtension::Decode]
22924:OcspService:
}CX509SubjectKeyIdentifierExtension::CX509SubjectKeyIdentifierEx
tension
22924:OcspService:      [CX509Object::CX509Object]
22924:OcspService:      [CX509Extension::CX509Extension]
22924:OcspService:      [CX509Extension::CX509Extension]
22924:OcspService:
[CX509BasicConstraintsExtension::Decode]
22924:OcspService:      [CX509Object::CX509Object]
22924:OcspService:      [CX509Extension::CX509Extension]

```

```

22924:OcspService:      [CX509Extension::CX509Extension]
22924:OcspService:
{CX509AuthorityKeyIdentifierExtension::CX509AuthorityKeyIdentifi
erExtension
22924:OcspService:
[CX509AuthorityKeyIdentifierExtension::Decode]
22924:OcspService:
}CX509AuthorityKeyIdentifierExtension::CX509AuthorityKeyIdentifi
erExtension
22924:OcspService:      [CX509Object::CX509Object]
22924:OcspService:      [CX509Extension::CX509Extension]
22924:OcspService:      [CX509Extension::CX509Extension]
22924:OcspService:      [CX509Object::CX509Object]
22924:OcspService:      [CX509Extension::CX509Extension]
22924:OcspService:      [CX509Extension::CX509Extension]
22924:OcspService:      }CX509Certificate2::CX509Certificate2
22924:OcspService:
[CX509OCSPResponseBuilder::AddX509Certificate]
22924:OcspService:      [CX509OCSPResponseBuilder::SetStatus]
22924:OcspService:      {CSupportOfCryptX509::Sign
22924:OcspService:      {CX509OCSPResponseBuilder::Create
22924:OcspService:
{CX509OCSPResponseBuilder::GetSigningData
22924:OcspService:      {CX500DistinguishedName::Encode
22924:OcspService:      [CoidX500Name::Encode]
22924:OcspService:      }CX500DistinguishedName::Encode
22924:OcspService:      {CX509Encoder::EncodeListOfX509
22924:OcspService:      {CX509SingleResponse::Encode
22924:OcspService:      {CX509CertID::Encode
22924:OcspService:
[CX509Encoder::EncodeAlgorithmIdentifier]
22924:OcspService:      }CX509CertID::Encode
22924:OcspService:      }CX509SingleResponse::Encode
22924:OcspService:      }CX509Encoder::EncodeListOfX509
22924:OcspService:
}CX509OCSPResponseBuilder::GetSigningData
22924:OcspService:      {CPkcs11Signer::PkcsSign
22924:OcspService:      [CPkcs11Session::SignData]
22924:OcspService:      }CPkcs11Signer::PkcsSign
22924:OcspService:      {CX509OCSPResponseBuilder::Build
22924:OcspService:
[CX509Encoder::EncodeAlgorithmIdentifier]
22924:OcspService:      {CX509Encoder::EncodeListOfX509
22924:OcspService:      [CX509Certificate2::Encode]
22924:OcspService:      }CX509Encoder::EncodeListOfX509
22924:OcspService:      }CX509OCSPResponseBuilder::Build
22924:OcspService:      }CX509OCSPResponseBuilder::Create
22924:OcspService:      }CSupportOfCryptX509::Sign
22924:OcspService:      }COcspResponseMaker::MakeResponse
22924:OcspService:      Отправлен ответ

```

Листинг В.6 – Фрагмент лога обработки некорректного запроса

```

22924:OcspService:    Получен запрос
22924:OcspService:    {COcspResponseMaker::MakeResponse
22924:OcspService:    [CX509Object::CX509Object]
22924:OcspService:    [CX509OCSPRequest::CX509OCSPRequest]
22924:OcspService:    {CX509OCSPRequest::CX509OCSPRequest
22924:OcspService:    {CX509OCSPRequest::Decode
22924:OcspService:    {CX509OCSPRequest::ParseTBS
22924:OcspService:    - В структуре сертификата имеется
ошибка FAILED! Line: 269 Status: 0x80093103

```

Листинг В.7 – Ответ о статусе сертификата

```

[0 ( 15.06.2020 16:42:33 ) ]
Приверяемый сертификат: c7 3b 1f 86 98 3e c0 83 4c 16 17 42 e5
00 01 03
Отправка запроса: успешно
Статус ответа: успешно
Проверка подписи ответа: успешно (подпись действительна)
Статус сертификата: неизвестный
Длительность (сеть): 197
Длительность (общая): 212

```

Листинг В.8 – Ответ о статусе сертификата

```

[0 ( 15.06.2020 16:49:12 ) ]
Приверяемый сертификат: 12 2a 94 86 98 3e c0 83 4c 16 17 42 e5
00 01 03
Отправка запроса: успешно
Статус ответа: успешно
Проверка подписи ответа: успешно (подпись действительна)
Статус сертификата: отозванный
Длительность (сеть): 234
Длительность (общая): 255

```

Приложение Г
Исходный код функций тестирования
(рекомендуемое)

Листинг Г.1 – TestFunctions.h

```

#pragma once
/// <summary>
/// Создание TSP ответа на основе другого TSP ответа
/// </summary>
/// <param name="pResponseSourceData">Данные TSP ответа, который
копируется.</param>
/// <param name="responseSourceDataSize">Размер TSP ответа, который
копируется.</param>
/// <param name="ppResponseData">Данные TSP ответа.</param>
/// <param name="pResponseDataSize">Размер TSP ответа.</param>
/// <returns>Статус выполнения операции.</returns>
extern "C" __declspec(dllexport)
HRESULT TSPResponseTest (
    __in_bcount(responseSourceDataSize)    PBYTE pResponseSourceData,
    __in                                     DWORD
    responseSourceDataSize,
    __out_bcount(*pResponseDataSize)       PBYTE*    ppResponseData,
    __out                                     DWORD*
    pResponseDataSize);
/// <summary>
/// Создание TSP запроса на основе другого TSP запроса
/// </summary>
/// <param name="pRequestSourceData">Данные TSP запроса, который
копируется.</param>
/// <param name="RequestSourceDataSize">Размер TSP запроса, который
копируется.</param>
/// <param name="ppRequestData">Данные TSP запроса.</param>
/// <param name="pRequestDataSize">Размер TSP запроса.</param>
/// <returns>Статус выполнения операции.</returns>
extern "C" __declspec(dllexport)
HRESULT TSPRequestTest (
    __in_bcount(RequestSourceDataSize)     PBYTE pRequestSourceData,
    __in                                     DWORD
    RequestSourceDataSize,
    __out_ecount(*pRequestDataSize)        PBYTE*
    ppRequestData,
    __out                                     DWORD*
    pRequestDataSize);
/// <summary>
/// Создание OSCP ответа на основе другого OSCP ответа
/// </summary>
/// <param name="pResponseSourceData">Данные OSCP ответа, который
копируется.</param>
/// <param name="responseSourceDataSize">Размер OSCP ответа, который
копируется.</param>
/// <param name="ppResponseData">Данные OSCP ответа.</param>
/// <param name="pResponseDataSize">Размер OSCP ответа.</param>
/// <returns>Статус выполнения операции.</returns>
extern "C" __declspec(dllexport)
HRESULT OSCPResponseTest (
    __in_bcount(responseSourceDataSize)     PBYTE pResponseSourceData,
    __in                                     DWORD
    responseSourceDataSize,
    __out_bcount(*pResponseDataSize)        PBYTE*    ppResponseData,
    __out                                     DWORD*
    pResponseDataSize);
/// <summary>
/// Создание OSCP запроса на основе другого OSCP запроса
/// </summary>
/// <param name="pRequestSourceData">Данные OSCP запроса, который
копируется.</param>

```

```

/// <param name="requestSourceDataSize">Размер OCSP запроса, который
копируется.</param>
/// <param name="ppRequestData">Данные OCSP запроса.</param>
/// <param name="pRequestDataSize">Размер OCSP запроса.</param>
/// <returns>Статус выполнения операции.</returns>
extern "C" __declspec(dllexport)
HRESULT OCSPRequestTest(
    __in_bcount(requestSourceDataSize)          PBYTE pRequestSourceData,
    __in                                          DWORD
    requestSourceDataSize,
    __out_bcount(*pRequestDataSize)             PBYTE*
    ppRequestData,
    __out                                        DWORD*
    pRequestDataSize);
/// <summary>
/// Создание списка отзывов на основе другого списка отзывов
/// </summary>
/// <param name="pCrlSourceData">Данные списка отзывов, который
копируется.</param>
/// <param name="crlSourceDataSize">Размер списка отзывов, который
копируется.</param>
/// <param name="ppCrlData">Данные списка отзывов.</param>
/// <param name="pCrlDataSize">Размер списка отзывов.</param>
/// <returns>Статус выполнения операции.</returns>
extern "C" __declspec(dllexport)
HRESULT CrlTest(
    __in_bcount(crlSourceDataSize)              PBYTE pCrlSourceData,
    __in                                          DWORD crlSourceDataSize,
    __out_bcount(*pCrlDataSize)                 PBYTE* ppCrlData,
    __out                                        DWORD*
    pCrlDataSize);
/// <summary>
/// Создание сертификата на основе другого сертификата
/// </summary>
/// <param name="pCertificateSourceData">Данные сертификата, который
копируется.</param>
/// <param name="certificateSourceDataSize">Размер сертификата, который
копируется.</param>
/// <param name="ppCertificateData">Данные сертификата.</param>
/// <param name="pCertificateDataSize">Размер сертификата.</param>
/// <returns>Статус выполнения операции.</returns>
extern "C" __declspec(dllexport)
HRESULT CertificateTest(
    __in_bcount(certificateSourceDataSize)       PBYTE
    pCertificateSourceData,
    __in                                          DWORD
    certificateSourceDataSize,
    __out_bcount(*pCertificateDataSize)          PBYTE*
    ppCertificateData,
    __out                                        DWORD*
    pCertificateDataSize);

/// <summary>
/// Генерация ответа на OCSP запрос
/// </summary>
/// <param name="pRequest">Данные OCSP запроса.</param>
/// <param name="requestSize">Размер данных OCSP запроса.</param>
/// <param name="pResponse">Данные OCSP ответа.</param>
/// <param name="pResponseSize">Размер OCSP ответа.</param>
/// <returns>Статус выполнения операции.</returns>
extern "C" __declspec(dllexport)
HRESULT
OCSPResponse(
    __in_bcount(RequestSize)                    PBYTE pRequest,

```

```

        __in                                DWORD requestSize,
        __out_bcount(*pResponseSize) PBYTE pResponse,
        __out                                DWORD*          pResponseSize);
/// <summary>
/// Генерация ответа на TSP запрос
/// </summary>
/// <param name="pRequest">Данные TSP запроса.</param>
/// <param name="requestSize">Размер данных TSP запроса.</param>
/// <param name="pResponse">Данные TSP ответа.</param>
/// <param name="pResponseSize">Размер TSP ответа.</param>
/// <returns>Статус выполнения операции.</returns>
extern "C" __declspec(dllexport)
HRESULT
TSPResponse(
    __in_bcount(RequestSize)          PBYTE pRequest,
    __in                                DWORD requestSize,
    __out_bcount(*pResponseSize) PBYTE pResponse,
    __out                                DWORD*          pResponseSize);
extern "C" __declspec(dllexport)
HRESULT TestTokenLogin();

```

Листинг Г.2 – TestFunctions.cpp

```

#include "stdafx.h"
CSupportOfCryptX509* g_Signer = new CSupportOfCryptX509(new
CNullSigner(), FALSE);
#pragma region Extension
template<class T>
CX509Extension* GetNewExtension(T* ext)
{
    return ext;
}
template<>
CX509Extension* GetNewExtension(CX509PrivateKeyUsagePeriodExtension*
ext)
{
    CX509PrivateKeyUsagePeriodExtension* newExt = new
CX509PrivateKeyUsagePeriodExtension();
    newExt->SetNotAfterDateTime(ext->GetNotAfterDate());
    newExt->SetNotBeforeDateTime(ext->GetNotBeforeDate());

    delete ext;
    return newExt;
}
template<>
CX509Extension* GetNewExtension(CX509CertsAuthorityVersionExtension*
ext)
{
    CX509CertsAuthorityVersionExtension* newExt = new
CX509CertsAuthorityVersionExtension();
    newExt->SetCertificateVersion(ext->GetCertificateVersion());
    newExt->SetKeyVersion(ext->GetKeyVersion());
    delete ext;
    return newExt;
}
template<>
CX509Extension* GetNewExtension(CX509BasicConstraintsExtension* ext)
{
    CX509BasicConstraintsExtension* newExt = new
CX509BasicConstraintsExtension();
    newExt->SetcA(ext->GetcA());
    if (ext->GetPathLenConstraint())
    {
        newExt->SetPathLenConstraint(*ext->GetPathLenConstraint());
    }
}

```

```

        }
        delete ext;
        return newExt;
    }
    template<>
    CX509Extension* GetNewExtension(CX509SubjectKeyIdentifierExtension* ext)
    {
        CX509SubjectKeyIdentifierExtension* newExt = new
CX509SubjectKeyIdentifierExtension();
        newExt->SetKeyIdentifier(ext->GetKeyIdentifier());
        delete ext;
        return newExt;
    }
    template<>
    CX509Extension* GetNewExtension(CX509KeyUsageExtension* ext)
    {
        CX509KeyUsageExtension* newExt = new CX509KeyUsageExtension();
        WORD us;
        ext->GetKeyUsages(&us);

        newExt->SetKeyUsageFlags(us);
        delete ext;
        return newExt;
    }
    template<>
    CX509Extension* GetNewExtension(CX509ExtKeyUsageExtension* ext)
    {
        CX509ExtKeyUsageExtension* newExt = new
CX509ExtKeyUsageExtension();
        for (DWORD i = 0; i < ext->GetKeyPurposeIdCount(); ++i)
        {
            newExt->AddKeyPurposeId(new COid(*ext->GetKeyPurposeId(i)));
        }

        delete ext;
        return newExt;
    }
    template<>
    CX509Extension* GetNewExtension(CX509AuthorityKeyIdentifierExtension*
ext)
    {
        CX509AuthorityKeyIdentifierExtension* newExt = new
CX509AuthorityKeyIdentifierExtension();
        newExt->SetKeyIdentifier(ext->GetKeyIdentifier());
        delete ext;
        return newExt;
    }
    template<>
    CX509Extension* GetNewExtension(CX509CertificatePoliciesExtension* ext)
    {
        CX509CertificatePoliciesExtension* newExt = new
CX509CertificatePoliciesExtension();
        for (DWORD i = 0; i < ext->GetCountPolicies(); ++i)
        {
            newExt->AddPolicy(ext->GetPolicy(i));
        }
        delete ext;
        return newExt;
    }
    template <class T, class Source, class Dest>
    HRESULT TestExtension(Source source, Dest dest, PWCHAR s)
    {
        T* ext = NULL;

```

```

        HRESULT Status = source->GetExtension(s, wcslen(s),
(CX509Extension**) &ext);
        if (Status == HRESULT_FROM_WIN32(ERROR_NO_MORE_ITEMS))
        {
            return S_OK;
        }
        CHECK_HR_STATUS_RET1(Status);

        Status = dest->SetX509Extension(GetNewExtension(ext));
        CHECK_HR_STATUS_RET1(Status);
        return S_OK;
    }
#pragma endregion
template <class Source, class Dest>
HRESULT TestExtensions(Source source, Dest dest)
{
    HRESULT Status;

    Status = TestExtension<CX509SubjectKeyIdentifierExtension>(source,
dest, OID_X509_SUBJECT_KEY_IDENTIFIER_EXT);
    CHECK_HR_STATUS_RET1(Status);
    Status =
TestExtension<CX509AuthorityKeyIdentifierExtension>(source, dest,
OID_X509_AUTHORITY_KEY_IDENTIFIER_EXT);
    CHECK_HR_STATUS_RET1(Status);
    Status = TestExtension<CX509KeyUsageExtension>(source, dest,
OID_X509_KEY_USAGE_EXTENSION);
    CHECK_HR_STATUS_RET1(Status);
    Status = TestExtension<CX509CRLReasonExtension>(source, dest,
OID_X509_CRL_REASON_EXTENSION);
    CHECK_HR_STATUS_RET1(Status);
    Status = TestExtension<CX509CrlNumberExtension>(source, dest,
OID_X509_CRL_NUMBER_EXTENSION);
    CHECK_HR_STATUS_RET1(Status);
    Status = TestExtension<CX509DeltaCRLExtension>(source, dest,
OID_X509_CRL_DELTA_INDICATOR_EXTENSION);
    CHECK_HR_STATUS_RET1(Status);
    Status = TestExtension<CX509CertificatePoliciesExtension>(source,
dest, OID_X509_CERTIFICATES_POLICIES_EXTENSION);
    CHECK_HR_STATUS_RET1(Status);
    Status = TestExtension<CX509FreshestCRLExtension>(source, dest,
OID_X509_CRL_FRESHEST_EXTENSION);
    CHECK_HR_STATUS_RET1(Status);
    Status = TestExtension<CX509CrlDistributionPointExtension>(source,
dest, OID_X509_CRL_DISTR_POINT_EXT);
    CHECK_HR_STATUS_RET1(Status);
    Status = TestExtension<CX509BasicConstraintsExtension>(source,
dest, OID_X509_BASIC_CONSTRAINTS_EXTENSION);
    CHECK_HR_STATUS_RET1(Status);
    Status = TestExtension<CX509ExtKeyUsageExtension>(source, dest,
OID_X509_EXTENDED_KEY_USAGE_EXTENSION);
    CHECK_HR_STATUS_RET1(Status);
    Status =
TestExtension<CX509PrivateKeyUsagePeriodExtension>(source, dest,
OID_X509_PRIVATE_KEY_USAGE_PERIOD_EXTENSION);
    CHECK_HR_STATUS_RET1(Status);
    Status = TestExtension<CX509Extension>(source, dest,
OID_X509_PKIX_OCSP_NONCE);
    CHECK_HR_STATUS_RET1(Status);
    Status =
TestExtension<CX509CertsAuthorityVersionExtension>(source, dest,
OID_X509_CERTS_AUTHORITY_VERSION_EXT);
    CHECK_HR_STATUS_RET1(Status);

```

```

        Status = TestExtension<CX509CryptosoftExtension>(source, dest,
OID_X509_CRYPTOSOFT_CERT_CHARACTER_EXT);
        CHECK_HR_STATUS_RET1(Status);
        Status = TestExtension<CX509CryptosoftExtension>(source, dest,
OID_X509_CRYPTOSOFT_QPOS_CERT_CHARACTER_EXT);
        CHECK_HR_STATUS_RET1(Status);
        return S_OK;
    }
    /// <summary>
    /// Создание TSP ответа на основе другого TSP ответа
    /// </summary>
    /// <param name="pResponseSourceData">Данные TSP ответа, который
копируется.</param>
    /// <param name="responseSourceDataSize">Размер TSP ответа, который
копируется.</param>
    /// <param name="ppResponseData">Данные TSP ответа.</param>
    /// <param name="pResponseDataSize">Размер TSP ответа.</param>
    /// <returns>Статус выполнения операции.</returns>
    HRESULT TSPResponseTest(
        __in_bcount(responseSourceDataSize)        PBYTE
        pResponseSourceData,
        __in                                          DWORD
        responseSourceDataSize,
        __out_bcount(*pResponseDataSize)            PBYTE*
        ppResponseData,
        __out                                         DWORD*
        pResponseDataSize)
    {
        LOG_PROC_ENTRY;
        HRESULT Status = E_FAIL;
        CX509TSPResponseBuilder *pResponseBuilder = new
CX509TSPResponseBuilder();
        PBYTE          pBytesData          = NULL;
        PWCHAR          pCharsData          = NULL;
        DWORD           DataSize            = 0;
        CX509SignerInfo* pSignerInfo        = NULL;
        FILETIME         Time               = { 0 };
        CByteBuffer      Data;
        CX509TSPResponse* pResponseSource = new
CX509TSPResponse((PBYTE)pResponseSourceData, responseSourceDataSize);
        Status = pResponseSource->GetInitStatus();
        CHECK_HR_STATUS_GOTO(Status, Finally);
        pResponseBuilder->SetNonce(*pResponseSource->GetNonce());
        pResponseBuilder->SetAccuracy(pResponseSource->GetAccuracy());
        pResponseBuilder->SetOrdering(pResponseSource->GetOrdering());
        pResponseBuilder->SetPKIStatus(pResponseSource->GetPKIStatus());
        pResponseSource->GetGenTime(&Time);
        pResponseBuilder->SetGenTime(Time);
        COid* Oid = NULL;
        Status = pResponseSource->GetDigestAlgorithm(0, &Oid);
        CHECK_HR_STATUS_GOTO(Status, Finally);
        Status = pResponseBuilder->SetDigestAlgorithm(Oid);
        CHECK_HR_STATUS_GOTO(Status, Finally);
        delete Oid;
        Status = pResponseSource->GetHashMessageAlgorithmOid(pCharsData,
&DataSize);
        CHECK_HR_STATUS_GOTO(Status, Finally);
        pCharsData = new WCHAR[DataSize];
        Status = pResponseSource->GetHashMessageAlgorithmOid(pCharsData,
&DataSize);
        CHECK_HR_STATUS_GOTO(Status, Finally);
        Status = pResponseBuilder->SetHashMessageAlgorithm(pCharsData,
DataSize);
        CHECK_HR_STATUS_GOTO(Status, Finally);

```

```

delete[] pCharsData;
pCharsData = NULL;
DataSize = 0;
Status = pResponseSource->GetPolicyOid(pCharsData, &DataSize);
CHECK_HR_STATUS_GOTO(Status, Finally);
pCharsData = new WCHAR[DataSize];
Status = pResponseSource->GetPolicyOid(pCharsData, &DataSize);
CHECK_HR_STATUS_GOTO(Status, Finally);
Status = pResponseBuilder->SetPolicy(pCharsData, DataSize);
CHECK_HR_STATUS_GOTO(Status, Finally);
delete[] pCharsData;
pCharsData = NULL;
DataSize = 0;
Status = pResponseSource->GetSerialNumber(pBytesData, &DataSize);
CHECK_HR_STATUS_GOTO(Status, Finally);
pBytesData = new BYTE[DataSize];
Status = pResponseSource->GetSerialNumber(pBytesData, &DataSize);
CHECK_HR_STATUS_GOTO(Status, Finally);
Status = pResponseBuilder->SetSerialNumber(pBytesData, DataSize);
CHECK_HR_STATUS_GOTO(Status, Finally);
delete[] pBytesData;
pBytesData = NULL;
DataSize = 0;
Status = pResponseSource->GetHashedMessage(pBytesData, &DataSize);
CHECK_HR_STATUS_GOTO(Status, Finally);
pBytesData = new BYTE[DataSize];
Status = pResponseSource->GetHashedMessage(pBytesData, &DataSize);
CHECK_HR_STATUS_GOTO(Status, Finally);
Status = pResponseBuilder->SetHashedMessage(pBytesData, DataSize);
CHECK_HR_STATUS_GOTO(Status, Finally);
delete[] pBytesData;
pBytesData = NULL;
DataSize = 0;
Status = pResponseSource->GetSignerInfo(0, &pSignerInfo);
CHECK_HR_STATUS_GOTO(Status, Finally);
pResponseBuilder->SetSubjectIdentifier(pSignerInfo-
>GetIssuerAndSerialNumber());
delete pSignerInfo;
pSignerInfo = NULL;
DataSize = 36;
pBytesData = new BYTE[DataSize];
memset(pBytesData, 0, DataSize);
pResponseBuilder->SetCertificateHash(pBytesData, DataSize);
delete[] pBytesData;
pBytesData = NULL;
DataSize = 0;
Status = g_Signer->Sign(*pResponseBuilder, Data);
CHECK_HR_STATUS_GOTO(Status, Finally);
*ppResponseData = new BYTE[Data.Size];
memcpy(*ppResponseData, Data, Data.Size);
*ppResponseDataSize = Data.Size;
Finally:
delete[] pBytesData;
delete[] pCharsData;
delete pSignerInfo;
delete pResponseBuilder;
delete pResponseSource;
return Status;
}
/// <summary>
/// Создание TSP запроса на основе другого TSP запроса
/// </summary>
/// <param name="pRequestSourceData">Данные TSP запроса, который
копируется.</param>

```

```

    /// <param name="RequestSourceDataSize">Размер TSP запроса, который
копируется.</param>
    /// <param name="ppRequestData">Данные TSP запроса.</param>
    /// <param name="pRequestDataSize">Размер TSP запроса.</param>
    /// <returns>Статус выполнения операции.</returns>
extern "C" __declspec(dllexport)
HRESULT TSPRequestTest(
    __in_bcount(RequestSourceDataSize)        PBYTE
    pRequestSourceData,
    __in                                         DWORD
    RequestSourceDataSize,
    __out_ecount(*pRequestDataSize)           PBYTE*
    ppRequestData,
    __out                                       DWORD*
    pRequestDataSize)
{
    LOG_PROC_ENTRY;
    HRESULT Status = E_FAIL;
    CX509TSPRequest* pRequestSource = new
CX509TSPRequest(pRequestSourceData, RequestSourceDataSize);
    PBYTE          pBytesData = NULL;
    PWCHAR          pCharsData = NULL;
    DWORD          DataSize = 0;
    INT64 *rr;
    CAsn1 * Res = NULL;
    CX509TSPRequest *pRequest = new CX509TSPRequest();
    Status = pRequestSource->GetInitStatus();
    CHECK_HR_STATUS_GOTO(Status, Finally);
    Status = TestExtensions(pRequestSource, pRequest);
    CHECK_HR_STATUS_GOTO(Status, Finally);
    Status = pRequestSource->GetHashAlgorithmOid(pCharsData,
&DataSize);
    CHECK_HR_STATUS_GOTO(Status, Finally);
    pCharsData = new WCHAR[DataSize];
    Status = pRequestSource->GetHashAlgorithmOid(pCharsData,
&DataSize);
    CHECK_HR_STATUS_GOTO(Status, Finally);
    Status = pRequest->SetHashAlgorithm(pCharsData, DataSize);
    CHECK_HR_STATUS_GOTO(Status, Finally);
    delete[] pCharsData;
    pCharsData = NULL;
    DataSize = 0;
    Status = pRequestSource->GetHashedMessage(pBytesData, &DataSize);
    CHECK_HR_STATUS_GOTO(Status, Finally);
    pBytesData = new BYTE[DataSize];
    Status = pRequestSource->GetHashedMessage(pBytesData, &DataSize);
    CHECK_HR_STATUS_GOTO(Status, Finally);
    Status = pRequest->SetHashedMessage(pBytesData, DataSize);
    CHECK_HR_STATUS_GOTO(Status, Finally);
    delete[] pBytesData;
    pBytesData = NULL;
    DataSize = 0;
    rr = pRequestSource->GetNonce();
    pRequest->SetNonce(*rr);
    delete rr;
    Status = pRequest->Encode(&Res);
    CHECK_HR_STATUS_GOTO(Status, Finally);
    *ppRequestData = NULL;
    *pRequestDataSize = 0;
    Status = Res->Encode(*ppRequestData, (size_t*)pRequestDataSize);
    CHECK_HR_STATUS_GOTO(Status, Finally);
    *ppRequestData = new BYTE[*pRequestDataSize];
    Status = Res->Encode(*ppRequestData, (size_t*)pRequestDataSize);
    CHECK_HR_STATUS_GOTO(Status, Finally);

```

```

Finally:
    delete[]    pBytesData;
    delete[]    pCharsData;
    delete      Res;
    delete      pRequestSource;
    delete      pRequest;
    return Status;
}
/// <summary>
/// Создание OCSP ответа на основе другого OCSP ответа
/// </summary>
/// <param name="pResponseSourceData">Данные OCSP ответа, который
копируется.</param>
/// <param name="responseSourceDataSize">Размер OCSP ответа, который
копируется.</param>
/// <param name="ppResponseData">Данные OCSP ответа.</param>
/// <param name="pResponseDataSize">Размер OCSP ответа.</param>
/// <returns>Статус выполнения операции.</returns>
HRESULT OCSPResponseTest (
    __in_bcount(responseSourceDataSize)        PBYTE
    pResponseSourceData,
    __in                                          DWORD
    responseSourceDataSize,
    __out_bcount(*pResponseDataSize)            PBYTE*
    ppResponseData,
    __out                                        DWORD*
    pResponseDataSize)
{
    LOG_PROC_ENTRY;
    HRESULT Status = E_FAIL;
    PBYTE    pBytesData = NULL;
    PWCHAR    pCharsData = NULL;
    DWORD    DataSize = 0;
    FILETIME Time = { 0 };
    CX509SingleResponse* pSingleResponse;
    CX509SingleResponse* pCx509SingleResponse;
    CAsn1 * dd;
    CX509SingleResponse * ddd;
    CByteBuffer Data;
    CX509OCSPResponse* pResponseSource = new
CX509OCSPResponse(pResponseSourceData, responseSourceDataSize);
    CX509OCSPResponseBuilder *pResponseBuilder = new
CX509OCSPResponseBuilder();
    Status = pResponseSource->GetInitStatus();
    CHECK_HR_STATUS_GOTO(Status, Finally);
    pResponseBuilder->SetStatus(X509OCSPResponseStatus::SUCCESSFUL);
    Status = TestExtensions(pResponseSource, pResponseBuilder);
    CHECK_HR_STATUS_GOTO(Status, Finally);
    Status = pResponseBuilder->SetName(pResponseSource->GetName());
    CHECK_HR_STATUS_GOTO(Status, Finally);
    for (int i = 0; i < pResponseSource->GetCountSingleResponse();
++i)
    {
        pSingleResponse = NULL;
        Status = pResponseSource->GetSingleResponse(i,
&pSingleResponse);
        CHECK_HR_STATUS_GOTO(Status, Finally);
        pCx509SingleResponse = new CX509SingleResponse();
        memset(&Time, 0, sizeof FILETIME);
        pSingleResponse->GetThisUpdateDate(&Time);
        pCx509SingleResponse->SetThisUpdateDate(&Time);
        pCx509SingleResponse->SetNextUpdateDate(pSingleResponse-
>GetNextUpdateDate());
        pSingleResponse->GetRevocationTime(&Time);

```

```

        pCx509SingleResponse->SetRevocationTime(&Time);
        pCx509SingleResponse->SetCertID(pSingleResponse->
>GetCertID());
        pCx509SingleResponse->SetStatus(pSingleResponse->
>GetStatus());
        pCx509SingleResponse->SetReason(pSingleResponse->
>GetReason());

        dd = NULL;
        pCx509SingleResponse->Encode(&dd);
        ddd = new CX509SingleResponse(*dd);
        Status = pResponseBuilder->AddX509Response(ddd);
        CHECK_HR_STATUS_GOTO(Status, Finally);
        delete pSingleResponse;
        delete pCx509SingleResponse;
        delete dd;
    }
    for (int i = 0; i < pResponseSource->GetCountCertificate(); ++i)
    {
        CX509Certificate2* pRevokedCertificate = NULL;
        Status = pResponseSource->GetCertificate(i,
&pRevokedCertificate);
        CHECK_HR_STATUS_GOTO(Status, Finally);
        Status = pResponseBuilder->
>AddX509Certificate(pRevokedCertificate);
        CHECK_HR_STATUS_GOTO(Status, Finally);
    }
    pResponseSource->GetProducedAt(&Time);
    pResponseBuilder->SetProducedAtDateTime(Time);
    Status = g_Signer->Sign(*pResponseBuilder, Data);
    CHECK_HR_STATUS_GOTO(Status, Finally);
    *ppResponseData = new BYTE[Data.Size];
    memcpy(*ppResponseData, Data, Data.Size);
    *pResponseDataSize = Data.Size;
Finally:
    delete[] pBytesData;
    delete[] pCharsData;
    delete pResponseSource;
    delete pResponseBuilder;
    return Status;
}
/// <summary>
/// Создание OCSP запроса на основе другого OCSP запроса
/// </summary>
/// <param name="pRequestSourceData">Данные OCSP запроса, который
копируется.</param>
/// <param name="requestSourceDataSize">Размер OCSP запроса, который
копируется.</param>
/// <param name="ppRequestData">Данные OCSP запроса.</param>
/// <param name="pRequestDataSize">Размер OCSP запроса.</param>
/// <returns>Статус выполнения операции.</returns>
HRESULT OCSPRequestTest(
    __in_bcount(requestSourceDataSize)        PBYTE
pRequestSourceData,
    __in                                         DWORD
requestSourceDataSize,
    __out_bcount(*pRequestDataSize)           PBYTE*
ppRequestData,
    __out                                       DWORD*
pRequestDataSize)
{
    LOG_PROC_ENTRY;
    HRESULT Status = E_FAIL;
    PBYTE pBytesData = NULL;
    PWCHAR pCharsData = NULL;

```

```

        DWORD      DataSize = 0;
        FILETIME    time;
        CByteBuffer Data;
        CX509OCSPRequest* pRequestSource = new
CX509OCSPRequest((PBYTE)pRequestSourceData, requestSourceDataSize);
        Status = pRequestSource->GetInitStatus();
        CHECK_HR_STATUS_GOTO(Status, Finally);
        CX509OCSPRequestBuilder* pRequestBuilder = new
CX509OCSPRequestBuilder();
        Status = TestExtensions(pRequestSource, pRequestBuilder);
        CHECK_HR_STATUS_GOTO(Status, Finally);
        Status = pRequestBuilder->SetRequesterName(pRequestSource-
>GetRequesterName());
        CHECK_HR_STATUS_GOTO(Status, Finally);
        for (int i = 0; i < pRequestSource->GetCountRequest(); ++i)
        {
            CX509Request* pRequest = NULL;
            Status = pRequestSource->GetRequest(i, &pRequest);
            CHECK_HR_STATUS_GOTO(Status, Finally);
            CX509Request* pCx509SingleRequest = new CX509Request();
            memset(&time, 0, sizeof FILETIME);
            pCx509SingleRequest->SetCertID(pRequest->GetCertID());
            CAsn1* dd = NULL;
            Status = pCx509SingleRequest->Encode(&dd);
            CHECK_HR_STATUS_GOTO(Status, Finally);
            CX509Request* ddd = new CX509Request(*dd);
            Status = ddd->GetInitStatus();
            CHECK_HR_STATUS_GOTO(Status, Finally);
            Status = pRequestBuilder->AddX509Request(ddd);
            CHECK_HR_STATUS_GOTO(Status, Finally);
            delete pRequest;
            delete pCx509SingleRequest;
            delete dd;
        }
        for (int i = 0; i < pRequestSource->GetCountCertificate(); ++i)
        {
            CX509Certificate2* pRevokedCertificate = NULL;
            pRequestSource->GetCertificate(i, &pRevokedCertificate);
            pRequestBuilder->AddX509Certificate(pRevokedCertificate);
        }
        Status = g_Signer->Sign(*pRequestBuilder, Data);
        CHECK_HR_STATUS_GOTO(Status, Finally);
        *ppRequestData = new BYTE[Data.Size];
        memcpy(*ppRequestData, Data, Data.Size);
        *pRequestDataSize = Data.Size;
    Finally:
        delete[] pBytesData;
        delete[] pCharsData;
        delete pRequestSource;
        delete pRequestBuilder;
        return Status;
    }

    /// <summary>
    /// Создание списка отзывов на основе другого списка отзывов
    /// </summary>
    /// <param name="pCrlSourceData">Данные списка отзывов, который
копируется.</param>
    /// <param name="crlSourceDataSize">Размер списка отзывов, который
копируется.</param>
    /// <param name="ppCrlData">Данные списка отзывов.</param>
    /// <param name="pCrlDataSize">Размер списка отзывов.</param>
    /// <returns>Статус выполнения операции.</returns>
extern "C" __declspec(dllexport)
HRESULT CrlTest(

```

```

        __in_bcount(crlSourceDataSize)                PBYTE pCrlSourceData,
        __in                                              DWORD
crlSourceDataSize,
        __out_bcount(*pCrlDataSize)                  PBYTE*      ppCrlData,
        __out                                              DWORD*
pCrlDataSize)
{
    LOG_PROC_ENTRY;
    HRESULT Status = E_FAIL;
    PBYTE pBytesData = NULL;
    PWCHAR pCharsData = NULL;
    DWORD DataSize = 0;
    FILETIME time = { 0 };
    CByteBuffer Data;
    map<vector<BYTE>, CX509RevokedCertificate*> RCert;
    CX509CrlBuilder CrlBuilder;
    CX509Crl* pCrl = new CX509Crl(pCrlSourceData, crlSourceDataSize);
    Status = pCrl->GetInitStatus();
    CHECK_HR_STATUS_GOTO(Status, Finally);
    Status = pCrl->GetSignatureAlgorithmOid(pCharsData, &DataSize);
    CHECK_HR_STATUS_GOTO(Status, Finally);
    pCharsData = new WCHAR[DataSize];
    Status = pCrl->GetSignatureAlgorithmOid(pCharsData, &DataSize);
    CHECK_HR_STATUS_GOTO(Status, Finally);
    Status = CrlBuilder.SetAlgorithmIdentifier(pCharsData, DataSize);
    CHECK_HR_STATUS_GOTO(Status, Finally);
    delete[] pCharsData;
    pCharsData = NULL;
    DataSize = 0;
    Status = TestExtensions(pCrl, &CrlBuilder);
    CHECK_HR_STATUS_GOTO(Status, Finally);
    Status = CrlBuilder.SetIssuerName(pCrl->GetIssuerName());
    CHECK_HR_STATUS_GOTO(Status, Finally);
    pCrl->GetThisUpdateDate(&time);
    CrlBuilder.SetThisUpdateDateTime(time);
    CrlBuilder.SetNextUpdateDateTime(pCrl->GetNextUpdateDate());
    RCert = pCrl->GetRevokedCertificates();
    for (auto i = RCert.begin(); i != RCert.end(); ++i)
    {
        CX509RevokedCertificate* pRevokedCertificate =
(CX509RevokedCertificate*)(i->second)->Copy();

        CX509RevokedCertificate* pCX509SingleResponse = new
CX509RevokedCertificate();
        memset(&time, 0, sizeof FILETIME);
        Status = pRevokedCertificate->GetSerialNumber(pBytesData,
&DataSize);

        CHECK_HR_STATUS_GOTO(Status, Finally);
        pBytesData = new BYTE[DataSize];
        Status = pRevokedCertificate->GetSerialNumber(pBytesData,
&DataSize);

        CHECK_HR_STATUS_GOTO(Status, Finally);
        Status = pCX509SingleResponse->SetSerialNumber(pBytesData,
DataSize);

        CHECK_HR_STATUS_GOTO(Status, Finally);
        delete[] pBytesData;
        pBytesData = NULL;
        DataSize = 0;
        pRevokedCertificate->GetRevocationDate(&time);
        pCX509SingleResponse->SetRevocationDate(time);
        CAsn1 * pNewRevokedCertificateAsn1 = NULL;
        Status = pCX509SingleResponse-
>Encode(&pNewRevokedCertificateAsn1);
        CHECK_HR_STATUS_GOTO(Status, Finally);
    }
}

```

```

        CX509RevokedCertificate * pNewRevokedCertificate = new
CX509RevokedCertificate(*pNewRevokedCertificateAsn1);
        Status = pNewRevokedCertificate->GetInitStatus();
        CHECK_HR_STATUS_GOTO(Status, Finally);
        Status =
CrlBuilder.AddRevokedCertificate(pNewRevokedCertificate);
        CHECK_HR_STATUS_GOTO(Status, Finally);
        delete pRevokedCertificate;
        delete pNewRevokedCertificateAsn1;
        delete pCX509SingleResponse;
    }
    Status = g_Signer->Sign(CrlBuilder, Data);
    CHECK_HR_STATUS_GOTO(Status, Finally);
    *ppCrlData = new BYTE[Data.Size];
    memcpy(*ppCrlData, Data, Data.Size);
    *pCrlDataSize = Data.Size;
Finally:
    delete[] pBytesData;
    delete[] pCharsData;
    delete pCrl;
    return Status;
}
/// <summary>
/// Создание сертификата на основе другого сертификата
/// </summary>
/// <param name="pCertificateSourceData">Данные сертификата, который
копируется.</param>
/// <param name="certificateSourceDataSize">Размер сертификата, который
копируется.</param>
/// <param name="ppCertificateData">Данные сертификата.</param>
/// <param name="pCertificateDataSize">Размер сертификата.</param>
/// <returns>Статус выполнения операции.</returns>
extern "C" __declspec(dllexport)
HRESULT CertificateTest(
    __in_bcount(certificateSourceDataSize)          PBYTE
pCertificateSourceData,
    __in                                              DWORD
certificateSourceDataSize,
    __out_bcount(*pCertificateDataSize)             PBYTE*
ppCertificateData,
    __out                                            DWORD*
pCertificateDataSize)
{
    LOG_PROC_ENTRY;
    HRESULT Status = E_FAIL;
    PBYTE pBytesData = NULL;
    PWSTR pCharsData = NULL;
    DWORD DataSizeBytes = 0;
    DWORD DataSizeChars = 0;
    CX509Certificate2Builder *pCertificateBuilder = NULL;
    FILETIME time = { 0 };
    CByteBuffer Data;
    CX509Certificate2* pCertificate = new
CX509Certificate2(pCertificateSourceData, certificateSourceDataSize);
    Status = pCertificate->GetInitStatus();
    CHECK_HR_STATUS_GOTO(Status, Finally);
    pCertificateBuilder = new CX509Certificate2Builder();
    Status = pCertificate->GetSerialNumber(pBytesData,
&DataSizeBytes);
    CHECK_HR_STATUS_GOTO(Status, Finally);
    pBytesData = new BYTE[DataSizeBytes];
    Status = pCertificate->GetSerialNumber(pBytesData,
&DataSizeBytes);
    CHECK_HR_STATUS_GOTO(Status, Finally);

```

```

        Status = pCertificateBuilder->SetSerialNumber(pBytesData,
DataSizeBytes);
        CHECK_HR_STATUS_GOTO(Status, Finally);
        delete[] pBytesData;
        pBytesData = NULL;
        DataSizeBytes = 0;
        CAsn1* Par = NULL;
        Status = pCertificate->GetPublicKey(pBytesData, &DataSizeBytes,
pCharsData, &DataSizeChars);
        CHECK_HR_STATUS_GOTO(Status, Finally);
        pBytesData = new BYTE[DataSizeBytes];
        pCharsData = new WCHAR[DataSizeChars];
        Status = pCertificate->GetPublicKey(pBytesData, &DataSizeBytes,
pCharsData, &DataSizeChars, &Par);
        CHECK_HR_STATUS_GOTO(Status, Finally);
        Status = pCertificateBuilder->SetPublicKey(pCharsData,
DataSizeChars, pBytesData, DataSizeBytes, Par);
        CHECK_HR_STATUS_GOTO(Status, Finally);
        delete[] pBytesData;
        delete[] pCharsData;
        pBytesData = NULL;
        pCharsData = NULL;
        DataSizeBytes = 0;
        DataSizeChars = 0;
        Status = pCertificate->GetSignatureAlgorithmOid(pCharsData,
&DataSizeChars);
        CHECK_HR_STATUS_GOTO(Status, Finally);
        pCharsData = new WCHAR[DataSizeChars];
        Status = pCertificate->GetSignatureAlgorithmOid(pCharsData,
&DataSizeChars);
        CHECK_HR_STATUS_GOTO(Status, Finally);
        Status = pCertificateBuilder->SetAlgorithmIdentifier(pCharsData,
DataSizeChars);
        CHECK_HR_STATUS_GOTO(Status, Finally);
        delete[] pCharsData;
        pCharsData = NULL;
        DataSizeChars = 0;
        Status = TestExtensions(pCertificate, pCertificateBuilder);
        Status = pCertificate->GetSubjectName(OID_COMMON_NAME,
wcslen(OID_COMMON_NAME), pCharsData, &DataSizeChars);
        CHECK_HR_STATUS_GOTO(Status, Finally);
        pCharsData = new WCHAR[DataSizeChars];
        Status = pCertificate->GetSubjectName(OID_COMMON_NAME,
wcslen(OID_COMMON_NAME), pCharsData, &DataSizeChars);
        CHECK_HR_STATUS_GOTO(Status, Finally);
        Status = pCertificateBuilder->SetSubjectName(OID_COMMON_NAME,
wcslen(OID_COMMON_NAME), pCharsData, DataSizeChars);
        CHECK_HR_STATUS_GOTO(Status, Finally);
        delete[] pCharsData;
        pCharsData = NULL;
        DataSizeChars = 0;
        Status = pCertificate->GetIssuerName(OID_COMMON_NAME,
wcslen(OID_COMMON_NAME), pCharsData, &DataSizeChars);
        CHECK_HR_STATUS_GOTO(Status, Finally);
        pCharsData = new WCHAR[DataSizeChars];
        Status = pCertificate->GetIssuerName(OID_COMMON_NAME,
wcslen(OID_COMMON_NAME), pCharsData, &DataSizeChars);
        CHECK_HR_STATUS_GOTO(Status, Finally);
        Status = pCertificateBuilder->SetIssuerName(OID_COMMON_NAME,
wcslen(OID_COMMON_NAME), pCharsData, DataSizeChars);
        CHECK_HR_STATUS_GOTO(Status, Finally);
        delete[] pCharsData;
        pCharsData = NULL;
        DataSizeChars = 0;

```

```

        time = pCertificate->GetNotAfterDate();
        pCertificateBuilder->SetNotAfterDateTime(time);
        time = pCertificate->GetNotBeforeDate();
        pCertificateBuilder->SetNotBeforeDateTime(time);
        Status = g_Signer->Sign(*pCertificateBuilder, Data);
        CHECK_HR_STATUS_GOTO(Status, Finally);
        *ppCertificateData = new BYTE[Data.Size];
        memcpy(*ppCertificateData, Data, Data.Size);
        *pCertificateDataSize = Data.Size;
    Finally:
        delete[] pBytesData;
        delete[] pCharsData;
        delete pCertificate;
        delete pCertificateBuilder;
        return Status;
    }
    HRESULT TestTokenLogin()
    {
        LOG_PROC_ENTRY;
        CK_RV rv = CKR_OK;
        auto t = new CPkcs11();
        if (!t->IsInitialized())
        {
            LogMsg("Не инициализировались");
            return E_FAIL;
        }
        DWORD Count = 0;
        CK_SLOT_ID* p = NULL;
        SlotInfo* p1 = NULL;
        rv = t->GetSlotList(p, Count);
        CHECK_CK_STATUS_RETRV("GetSlotList", rv);
        Count = 0;
        rv = t->GetSlotInfo(p1, Count);
        CHECK_CK_STATUS_RETRV("GetSlotList", rv);
        if (Count == 0)
        {
            LogMsg("Нет слотов");
            return E_FAIL;
        }
        auto Slot = new CPkcs11Slot(t, p[2]);
        rv = Slot->Login((PBYTE)"11111111", 8);
        CHECK_CK_STATUS_RETRV("Login", rv);
        LOG_WRITE("Залогинились");
        CPkcs11Session* Session = NULL;
        rv = Slot->CreateSession(FALSE, Session);
        CHECK_CK_STATUS_RETRV("CreateSession", rv);
        CK_OBJECT_HANDLE Key = CK_INVALID_HANDLE;
        BYTE KeyId[] = { 0x80, 0x27, 0xd1, 0x1f, 0x8a, 0x5e, 0xe3, 0x46,
0xba, 0x5d, 0xed, 0xd2, 0x88, 0x9c, 0xb8, 0x89 };
        rv = Session->FindKey(KeyId, sizeof(KeyId), Key);
        CHECK_CK_STATUS_RETRV("FindKey", rv);

        const CK_MECHANISM MechanismSign = {
CKM_GOSTR3410_WITH_GOSTR3411_12_512, NULL, 0 };
        const CK_MECHANISM MechanismDigest = {
CKM_GOSTR3411_12_512, NULL, 0 };
        CHorNetSigner* Signer = new CHorNetSigner(Session, Key, 128, 64,
MechanismSign, MechanismDigest);
        g_Signer = new CSupportOfCryptX509(Signer, FALSE);
        if (!g_Signer)
        {
            LogMsg("g_Signer не создан успешно");
            return E_FAIL;
        }
    }

```

```

        return S_OK;
    }
    /// <summary>
    /// Генерация OCSP ответа, содержащего код ошибки
    /// </summary>
    /// <param name="ResponseStatus">Ошибка.</param>
    /// <param name="pResponse">Данные OCSP ответа.</param>
    /// <param name="pResponseSize">Размер OCSP ответа.</param>
    /// <returns>Статус выполнения операции.</returns>
    HRESULT
    FailedResponse(
        __in X509OCSPResponseStatus
    ResponseStatus,
        __out_bcount(*pResponseSize) BYTE*
    pResponse,
        __out DWORD*
        pResponseSize)
    {
        LOG_PROC_ENTRY;
        HRESULT Status = E_FAIL;
        CX509OCSPResponseBuilder ResponseBuilder;
        ResponseBuilder.SetStatus(ResponseStatus);
        CByteBuffer Data;
        Status = ResponseBuilder.Create(NULL, Data);
        CHECK_HR_STATUS_RET1(Status);
        memcpy(pResponse, Data.Data, Data.Size);
        *pResponseSize = Data.Size;
        return S_OK;
    }
#define CHECK_HR_FOR_RESPONSE(hr) if (FAILED(hr)) \
    { \
        delete pSingleRequest;\
        delete pSingleResponse;\
        return
FailedResponse(X509OCSPResponseStatus::INTERNAL_ERROR, pResponse,
pResponseSize);\
    }
    /// <summary>
    /// Генерация ответа на OCSP запрос
    /// </summary>
    /// <param name="pRequest">Данные OCSP запроса.</param>
    /// <param name="requestSize">Размер данных OCSP запроса.</param>
    /// <param name="pResponse">Данные OCSP ответа.</param>
    /// <param name="pResponseSize">Размер OCSP ответа.</param>
    /// <returns>Статус выполнения операции.</returns>
    HRESULT
    OCSPResponse(
        __in_bcount(RequestSize) PBYTE pRequest,
        __in DWORD requestSize,
        __out_bcount(*pResponseSize) PBYTE pResponse,
        __out DWORD*
    pResponseSize)
    {
        LOG_PROC_ENTRY;
        HRESULT Status = E_FAIL;
        if (!pRequest)
            return E_INVALIDARG;
        CX509OCSPRequest Request(pRequest, requestSize);
        CX509OCSPResponseBuilder ResponseBuilder;
        Status = Request.GetInitStatus();
        if (FAILED(Status))
        {
            FailedResponse(X509OCSPResponseStatus::MALFORMED_REQUEST,
pResponse, pResponseSize);

```

```

        return S_OK;
    }
    LPWSTR Name = L"OCSP";
    ResponseBuilder.SetName(OID_COMMON_NAME, wcslen(OID_COMMON_NAME),
Name, wcslen(Name));
    for (int i = 0; i < Request.GetCountRequest(); ++i)
    {
        CX509Request*                pSingleRequest = NULL;
        CX509SingleResponse*         pSingleResponse = new
CX509SingleResponse();
        Status = Request.GetRequest(i, &pSingleRequest);
        CHECK_HR_FOR_RESPONSE(Status)
        SYSTEMTIME SysTime = { 0 };
        GetSystemTime(&SysTime);
        pSingleResponse->SetCertID(pSingleRequest->GetCertID());

        FILETIME Time = { 0 };
        SystemTimeToFileTime(&SysTime, &Time);
        pSingleResponse->SetThisUpdateDate(&Time);
        SysTime.wYear += 5;
        SystemTimeToFileTime(&SysTime, &Time);
        pSingleResponse->SetNextUpdateDate(new FILETIME(Time));
        pSingleResponse-
>SetStatus(CX509SingleResponse::CERT_STATUS_GOOD);
        Status = ResponseBuilder.AddX509Response(pSingleResponse);
        CHECK_HR_FOR_RESPONSE(Status)
        delete pSingleRequest;
    }
    SYSTEMTIME SysTime = { 0 };
    FILETIME Time = { 0 };
    GetSystemTime(&SysTime);
    SystemTimeToFileTime(&SysTime, &Time);
    ResponseBuilder.SetProducedAtDateTime(Time);
    ResponseBuilder.SetStatus(X509OCSPResponseStatus::SUCCESSFUL);
    CByteBuffer Data;
    Status = g_Signer->Sign(ResponseBuilder, Data);
    CHECK_HR_STATUS_RET1(Status);
    memcpy(pResponse, Data.Data, Data.Size);
    *pResponseSize = Data.Size;
    return S_OK;
}
/// <summary>
/// Генерация ответа на TSP запрос
/// </summary>
/// <param name="pRequest">Данные TSP запроса.</param>
/// <param name="requestSize">Размер данных TSP запроса.</param>
/// <param name="pResponse">Данные TSP ответа.</param>
/// <param name="pResponseSize">Размер TSP ответа.</param>
/// <returns>Статус выполнения операции.</returns>
extern "C" __declspec(dllexport)
HRESULT
TSPResponse(
    __in_bcount(requestSize)          PBYTE pRequest,
    __in                                DWORD requestSize,
    __out_bcount(*pResponseSize) PBYTE pResponse,
    __out                                DWORD*
pResponseSize)
{
    LOG_PROC_ENTRY;
    HRESULT Status = E_FAIL;
    CX509TSPRequest TSPRequest(pRequest, requestSize);
    Status = TSPRequest.GetInitStatus();
    CHECK_HR_STATUS_RET1(Status);

```

```

CX509TSPResponseBuilder *ResponseBuilder = new
CX509TSPResponseBuilder();
    PBYTE                pBytesData = NULL;
    PWCHAR                pCharsData = NULL;
    DWORD                DataSize = 0;
    SYSTEMTIME    SysTime = { 0 };
    FILETIME      Time  = { 0 };
    CByteBuffer    Data;
    INT64 *Nonce = TSPRequest.GetNonce();
    ResponseBuilder->SetNonce(*Nonce);
    ResponseBuilder->SetPKIStatus(PkiStatus::Granted);
    delete Nonce;
    Accuracy Accuracy = { 0, 1, 0 };
    ResponseBuilder->SetAccuracy(Accuracy);
    Status = ResponseBuilder-
>SetDigestAlgorithm(&COid(OID_GOST_HASH_ALG_ID_34_11_12_512,
wcslen(OID_GOST_HASH_ALG_ID_34_11_12_512)));
    CHECK_HR_STATUS_GOTO(Status, Finally);
    Status = TSPRequest.GetHashAlgorithmOid(pCharsData, &DataSize);
    CHECK_HR_STATUS_GOTO(Status, Finally);
    pCharsData = new WCHAR[DataSize];
    Status = TSPRequest.GetHashAlgorithmOid(pCharsData, &DataSize);
    CHECK_HR_STATUS_GOTO(Status, Finally);
    Status = ResponseBuilder->SetHashMessageAlgorithm(pCharsData,
DataSize);
    CHECK_HR_STATUS_GOTO(Status, Finally);
    delete[] pCharsData;
    pCharsData = NULL;
    DataSize = 0;
    Status = TSPRequest.GetPolicyOid(pCharsData, &DataSize);
    CHECK_HR_STATUS_GOTO(Status, Finally);
    if (DataSize)
    {
        pCharsData = new WCHAR[DataSize];
        Status = TSPRequest.GetPolicyOid(pCharsData, &DataSize);
        CHECK_HR_STATUS_GOTO(Status, Finally);
    }
    else
    {
        pCharsData = L"1.2.3.4";
        DataSize = wcslen(pCharsData);
    }
    Status = ResponseBuilder->SetPolicy(pCharsData, DataSize);
    CHECK_HR_STATUS_GOTO(Status, Finally);
    pCharsData = NULL;
    DataSize = 10;
    pBytesData = new BYTE[DataSize];
    memset(pBytesData, 0, DataSize);
    Status = ResponseBuilder->SetSerialNumber(pBytesData, DataSize);
    CHECK_HR_STATUS_GOTO(Status, Finally);
    delete[] pBytesData;
    pBytesData = NULL;
    DataSize = 0;
    Status = TSPRequest.GetHashedMessage(pBytesData, &DataSize);
    CHECK_HR_STATUS_GOTO(Status, Finally);
    pBytesData = new BYTE[DataSize];
    Status = TSPRequest.GetHashedMessage(pBytesData, &DataSize);
    CHECK_HR_STATUS_GOTO(Status, Finally);
    Status = ResponseBuilder->SetHashedMessage(pBytesData, DataSize);
    CHECK_HR_STATUS_GOTO(Status, Finally);
    delete[] pBytesData;
    pBytesData = NULL;
    DataSize = 0;
    IssuerAndSerialNumber *IssuerSerial = new IssuerAndSerialNumber();

```

```

        IssuerSerial->m_SerialNumber = CByteBuffer(10);
        IssuerSerial->m_Issuer = new CX500DistinguishedName();
        IssuerSerial->m_Issuer->AddName(OID_COUNTRY_NAME,
wcslen(OID_COUNTRY_NAME), L"RU", 3);
        ResponseBuilder->SetSubjectIdentifier(IssuerSerial);
        DataSize = 36;
        pBytesData = new BYTE[DataSize];
        memset(pBytesData, 0, DataSize);
        ResponseBuilder->SetCertificateHash(pBytesData, DataSize);
        delete[] pBytesData;
        pBytesData = NULL;
        DataSize = 0;
        GetSystemTime(&SysTime);
        SystemTimeToFileTime(&SysTime, &Time);
        ResponseBuilder->SetGenTime(Time);
        Status = g_Signer->Sign(*ResponseBuilder, Data);
        CHECK_HR_STATUS_GOTO(Status, Finally);
        memcpy(pResponse, Data.Data, Data.Size);
        *pResponseSize = Data.Size;
    Finally:
        delete[] pBytesData;
        delete[] pCharsData;
        delete ResponseBuilder;
        return Status;
}

```

Приложение Д
Результаты нагрузочного тестирования
(рекомендуемое)

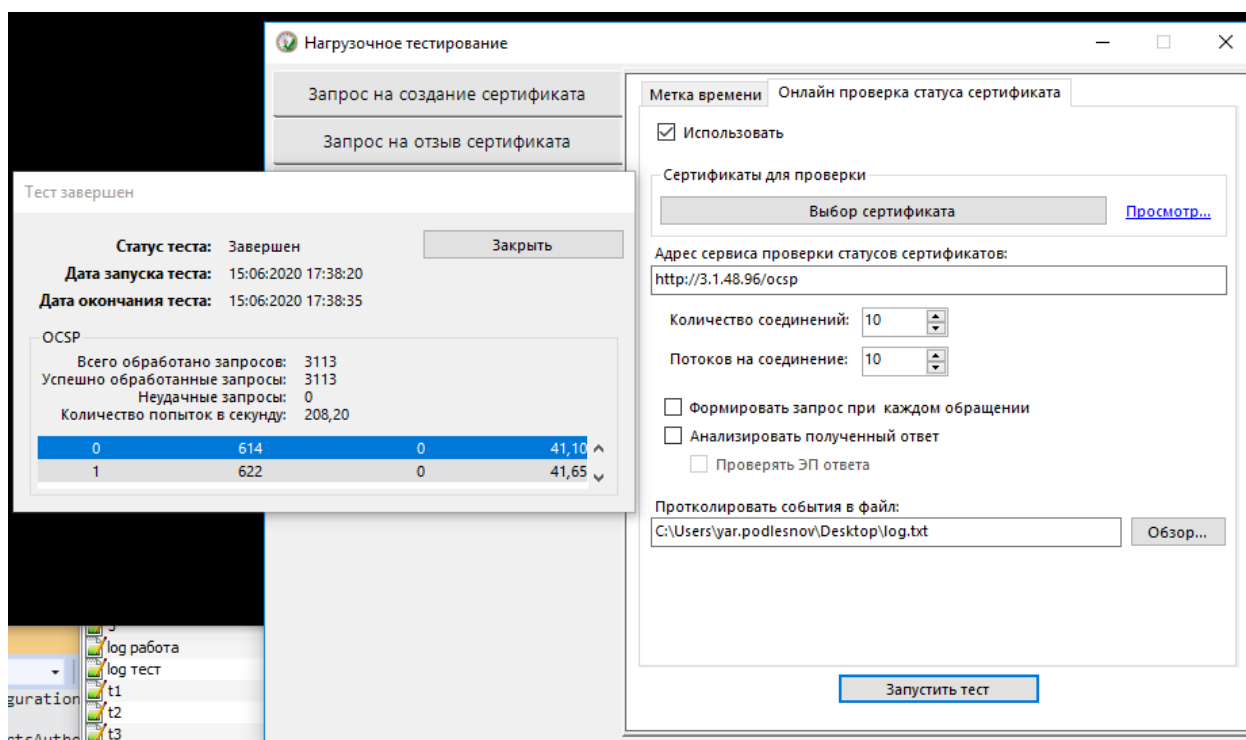


Рисунок Д.1 – Нагрузочное тестирование

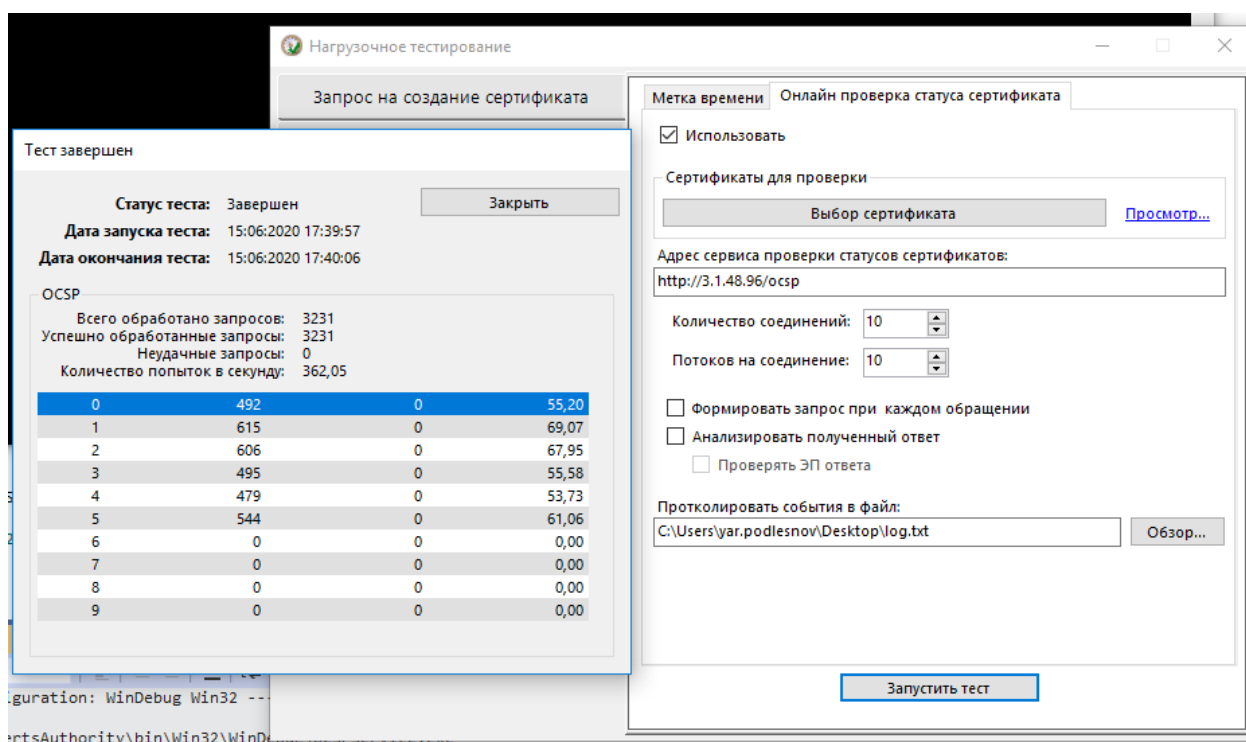


Рисунок Д.2 – Нагрузочное тестирование

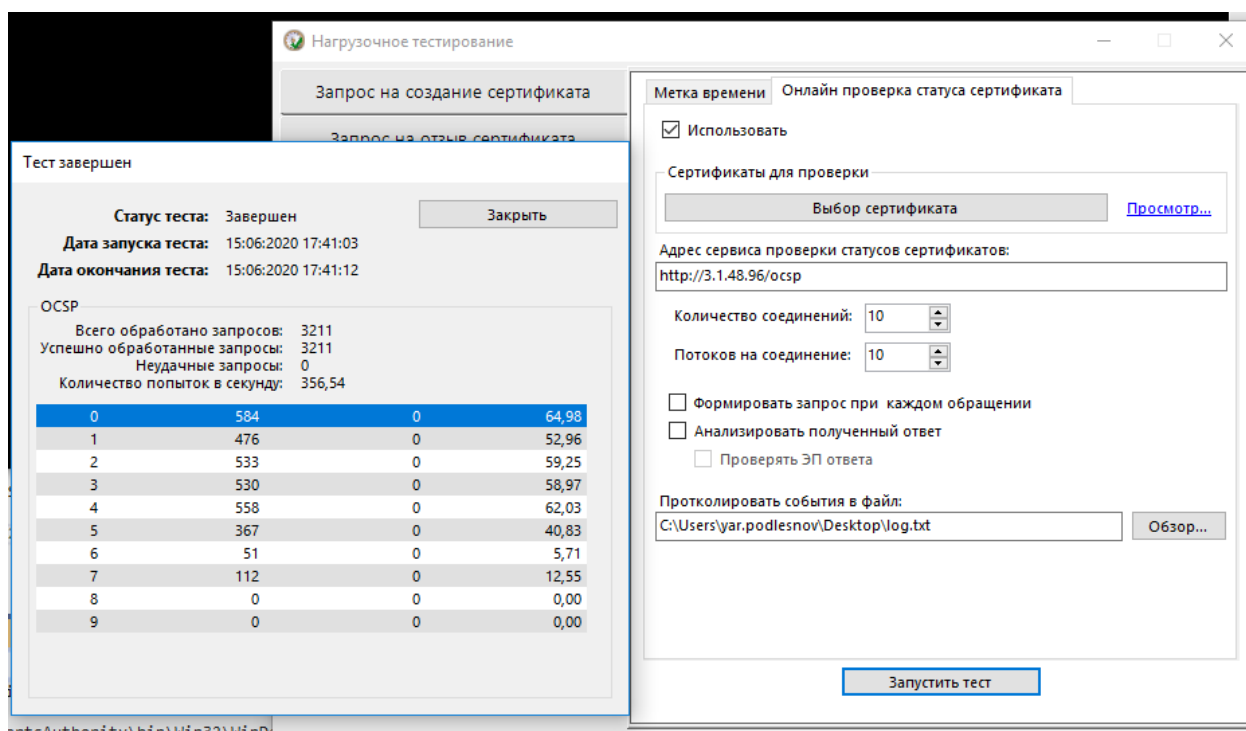


Рисунок Д.3 – Нагрузочное тестирование

Приложение Е
Акт о внедрении
(рекомендуемое)

АКТ
о внедрении результатов
выпускной квалификационной работы
Подлеснова Ярослава Евгеньевича

Программные средства поддержки онлайн-проверки статуса сертификата для операционной системы QR ОС, разработанные техником-программистом Подлесновым Я.Е – сотрудником ООО НТП «Криптософт», в ходе выполнения выпускной квалификационной работы, включено в состав программного комплекса, разрабатываемого ООО НТП «Криптософт».

Заместитель директора

«16» июня 2020 г.

А.В. Козин

