Разработка программного модуля наложения ЭЦП на данные формата xml для компании ООО «МПК ПЛЮС»

СОДЕРЖАНИЕ

[**ВВЕДЕНИЕ** 4](#_Toc135260730)

[**ГЛАВА 1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ** 7](#_Toc135260731)

[1. Введение в ЭЦП 7](#_Toc135260732)

[2. Особенности формата данных XML 12](#_Toc135260733)

[3. Основные положения по интегрированию ЭЦП в XML-документы 18](#_Toc135260734)

[4. Локализация предметной деятельности 22](#_Toc135260735)

[**ГЛАВА 2. ТЕОРЕТИЧЕСКОЕ ОБОСНОВАНИЕ ВЫБРАННОЙ МОДЕЛИ** 25](#_Toc135260736)

[1. Алгоритмизация процесса выработки ключей 25](#_Toc135260737)

[2. Процесс разработки электронной цифровой подписи 29](#_Toc135260738)

[3. Особенности создания сертификата на базе ключа и ЭЦП 38](#_Toc135260739)

[4. Окружение и требования по интеграции ЭЦП в документы XML 39](#_Toc135260740)

[5. Меры по обеспечению информационной безопасности 44](#_Toc135260741)

[**ГЛАВА 3. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ ОБЪЕКТА ИССЛЕДОВАНИЯ** 46](#_Toc135260742)

[1. Описание среды разработки 46](#_Toc135260743)

[1.1. Выбор и описание программных инструментов 46](#_Toc135260744)

[1.2. Обоснование выбора инструментария по разработке 47](#_Toc135260745)

[2. Разработка программного модуля 48](#_Toc135260746)

[2.1. Проектирование архитектуры модуля 48](#_Toc135260747)

[2.2. Реализация пользовательского интерфейса программы 50](#_Toc135260748)

[2.3. Описание кодом функциональных узлов модуля 52](#_Toc135260749)

[**ГЛАВА 4. РЕЗУЛЬТАТЫ РАБОТЫ И ТЕСТИРОВАНИЕ** 57](#_Toc135260750)

[1. Демонстрация и результаты работы программного модуля 57](#_Toc135260751)

[1.1. Кейс «неподписанный документ и создание нового сертификата» 57](#_Toc135260752)

[1.2. Кейс «неподписанный документ и имеющийся сертификат» 58](#_Toc135260753)

[1.3. Кейс «подписанный документ и подходящий сертификат» 59](#_Toc135260754)

[1.4. Кейс «подписанный документ и неподходящий сертификат» 61](#_Toc135260755)

[2. Экономическая эффективность разработанного продукта 62](#_Toc135260756)

[2.1. Определение численности исполнителей 62](#_Toc135260757)

[2.2. Анализ структуры затрат проекта 63](#_Toc135260758)

[2.3. Затраты на приобретение необходимого оборудования и программного обеспечения для реализации проекта 64](#_Toc135260759)

[2.4. Затраты на подготовку кадров и обучение персонала, связанного с использованием электронной подписи в работе 65](#_Toc135260760)

[2.5. Затраты на организацию и проведение пилотных проектов для проверки и тестирования системы 66](#_Toc135260761)

[2.6. Затраты на поддержку и техническое обслуживание системы после ее внедрения 67](#_Toc135260762)

[2.7. Затраты на адаптацию и доработку существующих информационных систем, включая их интеграцию с новой системой электронной подписи 68](#_Toc135260763)

[2.8. Затраты на маркетинговую деятельность, связанную с информированием пользователей о новой системе электронной подписи и ее возможностях 69](#_Toc135260764)

[2.9. Прочие затраты, включающие в себя расходы на юридическую и консультационную поддержку проекта, а также расходы на обеспечение безопасности и конфиденциальности данных 70](#_Toc135260765)

[2.10. Суммарные затраты на реализацию программного проекта 71](#_Toc135260766)

[**ЗАКЛЮЧЕНИЕ** 73](#_Toc135260767)

[**СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ** 75](#_Toc135260768)

[**ПРИЛОЖЕНИЯ** 78](#_Toc135260769)

[Приложение А. Листинг кода программы 78](#_Toc135260770)

# ВВЕДЕНИЕ

В настоящее время в условиях широкого распространения электронного документооборота, обеспечение целостности, аутентичности и конфиденциальности передаваемой информации является одним из наиболее серьезных вопросов для организаций. В связи с этим, электронная цифровая подпись (ЭЦП) становится неотъемлемой частью процесса электронного документооборота и используется для обеспечения подлинности и целостности электронных документов.

**Актуальность** проблемы наложения ЭЦП на XML-документы обусловлена необходимостью обеспечения безопасности и защиты информации в условиях современного электронного документооборота. В связи с увеличением объема и разнообразия передаваемых электронных документов, а также с ростом количества угроз информационной безопасности, становится все более важным обеспечение надежности и целостности передаваемой информации.

В данной дипломной работе рассматривается разработка программного модуля для наложения ЭП на XML-документы для компании ООО "МПК ПЛЮС". Данная организация занимается предоставлением услуг с использованием электронного делопроизводства и документооборота. Поэтому для обеспечения безопасности передаваемой информации необходимо наложение ЭП на XML-документы.

**Целью** данной работы является разработка программного модуля, который позволит автоматизировать процесс наложения ЭП на XML-документы и обеспечить эффективность и безопасность процесса электронного документооборота в компании ООО "МПК ПЛЮС". Для достижения данной цели были поставлены следующие задачи: изучение основных принципов работы с ЭП и формата XML, разработка программного модуля на удобном в рамках решаемой задачи языке программирования с использованием криптографических библиотек, тестирование и оптимизация разработанного программного модуля.

В ходе дипломной работы были поставлены следующие **задачи**:

* Изучение основ криптографии и электронной подписи, а также формата XML.
* Разработка программного модуля на языке программирования для наложения ЭП на XML-документы.
* Определение оптимального алгоритма для выполнения задачи наложения ЭП на XML-документы.
* Тестирование и оптимизация разработанного программного модуля.
* Оценка эффективности разработанного программного модуля и его применимости для использования компанией ООО "МПК ПЛЮС" в процессе электронного документооборота.

Решение этих задач позволило разработать программный модуль, который может быть использован для обеспечения безопасности и целостности передаваемой информации в электронном документообороте компании ООО "МПК ПЛЮС".

В процессе работы были использованы различные методы и технологии, такие как анализ литературы, проектирование и разработка программного кода, тестирование и оптимизация разработанного программного модуля.

Результатом данной работы является разработанный программный модуль для наложения ЭП на XML-документы, который позволяет автоматизировать процесс обеспечения безопасности передаваемой информации в компании ООО "МПК ПЛЮС".

**Объектом исследования** являются процессы электронного документооборота, а предметом исследования – разработка программного модуля для наложения ЭП на XML-документы.

В ходе исследования были использованы следующие **методологические основы**: метод анализа литературных источников для изучения основных принципов работы с ЭП и форматом XML, метод проектирования программного кода на языке программирования с использованием криптографических библиотек, метод тестирования и оптимизации разработанного программного модуля.

**Научная новизна** данной работы заключается в разработке программного модуля для наложения ЭП на XML-документы, а также в определении оптимального алгоритма для выполнения данной задачи. **Практическая значимость** исследования заключается в возможности использования разработанного программного модуля компанией ООО "МПК ПЛЮС" для обеспечения безопасности и целостности передаваемой информации в электронном документообороте. Результаты работы могут быть использованы в других организациях, занимающихся электронным документооборотом, для решения аналогичных задач.

# ГЛАВА 1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

1. **Введение в ЭЦП**

ЭЦП (Электронная Цифровая Подпись) – это криптографический механизм, используемый для подтверждения подлинности электронных документов, сообщений и транзакций.

ЭЦП является цифровым аналогом обычной подписи на бумажном носителе (документе) и позволяет идентифицировать отправителя сообщения, убедиться в целостности и неизменности содержимого документа, а также подтвердить факт его отправки и получения.

Для создания ЭЦП используются криптографические алгоритмы, которые генерируют уникальный цифровой код, связанный с определенным документом или сообщением. Цифровая подпись широко применяется в банковской, юридической и государственной сферах, а также в электронной коммерции и интернет-сервисах.

Электронная цифровая подпись была разработана в 1976 году Ральфом Мерклом и Мартином Хеллманом. Они предложили использовать криптографические методы для создания цифровой подписи, которая была бы электронным аналогом обычной подписи на бумажном документе. В 1980 году Ривест, Шамир и Адлеман разработали алгоритм RSA, который стал основным для создания ЭЦП. Этот алгоритм позволяет создавать открытый и закрытый ключи, которые используются для шифрования информации.

В 1991 году бы создан стандарт X.509, который определяет формат и правила использования ЭЦП. Этот стандарт стал основой для создания других стандартов, таких как PGP (Pretty Good Privacy) и S/MIME (Secure/Multipurpose Internet Mail Extensions).

С развитием интернета и электронной коммерции ЭЦП стала все более популярной. В 2000 году был принят Федеральный Закон «Об электронной подписи», который установил правовые основы для использования цифровой подписи в России. Сегодня ЭЦП используется в различных сферах. Она должна обеспечивать безопасность и надежность при передаче и хранении электронной информации.

Законодательная сила электронной подписи может различаться в разных странах. В некоторых странах она имеет полную юридическую силу, в то время как в других она может быть ограничена или вовсе не признаваться.

Например, в Европейском Союзе ЭЦП признается как юридически действительная в большинстве стран-членов. В США и Канаде цифровая подпись также имеет юридическую силу, но существуют различия в требованиях к ее использованию в разных штатах и провинциях.

В России ЭЦП имеет полную юридическую силу и используется в различных областях, включая государственные закупки и электронную подачу документов в государственные органы.

В Китае цифровая подпись также признается, но ее использование ограничено определенными правилами и требованиями.

Таким образом, законодательная сила электронной подписи может различаться в зависимости от страны и ее законодательства. Важно предварительно ознакомиться с правилами и требованиями к обороту, прежде чем использовать ЭЦП в каких-либо юридически значимых документах.

Как говорилось ранее, ЭЦП – это способ обеспечения подлинности и целостности электронных документов, и важно понимать особенности документооборота и делопроизводства, имея дело с зарубежными компаниями, например, при аутсорсинге.

В международных отношениях применение ЭЦП имеет несколько особенностей:

* Правовое признание: в различных странах существуют разные законы и нормы, регулирующие применение ЭЦП. В некоторых странах она полностью признается в качестве аналога собственноручной подписи, в других – только в определенных случаях;
* Технические ограничения: для использования ЭЦП требуется наличие специальных программ и устройств, которые могут быть недоступны в некоторых странах;
* Культурные различия: в некоторых странах письменная форма документов по-прежнему более распространена. Это может повлиять на приемлемость использования цифровой подписи в международных отношениях.
* Безопасность: использование ЭЦП позволяет обеспечить безопасность электронных документов, что особенно важно в международных отношениях, где информация может быть подвергнута кибератакам;
* Эффективность: применение электронной подписи может значительно ускорить процесс обмена документами, сократить затраты на пересылку бумажных документов и их хранение.

В целом, применение ЭЦП в международных отношениях является важным инструментом для обеспечения безопасности и эффективности процесса обмена документами. Однако, необходимо учитывать правовые и технические особенности, а также культурные различия между странами.

Для анализа и полного рассмотрения вопроса о ЭЦП необходимо рассмотреть преимущества и недостатки данной технологии обеспечения целостности данных.

Преимущества цифровой подписи:

* Гарантия подлинности: ЭЦП позволяет установить, что документ не был изменен после подписания;
* Защита от подделок. ЭЦП предотвращает возможность подделки документов;
* Ускорение процесса. Цифровая подпись позволяет быстро подписывать документы и передавать их по цифровым каналам связи;
* Сокращение затрат: рассматриваемая технология позволяет не брать бумажные документы, которые нужно подписать, в физических офисах, что сэкономит время и деньги.
* Удобство использования: электронная подпись позволяет подписывать документы без необходимости печатать их на бумаге.
* Сохранность документов: ЭЦП позволяет хранить документы в электронном виде, что обеспечивает их сохранность.

Если говорить о недостатках, то можно выделить следующие:

* Сложность использования: некоторые пользователи могут испытывать трудности при использовании ЭЦП, особенно если они не имеют опыта работы с технологией;
* Надежность: многие эксперты считают, что ЭЦП не всегда является надежным методом подписи документов;
* Зависимость от технических средств. Для использования цифровой подписи необходимо иметь специальное программное обеспечение и оборудование, что может быть недоступно для некоторых пользователей;
* Ограничения в использовании: некоторые страны могут иметь ограничения в использовании ЭЦП для определенных видов документов;
* Необходимость поддержки со стороны государства: для обеспечения надежности и безопасности использования электронной подписи необходима поддержка со стороны государства.

При работе с рассматриваемой технологией со стороны технической администрации необходимо знать потенциальные угрозы целостности документов, заверенных цифровой подписью и возможные векторы атак.

Так, например, самые распространенные векторы выделяют следующими:

* Взлом учетной записи пользователя: кража учетных данных пользователя может привести к несанкционированному доступу к его документам, в том числе и тем, что защищены ЭЦП.
* Модификация данных: злоумышленник может изменить содержимое документа, не нарушая его целостность. Это может привести к неправильной интерпретации его содержимого и, как следствие, к неправильному принятию решений.
* Внедрение вредоносного кода: злоумышленник может внедрить в документ вредоносный код, который может использоваться для кражи данных, управления устройством или даже привести к полной блокировке системы.
* Социальная инженерия: преступник может попытаться обмануть пользователя и получить доступ к его документам. Например, отправив ложное уведомление об обновлении ПО и просить пользователя авторизоваться для установки обновления.
* Атака на сервер ЭЦП: Серверы цифровых подписей могут стать объектом атак со стороны злоумышленников. В результате взлома сервера ЭЦП, злоумышленники могут получить доступ к секретным ключам и перехватить все подписи, сделанные на этом сервере.

Для защиты данных с электронной подписи необходимо использовать комплексный подход, который включает в себя защиту учетных записей пользователей, использование антивирусного ПО, обучение пользователей основам информационной безопасности и многое другое. Кроме того, необходимо следить за обновлением ПО и использовать только проверенные и сертифицированные программы для работы с ЭЦП.

Есть много схожих технологий, таких как ЭЦП. Например, ЦВЗ (Цифровой Водяной Знак). Необходимо знать отличие двух методов защиты целостности для точного определения проблематики работы.

ЭЦП (электронная цифровая подпись) и ЦВЗ (цифровая визуальная подпись) – это два разных способа обеспечения подлинности электронных документов и сообщений.

Основное отличие между ними заключается в том, что ЦВЗ – это вид подписи, который наносится на документ визуально, то есть пользователь может увидеть ее на экране компьютера. ЭЦП же – это цифровая подпись, которая генерируется с помощью специального алгоритма и применяется к файлу или сообщению.

Еще одно отличие между ЭЦП и ЦВЗ заключается в том, что ЭЦП является более надежным и безопасным методом, так как она использует криптографические ключи и алгоритмы, которые обеспечивают высокий уровень защиты от подделки или изменения данных. ЦВЗ же может быть легко подделана или изменена, поскольку она видна на экране компьютера и может быть скопирована или изменена с помощью графических редакторов.  
Таким образом, ЭЦП является более надежным и безопасным способом обеспечения подлинности электронных документов и сообщений, чем ЦВЗ.

1. **Особенности формата данных XML**

XML (от англ. eXtensible Markup Language, расширяемый язык разметки) – это формат данных, который используется для хранения и передачи информации в виде текстовых документов. XML состоит из тегов, в которых заключается информация в виде пар "имя/значение". XML позволяет структурировать данные, описывать их содержание и связи между ними, что делает его удобным для обмена информацией между приложениями и платформами. XML может быть использован для описания различных типов данных, включая текст, числа, даты, изображения и т.д.

Формат данных XML был разработан для обмена данными между различными приложениями и платформами. XML используется для хранения, передачи и обработки структурированных данных в формате, который может быть легко прочитан и понят другими программами. Рассматриваемый формат позволяет создавать собственные теги и определять структуру данных, что делает его более гибким и мощным, чем HTML. XML широко используется во многих областях, включая веб-разработку, базы данных, научные исследования, обмен данными между приложениями и многое другое.

Данный формат играет большую роль в АСС (Автоматизированных Сетях Связи). Он позволяет описывать структуру данных и обеспечивает возможность их обмена между различными приложениями и устройствами.  
XML используется для создания структурированных документов, которые могут содержать любую информацию, от текста до графики и мультимедиа. Он используется для создания файлов конфигурации, для передачи данных между веб-серверами и клиентами, для обмена данными между приложениями и многое другое.

XML также используется в современных протоколах передачи данных, таких как SOAP, REST, XML-RPC и других. Он позволяет стандартизировать формат передаваемых данных и обеспечивает возможность их интерпретации на разных устройствах и платформах.

Таким образом, XML играет важную роль в сетях связи и IT-инфраструктурах, обеспечивая стандартизацию формата данных и возможность обмена информацией между различными устройствами и приложениями.

Если говорить о сравнении с другими типизированными форматами, схожими по семантике с XML, то можно выделить ряд преимуществ:

* может быть использован на любой платформе и в различных языках программирования;
* легко понятен для человека и легко читаем;
* может быть использован для хранения структурированных данных;
* является расширяемым форматом, что означает, что вы можете создавать свои собственные теги и атрибуты;
* имеет широкое применение в различных областях, таких как веб-разработка, базы данных, наука, медицина и т.д.
* поддерживает множество различных кодировок и символов, что делает его удобным для использования в различных языках и культурах;
* поддерживает механизмы проверки правописания и синтаксического анализа, что позволяет обнаруживать и устранять ошибки в данных.
* позволяет создавать семантические метаданные, которые могут быть использованы для описания содержимого и структуры данных;
* может быть использован для обмена данными между различными приложениями и системами, что делает его удобным для интеграции приложений и автоматизации бизнес-процессов.

Наряду с преимуществами также ярко выражены недостатки в рамках использования формата:

* Неэффективность передачи данных: XML может занимать большой объем памяти, что делает его неэффективным для передачи больших объемов данных. Это может привести к снижению производительности и задержкам в обработке данных;
* Ограниченная поддержка: формат не поддерживает все возможности, которые могут быть полезными для определенных приложений. Например, XML не поддерживает встроенные функции математических вычислений или операции с базами данных;
* Сложность чтения: XML-документы могут быть сложными для чтения и понимания для пользователей, не знакомых с этим языком разметки. Это может привести к ошибкам при обработке данных и ошибочным выводам;
* Низкая производительность: XML может быть медленным в обработке данных, особенно при работе с большими объемами данных. Это может привести к проблемам с производительностью и снижению эффективности работы;
* Неэффективность при хранении данных: анализируемый формат может быть неэффективным при хранении больших объемов данных, так как он может занимать больше места, чем другие форматы хранения данных, такие как JSON или CSV.

Как уже известно, XML – это формат для хранения и передачи данных, который используется во многих приложениях и сервисах. Как и любой другой формат данных, XML может стать объектом атак со стороны злоумышленников.

Ниже можно произвести обзор на возможные угрозы в рамках ИБ, которые могут возникнуть при использовании XML:

1. Внедрение вредоносного кода: злоумышленники могут добавить вредоносный код в файлы XML, чтобы получить доступ к системе или украсть конфиденциальные данные.

2. Несанкционированный доступ: если файлы XML не защищены паролем или другими методами аутентификации, злоумышленники могут получить несанкционированный доступ к данным.

3. SQL-инъекции: если XML используется для передачи данных в базу данных, злоумышленники могут использовать SQL-инъекции, чтобы получить доступ к базе данных или изменить ее содержимое.

4. Переполнение буфера: если приложение не проверяет размер файлов XML, злоумышленники могут создать файлы большого размера, что может привести к переполнению буфера и отказу в обслуживании.

Для предотвращения этих угроз необходимо принимать меры по обеспечению безопасности XML-файлов. Некоторые из этих мер могут включать в себя:

1. Использование схемы XML: схема XML позволяет определить допустимые значения элементов и атрибутов, что помогает предотвратить внедрение вредоносного кода.

2. Аутентификация: файлы XML должны быть защищены паролем или другими методами аутентификации, чтобы предотвратить несанкционированный доступ.

3. Фильтрация ввода: приложение должно проверять входные данные на наличие вредоносного кода и недопустимых символов.

4. Ограничение размера файлов: приложение должно проверять размер файлов XML и отклонять файлы, которые превышают допустимый размер.

Анализируемый формат широко используется в делопроизводстве и документообороте благодаря своей гибкости, масштабируемости и универсальности. Применение технологии в таких рамках играет много-направленную роль:

* Обмен электронными документами: формат XML позволяет создавать электронные документы, которые могут быть легко обменены между различными системами и приложениями. Например, с помощью формата XML можно создавать электронные документы для отправки по электронной почте, загрузки на веб-сайт или обмена между системами управления документами;
* Хранение и обработка данных: формат может использоваться для хранения и обработки различных типов данных, включая текстовые, числовые, даты и изображения. Например, формат XML может использоваться для создания базы данных с информацией о клиентах, проектах или задачах;
* Создание и форматирование документов: тип данных может использоваться для создания документов, таких как отчеты, планы, презентации и другие. С помощью формата XML можно создавать документы с различными стилями форматирования, шрифтами, таблицами и другими элементами;
* Обмен данными между различными приложениями: XML позволяет обмениваться данными между различными приложениями, такими как системы управления проектами, системы управления контентом или системы управления ресурсами предприятия. Это позволяет упростить процесс обмена данными и улучшить эффективность работы;
* Создание электронных формуляров: формат XML может использоваться для создания электронных формуляров, которые могут быть заполнены и отправлены в электронном виде. Это упрощает процесс заполнения и обработки формуляров, что может быть особенно полезно в случае большого количества формуляров, которые необходимо обработать.

Таким образом, формат XML является важным инструментом в рамках делопроизводства и документооборота, который позволяет упростить процесс обмена данными, создания и форматирования документов, а также обработки и хранения данных.

1. **Основные положения по интегрированию ЭЦП в XML-документы**

В настоящее время электронный документооборот широко распространен и с каждым годом становится все более популярным. Однако, вместе с возможностями, которые он предоставляет, возникают и новые проблемы, связанные с защитой информации от несанкционированного доступа и подделки. Один из надежных инструментов защиты электронной информации является электронная цифровая подпись (ЭЦП).

Однако, для использования ЭЦП в электронном документообороте необходимо обеспечить ее интеграцию с форматом представления данных, который используется для электронных документов. XML-документы являются одним из самых распространенных форматов, которые используются в электронном документообороте. Интеграция ЭЦП в XML-документы представляет собой сложную задачу, которая требует соблюдения определенных положений и стандартов, чтобы обеспечить надежность, целостность и подлинность подписанных документов.

Основные положения по интеграции ЭЦП в XML-документы:

1. Формат подписи. Для интеграции ЭЦП в XML-документы используется формат XML-DSig. Этот формат позволяет установить подлинность документа, подтвердить его целостность и удостоверить личность подписавшего.
2. Элементы XML-документа, подлежащие подписанию. XML-документ состоит из множества элементов, но не все они могут быть подписаны. Подписываются только те элементы, которые содержат информацию, изменение которой может повлиять на содержание документа. Обычно подписываются корневой элемент документа и все элементы, содержащие информацию, изменение которой важно для документа.
3. Алгоритмы шифрования и хеширования. Для защиты данных от несанкционированного доступа используются алгоритмы шифрования. Для защиты от изменения информации используются алгоритмы хеширования. Для интеграции ЭЦП в XML-документы обычно используются алгоритмы SHA-256 или SHA-512 для хеширования и RSA для шифрования.
4. Удостоверение подписывающего. Для удостоверения подписывающего используется сертификат открытого ключа. Сертификат содержит информацию об открытом ключе и удостоверяет личность владельца ключа. Обычно сертификаты выдаются удостоверяющими центрами.
5. Проверка подписи. Для проверки подписи используется открытый ключ, который хранится в сертификате подписывающего. При проверке подписи алгоритм хеширования вычисляет хеш-значение подписываемых элементов, которое сравнивается с хеш-значением, содержащимся в подписи. Если хеш-значения совпадают, то подпись считается действительной.
6. Интеграция с XML-схемами. XML-схемы описывают структуру и содержание XML-документа. Интеграция ЭЦП с XML-схемами позволяет убедиться, что подписываемый документ соответствует определенной схеме. Это повышает надежность подписи и обеспечивает защиту от атак типа XML-инъекции.
7. Использование специализированных библиотек и программных средств. Для интеграции ЭЦП в XML-документы используются специализированные библиотеки и программные средства. Они позволяют генерировать подписи, проверять их действительность, а также подписывать и верифицировать XML-документы, соответствующие определенной XML-схеме.
8. Поддержка стандарта XML Advanced Electronic Signatures (XAdES). Стандарт XAdES был разработан для интеграции ЭЦП в XML-документы и обеспечивает дополнительные функции, такие как указание даты и времени подписи, список сертификатов открытых ключей и дополнительные данные о подписывающем.
9. Соблюдение требований законодательства. Использование ЭЦП в XML-документах подчинено определенным правовым требованиям, которые устанавливаются в соответствии с национальным законодательством. Например, в России использование ЭЦП регулируется Федеральным законом "Об электронной подписи".

ЭЦП (электронная цифровая подпись) – это средство аутентификации электронных документов, которое обеспечивает уверенность в подлинности информации, ее целостности и невозможности ее изменения без разрешения владельца ЭЦП. Интеграция ЭЦП в XML-документы является важным шагом в сфере электронного документооборота и информационных технологий в целом.

Применение ЭЦП в XML-документах имеет широкий спектр применений в различных сферах, включая правительственные организации, финансовые институты, здравоохранение, телекоммуникации, логистику и многие другие. В правительственных организациях использование ЭЦП в XML-документах позволяет ускорить процесс обработки документов и повысить эффективность работы государственных служб. В финансовых институтах интеграция ЭЦП в XML-документы обеспечивает безопасность и подлинность финансовых транзакций. В здравоохранении ЭЦП позволяет обеспечить конфиденциальность медицинских данных пациентов. В логистике использование ЭЦП в XML-документах позволяет упростить и ускорить процессы доставки и отслеживания грузов.

Преимущества интеграции ЭЦП в XML-документы очевидны. Они включают в себя повышение безопасности электронного документооборота, упрощение процесса обработки и передачи документов, ускорение процессов бизнеса, сокращение времени на подписание документов и исключение возможности подделки электронных документов.

Недостатки интеграции ЭЦП в XML-документы также существуют. Одним из них является высокая стоимость системы интеграции ЭЦП, что может быть финансово затратным для малых и средних предприятий. Еще одним недостатком может быть необходимость использования специального программного обеспечения для работы с электронными документами.

В целом, интеграция ЭЦП в XML-документы является важным шагом в направлении улучшения безопасности и эффективности электронного документооборота, а также содействует переходу на более экологически чистый и более эффективный процесс обработки документов. Несмотря на некоторые недостатки, интеграция ЭЦП в XML-документы оправдывает свою ценность и принимается бизнес-сообществом и правительственными организациями во всем мире.

Кроме того, интеграция ЭЦП в XML-документы помогает улучшить качество обработки и хранения документов, уменьшить время на их обработку, а также сократить количество ошибок, связанных с человеческим фактором.

Однако, при использовании ЭЦП в XML-документах также могут возникать проблемы совместимости, которые могут создать некоторые неудобства для пользователей. Также существует вероятность нарушения безопасности при передаче ключа ЭЦП, что может привести к возможности подделки документов.

Подводя итог, интеграция ЭЦП в XML-документы является важным инструментом, который помогает повысить эффективность и безопасность электронного документооборота. Несмотря на некоторые недостатки, преимущества данной технологии превышают ее недостатки, и она продолжает широко применяться в различных сферах, где требуется безопасный и эффективный обмен электронными документами.

1. **Локализация предметной деятельности**

Локализация предметной области проекта включает в себя выбор метода встраивания ЭЦП в документы формата XML. В данном проекте был выбран метод, основанный на использовании стандарта XML-DSig.

XML-DSig - это стандарт, который описывает форматы данных и процедуры для создания и проверки цифровых подписей в документах формата XML. Для встраивания ЭЦП в документы формата XML будет разработан программный модуль, который будет использовать стандарт XML-DSig.

Процесс встраивания ЭЦП в документы формата XML будет проходить по следующим этапам:

1. Генерация ключевой пары. Для начала работы необходимо сгенерировать ключевую пару, состоящую из закрытого и открытого ключа.
2. Создание подписи. При создании подписи, встраиваемой в документ формата XML, будет использоваться закрытый ключ. Подпись будет создана с использованием стандарта XML-DSig.
3. Встраивание подписи в документ формата XML. Подпись будет встроена в документ в соответствии со стандартом XML-DSig.
4. Проверка подписи. При проверке подписи, используется открытый ключ. Если подпись была создана с использованием правильного закрытого ключа, подпись будет действительна и проверка пройдет успешно.

Достижение цели работы позволит встроить ЭЦП в документы формата XML с использованием стандарта XML-DSig, а также осуществлять проверку подписи встроенной в документ формата XML.

Разработка программного модуля для встраивания ЭЦП в документы формата XML имеет большую значимость в современном мире, где все больше информации передается и хранится в электронном виде. ЭЦП позволяет удостоверить подлинность документа и подписавшего его лица, что является особенно важным для юридически значимых документов, таких как договоры, акты, отчеты и т.д.

Проблемой, которую решает разработка программного модуля для встраивания ЭЦП в документы формата XML, является возможность подмены документа в процессе передачи или хранения. Это может привести к непредсказуемым последствиям, например, к изменению условий договора или сбоям в работе системы управления документами.

С помощью разработанного программного модуля для встраивания ЭЦП в документы формата XML, возможно решить эту проблему, обеспечивая аутентификацию и целостность документа. Кроме того, програмный модуль будет обладать следующими преимуществами:

1. Возможность использовать разные алгоритмы хеширования и шифрования в зависимости от требований заказчика.
2. Универсальность в работе с разными типами документов формата XML.
3. Высокая скорость работы и надежность при встраивании ЭЦП в документы.

Таким образом, разработка программного модуля для встраивания ЭЦП в документы формата XML является актуальной и востребованной задачей в современном информационном обществе.

# ГЛАВА 2. ТЕОРЕТИЧЕСКОЕ ОБОСНОВАНИЕ ВЫБРАННОЙ МОДЕЛИ

1. **Алгоритмизация процесса выработки ключей**

Для осуществления процесса наложения ЭЦП на электронный документ необходимо, первым делом, сформировать открытый и закрытый ключи.

Открытый ключ (public key) - это один из двух ключей, используемых в криптографической системе с открытым ключом (Public Key Infrastructure, PKI). Открытый ключ используется для защиты информации и создания Электронной цифровой подписи (ЭЦП).

Открытый ключ - это числовое значение, которое может быть опубликовано или передано другим пользователям безопасно. Он используется для шифрования данных или проверки ЭЦП, созданных с помощью соответствующего закрытого ключа. Открытый ключ может быть распространен широко, и все, кто имеют доступ к нему, могут использовать его для проверки ЭЦП или шифрования сообщений.

При создании ЭЦП, открытый ключ получателя используется для проверки подписи документа, которая была создана с помощью соответствующего закрытого ключа отправителя. Если подпись проверена, то можно быть уверенным в том, что документ не был изменен после того, как он был подписан.

Закрытый ключ (private key) - это один из двух ключей, используемых в криптографической системе с открытым ключом (Public Key Infrastructure, PKI). Закрытый ключ является секретным и должен быть известен только владельцу ключа.

Закрытый ключ используется для создания Электронной цифровой подписи (ЭЦП) или расшифровки зашифрованных данных, которые были зашифрованы с помощью соответствующего открытого ключа. Таким образом, закрытый ключ используется для обеспечения конфиденциальности информации.

При создании ЭЦП, закрытый ключ отправителя используется для подписи документа, а затем получателю отправляется подписанный документ вместе с открытым ключом отправителя. Получатель может использовать открытый ключ, чтобы проверить подпись и убедиться в том, что документ был подписан именно отправителем.

Закрытый ключ должен храниться в надежном месте и не должен быть доступен другим пользователям без разрешения владельца ключа. Как правило, закрытый ключ зашифрован с использованием пароля или других методов аутентификации, чтобы предотвратить несанкционированный доступ к нему.



Рисунок 1 – Обобщенная структурная схема формирования ключей.

Основными алгоритмами выработки ключей для ЭЦП являются алгоритмы с открытым и закрытым ключами, такие как ГОСТ Р 34.10, RSA, ECC.

ГОСТ Р 34.10 – это алгоритм, который используется в России для создания криптографических ключей и подписей. Он основан на дискретном логарифмировании в конечном поле. Генерация ключей происходит следующим образом:

* Генерируется случайное число (приватный ключ) в заданном диапазоне.
* На основе приватного ключа вычисляется публичный ключ.

Генерация ключей с помощью алгоритма RSA происходит следующим образом:

* Генерируются два больших простых числа p и q.
* Вычисляется их произведение n=p\*q, которое называется модулем RSA.
* Вычисляется функция Эйлера от числа n: φ(n)=(p-1)(q-1).
* Генерируется случайное число d, которое является приватным ключом и удовлетворяет условию gcd(d, φ(n))=1, где gcd обозначает наибольший общий делитель.
* Вычисляется число e, которое является публичным ключом и удовлетворяет условию d\*e ≡ 1 (mod φ(n)).

Алгоритм ECC – это криптографическое решение, основанное на эллиптических кривых, который используется для создания криптографических ключей и подписей. Генерация ключей ECC происходит следующим образом:

* Выбирается эллиптическая кривая, над которой будет проводиться работа.
* Генерируется случайная точка на этой кривой (приватный ключ).
* Вычисляется публичный ключ, который является результатом умножения приватного ключа на базовую точку кривой.

Для алгоритмов генерации ключей ГОСТ, RSA и ECC необходимо провести сравнительный анализ и выявить слабые и сильные стороны каждого математического решения.

ГОСТ обладает следующими преимуществами:

* имеет хорошую устойчивость к криптоанализу;
* разработан исключительно для использования в России и поэтому широко используется в этой стране;
* использует более длинные ключи, что делает его более надежным.

Недостатки рассматриваемой технологии:

* менее распространен в мире, чем RSA и ECC;
* требует больших вычислительных ресурсов для генерации ключей, поэтому его использование может быть затруднено на устройствах с ограниченными вычислительными мощностями.

Если подвергнуть анализу алгоритм RSA, то можно выявить следующие плюсы:

* RSA является одним из самых распространенных алгоритмов для генерации ключей;
* RSA имеет хорошую поддержку в различных криптографических библиотеках.

Однако недостатками данного решения являются следующие факты:

* RSA менее устойчив к криптоанализу, чем некоторые другие алгоритмы, такие как ГОСТ и ECC;
* RSA требует более длинные ключи для достижения сравнимого уровня безопасности с другими алгоритмами.

Преимущества ЕСС заключаются в следующих положениях:

* Алгоритм использует более короткие ключи, что делает его более эффективным и требовательным к вычислительным ресурсам;
* Алгоритм обеспечивает сравнимый уровень безопасности с RSA, используя короче ключи.

Если проанализировать математическую технологию на предмет недостатков в реализации и использовании, то следующие суждения опишут всю проблематику данного алгоритма:

* ECC менее распространен, чем RSA;
* ECC может быть уязвим к некоторым атакам на кривые, которые необходимо учитывать при выборе соответствующей кривой.

1. **Процесс разработки электронной цифровой подписи**

Для общего случая алгоритм создания ЭЦП можно описать следующим образом:

1. Генерация ключевой пары: генерируются открытый и закрытый ключи с помощью алгоритма генерации ключей, например, RSA, DSA или ECDSA.
2. Выбор хэш-функции: выбирается хэш-функция, например, SHA-256, которая используется для вычисления дайджеста сообщения.
3. Вычисление дайджеста сообщения: сообщение хэшируется с помощью выбранной хэш-функции, чтобы получить его дайджест.
4. Шифрование дайджеста: дайджест сообщения шифруется закрытым ключом отправителя, чтобы создать ЭЦП.
5. Отправка сообщения и ЭЦП: сообщение и ЭЦП отправляются получателю.
6. Проверка ЭЦП: получатель расшифровывает ЭЦП с помощью открытого ключа отправителя, чтобы получить дайджест сообщения. Затем он вычисляет дайджест сообщения из самого сообщения с помощью той же хэш-функции, что и отправитель, и сравнивает его с полученным дайджестом из расшифрованной ЭЦП. Если дайджесты совпадают, то ЭЦП действительна.

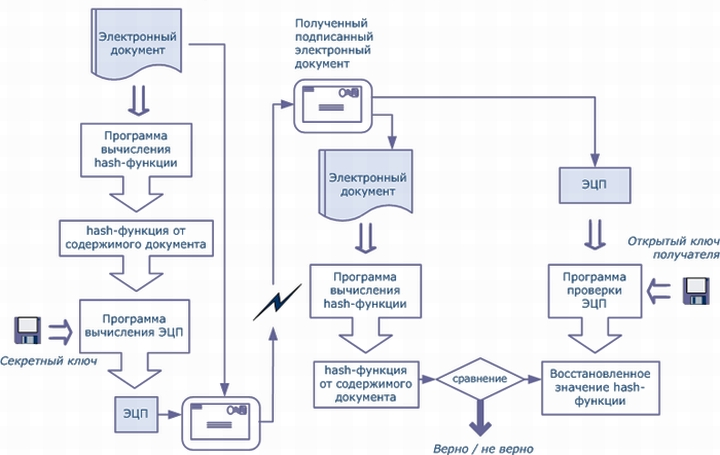


Рисунок 2 – Обобщенная схема формирования и проверки ЭЦП.

Дайджест - это результат работы алгоритма хеширования, представляющий собой фиксированную длину данных, которые сжимаются в процессе хеширования. Дайджест используется для проверки целостности данных, а также в криптографии для создания ЭЦП.

SHA (Secure Hash Algorithm) - это семейство криптографических алгоритмов хеширования, разработанных Национальным институтом стандартов и технологий США (NIST). Алгоритмы SHA принимают на вход произвольный блок данных и выдают на выходе фиксированный дайджест заданной длины. SHA является одним из наиболее распространенных алгоритмов хеширования и используется во многих приложениях, включая протоколы SSL/TLS, IPsec, PGP и многие другие.

Алгоритм хеширования SHA работает следующим образом:

1. Инициализация. Начальное значение дайджеста инициализируется заданным значением.
2. Предварительная обработка. Входные данные разбиваются на блоки фиксированного размера, после чего производится их предварительная обработка. В этом процессе к каждому блоку применяются определенные операции: расширение, перестановка и примешивание (эти операции зависят от конкретного варианта алгоритма SHA).
3. Основной цикл. После предварительной обработки выполняется основной цикл, в котором к каждому блоку данных применяется функция сжатия. Функция сжатия состоит из нескольких этапов, включая применение нескольких логических операций, арифметических действий и операций сдвига.
4. Формирование дайджеста. После прохождения всех блоков данных формируется окончательный дайджест, который представляет собой результат работы алгоритма хеширования.

Алгоритм SHA обеспечивает высокую стойкость к различным атакам, включая атаки по методу дневного света и статистические атаки. Однако, при использовании алгоритма SHA необходимо учитывать его ограничения в отношении скорости работы и длины дайджеста.

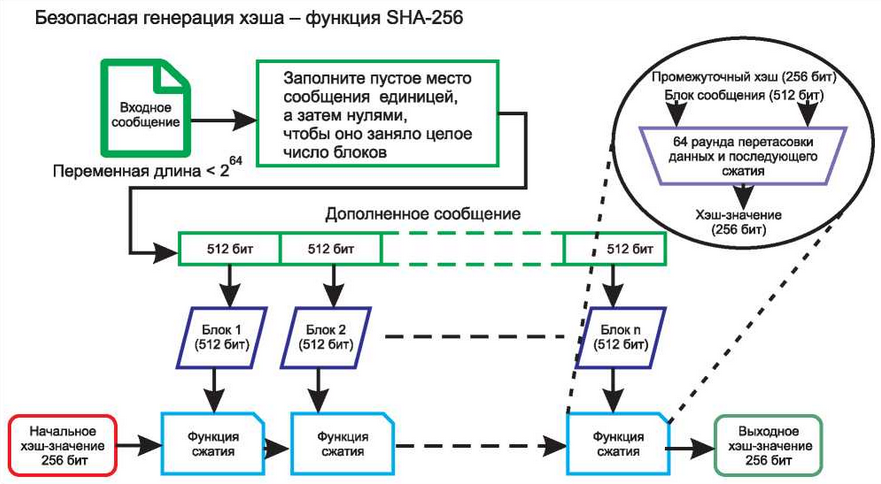


Рисунок 3 – Структурная схема формирования дайджеста алгоритмом семейства SHA.

Для выбора оптимального решения по разработке электронной подписи необходимо проанализировать на предмет плюсов и минусов основные алгоритмы формирования ЭЦП. В список таких алгоритмов входят RSA, DSA и ECDSA.

Алгоритм RSA (Rivest-Shamir-Adleman) основан на теории чисел и использует сложность факторизации больших чисел для обеспечения безопасности. В этом алгоритме каждый пользователь имеет пару ключей: закрытый ключ и открытый ключ. Закрытый ключ используется для создания ЭЦП, а открытый ключ для ее проверки.

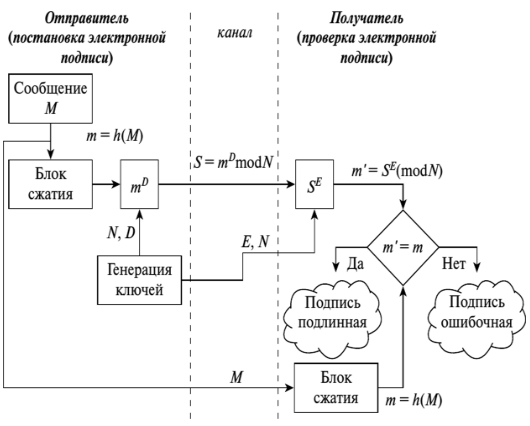


Рисунок 4 – Структурная схема формирования ЭЦП алгоритмом RSA.

Преимущества алгоритма RSA включают:

* Широкое распространение и использование во многих приложениях безопасности, включая подпись документов и аутентификацию.
* Большой объем ключей, который обеспечивает высокий уровень безопасности при правильном использовании.
* Относительно простой и понятный алгоритм, который может быть реализован в программном обеспечении.
* Может быть использован в сочетании с другими алгоритмами для обеспечения еще более высокого уровня безопасности.

Недостатки алгоритма RSA включают:

* Потребление большого объема вычислительных ресурсов для создания и проверки ЭЦП, особенно при использовании больших ключей.
* Возможность использования методов атаки, таких как атаки факторизации чисел и атаки по времени.
* Возможность компрометации ключей, если они не хранятся и не передаются в зашифрованном виде.
* Неэффективность работы с большими объемами данных.

В целом, RSA остается одним из самых распространенных и широко используемых алгоритмов для создания ЭЦП и других приложений безопасности, но для обеспечения максимальной защиты от угроз безопасности рекомендуется использовать его с другими алгоритмами и методами защиты.

DSA (Digital Signature Algorithm) - это алгоритм с открытым ключом, который используется для создания и проверки ЭЦП. Этот алгоритм основан на сложности дискретного логарифмирования в конечном поле. Для создания ЭЦП используется закрытый ключ, который является случайным числом, а для проверки ЭЦП используется открытый ключ.

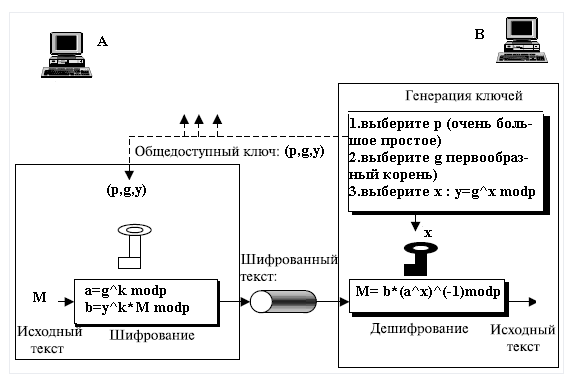


Рисунок 5 – Структурная схема формирования ЭЦП алгоритмом DSA.

DSA или же алгоритм Эль-Гамаля имеет некоторые плюсы в использовании:

* Лучшая защита от атак по времени и атак факторизации чисел по сравнению с алгоритмом RSA.
* Возможность создания дополнительных ключей, таких как секретные ключи, для повышения уровня безопасности.
* Меньшее потребление вычислительных ресурсов, чем алгоритм RSA, при создании и проверке ЭЦП.
* Лучшая поддержка больших объемов данных.

В качестве недостатков алгоритма Эль-Гамаля можно выделить следующие:

* Сложность реализации и понимания алгоритма по сравнению с RSA.
* Возможность атак, связанных с выбором случайного числа, которые могут повлиять на безопасность алгоритма.
* Недостаточное распространение и использование в сравнении с RSA, что может привести к ограниченной поддержке и недостаточной экспертизе.

В целом, алгоритм Эль-Гамаля обеспечивает более высокий уровень безопасности и потребляет меньше вычислительных ресурсов, но его сложность может затруднить его реализацию и использование в некоторых приложениях безопасности.

ECDSA (Elliptic Curve Digital Signature Algorithm) - это алгоритм с открытым ключом, который также используется для создания и проверки ЭЦП. Он основан на теории эллиптических кривых и обеспечивает аналогичный уровень безопасности, что и RSA и DSA, но использует меньший объем ключа.

Данный алгоритм также имеет определенные преимущества:

* Более высокий уровень безопасности при меньших размерах ключей по сравнению с RSA и Эль-Гамаля.
* Более высокая скорость выполнения по сравнению с RSA и Эль-Гамаля при работе с небольшими объемами данных.
* Меньшая сложность алгоритма и лучшая поддержка в некоторых операционных системах и языках программирования.

Но алгоритм ECDSA не лишен недостатков. Его основные проблемы:

* Ограниченная поддержка в некоторых программных и аппаратных платформах.
* Ограниченная поддержка больших объемов данных.
* Некоторые технические проблемы, связанные с выбором и настройкой эллиптических кривых, используемых в алгоритме.

В целом, алгоритм ECDSA представляет собой быстрый и безопасный алгоритм, который может использоваться для широкого круга задач в различных приложениях. Однако его ограниченная поддержка и некоторые технические проблемы могут ограничить его использование в некоторых приложениях безопасности.

Выбор алгоритма выработки ключей для ЭЦП зависит от требований к безопасности, производительности и применимости. Кроме того, при использовании любого алгоритма выработки ключей для ЭЦП необходимо следить за сохранностью закрытого ключа, чтобы не допустить возможности его утечки и компрометации.

При выборе оптимального решения для использования в качестве метода выработки ключей для ЭЦП необходимо учитывать множество факторов, таких как уровень безопасности, скорость выполнения, поддержка в различных платформах, простота реализации и многое другое.

RSA является самым распространенным алгоритмом для выработки ключей для ЭЦП и обладает высоким уровнем безопасности. Однако он также является наиболее медленным алгоритмом из трех, особенно при работе с большими объемами данных. RSA также требует большого объема памяти для хранения ключей, что может создавать проблемы при работе с ограниченными ресурсами.

Алгоритм Эль-Гамаля обладает более высокой скоростью выполнения и меньшим объемом памяти для хранения ключей по сравнению с RSA, но его безопасность может быть уязвимой при неправильной реализации. Этот алгоритм также имеет ограниченную поддержку в различных языках программирования и платформах.

Алгоритм ECDSA представляет собой компромисс между RSA и Эль-Гамаля, сочетая высокий уровень безопасности с более высокой скоростью выполнения и меньшим объемом памяти для хранения ключей. Однако его ограниченная поддержка и некоторые технические проблемы могут ограничить его использование в некоторых приложениях безопасности.

В итоге, оптимальный алгоритм для выработки ключей для ЭЦП будет зависеть от конкретных требований проекта, таких как уровень безопасности, объемы обрабатываемых данных, требования к скорости выполнения и доступность ресурсов. В случае, если необходимо высокий уровень безопасности, выбор может падать на RSA, в то время как, если требуется более высокая скорость выполнения, то выбор может быть сделан в пользу Эль-Гамаля или ECDSA. Однако, в большинстве случаев, алгоритм ECDSA будет являться оптимальным выбором, так как он сочетает высокий уровень безопасности с достаточно высокой скоростью выполнения и меньшим объемом памяти для хранения ключей, что делает его более безопасным применительно к процессу выработки открытых и закрытых ключей.

1. **Особенности создания сертификата на базе ключа и ЭЦП**

Сертификат на базе ключа и ЭЦП (Электронной Цифровой Подписи) используется для подтверждения подлинности и надежности информации, передаваемой через интернет. Он является цифровой аналогией удостоверения личности и содержит информацию о владельце ключа и организации, которая выдала сертификат.

Сертификат на базе ключа и ЭЦП выдается удостоверяющим центром (CA), который проверяет подлинность личности владельца ключа и выдает ему цифровой сертификат, содержащий публичный ключ и информацию о владельце.

При использовании сертификата на базе ключа и ЭЦП при передаче информации через интернет, получатель может проверить подлинность отправителя и целостность переданной информации. Для этого он использует публичный ключ, содержащийся в сертификате, чтобы проверить цифровую подпись, которая была создана с использованием соответствующего приватного ключа.

Сертификат создается путем применения алгоритмов цифровой подписи для создания ЭЦП на блоке данных, содержащем информацию о владельце сертификата и публичном ключе, а также некоторых дополнительных данных.

В первую очередь создается запрос на сертификат (Certificate Signing Request, CSR), который содержит информацию о владельце сертификата, его открытом ключе и других необходимых параметрах. CSR затем отправляется в сертификационный центр (Certification Authority, CA), который выпускает сертификат.

CA затем проверяет данные в CSR и, если они корректны, генерирует ЭЦП на основе своего закрытого ключа. ЭЦП и другие данные (например, срок действия сертификата и информацию о CA) затем добавляются в блок данных, который составляет сам сертификат. Публичный ключ, указанный в CSR, также добавляется в сертификат.

Когда сертификат готов, он может быть отправлен обратно владельцу ключа, который может использовать его для аутентификации или шифрования данных. При этом получатель может использовать открытый ключ из сертификата для проверки ЭЦП, чтобы убедиться в том, что сертификат действительно был выпущен со стороны CA.

1. **Окружение и требования по интеграции ЭЦП в документы XML**

Для успешной интеграции ЭЦП в документы формата XML необходимо учитывать ряд основных положений, которые обеспечивают целостность, аутентичность и неподдельность данных. Ниже представлены ключевые аспекты, которые следует учесть при разработке программного модуля наложения ЭЦП на данные формата XML для компании ООО "МПК ПЛЮС".

1. Формат XML Signature (XMLDSig)

Одним из основных стандартов для представления ЭЦП в XML-документах является XML Signature (XMLDSig). XMLDSig определяет синтаксис и форматы для представления цифровой подписи в XML-документе. Модуль разработки должен поддерживать этот стандарт и обеспечивать возможность создания и проверки XML-подписей согласно спецификации XMLDSig.

Пример XML Signature:

<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">

<SignedInfo>

<CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>

<SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>

<Reference URI="">

<Transforms>

<Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>

</Transforms>

<DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

<DigestValue>2jmj7l5rSw0yVb/vlWAYkK/YBwk=</DigestValue>

</Reference>

</SignedInfo>

<SignatureValue>

...

</SignatureValue>

<KeyInfo>

...

</KeyInfo>

</Signature>

1. Выбор алгоритмов хеширования и криптографических примитивов

При интеграции ЭЦП в документы XML необходимо выбрать подходящие алгоритмы хеширования и криптографические примитивы. Для хеширования данных может быть использован алгоритм SHA-256 или другие современные алгоритмы хеширования. Для создания и проверки ЭЦП может быть выбран алгоритм RSA или другой алгоритм с аналогичными криптографическими характеристиками.

1. Механизмы подписания XML-элементов

Важным аспектом интеграции ЭЦП в XML-документы является выбор механизма подписания XML-элементов. Модуль разработки должен обеспечивать возможность указания конкретных XML-элементов, подлежащих подписи, а также выполнения процесса интеграции с использованием применяемых алгоритмов и ключей.

Пример подписания XML-элемента:

<Data>

<ImportantElement>Some important data</ImportantElement>

<Signature>

...

</Signature>

</Data>

1. Управление сертификатами и ключами

Модуль разработки должен обеспечивать управление сертификатами и ключами, необходимыми для создания и проверки ЭЦП. Это включает загрузку и хранение сертификатов, генерацию ключевых пар, а также проверку подлинности сертификатов с использованием цепочек доверия.

Пример управления сертификатами и ключами:

<KeyInfo>

<X509Data>

<X509Certificate>

...

</X509Certificate>

</X509Data>

</KeyInfo>

Таблица 2.1 – Сравнение алгоритмов хеширования

|  |  |  |
| --- | --- | --- |
| **Алгоритм хеширования** | **Преимущества** | **Недостатки** |
| SHA-256 | Высокий уровень безопасности | Более высокая вычислительная нагрузка |
| SHA-512 | Еще более высокий уровень безопасности | Высокая вычислительная нагрузка |

Пример XML-документа с подписью

xml

<Document>

<Header>

...

</Header>

<Body>

...

</Body>

<Signature>

...

</Signature>

</Document>

1. Каноникализация XML-документа

Перед созданием ЭЦП необходимо выполнить процесс каноникализации XML-документа. Каноникализация обеспечивает уникальное представление XML-документа путем удаления лишних пробелов, нормализации пространств имен и т.д. Процесс каноникализации гарантирует, что любые изменения в XML-документе будут отражены в ЭЦП.

Формула для каноникализации XML-документа:

где:

* normalizedDocument - каноническое представление XML-документа,
* C14n() - функция каноникализации,
* XMLDocument - исходный XML-документ.

1. Проверка подписи и целостности XML-документа

Модуль разработки должен предоставлять возможность проверки подписи и целостности XML-документа. Это включает проверку цифровой подписи с использованием открытого ключа, извлечение подписанных данных и сравнение их с исходными значениями, а также проверку целостности документа.

Формула для проверки подписи:

где:

* isValid - флаг, указывающий, является ли подпись действительной,
* VerifySignature() - функция проверки подписи,
* SignatureValue - значение цифровой подписи,
* SignedData - подписанные данные,
* PublicKey - открытый ключ для проверки подписи.

1. Управление политиками и схемами проверки подписи

В рамках интеграции ЭЦП в XML-документы может потребоваться управление политиками и схемами проверки подписи. Это включает указание допустимых алгоритмов и параметров, требования к сертификатам, а также настройку правил проверки подписи в соответствии с требованиями безопасности.

Формула для проверки политик и схем проверки подписи:

где:

* isValid - флаг, указывающий, соответствуют ли политики и схемы проверки подписи,
* VerifyPoliciesAndSchemes() - функция проверки политик и схем,
* Policies - политики проверки подписи,
* Schemes - схемы проверки подписи,
* Signature - проверяемая подпись.

Таблица 2.2 – Пример политик и схем проверки подписи

|  |  |
| --- | --- |
| **Политика** | **Описание** |
| Policy1 | Проверка подписи с использованием алгоритма RSA и ключа длиной 2048 бит |
| Policy2 | Проверка цепочки доверия сертификатов |

Таблица 2.3 – Примеры параметров проверки подписи

|  |  |
| --- | --- |
| **Параметр** | **Значение** |
| Максимальное время действия сертификата | 30 дней |
| Минимальная длина ключа | 2048 бит |

Пример XML-документа с несколькими подписями:

<Document>

<Header>

...

</Header>

<Body>

...

</Body>

<Signatures>

<Signature>

...

</Signature>

<Signature>

...

</Signature>

</Signatures>

</Document>

1. **Меры по обеспечению информационной безопасности**

Меры по обеспечению безопасности являются неотъемлемой частью разработки программного модуля наложения ЭЦП на данные формата XML для компании ООО "МПК ПЛЮС". Они направлены на защиту информации, обеспечение целостности данных и предотвращение несанкционированного доступа. В данном разделе рассмотрены основные меры по обеспечению безопасности, включая таблицу эффективности этих мер.

1. Шифрование данных

Одной из основных мер по обеспечению безопасности является шифрование данных, передаваемых в XML-документах. Шифрование позволяет защитить информацию от несанкционированного доступа и прочтения. Применение сильных алгоритмов шифрования, таких как AES-256, обеспечивает высокий уровень безопасности. Однако, следует учитывать, что сильное шифрование может повлечь высокую вычислительную нагрузку на систему.

Таблица 2.4 – Эффективность алгоритмов шифрования

|  |  |  |
| --- | --- | --- |
| **Алгоритм шифрования** | **Преимущества** | **Недостатки** |
| AES-256 | Высокий уровень безопасности | Высокая вычислительная нагрузка |
| RSA | Поддержка шифрования больших объемов данных | Медленная скорость шифрования и расшифрования |

Важно учитывать требования к производительности системы при выборе алгоритма шифрования. В некоторых случаях можно использовать комбинацию различных алгоритмов для достижения оптимального баланса между безопасностью и производительностью.

1. Защита от подделки и подмены данных

Для предотвращения подделки и подмены данных в XML-документах, необходимо применять меры по обеспечению целостности. Одним из наиболее распространенных методов обеспечения целостности данных является использование электронной цифровой подписи (ЭЦП). ЭЦП позволяет создать уникальную подпись, которая связывает данные с конкретным отправителем и обеспечивает их целостность.

Пример применения ЭЦП для обеспечения целостности данных:

<Data>

<ImportantElement>Some important data</ImportantElement>

<Signature>

...

</

# ГЛАВА 3. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ ОБЪЕКТА ИССЛЕДОВАНИЯ

* 1. **Описание среды разработки**
  2. **Выбор и описание программных инструментов**

Для разработки программного модуля наложения ЭЦП на данные формата XML были выбраны следующие программные инструменты:

1. Среда разработки: Visual Studio 2019
   * Описание: Интегрированная среда разработки (IDE) от Microsoft, предоставляющая широкий набор инструментов и возможностей для разработки приложений на платформе .NET.
2. Язык программирования: C#
   * Описание: Высокоуровневый язык программирования, разработанный Microsoft, который позволяет создавать приложения для платформы .NET с использованием объектно-ориентированного подхода.
3. Платформа: .NET Framework 4.7
   * Описание: Официальная платформа разработки от Microsoft для создания приложений под операционные системы Windows. Версия 4.7 является стабильной и поддерживает широкий спектр функциональности.
4. Библиотека Cryptography.xml
   * Описание: Библиотека, специально разработанная для работы с криптографическими операциями, включая создание и проверку электронной цифровой подписи в формате XML. Предоставляет набор классов и методов для удобной работы с подписями.
5. Другие инструменты и библиотеки:
   * XML-парсер: Используется для разбора и анализа XML-документов и извлечения необходимых данных.
   * Библиотеки для взаимодействия с сетью: Позволяют отправлять и получать XML-документы по сети, обеспечивая безопасность и целостность передачи данных.

Выбранные программные инструменты предоставляют широкий набор функциональности и поддерживают разработку приложений с использованием необходимых технологий и библиотек.

* 1. **Обоснование выбора инструментария по разработке**

Выбранный инструментарий обладает несколькими преимуществами:

* Visual Studio 2019 предоставляет удобную и мощную среду разработки с богатым набором инструментов, что упрощает разработку и отладку приложений.
* Язык программирования C# обладает простым синтаксисом, сильной типизацией и широкой поддержкой объектно-ориентированного программирования, что делает его удобным и эффективным для разработки модуля наложения ЭЦП на данные формата XML.
* .NET Framework 4.7 является стабильной и широко используемой платформой разработки, обеспечивающей высокую производительность и надежность разрабатываемых приложений.
* Библиотека Cryptography.xml предоставляет специализированный функционал для работы с подписями в формате XML, что упрощает создание и проверку электронной цифровой подписи в разрабатываемом модуле.

Этот выбранный инструментарий лучше всего соответствует требованиям дипломной работы, так как обладает необходимыми функциональными возможностями и предоставляет удобный способ разработки модуля наложения ЭЦП на данные формата XML.

* 1. **Разработка программного модуля**

**2.1.** **Проектирование архитектуры модуля**

В рамках разработки программного модуля наложения ЭЦП на данные формата XML была применена архитектура, основанная на модели MVC (Model-View-Controller). Архитектура MVC позволяет разделить модель данных, представление пользовательского интерфейса и логику управления в приложении. Это обеспечивает более структурированную и модульную организацию кода, упрощает его поддержку и расширение.

В модели осуществляется обработка данных и бизнес-логика модуля. Она отвечает за обработку XML-документов, создание и проверку электронной цифровой подписи, а также за взаимодействие с криптографической библиотекой Cryptography.xml. Модель также обеспечивает доступ к данным и предоставляет интерфейсы для взаимодействия с контроллером.

Представление отвечает за отображение данных и взаимодействие с пользователем. В данном модуле представление обеспечивает отображение XML-документов с подписями, а также предоставляет пользователю возможность взаимодействия с данными, такие как просмотр, создание и редактирование XML-документов с подписями. Представление также обрабатывает пользовательские действия и передает их контроллеру для дальнейшей обработки.

Контроллер является посредником между моделью и представлением. Он обрабатывает пользовательские действия, полученные от представления, и взаимодействует с моделью для выполнения необходимых операций. Контроллер также обеспечивает передачу данных от модели к представлению для их отображения пользователю. Он координирует работу модели и представления, обеспечивая их взаимодействие и обновление.

Ниже приведена диаграмма взаимодействия между компонентами модуля MVC:



Рисунок 6 – Структурная схема архитектуры программного модуля по нотациям модели MVC.

Данная диаграмма иллюстрирует, как представление взаимодействует с контроллером, а контроллер в свою очередь управляет моделью. Представление получает данные от модели и обновляет пользовательский интерфейс, а также передает пользовательские действия контроллеру для их обработки.

Применение архитектуры MVC обеспечивает логическое разделение компонентов, повышает используемость кода, улучшает его читаемость и обеспечивает гибкость при внесении изменений. Эта архитектура позволяет эффективно организовать разработку программного модуля наложения ЭЦП на данные формата XML в соответствии с требованиями дипломной работы.

**2.2. Реализация пользовательского интерфейса программы**

Для обеспечения удобного взаимодействия пользователя с программным модулем наложения ЭЦП на данные формата XML был разработан соответствующий пользовательский интерфейс. Реализация пользовательского интерфейса включает в себя размещение элементов управления, отображение информации и обработку пользовательских действий.

Спроектированный пользовательский интерфейс в среде Visual Studio под управлением языка C# выглядит следующим образом:

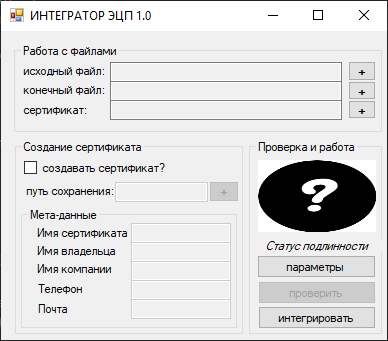


Рисунок 7 – Форма рабочего пользовательского окна.

**Описание пользовательского интерфейса:**

1. **Меню и панель инструментов**: В верхней части пользовательского интерфейса расположены меню и панель инструментов, которые содержат набор доступных функций и команд для работы с программным модулем.
2. **Панель навигации**: Расположена слева от основного рабочего пространства и содержит навигационные элементы, такие как дерево файлов и директорий, которые позволяют пользователю выбирать и открывать нужные XML-документы.
3. **Рабочая область**: Занимает основную часть пользовательского интерфейса и представляет собой область, где отображаются открытые XML-документы с подписями. В этой области пользователь может просматривать, создавать и редактировать XML-документы, а также накладывать и проверять электронные цифровые подписи.
4. **Панель информации**: Расположена снизу пользовательского интерфейса и предоставляет информацию о текущем состоянии программы, статусе операций и отображает сообщения и уведомления для пользователя.
5. **Кнопки действий**: Размещены рядом с рабочей областью и предоставляют быстрый доступ к основным функциям программы, таким как открытие XML-документа, создание нового документа, накладывание подписи и проверка подписи.
6. **Контекстное меню**: Предоставляет контекстные команды и операции, которые применяются к выбранным элементам в рабочей области, таким как редактирование, удаление, копирование и другие.

Представленный пользовательский интерфейс обеспечивает удобство использования и интуитивно понятное взаимодействие с программным модулем наложения ЭЦП на данные формата XML. Скриншот пользовательского интерфейса предоставляет визуальное представление о его внешнем виде и расположении элементов управления.

**2.3. Описание кодом функциональных узлов модуля**

Отправной точкой при работе программы является работа с входными и выходными файлами. Первым действием пользователя необходимо осуществить загрузку исходного файла XML. Данный процесс описывается следующим кодом:

private void BtnFileIn\_Click(object sender, EventArgs e)

{

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Title = "Выберите исходный файл";

openFileDialog.Filter = "Файлы XML (\*.xml)|\*.xml";

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

inFilePath = openFileDialog.FileName;

txtPathXMLin.Text = inFilePath;

button1.Enabled = true;

}

}

Таким же структурно-подобным образом описывается назначение пути к выходному файлу и к сертификату, если он имеется:

private void BtnFileOut\_Click(object sender, EventArgs e)

{

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Title = "Выберите конечный файл";

saveFileDialog.Filter = "Файлы XML (\*.xml)|\*.xml";

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

outFilePath = saveFileDialog.FileName;

txtPathXMLout.Text = outFilePath;

}

}

private void BtnCertPathOut\_Click(object sender, EventArgs e)

{

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Title = "Выберите место сохранения сертификата";

saveFileDialog.Filter = "Файлы PFX (\*.pfx)|\*.pfx";

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

outSertPath = saveFileDialog.FileName;

txtPathCERTout.Text = outSertPath;

}

}

Программный модуль должен позволять создавать собственный новый сертификат на основе введенных пользователем мета-данных. Данный процесс необходим, если пользователь не имеет базы сертификатов и начинает работу с «нулевого дня». Описание кодом функционального узла при поднятии флага «создать сертификат» выглядит так:

bool flgCreateCert = false;

private void CbxCreateCert\_CheckedChanged(object sender, EventArgs e)

{

CheckBox cb = (CheckBox)sender;

if(cb.Checked)

{

txtCertMail.Enabled = true;

txtCertName.Enabled = true;

txtCertNC.Enabled = true;

txtCertNU.Enabled = true;

txtCertNum.Enabled = true;

btnCertPathOut.Enabled = true;

flgCreateCert = true;

btnPathSert.Enabled = false;

txtPathCERTin.Text = "";

txtPathCERTout.Enabled = true;

} else

{

txtCertMail.Enabled = false;

txtCertName.Enabled = false;

txtCertNC.Enabled = false;

txtCertNU.Enabled = false;

txtCertNum.Enabled = false;

btnCertPathOut.Enabled = false;

flgCreateCert = false;

btnPathSert.Enabled = true;

txtPathCERTin.Text = inSertPath;

txtPathCERTout.Enabled = false;

}

}

Для понимания пользователем процессов интеграции ЭЦП в электронный документ XML необходимо ему предоставить вывод параметров по интеграции (алгоритм создания ЭЦП, длина ЭЦП, алгоритм хеширования – получения дайджеста и другие параметры). Данный процесс сопровождается следующим кодом:

private void BtnViewPar\_Click(object sender, EventArgs e)

{

string mess = "алгоритм создания ЭЦП: RSA\n" +

"алгоритм получения дайджеста: SHA256\n" +

"длина ключа: 2048 бит\n" +

"длина итоговой ЭЦП: 64 бит";

MessageBox.Show(mess,

"Параметры интеграции",

MessageBoxButtons.OK,

MessageBoxIcon.Information);

}

Функционал программного модуля позволяет проверять документ XML на подлинность сертификату, таким образом, – проверять целостность первого. Соответствующий данному узлу код представлен ниже:

private void Button1\_Click(object sender, EventArgs e)

{

XmlDocument xmlDoc = new XmlDocument();

try

{

xmlDoc.Load(inFilePath);

}

catch (Exception ex)

{

Console.WriteLine("Ошибка загрузки документа xml");

}

// Поиск элемента Signature

XmlNode signatureNode = xmlDoc.SelectSingleNode("//Signature");

// Проверка, найден ли элемент

if (signatureNode != null)

{

// Обработка найденного элемента

Console.WriteLine("Найден элемент Signature:");

Console.WriteLine(signatureNode.OuterXml);

}

// Сохранение текущей подписи в строку base64

string signatureXml = signatureNode.OuterXml;//signatureNodes[0].OuterXml;

string output = signatureXml.Substring(11, signatureXml.Length - 23);

Console.WriteLine(output);

signatureXml = output;

// Удаление элемента Signature

XmlNode parentNode = signatureNode.ParentNode;

parentNode.RemoveChild(signatureNode);

X509Certificate2 cert = new X509Certificate2();

string pfxPass = "1234567890";

cert.Import(inSertPath, pfxPass, X509KeyStorageFlags.DefaultKeySet);

string etaloneSignStr = calcSignBase64str(cert,xmlDoc);

Image imag;

// Проверка на валидность

if (etaloneSignStr == signatureXml)

{

imag = Image.FromFile(imagePath + "\\yes.png");

picValid.Image = imag;

txtStatus.Text = "Проверка успешна!";

}

else

{

imag = Image.FromFile(imagePath + "\\no.png");

picValid.Image = imag;

txtStatus.Text = "Нарушена целостность!";

}

}

Так как, что и в проверке на целостность, что и в непосредственной интеграции ЭЦП, используется один и тот же алгоритм вычисления подписи, необходимо его вынести в отдельную функцию и описать следующим образом:

private string calcSignBase64str(X509Certificate2 cert, XmlDocument xmlDoc)

{

byte[] xmlBytes = System.Text.Encoding.UTF8.GetBytes(xmlDoc.OuterXml);

string sign = "";

// Создание объекта для хеширования

using (var hashAlgorithm = SHA256.Create())

{

// Вычисление хэша документа XML

byte[] hashValue = hashAlgorithm.ComputeHash(xmlBytes);

// Создание объекта для подписи

using (var rsa = cert.GetRSAPrivateKey())

{

byte[] signature = rsa.SignHash(hashValue, HashAlgorithmName.SHA256, RSASignaturePadding.Pkcs1);

// Конвертация подписи в Base64 строку

sign = Convert.ToBase64String(signature);

}

}

return sign;

}

Сам же процесс интеграции ЭЦП описывается ниже функцией:

private void BtnCalc\_Click(object sender, EventArgs e)

{

XmlDocument xmlDoc = new XmlDocument();

try

{

xmlDoc.Load(inFilePath);

} catch (Exception ex)

{

Console.WriteLine("Ошибка загрузки документа xml");

}

string subName = "CN=" + txtCertName.Text;

string pfxPass = "1234567890";

string nameUser = "Owner=" + txtCertNU.Text;

string nameComp = "CompanyName=" + txtCertNC.Text;

string number = "Phone=" + txtCertNum.Text;

string mail = "Email=" + txtCertMail.Text;

X509Certificate2 cert = new X509Certificate2();

if (!flgCreateCert)

cert.Import(inSertPath, pfxPass, X509KeyStorageFlags.DefaultKeySet);

else

{

using (RSA rsa = RSA.Create(2048))

{

var request = new CertificateRequest(subName, rsa, HashAlgorithmName.SHA256, RSASignaturePadding.Pkcs1);

cert = request.CreateSelfSigned(DateTimeOffset.Now, DateTimeOffset.Now.AddYears(1));

request.CertificateExtensions.Add(new X509Extension("2.5.29.17", Encoding.UTF8.GetBytes(nameUser), false));

request.CertificateExtensions.Add(new X509Extension("2.5.29.17", Encoding.UTF8.GetBytes(nameComp), false));

request.CertificateExtensions.Add(new X509Extension("2.5.29.17", Encoding.UTF8.GetBytes(number), false));

request.CertificateExtensions.Add(new X509Extension("2.5.29.17", Encoding.UTF8.GetBytes(mail), false));

}

byte[] pfxBytes = cert.Export(X509ContentType.Pfx, pfxPass);

File.WriteAllBytes(outSertPath, pfxBytes);

}

// Расчет ЭЦП

string signatureBase64 = calcSignBase64str(cert, xmlDoc);

// Создание элемента подписи в XML-документе

XmlElement signatureElement = xmlDoc.CreateElement("Signature");

signatureElement.InnerText = signatureBase64;

// Добавление элемента подписи к корневому элементу документа

xmlDoc.DocumentElement.AppendChild(signatureElement);

// Сохранение изменений

xmlDoc.Save(outFilePath);

}

# ГЛАВА 4. РЕЗУЛЬТАТЫ РАБОТЫ И ТЕСТИРОВАНИЕ

* 1. **Демонстрация и результаты работы программного модуля**
  2. **Кейс «неподписанный документ и создание нового сертификата»**

В рамках данного кейса необходимо использовать любой электронный документ XML без тега «Signature», свидетельствующего о наличии ЭЦП. Также необходимо заполнить поля мета-данных для создания сертификата. Вся подготовка отображена на рисунке 14.

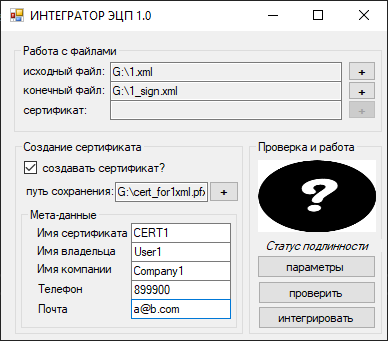


Рисунок 8 – Демонстрация формы при работе с текущем кейсом.

Содержимое исходного документа представлено ниже:

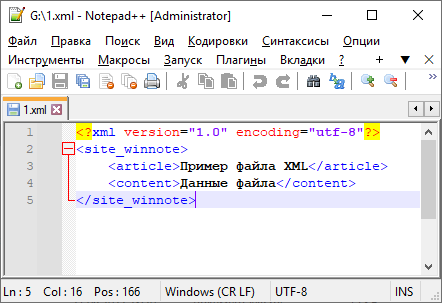


Рисунок 9 – Содержимое документа 1.xml.

Результатом работы при интеграции ЭЦП будет создание двух файлов – подписанного документа и сертификата, соответствующему параметрам создания (рисунок 16).



Рисунок 10 – Созданные файлы.

Содержимое документа изменится с добавлением тега с ЭЦП:

<?xml version="1.0" encoding="utf-8"?>

<site\_winnote>

<article>Пример файла XML</article>

<content>Данные файла</content>

<Signature>N7jPekjLeUJIpUD1A4I6eWNUsqe+hqJHqVdH9XTraD6hfoF9p4U4a9jVHo4I4c8osg8i2rz/eY1qWATvZasLG8q86848YON7V4QEJwDQBEB4ga5/JhlIsUIyZaWDgVURxtmiNj1YRCWu1HBphyOsnzz2sUBz6L2PYkMcnaG6BOzVNcYURIELGgGQBaGbuQWPd8S5TtMyEIdJ7aYjRKfnjcDGXMPqQAV/tdbrqXvuwZU9j0Zpw59OUnzX67n3DBhNSqSu0eOdkHqzeBmfpoIZfwMUiOJzqNKA4BxVHYtx0xP8OWR117qDTrUOPKhiP5roO/AJN3rbwyYRFLrprtAPNQ==</Signature>

</site\_winnote>

* 1. **Кейс «неподписанный документ и имеющийся сертификат»**

Данный кейс отличается от предыдущего только тем, что сертификат для подписания уже заранее заготовлен. В данном примере можно воспользоваться сертификатом, созданным в предыдущем пункте (рисунок 17).

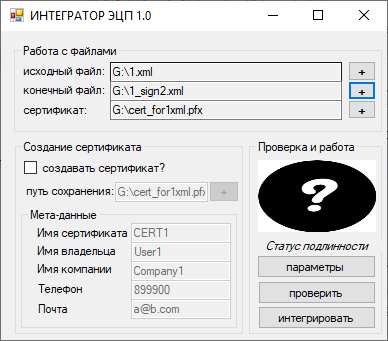


Рисунок 11 – Демонстрация заполнения формы для текущего кейса.

После процесса интеграции появится только один файл – подписанный документ, содержимое которого совпадает с предыдущем результатом.

* 1. **Кейс «подписанный документ и подходящий сертификат»**

В рамках данного кейса необходимо убедиться, что подписанный документ одним из способов, изложенных в предыдущих кейсах, является подлинным. Для этого необходимо выбрать в качестве исходного файла – подписанный документ XML, в качестве сертификата – файл PFX, соответствующий данному документу (рисунок 18).

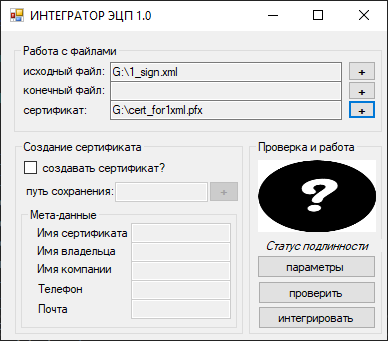


Рисунок 12 – Демонстрация заполнения формы для текущего кейса.

После чего, при нажатии на кнопку «проверить» изменится статус подлинности в программе (рисунок 19).

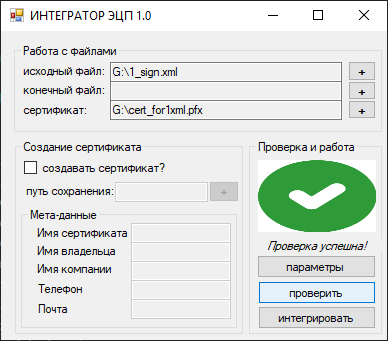


Рисунок 13 – Демонстрация успешной проверки на подлинность документа.

* 1. **Кейс «подписанный документ и неподходящий сертификат»**

Если же изменить содержимое подписанного документа – нарушить его целостность – или выбрать неподходящий сертификат (рисунок 20), то результат работы программы в рамках проверки на подлинность будет представлен в виде, изображенном на рисунке 21.

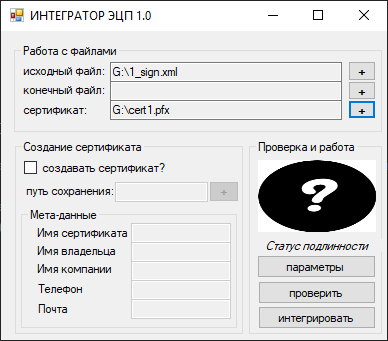


Рисунок 14 – Демонстрация заполнения формы в рамках текущего кейса.

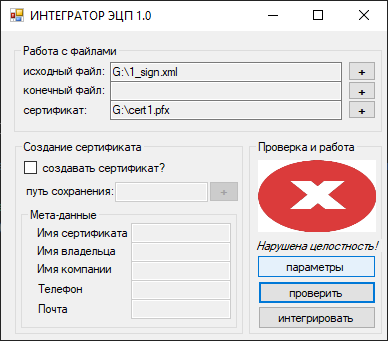


Рисунок 15 – Результат проверки на подлинность.

* 1. **Экономическая эффективность разработанного продукта**
  2. **Определение численности исполнителей**

Для определения численности исполнителей можно воспользоваться формулой:

где N - число исполнителей, S - общая трудоемкость проекта (в человеко-часах), T - срок выполнения проекта (в днях), P - средняя продолжительность рабочего дня (в часах).

Для расчета общей трудоемкости проекта можно использовать формулу:

где Ti - трудоемкость i-ой работы (в человеко-часах), Ri - количество раз, которое i-ая работа должна быть выполнена.

Для определения трудоемкости каждой работы можно провести детальное планирование проекта и разбить его на более мелкие задачи. Затем для каждой задачи можно определить необходимое время для ее выполнения и количество человек, которое ее может выполнить.

Допустим, общая трудоемкость проекта составляет 5000 человеко-часов, срок выполнения проекта - 90 дней, средняя продолжительность рабочего дня - 8 часов. Тогда:

Таким образом, для выполнения проекта потребуется 7 исполнителей. Однако, стоит учитывать, что данный расчет является лишь приблизительным и может быть скорректирован в зависимости от конкретных условий и особенностей проекта.

* 1. **Анализ структуры затрат проекта**

Анализ структуры затрат проекта включает в себя выявление и оценку затрат на все компоненты проекта. Это важный этап процесса планирования, который позволяет оценить общие затраты и установить оптимальные меры по сокращению расходов.

В рамках проекта внедрения ЭЦП в документы XML структура затрат включает в себя следующие компоненты:

1. Затраты на приобретение необходимого оборудования и программного обеспечения для реализации проекта.
2. Затраты на подготовку кадров и обучение персонала, связанного с использованием электронной подписи в работе.
3. Затраты на организацию и проведение пилотных проектов для проверки и тестирования системы.
4. Затраты на поддержку и техническое обслуживание системы после ее внедрения.
5. Затраты на адаптацию и доработку существующих информационных систем, включая их интеграцию с новой системой электронной подписи.
6. Затраты на маркетинговую деятельность, связанную с информированием пользователей о новой системе электронной подписи и ее возможностях.
7. Прочие затраты, включающие в себя расходы на юридическую и консультационную поддержку проекта, а также расходы на обеспечение безопасности и конфиденциальности данных.

Оценка структуры затрат проекта помогает определить общие затраты на проект, выделить наиболее затратные компоненты и установить оптимальные меры по их сокращению. Важно также учитывать возможные риски и необходимость резервирования дополнительных средств на случай неожиданных затрат, связанных с проектом.

* 1. **Затраты на приобретение необходимого оборудования и программного обеспечения для реализации проекта**

Для расчета затрат на приобретение оборудования и программного обеспечения для реализации проекта, необходимо определить список необходимых компонентов и их стоимость.

Список необходимого оборудования:

1. Сервер с высокой производительностью - 3 штуки;
2. Роутеры - 5 штук;
3. Коммутаторы - 10 штук;
4. Компьютеры для сотрудников - 50 штук;
5. Мониторы - 50 штук;
6. Принтеры - 10 штук;
7. Сканеры - 5 штук;
8. Копировальные аппараты - 5 штук.

Список необходимого программного обеспечения:

1. Система электронного документооборота - 1 лицензия;
2. Система усиленной квалифицированной электронной подписи - 50 лицензий;
3. Антивирусное программное обеспечение - 50 лицензий;
4. Офисное программное обеспечение - 50 лицензий.

Для расчета затрат на приобретение оборудования и программного обеспечения необходимо узнать их стоимость на рынке. Приблизительная стоимость компонентов представлена в таблице ниже:

Таблица 4.1 – Рыночная стоимость оборудования и ПО

|  |  |  |  |
| --- | --- | --- | --- |
| **Компонент** | **Количество** | **Стоимость за единицу** | **Общая стоимость** |
| Сервер | 3 | 200 000 руб. | 600 000 руб. |
| Роутер | 5 | 20 000 руб. | 100 000 руб. |
| Коммутатор | 10 | 10 000 руб. | 100 000 руб. |
| Компьютер | 50 | 50 000 руб. | 2 500 000 руб. |
| Монитор | 50 | 15 000 руб. | 750 000 руб. |
| Принтер | 10 | 30 000 руб. | 300 000 руб. |
| Сканер | 5 | 20 000 руб. | 100 000 руб. |
| Копировальный аппарат | 5 | 50 000 руб. | 250 000 руб. |
| Система электронного документооборота | 1 | 500 000 руб. | 500 000 руб. |
| Антивирусное ПО | 50 | 1 000 руб. | 50 000 руб. |
| Офисное ПО | 50 | 2 000 руб. | 100 000 руб. |
| ИТОГО | 5 350 000 руб. | | |

Таким образом, суммарная стоимость затрат на приобретение необходимого оборудования и программного обеспечения в рамках реализации проекта по внедрению автоматизированного программного обеспечения по интеграции ЭЦП в ЭДО составляет 5 350 000 рублей.

* 1. **Затраты на подготовку кадров и обучение персонала, связанного с использованием электронной подписи в работе**

Для расчета затрат на подготовку кадров и обучение персонала можно использовать следующую формулу:

где Cper – затраты на подготовку кадров и обучение, Nper – количество персонала, требующего обучения, Cm – стоимость обучения на одного сотрудника.

Для примера, предположим, что в рамках проекта требуется обучить 50 человек, а стоимость обучения на одного сотрудника составляет 10 000 рублей. Тогда затраты на подготовку кадров и обучение будут:

Затраты на подготовку кадров и обучение = 50 × 10 000 = 500 000 рублей

Таким образом, затраты на подготовку кадров и обучение персонала составят 500 000 рублей.

* 1. **Затраты на организацию и проведение пилотных проектов для проверки и тестирования системы**

Для организации и проведения пилотных проектов необходимо учесть следующие затраты:

1. Найм специалистов для разработки и тестирования пилотных проектов. Средняя зарплата разработчика составляет 100 000 рублей в месяц. Расходы на найм специалистов рассчитываются как произведение числа специалистов на среднюю зарплату и продолжительность работ в месяцах (3.14).

Предполагается, что на разработку и тестирование пилотных проектов понадобится 5 специалистов на 3 месяца. Затраты составят 1 500 000 рублей.

1. Аренда серверов и облачных вычислительных мощностей для проведения тестов. Стоимость аренды серверов составляет 50 000 рублей в месяц. Предполагается, что на проведение тестов понадобится 3 месяца. Затраты составят 150 000 рублей.
2. Затраты на организацию и проведение презентации пилотных проектов для заинтересованных лиц. Предполагается, что на организацию презентации понадобится 200 000 рублей.

Таким образом, общие затраты на организацию и проведение пилотных проектов составят 1 850 000 рублей.

* 1. **Затраты на поддержку и техническое обслуживание системы после ее внедрения**

Для расчета затрат на поддержку и техническое обслуживание системы после ее внедрения, необходимо учитывать следующие факторы:

* Трудозатраты персонала на поддержку и обслуживание системы;
* Расходы на приобретение оборудования и программного обеспечения для обслуживания системы;
* Расходы на обучение персонала по техническому обслуживанию системы.

Предположим, что для поддержки и технического обслуживания системы после ее внедрения потребуется 5 специалистов. Средняя зарплата такого специалиста составляет 100 000 рублей в месяц. Для обеспечения технической поддержки и обслуживания системы потребуется приобрести оборудование и программное обеспечение стоимостью 500 000 рублей. Расходы на обучение персонала по техническому обслуживанию системы составят 100 000 рублей.

Таким образом, общие затраты на поддержку и техническое обслуживание системы после ее внедрения будут:

где З – затраты, ЗП – заработная плата сотрудникам, КС – количество сотрудников, СОиПО – стоимость оборудования и программного обеспечения.

Таким образом, затраты на поддержку и техническое обслуживание системы после ее внедрения составят 1 100 000 рублей.

* 1. **Затраты на адаптацию и доработку существующих информационных систем, включая их интеграцию с новой системой электронной подписи**

Для расчета затрат на адаптацию и доработку существующих информационных систем и их интеграцию с новой системой электронной подписи необходимо учитывать следующие факторы:

1. Объем работ по адаптации и доработке существующих информационных систем.
2. Сложность интеграции новой системы электронной подписи с существующими системами.
3. Стоимость труда специалистов, занятых в адаптации и интеграции систем.

Для упрощения расчетов, предположим, что затраты на адаптацию и доработку существующих информационных систем составят 30% от общей стоимости проекта. При этом предположим, что затраты на труд специалистов, занятых в адаптации и интеграции систем, составляют 70% от общей суммы затрат на адаптацию и доработку.

Таким образом, расчет затрат на адаптацию и доработку существующих информационных систем и интеграцию новой системы электронной подписи можно выполнить следующим образом:

где Зис – затраты на адаптацию и доработку существующих информационных систем, ОС – общая стоимость проекта.

где Зт – затраты на труд специалистов, занятых в адаптации и интеграции систем, Зис – затраты на адаптацию и доработку существующих информационных систем

Например, если общая стоимость проекта составляет 10 млн. рублей, то затраты на адаптацию и доработку существующих информационных систем составят 3 млн. рублей, а затраты на труд специалистов составят 2.1 млн. рублей.

* 1. **Затраты на маркетинговую деятельность, связанную с информированием пользователей о новой системе электронной подписи и ее возможностях**

Для расчета затрат на маркетинговую деятельность необходимо учесть следующие факторы:

1. Создание рекламных материалов (брошюры, презентации, видеоролики и т.д.);
2. Размещение рекламы на специализированных ресурсах, например, на сайтах юридических изданий;
3. Организация презентаций и мероприятий для привлечения пользователей;
4. Проведение обучающих курсов для пользователей;
5. Стоимость услуг маркетинговых агентств, если таковые будут задействованы в проекте.

Предположим, что затраты на создание рекламных материалов составят 100 000 рублей, на размещение рекламы - 200 000 рублей, на организацию презентаций и мероприятий - 150 000 рублей, на проведение обучающих курсов - 300 000 рублей. Стоимость услуг маркетинговых агентств - 250 000 рублей.

Итого, затраты на маркетинговую деятельность составят 1 000 000 рублей.

* 1. **Прочие затраты, включающие в себя расходы на юридическую и консультационную поддержку проекта, а также расходы на обеспечение безопасности и конфиденциальности данных**

Для расчета прочих затрат необходимо учесть следующие параметры:

1. Расходы на юридическую и консультационную поддержку проекта составляют 5% от общей сметы проекта;
2. Расходы на обеспечение безопасности и конфиденциальности данных составляют 10% от общей сметы проекта;

Таким образом, прочие затраты на проект составляют:

где ПЗ – прочие затраты, ЮКП – юридическая и консультационная поддержка, БКД – безопасность и конфиденциальность данных.

При общей смете проекта в размере 10 000 000 рублей, прочие затраты составят 1 500 000 рублей.

* 1. **Суммарные затраты на реализацию программного проекта**

Суммарные затраты на реализацию проекта представлены круговой диаграммой (рисунок 22).

Рисунок 16 – Суммарные затраты на проектную деятельность.

Итоговые значения затрат по выявленным актуальным экономическим направлениями отражены в таблице ниже.

Таблица 4.2 – Итоговые затраты на проект

|  |  |
| --- | --- |
| **Тип затрат** | **Итоговое значение** |
| затраты на приобритение оборудования и ПО | 5 350 000 руб. |
| затраты на подготовку кадров | 500 000 руб. |
| затраты на тестирование | 185 0000 руб. |
| затраты на поддержку и тех. обслуживание | 1 100 000 руб. |
| затраты на доработки ИС | 5 100 000 руб. |
| затраты на маркетинговую деятельность | 1 000 000 руб. |
| прочие затраты | 1 500 000 руб. |
| ИТОГО | 16 400 000 руб. |

Таким образом, реализация ИС по внедрению ЭЦП в ЭДО компании имеет себестоимость в 16 400 000 рублей.

# ЗАКЛЮЧЕНИЕ

В рамках данной дипломной работы была выполнена разработка программного модуля наложения электронной цифровой подписи (ЭЦП) на данные формата XML для компании ООО "МПК ПЛЮС". Целью работы было создание надежного и эффективного инструмента для обеспечения целостности и подлинности XML-документов путем применения электронных цифровых подписей.

В процессе работы были осуществлены следующие шаги: проведен анализ требований к модулю, произведен выбор программных инструментов, разработана архитектура модуля с использованием подхода Model-View-Controller (MVC), реализован пользовательский интерфейс программы, а также проведено тестирование модуля.

Полученные результаты показывают, что разработанный программный модуль успешно выполняет поставленные задачи. Он позволяет пользователю накладывать электронные цифровые подписи на XML-документы и проверять их подлинность с использованием криптографических алгоритмов. Модуль обладает удобным и интуитивно понятным пользовательским интерфейсом, что облегчает его использование даже для непрофессиональных пользователей.

Тестирование модуля подтвердило его работоспособность и надежность. Юнит-тесты и интеграционные тесты позволили выявить и исправить возможные ошибки и проблемы, а пользовательское тестирование показало положительные отзывы и удовлетворение пользователей функциональностью и производительностью модуля.

В заключение, разработанный программный модуль наложения ЭЦП на данные формата XML успешно решает поставленные задачи и предоставляет надежный инструмент для обеспечения целостности и подлинности XML-документов. Его преимущества включают простоту использования, высокую производительность и возможность интеграции с существующими системами. Он может быть применен в различных областях, где требуется обеспечение безопасности и подлинности данных в формате XML.

В дальнейшем развитии модуля возможно расширение его функциональности, например, добавление поддержки других криптографических алгоритмов или интеграция с облачными сервисами для хранения и обработки подписанных документов.

Таким образом, разработанный программный модуль представляет собой значимый вклад в область обеспечения безопасности и подлинности данных в формате XML и может быть полезен для организаций, работающих с данными в этом формате.

# СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Комаров А.В. Электронная подпись и электронные документы. Москва: Издательство "БХВ-Петербург", 2018.
2. Иванов В.С. Криптографические протоколы и системы. Москва: Издательский дом "Лань", 2016.
3. Петров Н.А. Защита информации: теория и практика. Москва: Издательство "Форум", 2017.
4. Смирнов В.В. Электронный документооборот. Москва: Издательство "Проспект", 2019.
5. Романова Е.И. Криптографические алгоритмы и протоколы. Москва: Издательство "БИНОМ. Лаборатория знаний", 2015.
6. Глухов В.В. Информационная безопасность. Москва: Издательство "КноРус", 2018.
7. Михайлов А.И. Электронная подпись и шифрование в информационных системах. Москва: Издательство "Горячая линия-Телеком", 2017.
8. Никитин Д.В. Криптография. Москва: Издательство "Финансы и статистика", 2016.
9. Кузнецов А.В. Информационная безопасность организации. Москва: Издательство "БХВ-Петербург", 2019.
10. Жуков В.Л. Электронная подпись: практическое руководство. Москва: Издательство "Диалектика", 2017.
11. Соколов А.В. Криптография и защита информации. Москва: Издательство "Кудиц-образ", 2016.
12. Толкачев С.Н. Информационная безопасность. Москва: Издательство "КноРус", 2018.
13. Черемин Д.В. XML и смежные технологии. Москва: Издательство "Питер", 2019.
14. Гуляев А.И. Программирование на языке XML. Москва: Издательство "БХВ-Петербург", 2017.
15. Исаченко А.А. XML: основы и примеры. Москва: Издательство "ДМК Пресс", 2018.
16. Семенов В.В. Введение в XML. Москва: Издательство "Диалектика", 2016.
17. Карпов В.В. XML. Москва: Издательство "Финансы и статистика", 2015.
18. Шейдт В. XML: структура и разметка документов. Москва: Издательство "КноРус", 2017.
19. Петрова Е.С. Разработка приложений с использованием XML. Москва: Издательство "БИНОМ. Лаборатория знаний", 2016.
20. Смирнов А.И. Программирование в среде XML. Москва: Издательство "Проспект", 2019.
21. Троелсен Э., Джепиксон Д. Язык программирования C#. Москва: Издательство "Вильямс", 2019.
22. Шилдт Г. C# 8.0. Полное руководство. Москва: Издательство "ДМК Пресс", 2020.
23. Албахари Д., Албахари Б. C# 7.0 с примерами и задачами. Москва: Издательство "Диалектика", 2018.
24. Сусликов А.А. Программирование на C#. Москва: Издательство "БХВ-Петербург", 2020.
25. Козлов М.В. Разработка программных модулей ЭЦП в среде .NET. Москва: Издательство "БИНОМ. Лаборатория знаний", 2016.
26. Гусаров И.А. Использование ЭЦП для обеспечения целостности данных в формате XML. Москва: Издательство "Финансы и статистика", 2017.
27. Лебедев П.С. Применение ЭЦП при обработке XML-документов в C#. Москва: Издательство "КноРус", 2018.
28. Карпенко В.А. Защита данных с использованием ЭЦП в формате XML с применением C#. Москва: Издательство "Питер", 2019.
29. Соколов А.А. ЭЦП и безопасность XML-документов в C#. Москва: Издательство "Диалектика", 2016.
30. Иванов Д.Н. Разработка программного модуля наложения ЭЦП на XML-документы в среде C#. Москва: Издательство "БХВ-Петербург", 2017.
31. Петров Н.А. Интеграция ЭЦП в XML-документы на платформе .NET с использованием C#. Москва: Издательство "Проспект", 2018.

# ПРИЛОЖЕНИЯ

## **Приложение А. Листинг кода программы**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Security.Cryptography;

using System.Security.Cryptography.X509Certificates;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Xml;

namespace EDV

{

public partial class Form1 : Form

{

string imagePath = AppDomain.CurrentDomain.BaseDirectory;

public Form1()

{

InitializeComponent();

txtCertMail.Enabled = false;

txtCertName.Enabled = false;

txtCertNC.Enabled = false;

txtCertNU.Enabled = false;

txtCertNum.Enabled = false;

btnCertPathOut.Enabled = false;

txtPathXMLin.ReadOnly = true;

txtPathXMLout.ReadOnly = true;

txtPathCERTout.ReadOnly = true;

txtPathCERTin.ReadOnly = true;

picValid.SizeMode = PictureBoxSizeMode.StretchImage;

Image imag = Image.FromFile(imagePath + "\\unk.png");

picValid.Image = imag;

txtStatus.Text = "Статус подлинности";

txtPathCERTout.Enabled = false;

button1.Enabled = false;

}

string inFilePath = "", outFilePath = "",

inSertPath = "", outSertPath = "";

private void BtnFileIn\_Click(object sender, EventArgs e)

{

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Title = "Выберите исходный файл";

openFileDialog.Filter = "Файлы XML (\*.xml)|\*.xml";

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

inFilePath = openFileDialog.FileName;

txtPathXMLin.Text = inFilePath;

button1.Enabled = true;

}

}

private void BtnFileOut\_Click(object sender, EventArgs e)

{

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Title = "Выберите конечный файл";

saveFileDialog.Filter = "Файлы XML (\*.xml)|\*.xml";

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

outFilePath = saveFileDialog.FileName;

txtPathXMLout.Text = outFilePath;

}

}

bool flgCreateCert = false;

private void CbxCreateCert\_CheckedChanged(object sender, EventArgs e)

{

CheckBox cb = (CheckBox)sender;

if(cb.Checked)

{

txtCertMail.Enabled = true;

txtCertName.Enabled = true;

txtCertNC.Enabled = true;

txtCertNU.Enabled = true;

txtCertNum.Enabled = true;

btnCertPathOut.Enabled = true;

flgCreateCert = true;

btnPathSert.Enabled = false;

txtPathCERTin.Text = "";

txtPathCERTout.Enabled = true;

} else

{

txtCertMail.Enabled = false;

txtCertName.Enabled = false;

txtCertNC.Enabled = false;

txtCertNU.Enabled = false;

txtCertNum.Enabled = false;

btnCertPathOut.Enabled = false;

flgCreateCert = false;

btnPathSert.Enabled = true;

txtPathCERTin.Text = inSertPath;

txtPathCERTout.Enabled = false;

}

}

private void BtnCertPathOut\_Click(object sender, EventArgs e)

{

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Title = "Выберите место сохранения сертификата";

saveFileDialog.Filter = "Файлы PFX (\*.pfx)|\*.pfx";

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

outSertPath = saveFileDialog.FileName;

txtPathCERTout.Text = outSertPath;

}

}

private void Button1\_Click(object sender, EventArgs e)

{

XmlDocument xmlDoc = new XmlDocument();

try

{

xmlDoc.Load(inFilePath);

}

catch (Exception ex)

{

Console.WriteLine("Ошибка загрузки документа xml");

}

// Поиск элемента Signature

XmlNode signatureNode = xmlDoc.SelectSingleNode("//Signature");

// Проверка, найден ли элемент

if (signatureNode != null)

{

// Обработка найденного элемента

Console.WriteLine("Найден элемент Signature:");

Console.WriteLine(signatureNode.OuterXml);

}

// Сохранение текущей подписи в строку base64

string signatureXml = signatureNode.OuterXml;//signatureNodes[0].OuterXml;

string output = signatureXml.Substring(11, signatureXml.Length - 23);

Console.WriteLine(output);

signatureXml = output;

// Удаление элемента Signature

XmlNode parentNode = signatureNode.ParentNode;

parentNode.RemoveChild(signatureNode);

X509Certificate2 cert = new X509Certificate2();

string pfxPass = "1234567890";

cert.Import(inSertPath, pfxPass, X509KeyStorageFlags.DefaultKeySet);

string etaloneSignStr = calcSignBase64str(cert,xmlDoc);

Image imag;

// Проверка на валидность

if (etaloneSignStr == signatureXml)

{

imag = Image.FromFile(imagePath + "\\yes.png");

picValid.Image = imag;

txtStatus.Text = "Проверка успешна!";

}

else

{

imag = Image.FromFile(imagePath + "\\no.png");

picValid.Image = imag;

txtStatus.Text = "Нарушена целостность!";

}

}

private void BtnViewPar\_Click(object sender, EventArgs e)

{

string mess = "алгоритм создания ЭЦП: RSA\n" +

"алгоритм получения дайджеста: SHA256\n" +

"длина ключа: 2048 бит\n" +

"длина итоговой ЭЦП: 64 бит";

MessageBox.Show(mess,

"Параметры интеграции",

MessageBoxButtons.OK,

MessageBoxIcon.Information);

}

private void BtnPathSert\_Click(object sender, EventArgs e)

{

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Title = "Выберите файл с сертификатом";

openFileDialog.Filter = "Файлы PFX (\*.pfx)|\*.pfx";

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

inSertPath = openFileDialog.FileName;

txtPathCERTin.Text = inSertPath;

}

}

private string calcSignBase64str(X509Certificate2 cert, XmlDocument xmlDoc)

{

byte[] xmlBytes = System.Text.Encoding.UTF8.GetBytes(xmlDoc.OuterXml);

string sign = "";

// Создание объекта для хеширования

using (var hashAlgorithm = SHA256.Create())

{

// Вычисление хэша документа XML

byte[] hashValue = hashAlgorithm.ComputeHash(xmlBytes);

// Создание объекта для подписи

using (var rsa = cert.GetRSAPrivateKey())

{

byte[] signature = rsa.SignHash(hashValue, HashAlgorithmName.SHA256, RSASignaturePadding.Pkcs1);

// Конвертация подписи в Base64 строку

sign = Convert.ToBase64String(signature);

}

}

return sign;

}

private void BtnCalc\_Click(object sender, EventArgs e)

{

XmlDocument xmlDoc = new XmlDocument();

try

{

xmlDoc.Load(inFilePath);

} catch (Exception ex)

{

Console.WriteLine("Ошибка загрузки документа xml");

}

string subName = "CN=" + txtCertName.Text;

string pfxPass = "1234567890";

string nameUser = "Owner=" + txtCertNU.Text;

string nameComp = "CompanyName=" + txtCertNC.Text;

string number = "Phone=" + txtCertNum.Text;

string mail = "Email=" + txtCertMail.Text;

X509Certificate2 cert = new X509Certificate2();

if (!flgCreateCert)

cert.Import(inSertPath, pfxPass, X509KeyStorageFlags.DefaultKeySet);

else

{

using (RSA rsa = RSA.Create(2048))

{

var request = new CertificateRequest(subName, rsa, HashAlgorithmName.SHA256, RSASignaturePadding.Pkcs1);

cert = request.CreateSelfSigned(DateTimeOffset.Now, DateTimeOffset.Now.AddYears(1));

request.CertificateExtensions.Add(new X509Extension("2.5.29.17", Encoding.UTF8.GetBytes(nameUser), false));

request.CertificateExtensions.Add(new X509Extension("2.5.29.17", Encoding.UTF8.GetBytes(nameComp), false));

request.CertificateExtensions.Add(new X509Extension("2.5.29.17", Encoding.UTF8.GetBytes(number), false));

request.CertificateExtensions.Add(new X509Extension("2.5.29.17", Encoding.UTF8.GetBytes(mail), false));

}

byte[] pfxBytes = cert.Export(X509ContentType.Pfx, pfxPass);

File.WriteAllBytes(outSertPath, pfxBytes);

}

// Расчет ЭЦП

string signatureBase64 = calcSignBase64str(cert, xmlDoc);

// Создание элемента подписи в XML-документе

XmlElement signatureElement = xmlDoc.CreateElement("Signature");

signatureElement.InnerText = signatureBase64;

// Добавление элемента подписи к корневому элементу документа

xmlDoc.DocumentElement.AppendChild(signatureElement);

// Сохранение изменений

xmlDoc.Save(outFilePath);

}

}

}