В современном мире происходит сокращение использования бумажных носителей информации, благодаря развитию информационных технологий можно перейти на системы электронного документооборота. Но чем больше развиваются системы и увеличиваются способы обмена информацией, тем чаще возникают угрозы подделки электронных документов.

Для того чтобы иметь возможность безопасно производить обмен данными в электронном виде, необходимо применять надёжные методы защиты. К таким методам относятся криптографические алгоритмы электронной подписи, хеширования и шифрования.

Павлова А.С. — студентка Набережночелнинского института (филиал) ФГАОУ ВО «Казанского федерального университета».
Ерёмина И.И. — к.п.н, доцент Набережночелнинского института (филиал) ФГАОУ ВО «Казанского федерального университета».
Лысанов Д.М. — к.т.н, доцент Набережночелнинского института (филиал) ФГАОУ ВО «Казанского федерального университета».

9 786203 580334

Алёна Павлова
Ирина Ерёмина
Денис Лысанов

Криптографические алгоритмы

в программных продуктах линейки «1С:МЕДИЦИНА»

LAP LAMBERT Academic Publishing

Алёна Павлова
Ирина Ерёмина
Денис Лысанов

Криптографические алгоритмы

Алёна Павлова
Ирина Ерёмина
Денис Лысанов

Криптографические алгоритмы

в программных продуктах линейки «1С:МЕДИЦИНА»

LAP LAMBERT Academic Publishing RU

Imprint

Cover image: www.ingimage.com

Publisher:
LAP LAMBERT Academic Publishing
is a trademark of
International Book Market Service Ltd., member of OmniScriptum Publishing Group
17 Meldrum Street, Beau Bassin 71504, Mauritius
Printed at: see last page
ISBN: 978-620-3-58033-4

СОДЕРЖАНИЕ

ВВЕДЕНИЕ ... 3
Глава 1. Криптографические алгоритмы: необходимость использования и принципы работы ... 6
1.1. Обеспечение защиты информации в системах электронного документооборота ... 6
1.2. Криптографические методы защиты информации ... 7
1.3. Формат электронной подписи XMLDSig ... 10
1.4. Стандарты криптографических алгоритмов ... 12
1.5. Необходимые средства для создания электронной подписи и шифрования данных ... 14
Глава 2. Применение методов и средств криптографии в программных продуктах линейки «1С:Медицина» ... 16
2.1. Места использования электронной подписи и шифрования в конфигурациях «1С:Медицина» ... 16
2.2. Механизм криптографии платформы «1С:Предприятие» ... 18
2.3. Криптографические методы конфигурации «1С:Библиотека стандартных подсистем» ... 20
2.4. Методы подсистемы «Электронный документооборот с контролирующими органами» ... 24
Глава 3. Реализация поддержки криптографических алгоритмов ГОСТ Р 34.10-2012 и ГОСТ Р 34.11-2012 ... 28
3.1. Определение этапов реализации ... 28
3.2. Добавление подписания по ГОСТ Р 34.10-2012 и формирования хеша по ГОСТ Р 34.11-2012 ... 29
3.3. Настройка шифрования и расшифровки сообщений с использованием ГОСТ Р 34.10-2012 ... 40
3.4. Тестирование ... 55
Глава 4. Реализация шифрования и расшифровки сообщений на стороне сервера ... 58

4.1. Определение этапов выполнения задачи 58
4.2. Описание программной реализации 60
4.3. Тестирование 65
ЗАКЛЮЧЕНИЕ 71
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 72
ПРИЛОЖЕНИЕ 1. 78
Параметры алгоритмов электронной подписи и хеширования 78
ПРИЛОЖЕНИЕ 2. 79
Реализованные методы для шифрования и расшифровки сообщений на стороне сервера 79

ВВЕДЕНИЕ

С февраля 2020 года в России (в Татарстане с марта 2020 года) началось распространение короновируса (COVID-19), что привело к введению дистанционного формата практически во всех сферах, поэтому информационные технологии приобрели новую значимость. Многие компании перешли с бумажных носителей на системы электронного документооборота.

В связи с активным внедрением таких систем и непрерывному обмену информацией всё чаще возникают угрозы подделки электронных документов.

Для того чтобы иметь возможность наиболее безопасно производить обмен данными в электронном виде, необходимо применять надёжные методы защиты. На сегодняшний день к таким методам относятся криптографические алгоритмы электронной подписи, хеширования и шифрования.

С каждым годом уровень уязвимости действующих криптографических алгоритмов возрастает. В связи с этим требуется утверждать новые стандарты, использующие усовершенствованные алгоритмы, которые могут лучше противостоять возможным атакам.

С 1 января 2019 года в Российской Федерации введены в использование стандарты ГОСТ Р 34.10-2012 и ГОСТ Р 34.11-2012 для электронной подписи и хеширования, соответственно.

Исходя из требований законодательства, все информационные системы, поддерживающие электронный документооборот, должны иметь возможность работать по обоим ГОСТам электронной подписи и хеширования в 2021 году. К таким системам относятся программные продукты линейки «1С:Медицина»: «1С:Медицина. Поликлиника», «1С:Медицина. Больница», «1С:Медицина. Больничные», которые предназначены для автоматизации деятельности медицинских организаций, а

также для мониторинга здоровья населения Республики Татарстан и других регионах России.

После знакомства с программным кодом было обнаружено, что операции шифрования и расшифровки в конфигурациях линейки «1С:Медицина» осуществляется только в клиентском варианте работы. Для повышения удобства использования медицинского документооборота было решено добавить возможность шифрования и расшифровки на стороне сервера, используя ключи по ГОСТ Р 34.10-2001 и ГОСТ Р 34.10-2012, при работе в клиент-серверном режиме платформы «1С:Предприятие».

Таким образом, целью исследования является реализация поддержки криптографических алгоритмов ГОСТ Р 34.10-2012 и ГОСТ Р 34.11-2012 в программных продуктах линейки «1С:Медицина». Для достижения поставленной цели были определены следующие задачи:

1. проанализировать предметную область и рассмотреть стандарты криптографических алгоритмов;

2. определить методы и средства криптографии, используемые в программных продуктах линейки «1С:Медицина»;

3. добавить возможность подписания по ГОСТ Р 34.10-2012 и формирования хеша по ГОСТ Р 34.11-2012;

4. настроить операции шифрования и расшифровки с использованием ключей по ГОСТ Р 34.10-2012;

5. обеспечить возможность шифрования и расшифровки сообщений на стороне сервера;

6. протестировать использование алгоритмов ГОСТ Р 34.10-2012 и ГОСТ Р 34.11-2012 в программах линейки «1С:Медицина».

Новизна исследования состоит в том, что реализованная поддержка криптографических алгоритмов ГОСТ Р 34.10-2012 и ГОСТ Р 34.11-2012 добавляет новую функциональность в программные продукты линейки «1С:Медицина», что позволяет выполнить требования законодательства

Российской Федерации об использовании двух ГОСТов как для электронной подписи, так для хеширования в системах электронного документооборота.

Практическая значимость работы заключается во внедрении поддержки криптографических алгоритмов ГОСТ Р 34.10-2012 и ГОСТ Р 34.11-2012 в программные продукты линейки «1С:Медицина», которые используются в медицинских организациях по всей России.

Структура исследования. Работа состоит из введения, четырёх глав, заключения, списка литературы из 50 источников и 2 приложений. В работе содержится 15 рисунков и 24 листинга программного кода. Объём составляет 89 страниц.

В первой главе объясняется необходимость использования электронной подписи, хеширования и шифрования в системах электронного документооборота, рассматриваются принципы работы и стандарты криптографических алгоритмов, описывается формат электронной подписи XMLDSig, а также приводятся средства, необходимые для работы с методами криптографии.

Во второй главе описывается использование электронной подписи и шифрования в программных продуктах линейки «1С:Медицина», а также рассматриваются криптографические методы и средства, реализованные на платформе «1С:Предприятие».

Третья глава содержит описание реализации поддержки криптографических алгоритмов ГОСТ Р 34.10-2012 и ГОСТ Р 34.11-2012 в конфигурациях линейки «1С:Медицина».

В четвёртой главе описывается программная реализация шифрования и расшифровки сообщений на стороне сервера для использования в клиент-серверном режиме работы платформы «1С:Предприятие».

В заключении перечисляются результаты, полученные при выполнении исследования.

Глава 1. Криптографические алгоритмы: необходимость использования и принципы работы

1.1. Обеспечение защиты информации в системах электронного документооборота

Для мониторинга здоровья населения Республики Татарстан в связи с ситуацией в мире важно использовать электронный документооборот, для предотвращения короновирусной инфекции.

Электронный документооборот — это механизм, который включает в себя приём, рассылку, хранение и повторное использование документов в электронном виде [19]. Такой механизм предоставляет ряд преимуществ по сравнению с обычным документооборотом: снижение затрат на распечатку, пересылку, хранение, а также значительное сокращение времени поиска и доставки документов. Однако, несмотря на преимущества, в электронные документы, в отличие от бумажных аналогов, легче внести изменения и труднее доказать факт их подлинности [23]. Также необходимо, чтобы электронный документ обладал юридической силой [3, 18].

Для того чтобы значительно снизить риски подделки данных при осуществлении различных операций с электронными документами, в информационных системах безопасность обеспечивается при помощи электронной подписи и шифрования данных.

Электронная подпись позволяет получателю «убедиться в аутентичности источника информации (иными словами, в том, кто является автором), а также проверить, была ли информация изменена (искажена), пока находилась в пути» [34].

Согласно Федеральному закону «Об электронной подписи» [29], электронная подпись — это реквизит электронного документа, который предназначен для защиты этого документа от подделки путём криптографических преобразований информации. Также, электронный документ, подписанный электронной подписью, имеет юридическую силу, как и бумажный, подписанный вручную.

Поскольку обмен электронными документами происходит по сети Интернет и очень часто по незащищённым каналам связи, то вся передаваемая информация может быть подвержена атакам, которые направлены на нарушение её конфиденциальности. В таком случае используется шифрование, позволяющее обеспечить секретность сообщения [32], предоставляя доступ к его содержанию только тем лицам, между которыми происходит обмен данными.

1.2. Криптографические методы защиты информации

Способы формирования и проверки электронной подписи, процедуры шифрования и расшифровки документов относятся к науке, называемой криптографией, которая занимается построением и исследованием математических методов преобразований информации [17, с. 7].

В настоящий момент формирование электронной подписи осуществляется при помощи схем, которые основаны на симметричных и асимметричных алгоритмах шифрования [24].

При симметричном шифровании применяется один и тот же криптографический ключ для шифрования и расшифровки сообщений. Существенным недостатком такого способа шифрования является сложность передачи секретного ключа, поскольку для этого нужны защищённые каналы связи.

В асимметричных алгоритмах шифрования не нужно передавать ключ для расшифровки по каналам связи. При использовании таких алгоритмов данные шифруются при помощи открытого ключа, а процедура их расшифровки может быть произведена только при наличии закрытого [33]. Для шифрования и расшифровки сообщений возможно использовать открытый и закрытый ключи только из одной пары, поскольку они связаны друг с другом математическими зависимостями. Стоит отметить, что при наличии открытого ключа для шифрования нельзя вычислить закрытый ключ и, соответственно, расшифровать сообщение. Это позволяет максимально

безопасно производить обмен данными, так как закрытый ключ содержится только у получателя и не передаётся по каналам связи.

Несмотря на то, что асимметричные алгоритмы обеспечивают высокую степень защиты информации, их следует применять только для обмена сообщениями небольшого объёма. Связано это с тем, что асимметричные алгоритмы используют более сложные математические вычисления, чем симметричные, и поэтому требуют больше вычислительных ресурсов на выполнение процедур шифрования и расшифровки.

Для наиболее безопасного обмена данными в настоящее время применяются комбинированные схемы шифрования [21], которые позволяют в значительной степени избежать недостатков симметричного и асимметричного шифрования. В подобных схемах сообщение шифруется симметричным методом, а секретный ключ — по алгоритму асимметричного шифрования. Процесс расшифровки также происходит при помощи двух методов:

1. асимметричным метод шифрования вычисляется секретный ключ;

2. симметричным методом расшифровывается сообщение при помощи вычисленного ключа.

В симметричных схемах электронной подписи, как и в симметричном шифровании, применяется один и тот же криптографический ключ для формирования подписи и её проверки. Такие схемы основаны на использовании блочных шифров [2] и работают по принципу разбиения информации на отдельные блоки, каждый из которых подлежит процедуре подписания.

Асимметричные схемы электронной подписи относятся к криптосистемам с открытым ключом и наиболее часто применяются для обеспечения защиты информации. Это связано с тем, что для электронной подписи, с момента её возникновения, не удалось реализовать эффективные алгоритмы, которые основаны на симметричных шифрах, ввиду того, что необходимо подписывать каждый блок передаваемой информации, а это

приводит к значительному увеличению объёма подписываемого документа [24]. Также стоит отметить, что при использовании симметричной схемы электронной подписи затруднено определение автора документа, поскольку секретный ключ известен как подписывающей, так и проверяющей сторонам.

Подписание документов в асимметричных схемах электронной подписи осуществляется при помощи закрытого ключа, а проверка — с применением открытого. Использование асимметричной схемы электронной подписи выглядит следующим образом [38]:

1. Генерируется контейнер закрытого ключа, а также вычисляется соответствующий ему открытый ключ.

2. При помощи закрытого ключа, находящегося в контейнере, формируется подпись для электронного документа.

3. Проверка подписи осуществляется с помощью открытого ключа, связанного с закрытым ключом, формировавшим подпись документа.

Поскольку подписываемые документы, как правило, имеют достаточно большой объем, то чаще всего электронная подпись ставится не на сам документ, а на его хеш [20], который формируется при помощи хеш-функции [39, 26]. Для генерации хеша используется криптографический алгоритм хеширования, способный преобразовывать произвольный массив данных в строку фиксированной длины, состоящую из цифр и букв.

При использовании алгоритма хеширования процесс формирования электронной подписи выглядит следующим образом:

1. Вычисляется значение хеш-функции для электронного документа, в результате чего получается строка символов, называемая хешем.

2. Происходит подписание полученного хеша при помощи закрытого ключа по алгоритму электронной подписи.

При использовании алгоритма хеширования для проверки подписи сначала формируется хеш документа. Затем этот хеш сравнивается с хешем, который установлен на сам документ. Также осуществляется проверка

подписи при помощи открытого ключа. Если хеши совпадают и подпись верна, то документ будет считаться подлинным.

1.3. Формат электронной подписи XMLDSig

На сегодняшний день существует большое количество форматов электронных документов [13, 25, 31]: DOC, DOCX, PDF, TXT, RTF, XML и др. Наиболее удобным форматом обмена документами считается XML [37, 48]. Это связано с тем, что для XML-документов существуют определённые правила разметки, благодаря которым соблюдается единая структура, позволяющая корректно производить разбор документа [43]. Стоит отметить, что для создания и чтения таких документов не обязательно устанавливать специализированное программное обеспечение, а достаточно использовать простые текстовые редакторы. Для обеспечения безопасности XML-документов рекомендуется использовать электронную подпись, которая определена в стандарте XML Digital Signature (XMLDSig), разработанным консорциумом W3C (World Wide Web Consortium) в 2002 году [50]. Электронные подписи могут быть сформированы и по другим стандартам, например, PKCS#7 [44] или CAdES [45]. Форматы подписи отличаются тем, что подпись XMLDSig после формирования помещается внутрь самого XML-документа, а подпись по стандарту PKCS#7 или CAdES находится в отдельном файле. Рекомендованный стандарт XMLDSig определяет синтаксис представления электронной подписи, который включает в себя правила её обработки, а также обеспечивает целостность данных, установление подлинности сообщений и подтверждение принадлежности документа подписавшему лицу. Структура 11 подписи XMLDSig представлена в листинге 1.

Листинг 1. Структура XMLDSig

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

```
<CanonicalizationMethod Algorithm=
   "http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
<SignatureMethod=
   "urn:ietf:params:xml:ns:cpxmlsec:algorithms:gostr34102001-gostr3411"/>
<Reference URI="#FDB437C278">
   <Transforms>
     <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
   </Transforms>
   <DigestMethod Algorithm=
     "urn:ietf:params:xml:ns:cpxmlsec:algorithms:gostr3411"/>
   <DigestValue>gGGf2w...</DigestValue>
  </Reference>
</SignedInfo>
<SignatureValue>uqTr7...</SignatureValue>
<KeyInfo .../>
</Signature>
```


Элементы подписи XMLDSig имеют следующие значения [50]:

- Signature — указывает область подписи в документе;
- SignedInfo — определяет данные и алгоритмы, используемые для формирования подписи;
- CanonicalizationMethod — указывает алгоритм каноникализации (канонизации) элемента SignedInfo XML-документа перед подписанием;
- SignatureMethod — определяет алгоритм, используемый для формирования и проверки подписи;
- Reference — содержит информацию о преобразованиях, которые применяются к документу, и содержит алгоритм и значение хеша;
- Transforms и Transform — описывают преобразования, которые должны применяться к документу до вычисления хеша;
- DigestMethod — определяет алгоритм вычисления хеша;

- DigestValue — хранит значение хеша;
- SignatureValue — определяет сформированное значение подписи;
- KeyInfo — указывает ключ, который используется для проверки подписи.

Перед тем как формировать подпись в формате XMLDSig, XML-документ должен соответствовать строгим правилам составления и требованиям к содержанию. Процесс приведения XML-документов к определённому формату называют каноникализацией (канонизацией) [41]. Консорциумом W3C (World Wide Web Consortium) определены несколько стандартов канонизации, в том числе Canonical XML [40] и Exclusive XML Canonicalization [42], которые используются в программных продуктах линейки «1С:Медицина». Отличие этих двух стандартов заключается в том, что Exclusive XML Canonicalization предназначен для документов, которые являются вложенными в другие, а Canonical XML определяет канонизацию обычного XML-документа.

К правилам канонизации относятся: использование кодировки UTF-8, применение двойных кавычек для значений атрибутов, организация определённого порядка элементов, удаление лишних объявлений пространств имён и т. д.

1.4. Стандарты криптографических алгоритмов

На сегодняшний день в Российской Федерации действуют стандарты криптографических алгоритмов электронной подписи и хеширования линейки ГОСТ Р 34.10 и ГОСТ Р 34.11, соответственно. Впервые введены в использование в 1995 году алгоритмы ГОСТ Р 34.10-94 [9] и ГОСТ Р 34.11-94 [11].

Для шифрования действуют стандарты ГОСТ 28147-89 [6] и ГОСТ Р 34.12-2015 [12], схемы работы которых относятся к симметричным шифрам.

В связи с повышением производительности компьютеров, активным продвижением и внедрением различных современных информационных

технологий уровень уязвимости действующих криптографических алгоритмов значительно возрастает.

Стоит отметить, что в настоящее время атаки на алгоритмы осуществляются в большей степени при помощи вредоносных аппаратных устройств. Такие устройства создаются для скрытого внедрения в информационные системы аппаратных закладок, которые маскируются под различные электронные устройства и способны так изменять схему работы криптографических алгоритмов, чтобы в результате был осуществлён несанкционированный доступ к конфиденциальным данным [28].

Для снижения рисков подделки данных утверждаются новые стандарты, использующие усовершенствованные алгоритмы, которые могут лучше противостоять возможным атакам.

В Российской Федерации уже осуществлялась замена алгоритма электронной подписи в 2002 году, когда вместо ГОСТ Р 34.10-94 был принят ГОСТ Р 34.10-2001 [7]. В ГОСТ Р 34.10-94 операции проводились над конечными алгебраическими полями [17, с. 78], а в ГОСТ Р 34.10-2001 — на эллиптических кривых, свойства которых рассмотрены в [27]. Оба ГОСТа электронной подписи используют функцию хеширования по стандарту ГОСТ Р 34.11-94.

Согласно выписке из документа ФСБ России [5], стандарт ГОСТ Р 34.102001 действителен до 31 декабря 2018 года, а формирование электронной подписи с 1 января 2019 года должно осуществляться только по ГОСТ Р 34.10-2012 [8]. Однако 7 сентября 2018 года был опубликован документ [35], согласно которому действие ГОСТ Р 34.10-2001 продлевается до 31 декабря 2019 года и при этом с 1 января 2019 года вводится в использование ГОСТ Р 34.10-2012.

Таким образом, в 2019 году допускается использование электронной подписи по двум стандартам.

Переход на алгоритм электронной подписи ГОСТ Р 34.10-2012 отложен ввиду того, что замена стандартов всегда является достаточно длительным

процессом, поскольку должны быть внесены изменения во все системы электронного документооборота, лицензированы средства криптографической защиты информации и подготовлены к выпуску сертификаты по новым алгоритмам.

При подписании по ГОСТ Р 34.10-2012 должен использоваться алгоритм хеширования по ГОСТ Р 34.11-2012 [10]. Основным видимым отличием ГОСТ Р 34.10-2012 от ГОСТ Р 34.10-2001 является возможность выбора длины хеша, поскольку стандарт ГОСТ Р 34.11-2012 позволяет формировать хеш длиной 256 или 512 бит, а ГОСТ Р 34.11-94 — только 256 бит.

1.5. Необходимые средства для создания электронной подписи и шифрования данных

Для того чтобы иметь возможность устанавливать на документы электронную подпись и организовать безопасный обмен данными при помощи шифрования, необходимо:

1. Получить квалифицированный сертификат электронной подписи и контейнер с закрытым ключом в аккредитованном удостоверяющем центре [36].

2. Установить средства криптографической защиты информации (СКЗИ), которые способны осуществлять шифрование и расшифровку данных, а также отвечать за работу с электронной подписью [17, с. 212].

Сертификат открытого ключа должен быть действительным, то есть не иметь просроченного срока действия и не быть включенным в список отозванных сертификатов (CRL), и содержать следующую информацию [29]:

1. уникальный номер сертификата, а также даты начала и окончания срока его действия;

2. фамилия, имя и отчество (если имеется) — для физических лиц, наименование организации и адрес — для юридических;

3. уникальный ключ проверки электронной подписи (открытый ключ);

4. названия и идентификаторы алгоритмов, которые используются для формирования электронной подписи;

5. данные удостоверяющего центра, который выдал сертификат.

СКЗИ подразделяются на следующие типы [17, с. 237]:

1. Программные СКЗИ, в которых реализация поддержки криптографических алгоритмов осуществляется при помощи языков программирования в виде модулей, библиотек или отдельных программ.

2. Аппаратные СКЗИ, в которых криптографические алгоритмы реализованы в микросхемах, процессорах или других аппаратных модулях.

3. Программно-аппаратные СКЗИ, состоящие из взаимосвязанной программной и аппаратной части с функциями криптографической защиты.

На сегодняшний день наиболее часто используемыми являются программные СКЗИ, поскольку они легче остальных внедряются в системы электронного документооборота. Аппаратные и программно-аппаратные средства являются более надёжными, но при этом дорогими и сложно внедряемыми.

Программные СКЗИ называются криптопровайдерами [17, с. 243]. Наиболее распространёнными и сертифицированными Федеральной службой безопасности России являются криптопровайдеры КриптоПро CSP [22] и ViPNet CSP [49]. Такие криптопровайдеры содержат реализацию алгоритмов линейки ГОСТ Р 34.10 и ГОСТ Р 34.11.

К аппаратным СКЗИ относятся смарт-карты и USB-ключи (токены), в которых работа криптографических алгоритмов выполняется встроенным микропроцессором [4, с. 132]. Для работы с электронной подписью и шифрованием на такие устройства производится запись сертификата и закрытого ключа.

Глава 2. Применение методов и средств криптографии в программных продуктах линейки «1С:Медицина»

В период пандемии востребованность использования электронной подписи и шифрования во многих системах электронного документооборота только возрастает. Одними из таких систем являются программные продукты линейки «1С:Медицина», которые предназначаются для автоматизации деятельности медицинских организаций. Под продуктами линейки «1С:Медицина» подразумеваются программы «1С:Медицина.Поликлиника», «1С:Медицина.Больница» и «1С:Медицина.Больничные».

2.1. Места использования электронной подписи и шифрования в конфигурациях «1С:Медицина»

Электронная подпись и шифрование в конфигурациях «1С:Медицина» используются в электронных листках нетрудоспособности, медицинских документах и при обмене данными с федеральными веб-сервисами.

Электронный листок нетрудоспособности

Электронный листок нетрудоспособности (ЭЛН) является аналогом обычного бумажного больничного листа и введён в использование с 1 июля 2017 года. Цель ЭЛН — подтвердить освобождение человека от работы по причине временной нетрудоспособности по электронным каналам связи.

В конфигурациях «1С:Медицина» поддерживается обмен данными с Фондом социального страхования (ФСС) для получения номеров, открытия, продления и закрытия ЭЛН, а также поиска ранее зарегистрированных сведений в базе ФСС. Обмен данными может осуществляться с тестовым или продуктивным сервисом ФСС. Тестовый сервис предназначен для отладки.

ЭЛН создаются в формате XML, а их подпись формируется по стандарту XMLDSig. При оформлении и после закрытия ЭЛН подписывается электронной подписью врача. При обмене с ФСС на документы ставится подпись медицинской организации.

Для обеспечения секретности отправляемых сообщений в конфигурациях «1С:Медицина» используется следующая схема шифрования:

1. при отправке сообщения в ФСС:

- В сообщение добавляется открытый ключ сертификата медицинской организации для того, чтобы получатель, в данном случае ФСС, в последующем мог зашифровать свой ответ.
- Данные шифруются секретным ключом по симметричному методу ГОСТ 28147-89.
- Ключ зашифровывается по асимметричному алгоритму шифрования, при помощи открытого ключа сертификата уполномоченного лица ФСС.

2. при получении ответа от ФСС, зашифрованного открытым ключом сертификата медицинской организации:

- Ключ для расшифровки вычисляется при помощи закрытого ключа медицинской организации.
- Сообщение расшифровывается по симметричному методу с использованием полученного секретного ключа.

Медицинский документ

Медицинские документы являются подтверждением различных медицинских услуг таких как приём у врача, проведение лабораторных исследований и т. д. Медицинские документы являются частью электронных медицинских карт, которые содержат персональные данные пациента, а также хранят историю болезней и информацию о назначенном лечении. Электронная подпись медицинских документов соответствует стандарту XMLDSig.

Федеральные веб-сервисы являются частью Единой государственной информационной системы здравоохранения (ЕГИСЗ).

К федеральным веб-сервисам относятся:

1. Реестр электронных медицинских документов (РЭМД) — осуществляет регистрацию медицинских документов и организует их обмен

между различными медицинскими информационными системами, в том числе из разных регионов.

2. Интегрированная электронная медицинская карта (ИЭМК) — используется для хранения данных пациентов, которые могут быть использованы различными медицинскими организациями.

3. Федеральная электронная регистратура (ФЭР) — предназначается для записи пациента на приём к врачу через Единый портал государственных услуг (ЕПГУ).

Электронная подпись в формате XMLDSig является обязательной при отправке сообщений в федеральный веб-сервис РЭМД. При взаимодействии с остальными сервисами электронная подпись является необязательной, поскольку передача данных происходит в закрытой сети.

Криптографические методы и средства конфигураций линейки «1С:Медицина»

Работа с электронной подписью и шифрованием в конфигурациях линейки «1С:Медицина» организована при помощи встроенного механизма криптографии платформы «1С:Предприятие», методов конфигурации «1С:Библиотека стандартных подсистем» и подсистемы «Электронный документооборот с контролирующими органами».

2.2. Механизм криптографии платформы «1С:Предприятие»

Для организации работы с электронной подписью и шифрованием в системе «1С:Предприятие» существует встроенный механизм криптографии, который не содержит реализации алгоритмов подписи, хеширования и шифрования, а предоставляет набор объектов и методов, позволяющих обеспечить взаимодействие с внешними модулями СКЗИ, которые реализуют криптографические алгоритмы.

Доступ к функциям криптопровайдера, установленного в операционной системе, осуществляется при помощи объекта МенеджерКриптографии. Для

его создания используется конструктор Новый МенеджерКриптографии(...) с тремя параметрами:

1. ИмяМодуляКриптографии — имя модуля криптографии, например, Crypto-Pro GOST R 34.10-2001 Cryptographic Service Provider.

2. ПутьМодуляКриптографии — путь к файлу-библиотеке, реализующей набор криптографических функций. Используется только в операционных системах семейства Linux.

3. ТипМодуляКриптографии — тип модуля, к которому относится криптопровайдер, например, для Crypto-Pro GOST R 34.10-2001 75.

Для работы с объектом МенеджерКриптографии указываются следующие свойства и используются методы:

1. Свойства:

- АлгоритмПодписи;
- АлгоритмХеширования;
- АлгоритмШифрования;
- ВключениеСертификатовВПодпись;
- ПарольДоступаКЗакрытомуКлючу.

2. Методы:

- Зашифровать(...);
- Подписать(...);
- ПолучитьИнформациюМодуляКриптографии(...);
- ПолучитьХранилищеСертификатов(...);
- ПроверитьПодпись(...);
- ПроверитьСертификат(...);
- Расшифровать(...).

Криптографический объект МенеджерКриптографии формирует подпись по стандарту PKCS#7 и возвращает её в виде двоичных данных или записывает в отдельный файл. Форматом зашифрованных данных является CMS [46], который базируется на PKCS#7.

Для работы с сертификатами используется объект СертификатКриптографии, который предоставляет доступ к следующей информации:

- срок действия и серийный номер сертификата;
- информация об организации, выдавшей сертификат;
- данные открытого ключа;
- отпечаток, который является хешем сертификата;
- информация о владельце сертификата.

Встроенные средства платформы «1С:Предприятие» не содержат средств для работы с подписью XMLDSig и шифрованием документов XML напрямую. Для этого используются внешние компоненты, входящие в состав конфигурации «1С:Библиотека стандартных подсистем» и подсистемы «Электронный документооборот с контролирующими органами».

2.3. Криптографические методы конфигурации «1С:Библиотека стандартных подсистем»

Для различных криптографических операций, в том числе с применением встроенных средств криптографии, используются методы подсистемы «Электронная подпись» [14], которая входит в состав конфигурации «1С:Библиотека стандартных подсистем», разработанной и поддерживаемой фирмой «1С».

Библиотека стандартных подсистем (БСП) — набор подсистем, реализующих базовую функциональность конфигураций и предназначенных для упрощения разработки на платформе «1С:Предприятие» [15]. БСП не является самостоятельной информационной системой. Каждая подсистема этой библиотеки используется как вспомогательный инструмент для разработки необходимой конфигурации. Благодаря применению единых модулей, была достигнута большая унификация прикладных решений, а это позволило программистам уменьшить время на их изучение.

Подсистема «Электронная подпись» предоставляет методы для:

1. настройки взаимодействия конфигурации с криптопровайдерами, установленными в системе;

2. использования сертификатов электронной подписи, которые установлены на компьютере;

3. различных криптографических операций с объектами МенеджерКриптографии и СертификатКриптографии.

Также подсистема имеет в своём составе внешнюю компоненту XMLDSIG, которая используется для подписания и проверки подписи XML-документов. Необходимость использования внешних компонент заключается в решении задач, которые очень сложно или невозможно реализовать на встроенном в «1С:Предприятие» языке программирования [1]. В конфигурациях 1С внешние компоненты обычно хранятся в виде макета.

Подписание XML-документов в БСП может выполняться на стороне клиента или сервера при помощи методов компоненты XMLDSIG. Последовательность действий при подписании показана на рис. 1.

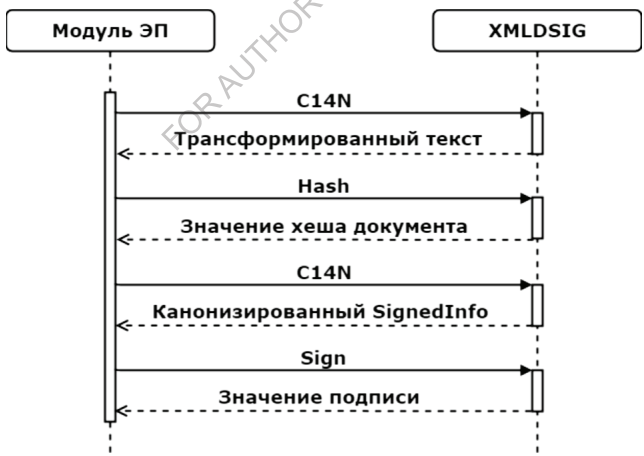

Рисунок 1. Процесс подписания XML-документа в БСП

Вызовы методов компоненты реализованы в следующих общих модулях:

- ЭлектроннаяПодписьСлужебныйКлиент — при формировании подписи на стороне клиента;
- ЭлектроннаяПодписьСлужебный — при выполнении подписания на стороне сервера.

Перед формированием подписи документ XML подвергается трансформации при помощи метода C14N() компоненты XMLDSIG. Трансформация требуется из-за необходимости изменения порядка или фильтрации некоторых частей документа. Стоит отметить, что в данном случае такие преобразования осуществляется по алгоритму канонизации.

Для выполнения трансформации XML-документа в метод C14N() передаются следующие значения параметров:

- КонвертSOAP — XML-документ, в котором будет выполнена трансформация;
- XPath — указание элементов XML-документа, которые требуется преобразовать.

После выполнения трансформации для XML-документа генерируется значение хеша при помощи функции Hash компоненты XMLDSIG. BHash передаются следующие параметры:

- ТрансформированныйТекстXML — документ после процедуры трансформации;
- OIDАлгоритмаХеширования — идентификатор алгоритма хеширования;
- ТипКриптопровайдера — тип модуля криптопровайдера, который установлен в операционной системе.

Перед формированием подписи и после установки хеша в документе XML канонизируется элемент SignedInfo, определяющий данные и алгоритмы для формирования подписи, при помощи метода C14N() компоненты XMLDSIG. Для канонизации в C14N() нужно передавать такие значения параметров:

• КонвертSOAP — XML-документ, в котором нужно выполнить канонизацию;

• XPath — указание канонизируемой области, в данном случае тегаSignedInfo.

После приведения документа к каноническому виду и формирования хеша осуществляется подписание функцией Sign компоненты XMLDSIG. Передаваемые параметры в функцию:

• КаноникализированныйТекстXMLSignedInfo — документ с канонизированной областью SignedInfo;

• СертификатКриптографииBase64 — строка, закодированная по алгоритму base64, который преобразует данные сертификата в последовательность из:

– символов латинского алфавита AZ и az,

– цифр 09,

– символов s+» и s/»;

• ПарольДоступаКЗакрытомуКлючу — пароль от контейнера закрытого ключа.

Несмотря на широкую и достаточно универсальную функциональность БСП, иногда возникает необходимость доработки или замены некоторых её методов при их внедрении в различные типовые конфигурации. Например, в конфигурациях линейки «1С:Медицина» для канонизации XML-документов вместо метода компоненты C14N(...) используется метод платформы «1С:Предприятие» КанонизироватьВСтроку(...) объекта ПреобразованиеККаноническомуXML. При этом в методе указывается параметр КаноническийXML, который определяет алгоритм приведения документа к каноническому виду. Отличие состоит в том, что компонента канонизирует документ по стандарту Exclusive XML Canonicalization, а метод платформы КанонизироватьВСтроку(...) с указанным типом КаноническийXML — по стандарту Canonical XML.

2.4. Методы подсистемы «Электронный документооборот с контролирующими органами»

При обмене с ФСС для организации процессов шифрования и расшифровки используются методы и компонента подсистемы «Электронный документооборот с контролирующими органами» (ЭДКО), которая разработана фирмой «1С». Методы подсистемы ЭДКО позволяют:

1. выполнять шифрование и расшифровку сообщений с помощью подключаемой внешней компоненты;

2. производить проверку сертификата открытого ключа;

3. осуществлять поиск сертификата по его отпечатку в хранилище;

4. преобразовывать сертификат в строку base64;

5. получать свойства криптопровайдеров, поддерживающих алгоритмы линейки ГОСТ Р.

Для шифрования и расшифровки сообщений в конфигурациях линейки «1С:Медицина» используется компонента шифрования, которая имеет название «Компонента обмена».

Схема обмена данными с ФСС с применением шифрования и расшифровки представлена на рис. 2.

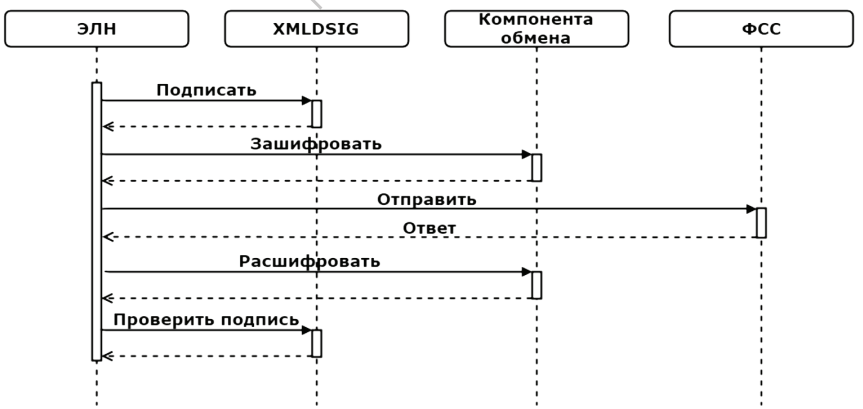

Рисунок 2. Обмен данными с ФСС

Сначала ЭЛН подписывается сертификатом медицинской организации (МО) с использованием компоненты XMLDSIG. Далее подписанный ЭЛН зашифровывается при помощи метода Зашифровать(...) компоненты «Компонента обмена» со следующими параметрами:

- ИсходныйФайл — путь к файлу, который должен быть зашифрован;
- СертификатОтправителяСерийныйНомер — серийный номер сертификата МО;
- СертификатОтправителяПоставщик — информация об издателе сертификата МО;
- СертификатПолучателяСерийныйНомер — серийный номер сертификата ФСС;
- СертификатПолучателяПоставщик — информация об издателе сертификата ФСС;
- ИмяКаталогаРезультата — путь к каталогу, в который будет помещён зашифрованный файл;
- НазваниеАлгоритмаШифрования — название алгоритма шифрования (по умолчанию используется ГОСТ 28147-89);
- АлгоритмКлюча — ID алгоритма генерации секретного (симметричного) ключа (для ГОСТ 28147-89 указывается 26142);
- Режим — режим шифрования данных (для ГОСТ 28147-89 используется блочное шифрование).

Для шифрования сообщения генерируется симметричный ключ и случайным образом выбирается вектор инициализации, который используется для изменения значения симметричного ключа. Далее, уже модифицированным ключом зашифровывается сообщение по симметричному методу.

Если не использовать вектор инициализации, то два одинаковых сообщения после процедуры шифрования одним и тем же ключом будут представлять собой одинаковую последовательность символов. В случае перехвата подобного сообщения злоумышленнику будет значительно проще

подобрать шаблон для его расшифровки, так как такой текст мог уже передаваться по каналу связи.

Для шифрования симметричного ключа «Компонента обмена» использует асимметричный алгоритм Диффи-Хеллмана, принцип работы которого описывается в [30], и открытый ключ сертификата ФСС.

Следует отметить, что в асимметричном алгоритме для шифрования или расшифровки симметричного ключа используются ключи по стандартам линейки ГОСТ Р 34.10. Поэтому для выполнения операции шифрования или расшифровки компоненту предварительно требуется проинициализировать по алгоритму линейки ГОСТ Р 34.10 при помощи криптопровайдера с использованием метода этой компоненты СоздатьМенеджераКриптографии(...) с параметрами:

- Имя — название криптопровайдера;
- Путь — путь к файлу-библиотеке криптопровайдера (используется только в Linux);
- Тип — тип модуля криптопровайдера.

При формировании зашифрованного сообщения в него требуется поместить открытый ключ сертификата МО для того, чтобы ФСС смог зашифровать свой ответ. После отправки сообщения от ФСС приходит ответ, который расшифровывается при помощи метода компонентыРасшифроватьФайл(...) со следующими параметрами:

- СертификатПолучателяСерийныйНомер — серийный номер сертификата МО;
- СертификатПолучателяПоставщик — информация об издателе сертификата МО;
- ЗашифрованныйФайл — путь к файлу, который нужно расшифровать;
- ИмяФайлаРезультата — путь к файлу, в который нужно поместить расшифрованное сообщение;

• ОткрытыйКлюч — открытый ключ сертификата МО, которым зашифровано сообщение от ФСС;

• СимметричныйКлюч — ключ для расшифровки сообщения симметричным методом ГОСТ 28147-89;

• ИнициализационныйВектор — числовое значение, при помощи которого был модифицирован симметричный ключ.

При расшифровке происходит обращение к закрытому ключу сертификата МО, который находится во внешнем модуле криптографии. Пароль от закрытого ключа вводится не в «1С», а в форме внешнего модуля криптографии. Например, такая форма криптопровайдера ViPNet показана на рис. 3.

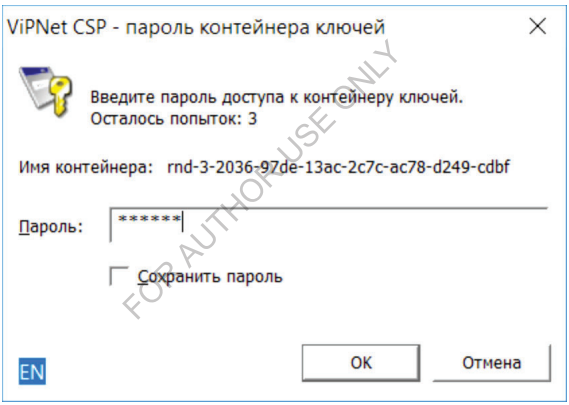

Рисунок 3. Форма ввода пароля от контейнера закрытого ключа

После расшифровки проверяется подпись сообщения при помощи компоненты XMLDSIG.

Глава 3. Реализация поддержки криптографических алгоритмов ГОСТ Р 34.10-2012 и ГОСТ Р 34.11-2012

3.1. Определение этапов реализации

Ранее, при использовании алгоритмов подписи ГОСТ Р 34.10-2001 и хеширования ГОСТ Р 34.11-94, все параметры задавались в явном виде в программном коде конфигураций линейки «1С:Медицина». В 2019 году может использоваться один из трёх алгоритмов подписи: ГОСТ Р 34.10-2001, ГОСТ Р 34.10-2012 в двух вариантах — с длиной хеша 256 или 512 бит. Данные алгоритма подписи содержатся в сертификате открытого ключа. Следовательно, необходимо программно определять алгоритм выбранного сертификата и использовать соответствующий ему алгоритм хеширования. Для обмена данными ЭЛН с ФСС должно быть настроено асимметричное шифрование, в котором используются ключи по стандарту линейки ГОСТ Р 34.10 из сертификата электронной подписи.

Также нужно предусмотреть, чтобы ГОСТ Р 34.10-2001 и ГОСТ Р 34.11-94 можно было наиболее просто снять с поддержки с 1 января 2020 года.

Таким образом, реализация поддержки ГОСТ Р 34.10-2012 и ГОСТ Р 34.11-2012 состоит из следующих этапов:

1. удалить все предопределённые шаблоны по алгоритмам ГОСТ Р 34.102001 и ГОСТ Р 34.11-94 из программного кода конфигураций;
2. организовать получение алгоритма электронной подписи из сертификата открытого ключа;
3. определить шаблоны для заполнения параметрами алгоритмов подписи и хеширования;
4. добавить поддержку криптопровайдеров по ГОСТам 2012 года;
5. настроить шифрование с использованием алгоритма ГОСТ Р 34.10-2012.

3.2. Добавление подписания по ГОСТ Р 34.10-2012 и формирования хеша по ГОСТ Р 34.11-2012

Реализацию подписания по ГОСТ Р 34.10-2012 и формирования хеша по ГОСТ Р 34.11-2012 рассмотрим на примере конфигурации «1С:Медицина. Поликлиника». На момент начала выполнения задачи актуальной являлась версия 2.1.4. Необходимо было добавить возможность подписывать ЭЛН, медицинские документы и сообщения, отправляемые в РЭМД, по алгоритму ГОСТ Р 34.102012, используя при этом алгоритм хеширования по ГОСТ Р 34.11-2012. Также требовалось сохранить поддержку подписи по ГОСТ Р 34.10-2001 и хеширования по ГОСТ Р 34.11-94.

Процедура подписания в конфигурациях «1С:Медицина» может осуществляться на клиенте или на сервере средствами БСП. Общая схема подписания для XML-документов, ранее используемая в конфигурациях линейки «1С:Медицина», представлена на рис. 4.

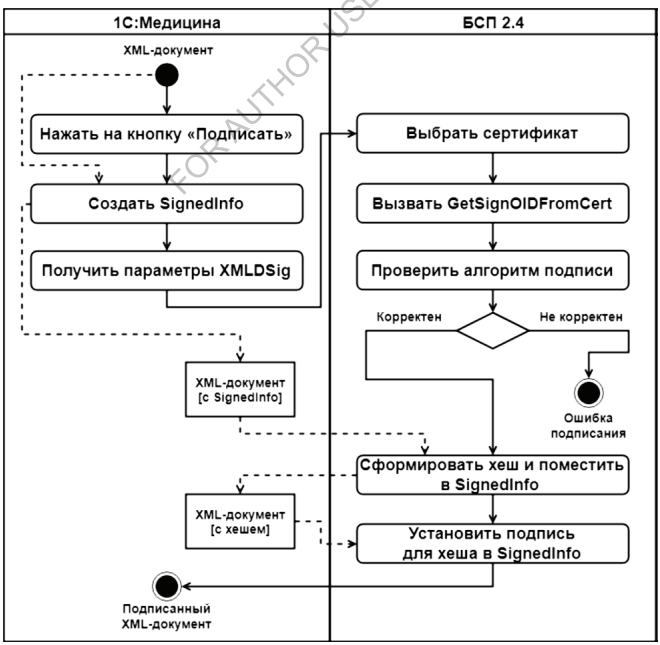

Рисунок 4. Схема подписания в конфигурациях «1С:Медицина»

В такой схеме при вызове метода подписания сначала формируется XML-шаблон SignedInfo, в который далее будут записаны значения хеша и подписи. Затем создаётся структура XMLDSig, содержащая параметры алгоритмов по ГОСТ Р 34.10-2001 и ГОСТ Р 34.11-94. После этого выбирается сертификат для подписания. Далее вызывается метод GetSignOIDFromCert(...) компоненты XMLDSIG, который позволяет извлекать из сертификата идентификатор алгоритма подписи — OID (Object Identifier). В параметры GetSignOIDFromCert(...) необходимо передавать компоненту XMLDSIG, которая подключается методом ПодключитьВнешнююКомпоненту(...) или НачатьПодключениеВнешнейКомпоненты(...), и сертификат в формате base64. Метод извлечения алгоритма из сертификата на сервере представлен в листинге 2.

Листинг 2. Метод извлечения алгоритма из сертификата

```
Функция GetSignOIDFromCert(ОбъектКомпоненты,
СертификатКриптографииBase64)
    Попытка
        OIDАлгоритмаПодписи =
            ОбъектКомпоненты.GetSignOIDFromCert(СертификатКриптографии
Base64);
        Исключение
        ВызватьИсключение
        НСтр("ru = 'Ошибка вызова метода GetSignOIDFromCert компоненты
XMLDSig.'")
          + Символы.ПС +
ПодробноеПредставлениеОшибки(ИнформацияОбОшибке());
        КонецПопытки;
        Если OIDАлгоритмаПодписи = Неопределено Тогда
        ВызватьИсключение
```

```
        НСтр("ru = 'Ошибка вызова метода GetSignOIDFromCert компоненты
XMLDSig.'")
                + Символы.ПС + ОбъектКомпоненты.GetLastError();
        КонецЕсли;
        Возврат OIDАлгоритмаПодписи;
КонецФункции
```

Метод GetSignOIDFromCert(...) ранее вызывался только для того, чтобы определять правильно ли задан OID алгоритма подписания. Если OID алгоритма сертификата совпадает с алгоритмом подписи, указанном в параметрах XMLDSig, то XML-документ подвергается процедуре канонизации и затем для него будут сформированы хеш и электронная подпись. В случае, когда алгоритм сертификата отличается от алгоритма в структуре XMLDSig, подписание не будет выполнено и операция завершится с уведомлением об ошибке. Такая проверка алгоритма, выполняемая при подписании на клиенте, показана в листинге 3.

Листинг 3. Проверка алгоритма подписи

```
Процедура Подписание_ПослеВыполненияGetSignOIDFromCert(
    OIDАлгоритмаПодписи, Параметры, Контекст) Экспорт
        Если
        OIDАлгоритмаПодписи <>
Контекст.ДанныеАлгоритмаПодписания.OIDАлгоритмаПодписи
        Тогда
        ЗавершитьОперациюСОшибкой(...);
        Иначе
        XMLDSig.КанонизироватьXML(...);
    КонецЕсли;
КонецПроцедуры
```

Поскольку данные алгоритмов по ГОСТ Р 34.10-2001 и ГОСТ Р 34.11-94 строго задавались в программном коде ещё до выбора сертификата, не было возможности использовать какие-либо другие алгоритмы.

Для того чтобы добавить поддержку ГОСТ Р 34.10-2012 и ГОСТ Р 34.112012, необходимо было в первую очередь удалить заранее определённые параметры алгоритмов из шаблона SignedInfo и структуры XMLDSig. Затем проверять допустим ли алгоритм, извлечённый из сертификата, для создания подписи. Далее добавить определение параметров для формирования электронной подписи, в которые должен входить соответствующий алгоритм хеширования. Затем заполнять этими параметрами шаблон SignedInfo и структуру XMLDSig. Только после этих операций формировать хеш и подпись документа. Таким образом, нужно было изменить схему подписания (см. рис. 4) так, как показано на рис. 5.

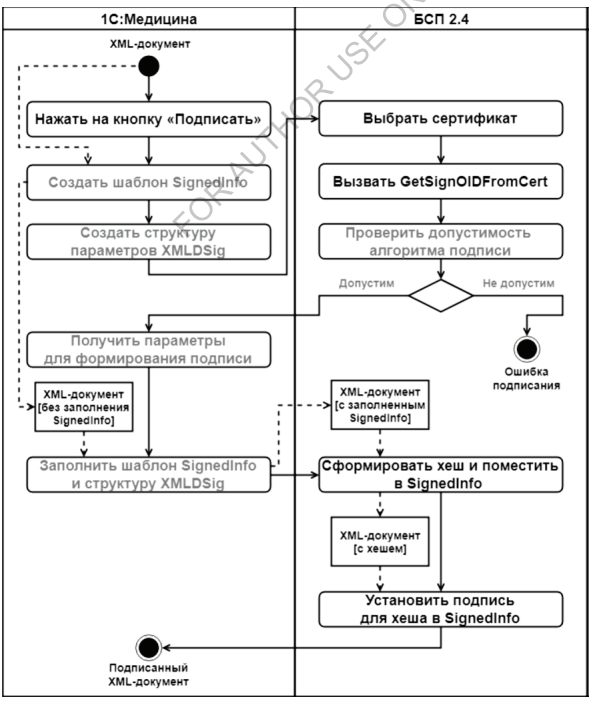

Рисунок 5. Схема подписания с использованием алгоритма из сертификата

Сначала из программного кода были удалены все предопределённые параметры алгоритмов ГОСТ Р 34.10-2001 и ГОСТ Р 34.11-94, которые представлены в листинге 4.

Листинг 4. Параметры XMLDSig

```
ПараметрыXMLDSig_.ИмяАлгоритмаПодписи = "GOST R 34.10-2001";
ПараметрыXMLDSig_.OIDАлгоритмаПодписи = "1.2.643.2.2.3"; // ГОСТ R 34.10-2001.
ПараметрыXMLDSig_.ИмяАлгоритмаХеширования = "GOST R 34.11-94";
ПараметрыXMLDSig_.OIDАлгоритмаХеширования = "1.2.643.2.2.9"; // ГОСТ R 34.11-94.
```

Заранее определённые параметры XMLDSig были удалены из программного кода следующих функций:

- ПолучитьПараметрыXMLDSig(Знач ПодписываемыеДанные) — для подписи ЭЛН;
- ПолучитьПараметрыXMLDSigПоSignedInfo(ДокументDOM, НомерВхождения) — для проверки подписи ЭЛН;
- ПолучитьПараметрыXMLDSig(ИдПодписи) — для подписания медицинского документа;
- ПодписатьSoapСообщение(ПодписываемыйXml, Знач Сертификат = Неопределено) — для подписания сообщения, отправляемого в РЭМД.

Ранее в коде также задавались URI (Uniform Resource Identifier) алгоритмов подписи и хеширования в элементах SignatureMethod и DigestMethod XMLшаблона для SignedInfo как показано в листинге 5.

Листинг 5. Заданные URI алгоритмов подписи и хеширования

```
SignedInfo_ =
"<SignedInfo xmlns=""http://www.w3.org/2000/09/xmldsig#"">
| ...
| <SignatureMethod Algorithm=
```

```
| ""http://www.w3.org/2001/04/xmldsig-more#gostr34102001-gostr3411"">
| <Reference>
| ...
| <DigestMethod Algorithm=
| ""http://www.w3.org/2001/04/xmldsig-more#gostr3411""/>
| ...
| </Reference>
|</SignedInfo>";
```

Для дальнейшей подстановки необходимых URI были удалены предопределённые значения алгоритмов подписи и хеширования, а также добавлены параметры %SignatureMethod% и %DigestMethod% как показано в листинге 6.

Листинг 6. Шаблон SignedInfo

```
SignedInfo_ =
    "<SignedInfo xmlns=""http://www.w3.org/2000/09/xmldsig#"">
    | ...
    | <SignatureMethod Algorithm=""%SignatureMethod%"">
    | <Reference>
    | ...
    | <DigestMethod Algorithm=""%DigestMethod%""/>
    | ...
    | </Reference>
    |</SignedInfo>";
```


Далее в общем модуле XMLDSig была написана функция для получения параметров подписания по OID алгоритма, извлечённому из сертификата. Фрагмент функции представлен в листинге 7.

Листинг 7. Получение параметров для формирования подписи

```
Функция ПолучитьПараметрыДляФормированияЭП(OIDАлгоритмаПодписи)
Экспорт
    Параметры_ = Новый Структура;
    ...
    Если OIDАлгоритмаПодписи = "1.2.643.2.2.3" Тогда
        Параметры_.ИмяАлгоритмаПодписи = "GOST R 34.10-2001";
        Параметры_.OIDАлгоритмаПодписи = OIDАлгоритмаПодписи;
        Параметры_.URIАлгоритмаПодписи =
          "urn:ietf:params:xml:ns:cpxmlsec:algorithms:gostr34102001-
gostr3411";
        Параметры_.ИмяАлгоритмаХеширования = "GOST R 34.11-94";
        Параметры_.OIDАлгоритмаХеширования = "1.2.643.2.2.9";
        Параметры_.URIАлгоритмаХеширования =
          "urn:ietf:params:xml:ns:cpxmlsec:algorithms:gostr3411";
    ИначеЕсли OIDАлгоритмаПодписи = "1.2.643.7.1.1.3.2" Тогда
      ...
    ИначеЕсли OIDАлгоритмаПодписи = "1.2.643.7.1.1.3.3" Тогда
      ...
    КонецЕсли;
    Возврат Параметры_;
КонецФункции
```


Все параметры используемых алгоритмов подписи и хеширования представлены в прил. 1.

Для того чтобы в 2020 году можно было легко снять с поддержки стандарт ГОСТ Р 34.10-2001, в структуру, возвращаемую функцией ПараметрыXMLDSig(), был добавлен массив допустимых OID алгоритмов как показано в листинге 8.

Листинг 8. Массив допустимых алгоритмов

```
Функция ПараметрыXMLDSig() Экспорт
```

```
ДанныеАлгоритмаПодписания_ = Новый Структура;
МассивДопустимыхАлгоритмов_ = Новый Массив;
МассивДопустимыхАлгоритмов_.Добавить("1.2.643.2.2.3");
МассивДопустимыхАлгоритмов_.Добавить("1.2.643.7.1.1.3.2");
МассивДопустимыхАлгоритмов_.Добавить("1.2.643.7.1.1.3.3");
ДанныеАлгоритмаПодписания_.Вставить(
"ДопустимыеАлгоритмыПодписи", МассивДопустимыхАлгоритмов_);
Возврат ДанныеАлгоритмаПодписания_;
КонецФункции
```

Далее в процедуру Подписание_ПослеВыполненияGetSignOIDFromCert(...), которая выполняется в процессе подписания на клиенте, была добавлена проверка того, что алгоритм, извлекаемый из сертификата, является допустимым для формирования подписи. Также организовано получение и заполнение параметров структуры XMLDSig. Реализация представлена в листинге 9.

Листинг 9. Получение параметров и заполнение структуры XMLDSig

```
Процедура Подписание_ПослеВыполненияGetSignOIDFromCert(
OIDАлгоритмаПодписи, Параметры, Контекст) Экспорт
...
ДанныеАлгоритмаПодписания_ = Контекст.ДанныеАлгоритмаПодписания;
ДопустимыеАлгоритмыПодписи_ =
ДанныеАлгоритмаПодписания_.ДопустимыеАлгоритмыПодписи;
Если ИндексАлгоритмаВМассиве_ = Неопределено Тогда
ЗавершитьОперациюСОшибкой(...);
Иначе
// Получение параметров алгоритмов подписи и хеширования.
ПараметрыАлгоритмов_ =
```

```
XMLDSig.ПолучитьПараметрыДляФормированияЭП(OIDАлгоритмаП
одписи);
// Заполнение структуры параметров.
ЗаполнитьЗначенияСвойств(
Контекст.ДанныеАлгоритмаПодписания, ПараметрыАлгоритмов_);
...
КонецЕсли;
КонецПроцедуры
```

Параметр Контекст заполняется ранее в процедуре НачатьПодписание(...) и содержит XML-документ, структуру для заполнения данными алгоритмов подписи и хеширования (структуру XMLDSig), сертификат открытого ключа в формате base64 и информацию о криптопровайдере, при помощи которого осуществляется подписание.

Для заполнения элемента SignedInfo соответствующими URI алгоритмов подписи и хеширования был добавлен код из листинга 10.

Листинг 10. Добавление URI алгоритмов подписи и хеширования

```
Процедура Подписание_ПослеВыполненияHash_ПодписываемыйТег(
DigestValue, Параметры, Контекст) Экспорт
...
ДанныеАлгоритмаПодписания_ = Контекст.ДанныеАлгоритмаПодписания;
SignatureMethod_ = ДанныеАлгоритмаПодписания_.URIАлгоритмаПодписи;
Контекст.КонвертSOAP = СтрЗаменить(
Контекст.КонвертSOAP, "%SignatureMethod%", SignatureMethod_);
DigestMethod_ =
ДанныеАлгоритмаПодписания_.URIАлгоритмаХеширования;
Контекст.КонвертSOAP = СтрЗаменить(
Контекст.КонвертSOAP, "%DigestMethod%", DigestMethod_);
...
КонецПроцедуры
```

Заполняемая переменная КонвертSOAP может являться обычным подписываемым XML-документом или сообщением, которое формируется для отправки. Обозначение SOAP (Simple Object Access Protocol) подразумевает определение протокола [47], который используется для обмена структурированной информацией в формате XML.

Далее такая же схема получения параметров алгоритмов и заполнения соответствующих шаблонов была реализована при проверке электронной подписи XML-документов.

Затем подписание с использованием алгоритма из сертификата было добавлено на сервер в функцию Подписать(...). Внесённые изменения представлены в листинге 11.

Листинг 11. Подписание на сервере

```
Функция Подписать(Знач КонвертSOAP, ДанныеАлгоритмаПодписания,
    СертификатКриптографии, МенеджерКриптографии) Экспорт
    ...
    Сертификат_ = СертификатКриптографии.Выгрузить();
    СертификатКриптографииBase64_ =
    ЭлектроннаяПодписьСлужебныйКлиентСервер.СертификатКриптогра
фииBase64(
    Сертификат_);
    OIDАлгоритмаПодписи_ =
    GetSignOIDFromCert(ОбъектКомпоненты,
СертификатКриптографииBase64_);
    ДопустимыеАлгоритмыПодписи_ =
    ДанныеАлгоритмаПодписания.ДопустимыеАлгоритмыПодписи;
    ИндексАлгоритмаВМассиве_ =
    ДопустимыеАлгоритмыПодписи_.Найти(OIDАлгоритмаПодписи_);
    Если ИндексАлгоритмаВМассиве_ <> Неопределено Тогда
    Параметры_ =
```

```
XMLDSig.ПолучитьПараметрыДляФормированияЭП(OIDАлгоритмаПодписи_);
ЗаполнитьЗначенияСвойств(ДанныеАлгоритмаПодписания, Параметры_);
SignatureMethod_ = ДанныеАлгоритмаПодписания.URIАлгоритмаПодписи;
КонвертSOAP =
СтрЗаменить(КонвертSOAP, "%SignatureMethod%", SignatureMethod_);
DigestMethod_ = ДанныеАлгоритмаПодписания.URIАлгоритмаХеширования;
КонвертSOAP = СтрЗаменить(КонвертSOAP, "%DigestMethod%", DigestMethod_);
КонецЕсли;
...
КонецФункции
```


Такое же определение алгоритма сертификата и заполнение параметров было добавлено в серверную функцию ПроверитьПодпись(...).

Для организации работы с криптопровайдерами, поддерживающими алгоритмы ГОСТ Р 34.10-2012 и ГОСТ Р 34.11-2012, были добавлены настройки в модуль менеджера справочника ПрограммыЭлектроннойПодписиИШифрования. Например, добавленные настройки для ViPNet CSP представлены в листинге 12.

Листинг 12. Настройки криптопровайдеров

```
Функция ПоставляемыеНастройкиПрограмм() Экспорт
Настройки = Новый ТаблицаЗначений;
...
// ViPNet CSP (ГОСТ 2012)
Настройка = Настройки.Добавить();
Настройка.Представление = НСтр("ru = 'ViPNet CSP (ГОСТ 2012)'");
```

```
Настройка.ИмяПрограммы = "Infotecs GOST 2012/512 Cryptographic Service
Provider";
Настройка.ТипПрограммы = 77;
Настройка.АлгоритмПодписи = "GOST 34.10-2012 256";
Настройка.АлгоритмХеширования = "GOST 34.11-2012 256";
Настройка.АлгоритмШифрования = "GOST 28147-89";
...
КонецФункции
```

Все настройки криптопровайдеров по ГОСТ Р 34.10-2012 были взяты из БСП версии 3.0.1 [16], которая являлась самой актуальной на момент выполнения задач исследования.

3.3. Настройка шифрования и расшифровки сообщений с использованием ГОСТ Р 34.10-2012

После реализации подписания по ГОСТ Р 34.10-2012 необходимо было настроить асимметричное шифрование, которое используется в конфигурациях линейки «1С:Медицина» для обмена данными ЭЛН с ФСС.

Ранее компонента, при помощи которой осуществляется шифрование и расшифровка, поддерживала только ГОСТ Р 34.10-2001. При этом параметры алгоритмов используемых сертификатов ФСС и МО не определялись, так как компонента автоматически инициализировалась по ГОСТ Р 34.10-2001. Для возможности выполнять операции с использованием ГОСТ Р 34.10-2012 у разработчиков фирмы «1С» была запрошена новая версия компоненты. Такая компонента поддерживает асимметричное шифрование с применением ключей по ГОСТ Р 34.10-2001 и ГОСТ Р 34.10-2012.

Также перед отправкой в ФСС необходимо добавлять в XML-сообщение зашифрованный симметричный ключ с параметрами вектора инициализации и открытого ключа сертификата ФСС. Для этого сначала

формируется шаблон в виде массива, в который затем добавляются значения ключей и вектора. Такой массив после формирования преобразуется в строку формата base64 и вставляется в XML-сообщение. Разбор симметричного ключа из зашифрованного сообщения от ФСС также осуществляется по определённому шаблону. Ранее такие шаблоны были определены только для ГОСТ Р 34.10-2001.

Таким образом, для настройки асимметричного шифрования при обмене с ФСС было необходимо:

1. добавить определение алгоритмов сертификатов ФСС и МО перед выполнением процедур шифрования и расшифровки;

2. реализовать инициализацию компоненты, шифрование и расшифровку по указанному алгоритму;

3. добавить схему формирования зашифрованного симметричного ключа для вставки в XML при использовании ГОСТ Р 34.10-2012;

4. реализовать разбор зашифрованного симметричного ключа из XML при использовании ГОСТ Р 34.10-2012.

Схема шифрования и расшифровки сообщений при обмене данными с ФСС, в которой применялся только алгоритм ГОСТ Р 34.10-2001, представлена на рис. 6.

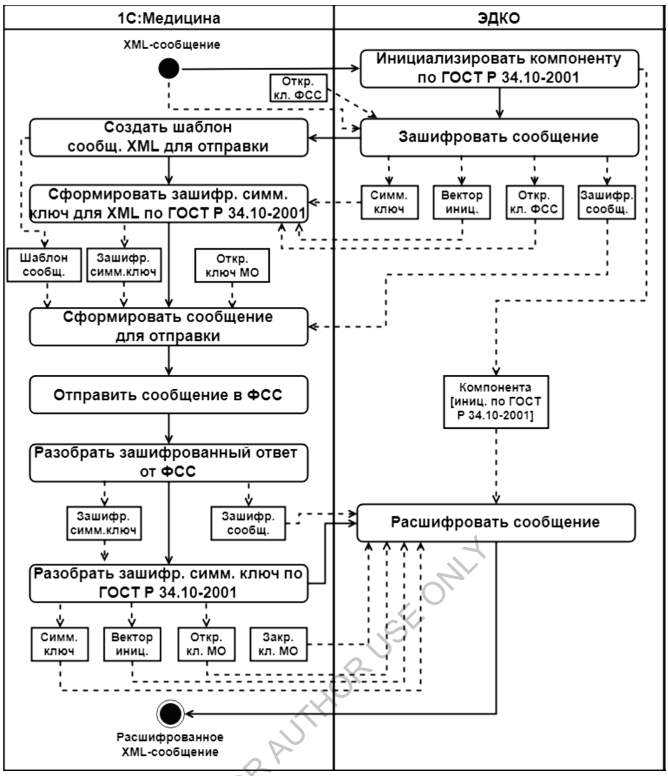

Рисунок 6. Схема шифрования и расшифровки сообщений

В такой схеме сначала инициализируется компонента по алгоритму ГОСТ Р 34.10-2001. Затем эта компонента шифрует по симметричному методу XMLсообщение, а сгенерированный симметричный ключ зашифровывает при помощи открытого ключа сертификата ФСС. В результате шифрования компонента возвращает четыре объекта: зашифрованный симметричный ключ, вектор инициализации, открытый ключ сертификата ФСС и зашифрованное сообщение. Значения всех этих объектов представляются в формате base64.

После шифрования создаётся шаблон XML-сообщения для последующей отправки, в который далее будет добавлено зашифрованное компонентой сообщение, зашифрованный симметричный ключ и сертификат

МО. Далее формируется массив, в который сначала вносится значение симметричного ключа, зашифрованного компонентой, а затем добавляются вектор инициализации и открытый ключ сертификата ФСС, при помощи которого компонента зашифровала симметричный ключ. Затем массив преобразовывается в строку формата base64. Такая строка является зашифрованным симметричным ключом для вставки в XML и содержит все параметры, которые позволят ФСС расшифровать сообщение.

Параметры открытого ключа сертификата ФСС необходимо добавлять в сообщение ввиду того, что необходима проверка соответствия открытого ключа закрытому перед расшифровкой. Это связано с тем, что ФСС может предоставлять несколько сертификатов для шифрования и поэтому нужно правильно выбирать для расшифровки необходимый закрытый ключ из имеющихся.

Далее в созданный шаблон сообщения для отправки добавляются: зашифрованное компонентой сообщение, зашифрованный симметричный ключ и сертификат МО, открытый ключ которого будет необходим для шифрования ответа от ФСС. После этих операций сообщение отправляется в ФСС.

Получив ответ от ФСС, происходит разбор принятого сообщения. Результатом такого разбора является вложенное зашифрованное сообщение и зашифрованный симметричный ключ в формате base64. Этот ключ затем преобразовывается в массив, который разбивается на три объекта: симметричный ключ, зашифрованный открытым ключом МО, вектор инициализации и открытый ключ сертификата МО, при помощи которого был зашифрован симметричный ключ. После этих операций компонента, инициализированная ранее по ГОСТ Р 34.10-2001, производит вычисление симметричного ключа закрытым ключом МО, а затем выполняет расшифровку сообщения при помощи вычисленного симметричного ключа.

Следует отметить, что передача открытого ключа из сообщения в метод расшифровки необходима из-за того, что компонента обращается к

закрытому ключу по соответствующему открытому. Для добавления возможности использовать в асимметричном шифровании ключи по ГОСТ Р 34.10-2012 схема шифрования и расшифровки (см. рис. 6) была изменена так, как показано на рис. 7.

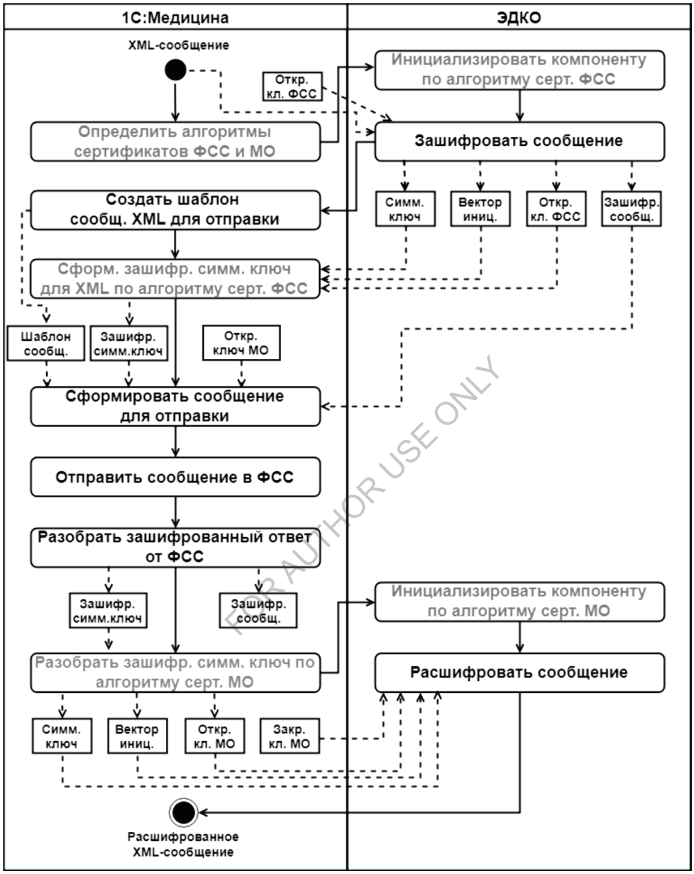

Рисунок 7. Схема обмена с ФСС с применением ключей сертификатов

В первую очередь была написана функцияПолучитьOIDАлгоритмаПодписи(...) в модуле ЭлектроннаяПодписьСлужебный для определения алгоритмов электронной подписи сертификатов ФСС и МО. В такой функции происходит обращение

к методу GetSignOIDFromCert(...) компоненты XMLDSIG. Код функции представлен в листинге 13.

Листинг 13. Получение OID алгоритма подписи

```
Функция ПолучитьOIDАлгоритмаПодписи(СертификатBase64) Экспорт
    ОбъектКомпоненты_ = ПодключитьВнешнююКомпонентуXMLDSig();
    OIDАлгоритмаПодписи_ = GetSignOIDFromCert(ОбъектКомпоненты_,СертификатBase64);
    Возврат OIDАлгоритмаПодписи_;
КонецФункции
```

Обращение с клиента к ПолучитьOIDАлгоритмаПодписи(...) организовано через одноимённую функцию в общем модуле ЭлектроннаяПодписьСлужебныйВызовСервера.

Поскольку в процессе инициализации компоненты шифрования необходимо использовать не идентификаторы алгоритмов, а их названия, была определена функция из листинга 14 в общем модуле XMLDSig.

Листинг 14. Определение алгоритмов сертификатов

```
Функция ПолучитьНазванияАлгоритмовПодписи(СертификатыBase64) Экспорт
    АлгоритмыПодписи_ = Новый Структура;
    Для Каждого Элемент_ Из СертификатыBase64 Цикл
    OIDАлгоритмаПодписи_ = ЭлектроннаяПодписьСлужебныйВызовСервера.ПолучитьOIDАлгоритмаПодписи(
    Элемент_.Значение);
    Если OIDАлгоритмаПодписи_ = "1.2.643.2.2.3" Тогда
    АлгоритмыПодписи_.Вставить(Элемент_.Ключ, "GOST R 34.10-2001");
    ИначеЕсли OIDАлгоритмаПодписи_ = "1.2.643.7.1.1.3.2" Тогда
    АлгоритмыПодписи_.Вставить(Элемент_.Ключ, "GOST R 34.10-2012-256");
```

```
        ИначеЕсли OIDАлгоритмаПодписи_ = "1.2.643.7.1.1.3.3" Тогда
        АлгоритмыПодписи_.Вставить(Элемент_.Ключ, "GOST R 34.10-2012-
512");
        КонецЕсли;
        КонецЦикла;
        Возврат АлгоритмыПодписи_;
КонецФункции
```

Затем в функцию ШифрованиеОтправкаРасшифровка_ОбработкаЭтапа(...), в которой производится обращение к методам подсистемы ЭДКО, был добавлен код из листинга 15 для получения названий алгоритмов подписи сертификатов ФСС и МО.

Листинг 15. Получение названий алгоритмов подписи

```
Функция ШифрованиеОтправкаРасшифровка_ОбработкаЭтапа(
        РезультатЭтапа, ТекущееСостояние) Экспорт
        ...
        СертификатыBase64_ = Новый Структура;
        СертификатыBase64_.Вставить(
        "АлгоритмПодписиСертификатаФСС",
СертификатФСС_.СертификатBase64);
        СертификатыBase64_.Вставить(
        "АлгоритмПодписиСертификатаМО",
СертификатМО_.СертификатBase64);
        АлгоритмыПодписи_ =
        XMLDSig.ПолучитьНазванияАлгоритмовПодписи(СертификатыBase64
_);
        ТекущееСостояние.Вставить(
        "АлгоритмПодписиСертификатаФСС",
        АлгоритмыПодписи_.АлгоритмПодписиСертификатаФСС);
```

```
    ТекущееСостояние.Вставить(
    "АлгоритмПодписиСертификатаМО",
    АлгоритмыПодписи_.АлгоритмПодписиСертификатаМО);
    ...
КонецФункции
```

Далее в процедуры Зашифровать(...) и РасшифроватьФайл(...) модуля КриптографияЭДКОСлужебныйКлиент был добавлен параметр Алгоритм, который определяет название алгоритмов сертификатов ФСС и МО, используемых для шифрования и расшифровки, соответственно.

Затем добавлена передача параметра Алгоритм для инициализации компоненты шифрования в метод ПредварительноСоздатьМенеджерКриптографии(...), который вызывается из процедуры Зашифровать(...) как показано в листинге 16.

Листинг 16. Инициализация компоненты шифрования

```
Процедура Зашифровать(..., СертификатПолучателя, ..., Алгоритм) Экспорт
    ...
    ПараметрыОпределенияАлгоритма =
    Новый Структура("Алгоритм, Сертификат", Алгоритм,
СертификатПолучателя);
    ПредварительноСоздатьМенеджерКриптографии(...,
ПараметрыОпределенияАлгоритма);
КонецПроцедуры
```

Метод ПредварительноСоздатьМенеджерКриптографии(...) так же вызывается при расшифровке сообщения и поэтому параметр Алгоритм аналогичным способом был добавлен в процедуру РасшифроватьФайл(...).

Из метода ПредварительноСоздатьМенеджерКриптографии(...) происходит последовательный вызов процедур для подготовки к работе компоненты, названия который начинаются с

СоздатьМенеджерКриптографии. Процедура для инициализации компоненты представлена в листинге 17.

Листинг 17. Инициализация компоненты шифрования

```
Процедура СоздатьМенеджерКриптографииИнициализация(
    ОповещениеОЗавершении, ВходящийКонтекст)
    Имя = ВходящийКонтекст.ДоступныеКриптопровайдеры[
    ВходящийКонтекст.ТекущийКриптопровайдер].Имя;
    Путь = ВходящийКонтекст.ДоступныеКриптопровайдеры[
    ВходящийКонтекст.ТекущийКриптопровайдер].Путь;
    Тип = ВходящийКонтекст.ДоступныеКриптопровайдеры[
    ВходящийКонтекст.ТекущийКриптопровайдер].Тип;
    ВходящийКонтекст.МенеджерКриптографии.НачатьВызовСоздатьМен
еджераКриптографии(
    ОповещениеОЗавершении, Имя, Путь, Тип);
КонецПроцедуры
```


Использование алгоритма сертификата необходимо для получения параметров криптопровайдера, при помощи которого инициализируется компонента шифрования. Ранее, при использовании только алгоритма ГОСТ Р 34.10-2001, был задан один вариант свойств как, например, показано в листинге 18 для КриптоПро.

Листинг 18. Параметры КриптоПро по ГОСТ Р 34.10-2001

```
Функция КриптопровайдерCryptoPro() Экспорт
    Свойства = Новый Структура();
    Свойства.Вставить(
    "Имя", "Crypto-Pro GOST R 34.10-2001 Cryptographic Service Provider");
    Свойства.Вставить("Путь", "");
    Свойства.Вставить("Тип", 75);
    Свойства.Вставить("Представление", "CryptoPro CSP");
    Возврат Новый ФиксированнаяСтруктура(Свойства);
```

КонецФункции

Для использования параметров криптопровайдеров по ГОСТ Р 34.10-2012 функции были изменены как показано в листинге 19.

Листинг 19. Параметры криптопровайдеров

```
Функция Криптопровайдер***(Алгоритм) Экспорт
    ...
    Если Алгоритм = "GOST R 34.10-2012-256" Тогда
    ...
    ИначеЕсли Алгоритм = "GOST R 34.10-2012-512" Тогда
    ...
    ИначеЕсли Алгоритм = "GOST R 34.10-2001" Тогда
    ...
    КонецЕсли;
    ...
КонецФункции
```


*** название криптопровайдера ViPNet или CryptoPro.

Далее необходимо было добавить схему формирования зашифрованного симметричного ключа для вставки в сообщение XML, которое будет отправлено в ФСС. Операция формирования ключа осуществляется в общем модуле ОбменДаннымиФССCервер в функции ПолучитьДанныеДляЗашифрованногоXML(...), фрагмент которой представлен в листинге 20.

Листинг 20. Формирование симметричного ключа для вставки в XML

```
Функция ПолучитьДанныеДляЗашифрованногоXML(
    Знач СимметричныйКлючВBase64, Знач ОткрытыйКлючВBase64,
    Знач ИнициализационныйВекторВBase64, Знач ЗашифрованныеДанныеВBase64)
    ...
```

```
// формирование симметричного ключа для вставки в XML
ДвоичныеДанныеСимметричногоКлюча =
Base64Значение(СимметричныйКлючВBase64);
ДвоичныеДанныеСимметричногоКлюча.Записать(ВременныйФайлСим
метричногоКлюча);
МассивСимметричногоКлюча =
ПрочитатьФайлВМассив(ВременныйФайлСимметричногоКлюча);
...
МассивОткрытогоКлюча =
ПрочитатьФайлВМассив(ВременныйФайлОткрытогоКлюча);
// заголовок симметричного ключа для XML
МассивСимметричногоКлючаДляXML =
ПолучитьМассивБайтовИзСтрокиВШестнадцатеричномПредставлении
(
"30 81 A4 30 28 04 20");
...
// добавление открытого ключа отправителя
КолВо_ = МассивОткрытогоКлюча.Количество();
Для ИндексМассива = КолВо_ 64 По КолВо_ 1 Цикл
МассивСимметричногоКлючаДляXML.Добавить(МассивОткрытогоКл
юча[ИндексМассива]);
КонецЦикла;
// добавление инициализационного вектора симметричного ключа
Для ИндексМассива = 16 По 23 Цикл
МассивСимметричногоКлючаДляXML.Добавить(
МассивСимметричногоКлюча[ИндексМассива]);
КонецЦикла;
...
ВременныйФайлСимметричногоКлючаДляXML =
ПолучитьИмяВременногоФайла();
```

```
ЗаписатьФайлИзМассива(
ВременныйФайлСимметричногоКлючаДляXML,
МассивСимметричногоКлючаДляXML);
ДвоичныеДанныеСимметричногоКлючаДляXML =
Новый
ДвоичныеДанные(ВременныйФайлСимметричногоКлючаДляXML);
СимметричныйКлючДляXMLBBase64 =
Base64Строка(ДвоичныеДанныеСимметричногоКлючаДляXML);
...
Возврат СимметричныйКлючДляXMLBBase64;
КонецФункции
```

Для добавления схемы формирования ключа с использованием ГОСТ Р 34.10-2012 в функцию ПолучитьДанныеДляЗашифрованногоXML(...) сначала был добавлен параметр Алгоритм, а затем определены условия выбора необходимых параметров как показано в листинге 21.

Листинг 21. Выбор параметров для формирования симметричного ключа

```
Функция ПолучитьДанныеДляЗашифрованногоXML(
Знач СимметричныйКлючBBase64, Знач ОткрытыйКлючBBase64,
Знач ИнициализационныйВекторBBase64, Знач
ЗашифрованныеДанныеBBase64,
Алгоритм)
...
// заголовок симметричного ключа для XML
Если Алгоритм = "GOST R 34.10-2012-256" Тогда
МассивСимметричногоКлючаДляXML =
ПолучитьМассивБайтовИзСтрокиВШестнадцатеричномПредставлении
(
"30 81 A4 30 28 04 20");
ИначеЕсли Алгоритм = "GOST R 34.10-2012-512" Тогда
```

```
...
ИначеЕсли Алгоритм = "GOST R 34.10-2001" Тогда
...
КонецЕсли;
...
// добавление открытого ключа отправителя
КолВо_ = МассивОткрытогоКлюча.Количество() -
?(Алгоритм = "GOST R 34.10-2012-512", 128, 64);
Для ИндексМассива = КолВо_ По КолВо_ 1 Цикл
МассивСимметричногоКлючаДляXML.Добавить(МассивОткрытогоКлюча[ИндексМассива]);
КонецЦикла;
...
КонецФункции
```


Также изменения необходимо было внести в функцию разбора сообщения от ФСС, фрагмент которой представлен в листинге 22.

Листинг 22. Разбор зашифрованного симметричного ключа из XML

```
Функция РазобратьДанныеИзЗашифрованногоXML(
Знач СимметричныйКлючИзXMLBBase64, Знач ЗашифрованныеДанныеИзXMLBBase64,
СимметричныйКлючBBase64, ОткрытыйКлючBBase64,
ИнициализационныйВекторBBase64, ЗашифрованныеДанныеBBase64)
// извлечение симметричного ключа из данных XML
ВременныйФайлСимметричногоКлючаИзXML = ПолучитьИмяВременногоФайла();
ДвоичныеДанныеСимметричногоКлючаИзXML =
Base64Значение(СимметричныйКлючИзXMLBBase64);
ДвоичныеДанныеСимметричногоКлючаИзXML.Записать(
ВременныйФайлСимметричногоКлючаИзXML);
```

```
МассивСимметричногоКлючаИзXML =
ПрочитатьФайлВМассив(ВременныйФайлСимметричногоКлючаИзXML);
// заголовок симметричного ключа
МассивСимметричногоКлюча =
ПолучитьМассивБайтовИзСтрокиВШестнадцатеричномПредставлении(
"01 20 00 00 1E 66 00 00 FD 51 4A 37 1E 66 00 00");
Для ИндексМассива = 159 По 166 Цикл
МассивСимметричногоКлюча.Добавить(
МассивСимметричногоКлючаИзXML[ИндексМассива]);
КонецЦикла;
ВременныйФайлСимметричногоКлюча = ПолучитьИмяВременногоФайла();
ЗаписатьФайлИзМассива(
ВременныйФайлСимметричногоКлюча,
МассивСимметричногоКлюча);
ДвоичныеДанныеСимметричногоКлюча =
Новый ДвоичныеДанные(ВременныйФайлСимметричногоКлюча);
СимметричныйКлючВBase64 = Base64Строка(ДвоичныеДанныеСимметричногоКлюча);
...
// извлечение открытого ключа из данных XML
МассивОткрытогоКлюча =
ПолучитьМассивБайтовИзСтрокиВШестнадцатеричномПредставлении(
"06 20 00 00 23 2E 00 00 4D ...");
Для ИндексМассива = 93 По 156 Цикл
МассивОткрытогоКлюча.Добавить(МассивСимметричногоКлючаИзXML[ИндексМассива]);
```

```
    КонецЦикла;
    ...
    ОткрытыйКлючВBase64 =
Base64Строка(ДвоичныеДанныеПубличногоКлюча);
    // извлечение инициализационного вектора из данных XML
    ДвоичныеДанныеИнициализационногоВектора =
    Новый
ДвоичныеДанные(ВременныйФайлИнициализационногоВектора);
    ИнициализационныйВекторВBase64 =
    Base64Строка(ДвоичныеДанныеИнициализационногоВектора);
    ...
КонецФункции
```

Для добавления схемы разбора зашифрованного симметричного ключа по ГОСТ Р 34.10-2012 функция РазобратьДанныеИзЗашифрованногоXML(...) была изменена как показано в листинге 23.

Листинг 23. Выбор параметров для разбора симметричного ключа

```
Функция РазобратьДанныеИзЗашифрованногоXML(
    Знач СимметричныйКлючИзXMLBBase64, Знач
ЗашифрованныеДанныеИзXMLBBase64,
    СимметричныйКлючВBase64, ОткрытыйКлючВBase64,
    ИнициализационныйВекторВBase64, ЗашифрованныеДанныеВBase64,
    Алгоритм)
    ...
    // извлечение симметричного ключа из данных XML
    МассивСимметричногоКлюча =
    ПолучитьМассивБайтовИзСтрокиВШестнадцатеричномПредставлении
(
    "01 20 00 00 1E 66 00 00 FD 51 4A 37 1E 66 00 00");
    ИндексНачала = ?(Алгоритм = "GOST R 34.10-2012-256", 164,
```

```
    ?(Алгоритм = "GOST R 34.10-2012-512", 232, 159));
    ИндексКонца = ?(Алгоритм = "GOST R 34.10-2012-256", 171,
    ?(Алгоритм = "GOST R 34.10-2012-512", 239, 166));
    Для ИндексМассива = ИндексНачала По ИндексКонца Цикл
    МассивСимметричногоКлюча.Добавить(
    МассивСимметричногоКлючаИзXML[ИндексМассива]);
    КонецЦикла;
    ...
    // извлечение открытого ключа из данных XML
    ИндексНачала = ?(Алгоритм = "GOST R 34.10-2012-256", 98,
    ?(Алгоритм = "GOST R 34.10-2012-512", 102, 93));
    ИндексКонца = ?(Алгоритм = "GOST R 34.10-2012-256", 161,
    ?(Алгоритм = "GOST R 34.10-2012-512", 229, 156));
    Для ИндексМассива = ИндексНачала По ИндексКонца Цикл
    МассивОткрытогоКлюча.Добавить(
    МассивСимметричногоКлючаИзXML[ИндексМассива]);
    КонецЦикла;
    ...
КонецФункции
```


3.4. Тестирование

При проверке подписания, хеширования и шифрования использовалась конфигурация «1С:Медицина. Поликлиника», в которой была реализована поддержка алгоритмов ГОСТ Р 34.10-2012 и ГОСТ Р 34.11-2012. Для того чтобы протестировать реализованную функциональность, были выполнены следующие действия:

1. Установлен криптопровайдер ViPNet CSP версии 4.2 на компьютер.

2. Созданы запросы на сертификаты электронной подписи по ГОСТ Р 34.10-2012-256 и и ГОСТ Р 34.10-2012-512 при помощи службы

сертификации, которая входит в состав установленного криптопровайдера ViPNet CSP.

3. Установлены контейнеры закрытого ключа, которые формируются после создания запроса на сертификат.

4. Получены сертификаты в тестовом удостоверяющем центре Infotecs на сайте http://testcert.infotecs.ru/. Сведения и состав полученного сертификата по ГОСТ Р 34.10-2012-256 показаны на рис. 8.

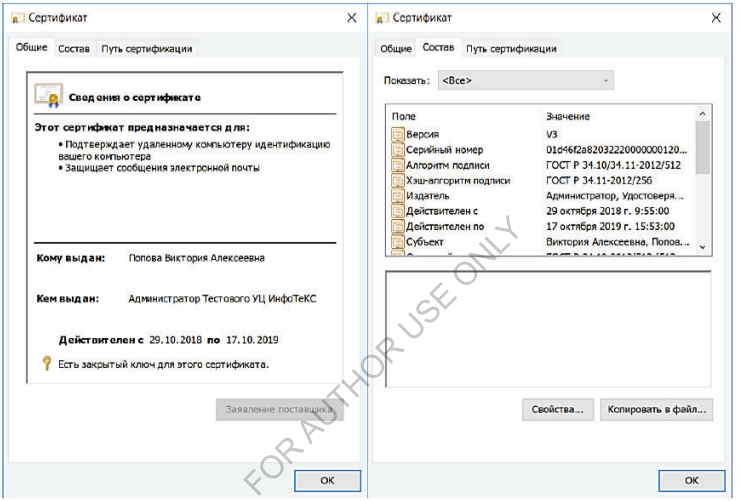

Рисунок 8. Сведения и состав сертификата по ГОСТ Р 34.10-2012

5. Произведена установка сертификатов в личное хранилище пользователя.

6. Добавлены контейнеры закрытого ключа в программу криптопровайдера.

7. Помещены установленные сертификаты в соответствующие контейнеры через программу ViPNet.

8. Добавлены сертификаты в информационную базу «1С:Медицина. Поликлиника» в режиме «1С:Предприятие».

9. Получен с сайта ФСС (http://cabinets-test.fss.ru/eln.html) и установлен сертификат уполномоченного лица ФСС.

После выполнения вышеописанных действий было протестировано подписание ЭЛН, медицинских документов и сообщений, отправляемых в РЭМД. Также было проверено шифрование при обмене данными ЭЛН с ФСС.

В результате все проверки завершились успешно и реализованная поддержка криптографических алгоритмов ГОСТ Р 34.10-2012 и ГОСТ Р 34.112012 вошла в следующие релизы программных продуктов:

- «1С:Медицина. Больничные» версии 2.0.3.1 от 17.12.2018;
- «1С:Медицина. Поликлиника» версии 2.1.5.1 от 25.12.2018;
- «1С:Медицина. Больница» версии 1.4.5.1 от 25.12.2018.

Глава 4. Реализация шифрования и расшифровки сообщений на стороне сервера

4.1. Определение этапов выполнения задачи

Также в рамках выполнения задач работы необходимо было реализовать на стороне сервера операции шифрования и расшифровки сообщений, которые применяются для обмена данными ЭЛН с ФСС. Ранее эти операции могли выполняться только на стороне клиента.

При наличии у организации компьютера с сервером «1С:Предприятие» наиболее удобным решением является установить на него криптопровайдер и необходимые сертификаты, а затем настроить взаимодействие этого компьютера с клиентскими. В таком случае все компьютеры, подключённые к серверу, смогут использовать в режиме «1С:Предприятие» функциональность криптопровайдера и сертификаты электронной подписи. Поскольку раньше не было возможности шифровать и расшифровывать сообщения на стороне сервера, приходилось множество раз производить установку необходимого сертификата и переносить контейнер закрытого ключа на каждый клиентский компьютер врача в медицинском учреждении, использующем продукты линейки «1С:Медицина».

Но при наличии серверного компьютера не всегда имеется возможность устанавливать на него сертификаты и криптопровайдер. Поэтому подсистемой БСП «Электронная подпись» предоставляются настройки, которые позволяют определять пользователю в режиме «1С:Предприятие», где могут производиться, на стороне клиента или сервера, операции подписания, шифрования и расшифровки, а также проверки подписей и сертификатов. Эти настройки находятся в форме «Настройки электронной подписи и шифрования», где необходимо установить в нужное состояние пометки «Подписывать и шифровать на сервере» и «Проверять подписи и сертификаты на сервере» как показано на рис. 9.

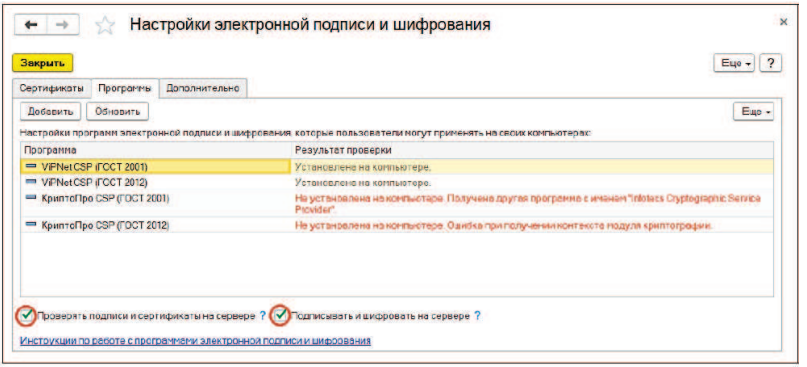

Рисунок 9. Форма «Настройки электронной подписи и шифрования»

По умолчанию опции «Подписывать и шифровать на сервере» и «Проверять подписи и сертификаты на сервере» не включены и все функции криптографии выполняются только на клиенте. Если включена возможность выполнять криптографические операции на сервере, то до начала шифрования и расшифровки сообщений нужно проверять куда установлены сертификаты МО и ФСС. Когда сертификат установлен на клиентском компьютере, процедура шифрования (или расшифровки) должна выполниться на стороне клиента, иначе на стороне сервера.

Таким образом, для реализации шифрования и расшифровки сообщений на стороне сервера для работы в клиент-сервером режиме были определены следующие этапы:

1. добавить автоматическое определение мест выполнения шифрования и расшифровки — на стороне клиента или сервера;

2. написать функции для шифрования и расшифровки сообщений на стороне сервера с использованием ключей сертификатов по ГОСТ Р 34.10-2001 и ГОСТ Р 34.10-2012;

3. протестировать возможность шифрования и расшифровки сообщений на стороне сервера в клиент-серверном режиме работы.

4.2. Описание программной реализации

Ранее операции шифрования и расшифровки сообщений при обмене с ФСС выполнялись как показано на рис. 10.

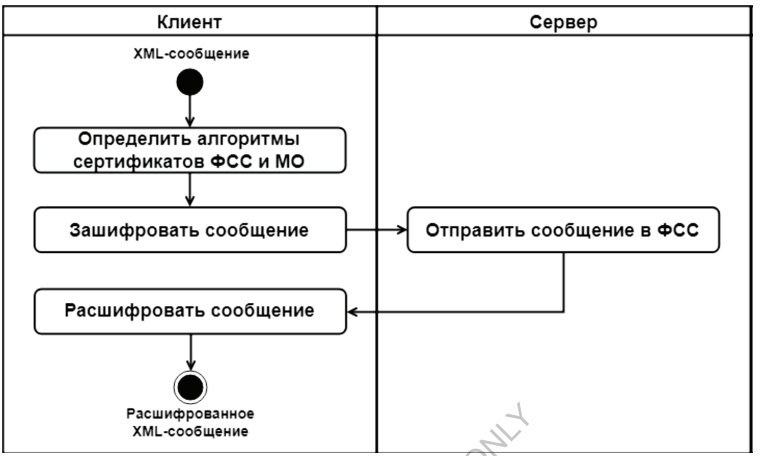

Рисунок 10. Шифрование и расшифровка сообщений на стороне клиента

Для добавления возможности шифровать и расшифровывать сообщения на стороне сервера нужно было изменить схему (см. рис. 10) так, как показано на рис. 11.

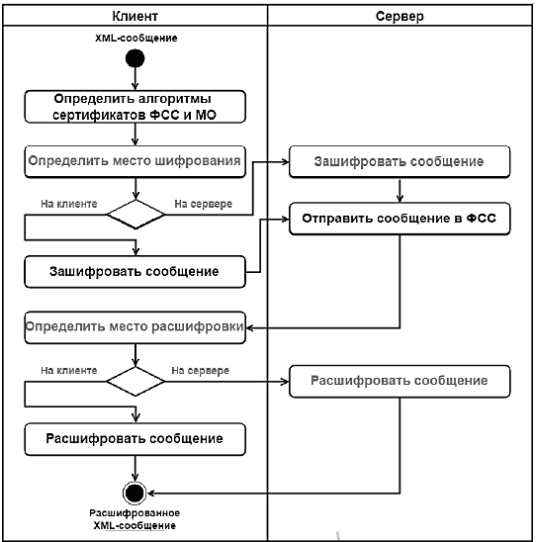

Рисунок 11. Добавление шифрования и расшифровки на стороне сервера

В изменённой схеме определение алгоритмов сертификатов ФСС и МО решено было оставить на стороне клиента. Связано это с тем, что извлекать алгоритм нужно из сертификата, представленного в формате base64, а в такой формат можно выгрузить сертификаты, которые установлены как на клиентском компьютере, так и на серверном.

Для того чтобы определить, где должны производиться операции шифрования и расшифровки сообщений, в первую очередь необходимо проверять поставлена ли в форме «Настройки электронной подписи и шифрования» пометка «Подписывать и шифровать на сервере». Для такой проверки используется метод общего модуля ЭлектроннаяПодписьКлиентСервер ОбщиеНастройки(), который входит в состав БСП. За возможность формирования подписи, а также шифрования и расшифровки сообщений на стороне сервера отвечает настройка СоздаватьЭлектронныеПодписиНаСервере.

Проверка возможности шифрования и расшифровки на сервере была добавлена в функцию

ШифрованиеОтправкаРасшифровка_ОбработкаЭтапа(...) как показано в листинге 24.

Листинг 24. Проверка возможности шифрования и расшифровки на сервере

```
ОбщиеНастройки_ = ЭлектроннаяПодписьКлиентСервер.ОбщиеНастройки();
    Если Не ОбщиеНастройки_.СоздаватьЭлектронныеПодписиНаСервере
Тогда
    Результат_ = Новый Массив;
    Результат_.Добавить(Новый Структура("ШифрованиеНаКлиенте",
Истина));
    Результат_.Добавить(Новый Структура("РасшифровкаНаКлиенте",
Истина));
    Возврат
    ШифрованиеОтправкаРасшифровка_ОбработкаЭтапа(Результат_,
ТекущееСостояние);
    КонецЕсли;
```


Далее была добавлена проверка сертификатов для шифрования и расшифровки, которая предполагает следующее: если сертификат установлен на клиентском компьютере, то операцию шифрования (или расшифровки) выполнять 58 на клиенте, иначе на сервере. Для выполнения таких проверок использовались методы БСП.

Сначала необходимо проверить, что сертификат для шифрования или расшифровки установлен на клиентском компьютере. Эта проверка осуществляется при помощи метода ПолучитьСертификатПоОтпечатку(...). Затем определяется установлен ли на компьютере криптопровайдер. Для выполнения шифрования сообщения подходит любой криптопровайдер, а для расшифровки только тот, в котором находится контейнер закрытого ключа, связанный с сертификатом МО. Для проверки наличия криптопровайдера на компьютере используется метод СоздатьМенеджерКриптографии(...).

Далее нужно было реализовать на стороне сервера процессы шифрования и расшифровки сообщений. В клиентском варианте работы эти операции выполнялись при помощи механизма асинхронных вызовов. На стороне сервера такой механизм использовать невозможно, поскольку поддерживаются только синхронные вызовы методов.

Механизм синхронных вызовов работает следующим образом:

1. Выполняется вызов какой-либо процедуры или функции.

2. Приостанавливается выполнение метода, из которого происходит вызов.

3. Работа метода продолжается после завершения работы вызванного и всех вложенных методов.

Стоит отметить, что при синхронных вызовах блокируется пользовательский интерфейс системы.

Асинхронные вызовы не блокируют пользовательский интерфейс. При их использовании результат работы метода будет сразу возвращён в вызывающую функцию или процедуру, которая является основным потоком выполнения. При этом все вложенные методы вызванной процедуры продолжат выполняться параллельно с основным потоком.

Шифрование и расшифровку сообщений на стороне сервера необходимо было реализовать по аналогии с клиентским вариантом работы, но без использования механизма асинхронных вызовов. Например, компонента шифрования для работы на клиенте предоставляет метод НачатьВызовЗашифровать(...), в который первым параметром передаётся объект ОписаниеОповещения, содержащий название процедуры, которая будет вызвана после выполнения асинхронного метода. Для шифрования сообщений на стороне сервера нужно использовать метод Зашифровать(...), в который передаётся на один параметр меньше чем в НачатьВызовЗашифровать(...), так как не нужно использовать ОписаниеОповещения.

Таким образом, для шифрования и расшифровки сообщений на стороне сервера необходимо было реализовать следующие операции:

1. Запись сообщения для шифрования или расшифровки, предварительно закодированного по формату base64, в файл, сохраняемый на компьютер.

2. Инициализация компоненты по алгоритму сертификата ФСС или МО.

3. Шифрование и расшифровка сообщения при помощи методов компоненты «Компонента обмена».

4. Чтение зашифрованного или расшифрованного сообщения из файла, сохранённого на компьютер.

Запись сообщения в файл необходима ввиду того, что компонента может шифровать или расшифровывать только те данные, которые сохранены в файлы на компьютере. Результат, то есть зашифрованное или расшифрованное сообщение, также помещается в файл, из которого затем происходит чтение данных средствами платформы «1С:Предприятие».

Для начала был создан серверный модуль КриптографияЭДКОСлужебный. Затем в этом модуле были реализованы следующие функции:

- ЗашифроватьСообщение(...) шифрование сообщения;
- РасшифроватьСообщение(...) расшифровка сообщения;
- ЗаписатьСтрокуBase64ВФайл(...) запись сообщения в формате base64 в файл;
- ПрочитатьФайлВСтрокуBase64(...) чтение файла в строку формата base64;
- ОбъектВнешнейКомпонентыОбмена(...) создание и инициализация объекта компоненты;
- ЗаполнитьПараметрыКриптографии(...) заполнение параметров криптопровайдеров для инициализации компоненты;

- СоздатьФайлНаСервере(...) создание файла на серверном компьютере;
- УдалитьФайлыНаСервере(...) удаление всех файлов, созданных в процессе шифрования и расшифровки сообщений.

Программный код всех функций представлен в прил. 2.

4.3. Тестирование

Для того чтобы протестировать шифрование и расшифровку сообщений на стороне сервера, использовался удалённый доступ к серверному компьютеру с установленной операционной системой Windows 10.

Для проведения тестирования было необходимо на серверном компьютере:

- Установить сервер «1С:Предприятие».
- Настроить службу «Агент сервера 1С:Предприятия».
- Установить сервер баз данных MS SQL SERVER.
- Выполнить установку криптопровайдера.
- Добавить контейнер закрытого ключа.
- Произвести установку сертификатов ФСС и МО.

Для установки сервера «1С:Предприятие» нужно запустить файл setup.exe, который входит в дистрибутив платформы, и затем выбрать в списке опции «Сервер 1С:Предприятия» и «Администрирование сервера 1С:Предприятия» как показано на рис. 12.

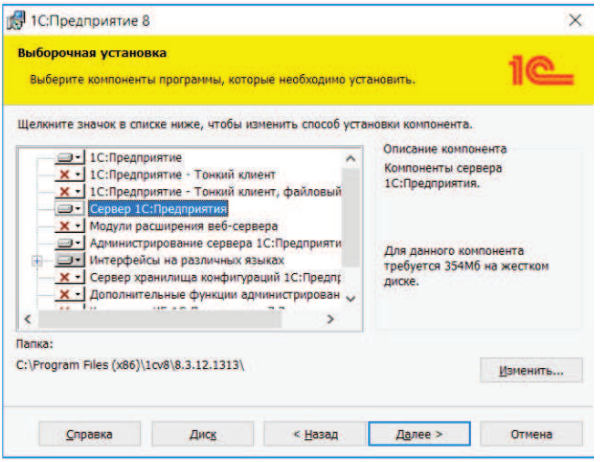

Рисунок 12. Выбор опций для установки сервера

Также при установке сервера нужно создать или выбрать уже существующего пользователя, который будет обладать правами на запуск сервера. Этот этап процесса установки представлен на рис. 13.

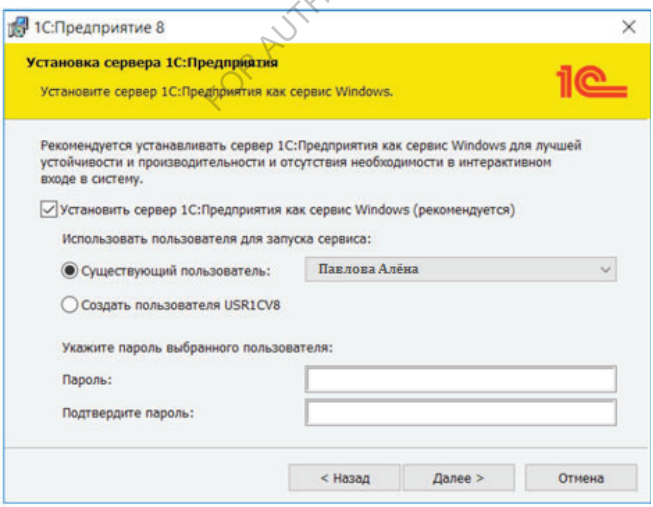

Рисунок 13. Указание пользователя с правами на запуск сервера

Затем выполнялась установка сервера баз данных MS SQL Server 2017. В процессе установки использовались настройки по умолчанию и был указан пароль для администратора SQL SERVER ssa» в форме, представленной на рис. 15.

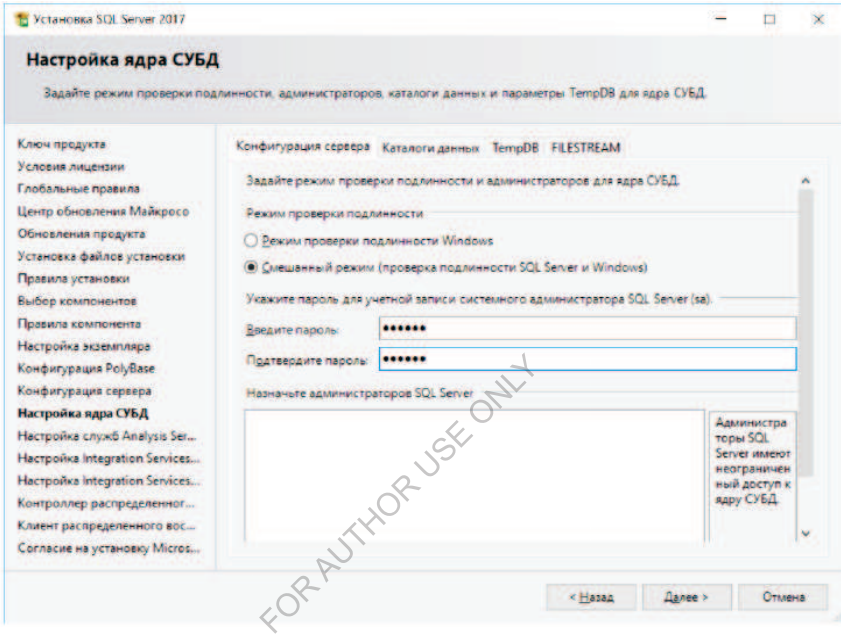

Рисунок 14. Установка MS SQL Server

Далее был установлен криптопровайдер КриптоПро версии 4.0, сгенерирован контейнер закрытого ключа и получен тестовый сертификат для МО при помощи утилиты КриптоПро CADESCOM.

После установки всех необходимых компонентов на серверный компьютер нужно было настроить использование платформы «1С:Предприятие» в клиентсерверном режиме. В первую очередь нужно с клиентского компьютера запустить программу «1С:Предприятие» и создать информационную базу, указав при этом тип расположения «На сервере 1С:Предприятия». Далее вводится название кластера серверов, имя

информационной базы, тип СУБД и данные сервера базы данных. Все введённые параметры представлены на рис. 16.

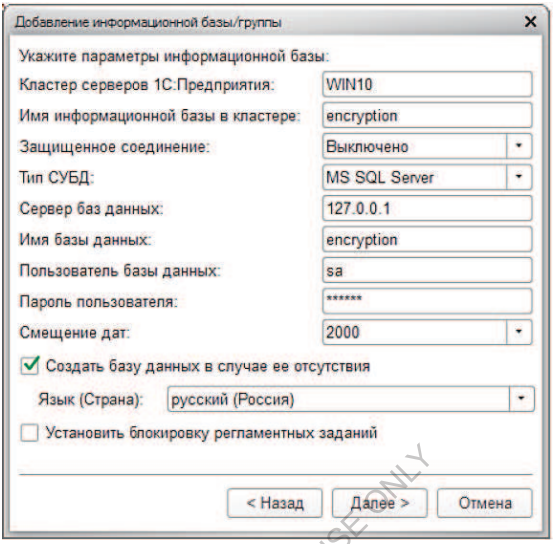

Рисунок 15. Добавление информационной базы

Далее в созданную базу была загружена конфигурация «1С:Медицина. Поликлиника». В первую очередь был протестирован вариант, когда сертификаты ФСС и МО установлены на серверном компьютере. Затем проверены варианты, когда:

1. сертификат ФСС установлен на клиентском компьютере, а МО на серверном;

2. сертификат МО установлен на клиентском компьютере, а ФСС на серверном.

Также были протестированы операции шифрования и расшифровки сообщений в случае установки обоих сертификатов на клиентский компьютер.

В конечном итоге все проверки завершились успешно и, таким образом, было показано, что операции шифрования и расшифровки сообщений на стороне сервера реализованы успешно.

Реализованную возможность шифрования и расшифровки сообщений на стороне сервера планируется включить в релизы программ «1С:Медицина. Поликлиника», «1С:Медицина. Больница» и «1С:Медицина. Больничные».

Особенности шифрования и расшифровки сообщений в операционных системах семейства Linux

Следует отметить, что операции шифрования и расшифровки сообщений, выполняемые на стороне клиента в конфигурациях линейки «1С:Медицина», не работали на компьютерах с установленной операционной системой семейства Linux. Проблема заключалась в том, что программа в режиме «1С:Предприятие» без каких-либо уведомлений завершала свою работу при попытке инициализации компоненты «Компонента обмена».

Отличие работы компоненты в разных операционных системах состоит в том, что для Windows компонента предоставляется в виде dll-библиотеки, а в Linux применяется библиотека с расширением so.

Поскольку ранее операции шифрования и расшифровки выполнялись только на стороне клиента, предполагалось, что компонента не поддерживает механизм асинхронных вызовов в операционных системах Linux. Поэтому было решено проверить может ли компонента шифровать и расшифровывать сообщения на стороне сервера, где применяются только синхронные вызовы методов.

Для тестирования шифрования и расшифровки сообщений использовался серверный компьютер с установленной операционной системой Ubuntu 14.04. После выполнения необходимых настроек средств криптографии и конфигурации «1С:Медицина. Поликлиника» было произведено тестирование шифрования сообщения. В ходе такой проверки сообщение не было зашифровано, так как при попытке инициализации компоненты программа завершала свою работу без вывода ошибок.

Таким образом, было определено, что компонента не поддерживает шифрование и расшифровку сообщений в операционной системе Ubuntu

14.04, как при использовании асинхронного механизма вызова методов, так и синхронного.

В дальнейшем тестирование шифрования и расшифровки в операционных системах семейства Linux будет проведено после выпуска новой версии компоненты фирмой «1С».

ЗАКЛЮЧЕНИЕ

Актуальность и востребованность криптографических алгоритмов возрастает, этому способствуют ряд причин, одной из которых является распространение опаснейшего вируса. Так как передача файлов происходит без личного взаимодействия людей и минимальными угрозами подделки электронных документов.

При выполнении работы были решены следующие задачи:

1. проанализирована предметная область и рассмотрены стандарты криптографических алгоритмов;

2. в программных продуктах линейки «1С:Медицина»:

2.1. определены используемые методы и средства криптографии;

2.2. добавлена возможность подписания по ГОСТ Р 34.10-2012 и формирования хеша по ГОСТ Р 34.11-2012;

2.3. настроены операции шифрования и расшифровки сообщений с использованием ключей по алгоритму ГОСТ Р 34.10-2012;

2.4. обеспечена возможность шифрования и расшифровки сообщений на стороне сервера;

2.5. протестировано использование алгоритмов ГОСТ Р 34.10-2012 и ГОСТ Р 34.11-2012.

В результате выполнения задач была достигнута поставленная цель — реализована поддержка криптографических алгоритмов ГОСТ Р 34.10-2012 и ГОСТ Р 34.11-2012 в программных продуктах линейки «1С:Медицина».

Также при выполнении работы в программных продуктах линейки «1С:Медицина» была реализована возможность выполнять операции шифрования и расшифровки сообщений на стороне сервера.

Реализованная поддержка криптографических алгоритмов ГОСТ Р 34.102012 и ГОСТ Р 34.11-2012 вошла в релиз в программных продуктов линейки «1С:Медицина» в декабре 2018 года.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. 1С:Предприятие 8.3.13. Документация: Глава 35. Внешние компоненты [Электронный ресурс].[Б. м.: б. и.]. URL: https://its.1c.ru/db/%20v8313doc#bookmark:dev:TI000001194 (дата обращения: 29.03.2019).

2. Алгоритмы шифрования данных [Текст] / М. В. Коновалов, У. В. Михайлова, А. А. Хусаинов, Р. Ж. Санарбаев // Актуальные проблемы современной науки, техники и образования. — 2013. — Т. 2, №71. — С. 159-161.

3. Астахова, А. С. Электронная цифровая подпись как фактор сохранения целостности и аутентичности документа [Текст] / А. С. Астахова, Е. П. Чадаева // Известия Томского политехнического университета. — 2012. — №6.— С. 153-157.

4. Аутентификация. Теория и практика обеспечения безопасного доступа к информационным ресурсам [Текст] / А. А. Афанасьев, Л. Т. Веденьев, А. А. Воронцов [и др.]. — М.: Горячая линия — Телеком, 2012. — 550 с.

5. Выписка из документа ФСБ России №149/7/1/3-58 от 31.01.2014 «О порядке перехода к использованию новых стандартов ЭЦП и функции хэшированияї [Электронный ресурс].— [Б. м.: б. и.].— URL: https://tc26.ru/%20upload/medialibrary/5f7/5f724550477ed7eeeb2ab025ed7ffc79.pdf (дата обращения: 15.10.2020).

6. ГОСТ 28147-89. Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования. — Введ. 01.07.90. [Текст]. — М.: Изд-во стандартов, 1989. — 28 с.

7. ГОСТ Р 34.10-2001. Информационная технология (ИТ). Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи. — Взамен ГОСТ Р 34.10-94; Введ. 01.07.2002. [Текст]. — М.: ИПК Изд-во стандартов, 2001. — 16 с.

8. ГОСТ Р 34.10-2012. Информационная технология (ИТ). Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи. — Взамен ГОСТ Р 34.10-2001; Введ. 01.01.2013. [Текст]. — М.: Стандартинформ, 2012. — 33 с.

9. ГОСТ Р 34.10-94. Информационная технология (ИТ). Криптографическая защита информации. Процедуры выработки и проверки электронной цифровой подписи на базе асимметричного криптографического алгоритма. — Введ. 01.01.95. [Текст]. — М.: Изд-во стандартов, 1994. — 18 с.

10. ГОСТ Р 34.11-2012. Информационная технология (ИТ). Криптографическая защита информации. Функция хэширования. — Взамен ГОСТ Р 34.1194; Введ. 01.01.2013. [Текст]. — М.: Стандартинформ, 2013. — 24 с.

11. ГОСТ Р 34.11-94. Информационная технология (ИТ). Криптографическая защита информации. Функция хэширования. — Введ. 01.01.95. [Текст]. — М.: Изд-во стандартов, 1994. — 16 с.

12. ГОСТ Р 34.12-2015. Информационная технология (ИТ). Криптографическая защита информации. Блочные шифры. — Введ. 01.01.2016. [Текст]. — М.: Стандартинформ, 2016. — 15 с.

13. Бессонов, В. А. Обзор современных форматов электронных документов [Текст] / В. А. Бессонов // Математика программных систем: межвуз. сб. науч. ст.— Пермь: Пермский государственный национальный исследовательский университет, 2012. — С. 120-131.

14. Библиотека стандартных подсистем 2.4.6. Документация: 3.61. Электронная подпись [Электронный ресурс]. — [Б. М.: б. и.]. — URL: https://its.1c.ru/db/bsp246doc#content:76:hdoc (дата обращения: 28.03.2020).

15. Библиотека стандартных подсистем 2.4.6. Документация: Глава 1. Состав библиотеки [Электронный ресурс]. — [Б. М.: б. и.]. — URL: https://its.1c.ru/db/bsp246doc#content:2:hdoc (дата обращения: 28.03.2020).

16. Библиотека стандартных подсистем 3.0.1. Документация [Электронный ресурс]. — [Б. М.: б. и.]. — URL: https://its.1c.ru/db/bsp301doc (дата обращения: 07.05.2020).

17. Бутакова, Н. Г. Криптографические методы и средства защиты информации: учебное пособие [Текст] / Н. Г. Бутакова, Н. В. Федоров. — Спб.: ИЦ «Интермедия», 2016. — 384 с.

18. Жаботинский, Д. В. Юридическая сила электронного документа, заверенного электронной цифровой подписью в коммерческой деятельности [Текст] / Д. В. Жаботинский // Известия высших учебных заведений. Уральский регион. — 2010. — №3. — С. 21—23.

19. Жильников, А. Ю. Электронный документооборот [Текст] / А. Ю Жильников, А. С. Михайлова // Территория науки. — 2017. — №2. — С. 116-120.

20. Жуковина, О. А. Система электронного документооборота, её назначение и проблемы внедрения [Текст] / О. А. Жуковина, Н. Г. Зубова // Вестник Белгородского университета кооперации, экономики и права. — 2012. — №2. — С. 246-251.

21. Ильинская, Е. В. Безопасный обмен электронными документами [Текст] / Е. В. Ильинская, К. А. Павленко // Экономическая безопасность социальноэкономических систем: вызовы и возможности: сб. науч. тр. междунар. научпрактич. конф. — Белгород: Белгородский государственный национальный исследовательский университет, 2020. — С. 275-277.

22. КриптоПро CSP. Описание реализации [Текст]. — М.: ООО «КРИПТОПРО», 2016. — 32 с.

23. Курченков, К. Б. Критерии разработки систем электронного документооборота [Текст] / К. Б. Курченков // Вестник Воронежского института высоких технологий. — 2014. — №12. — С. 102-106.

24. Лаврова, О. С. Симметричные и асимметричные схемы электронной цифровой подписи [Текст] / О. С. Лаврова // Известия Юго-Западного государственного университета. — 2011. — №1. — С. 84-85.

25. Ланин, В. В. Классификация форматов электронных документов [Текст] / В. В. Ланин // Математика программных систем: межвуз. сб. науч. ст. — Пермь: Пермский государственный национальный исследовательский университет, 2013. — С. 4-9.

26. Ланских, В. Г. Повышение криптографической стойкости функций хеширования [Текст] / В. Г. Ланских, А. М. Ланских, Л. В. Пешнина // ИТ Арктика. — 2016. — №3. — С. 21-35.

27. Лёвин, В. Ю. Анализ повышения криптографической сложности систем при переходе на эллиптические кривые [Текст] / В. Ю. Лёвин, В. А. Носов // Интеллектуальные системы. — 2008. — №1. — С. 253-270.

28. Макаренко, С. И. Информационное оружие в технической сфере: терминология, классификация, примеры [Текст] / С. И. Макаренко // Системы управления, связи и безопасности. — 2016. — №3. — С. 292-376.

29. Об электронной подписи: федер. закон: [принят Гос. Думой 25 марта 2011 г.: одобрен Советом Федерации 30 марта 2011 г.]: по состоянию на 6 апр. 2011 г. [Текст] // Российская газета. — 2011. —8 апр. — С. 17.

30. Прудковский, Н. С. Асимметричное шифрование [Текст] / Н. С. Прудковский // Новая наука: опыт, традиции, инновации. — 2017. — №1-2. — С. 254- 257.

31. Работа с форматами doc, docx, rtf, pdf [Электронный ресурс]. [Б. М.: б. и.]. — URL: https://msoffice-prowork.com/%20rabota-s-formatami-doc-docx-rtf-pdf/ (дата обращения: 23.10.2020).

32. Секушина, С. А. Возможности использования алгоритмов шифрования в системах обработки электронных документов [Текст] / С. А. Секушина, А. А. Сапрыкин // Вестник Воронежского института высоких технологий. — 2014. — №13.— С. 120-122.

33. Смирнова, М. А. Асимметричное шифрование как способ проверки подлинности автора сообщения [Текст] / М. А. Смирнова, И. А. Самойлова, Л. В. Устинова // Актуальные проблемы современности. — 2017. — №2.— С. 184-188.

34. Трофимов, Е. И. Электронная подпись как средство защиты электронных документов от подделки [Текст] / Е. И. Трофимов // Педагогическое образование на Алтае. — 2014.— №2.— С. 446-447.

35. Уведомление об организации перехода на использование схемы электронной подписи по ГОСТ Р 34.10-2012 [Электронный ресурс]. — [Б. М.: б. и.]. — URL: https://goo.gl/4rfW2m (дата обращения: 15.10.2020).

36. Удостоверяющие центры в системе электронного документооборота [Текст] / И. Ф. Михалевич, А. О. Жуков, Д. В. Пантюхов, Ю. В. Лёвин // Известия Орловского государственного технического университета. Серия: Информационные системы и технологии. — 2004. — №5. — С. 180-187.

37. Управление данными XML: подходы к определению документов XML [Электронный ресурс]. — [Б. М.: б. и.]. — URL: http://citforum.ru/internet/%20xml/harold/ (дата обращения: 23.10.2020).

38. Черкасов, Д. Ю. Защита документов при помощи электронной цифровой подписи [Текст] / Д. Ю. Черкасов, В. В. Иванов // ECONOMICS. — 2016.— №6.— С. 51-53.

39. Чернова, А. Я. Анализ системы формирования и проверки электронной подписи [Текст] / А. Я. Чернова // Вестник Пензенского государственного университета. — 2017.— №3.— С. 108-111.

40. Canonical XML Version 1.1. W3C Recommendation 2 May 2008 [Электронный ресурс]. — [Б. М.: б. и.].— URL: https://www.w3.org/TR/xml-c14n/ (дата обращения: 24.10.2020).

41. Canonicalization of an XML document [Электронный ресурс]. — [Б. М.: б. и.]. — URL: https://www.di-mgt.com.au/xmldsig-c14n.html (дата обращения: 24.10.2020).

42. Exclusive XML Canonicalization Version 1.0. W3C Recommendation 18 July 2002 [Электронный ресурс]. — [Б. М.: б. и.]. — URL: https://www.w3.org/%20TR/xml-exc-c14n/ (дата обращения: 24.10.2020).

43. Extensible Markup Language (XML) 1.0 (Fifth Edition) W3C Recommendation 26 November 2008 [Электронный ресурс]. — [Б. М.: б. и.]. — URL: https://www.w3.org/TR/xml/ (дата обращения: 24.10.2020). 73

44. RFC 2315 PKCS #7: Cryptographic Message Syntax Version 1.5 [Электронный ресурс]. — [Б. М.: б. и.]. — URL: https://tools.ietf.org/html/ (дата обращения: 25.10.2020).

45. RFC 5126 CMS Advanced Electronic Signatures (CAdES) [Электронный ресурс]. — [Б. М.: б. и.]. — URL: https://tools.ietf.org/html/rfc5126 (дата обращения: 25.10.2020).

46. RFC 5652 Cryptographic Message Syntax (CMS) [Электронный ресурс]. — [Б. М.: б. и.]. — URL: https://tools.ietf.org/html/rfc5652 (дата обращения: 25.03.2020).

47. SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). W3C Recommendation 27 April 2007 [Электронный ресурс]. — [Б. М.: б. и.]. — URL: https://www.w3.org/TR/soap12-part1/ (дата обращения: 03.04.2020).

48. Tekli, J. An overview on XML similarity: Background, current trends and future directions [Текст] / J. Tekli, R. Chbeir, K. Yetongnon // Computer Science Review. — 2009. — Т. 3, №3. — С. 151-173.

49. ViPNet CSP 4.2. Руководство пользователя [Текст]. — М.: ОАО «ИнфоТеКС», 2015. — 222 с.

50. XML Signature Syntax and Processing Version 1.1. W3C Recommendation 11 April 2013 [Электронный ресурс]. — [Б. М.: б. и.]. — URL: https://www.w3.org/TR/xmldsig-core/ (дата обращения: 19.10.2020).

ПРИЛОЖЕНИЕ 1.

Параметры алгоритмов электронной подписи и хеширования

Название алгоритма	OID	URI
ГОСТ Р 34.10-2001	1.2.643.2.2.3	urn:ietf:params:xml:ns:cpxmlsec:algorithms:gostr34102001-gostr3411
ГОСТ Р 34.10-94	1.2.643.2.2.9	urn:ietf:params:xml:ns:cpxmlsec:algorithms:gostr3411
ГОСТ Р 34.10-2012-256	1.2.643.7.1.1.3.2	urn:ietf:params:xml:ns:cpxmlsec:algorithms:gostr34102012-gostr34112012-256
ГОСТ Р 34.10-2012-256	1.2.643.7.1.1.2.2	urn:ietf:params:xml:ns:cpxmlsec:algorithms:gostr34112012-256
ГОСТ Р 34.10-2012-512	1.2.643.7.1.1.3.3	urn:ietf:params:xml:ns:cpxmlsec:algorithms:gostr34102012-gostr34112012-512
ГОСТ Р 34.10-2012-512	1.2.643.7.1.1.2.3	urn:ietf:params:xml:ns:cpxmlsec:algorithms:gostr34112012-512

ПРИЛОЖЕНИЕ 2.

Реализованные методы для шифрования и расшифровки сообщений на стороне сервера

```
Функция ЗашифроватьСообщение(СертификатОтправителя,
    СертификатПолучателя, СообщениеBase64, АлгоритмПодписи) Экспорт

    ИмяФайлаДанных_ = ЗаписатьСтрокуBase64ВФайл(СообщениеBase64,
        ".xml");
    ИмяКаталогаРезультата_ =
        ОбщегоНазначения.СоздатьВременныйКаталог();
    УдаляемыеФайлы_ = Новый Массив;
    УдаляемыеФайлы_.Добавить(ИмяФайлаДанных_);
    УдаляемыеФайлы_.Добавить(ИмяКаталогаРезультата_);

    ПараметрыКриптографии_ = Новый Структура(
        "АлгоритмШифрования, АлгоритмКлюча, Режим", 0, 26142, 1);

    Если ЗначениеЗаполнено(СертификатОтправителя) Тогда
        СертификатОтправителяСерийныйНомер_ =
            СертификатОтправителя.СерийныйНомер;
        СертификатОтправителяПоставщик_ =
            СертификатОтправителя.Поставщик;
    Иначе
        СертификатОтправителяСерийныйНомер_ = "";
        СертификатОтправителяПоставщик_ = "";
    КонецЕсли;

    ОписаниеОшибки_ = "";
    МенеджерКриптографии_ = ОбъектВнешнейКомпонентыОбмена(
        АлгоритмПодписи, ОписаниеОшибки_);
```

```
Если Не ЗначениеЗаполнено(ОписаниеОшибки_) Тогда
    Попытка
        МенеджерКриптографии_.Зашифровать(
            ИмяФайлаДанных_, // исходный файл
            СертификатОтправителяСерийныйНомер_,
            СертификатОтправителяПоставщик_,
            СертификатПолучателя.СерийныйНомер,
            СертификатПолучателя.Поставщик,
            ИмяКаталогаРезультата_, // путь к каталогу с результатом
            ПараметрыКриптографии_.АлгоритмШифрования,
            ПараметрыКриптографии_.АлгоритмКлюча,
            ПараметрыКриптографии_.Режим);
        Исключение
            ОписаниеОшибки_ = НСтр("ru='Не удалось зашифровать
                файл.'");
        КонецПопытки;
КонецЕсли;

Если ЗначениеЗаполнено(ОписаниеОшибки_) Тогда
        РезультатОшибка_ = Новый Структура;
        РезультатОшибка_.Вставить("Выполнено", Ложь);
        РезультатОшибка_.Вставить("ОписаниеОшибки",
            ОписаниеОшибки_);
        Возврат РезультатОшибка_;
Иначе
        МассивФайлов_ = АлгоритмыДляКоллекций.СоздатьМассив(
            ИмяКаталогаРезультата_ + "_skey.bin",
            ИмяКаталогаРезультата_ + "_pubkey.bin",
            ИмяКаталогаРезультата_ + "_iv.bin",
            ИмяКаталогаРезультата_ + "_encrypt.bin");
```

```
            РезультатМассив_ = Новый Массив;
            Для Каждого Файл_ Из МассивФайлов_ Цикл
               Структура_ = Новый Структура;
               СтрокаBase64_ = ПрочитатьФайлВСтрокуBase64(Файл_);
               Структура_.Вставить("Выполнено", Истина);
               Структура_.Вставить("СтрокаBase64", СтрокаBase64_);
               РезультатМассив_.Добавить(Структура_);
            КонецЦикла;
            УдалитьФайлыНаСервере(УдаляемыеФайлы_);
            Возврат РезультатМассив_;
      КонецЕсли;
КонецФункции

Функция РасшифроватьСообщение(ДанныеДляРасшифровки,
      СертификатПолучателя, АлгоритмПодписи) Экспорт

      ИмяКаталога_ = КаталогВременныхФайлов();
      УдаляемыеФайлы_ = Новый Массив;
      УдаляемыеФайлы_.Добавить(ИмяКаталога_);

      ИмяЗашифрованногоФайла_ = ЗаписатьСтрокуBase64ВФайл(
         ДанныеДляРасшифровки.ЗашифрованныеДанныеBase64,,
            ИмяКаталога_);
      ИмяФайлаОткрытогоКлюча_ = ЗаписатьСтрокуBase64ВФайл(
         ДанныеДляРасшифровки.ОткрытыйКлючBase64,, ИмяКаталога_);
      ИмяФайлаСимметричногоКлюча_ = ЗаписатьСтрокуBase64ВФайл(
         ДанныеДляРасшифровки.СимметричныйКлючBase64,,
               ИмяКаталога_);
      ИмяФайлаИнициализационногоВектора_ =
               ЗаписатьСтрокуBase64ВФайл(
```

```
    ДанныеДляРасшифровки.ИнициализационныйВекторBase64,,
        ИмяКаталога_);

ИмяФайлаРезультата_ = СоздатьФайлНаСервере();
УдаляемыеФайлы_.Добавить(ИмяФайлаРезультата_);

ОписаниеОшибки_ = "";
МенеджерКриптографии_ = ОбъектВнешнейКомпонентыОбмена(
    АлгоритмПодписи, ОписаниеОшибки_);

Если Не ЗначениеЗаполнено(ОписаниеОшибки_) Тогда
    Попытка
        МенеджерКриптографии_.РасшифроватьФайл(
            СертификатПолучателя.СерийныйНомер,
            СертификатПолучателя.Поставщик,
            ИмяЗашифрованногоФайла_,
            ИмяФайлаРезультата_,
            ИмяФайлаОткрытогоКлюча_,
            ИмяФайлаСимметричногоКлюча_,
            ИмяФайлаИнициализационногоВектора_);
    Исключение
        ОписаниеОшибки_ = НСтр("ru='Не удалось расшифровать файл.'");
    КонецПопытки;
КонецЕсли;

РезультатСтруктура_ = Новый Структура;
Если ЗначениеЗаполнено(ОписаниеОшибки_) Тогда
    РезультатСтруктура_.Вставить("Выполнено", Ложь);
    РезультатСтруктура_.Вставить("ОписаниеОшибки",
        ОписаниеОшибки_);
```

```
        Иначе
            СтрокаBase64_ =
                ПрочитатьФайлВСтрокуBase64(ИмяФайлаРезультата_);
            РезультатСтруктура_.Вставить("Выполнено", Истина);
            РезультатСтруктура_.Вставить("СтрокаBase64", СтрокаBase64_);
        КонецЕсли;
        Возврат РезультатСтруктура_;
КонецФункции

Функция ЗаписатьСтрокуBase64ВФайл(СтрокаBase64,
    РасширениеФайла = Неопределено, ИмяКаталога = Неопределено)

    Если ИмяКаталога = Неопределено Тогда
        ИмяКаталога = КаталогВременныхФайлов();
    КонецЕсли;

    РасширениеФайла = СтрЗаменить(РасширениеФайла, ".", "");
    Если Не ЗначениеЗаполнено(РасширениеФайла) Тогда
        РасширениеФайла = "bin";
    КонецЕсли;
    ИмяВременногоФайла_ = ИмяКаталога +
        СтрЗаменить(Новый УникальныйИдентификатор, "-", "") +
        "." + РасширениеФайла;
    Попытка
        ДвоичныеДанные_ =
ПолучитьДвоичныеДанныеИзBase64Строки(СтрокаBase64);
        ДвоичныеДанные_.Записать(ИмяВременногоФайла_);
    Исключение
        ОписаниеОшибки_ =
            СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВСтроку(
```

```
        НСтр("ru = 'Во время сохранения файла %1 возникла ошибка.'"),
        ИмяВременногоФайла_);
    ВызватьИсключение ОписаниеОшибки_;
  КонецПопытки;
  Возврат ИмяВременногоФайла_;
КонецФункции

Функция ПрочитатьФайлВСтрокуBase64(ИмяФайла)
  СтрокаBase64_ = "";
  Попытка
    ДвоичныеДанные_ = Новый ДвоичныеДанные(ИмяФайла);
    СтрокаBase64_ =
      ПолучитьBase64СтрокуИзДвоичныхДанных(ДвоичныеДанные_);
  Исключение
    ОписаниеОшибки_ =
        СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВСтроку(
      НСтр("ru = 'Во время чтения файла %1 возникла ошибка.'"),
        ИмяФайла);
    ВызватьИсключение ОписаниеОшибки_;
  КонецПопытки;
  Возврат СтрокаBase64_;
КонецФункции

Функция ОбъектВнешнейКомпонентыОбмена(Алгоритм, ОписаниеОшибки)
  ВнешняяКомпонента_ = Неопределено;
  ИмяКомпоненты_ =
    ЭлектронныйДокументооборотСКонтролирующимиОрганамиВызовСер
вера.ПолучитьПутьВК();
  ПодключениеВыполнено_ = ПодключитьВнешнююКомпоненту(
    ИмяКомпоненты_, "ЭДОNative", ТипВнешнейКомпоненты.Native);
```

```
Если ПодключениеВыполнено_ Тогда
   Попытка
      ВнешняяКомпонента_ = Новый("Addin.ЭДОNative.CryptS");
Исключение
      ВнешняяКомпонента_ = Неопределено;
   КонецПопытки;
КонецЕсли;

Если ВнешняяКомпонента_ = Неопределено Тогда
   ВызватьИсключение
      НСтр("ru='Не удалось подключить внешнюю компоненту для работы
         с криптографией на сервере.'");
КонецЕсли;

ПараметрыКриптографии_ =
         ЗаполнитьПараметрыКриптографии(Алгоритм);
ДоступныеКриптопровайдеры_ =
   ПараметрыКриптографии_.ДоступныеКриптопровайдеры;
Криптопровайдеры_ = Новый Массив;
Если ЗначениеЗаполнено(ПараметрыКриптографии_.ОписаниеОшибки)
         Тогда
     ОписаниеОшибки = ПараметрыКриптографии_.ОписаниеОшибки;
Иначе
   Для Каждого Криптопровайдер_ из ДоступныеКриптопровайдеры_
         Цикл
   ОписаниеОшибки = "";
   КриптопровайдерИмя_ = Криптопровайдер_.Имя;
   КриптопровайдерПуть_ = Криптопровайдер_.Путь;
   КриптопровайдерТип_ = Криптопровайдер_.Тип;
   Попытка
```

```
        ВнешняяКомпонента_.СоздатьМенеджераКриптографии(
        КриптопровайдерИмя_,
        КриптопровайдерПуть_,
        КриптопровайдерТип_);
    Исключение
        ОписаниеОшибки = НСтр(
            "ru='Не удалось инициализировать компоненту.);
    КонецПопытки;

Если ЗначениеЗаполнено(ОписаниеОшибки) Тогда
    Продолжить;
КонецЕсли;

Имя_ = "";
Путь_ = "";
Тип_ = 0;
Если Криптопровайдеры_.Количество() = 0 Тогда
    Пока Истина =
            ВнешняяКомпонента_.ПолучитьСледующийКриптопровайдер(
        Имя_, Путь_, Тип_) Цикл
        Криптопровайдеры_.Добавить(Имя_);
    КонецЦикла;
КонецЕсли;

        // Проверяем установлен ли криптопровайдер в системе.
    Если Криптопровайдеры_.Найти(КриптопровайдерИмя_) =
                Неопределено Тогда
            Продолжить;
    Иначе
            Прервать;
```

```
    КонецЕсли;
  КонецЦикла;
  КонецЕсли;
  Возврат ВнешняяКомпонента_;
КонецФункции

Функция ЗаполнитьПараметрыКриптографии(Алгоритм = "")
  ЭтоLinux_ = ОбщегоНазначенияКлиентСервер.ЭтоLinuxКлиент();
  ПутьМодуляКриптографии_ = "";
  Если ЭтоLinux_ Тогда
    ПутьМодуляКриптографии_ =
    ЭлектронныйДокументооборотСКонтролирующимиОрганамиКлиентСе
        рвер.ПутьМодуляКриптографии();
  Если НЕ ЗначениеЗаполнено(ПутьМодуляКриптографии_) Тогда
    ОписаниеОшибки_ =
      НСтр("ru='Не указан путь модуля криптографии в настройках
        программы.'");
    Возврат Новый Структура("ДоступныеКриптопровайдеры,
ОписаниеОшибки",
    Неопределено,ОписаниеОшибки_);
    КонецЕсли;
  КонецЕсли;

  ДоступныеКриптопровайдеры_ =
  КриптографияЭДКОКлиентСервер.ПоддерживаемыеКриптопровайдеры
        (Алгоритм, ЭтоLinux_,,ПутьМодуляКриптографии_);
  Возврат Новый Структура("ДоступныеКриптопровайдеры,
        ОписаниеОшибки",
    ДоступныеКриптопровайдеры_,"");
КонецФункции
```

```
Функция СоздатьФайлНаСервере(Расширение = Неопределено) Экспорт
    ИмяКаталога_ = КаталогВременныхФайлов();
    Расширение = СтрЗаменить(Расширение, ".", "");
    Если Не ЗначениеЗаполнено(Расширение) Тогда
        Расширение = "bin";
    КонецЕсли;
    ИмяВременногоФайла_ = ИмяКаталога_ +
        СтрЗаменить(Новый УникальныйИдентификатор, "-", "") + "." +
            Расширение;
    Файл_ = Новый ТекстовыйДокумент;
    Файл_.Записать(ИмяВременногоФайла_);
    Возврат ИмяВременногоФайла_;
КонецФункции
```