МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

На правах рукописи

Шумилин Александр Сергеевич

МЕТОД ОБЕСПЕЧЕНИЯ БЕЗОПАСНОСТИ КОНФИДЕНЦИАЛЬНОЙ ИНФОРМАЦИИ В РАСПРЕДЕЛЕННОЙ МЕДИЦИНСКОЙ ОБЛАЧНОЙ СИСТЕМЕ

Специальность 2.3.6 - «Методы и системы защиты информации, информационная безопасность»

ДИССЕРТАЦИЯ

на соискание ученой степени кандидата технических наук

Научный руководитель - доктор технических наук, профессор Л.К. Бабенко

Таганрог – 2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире ежедневно каждый из нас применяет современные информационные и компьютерные технологии с целью решения существенного объема рутинных задач. Доступность и повсеместное использование сети Интернет обеспечивает популяризацию общения и знакомств в социальных сетях, широкие возможности поиска необходимой целевой информации, решение образовательных задач, создание программного обеспечения, предоставление развлекательных услуг, выполнение сложных математических расчетов, а также возможность предоставления различных социальных и медицинских услуг. Особую актуальность и существенный импульс в развитии информационных технологий придала, в том числе, пандемия коронавирусной инфекции COVID-19, вследствие чего практически все области жизнедеятельности и различные бизнес-процессы пришлось адаптировать под новые условия. Удаленный формат работы сотрудников, принципы и правила оказания различных услуг населению, проведение обучения в дистанционном формате и многие другие факторы активно способствовали масштабной информатизации и цифровизации тысяч предприятий и муниципальных организаций. Российская Федерация активно подхватила этот тренд, в первую очередь, для возможности предоставления услуг населению, запустив десятки информационных сервисов. Например, при поддержке Минцифры появилась возможность решать множество задач, связанных со сферой налогообложения, не выходя из дома. Отечественные сервисы «Госуслуги», «Кабинет налогоплательщика», а также «Единая государственная информационная система» в сфере здравоохранения демонстрируют обширный функционал и эффективно решают поставленные задачи.

Благодаря развитию информационных систем, процессы создания, накопления и обработки информации в различных сферах в настоящее время становятся все более актуальными. Особое внимание стоит уделить именно доменной области здравоохранения, потому что в последние несколько лет

активное развитие получили медицинские информационные системы (МИС), построенные на основе распределенных облачных архитектур, а также централизованные системы, управляемые единым сервером. Факт того, что эффективность оказываемой медицинской помощи полностью зависит от оперативности, удобства и полноты использования имеющейся у специалистов информации, является лежащим на поверхности и не требует дополнительных обоснований. Наличие задач, связанных с хранением, систематизацией и обработкой больших объемов данных обуславливает актуальность разработки и интеграции в медицинские учреждения современных информационных систем. Поскольку современная инфраструктура включает в себя электронные регистратуры и картотеки, наличие данных в электронном виде обеспечивает возможность медицинскому персоналу оперативно получать необходимую информации о пациенте, тем самым увеличивая скорость принятия решений, а как следствие ускоряя процесс постановки диагноза и выбор методов лечения [1].

Медицинские организации в силу законодательства [2–4] являются операторами персональных данных своих пациентов, поскольку принимают активное участие в сборе, накоплении, хранении, изменении, распространении и уничтожении любой информации, имеющей отношение к пользователям. В связи с этим, в настоящее время процессы создания, накопления и обработки информации в сфере здравоохранения становятся все более актуальными, что обусловлено масштабной информатизацией отрасли по всему миру. В процессе активного развития информационных технологий медицинские учреждения в ходе выполнения диагностических исследований обрабатывают и систематизируют большие объемы данных для последующей реабилитации и лечения пациентов [5]. Статистика работы единой медицинской информационной автоматизированной системы (ЕМИАС) города Москвы яркий пример такого развития цифровых технологий [6]. В качестве примера также можно выделить отечественный «Центр диагностики и телемедицины», который проводит эксперимент по использованию инновационных технологий в области компьютерного зрения (для анализа

медицинских изображений) в рамках системы здравоохранения города Москвы. В эксперименте принимают участие десятки компаний – разработчиков решений в области искусственного интеллекта. Эксперимент проводится на платформе ЕРИС (единый радиологический информационный сервис) и за время работы (с 2021 года) было проанализировано почти 10 миллионов обследований, 1,5 млн из которых в промежутке с января по май 2023 года [7].

Одной из проблем при проектировании медицинских информационных систем является потребность в интеграции систем защиты конфиденциальной информации [5], находящейся внутри медицинских информационных систем. Вместе с популяризацией таких систем участились и утечки данных, поскольку атаки хакеров стали главной проблемой, по отношении к персональным данным пользователей МИС. Данная проблема является актуальной как в мировом масштабе, так и в рамках Российской Федерации, что особенно важно в современных условиях импортозамещения и разработки отечественных систем безопасности. Статистика за 2020 год демонстрирует, что Россия вышла в лидеры по количеству утечек информации в мировых масштабах, а численный показатель составил 79,7% [8]. В 2022 году в нашей стране атакам подверглись такие крупные компании, как «Яндекс», «1С», «СДЭК», «Ozon», «Инфотекс», «Вкусно и точка», а катастрофическими последствиями стали размещения в открытом доступе сотен миллионов строк с персональными данными граждан страны. В качестве еще одного инцидента можно отметить утечку данных, после которой сканы паспортов выпускников и сотрудников высшей школы экономики оказались доступными в свободном формате. Общий рост утечек данных россиян вырос в 40 раз по сравнению с показателями 2021 года [9]. Кроме того, Роскомнадзор подтвердил утечки 600 млн записей о соотечественниках, которые были опубликованы в общий доступ. Суммарно более 140 очень крупных утечек персональных данных было зафиксировано за период с февраля по декабрь 2022 года [10].

В доменной области, связанной с медициной, к сожалению, утечки данных являются одной из самых актуальных проблем. Ситуация осложняется тем, что в

руки злоумышленников помимо личных данных пациентов попадают и результаты обследований, диагнозы, рекомендации к лечению, что усугубляет ситуацию и дает возможность злоумышленникам действовать более эффективно имея большой информационный ресурс для воздействия на жертву. Поскольку популяризация информационных систем для решения различных медицинских задач в России еще недостаточно актуальна и только набирает обороты, то объемы утечек в данной области кажутся не сильно большими на фоне аналогичных проблем зарубежных компаний. Однако подобные инциденты имеют место быть и были зафиксированы утечки среди различных представительств медицинских сообществ, в том числе государственного уровня.

После начала кампании по вакцинации населения Российской Федерации от новой коронавирусной инфекции, спустя несколько месяцев, в сети уже были опубликованы данные о 300 тысячах переболевших жителей Москвы. Вслед за этим еще одна утечка – база QR-кодов вакцинированных, а уже в 2023 году в открытый доступ попала база данных с информацией о сотне тысяч клиентов из лабораторий «Ситилаб». По общим подсчетам за 2022-2023 годы в нашей стране было похищено более 31 млн записей о пациентах из различных медицинских учреждений [10]. Если предположить, что каждая запись соответствует отдельному человеку, то это составляет более 20% от общего населения страны [11].

Что касается инцидентов в других странах, то подобные ситуации не является исключением. Так, например, американская компания Shields Health Care Group, специализирующаяся на диагностической визуализации МРТ и ПЭТ/КТ, радиологии и амбулаторных хирургических услугах, допустила утечку данных примерно двух миллионов человек (важно отметить, что это именно 2 млн уникальных записей), так как взлому хакеров подверглись внутренние системы компании [12]. Официальное заявление руководства гласит, что атака была обнаружена 28 марта 2022 года, после чего Shields Health Care Group наняла специалистов по кибербезопасности для расследования инцидента, однако изучение журналов логгирования показало, что хакеры имели доступ к системам

компании с 7 марта 2022 года по 21 марта 2022 года и могли получить доступ к данным, содержащим конфиденциальные данных о пациентах:

- фамилия, имя и отчество;
- номер социального страхования;
- дата рождения;
- адрес;
- диагноз;
- номер медицинской карты;
- идентификатор пациента;
- другая медицинская информация или данные о лечении.

Несанкционированный доступ со стороны злоумышленников продолжался на протяжении долгого времени, за которое удалось получить доступ к конфиденциальной информации.

В мае 2023 года компания «MCNA Dental» (Managed Care of North America), один из крупнейших в США поставщиков стоматологических услуг и сервисов медицинского страхования, сообщила о взломе своей информационной инфраструктуры. Киберпреступники украли данные приблизительно о 8,9 млн пользователей. В официальном уведомлении говорится, что подозрительная активность в компьютерной сети MCNA Dental была обнаружена 6 марта 2023-го, однако преступники получили доступ к серверам за неделю до фактической даты, а именно 26 февраля. Таким образом, у хакеров было достаточно времени, чтобы загрузить огромный массив конфиденциальных файлов [12]. Обе атаки демонстрируют критические недостатки информационных систем:

- возможность проникновения внутрь инфраструктуры и нахождение в ней длительное время будучи незамеченными,
- хранение данных в исходном незашифрованном виде, что позволяет свободно распоряжаться ими после факта компрометации.

В контексте источников конфиденциальной информации также можно отнести и различные показатели результатов медицинских исследований

пациентов, например, параметры физиологических сигналов, которые представлены в известных форматах хранения данных: EDF, CSV, DICOM, BMP, NRRD, NIFTI и т. д., их интерпретация и отклонения от нормы. Особую ценность могут представлять результаты расшифровки этих параметров, а также диагноз, поставленный специалистом и индивидуальные рекомендации к лечению, содержащие подробное описание необходимых препаратов, стоимость и места их приобретения.

Стоит отметить еще одну немаловажную проблему при компрометации медицинских данных. Помимо персональной информации, которую можно использовать в качестве шантажа и вымогательств есть еще один вид полезных данных – медицинские снимки, представляющие огромную ценность для разработчиков алгоритмов машинного обучения. Например, датасеты, содержащие снимки компьютерной томографии, рентген, флюорография, МРТ и другие способы визуализации крайне сложно найти бесплатно в открытом доступе. Многие компании готовы платить большие деньги за наборы таких изображений, потому что системы компьютерного зрения, способные анализировать такие данные являются очень дорогими и востребованными на сегодняшний день. Экономика данных здравоохранения растет существенными темпами, особенно в США, Великобритании, Китае и ряде других стран благодаря сотням начинающих компаний, которые ищут возможность улучшить медицину и здравоохранение с помощью инновационных способов обработки данных и новых технологий, главной из которых на данный момент конечно является искусственный интеллект. [14]. Таким образом, наличие базы данных в несколько миллионов изображений позволяет злоумышленникам получить дополнительный источник заработка продавая изображения заинтересованным компаниям, а сопутствующая информация о персональных данных будет являться подтверждением того факта, что злоумышленник является правообладателем. Соответственно наличие перечисленных выше факторов ставит под угрозу безопасность современных МИС,

поскольку киберпреступники могут извлечь много выгоды при успешной атаке на подобные системы.

В качестве примера можно проанализировать медицинское обследование, в процессе которого пациенту требуется выполнить назначенную процедуру и передать результаты лечащему врачу. Исследования могут передаваться в медицинские организации в автоматическом режиме по сети или посредством беспроводных каналов связи. Несанкционированное получение доступа со стороны злоумышленника к передаваемым конфиденциальным данным с целью их компрометации может оказаться критичным для пациента и для медицинского учреждения. Это обусловлено не только дискредитацией данных, но и подменой результатов медицинских обследований. Проблема усугубляется тем, что в случае искажения результатов обследования лечащий специалист не будет осведомлен, то возникает риск постановки некорректного диагноза, что впоследствии может нанести вред здоровью и даже поставить под угрозу жизнь пациента.

В связи с тем, что требованиями Федерального закона № 152-ФЗ «О персональных данных» установлена необходимость защиты персональных данных пациентов медицинских организаций, **ключевой задачей** при использовании облачной медицинской информационной системы для хранения, систематизации и обработки данных является **обеспечение безопасности** хранимой и передаваемой информации. Кроме того, медицинская информационная система является объектом критической инфраструктурой и попадает под действие Федерального закона №187 «О безопасности критической информационной инфраструктуры», который регулирует отношения в области безопасности таких объектов в целях устойчивого функционирования при проведении в отношении объекта компьютерных атак [4]. Такие задачи решались отечественной МИС от компании N3.Health, более того компания заявляет, что является официальным оператором передачи данных в единую систему государственной информационной системой в сфере здравоохранения (ЕГИСЗ), в соответствии с постановлениями Правительства №140 и №852.

Безопасность данных в облачной системе обеспечена следующим образом:

1. Для размещения информационной системы персональных данных N3. Health используется дата-центр, соответствующий уровню Tier III и требованиям законодательства к мероприятиям по физической защите информационных систем персональных данных [4].

2. Каждая информационная система персональных данных размещается в защищенной по требованиям ФСТЭК изолированной виртуальной среде, построенной на базе решений VMware.

3. Комплекс мер по обеспечению физической и информационной безопасности гарантирует полное соответствие требованиям Роскомнадзора, ФСТЭК и ФСБ.

4. Применяются сертифицированные ФСТЭК программно-аппаратные средства защиты информации.

Резюмируя вышеописанные факты, компания пользуется уже готовыми решениями в области информационной безопасности [15].

В диссертации содержится решение следующей **фундаментальной научной задачи**: в условиях попыток несанкционированных вторжений со стороны злоумышленников на данные, находящиеся в медицинских информационных системах, необходимо в пределах имеющихся ресурсов разработать такой метод защиты персональных данных, который обеспечит высокий уровень безопасности при условии отсутствия потерь производительности работы МИС.

Междисциплинарность работы обусловлена наличием поставленных, тесно связанных задач на стыке инженерных направлений науки (математики, информационной безопасности, криптографии, программирования, алгоритмизации) и прикладных задач в области медицины.

Тематика работы относится к приоритетному направлению развития науки, технологий и техники в РФ (утвержден Указом Президента РФ от 7 июля 2011 г., №899) «Безопасность и противодействие терроризму», чем **подтверждается актуальность** исследований в данной области.

Тематика исследований соответствует Стратегии научно-технологического развития Российской Федерации, утвержденной Указом Президента Российской Федерации от 01.12.2016 г. № 642, а именно, пункту **20 «а»** (переход к передовым цифровым, интеллектуальным производственным технологиям, роботизированным системам, новым материалам и способам конструирования, **создание систем обработки больших объемов данных,** машинного обучения и искусственного интеллекта) в части разработки средств анализа и систематизации больших объемов медицинских данных, а также **пункту 20 «д»** (**противодействие** техногенным, биогенным, социокультурным угрозам, терроризму и идеологическому экстремизму, а также **киберугрозам** и иным источникам опасности для общества, экономики и государства) в части разработки алгоритмов обеспечения безопасности конфиденциальной информации.

Целью работы является повышение эффективности механизмов обеспечения безопасности медицинских информационных систем за счет разработки и внедрения в их архитектуру метода обеспечения защиты конфиденциальных медицинских данных на основе протокола разделения секрета, реализованного по схеме Шамира, позволяющего снизить количество потенциальных угроз.

Достижение поставленной цели предполагает необходимость решения следующих задач:

1. Анализ существующих медицинских информационных систем, их архитектурных решений, а также способов обеспечения защиты данных в таких системах.

2. Разработка архитектуры распределенной облачной платформы хранения и систематизации конфиденциальных данных медицинских обследований, позволяющей получать и обрабатывать данные с различных аппаратных средств диагностики. Универсальность архитектуры должна позволять оперировать медицинскими форматами данных, зарегистрированных с использованием разных аппаратных средств, работающих на основе протокола

HL7;

3. Разработка метода обеспечения безопасности, построенного на основе схемы разделения секрета Шамира, для защиты медицинских данных пациентов, циркулирующих в облачной медицинской информационной системе;

4. Программная реализация разработанного метода обеспечения безопасности конфиденциальных медицинских данных;

5. Выполнение экспериментальных исследований и тестирование разработанного метода защиты путем получения практических результатов.

Объектом исследования являются технологии хранения, передачи и защиты конфиденциальной информации, находящейся в распределенной медицинской информационной системе, построенной на основе облачной архитектуры.

Предметом исследования являются методы и схемы разделения секрета, обеспечивающие конфиденциальность медицинских данных пациентов.

Теоретической основой исследования являются методы работы с большими данными, а также теоретические основы математической логики, основ алгоритмизации, методов программирования, криптографических методов защиты информации, методы оптимизации, методы теории реализации математических моделей с использованием прикладных программ.

Научная новизна. В работе получены следующие новые научные результаты:

1. Метод обеспечения безопасности конфиденциальной информации в распределенной медицинской облачной системе (МИС), отличающийся тем, что он основан на использовании схемы разделения секрета Шамира, позволяющей повысить безопасность МИС путем усложнения процесса компрометации файла с конфиденциальными данными. Файл разделяется на фрагменты, которые хранятся на разных серверах, что усложняет процедуру доступа к файлу со стороны злоумышленников, потому что для восстановления исходных данных требуется собрать части воедино.

2. Результаты моделирования работы протокола разделения секрета на основе схемы разделения секрета Шамира отличающиеся применением библиотеки MPI, позволяющей производить обработку и взаимодействие параллельных процессов. Получены оценки сокращения времени разделения файла медицинского обследования на фрагменты и восстановление в исходное состояние;

3. Архитектура облачной медицинской распределенной системы, отличающаяся возможностью интеграции средств для защиты конфиденциальных данных, позволяющих повысить защищенность медицинских данных пациентов. Архитектура медицинской системы позволяет работать как с классическими форматами представления файлов (PNG, PDF, EDF, TXT, JPG и т.д.), так и с медицинскими файлами (DICOM, Nifti, NRRD), обеспечивая возможность безопасного хранения данных на основе разработанного метода.

4. Результаты экспериментальных исследований с оценками защищенности системы при использовании предложенного метода.

Теоретическая значимость исследования заключается в формировании новых подходов к организации систем защиты медицинских данных на основе использования схем разделения секрета, теоретических знаний о процессе интеграции способов обеспечения защиты в медицинских информационных системах, имеющих облачную архитектуру. Полученные результаты могут служить основой для исследований в различных областях и направлениях информационной безопасности: исследование свойств безопасности криптографических протоколов и алгоритмов защиты информации, разработка, оценка и анализ алгоритмов защиты информации в облачных вычислениях. Кроме того, полученные результаты могут быть использованы при проектировании новых или улучшении действующих распределенных медицинских систем с целью повышения уровня защищенности данных пациентов (файлов с результатами медицинских обследований), что играет ключевую роль для улучшения сферы отечественной медицины.

Практическая значимость работы состоит в том, что полученные в ходе исследования результаты могут быть использованы при проектировании и разработке медицинских информационных систем с учетом обеспечения безопасного хранения данных. Использование разработанного метода позволяет снизить количество угроз со стороны злоумышленников и повысить уровень защищенности информации, циркулирующей в распределенной медицинской системе, построенной на основе современных криптографических протоколов.

В ходе разработки метода защиты конфиденциальных данных и проведенных экспериментальных исследованиях получены новые научные результаты, подтверждающие эффективность применения предложенного метода на основе схемы разделения секрета по сравнению с использованием средств защиты без интеграции представленного метода. Количество угроз в рамках медицинской системы может быть снижено на 45% по сравнению с состоянием системы до внедрения предлагаемого метода.

Полученные практические результаты и универсальность метода обеспечения безопасности медицинских данных позволяют использовать предлагаемый метод и программно-технические решения, основанные на нем, при разработке новых алгоритмов и подходов к защите информации в текущих условиях активной цифровизации сферы здравоохранения и импортозамещения отечественной медицины.

Публикации. Основные положения диссертации опубликованы в 20 научных печатных работах, в том числе: 4 – в ведущих рецензируемых научных журналах, входящих в перечень ВАК РФ (1 из которых входит в базу RSCI), 3 – в научных рецензируемых изданиях, индексируемых в базе Scopus, 13 – в материалах конференций и других изданиях. Получено 1 свидетельство о государственной регистрации программ для ЭВМ

Соответствие паспорту специальности. Диссертация соответствуют паспорту научной специальности 2.3.6 – Методы и системы защиты информации, информационная безопасность, пункту №5 «Методы, модели и средства

(комплексы средств) противодействия угрозам нарушения информационной безопасности в открытых компьютерных сетях, включая Интернет» и пункту №15 «Принципы и решения (технические, математические, организационные и др.) по созданию новых и совершенствованию существующих средств защиты информации и обеспечения информационной безопасности».

Реализация и внедрение результатов работы. Результаты диссертационных исследований, подтвержденные соответствующими актами, используются в:

1. Научно-производственной деятельности ООО «СиВижинЛаб» (г. Таганрог), а именно: выполнена апробация метода обеспечения безопасности конфиденциальной информации в рамках собственной МИС, разработанной в организации;

2. Научно-производственной деятельности ООО «Нейротех» (г. Таганрог), а именно выполнена интеграция авторского метода в подсистему безопасности в рамках информационной системы одного из пилотных проектов компании.

3. Научно-производственной деятельности ООО «Инженерный центр Интегра» (г. Таганрог), а именно выполнена апробация результатов проведенных исследований в рамках внутренних задач компании.

4. В учебно-исследовательском процессе на кафедре безопасности информационных технологий имени О.Б. Макаревича ИКТИБ ЮФУ: материалы диссертации планируется использовать в учебном процессе при подготовке студентов специальности 10.05.03 «Информационная безопасность автоматизированных систем» в дисциплине «Криптографические методы защиты информации». Результаты диссертационного исследования были использованы при выполнении гранта РФФИ в рамках проекта №20-37-90138 – «Аспиранты 2020» на тему «Разработка и реализация алгоритмов обеспечения безопасности конфиденциальной информации в распределенной медицинской облачной системе».

Апробация результатов. Основные положения и результаты работы докладывались и обсуждались на 9 отечественных и международных конференциях.

Личный вклад. Все результаты, составляющие содержание диссертации, получены автором самостоятельно.

Структура и объем диссертации. Диссертация написана на русском языке, состоит из введения, пяти глав, заключения, списка используемой литературы из 94 наименований и приложения. Полный объём диссертации составляет 146 страниц (в том числе приложения 28 стр.), включая 11 рисунков и 18 таблиц.

Во введении обоснована актуальность темы диссертации, определен объект исследования, сформулированы цель и задачи работы, показаны методы исследования, определена научная новизна, практическая и теоретическая значимость полученных результатов, приведены научные результаты, выносимые на защиту.

В первой главе представлен обзор подходов к обеспечению защиты конфиденциальных данных в распределенных информационных системах, а также современные направления развития таких подходов. В первую очередь, проведен анализ исследований авторов в рассматриваемой предметной области. Отмечены преимущества и недостатки наиболее известных методов и алгоритмов обеспечения безопасности. Продемонстрирована целесообразность использования методов разделения секрета в качестве основы при разработке метода обеспечения безопасности конфиденциальных данных с целью улучшения защиты информации, повышения защищенности результатов медицинских исследований от атак и угроз несанкционированного доступа со стороны злоумышленников.

Обосновано применение метода на основе схемы разделения секрета и проведен обзор наиболее популярных и эффективных схем, подходящих для использования в предлагаемом методе защиты данных. Сформулированы частные задачи, решением которых достигается цель исследования.

Во второй главе представлена архитектура медицинской облачной информационной системы, которая выступает в качестве объекта защиты. Представлены типы данных, которые находятся в экосистеме и представляют особую ценность, поскольку могут использоваться как медицинскими, так и исследовательскими организациями. Также продемонстрирована детализация уровней входящих состав медицинской системы дополнительных модулей, которые используются для обработки данных пациентов, взаимодействия ПО, авторизации и управлением потоками данных.

В главе описан функционал облачной информационной системы и продемонстрирована логика взаимодействия между подсистемами. Также уделено внимание протоколу HL7, благодаря которому реализована возможность интеграции с различными аппаратными средствами и другими медицинскими информационным системами.

Третья глава посвящена обоснованию выбора схемы разделения секрета, а также описанию экспериментальных исследований и полученным результатам.

Применяя различные схемы разделения секрета Шамира, Асмута-Блума, Блэкли и Карнин-Грин-Хеллмана, были проведены эксперименты для определения подходящей схемы для дальнейшего использования в качестве основы метода обеспечения безопасности. В данной главе также рассматривается реализация схемы Шамира, ее главная идея и основные фазы.

В первую очередь выполняется анализ ресурсоемкости рассматриваемых схем разделения секрета и проводятся эксперименты с различными значениями n и k (основные параметры) для оценки схем с целью получения результатов по разделению файла на части и восстановлению воедино. Отмечено, что схема разделения секрета Шамира оказалась оптимальной по сравнению с другими кандидатами, потому что время выполнения основных операций для данной схемы меньше в десятки и даже сотни раз (по сравнению с схемой Карнин-Грин-Хеллмана).

Помимо экспериментов, посвященных оценке времени работы, проводится анализ безопасности предлагаемого метода с помощью анализатора безопасности криптографических протоколов, при использовании различных схем разделения секрета. Метод, реализованный на основе протокола разделения секрета Шамира, был протестирован в программной среде анализатора с применением формализованного языка интерпретации протоколов. По результатам экспериментов определено, что после внедрения метода на основе схемы Шамира количество возможных угроз системы снижается на 45%. Таким образом, схема Шамира является эффективной, по сравнению с другими схемами, поэтому ее использование в качестве основы метода обеспечения безопасности является полностью обоснованным.

Четвертая глава посвящена разработке метода обеспечения защиты конфиденциальных данных в облачной медицинской информационной системе. Метод обеспечения защиты медицинских данных состоит из этапов шифрования и расшифрования файла, а также выполнения операции по разделению файла на фрагменты для их хранения на серверах и восстановлению файла воедино. Все шаги этапов подробно описаны.

В рамках разработанного метода предлагается использование схемы разделения секрета для разделения файла медицинского обследования на фрагменты при нахождении в экосистеме МИС, что в свою очередь позволяет предотвратить возможность компрометации файла, а также случайную утерю или уничтожение. Описываются все шаги работы метода в рамках предложенной архитектуры МИС, которые охватывают все роли, предусмотренные в информационной системе.

В пятой главе произведено моделирование распараллеливания операций разделения и слияния долей секрета по схеме Шамира с использованием библиотеки MPI. Была выполнена оценка времени разделения файла на фрагменты и слияние фрагментов обратно в единый файл. Результаты эксперимента показали, что можно добиться почти линейного ускорения и сократить время генерации 255

долей тестового файла в 1,87 раза, а время восстановления этих долей обратно в единый файл в 1,74 при работе в многопоточном режиме. Эксперимент был проведен на локальном компьютере, имеющем характеристики: 16 ГБ ОЗУ, Intel Core i7 – 11800H: 8-ядер по 2,3 ГГц с гиперпоточностью до 16 потоков.

ГЛАВА 1. ОБЗОР ПОДХОДОВ К ОБЕСПЕЧЕНИЮ ЗАЩИТЫ КОНФИДЕНЦИАЛЬНЫХ ДАННЫХ В РАСПРЕДЕЛЕННЫХ МЕДИЦИНСКИХ ИНФОРМАЦИОННЫХ СИСТЕМАХ

Облачные вычисления повсеместно используются в распределенных средах для удовлетворения потребностей пользователей в ресурсах и услугах. В современных условиях системы здравоохранения полагаются на подключенные через интернет интеллектуальные гаджеты. Эти устройства управляют огромными объемами данных, которые обрабатывают и собирают различные медицинские датчики, сохраняя при этом множество параметров, таких как производительность, пропускная способность и задержка. Часто в таких системах реализована балансировка нагрузки между интеллектуальными рабочими устройствами. Как распределенный, так и централизованный подход к управлению огромными объемами данных достигается за счет балансировщиков нагрузки, например различные алгоритмы обучения с подкреплением и методы Q-обучения для планирования ресурсов. Эти алгоритмы используются в облачных системах здравоохранения для прогнозирования наилучшего метода управления потреблением ресурсов.

Большое количество медицинских учреждений в нашей стране, а также по всему миру используют технологии удаленного мониторинга и обследования пациентов, выполняя перемещение полученных данных о пациенте между рабочими станциями различных специалистов и медучреждений. Благодаря использованию распределенных медицинских систем происходит сокращение времени посещения больниц, снижение больничных расходов и улучшение качества медицинской помощи. Функционал таких систем имеет огромное количество опций и часто представлен пользовательскими приложениями, обеспечивающими удаленный доступ к физиологическим данным пациента. Инструменты. Соответственно, можно записывать, передавать, хранить и

обрабатывать большие массивы медицинских данных за короткое время за счет интеграции с инфраструктурой клиники и периферийных вычислений. Однако применение децентрализованных систем приводит к тому, что необходимо решить вопрос с безопасностью данных и предотвратить возможность проникновения со стороны злоумышленника в медицинские данные пользователей, являющиеся конфиденциальными.

1.1 Распределенные информационные системы

В современном мире существует множество определений распределенной системы. В общем виде их можно трактовать следующим образом:

1) набор независимых друг от друга рабочих станций, которые составляют для своих пользователей общую систему, которая имеет единую инфраструктуру рабочего пространства;

2) набор связанных программных модулей, которые взаимодействуют между собой. Каждый программный модуль может быть представлен в виде программного компонента, который функционирует как автономный элемент.

Программные компоненты и пользователи в распределенных системах решают существенный объем рутинных задач независимо друг от друга. Подобный характер и принцип работы распределенных систем скрывает от пользователей различия между рабочими станциями и способы связи между ними.

К важным свойствам распределенных информационных систем относится возможность обеспечения единства работы, как пользователей, так и программных компонент. Такие системы должны быть легко масштабируемыми, а также обеспечивать высокий уровень отказоустойчивости. Если какой-либо сервис распределенной системы не доступен в определенный момент времени, информационная система должна продолжать обеспечивать стабильную работу, не информируя пользователя о подобных событиях.

С целью обеспечения связи различных ПК в распределенных системах является частой практикой проектировать дополнительный промежуточный уровень ПО, который выступает связующим звеном между прикладным уровнем и операционной системой [16].

Одна из основных и важных задач распределенной обработки данных –это удобный доступ к потребляемым ресурсам и контроль над совместным применением рабочих станций, серверов, файловой инфраструктуры, web-страниц и сети. Для того чтобы система распределенной обработки данных была эффективной, она должна соответствовать определенным требованиям, а именно: прозрачности, открытости, гибкости и масштабируемости.

Интерфейсы в распределенных системах определяют службы и компоненты, которые, как правило, описываются языком определения интерфейсов (Interface Definition Language. Интерфейс описывается с помощью синтаксиса служб – имен доступных функций, типов параметров, возвращаемых значений и др. Семантику служб описать сложнее, т. к. эти описания задаются средствами естественного языка [17].

Открытые системы должны обладать возможностью выполнять миграцию данных между системами без изменения интерфейса [17]. Основная ключевая особенность открытой распределенной системы – это возможность простого способа для настройки и конфигурации. Гибкая система внутренних компонент позволяет упростить процедуру администрирования таких систем, а также позволяет с легкостью масштабироваться как аппаратно, так и с помощью дополнительного программного функционала. В процессе организации и настройке открытых распределенных систем основной задачей является конфигурация системы таким образом, чтобы все входящие в состав компоненты системы имели возможность простой замены в случае необходимости.

Что касается масштабируемости систем, то она может быть оценена, принимая во внимание 3 параметра:

1. Размер – с его помощью оценивается условие трудоемкости подключения к системе новых пользователей и дополнительных ресурсов;

2. Площадь системы – необходима для выбора тех пользователей и ресурсов, которые могут быть распределены;

3. Управление системой строится таким образом, чтобы обеспечить максимальное удобство пользователей при работе.

В процессе масштабируемости распределенных систем часто может снижаться производительность. Для того, чтобы производительность системы не стремилась к некоторым пороговым значениям, при которых использование такой системы является затруднительным производится ограничение масштабируемости. В качестве примеров это может быть реализовано следующим образом: 1 сервер на нескольких пользователей; доступные в онлайн режиме некоторые файлы, к которым требуется доступ большинства сотрудников; настройка маршрутизации данных, в контексте локальных сетей.

1.1.1 Аппаратные решения

Распределенные системы можно разделить на две группы: мультипроцессорные, мультикомпьютерные. Системы, в которых компьютеры используют память совместно, называются мультипроцессорными. Если в распределенной системе каждый компьютер работает со своей памятью, то такие системы называются мультикомпьютерными.

Мультипроцессорные системы имеют единое адресное пространство, которое используется всеми процессорами. В мультикомпьютерных системах каждая машина использует свою память, например – обычная сеть компьютеров.

1.1.2 Программные решения

Наибольшее влияние на аппаратную часть распределенных систем оказывают программные решения. Они влияют на удобство работы пользователя, помогают им совместно управлять общими ресурсами (памятью, данными,

процессором и сетью). Распределенные системы выполняют функции менеджеров ресурсов, т. е. их работа аналогична работе операционной системы [17].

Операционные системы в распределенных системах делятся на две категории: сильно связанные и слабо связанные. Сильно связанными операционными системами называются распределенные операционные системы, которые используются для управления мультипроцессорными, мультикомпьютерными и гомогенными системами. Их основная их цель – скрывать подробности управления аппаратным обеспечением от пользователя [17].

Слабо связанными операционными системами называются сетевые операционные системы, которые используются для управления гетерогенными мультикомпьютерными комплексами. Наряду с традиционными функциями управления ресурсами они обеспечивают доступ удаленных клиентов к локальному программному, аппаратному обеспечению и данным [17].

При создании распределенной системы недостаточно служб сетевой операционной системы. К ним необходимо добавить дополнительные элементы для организации прозрачной структуры системы. Данные элементы образуют промежуточный уровень. Таким образом, программные средства играют основную роль в построении распределенной системы промежуточного уровня, между распределенными приложениями и операционными системами. Общая структура распределенной системы с промежуточным уровнем показана на рисунке 1.1.

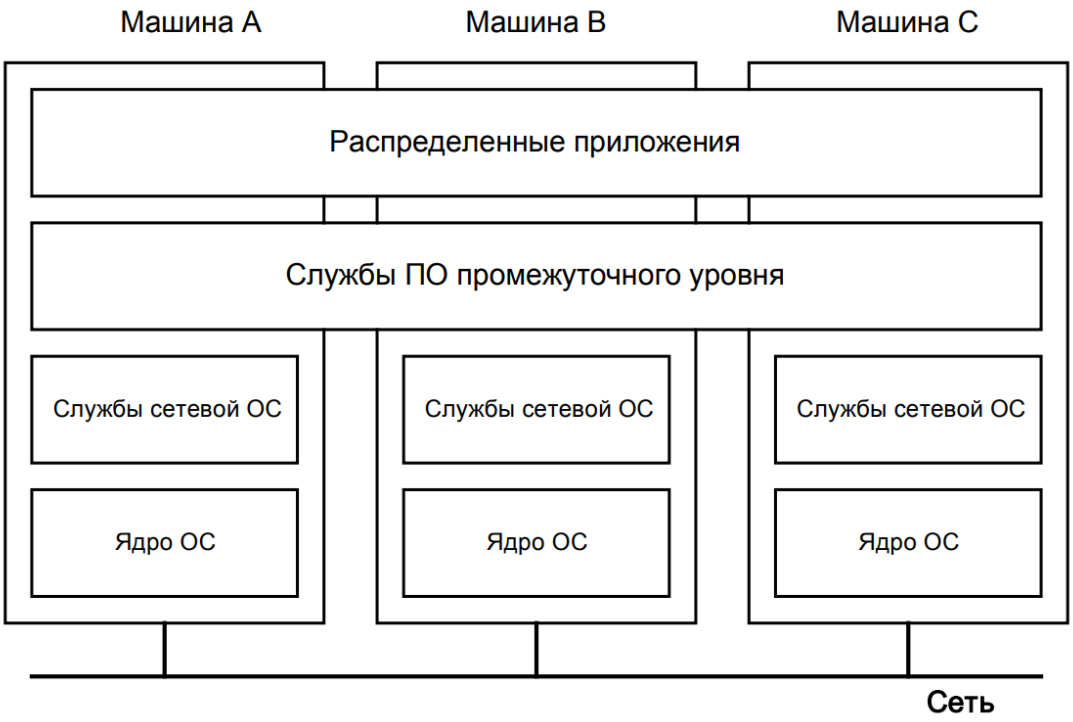


Рисунок 1.1 Структура децентрализованной системы с промежуточным уровнем

1.2 Анализ исследований в предметной области

Использование существующих способов обеспечения безопасности данных в информационных системах, среди работ авторов, опубликованных в научной литературе, содержат различные недостатки [18–20, 89], такие как: проблема распределения ключей для симметричного шифрования, а также достаточно строгие требования к времени работы алгоритмов, потреблению памяти и вычислительным ресурсам в целом. Работы некоторых научных коллективов, предлагающих подходы с использованием гомоморфного шифрования [21, 87-88] для решения задачи обеспечения безопасности данных выглядят не применимы с точки зрения скорости реализации таких алгоритмов, потому что операции выполняются достаточно медленно и проигрывают классическим алгоритмам в несколько раз. В другой научной работе [22] автор делает акцент на том, что наличие только гомоморфного шифрования будет недостаточным для обеспечения

безопасности зашифрованного текста, например, при адаптивных атаках с выбранным зашифрованным текстом.

Сама по себе проблема безопасности — основная причина, по которой многие организации не спешат внедрять облачные вычисления, поскольку поставщики облачных услуг сталкиваются с тремя важными юридическими проблемами:

1) Конфиденциальность данных напрямую связана с обеспечением защиты данных клиента, предоставляемых третьей стороной. Происходит переход контроля от клиента к третьей стороне, предоставляющей облачный сервис и технологический функционал. Нарушение любого письменного договора с третьей стороной может повлиять на конфиденциальность данных клиента.

2) Целостность данных связана с сохранением исходных данных, которые хранятся физически в различных местах на серверах, эксплуатируемых и контролируемых многочисленными организациями по всему миру. Кроме того, это связано с правильным выбором поставщиками облачных услуг необходимых данных (по указанию клиента), которые будут использоваться в облачных вычислениях. Следовательно, это является серьезной проблемой, поскольку поставщики облачных услуг не могут гарантировать ни защиту данных, которые в некоторых случаях хранятся в их центрах обработки данных, расположенных по всему миру, ни сам процесс выбора данных поставщиками облачных услуг.

3) На конфиденциальность данных также влияет функция взаимосвязи нескольких сервисов в облаке, как указано в [23], [24]. Поэтому такие проблемы влияют на облачную инфраструктуру с точки зрения администрирования. Кроме того, стоит отметить и сложности, связанные с передачей контроля поставщикам облачных услуг, которые определены как центральная административная единица. Чтобы обеспечить конфиденциальность и целостность данных, многие исследователи указали, что шифрование является одним из наиболее эффективных и безопасных методов для облачных вычислений [25 – 27, 90].

Анализ существующих исследований в рассматриваемой области среди отечественных ученых показал аналогичные результаты. Научные статьи из открытых источников по тематике исследования показали, что мнения соотечественников и иностранных коллег о необходимости применять методы и алгоритмы шифрования данных полностью схожи. В одной из рассмотренных научных работ коллектив авторов исследовал возможность использования облачных реализаций для повышения эффективности процесса интеграции медицинских информационных систем. Авторы определили шифрование, как один из эффективных способов обеспечения защиты данных в контексте МИС и продемонстрировали варианты использования [28, 91]. В работе подчеркивается проблема распределения ключей, которая присуща симметричным способам шифрования, что безусловно осложняет процесс работы с такими системами. Представленная проблема является наиболее значимой для технологии шифрования и заключается в том, что хранение ключей на облачном сервере не имеет смысла.

В научной работе [29] Керейтова М. Р. и Малыш В.Н. подробно описывают аналогичную проблему – обеспечение информационной безопасности конфиденциальных данных пациентов медицинских учреждений, как одну из наиболее актуальных при создании и проектировании медицинских информационных систем. Вопрос обеспечения безопасности рассматривается на примере распределенной информационной системы Департамента охраны здоровья населения Кемеровской области, охватывающей все лечебно-профилактические учреждения (ЛПУ) Кемеровской области. Авторами предложен комплексный подход к решению проблемы: введение контроля за рабочими станциями пользователей на предмет необычно высокой активности, отслеживание актуальности всех обновлений для имеющихся операционных систем и установленного программного обеспечения, использование многоуровневой аутентификации пользователей, которая предполагает использование USB-ключей, смарт-карт, токенов, паролей, файловых ключей. Описанный подход

может быть применим при хранении информации, однако он не учитывает механизмов обеспечения защиты в аспекте предотвращения утечки данных и несанкционированного доступа при передаче и обработки информации в системах с архитектурой клиент-сервер. В своей работе Керейтова М. Р. и Малыш В.Н продемонстрировали способы и средства, обеспечения защиты на уровне доступа к рабочим станциям пользователей системы.

Бойченко И. В. в своей научной работе [30] отмечает важность проблемы реализации прав граждан в области защиты персональных данных пациентов. Автор рассматривает возможность использования медицинских информационно-аналитических центров в структуре здравоохранения, акцентируя внимание лишь на правовом и юридическом аспектах проблемы. Предварительный анализ, проведенный автором, позволяет сделать вывод о большом потенциале использования облачных технологий в решении задач современного здравоохранения. Однако для их повсеместного внедрения требуется грамотное техническое решение, направленное на разработку методов обеспечения безопасности передаваемой информации и конфиденциальности персональных данных пациентов.

Что касается облачных технологий, Rohan Jathanna в своем исследовании [18] отмечает уязвимость облачных систем к атакам со стороны злоумышленников (DDoS-атаки, атаки с целью проникновения на сервер, несанкционированный доступ к базам данных). Для предотвращения потери доступа к конфиденциальным данным автор предлагает использовать возможности средств резервного копирования. Противодействие несанкционированному доступу достигается путём использования алгоритмов шифрования. Предлагаемые автором подходы имеют существенные недостатки. Система резервного копирования требует большого количества дополнительных вычислительных ресурсов и ресурсов памяти, а также обеспечения нового объекта защиты (ресурса с резервной копией). Эффективность используемых алгоритмов шифрования снижается в связи с наличием проблемы распределения ключей: необходимо предусмотреть

возможность передачи ключа от клиента на сервер по защищенному каналу связи. Реализация защищенного канала сложная и дорогостоящая задача поскольку требует материальные ресурсы и исследования в предметной области. Последствием компрометации ключа шифрования является полная потеря доступа к конфиденциальным данным.

В своей работе, посвященной классическим криптографическим алгоритмам Кривошеева Д. А. выделяет основные недостатки использования ассиметричных систем шифрования в медицинских облачных платформах [31]:

- большие затраты вычислительных ресурсов,
- количество времени, которое требуется для реализации вычислительных процессов.

В поисках компромиссного решения автор предлагает альтернативный подход к созданию симметричного ключа шифрования, основанный на использовании физиологического сигнала пациента в качестве «физиологической» подписи. Существенным недостатком предлагаемого метода является тот факт, что физиологические сигналы (электрокардиограмма, электромиограмма, электроэнцефалограмма и др.) могут изменяться в течение жизни человека. Соответственно, ключ шифрования, сформированный ранее, спустя определённое время может стать недействительным и, как следствие, доступ к персональным данным станет невозможен. Не менее важной проблемой предложенного метода видится возможность доступа к данным только со стороны их обладателя (пациента, который предоставил физиологический сигнал для формирования ключа шифрования). В результате возможность получения доступа к результатам обследования другими лицами (например, лечащим доктором, родственниками пациента, аналитиком системы здравоохранения и др.) затрудняется или вовсе исключается.

Анализ исследований позволяет сделать вывод о том, что рассмотренные авторами проблемы и предлагаемые способы обеспечения безопасности данных,

содержат недостатки и имеется потенциал для их устранения. Наиболее значимые из них представлены ниже:

- проблема распределения ключей для симметричных способов шифрования,
- высокие требования к вычислительным ресурсам, времени и памяти,
- привязка физиологического ключа к конкретному пациенту,
- скорость работы алгоритмов шифрования (актуально для гомоморфного шифрования).

В работах [32–37] авторы подходят к обеспечению защиты данных в облачных системах через гибридную модель безопасности. Ее концепция заключается в том, что отправитель генерирует 2 симметричных ключа. Для загрузки и шифрования данных в облаке размер данных варьируется соответствующим образом. Процесс шифрования для предложенной модели выглядит следующим образом. Сперва выполняется получение хэша данных для последующей проверки целостности. Затем генерируется случайный ключ blowfish для шифрования обычного текста в зашифрованный текст с помощью blowfish алгоритма. Выходной зашифрованный текст подается в качестве входных данных для шифрования его в зашифрованный текст с использованием алгоритма AES. Выходные данные алгоритма AES подаются на вход другого алгоритма для тройного шифрования, который действует как дополнение к зашифрованному тексту AES, который является окончательным зашифрованным текстом. ECC (elliptic curve cryptography) ключ, полученный со стороны получателя, используется для шифрования случайных ключей с целью их безопасной передачи по сети. Зашифрованный текст вместе с зашифрованными ключами и хешем обычного текста загружается в облако. Таким образом blowfish, AES и ECC объединены в некую гибридную модель для защиты данных от внешних воздействий. AES — это симметричный подход, тогда как ECC — хорошо известный асимметричный метод. Ключи создаются с использованием криптографии на основе эллиптических кривых (ECC), а для шифрования и

кодирования используются AES и Blowfish. В качестве входных данных принимаются текстовые файлы разных размеров. Обзоры основаны на многих факторах, включая корреляцию, время корреляции, размер шифрования файла и продолжительность шифрования файла. Эксперименты автора показывают, что этот подход может быть использован в сфере здравоохранения, поскольку технология подходит, а время, необходимое для шифрования и дешифрования, минимально. Однако при работе с файлами больших размеров, таких как изображения или видеозаписи дневного мониторинга потребление ресурсов значительно увеличивается из-за набора различных алгоритмов шифрования, а также отсутствия оптимизации при взаимодействии между данными алгоритмами. Однако один из критичных недостатков данного подхода – это передача ключей, используемых для шифрования по сети (незащищенному каналу связи).

В последнее время исследователями была проделана большая работа по обеспечению безопасности в облаках именно с использованием различных методов шифрования из-за резкого повышения требования в области безопасности как на законодательном, так и на уровне медицинской этики. Уже существует несколько различных алгоритмов, используемых в гибридной криптографии в облачных средах для защиты данных. Чтобы повысить эффективность большинства распространенных алгоритмов, исследователи предлагают все новые и новые гибридные модели для криптографической защиты.

Гибридное криптографическое решение, для облачных баз данных применительно к медицинским файлам, было предложено группой исследователей в работах [38–43]. Предложенная ими методология позволяет шифровать ключи с использованием асимметричного алгоритма при симметричном шифровании содержимого. Исследователи применили метод модульного умножения Монтгомери, чтобы повысить эффективность алгоритма RSA. Медицинские данные шифруются с помощью Blowfish и обрабатываются с использованием обновленной технологии RSA, сохраняясь при этом внутри облака. Результаты моделирования продемонстрировали преимущества гибридного подхода –

короткое время выполнения функции шифрования, эффективное управление ключами и большие простые числа для генерации ключей. Это также показывает, что с точки зрения сроков шифрования и дешифрования предлагаемая гибридная стратегия превосходит конкурирующие подходы (т. е. чистые AES и Blowfish).

Система, разработанная исследователем Batra [44] работает с использованием гибридного шифрования, которое состоит из четырех методов шифрования. Основная идея гибридного шифрования — объединить несколько алгоритмов шифрования для усиления облачной безопасности и защиты личной информации. Эти четыре алгоритма — AES, DES, RC4 и стеганография — используются в предлагаемой автором модели. Процесс состоит из трех частей: первая часть шифруется с помощью RC4, вторая часть — с помощью DES, а третья часть — с помощью AES. Затем ключ будет скрыт с помощью стеганографии — метода, при котором секретное сообщение записывается таким образом, что ни у кого, кроме отправителя и предполагаемого получателя, не возникает сомнений в его существовании.

Как заявили Biswas и коллектив ученых [45–46], стеганография повышает безопасность, а гибридная криптография ее улучшает. Авторы предложили использовать RSA и AES для создания такой гибридной криптографии. Они использовали AES для шифрования сообщения, открытый ключ RSA использовался для шифрования симметричного ключа, используемого для шифрования сообщений, что по заявлению авторов повышает безопасность. Для создания цифровой подписи создается хэш-значение сообщения, которое затем еще раз шифруется с использованием открытого ключа RSA. Эта цифровая подпись помогает проверить целостность сообщения на принимающей стороне. Полное сообщение создается путем объединения всех трех зашифрованных файлов — зашифрованного сообщения, зашифрованного ключа и зашифрованной цифровой подписи. Для включения всего сообщения использовался метод стеганографии LSB (least significant bit). Устойчивость предлагаемой системы к атакам, по мнению авторов весьма высока.

Гибридная архитектура шифрования, представленная в [47-49] включает модули для разбиения на части данных, шифрования, распространения, загрузки, индексирования, поиска и слияния данных. Их гибридный подход использует блочное шифрование и дешифрование AES и сети Фейстеля. Данные разделены на фрагменты и хранятся в мультиоблачной среде.

Еще одна модель гибридного шифрования была построена на методе, в котором дайджест (выход значения хеш-функции) сообщения и цифровая подпись были созданы на клиентском компьютере. Затем файл шифруется открытым ключом с использованием алгоритма AES и зашифрованный файл отправляется на облачный сервер. При получении на сервере сохраненное хеш-значение сравнивается с хэш-значением файла, загруженного в облако. Если он соответствует зашифрованному файлу, он расшифровывается с помощью закрытого ключа и отправляется запрошенному клиенту.

В таблице 1 показаны различные алгоритмы, которые использовались исследователями, а также их достижения и недостатки предложенных алгоритмов.

Таблица 1.1 – Анализ работ исследователей в смежной области

Автор, год	Используемый алгоритм	Достижения	Недостатки
Soman, 2017 [50]	AES, ECDSA + SHA256	Шифрование файлов осуществляется с помощью AES. SHA256 и ECDSA обеспечивают аутентификацию.	Небезопасный ключ для шифрования данных
Chinnasamy, 2018 [38]	Blowfish, авторский RSA	Blowfish преобразует обычный текст в зашифрованный текст. Расширенный RSA используется для шифрования секретного ключа Blowfish.	RSA невозможно реализовать из-за его размера, что делает его более ресурсоемким.

Batra, 2019 [44]	RC4, AES, DES + стеганография	Шифрование с использованием авторского алгоритма было выполнено после разделения на 3 части. Секретный ключ распространяется с использованием стеганографии изображений.	RC4 нельзя использовать с небольшим потоком данных.
Yoshita, 2021 [33]	AES, RSA	Двойное шифрование данных с использованием AES и RSA	Низкая скорость передачи данных и задержка из-за двойного шифрования.
Biswas, 2020 [45]	AES, RSA + стеганография	AES используется для шифрования сообщения, а ключ шифруется с помощью RSA. Затем все сообщение включается с использованием стеганографии LSB.	Низкая скорость передачи данных из-за большого количества данных.
Viswanath, 2021 [47]	AES и алгоритмы Фейстеля	Модель разделяет данные на блоки по 256 бит и затем зашифровывает их с помощью алгоритма AES и сетей Фейстеля.	Шифры ограничены в возможности распараллеливания.

Предлагаемый в диссертации подход направлен на исключение существующих недостатков за счет применения метода обеспечения безопасности персональных данных на основе схемы распределения секрета. Основными отличиями метода является повышение уровня безопасности для медицинских данных, а также скорость работы алгоритмов разделения и слияния долей секрета.

Дополнительно, метод обладает возможностью масштабируемости, что является очень важным фактором при работе в рамках облачных распределенных информационных систем, большая часть из которых построена на основе микросервисных архитектур. Стоит отметить, что метод может использоваться для обеспечения безопасности критической информационной инфраструктуры, которую представляют собой МИС в силу Федерального закона № 187 (закон регулирует отношения в области обеспечения безопасности критической информационной инфраструктуры для ее устойчивого функционирования при проведении в отношении ее компьютерных атак.

1.3 Технология одноразовых паролей

Технология одноразовых паролей представляет собой способ аутентификации, при котором для пользователя предлагается пароль на один сеанс входа в личный кабинет информационной системы. Кроме того, действие одноразового пароля может быть ограничено по времени.

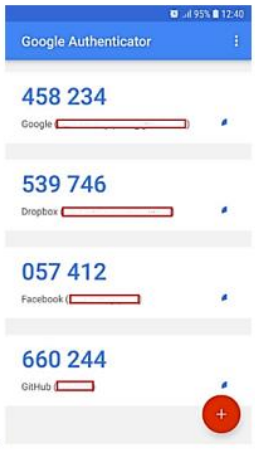
Ключевое преимущество одноразового пароля по сравнению с многоразовым заключается в том, что пароль не может быть использован повторно, а значит при компрометации такого пароля доступ к новой сессии будет не возможен. Применение одноразовых паролей не защищает от атак, основанных на активном вмешательстве в канал связи, используемый для аутентификации (атака «человек посередине») [51–53].

Технология одноразовых паролей считается надежной, но существует и ряд недостатков. Наиболее серьезная уязвимость – возможность подмены сервера аутентификации. В таком случае пользователь будет отправлять запросы злоумышленнику, который сможет использовать их для доступа к настоящему серверу. Другая уязвимость присуща только синхронным методам и связана с тем, что существует риск рассинхронизации информации на сервере и в программном или аппаратном обеспечении пользователя. Если в системе начальными данными

служат показания внутренних таймеров, и по каким-то причинам они перестают соответствовать [53].

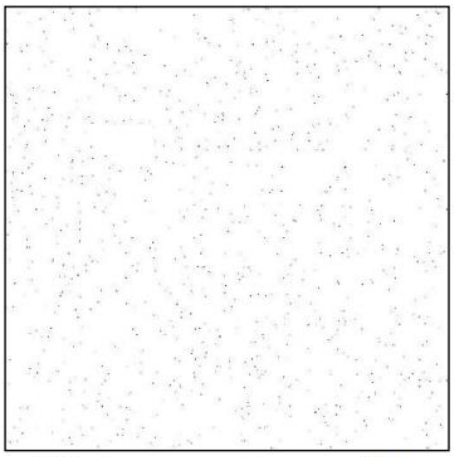
Процесс генерации пароля в таких системах осуществляется в зависимости от сценариев использования. В современном мире методы делятся на 3 основные группы:

1. Генераторы, основанные на псевдослучайных числах, использующих в качестве параметра временную метку. В практике широко применяющиеся методы генерации одноразовых паролей (OTP — One-Time Password) основанные на синхронизации времени, и среди них особое значение имеет TOPT (Time-based One-Time Password Algorithm) алгоритм. На рисунке 1.2 представлена интерпретация OTP [53].



2.

a) OTP генерированные в приложении Google Authenticator



b) Графическое изображение OTP

Рисунок 1.2 Интерпретации OTP (слева – генерация в приложении Google Authenticator, справа – графическое представление)

2. Генераторы псевдослучайных чисел, основанные на использовании счетчиков в качестве параметра. Одним из примеров, который относиться к данным методам генерации OTP является алгоритм HOTP (HMAC-based OTP). Функционирование данного алгоритма идентично со схемой, приведенной на рисунке 1.2, однако имеется отличие, которое заключается в использовании счетчика вместо метки времени. На рисунке 1.3 показан пример того, как выглядит

в графическом виде 1000 OTP, сгенерированных на основе HOTP из единого ключа [53].

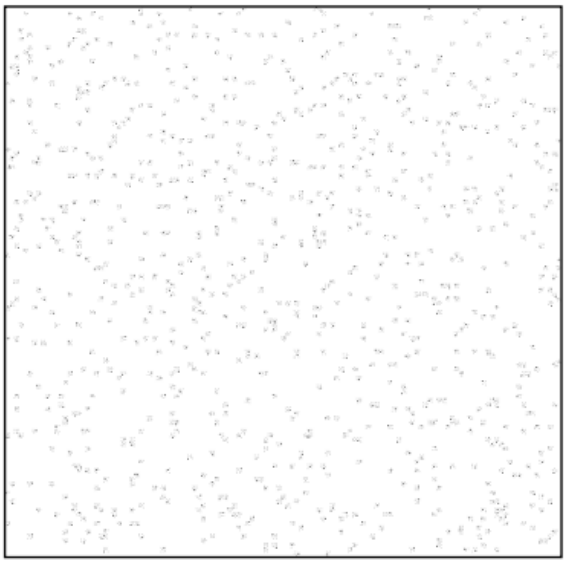


Рисунок 1.3 – Графический вид результата алгоритма HOTP

3. Генераторы паролей на основе накопления определенных символов. Иными словами, это OTP, которые состоят только из цифр и являются не стойкими к атакам полного перебора [53].

Информационный ресурс «passwordmeter» позволяет оценить OTP на предмет стойкости, результаты для нескольких вариантов показаны на рисунках 1.4, 1.5.

Test Your Password		Minimum Requirements
Password:	123456	• Minimum 8 characters in length • Contains 3/4 of the following items: - Uppercase Letters - Lowercase Letters - Numbers - Symbols
Hide:	☐	
Score:	4%	
Complexity:	Very Weak	

Additions	Type	Rate	Count	Bonus
Number of Characters	Flat	+(n*4)	6	+ 24
Uppercase Letters	Cond/Incr	+((len-n)*2)	0	0
Lowercase Letters	Cond/Incr	+((len-n)*2)	0	0
Numbers	Cond	+(n*4)	6	0
Symbols	Flat	+(n*6)	0	0
Middle Numbers or Symbols	Flat	+(n*2)	4	+ 8
Requirements	Flat	+(n*2)	1	0

Рисунок 1.4 Оценка простого пароля OTP в системе «The password meter»

Test Your Password		Minimum Requirements
Password:	123456Aa!$	• Minimum 8 characters in length • Contains 3/4 of the following items: - Uppercase Letters - Lowercase Letters - Numbers - Symbols
Hide:	☐	
Score:	100%	
Complexity:	Very Strong	

Additions		Type	Rate	Count	Bonus
✓	Number of Characters	Flat	+(n*4)	10	+ 40
✓	Uppercase Letters	Cond/Incr	+((len-n)*2)	1	+ 18
✓	Lowercase Letters	Cond/Incr	+((len-n)*2)	1	+ 18
✓	Numbers	Cond	+(n*4)	6	+ 24
✓	Symbols	Flat	+(n*6)	2	+ 12
✓	Middle Numbers or Symbols	Flat	+(n*2)	6	+ 12
✓	Requirements	Flat	+(n*2)	5	+ 10

Рисунок 1.5 Оценка сложного пароля OTP в системе «The password meter»

1.4 Способ проверки подлинности пользователей на основе схемы рукопожатия

Для взаимной проверки подлинности пользователей применяется процедура на основе механизма «рукопожатия». Такая процедура базируется на механизмах контроля типа запрос-ответ или отметка времени и заключается во взаимной проверке ключей, используемых сторонами. Процедуру рукопожатия обычно применяют в компьютерных сетях при организации сеанса связи между пользователем и сервером либо сервером и сервером.

Рисунок 1.6 показан представляет собой схему проверки подлинности между пользователями A и B (описание процедуры рукопожатия для 2 пользователей, которые пользуются единым ключом.

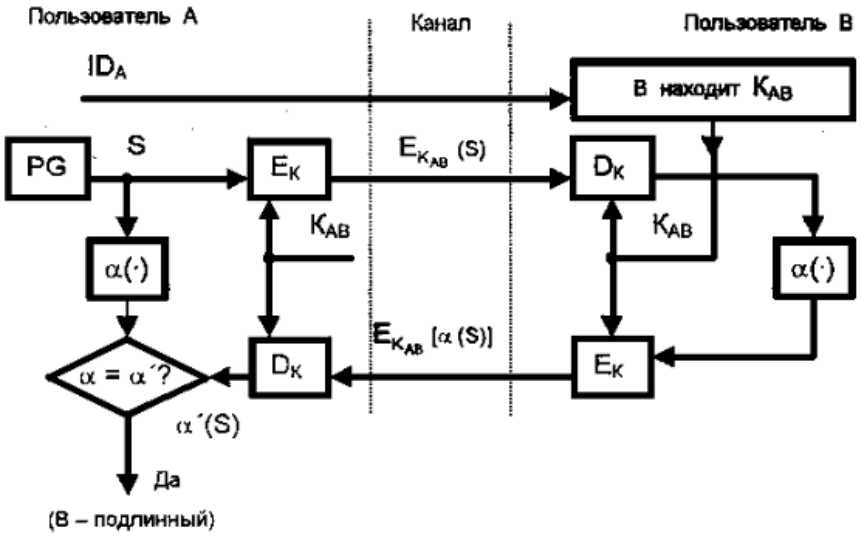


Рисунок 1.6 – Схема проверки подлинности между пользователями A и B

Алгоритм такой процедуры состоит из 6 шагов:

1. Пользователь *A* генерирует случайную последовательность *S* с помощью генератора *PG* и посылает пользователю *B* в виде криптограммы $E_{K \oplus B}(S)$.

2. Пользователь *B* расшифровывает эту криптограмму и раскрывает исходный вид последовательности *S*.

3. Оба пользователя преобразуют последовательность *S*, используя открытую одностороннюю функцию α(·).

4. Пользователь *B* шифрует сообщение α(*S*) и отправляет эту криптограмму пользователю *A*.

5. Пользователь *A* расшифровывает эту криптограмму и сравнивает полученное сообщение α'(*S*) с исходным α(*S*). Если эти сообщения равны, пользователь *A* признает подлинность пользователя *B*.

6. Пользователь *B* проверяет подлинность пользователя *A* таким же способом.

Обе проверки образуют процедуру рукопожатия, которая выполняется в начале любого сеанса связи между любыми двумя сторонами [94].

Достоинством процедуры рукопожатия является отсутствие секретной информации для обоих пользователей во время процедуры подтверждения подлинности.

1.5 Протоколы разделения секрета

Разделение секрета — это способ распределения секрета среди группы участников, каждому из которых достаётся своя доля. Секрет может воссоздать только коалиция участников из первоначальной группы, причём входить в коалицию должно не менее некоторого числа, которое известно всем участникам. Схемы разделения секрета применяются в случаях, когда существует значимая вероятность компрометации одного или нескольких хранителей секрета, но вероятность недобросовестного сговора значительной части участников считается пренебрежимо малой [56–57].

Самый простой пример схемы разделения секрета выглядит следующим образом: пусть имеется группа из t человек и сообщение s длины n, состоящее из двоичных символов. Если подобрать случайным образом такие двоичные сообщения $s_1, \dots, s_N$, что в сумме они будут равняться s, и распределить эти сообщения между всеми членами группы, получится, что прочесть сообщение будет возможно только в случае, если все члены группы соберутся вместе. В такой схеме есть существенная проблема: в случае утраты хотя бы одного из членов группы, секрет будет утерян для всей группы безвозвратно. Для решения такой проблемы был разработан подход на основе пороговой схемы [58].

В отличие от процедуры разбиения секрета, где $t=n$, в процедуре разделения секрета количество долей, которые нужны для восстановления секрета, может отличаться от того, на сколько долей секрет разделён. Такая схема носит названия пороговой схемы *(t, n)* где n — количество долей, на которые был разделён секрет, а t — количество долей, которые нужны для восстановления секрета. Идеи схем $t \neq n$ были независимо предложены Ади Шамиром и Джорджем Блэкли [59–61].

1.5.1 Схема Ади Шамира

Идея схемы заключается в том, что для интерполяции многочлена степени *k -1* необходимо *k* точек. Интерполяция невозможна если известно меньшее число точек. Если требуется разделить секрет между *n* людьми таким образом, чтобы восстановить его могли только *k* человек ($k \leq n$), он скрывается его в формулу многочлена степени *k-1*. Восстановить этот многочлен и исходный секрет можно только по *k* точкам. Количество же различных точек многочлена не ограничено (на практике оно ограничивается размером числового поля, в котором ведутся расчёты).

Достоинством схемы Шамира является масштабируемость. Чтобы увеличить число пользователей, необходимо добавить соответствующее число несекретных элементов к существующим. В то же время компрометация одной части секрета переводит схему из *(k, n)* пороговой в *(k-1, n-1)* пороговую.

Реализация алгоритма по схеме Шамира состоит из трех фаз:

1. *Подготовительная фаза.*

Доверенный сервер выбирает коэффициенты $a_{k-1}, a_{k-2}, \dots, a_1$ случайным образом, а также выбирается p – простое число, которое больше, чем *m.* Число p можно сообщать всем участникам, оно задает конечное поле размера *p.* Над этим полем строится многочлен степени *k-1* (случайно выбираются все коэффициенты многочлена кроме *m*):

$$f(x) = (m + a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + \cdots + a_1 x) \bmod p\,,$$

где m – разделяемый секрет (файл с обследованием), $a_{k-1}, a_{k-2}, \dots, a_1$- некоторые случайные числа [55, 94]

2. *Генерация долей*

Любой представитель коалиции получает свою долю секрета (значения построенного многочлена в *n* различных точках, важно отметить, что $x \neq 0$).

$$k_l = f(1) = (m + a_{k-1}1^{k-1} + a_{k-2}1^{k-2} + \cdots + a_1 1) \bmod p$$

…

$k_n = f(n) = (m + a_{k-1}n^{k-1} + a_{k-2}n^{k-2} + \cdots + a_1 n) \bmod p$

Каждая сторона получает долу секрета k_i с номером i, а также сторонам сообщается степень многочлена *k-1* и размер поля *p*. Коэффициенты $a_{k-1}, a_{k-2}, \ldots, a_1$ и сам секрет *m* далее не требуются.

3. Восстановление секрета.

На данном этапе любые *k* участников, зная координаты *k* различных точек многочлена, имеют возможность восстановить многочлен и все его коэффициенты, включая последний из них — разделяемый секрет.

Особенностью схемы является то, что вероятность раскрытия секрета в случае произвольных *k-1* долей оценивается как p^{-1}. То есть в результате интерполяции по *k-1* точке, абсолютно любой элемент поля может являться секретом с одинаковой вероятностью [94]. Попытка полного перебора всех возможных долей не позволит злоумышленникам получить информацию о секрете. Чтобы восстановить секрет, можно воспользоваться интерполяционной формулой, например формулой Лагранжа – многочлен минимальной степени, принимающий заданные значения в заданном наборе точек, то есть решающий задачу интерполяции.

$$f(x) = \sum_i l_i(x) y_i \bmod p,$$

где $l_i(x) = \prod \frac{x - xj}{x_i - x_j} \bmod p$, ($x_i$, y_i) – координаты точек многочлена [55].

1.5.2 Схема Блэкли

Используя реализацию схемы Блэкли, можно создать *(t, n)* – схему разделения секрета для любых *t* и *n*: для этого надо использовать размерность пространства, равную *t* и каждому из *n* участников дать одну гиперплоскость (учитывая условие, что множество *n*-мерных плоскостей всегда имеют пересечение

в 1 точке), проходящую через секретную точку. Тогда любые *t* из *n* гиперплоскостей будут однозначно пересекаться в секретной точке [55, 59, 60].

Схема Блэкли менее эффективна, чем схема Шамира: в схеме Шамира каждая доля такого же размера, как и секрет, а в схеме Блэкли каждая доля в *t* раз больше. Однако существуют улучшения схемы Блэкли, позволяющие повысить её эффективность, но такие улучшения более ресурсозатратны.

Морис Миньотт, Асмут и Блум предложили две схемы разделения секрета, основанные на китайской теореме об остатках. Для некоторого числа (в схеме Миньотта это сам секрет, в схеме Асмута — Блума — некоторое производное число) вычисляются остатки от деления на последовательность чисел, которые раздаются сторонам. Благодаря ограничениям на последовательность чисел восстановить секрет может только определённое число сторон [55, 61].

Пусть количество пользователей в группе равно *n*. В схеме Миньотта выбирается некоторое множество попарно взаимно простых чисел $\{m_1, m_2, \dots, m_n\}$ таких, что произведение *k-1* наибольших чисел меньше, чем произведение *k* наименьших из этих чисел. Пусть эти произведения равны *M* и *N*, соответственно. Число *k* называется порогом для конструируемой схемы по множеству $\{m_1, m_2, \dots, m_n\}$. В качестве секрета выбирается число *S* такое, для которого выполняется соотношение $M<S<N$. Части секрета распределяются между участниками группы следующим образом: каждому участнику выдается пара чисел $(r_i, m_{i)}$, где $r_i = S\ (mod\ m_i)$ [55].

Чтобы восстановить секрет, необходимо объединить $n \geq k$ фрагментов. В этом случае получим систему сравнений вида $x \equiv r_i = (mod\ m_i)$., множество решений которой можно найти, используя китайскую теорему об остатках. Секретное число *S* принадлежит этому множеству и удовлетворяет условию $S < m_1 \cdot m_2 \cdot m_t$. Также несложно показать, что если число фрагментов меньше *k*, то, чтобы найти секрет *S*, необходимо перебрать порядка $\frac{N}{M}$ целых чисел. При правильном выборе чисел m_i такой перебор практически невозможно реализовать. К примеру, если разрядность

m_i будет от 129 до 130 бит, а $k < 15$, то соотношение $\frac{N}{M}$ будет иметь порядок 2^{100}[55-56].

1.5.3 Схема Асмута — Блума

Такая схема является доработанной схемой Миньотта. В отличие от схемы Миньотта, её можно построить в таком виде, чтобы она была совершенной.

Подготовительная фаза.

Пусть M – общий секрет. Вычисляем простое число P, которое больше любого из секретов, который в свою очередь будет разделяться ($P > max(M)$).

Также вычисляются N взаимно простых чисел $d_1, d_2, \ldots, d_N$, таких что:

- $\forall i: d_i > P$;
- $\forall i: d_i < d_{i+1}$;

Затем выбирается такое *r,* что

$$\sqrt[K]{M`} < d_i \ll \sqrt[K-1]{M`}, i = 1, \ldots, n, \text{ где } M` = M + rP$$

Генерация долей секрета.

Доля секрета, которая выдается i-му участнику схемы представляет собой три числа $\{P, d_i, k_i\}$, где $k_i \equiv M`(mod\ d_i)$[2].

Восстановление секрета.

Когда K долей секрета собираются вместе и используя китайскую теорему об остатках, решается система уравнений относительно *M`:*

$$\{M` \equiv k_1(mod\ d_1)\ M` \equiv k_2(mod\ d_2)\ \ldots\ M` \equiv k_K(mod\ d_K)$$

Таким образом после решения, участники получают общий секрет M.

Также существуют и другие схемы, основанные на решении систем уравнений, например схема Карина-Грина-Хеллмана [55].

1.5.4 Схема Карнин – Грин – Хеллмана

Схема основывалась на невозможности решить систему с *K* неизвестными, имея менее *K* уравнений.

В рамках данной схемы выбираются *N+1* *K*-мерных векторов $V_0, V_1, \ldots, V_n$ так, чтобы любая матрица размером $K \times K$ составленная из этих векторов, имела ранг *K*. Пусть вектор *U* имеет размерность *K*.

Генерация долей секрета.

Секретом в схеме является матричное произведение $U^T V_0$. Долями секрета являются произведения $U^T V_i, 1 \leq i \leq n$.

Восстановление секрета.

Имея любые *K* долей, можно составить систему линейных уравнений размерности $K \times K$, неизвестными в которой являются коэффициенты *U*. Решив данную систему, можно найти *U*, а имея *U*, можно найти секрет. При этом система уравнений не имеет решения в случае, если долей меньше, чем *K* [55].

1.6 Выводы

В первой главе был рассмотрен обзор подходов к обеспечению защиты конфиденциальных данных в распределенных медицинских информационных системах, а также выполнен обзор способов проектирования таких систем. Проведен анализ различных способов обеспечения безопасности в рамках распределенных систем, таких как технология одноразовых паролей и применение протоколов разделения секрета. В результате были сформулированы следующие задачи, которые решаются в диссертационном исследовании:

1. Разработка метода обеспечения безопасности, построенного на основе схемы разделения секрета для защиты медицинских данных пациентов.
2. Определение наиболее подходящую схему разделения секрета для использования в качестве основы в рамках разработанного метода.
3. Разработка архитектуры облачной платформы медицинской информационной системы с учетом возможности интеграции метода защиты.
4. Программная реализация разработанного метода, выполнение моделирования работы метода в контексте обеспечения безопасности

конфиденциальных медицинских данных;

5. Проведение экспериментальных исследований и тестирование разработанного метода защиты, получение практических результатов для оценки эффективности.

ГЛАВА 2. РАЗРАБОТКА АРХИТЕКТУРЫ МЕДИЦИНСКОЙ ОБЛАЧНОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ

2.1 Описание архитектуры объекта защиты

Для решения основных задач - хранения и обработки информации необходимо разработать архитектуру информационной системы, которая бы позволила автоматизировать процессы взаимодействия и эффективно обеспечивать защиту систематизированных данных на основе методов шифрования. Для такой МИС предъявляются высокие требования к уровню безопасности, а также необходимо реализовать масштабируемость в условиях увеличения количества получаемых данных, прозрачность и открытость. Это означает что распределенная система должна скрывать разницу в способах представления данных и способах доступа к различным ресурсам. Такое свойство и определяет прозрачность доступа к данным. Также предполагается, что система должна обеспечивать прозрачность местоположения ресурса (скрывать его физическое расположение). Ресурсы должны иметь только логические имена, такие как указатель URL ресурса, не имеющий информации о местоположении файла.

Автором предложена архитектура распределенной платформы сбора, хранения, обработки и защиты конфиденциальных медицинских данных, которая состоит из нескольких уровней [62–63]. Общая схема представлена на рисунке 2.1. Данные, циркулирующие в системе могут использоваться как медицинскими, так и исследовательскими организациями, а платформа может выполнять задачу цифровизации бизнес-процессов за счет упрощения доступа специалистов к информации (сервисы телемедицины, SaaS сервисы, автоматизированные системы поддержки принятия решения) и научно-исследовательскую задачу (исследование алгоритмов обеспечения защиты информации, фармакологические исследования, анализ больших объемов данных различных типов обследований, разработку алгоритмов машинного обучения для автоматизированного анализа исследований).

Облачная платформа обладает следующим функционалом: предоставление удобных инструментов для передачи данных между пользователями системы; создание интерфейса и обширной базы данных для исследовательской системы анализа (в том числе, с использованием алгоритмов машинного обучения); создание интерфейсов для интеграции в существующие медицинские информационные системы (МИС); создание облачного сервиса (SaaS) для хранения, классификации и обработки данных, созданных с помощью различного оборудования с поддержкой множества популярных форматов данных; создание подсистемы обеспечения защиты результатов обследований с использованием предложенного автором метода [63, 67–68].

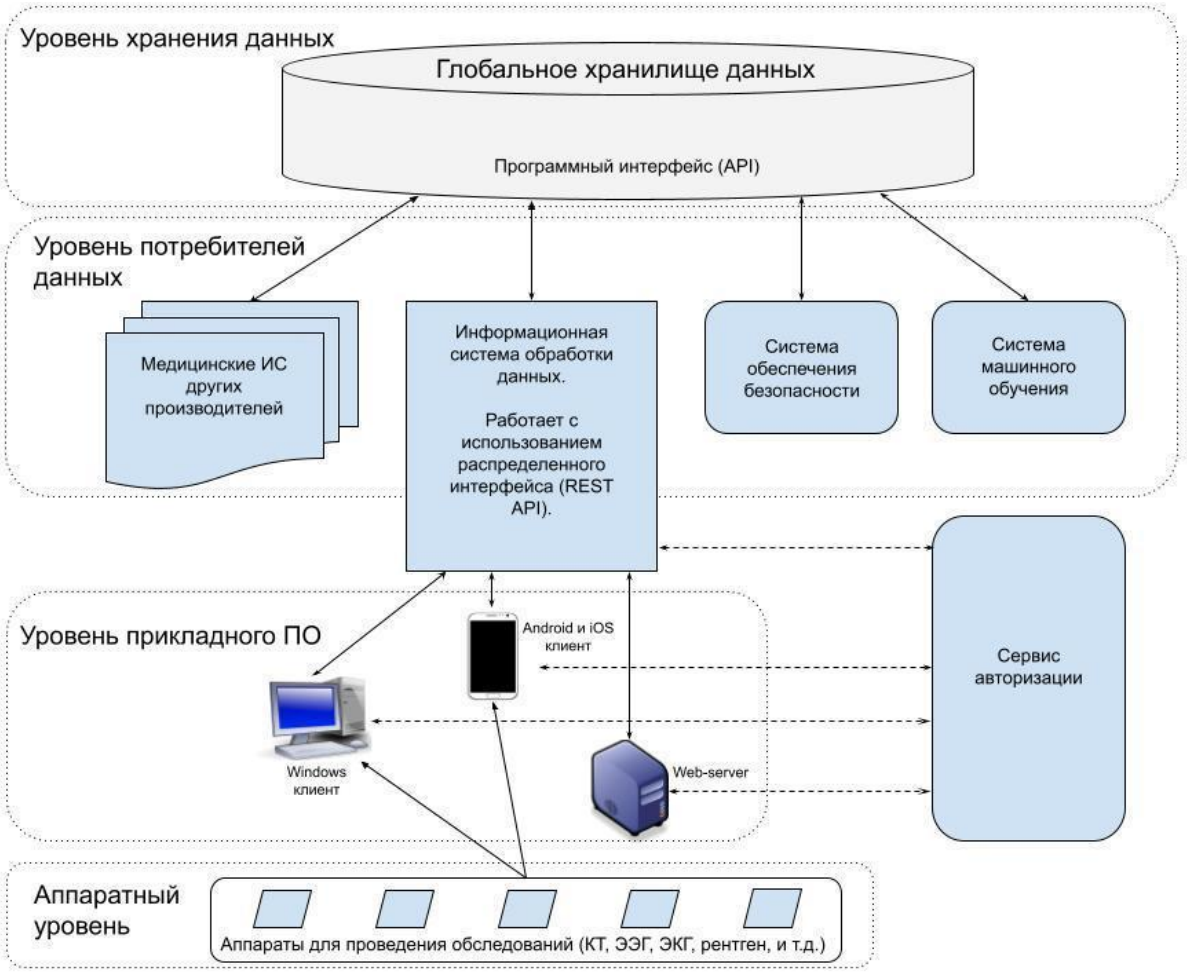


Рисунок 2.1 Общая схема организации модулей платформы

Архитектура медицинской системы состоит из четырех основных уровней, которые используются для обработки данных.

Уровень хранения данных – представляет собой глобальное хранилище конфиденциальных данных, которое включает в себя базу данных для хранения необработанных обследований и отчетов, а также антропометрическая, диагностическая, демографическая информация о пациентах с учетом персональных данных. Таким образом хранилище содержит полный объем информации для исследований и обучения машинных алгоритмов, но идентификация пациента возможна только по защищенному идентификатору.

Уровень потребителей данных включает системы, которые принимают и обрабатывают данные из Глобального хранилища или передают в него новые данные. Этот уровень связан с уровнем хранения данных через стандартизированный программный интерфейс (Storage API). Ядром платформы является *информационная система обработки данных*, реализующая управление потоками информации, логику групп и ролей, обеспечивает взаимодействие с конечными клиентскими приложениями по средствам распределенного интерфейса (REST API). Ключевой подсистемой данного уровня является подсистема механизмов защиты, которая представляет собой метод обеспечения защиты конфиденциальных данных на основе схемы разделения Шамира, предложенный, реализованный и исследованный автором [65, 67, 84].

Уровень прикладного ПО содержит программные средства конечных клиентов, где формируются и отображаются медицинские данные (обследования в виде сигналов, отчетные и персональные данные пациента). В этот слой входит программное обеспечение для ОС семейства Windows, клиенты на базе мобильных устройств, веб-сервисы.

Уровень аппаратных средств – непосредственно физические устройства для проведения обследований: электроэнцефалографы, кардиографы, системы биологической обратной связи, носимые медицинские устройства и т. д.

Являясь ядром платформы, *информационная система обработки данных* взаимодействует с прикладными приложениями по средствам REST API запросов. Авторизация пользователей осуществляется с помощью модуля авторизации, который получает шифрованный токен от внешнего сервиса авторизации и сравнивает его с токеном, полученным от пользователя. Это позволяет обеспечить дополнительную безопасность системы. Информационная система обработки данных в общем случае может состоять из параллельно работающих виртуальных узлов, обеспечивающих работу со своим пулом пользователей. Центральным элементом Информационной системы обработки данных является *менеджер управления*, который отрабатывает запросы пользователей и запросы интерфейсных модулей. *Менеджер управления* также обеспечивает подготовку данных к сохранению в глобальном хранилище данных, управляя конвертером форматов [68, 83, 85].

2.2 Информационная системы обработки данных

Для корректного функционирования информационной системы необходимо обеспечить работу модулей облачной инфраструктуры, которые позволят осуществлять сбор, хранение, систематизацию и защиту данных. Общая схема организации модулей платформы представлена на рисунке 2.2.

Многомодульная система обработки данных включает в себя как базы данных (пользователи, группы, сессии) и хранилища, так и различные вспомогательные системы. На этом уровне реализуется логика групп и ролей пользователей, а также обеспечивается взаимодействие с конечными клиентскими приложениями при помощи REST API запросов. Один из основных модулей – модуль защиты данных, который необходим для обеспечения безопасности данных. Основная задача – предотвратить несанкционированный доступ к хранилищу файлов с результатами медицинских исследований [81-82, 92, 93].

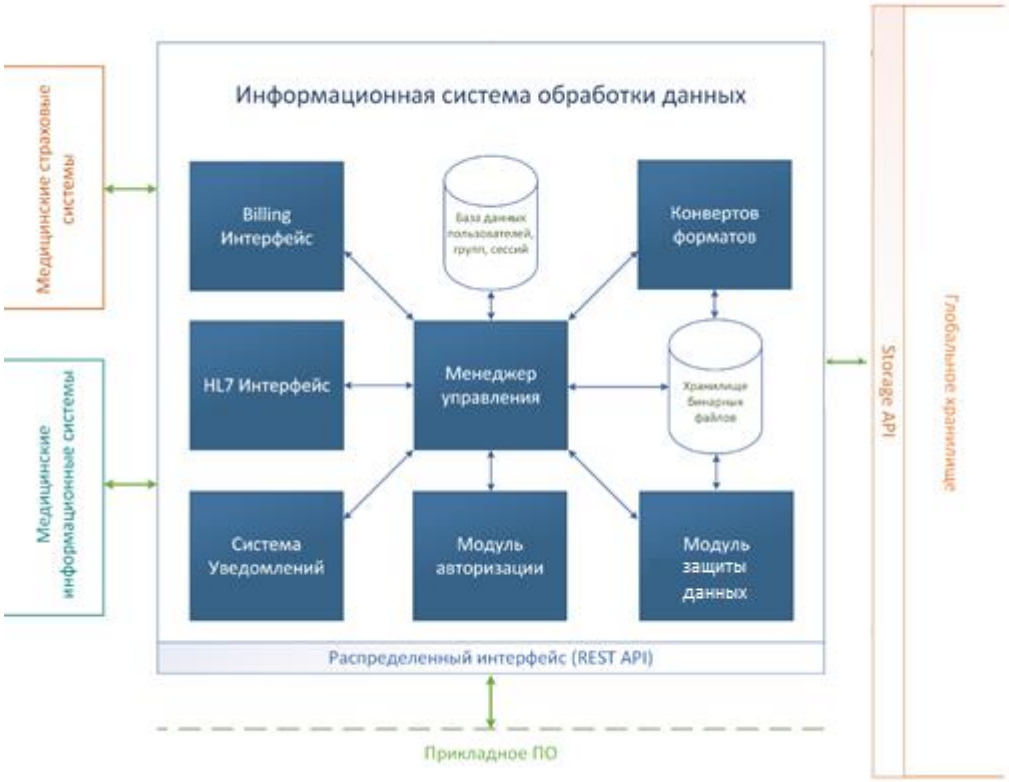


Рисунок 2.2 – Общая схема организации модулей платформы

Посредством менеджера управления происходит коммуникация с интерфейсом для взаимодействия с другими МИС и аппаратными средствами по протоколу HL7 (Health Level 7). Протокол представляет собой наивысший уровень медицинского документооборота и представляет собой стандартизированный свод правил, описывающий процедуры обмена, управления и интеграции цифровых данных в медицинских экосистемах. На 7 уровне выполняется поддержка и выполнение следующих операций:

1. Структурирование передаваемых данных;
2. Возможность проектирования систем;
3. Достижение согласованности передач;
4. Безопасность;
5. Идентификация участников;
6. Доступность информации.

Функциональность протокола HL7 представлена в таблице 2.1. в сравнении с другими современными медицинскими протоколами, используемыми в том числе для обеспечения безопасности данных пациентов.

Таблица 2.1 – Сравнение характеристик медицинских протоколов

Функционал	**HL-7**	**xDT**	**EdiFact**
Хранение и обмен изображениями	+	–	–
Графический диагноз	+	–	–
Архивирование	+	+	–
Изображения в документах	+	–	–
Видео в документах	+	–	–
Электронная история болезни	+	+	+
Преобразование данных	+	+	–
Информация о радиологии/КТ/МРТ	+	–	–
Регистрация пациентов	+	+	–

Виртуальная платформа системы классификации выделена в отдельную подсистему, в которой созданы виртуальные машины, реализующие различные алгоритмы обработки. Используя гипервизор и управляющую виртуальную машину, для формирования запросов в глобальное хранилище данных, можно создавать конфигурации классификаторов, как для различных исследовательских задач, так и для рутинных вычислений, позволяющих улучшить бизнес-процессы оказания медицинских услуг. В документе ГОСТ Р ИСО/HL7 27931–2015 не регламентируются вопросы аутентификации пользователей и конфиденциальности данных, которые передаются по каналам связи с использованием протокола. Подразумевается, что это всецело находится в компетенции приложения-отправителя и приложения получателя. Кроме того, до настоящего момента стандарт HL7 никак не специфицирует криптографические методы защиты, которые могут применяться при транспортировке сообщений от одной системы к другой. Подразумевается, что пользователи сами обеспечивают выполнение юридических и профессиональных требований, предъявляемых к обеспечению безопасности передаваемых данных [64].

2.3 Выводы

Предложена архитектура облачной платформы распределенного хранения, систематизации, обработки и защиты конфиденциальных данных (результатов медицинских обследований), позволяющая взаимодействовать с различными аппаратными средствами диагностики с целью формирования результирующих данных (итоговых файлов, содержащих информацию о проведенных обследованиях для пользователей системы). Отличительная особенность архитектуры заключается в том, что существует возможность интеграции средств защиты данных, циркулирующих внутри экосистемы, принимая во внимание модель угроз Долева-Яо, в которой одной из угроз является возможность злоумышленника быть авторизованным пользователем в сети. Такая угроза означает, что злоумышленник имеет право устанавливать соединение с любым другим пользователем, а как следствие получать информацию, содержащую конфиденциальные данные.

Разделение потоков данных на уровни, стандартизация протоколов передачи информации и предложенный метод защиты конфиденциальных данных, в совокупности позволяют разработать гибкую и надежную МИС, обладающую возможностью защиты данных пользователей. Архитектура позволяет выполнять процедуру интеграции с различными МИС и подключать используемые в медицинской практике аппаратные средства, а также индивидуальные мобильные устройства. Единое пространство для хранения данных предоставляет возможность осуществлять исследование большого массива классифицированной медицинской информации средствами машинного обучения.

Механизмы защиты, на основе предложенного метода предполагают снижение количества потенциальных угроз за счет разделения файла с конфиденциальными данными на определенное количество частей, что усложняет доступ злоумышленника при попытке компрометации, поскольку каждая часть файла будет храниться на отдельном сервере.

ГЛАВА 3. ЭКСПЕРИМЕНТАЛЬНЫЕ ИССЛЕДОВАНИЯ РАБОТЫ МЕТОДА ОБЕСПЕЧЕНИЯ БЕЗОПАСНОСТИ КОНФИДЕНЦИАЛЬНЫХ ДАННЫХ НА ОСНОВЕ СХЕМЫ ШАМИРА.

В ходе работы были проведены различные эксперименты, демонстрирующие зависимость времени разделения и восстановления секрета, а также объем используемой памяти в зависимости от параметров *N, K* и *M*. На основе полученных результатов была выбрана наиболее оптимальная схема разделения секрета, которая легла в основу предложенного метода защиты конфиденциальной информации.

Также на основании проведенного анализа и предварительных оценок общего количества рабочих станций, вовлеченных в работу с МИС, было определено, что на момент разработки архитектура сможет обеспечивать непрерывную работу 500 рабочих станций, которые образуют децентрализованную информационную систему. Стоит отметить, что для обеспечения корректной работы будет достаточно 60 серверов (из учета 10 рабочих станций на сервер, а также резервные мощности). Предполагается использовать в качестве серверов, устройства, размещенные на территории Российской Федерации, удовлетворяющие требованиям 152-ФЗ и предприняты меры защиты в соответствии с требуемым уровнем защищенности, согласно приказу ФСТЭК №21.

3.1 Состав медицинской информационной системы

Для подтверждения выполнения экспериментальных исследований метода по обеспечению безопасности медицинских данных было выполнено тестирование в экосистеме облачной медицинской информационной системы.

Серверная часть отвечает за бизнес-логику и работу с базой данных. Реализован набор REST API запросов для моделирования различных сценариев

взаимодействия пользователей и групп МИС. Для авторизации пользователей был использован сервис OAuth2.0. В качестве базы данных использовался инстанс MySQL на Amazon Relational Database Service (RDS). Вэб-интерфейс управления разработан с помощью фреймворка ASP.NET MVC Framework. Платформа может быть развернута на базе кластера инстансов от Amazon Elastic Compute Cloud (EC2).

Реализованная платформа поддерживает возможность интеграции с медицинскими информационными системами с помощью двухсторонней передачи данных по протоколу HL7. Для получения результатов, представляющих собой результаты медицинских исследований была произведена интеграция с программно-аппаратным комплексом для возможности регистрации ЭЭГ-сигналов, чтобы в последующем использовать этот сигнал в качестве файла для проведения экспериментальных исследований.

3.2 Обоснование выбора схемы разделения секрета перед проведением экспериментов

Перед экспериментами над параметрами были проанализированы сложность математического алгоритма и ресурсоемкость вычислений для четырех основных схем, выбранных в качестве наиболее подходящих для решения поставленной задачи. В таблице 3.1 приведено сравнение сложности алгоритма для этапов разделения на доли и восстановление долей секрета.

Таблица 3.1 – Анализ сложности математического алгоритма

	Шамира	**Асмута-Блума**	**Карнин-Грин-Хелмана**	**Блэкли**
Разделение секрета на доли	$O(N \times K)$	$O(N)$	$O(N)$	$O(K \times N)$
Восстановление долей секрета	$O(K^2)$	$O(K^2)$	$O(K^3)$	$O(K^3)$

Таблица 3.2 демонстрирует результаты проведенного анализа ресурсоемкости в процессе вычислений для файла, состоящего из 255 символов. Под ресурсоемкостью в данном случае понимается количество потребляемой оперативной памяти для процедур разделения секрета и восстановления обратно в единый файл. В качестве рабочей станции использовался персональный компьютер на ОС Ubuntu 20.04 с характеристиками: Intel Core i7-11800H, ядра: 8 x 2.3 ГГц, ОЗУ 16 ГБ, SSD 500 Гб.

Оперативная память является одним из наиболее критически важных составляющих для серверов, потому что любой процесс, запущенный на сервере, обращается к памяти и использует ее часть для выполнения операций. Во время работы серверов часто запущено множество фоновых процессов, обеспечивающих стабильность работы системы, поэтому большое потребление оперативной памяти может быть критичным для продолжения стабильной работы. Именно поэтому была проведена оценка ресурсоемкости процесса разделения и восстановления долей секрета для рассмотренных ниже схем с целью определения менее ресурсоемкой схемы для дальнейших исследований [1].

Таблица 3.2 – Анализ ресурсоемкости в процессе вычислений

	Шамира	**Асмута-Блума**	**Карнин–Грин Хелмана**	**Блэкли**
Разделение секрета на доли	*24 Кб*	*56 Кб*	*80 Кб*	*1 Мб*
Восстановление долей секрета	*112 Кб*	*319 Кб*	*82 Кб*	*1 Мб*

Дальнейшие эксперименты, которые упоминаются в диссертационном исследовании проведены для трех пороговых схем разделения секрета, а именно: схема Шамира, схема Асмута-Блума и схема Карнин-Грин-Хелмана. Схема Блэкли оказалась в десятки раз более ресурсоемкой по сравнению с другими и поэтому была исключена из дальнейших экспериментов.

3.3 Обоснование библиотеки для схем разделения секрета

Для того чтобы проводить эксперименты с СРС нужно использовать библиотеку языка программирования и определиться с требованиями, предъявляемыми к ней. В первую очередь нужно обеспечить возможность работы с арифметическими операциями, наличие функционала для работы с различными числами (задавать их размер и сравнивать), а также способность работать с простыми числами для того, чтобы проверить является ли заданное значение простым числом или нет. Такой функционал присущ всем схемам и позволяет задавать свободные коэффициенты и вычислять общий модуль. Библиотека также должна поддерживать работу с системами уравнений, чтобы реализовать схему Карнин-Григ-Хеллмана, которая построена на основе решения системы уравнений. Операции, производимые с математическим аппаратом, должны выполняться с высокой скоростью. Одним из полезных требований в контексте поддержки и быстрого решения возникающих вопросов является наличие обновляемого репозитория и активного процесса разработки, поскольку технологии развиваются стремительно и необходимо учитывать последние тенденции разработки алгоритмов при использовании программных компонент. Активное сообщество исследователей и разработчиков являются показателем качественной реализации инструмента.

Проведя анализ доступных математических библиотек для реализации СРС, были найдены различные варианты исполнения. Например, библиотека «sss», написанная на языке Python предоставляет API для разделения секретных данных на несколько частей и поддерживает множество существующих схем разделения секрета. Аналогичная библиотека под названием «Secret Sharing» позволяет работать только с схемой Шамира. Библиотека «mpvss-rs», написанная на языке Rust имеет поддерживаемый репозиторий, но из-за языка программирования не может быть универсальным решением. Одной из наиболее универсальных и одновременно полнофункциональных библиотек оказалась «Miracl». Реализация

библиотеки доступна на нескольких языках программирования, таких как C, C++, Go, Rust, Python, Java, Javascript и Swift. В первую очередь к достоинствам этой библиотеки относится репозиторий с открытым исходным кодом. Более того, репозиторий поддерживает в актуальном состоянии и обновляется. Доступ к функциям и структурам не требует никаких надстроек и может быть использован по прилагаемой в репозитории инструкции. В библиотеке хорошо реализован математический аппарат, включающий функционал работы с большими числами и простыми числами, поддержкой эллиптических кривых, арифметики в поле, а также немаловажные опции, например определение наибольшего общего делителя, наименьшего общего кратного, китайская теорема об остатках, работа с матрицами и системами линейных алгебраических уравнений, и уравнениями со степенями. Авторы заявляют, что в библиотеке заложен функционал заранее выбирать предел размеров чисел для наиболее эффективного выполнения предполагаемых операций. Это значит, что при условии изначальной оценки ресурсоемкости можно наиболее эффективно реализовывать различные математические алгоритмы и распределять ресурсы.

3.4 Эксперименты с использованием различных параметров

В ходе экспериментов, в первую очередь, исследуется влияние параметров N, K на время разделения и восстановления секрета для различных схем. Данные параметры являются основными в схемах разделения секрета. Так, например, параметр N – определяет возможное количество участников в пороговой схеме разделения (в нашем случае количество серверов). Параметр K используется для операции восстановления секрета и образует необходимое пороговое количество участников. Важно учесть, что условие $K \leq N$ всегда выполнимо [66].

Также было определено, что порог, при котором секрет может быть восстановлен, равен восьмидесяти процентам. Такое условие необходимо в случае технических неполадок некоторой части оборудования или если доступ к

некоторым серверам по каким-то причинам будет приостановлен, возможность получить доступ к части файла будет присутствовать. Иначе, при использовании формулы N = K для доступа к секрету, можно столкнуться с проблемами, которые не позволят раскрыть содержимое файла (например, если хотя бы 1 сервер выйдет из строя, тогда $N \neq K$).

В качестве файла (M) был использован обезличенный DICOM, полученный в результате компьютерной томографии легких, содержащий серию из 16 слайсов и дополнительную информацию в виде тегов. Размер файла составлял 1048576 байт (1024 кбайт). В качестве рабочей станции использовался персональный компьютер на ОС Ubuntu 20.04 с характеристиками: Intel Core i7-11800H, ядра: 8 x 2.3 ГГц, ОЗУ 16 ГБ, SSD 500 Гб [1]. Результаты представлены в таблице 3.3.

Таблица 3.3 – Исследование основных параметров схем разделения секрета при различных значениях N и K

	Параметры												
Схемы	Пар-ры схем	N=5	K=4	M=1024	N=10	K=8	M=1024	N=25	K=20	M=1024	N=50	K=40	M=1024
Шамира	Разделение (мс)	3,85			3,92			5			11,5		
	Восстановление (мс)	0,73			0,76			1,4			3,8		
Асмута-Блума	Разделение (мс)	79,31			116,75			634,8			870		
	Восстановление (мс)	0,82			1,2			2,4			13,1		
Карнин – Грин - Хелмана	Разделение (мс)	11,55			137,26			700			3648,96		
	Восстановление (мс)	0,7			70,7			408,58			2126,2		

Схема Шамира превосходит аналоги, основываясь на результатах замеров времени разделения и восстановления частей файла. Если для небольших значений N и K различия незначительны, то при увеличении этих параметров разница во времени, необходимом для выполнения операций, значительно возрастает (в десятки раз).

3.5 Эксперименты с различными размерами файла

Для того, чтобы удостовериться в эффективности схемы Шамира при использовании в качестве основного инструмента для разделения секрета в предлагаемом методе защиты информации, было принято решение исследовать зависимости параметров при работе с файлами различного размера. Так, для эксперимента были выбраны еще 3 файла в формате DICOM разного размера, с различным количеством слайсов и тегов. Результаты сравнения приведены в таблицах ниже (3.4, 3.5, 3.6) [1].

Таблица 3.4 – Исследование влияния размера файла на параметры схем (128 кб)

	Параметры												
Схемы	Пар-ры схем	N=5	K=4	M=128	N=10	K=8	M=128	N=25	K=20	M=128	N=50	K=40	M=128
Шамира	Разделение (мс)	1,16			1,18			1,51			3,48		
	Восстановление (мс)	0,49			0,51			0,6			1,12		
Асмута-Блума	Разделение (мс)	24			35,38			192,4			263,7		
	Восстановление (мс)	0,66			0,78			0,82			4		
Карнин – Грин - Хелмана	Разделение (мс)	3,55			41,6			212,12			1045,14		
	Восстановление (мс)	0,43			21,42			123,81			644,3		

Таблица 3.5 – Исследование влияния размера файла (4096 Кб) на параметры схем

	Параметры												
Схемы	Пар-ры схем	N=5	K=4	M=4096	N=10	K=8	M=4096	N=25	K=20	M=4096	N=50	K=40	M=4096
Шамира	Разделение (мс)	6,16			6,272			8			18,4		
	Восстановление (мс)	1,04			1,5			2,2			6,1		
Асмута-Блума	Разделение (мс)	127			186,8			1015,7			1392		
	Восстановление (мс)	1,1			1,9			3,8			20,9		
Карнин – Грин - Хелмана	Разделение (мс)	18,5			219,6			1120,8			5838,34		
	Восстановление (мс)	1			113,12			653,73			3402		

Таблица 3.6 – Исследование влияния размера файла (8192 Кб) на параметры схем

	Параметры												
Схемы	Пар-ры схем	N=5	K=4	M= 8192	N=10	K=8	M= 8192	N=25	K=20	M= 8192	N=50	K=40	M= 8192
Шамира	Разделение (мс)	7,32			7,5			9,2			21,9		
	Восстановление (мс)	1,26			1,72			2,76			7,2		
Асмута-Блума	Разделение (мс)	150,7			221,82			1206,11			1653,58		
	Восстановление (мс)	1,33			2,29			4,56			25		
Карнин – Грин - Хелмана	Разделение (мс)	21,95			260,81			1333,32			6933,1		
	Восстановление (мс)	1,18			134,32			776,3			4039,78		

По результатам экспериментов можно сделать вывод о том, что схема Шамира работает значительно быстрее для файлов различных размеров. Однако, стоит отметить, что файлы формата DICOM имеют сложную структуру и содержат информацию в разной интерпретации (изображения, текст, числа). Исходя из этого было принято решение сравнить рассмотренные схемы разделения секрета, используя файлы других форматов, но одинакового размера.

3.6 Эксперименты с различными типами файлов

Для проведения эксперимента были выбраны файлы различных типов, которые отличаются форматом интерпретации данных. Для данного эксперимента использовались файлы размером 1024 Кб следующих форматов:

- EDF (используется для представления бинарных данных) – результаты представлены в таблице 3.7.
- CSV (используется для представления табличных данных) – результаты представлены в таблице 3.8.
- BMP (используется для представления растровых файлов) – результаты представлены в таблице 3.9.

- TXT (используется для представления текстовых данных) – результаты представлены в таблице 3.10.
- SVG (используется для представления векторных файлов) – результаты представлены в таблице 3.11.

Таблица 3.7 – Исследование основных параметров схем разделения для файла в формате EDF

	Параметры											
Схемы	Пар-ры схем	N=5	K=4	M=1024	N=10	K=8	M=1024	N=25	K=20	M=1024	N=50	K=40	M=1024
Шамира	Разделение (мс)	3,29			3,36			4,27			9,81		
	Восстановление (мс)	Менее 1			Менее 1			1,17			3,25		
Асмута-Блума	Разделение (мс)	69,72			99,7			541,8			742,7		
	Восстановление (мс)	Менее 1			1,02			2,04			11,2		
Карнин – Грин - Хелмана	Разделение (мс)	9,86			117,21			598			3116,19		
	Восстановление (мс)	Менее 1			60,35			349			1815,8		

Таблица 3.8 – Исследование основных параметров схем разделения для файла в формате CSV

	Параметры											
Схемы	Пар-ры схем	N=5	K=4	M=1024	N=10	K=8	M=1024	N=25	K=20	M=1024	N=50	K=40	M=1024
Шамира	Разделение (мс)	3,12			3,21			4,09			9,44		
	Восстановление (мс)	Менее 1			Менее 1			1,04			3,11		
Асмута-Блума	Разделение (мс)	67,77			92,25			522,6			713,69		
	Восстановление (мс)	Менее 1			Менее 1			1,96			10,42		
Карнин – Грин - Хелмана	Разделение (мс)	9,17			112,1			571,22			3001,9		
	Восстановление (мс)	Менее 1			58,33			333,26			1786,47		

Таблица 3.9 – Исследование основных параметров схем разделения для файла в формате BMP

	Параметры

Схемы	Пар-ры схем	N=5	K=4	M=1024	N=10	K=8	M=1024	N=25	K=20	M=1024	N=50	K=40	M=1024
Шамира	Разделение (мс)	4,66			4,62			5,91			13,36		
	Восстановление (мс)	Менее 1			Менее 1			1,58			4,29		
Асмута-Блума	Разделение (мс)	95,23			128,42			742,55			1002,5		
	Восстановление (мс)	1,09			1,8			3,6			19,7		
Карнин – Грин - Хелмана	Разделение (мс)	13,6			164,82			812,63			4319,78		
	Восстановление (мс)	Менее 1			81,84			497,9			2555,11		

Таблица 3.10 – Исследование основных параметров схем разделения для файла в формате TXT

	Параметры												
Схемы	Пар-ры схем	N=5	K=4	M=1024	N=10	K=8	M=1024	N=25	K=20	M=1024	N=50	K=40	M=1024
Шамира	Разделение (мс)	2,96			2,95			3,86			8,99		
	Восстановление (мс)	Менее 1			Менее 1			Менее 1			2,93		
Асмута-Блума	Разделение (мс)	61,85			84,96			475,56			681,2		
	Восстановление (мс)	Менее 1			Менее 1			1,82			9,67		
Карнин – Грин - Хелмана	Разделение (мс)	8,43			104,9			548,7			2891,71		
	Восстановление (мс)	Менее 1			55,66			313,26			1696,9		

Таблица 3.11 – Исследование основных параметров схем разделения для файла в формате SVG

	Параметры												
Схемы	Пар-ры схем	N=5	K=4	M=1024	N=10	K=8	M=1024	N=25	K=20	M=1024	N=50	K=40	M=1024
Шамира	Разделение (мс)	4,73			4,83			6,15			14		
	Восстановление (мс)	Менее 1			Менее 1			1,74			4,59		
Асмута-Блума	Разделение (мс)	97,55			143,2			780,8			1058,11		
	Восстановление (мс)	1			1,51			2,96			15,85		
	Разделение (мс)	14,2			164,71			864,2			4414,08		

Карнин – Грин - Хелмана	Восстановление (мс)	Менее 1	86,96	498,34	2560,14

Исследования зависимости параметров разделения и восстановления секрета от типа файла демонстрируют превосходство схемы Шамира над другими реализациями. Этому свидетельствуют результаты экспериментов для файлов графических изображений. Если для текстовых и табличных файлов результаты не сильно отличаются, хотя схема Шамира является более эффективной и в этих случаях, то при работе с форматами BMP и SVG заметны различия во времени обработки в десятки и сотни раз.

Ниже, на рисунке 3.1 представлена визуализация зависимости времени разделения и восстановления секрета на основе схемы разделения секрета Шамира для параметров N=50, K=40 от размера файла.

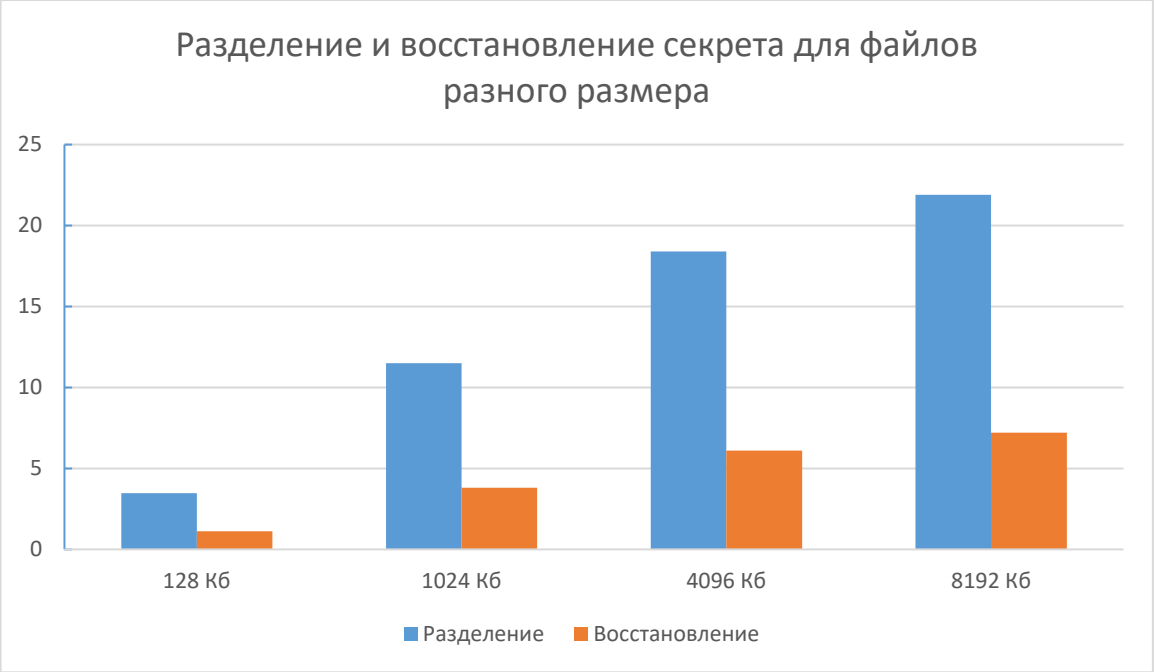


Рисунок 3.1 – Время выполнения (мс) основных операций в зависимости от размеров файла

Ниже, на рисунке 3.2 представлена визуализация зависимости времени разделения и восстановления секрета по схеме Шамира для параметров N=50, K=40 от типа файла.

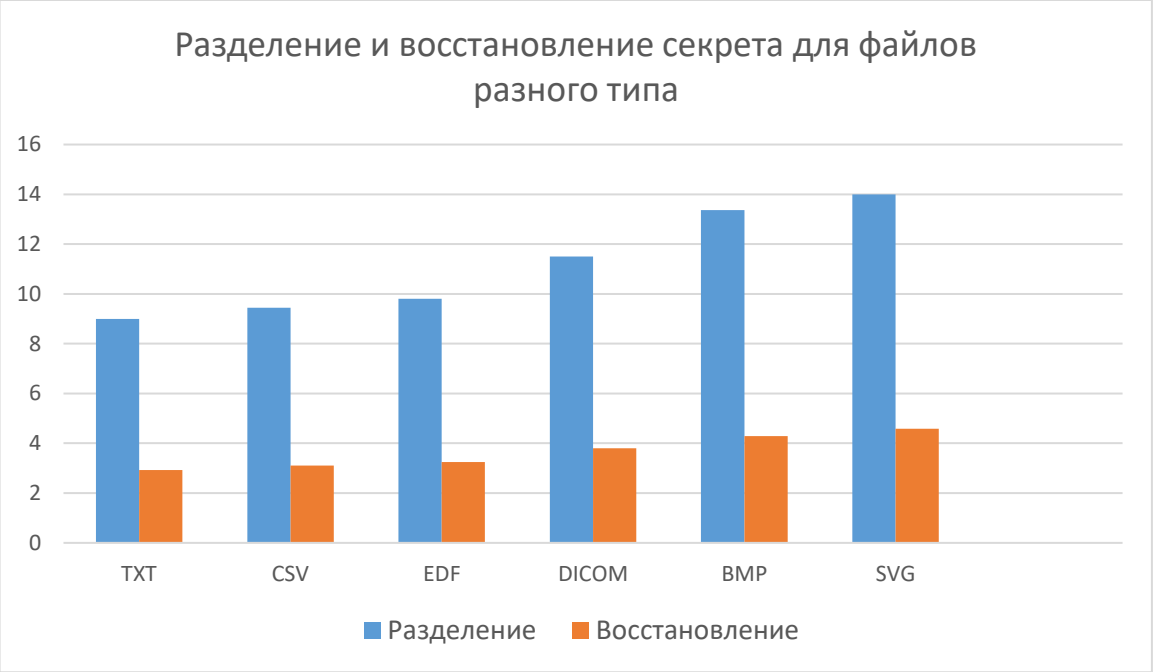


Рисунок 3.2 – Время выполнения (мс) основных операций в зависимости от типов файла

3.7 Анализ безопасности предлагаемого метода

После проведения нескольких серий экспериментов, в рамках которых была определена оптимальная с точки зрения параметров времени выполнения операций (разделения и сбора секрета) схема, возникла необходимость провести анализ эффективности предлагаемого метода, в рамках подсистемы защиты МИС при помощи различных средств анализа безопасности криптографических протоколов, чтобы подтвердить целесообразность применения выбранной схемы в качестве основы метода обеспечения безопасности. В качестве инструмента анализа, с точки зрения безопасности был использован анализаторов SPAN AVISPA (Automated Validation of Internet Security Protocols and Applications) tool.

Автоматизированные анализаторы криптографических протоколов позволяют выполнять верификацию свойств, обнаруживать потенциально возможные атаки на протоколы при условии невыполнения предписанных свойств, а также визуализировать этапы выполнения анализа и строить пошаговые диаграммы для визуальной репрезентации работы анализатора.

Целью AVISPA является анализ различных интернет-протоколов и приложений, чувствительных к безопасности. Эта технология позволяет ускорить разработку сетевых протоколов нового поколения, улучшает их безопасность и, следовательно, повышает общественное признание передовых распределенных приложений, реализованных с использованием данного анализатора.

Библиотека AVISPA представляет собой набор спецификаций протоколов безопасности, написанных на формализованном HLPSL (high level protocol specification language) языке командой разработчиков AVISPA, впоследствии автоматически проанализированных с помощью данного инструмента. Для каждого протокола существует описание его назначения, процесс обмена сообщениями в стиле Алисы и Боба, ожидаемые свойства безопасности и их классификация, а также обнаруженные атаки (если таковые имеются) и фактическая спецификация стандарта HLPSL. Учитывая факт того, что интерпретация на языке HLPSL является трудоемкой можно использовать более простой формализованный язык CAS+.

Оценка эффективности в рамках настоящего исследования проводилась для трех случаев с учетом особенностей модели угроз Долева-Яо:

1. анализ протокола на наличие уязвимостей до внедрения предлагаемого метода на основе схемы Шамира;
2. анализ протокола на наличие уязвимостей после внедрения предлагаемого метода с использованием схемы Шамира;
3. анализ протокола на наличие уязвимостей после внедрения предлагаемого метода, с использованием рассмотренных в экспериментах схем разделения секрета (Блэкли, Карин-Грин-Хеллмана).

В таблице 3.12 приведены сравнения в формализованном виде для состояния системы до и после применения предлагаемого метода защиты данных.

Поскольку оценить в количественном значении уровень защищенности системы, а также снижение угроз как до, так и после внедрения метода защиты не представляется возможным, было сделано следующее:

1. Подсчитано общее количество шагов от момента передачи сообщения (файла) между всеми сторонами (получателями) до конечного адресата.

2. Подсчитано количество unsafe ответов (небезопасных шагов) с точки зрения анализатора протоколов безопасности.

3. Найдено процентное отношение количества небезопасных шагов к общему количеству.

Процесс передачи конфиденциальной информации от одной роли к другой без использования предлагаемого метода состоит из 6 основных шагов (авторизация пользователей в системе, отправка файла на сервер, хранение файла, перемещение файла между сервером и рабочей станцией врача, работа с файлом и отправка файла конечному пользователю), а при интеграции метода на основе схемы Шамира количество шагов увеличивается до 14 (появляются этапы шифрования и расшифрования файла, а также разделение файла на фрагменты и восстановление исходного файла из фрагментов).

В результате моделирования получены следующие результаты. До внедрения метода на 4 из 6 шагов были обнаружены угрозы атаки. 2 угрозы определены на этапах отправки файла по каналу передачи от отправителя на сервер (в случае, когда злоумышленник является стороной, принимающей сообщение). Еще 2 угрозы представляют опасность в том случае, когда возникает ситуация, при которой злоумышленник может получить любое сообщение, передаваемое по сети. Исходя из описанного алгоритма формализованной оценки эффективности системы защиты, с учетом количества шагов, получено значение:

$$\left(1 - \frac{4}{6}\right) * 100\% = 33\%.$$

Аналогичный анализ системы после внедрения предлагаемого метода продемонстрировал 3 угрозы для случаев, когда злоумышленник может являться стороной, принимающей любые сообщения. Учитывая общее количество шагов получено следующее значение:

$$\left(1 - \frac{3}{14}\right) * 100\% = 78.5\%.$$

Принимая во внимание тот факт, что общее количество шагов увеличилось с 6 до 14, численная интерпретация полученных результатов демонстрирует снижение количества потенциальных угроз на 45%. Однако, если анализировать общее количество возможных угроз до и после внедрения предлагаемого метода защиты конфиденциальных данных, то их количество снизилось на 1. В таком случае защищенность системы к вероятным угрозам повысилась на 25%.

Однако, на текущем этапе в качестве схемы разделения секрета, лежащей в основе предлагаемого метода, была использована схема Шамира, потому что по результатам экспериментов в контексте ресурсоемкости она оказалась самой эффективной. Стоит отметить, что ресурсоемкость может быть достигнута в ущерб безопасности, поэтому были рассмотрены и другие схемы, которые использовались в предыдущих экспериментах.

Таблица 3.12 – сравнение эффективности до и после внедрения метода (в основе лежит схема Шамира)

Эффективность до внедрения метода	Эффективность после внедрения метода (схема Шамира)
33.3%	78,5%

Аналогичный эксперимент по анализу потенциальных уязвимостей был проведен для метода, в случае, когда в основе лежат схемы Асмута-Блума или Карин-Грин-Хеллмана. Для схемы Асмута-Блума, количество небезопасных шагов составило 6, а для схемы Карнин-Грин-Хеллмана – 5. Результаты представлены в таблице 3.13.

Таблица 3.13 – сравнение эффективности метода при использовании в качестве основы различных схем разделения секрета

Схема Шамира	Схема Асмута-Блума	Схема Карин-Грин-Хеллмана
78.5%	56%	67.9%

3.8 Выводы

В результате проведения экспериментальных исследований для четырех основных схем разделения секрета, были оставлены три основных кандидата для которых проводились дальнейшие эксперименты. Схема Блэкли была исключена из дальнейших экспериментов, потому что на первом этапе при исследовании параметров ресурсоемкости она оказалась не эффективной с точки зрения потребления оперативной памяти. На основе полученных практических результатов Схема Шамира зарекомендовала себя, как наиболее быстрая при анализе основных операций (разделения и восстановления частей файлов) для различных параметров тестируемых файлов, ее быстродействие превосходит аналоги в десятки раз.

Касательно анализа с точки зрения безопасности протоколов разделения секрета, было проведено моделирование с использованием анализатора безопасности протоколов. В результате экспериментов было выявлено, что схемы разделения секрета тоже отличаются показателями, хотя не так сильно в абсолютных величинах, как при экспериментах с ресурсоемкостью. Однако, протокол распределения секрета на основе схемы Шамира оказался наиболее стойким к возможным угрозам по сравнению со схемами Асмута-Блума и Карнин-Грин-Хеллмана.

Принимая во внимание модель угроз Долева-Яо, на основе которой проводилось моделирование было обнаружено, что предложенный метод обеспечения безопасности медицинских данных, в основе которого лежит схема разделения секрета Шамира позволяет снизить уровень угроз на 45% по сравнению с типовыми средствами обеспечения безопасности, интегрированным в МИС, включающие механизмы авторизации пользователей, посредством логина и пароля и передачей незашифрованного файла на сервер. Предлагаемый метод позволяет минимизировать риск компрометации файла для угрозы, когда злоумышленник

может получать любое сообщение, передаваемое по сети, потому что метод подразумевает передачу фрагментов файла, каждый из которых по отдельности не представляет никакой информации. В случае, если злоумышленник является стороной, принимающей сообщения, то он может получить восстановленный файл, однако он будет зашифрован на сеансовом ключе выработанным с одним из серверов, выбранным случайным образом и в таком случае злоумышленник не будет иметь возможности для дешифрования файла, поскольку он не устанавливал сеанс с сервером для генерации ключа.

ГЛАВА 4. РАЗРАБОТКА МЕТОДА ОБЕСПЕЧЕНИЯ ЗАЩИТЫ КОНФИДЕНЦИАЛЬНЫХ ДАННЫХ В ОБЛАЧНОЙ МЕДИЦИНСКОЙ ИНФОРМАЦИОННОЙ СИСТЕМЕ

4.1 Основа метода обеспечения безопасности медицинских данных

Анализируя подходы к обеспечению безопасности конфиденциальных данных в распределенных медицинских системах в рамках главы 1 настоящего исследования были обнаружены множественные факты, указывающие на слабую эффективность классических систем защиты в рамках современных облачных децентрализованных систем.

Предлагаемый метод, направленный на исключение обнаруженных недостатков, за счет применения протокола на основе схемы разделения секрета Шамира, ключевыми особенностями которого являются [66, 86]:

1. повышение уровня безопасности для данных, чувствительных к компрометации,
2. повышение уровня безопасности для систем, в которой злоумышленник может являться авторизованным пользователем,
3. скорость работы алгоритмов разделения и восстановления долей секрета,
4. масштабируемость, что является очень важным фактором при работе в рамках облачных распределенных информационных систем, большая часть из которых построена на основе микросервисных архитектур,
5. защита от случайной потери или уничтожения данных, в случае, когда один или несколько серверов выходят из строя (благодаря свойству избыточности данных, при котором задается пороговое значение долей секрета, сбор воедино которых позволит получить полный доступ к секрету).

Кроме того, предлагаемый подход может быть распараллелен на большое количество серверов (в случае наличия свободных мощностей), что позволит

сократить время выполнения операций разделения и слияния долей секрета, а как следствие привести к сокращению времени выполнения вычислений и ускорению работы подсистемы защиты.

Одно из значительных преимуществ метода заключается в то, что он может быть применим к многопользовательским и многоролевым системам, в которых медицинские данные попадают не напрямую от клиента на сервер, а проходят итерационно через определенное количество пользователей, каждый из которых может получать доступ к этим данным.

В общем виде реализация метода выглядит следующим образом:

1. Пользователь, обратившийся в медицинское учреждение, проходит процедуры, по результатам которых формируется некоторый файл с медицинскими данными.

2. Далее этот файл (обследование) отправляется на сервер. Для этого осуществляется шифрование, и процедура отправки на сервер выполняется в соответствии со следующим алгоритмом:

- Формируется общий ключ (одноразовый сеансовый) отправителя с одним из свободных серверов, выбранным **случайным образом**. Выработка общего ключа производится каждой из сторон (отправителя и сервера) по протоколу Диффи-Хеллмана.
- Затем, на локальном компьютере отправителя выполняется шифрование файла обследования с использованием общего ключа отправителя и случайно выбранного сервера, а затем передача уже зашифрованного файла на выбранный (на предыдущем шаге) случайным образом сервер.
- Случайный сервер, используя общий с отправителем ключ, осуществляет расшифрование файла. После чего расшифрованный файл разбивается на фрагменты равной длины.
- Случайный сервер зашифровывает (с использованием собственного секретного симметричным ключа) и оставляет для хранения у себя один из

фрагментов файла. Остальные фрагменты файла передаются оставшимся серверам в соответствии со следующим алгоритмом:

- устанавливается связь между сервером-отправителем (случайно выбранным на шаге 2) и каждым другим сервером облачной МИС;
- для каждого из серверов, с которыми устанавливается связь, осуществляется процедура верификации (проверка подлинности на предмет «свой-чужой») по протоколу взаимного рукопожатия,
- после успешной верификации каждого из серверов, с которыми устанавливается связь, выполняется генерация одноразового сеансового ключа между сервером-отправителем и всеми остальными серверами МИС. Выработка общего ключа производится аналогично каждой из сторон по протоколу Диффи-Хеллмана. Таким образом для каждого из оставшихся серверов формируется свой уникальный ключ общий сервером-отправителем.
- определенный фрагмент файла, предназначенный для конкретного сервера, шифруется сервером-отправителем на общем (с сервером-получателем) ключе и передается для хранения на сервер-получатель.

- Каждый из серверов, получив свой фрагмент файла, выполняет расшифрование, используя сеансовый ключ (общий с сервером-отправителем).

3. Теперь каждый из серверов-получателей зашифровывает расшифрованный (на предыдущем шаге) фрагмент файла своим секретным ключом для дальнейшего хранения. Таким образом, каждый сервер хранит свой фрагмент файла, зашифрованный на своем секретном ключе. Это означает, что исходный файл был разделен на определенное количество частей, каждая из которых была зашифрована и отправлена на случайный сервер, входящий в состав облачной МИС для хранения.

4. Расшифрование файла (скачивание с сервера) получателем выглядит следующим образом: получатель делает запрос к серверу (выбранному случайным образом) на получение данных (файла с обследованием).

5. Сервер, получив запрос, расшифровывает свой фрагмент файла обследования **своим** симметричным ключом. Для получения остальных фрагментов файла обследования сервер, выбранный случайным образом, устанавливает связь с оставшимися серверами в соответствии со следующим алгоритмом:

- для каждого из серверов, с которыми устанавливается связь, осуществляется процедура верификации (проверки подлинности на предмет «свой-чужой») по протоколу взаимного рукопожатия;
- после успешной верификации каждого из серверов, с которыми устанавливается связь, выполняется генерация одноразового сеансового ключа между случайным сервером (выбранным получателем) и всеми остальными серверами. Выработка общего ключа производится каждой из сторон по протоколу Диффи-Хеллмана. Таким образом для каждого из оставшихся серверов формируется свой уникальный, ключ общий со случайным сервером.
- каждый из серверов МИС, на котором хранится часть файла обследования обрабатываемого пациента, осуществляет расшифрование своего фрагмента файла обследования своим симметричным ключом;
- каждый фрагмент файла, хранящийся на серверах, шифруется одноразовым сеансовым ключом сервера (того, на котором хранится соответствующий фрагмент файла), выработанным совместно со случайно выбранным (получателем) сервером. Зашифрованные фрагменты файлов передаются на сервер, выбранный получателем.

6. Сервер, выбранный получателем, получив оставшиеся фрагменты файлов от остальных серверов, расшифровывает эти фрагменты с помощью

одноразовых сеансовых ключей, соответствующих определенным фрагментам файла (и серверам).

7. Затем сервер, выбранный получателем, осуществляет восстановление полного файла обследования с использованием фрагментов, расшифрованных на предыдущем шаге.

8. Этот же сервер, устанавливает связь с получателем для шифрования файла обследования с целью дальнейшей передачи получателю в соответствии со следующим алгоритмом:

- формирование общего ключа (одноразовый сеансовый ключ) получателя и сервера, который он выбрал. Выработка общего ключа выполняется каждой из сторон по протоколу Диффи-Хеллмана.
- шифрование на общем ключе (получателя и сервера) файла обследования и передача его получателю в зашифрованном виде.

9. Получатель, приняв зашифрованный файл обследования, расшифровывает его с использованием общего ключа (своего и сервера).

4.2 Описание работы метода обеспечения безопасности данных в медицинской информационной системе

Общая схема взаимодействия между участниками МИС

Последовательность действий состоит из нескольких этапов и описывает шаги метода, который применяется при обработке обследований пациента всеми участниками МИС (лаборантом, специалистом и доктором), их последующим разделением на части и шифрованием, расшифрованием.

Каждый из этапов предлагаемого метода определяет формат взаимодействия между ролями (например, пациент – доктор, пациент – лаборант, лаборант – специалист, специалист – доктор, доктор – пациент).

Ниже на рисунке 4.1 представлена общая схема взаимодействия между участниками МИС на всех этапах, а также процесс обработки медицинских обследований.

В рамках предложенного метода защиты используются следующие условные обозначения:

P – пациент;

Spec – специалист, который занимается формированием заключения по результатам обследования;

D – доктор (терапевт);

L – лаборант, который проводит обследования определенного профиля;

A – системный администратор МИС;

ID – уникальный идентификатор, присваиваемый каждому пользователю МИС;

F – файл, содержащий данные того или иного обследования;

S – сервер;

K_s – общий одноразовый сеансовый ключ (универсальное обозначение);

K_sL – общий одноразовый сеансовый ключ лаборанта и случайно выбранного сервера;

K_sSpec – общий одноразовый сеансовый ключ специалиста и случайно выбранного сервера;

K_sD – общий одноразовый сеансовый ключ доктора и случайно выбранного сервера;

S_R – сервер, выбранный случайным образом (универсальное обозначение);

K_{sec} – секретный симметричный ключ (универсальное обозначение);

F_R – случайный фрагмент файла F.

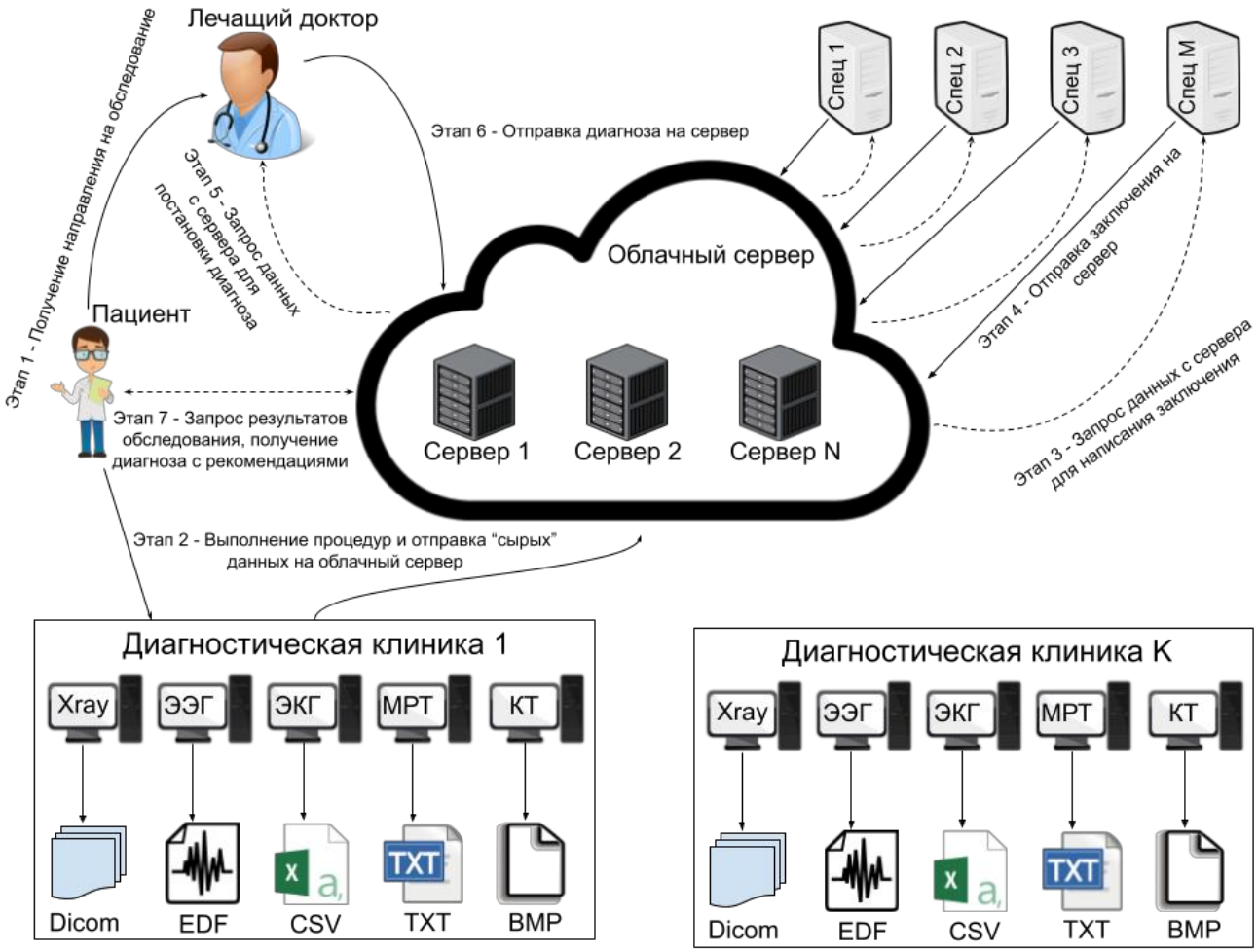


Рисунок 4.1 Общая схема процесса циркулирования данных в облачной МИС

4.3 Основные этапы метода обеспечения безопасности конфиденциальных данных в медицинской информационной системе

Этап 1 - Присвоение идентификатора пациенту.

Шаг 1. Пациент **P** обращается к лечащему доктору **D** (терапевту) за консультацией в случае появления жалоб.

$$P \rightarrow D$$

Шаг 2. При обращении в медицинское учреждение пациент регистрируется в системе электронной регистратуры и получает уникальный идентификатор **ID**. Идентификатор присваивается каждому пациенту один раз (при первом посещении любого медицинского учреждения, которое имеет доступ к МИС) и в дальнейшем

отображается в личном кабинете пациента. Таким образом, в МИС устанавливается соответствие между ФИО пациента и идентификатором.

$$ID_1 \rightarrow P_1$$

$$ID_2 \rightarrow P_2$$

...

$$ID_N \rightarrow P_N$$

Каждый доктор, работающий с облачной МИС, имеет базу данных соответствия ФИО пациента к присвоенному ID. Обращаться к базе данных может каждый из участников МИС (доктора, лаборанты, специалисты, системный администратор) в соответствии с правами доступа, определенными в матрице доступа.

Шаг 3. Доктор однозначно определяется и закрепляется за конкретным пациентом и работает с ним на протяжении всего курса обследования или лечения.

$$define: P_1 \rightarrow D_1,\ \ P_2 \rightarrow D_2, \dots, P_M \rightarrow D_M$$

Этап 2 - Шифрование результатов обследования и загрузка на сервер.

Данный этап описывает процесс прохождения медицинских процедур в клинике, на основании направления доктора. После посещения пациентом медицинского учреждения и выполнения назначенного обследования, для пациента формируется файл (отдельный для каждого обследования) с результатами в виде исходных данных (например, снимок компьютерной томографии), который должен быть отправлен на облачный сервер для последующего анализа специалистом.

Шаг 1. Пациент обращается в медицинское учреждение по направлению доктора для прохождения медицинских обследований, которые проводятся лаборантом **L**. Для каждого пациента может быть предусмотрено несколько видов, а также различное количество (различное количество для одного и того же вида) медицинских обследований. Все направления на прохождение медицинских обследований отображаются у пациента в личном кабинете. В случае

необходимости (например, при отсутствии финансовых возможностей) пациент имеет возможность пометить в личном кабинете какие-либо из направлений как «*отказ от прохождения*». Это будет означать, что конкретное обследование будет пропущено и доктору отправлено уведомление о том, что данное обследование не будет учтено для постановки итогового диагноза.

По умолчанию, обследование с заключением специалиста может быть отправлено доктору (для совокупного анализа и постановки итогового диагноза) только после того, как назначенная процедура будет выполнена пациентом.

В результате проведения каждого из назначенных обследований (в одной или нескольких клиниках) соответствующий лаборант **L** формирует отдельный файл **F** (DICOM, JSON, DOC, CSV, EDF, BMP и т.д.) с «сырыми» данными обследования (исходный физиологический сигнал, математические показатели, графики, снимки и др.), которые в дальнейшем требуют анализа со стороны специалиста **S** (написания заключения по конкретному обследованию).

Шаг 2. Отправка файла с обследованием на сервер.

Лаборант, сформировав по результатам медицинского обследования конкретный файл **F** (например, в формате DICOM), осуществляет его шифрование и отправку на сервер в соответствии со следующим алгоритмом:

- формируется общий ключ (одноразовый сеансовый) лаборанта K_sL с одним из серверов S_R, выбранным **случайным образом**. Выработка общего ключа производится каждой из сторон по протоколу Диффи-Хеллмана.

- локально выполняется шифрование файла обследования **F** (например, серия DICOM слайсов) на общем ключе лаборанта и случайно выбранного сервера, а затем передача этого файла в зашифрованном виде на выбранный случайным образом сервер S_R (на предыдущем шаге).

Шаг 3. Случайный сервер S_R, используя общий с лаборантом ключ, осуществляет расшифрование файла, полученного от лаборанта. После чего расшифрованный файл разбивается на фрагменты равной длины по схеме Шамира.

$$F = F_1 + F_2 + \cdots + F_N$$

Шаг 4. Сервер S_R (выбранный на шаге 2 случайным образом) зашифровывает своим секретным симметричным ключом $K_{sec}S_R$ и оставляет для хранения у себя один из фрагментов файла.

$$K_{sec}S_R \rightarrow F_R$$

Остальные фрагменты файла передаются всем оставшимся серверам в соответствии со следующим алгоритмом:

- устанавливается связь между сервером S_R, выбранным (на шаге 2) случайным образом, и каждым сервером облачной МИС;

$$S_R \rightarrow S_1$$
$$S_R \rightarrow S_2$$
$$\dots$$
$$S_R \rightarrow S_N$$

- для каждого из серверов, с которыми устанавливается связь, осуществляется процедура верификации (проверка подлинности на предмет «свой-чужой») по протоколу взаимного рукопожатия;

$$S_R: define\ x;\quad compute\ y = f(x)$$
$$S_R \rightarrow S_1:\quad S_R\ send\ x$$
$$S_1:\quad compute\ y' = f(x)$$
$$S_1 \rightarrow S_R:\quad S_1\ send\ y'$$
$$S_R:\quad check\ if\ y = y';\quad then\ S_R\ trusts\ S_1$$

- после успешной верификации каждого из серверов, с которыми устанавливается связь, выполняется генерация одноразового сеансового ключа между случайным сервером S_R (выбранным на шаге 2) и всеми остальными серверами МИС. Выработка общего ключа производится аналогично каждой из сторон по протоколу Диффи-Хеллмана. Таким образом для каждого из оставшихся серверов формируется свой уникальный ключ общий со случайным сервером S_R.

$$K_sS_1 \rightarrow S_1$$
$$K_sS_2 \rightarrow S_2$$

$$K_s S_N \rightarrow S_N$$

- определенный фрагмент файла **F**, предназначенный для конкретного сервера **S**, шифруется сервером S_R (выбранным случайным образом на шаге 2) на общем (с сервером S) ключе и передается для хранения на сервер S.

$$F_{1..n}\ for\ S_{1..n} cipher\ by\ K_s S_R \rightarrow S$$

Шаг 5. Каждый из серверов, получив свой фрагмент файла, выполняет расшифрование, используя сеансовый ключ (общий с сервером S_R, выбранным случайным образом на шаге 2).

Шаг 6. Теперь каждый из серверов зашифровывает расшифрованный (на шаге 5) фрагмент файла своим секретным ключом для дальнейшего хранения. Таким образом, каждый сервер хранит свой фрагмент файла, зашифрованный на своем секретном ключе. Это означает, что исходный файл (серия DICOM слайсов) был разделен на определенное количество частей на основе схемы Шамира, каждая из которых была зашифрована и отправлена на случайный сервер, входящий в состав облачной МИС для хранения.

Этап 3 - Расшифрование файла (скачивание с сервера) обследования для анализа и постановки заключения.

Данный этап предполагает расшифрование файла и последующее скачивание на компьютер специалиста для написания заключения на основе проведенного обследования.

Предусловие: в МИС предусмотрены специалисты различных медицинских профилей (ЭКГ, ЭЭГ, КТ, МРТ, ЭМГ, рентген и др.), которые осуществляют формирование медицинского заключения на основе анализа зарегистрированных лаборантами «сырых» данных. Для команды специалистов предусмотрен распределенный характер работы: каждый из специалистов имеет возможность получать файлы обследований случайных пациентов для анализа и написания заключения.

Распределение файлов между специалистами реализовано по типу **очереди**: первый файл, отправленный на сервер лаборантом, будет первым проанализирован специалистом (в рамках своего медицинского профиля).

Преимущества описанного подхода заключаются в том, что снижается время обслуживания каждого пациента, а также уменьшается субъективность при формировании заключения за счет обработки случайных обезличенных файлов обследований случайными специалистами.

Шаг 1. Специалист **Spec**, авторизовавшись в личном кабинете МИС, имеет возможность просмотра общего количества файлов обследований (**своего профиля**), ожидающих анализа. Также специалисту доступен определенный файл (который был раньше всех отправлен лаборантом на сервер) для анализа и формирования заключения.

Шаг 2. Для работы с файлом специалист делает запрос к серверу (выбранному случайным образом) на получение данных (файла с обследованием).

Шаг 3. Сервер, получив запрос, расшифровывает свой фрагмент файла обследования **своим** симметричным ключом. Для получения остальных фрагментов файла обследования сервер, выбранный случайным образом, устанавливает связь с оставшимися серверами в соответствии со следующим алгоритмом:

- для каждого из серверов, с которыми устанавливается связь, осуществляется процедура верификации (проверки подлинности на предмет «свой-чужой») по протоколу взаимного рукопожатия;

- после успешной верификации каждого из серверов, с которыми устанавливается связь, выполняется генерация одноразового сеансового ключа между случайным сервером (выбранным на шаге 3) и всеми остальными серверами МИС. Выработка общего ключа производится каждой из сторон по протоколу Диффи-Хеллмана. Таким образом для каждого из оставшихся серверов формируется свой уникальный ключ общий со случайным сервером.

- каждый из серверов МИС, на котором хранится файл обследования текущего обрабатываемого пациента, осуществляет расшифрование своего фрагмента файла обследования своим симметричным ключом;

- каждый фрагмент файла F, хранящийся на серверах, шифруется одноразовым сеансовым ключом сервера (того, на котором хранится соответствующий фрагмент файла), выработанным совместно со случайно выбранным (на шаге 3) сервером. Зашифрованные фрагменты файлов передаются на случайно выбранный (на шаге 3) сервер.

Шаг 4. Сервер, случайно выбранный на шаге 3, получив оставшиеся фрагменты файлов от остальных серверов, расшифровывает эти фрагменты с помощью одноразовых сеансовых ключей, соответствующих определенным фрагментам файла (и серверам).

Шаг 5. Затем сервер, случайно выбранный на шаге 3, осуществляет восстановление полного файла обследования, используя схему Шамира с использованием фрагментов, расшифрованных на шаге 4.

Шаг 6. Сервер, случайно выбранный на шаге 3, устанавливает связь со специалистом для шифрования файла обследования с целью дальнейшей передачи специалисту (для формирования заключения) в соответствии со следующим алгоритмом:

- сформировать общий ключ (одноразовый сеансовый ключ) специалиста и сервера, выбранным (на шаге 3) случайным образом. Выработка общего ключа производится каждой из сторон по протоколу Диффи-Хеллмана.

- зашифровать на общем ключе (специалиста и случайно выбранного сервера) файл обследования и передать его специалисту в зашифрованном виде.

Шаг 7. Специалист, получив зашифрованный файл обследования, расшифровывает его с использованием общего ключа (специалиста и случайно выбранного сервера).

Шаг 8. Специалист анализирует с медицинской точки зрения полученные данные и формирует на основании проведенного анализа заключение (описание),

указывая его в соответствующем поле файла обследования. После того, как соответствующее поле с заключением заполнено, файл с обследованием считается готовым к отправке на сервер для последующего изучения.

После выполнения всех шагов данного этапа файл с обследованием и заключением готов к шифрованию, передаче на сервер и разделению на N частей с последующим распределением на M серверов (N=M), где N – количество частей файла, для обеспечения достаточного уровня безопасности.

Этап 4 - Зашифрование файла обследования (после написания заключения) и загрузка обратно на сервер.

Основная цель на данном этапе заключается в отправке на сервер файла с обследованием и заключением (по конкретному медицинскому обследованию) в зашифрованном виде.

Шаг 1. Отправка медицинского обследования и заключения (единый файл) на сервер.

Специалист после постановки заключения для конкретного файла с обследованием **F**, осуществляет шифрование данного файла и отправку на сервер в соответствии со следующим алгоритмом:

- формируется общий ключ (одноразовый сеансовый ключ) специалиста K_sSpec с одним из серверов S_R, выбранных **случайным образом**. Выработка общего ключа производится каждой из сторон по протоколу Диффи-Хеллмана.

- локально выполняется шифрование файла обследования **F** на общем ключе (специалиста и случайно выбранного сервера) и передача этого файла на выбранный сервер S_R в зашифрованном виде.

Шаг 2. Случайный сервер S_R, используя общий (со специалистом) ключ, осуществляет расшифрование файла, полученного от специалиста. Расшифрованный файл разбивается на фрагменты равной длины.

$$F = F_1 + F_2 + \cdots + F_N$$

Шаг 3. Сервер S_R (выбранный на шаге 2 случайным образом) зашифровывает своим секретным симметричным ключом ($K_{sec}S_R$) и оставляет для хранения у себя один из фрагментов файла.

$$K_{sec}S_R \rightarrow F_R$$

Остальные фрагменты файла передаются всем оставшимся серверам в соответствии со следующим алгоритмом:

- устанавливается связь между сервером, выбранным (на шаге 2) случайным образом, и каждым сервером облачной МИС;

- для каждого из серверов, с которыми устанавливается связь, осуществляется процедура верификации (проверки подлинности на предмет «свой-чужой») по протоколу взаимного рукопожатия;

- после успешной верификации каждого из серверов, с которыми устанавливается связь, выполняется генерация одноразового сеансового ключа между случайным сервером (выбранным на шаге 2) и всеми остальными серверами МИС. Выработка общего ключа производится каждой из сторон по протоколу Диффи-Хеллмана. Таким образом для каждого из оставшихся серверов формируется свой уникальный ключ общий со случайным сервером.

- определенный фрагмент файла F, предназначенный для конкретного сервера S, шифруется сервером S_R (выбранным случайным образом на шаге 2) на общем (с сервером S) ключе и передается для хранения на сервер S.

$$F_{1..n}\ for\ S_{1..n} cipher\ by\ K_s S_R \rightarrow S$$

Шаг 4. Каждый из серверов, получив свой фрагмент файла, расшифровывает его, используя сеансовый ключ (общий с сервером S_R, выбранным случайным образом на шаге 2).

Шаг 5. Каждый из серверов зашифровывает расшифрованный (на шаге 4) фрагмент файла своим секретным ключом для дальнейшего хранения. Таким образом, каждый сервер хранит свой фрагмент файла, зашифрованный на своем секретном ключе.

После выполнения всех шагов данного этапа файл с обследованием и заключением специалиста представляет собой набор зашифрованных файлов равного размера, которые случайным образом распределены между определенным количеством серверов.

Этап 5 - Расшифрование файла (скачивание с сервера) обследования с заключением для постановки диагноза.

На данном этапе выполняется скачивание и расшифрование файла обследования с уже написанным заключением для того, чтобы установить диагноз доктором (терапевтом).

Шаг 1. Доктор **D**, авторизовавшись в личном кабинете МИС, имеет возможность просмотра общего количества пациентов, ожидающих постановки диагноза, а также общего количество файлов обследований (для каждого из **своих пациентов**) с заключениями от специалистов. Также доктору доступна обобщенная структура файлов всех обследований, сгруппированная по ID пациента для анализа и формирования заключения. Эта структура будет доступна по следующему принципу: пациент, все обследования которого были обработаны раньше всех, первым попадет к доктору для постановки диагноза. Сервер позволяет выполнять агрегирование всех обследований в единую структуру (таблица, файл) для конкретного пациента используя его идентификатор для удобства обработки доктором всех обследований (например, пациент выполнил несколько процедур и указал в личном кабинете, что все пройдено). Таким образом доктору доступна для анализа единая структура, содержащая обследования и заключения по всем направлениям, которые были рекомендованы пациенту для прохождения в клиниках, а не отдельные файлы, которые анализировали специалисты.

Для удобства, обобщенная структура называется файлом.

Шаг 2. Для работы с файлом доктор делает запрос к серверу (выбранному случайным образом) на получение данных (файла с обследованиями и заключениями).

Шаг 3. Сервер, получив запрос, расшифровывает **свой** фрагмент файла с обследованиями и заключениями **своим** симметричным ключом. Для получения остальных фрагментов файла сервер, выбранный случайным образом, устанавливает связь с оставшимися серверами в соответствии со следующим алгоритмом:

- для каждого из серверов, с которыми устанавливается связь, осуществляется процедура верификации (проверки подлинности на предмет «свой-чужой») по протоколу взаимного рукопожатия;

- после успешной верификации каждого из серверов, с которыми устанавливается связь, выполняется генерация одноразового сеансового ключа между случайным сервером (выбранным на шаге 2) и всеми остальными серверами МИС. Выработка общего ключа производится каждой из сторон по протоколу Диффи-Хеллмана. Таким образом для каждого из оставшихся серверов формируется свой уникальный ключ общий со случайным сервером.

- каждый из серверов МИС, на котором хранится файл с обследованиями и заключениями текущего обрабатываемого пациента, осуществляет расшифрование своего фрагмента файла обследования своим симметричным ключом;

- каждый фрагмент файла F, хранящийся на серверах, шифруется одноразовым сеансовым ключом сервера (того, на котором хранится соответствующий фрагмент файла), выработанным совместно со случайно выбранным (на шаге 2) сервером. Зашифрованные фрагменты файлов передаются на случайно выбранный (на шаге 2) сервер.

Шаг 4. Сервер, случайно выбранный на шаге 2, получив оставшиеся фрагменты файлов от остальных серверов, расшифровывает эти фрагменты с помощью одноразовых сеансовых ключей, соответствующих определенным фрагментам файла (и серверам).

Шаг 5. Затем сервер, случайно выбранный на шаге 2, осуществляет восстановление полного файла обследования с использованием фрагментов, расшифрованных на шаге 4.

Шаг 6. Сервер, случайно выбранный на шаге 2, устанавливает связь с доктором для шифрования файла (все выполненные обследования с заключениями) с целью дальнейшей передачи доктору для постановки диагноза в соответствии со следующим алгоритмом:

- сформировать общий ключ (одноразовый сеансовый ключ) доктора и сервера, выбранным (на шаге 2) случайным образом. Выработка общего ключа производится каждой из сторон по протоколу Диффи-Хеллмана.

- зашифровать на общем ключе (доктора и случайно выбранного сервера) файл и передать его доктору в зашифрованном виде.

Шаг 7. Доктор, получив зашифрованный файл обследования, расшифровывает его с использованием общего ключа (доктора и случайно выбранного сервера).

Шаг 8. Доктор анализирует сырые данные и заключения, поставленные специалистами по всем обследованиям (которые были пройдены пациентом). Затем доктор выполняет постановку диагноза на основании проведенного анализа обследований и заключений и указывает этот диагноз в соответствующем поле файла обследования. После того, как соответствующее поле с диагнозом заполнено, файл с обследованием считается готовым к отправке на сервер.

Стоит обратить внимание, что файл без диагноза (с пустым полем) не может быть отправлен на сервер, потому что в дальнейшем диагноз должен быть получен пациентом.

После выполнения всех шагов данного этапа файл с диагнозом готов к шифрованию, передаче на сервер и разделению на N частей с последующим распределением на M серверов (N=M).

Этап 6 - Зашифрование файла с диагнозом и рекомендациями по лечению, отправка данных на сервер

Основная цель на данном этапе заключается в отправке на сервер файла с диагнозом и рекомендациями по лечению (по конкретному пациенту) в

зашифрованном виде. После завершения этого этапа у пациента появится возможность ознакомится с диагнозом в личном кабинете МИС.

Шаг 1. Отправка файла с обследованием на сервер.

Доктор после постановки заключения для конкретного пациента **P** (с файлом обследования **F**), осуществляет шифрование данного файла и отправку на сервер в соответствии со следующим алгоритмом:

- формируется общий ключ (одноразовый сеансовый ключ) доктора K_sD с одним из серверов S_R, выбранных **случайным образом**. Выработка общего ключа производится каждой из сторон по протоколу Диффи-Хеллмана.

- локально выполняется шифрование файла обследования **F** на общем ключе (доктора и случайно выбранного сервера) и передача этого файла на выбранный сервер S_R в зашифрованном виде.

Шаг 2. Случайный сервер S_R, используя общий (с доктором) ключ, осуществляет расшифрование файла, полученного от доктора. Расшифрованный файл разбивается на фрагменты равной длины.

$$F = F_1 + F_2 + \cdots + F_N$$

Шаг 3. Сервер S_R (выбранный на шаге 2 случайным образом) зашифровывает своим секретным симметричным ключом ($K_{sec}S_R$) и оставляет для хранения у себя один из фрагментов файла.

$$K_{sec}S_R \rightarrow F_R$$

Остальные фрагменты файла передаются всем оставшимся серверам в соответствии со следующим алгоритмом:

- устанавливается связь между сервером, выбранным (на шаге 2) случайным образом, и каждым сервером облачной МИС;

- для каждого из серверов, с которыми устанавливается связь, осуществляется процедура верификации (проверки подлинности на предмет «свой-чужой») по протоколу взаимного рукопожатия;

- после успешной верификации каждого из серверов, с которыми устанавливается связь, выполняется генерация одноразового сеансового ключа

между случайным сервером (выбранным на шаге 2) и всеми остальными серверами МИС. Выработка общего ключа производится каждой из сторон по протоколу Диффи-Хеллмана. Таким образом для каждого из оставшихся серверов формируется свой уникальный ключ общий со случайным сервером.

- определенный фрагмент файла F, предназначенный для конкретного сервера S, шифруется сервером S_R (выбранным случайным образом на шаге 2) на общем (с сервером S) ключе и передается для хранения на сервер S.

Шаг 4. Каждый из серверов, получив свой фрагмент файла, расшифровывает его, используя сеансовый ключ (общий с сервером S_R, выбранным случайным образом на шаге 2).

Шаг 5. Каждый из серверов зашифровывает расшифрованный (на шаге 4) фрагмент файла своим секретным ключом для дальнейшего хранения. Таким образом, каждый сервер хранит свой фрагмент файла, зашифрованный на своем секретном ключе.

После выполнения всех шагов данного этапа файл с диагнозом, поставленным доктором, представляет собой набор зашифрованных файлов равного размера, которые случайным образом распределены между определенным количеством серверов.

Этап 7 - Получение данных с сервера пациентом и расшифрование файла с диагнозом и рекомендациями по лечению.

На данном этапе пациент получает файл с диагнозом и рекомендациями от доктора посредством обращения в личный кабинет МИС.

Шаг 1. Пациент **P**, авторизовавшись в личном кабинете МИС, имеет возможность просмотра своего диагноза.

Шаг 2. Для работы с файлом пациент делает запрос к серверу (выбранному случайным образом) на получение данных (файла с диагнозом).

Шаг 3. Сервер, получив запрос, расшифровывает свой фрагмент файла обследования **своим** симметричным ключом. Для получения остальных фрагментов файла обследования сервер, выбранный случайным образом,

устанавливает связь с оставшимися серверами в соответствии со следующим алгоритмом:

- для каждого из серверов, с которыми устанавливается связь, осуществляется процедура верификации (проверки подлинности на предмет «свой-чужой») по протоколу взаимного рукопожатия;

- после успешной верификации каждого из серверов, с которыми устанавливается связь, выполняется генерация одноразового сеансового ключа между случайным сервером (выбранным на шаге 3) и всеми остальными серверами МИС. Выработка общего ключа производится каждой из сторон по протоколу Диффи-Хеллмана. Таким образом для каждого из оставшихся серверов формируется свой уникальный ключ общий со случайным сервером.

- каждый из серверов МИС, на котором хранится файл обследования текущего обрабатываемого пациента, осуществляет расшифрование своего фрагмента файла обследования своим симметричным ключом;

- каждый фрагмент файла F, хранящийся на серверах, шифруется одноразовым сеансовым ключом сервера (того, на котором хранится соответствующий фрагмент файла), выработанным совместно со случайно выбранным (на шаге 3) сервером. Зашифрованные фрагменты файлов передаются на случайно выбранный (на шаге 3) сервер.

Шаг 4. Сервер, случайно выбранный на шаге 3, получив оставшиеся фрагменты файлов от остальных серверов, расшифровывает эти фрагменты с помощью одноразовых сеансовых ключей, соответствующих определенным фрагментам файла (и серверам).

Шаг 5. Затем сервер, случайно выбранный на шаге 3, осуществляет восстановление полного файла обследования с использованием фрагментов, расшифрованных на шаге 4.

Шаг 6. Сервер, случайно выбранный на шаге 3, устанавливает связь с пациентом для шифрования файла обследования с целью дальнейшей передачи

пациенту для получения файла с диагнозом в соответствии со следующим алгоритмом:

- сформировать общий ключ (одноразовый сеансовый ключ) пациента и сервера, выбранным (на шаге 3) случайным образом. Выработка общего ключа производится каждой из сторон по протоколу Диффи-Хеллмана.

- зашифровать на общем ключе (пациента и случайно выбранного сервера) файл и передать его пациенту в зашифрованном виде.

Шаг 7. Пациент, получив зашифрованный файл, расшифровывает его с использованием общего ключа (пациента и случайно выбранного сервера).

Шаг 8. Пациент получает возможность ознакомиться с диагнозом и рекомендациями от лечащего врача. После ознакомления с диагнозом пациент устанавливает в соответствующем поле личного кабинета галочку о получении информации (диагнозе и других данных), что информирует доктора об окончании работы с конкретным пациентом. Также у пациента есть возможность дополнительных консультаций после получения диагноза и рекомендаций со стороны врача (галочка в поле «нужна консультация). После выполнения ряда консультаций пациент должен проинформировать доктора об окончании работы.

Таким образом, после выполнения всех шагов данного этапа пациент имеет полный набор файлов обследований, заключений, диагноза и рекомендаций по лечению от лечащего врача, а доктор имеет представление о том, завершена ли работа с пациентом или ему требуется консультация.

ГЛАВА 5. ИСПОЛЬЗОВАНИЕ ТЕХНОЛОГИИ ПАРАЛЛЕЛЬНЫХ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛЕНИЙ MPI

5.1 Обоснование необходимости параллельных вычислений

В работе были исследованы способы распараллеливания основных операций, выполняемых по схеме разделения секрета Шамира с целью сокращения времени на разделение секрета (конфиденциального файла) и на объединение частей секрета в единый общий файл.

Масштабирование (параллельные вычисления) в анализе высокопроизводительных программ делится на две категории: «сильное» и «слабое» [69].

- сильное масштабирование описывает, как изменяется производительность по мере увеличения количества потоков при **фиксированном** размере задачи. То есть происходит увеличение числа потоков процессора, а размер файла остается неизменным.
- слабое масштабирование описывает, как изменяется производительность по мере увеличения количества потоков и размера задачи. В таком случае вместе с увеличением количества числа потоков увеличивается и размер файла или количество выполняемых операций.

По определению, алгоритм совместного использования секрета по схеме Шамира требует наличия двух параметров:

- необходимого количества общих ресурсов (**n**)
- порогового значения, для доступа к секрету (**t**).

Алгоритм вычисляет доли, генерируя случайное полиномиальное уравнение степени *t-1*. Секрет становится постоянной величиной в полиномиальном уравнении. При последовательном вычислении основная сложность заключается в том, что для каждого символа входных данных итерация вычисления должна выполняться *n* раз. Поскольку вычисление выполняется последовательно, то для

ускорения может быть представлена возможность реализовать параллельные вычисления [69–71].

Реализация алгоритма разделения секрета Шамира позволила исследовать преимущества распараллеливания алгоритма. Цель заключалась в том, чтобы ускорить процесс разделения и слияния долей секрета для файлов с учетом большого количества взаимодействующих сторон при этом без потерь в контексте безопасности.

5.2 Инструменты для распараллеливания кода

Для решения поставленной задачи был проведен анализ инструментов для распараллеливания программ, среди которых номинальными кандидатами оказались OpenMP и MPI [72–77].

OpenMP — представляет собой способ разработки на устройствах с общей оперативной памятью. Таким образом, параллельные вычисления выполняются в том случае, когда каждый параллельный поток имеет доступ ко всем имеющимся данным [76].

MPI — (Message Passing Interface) библиотека для разработки и ускорения программного кода на устройствах с распределенной памятью. В случае использования MPI параллелизм реализовывается, если отдельный процесс работает **в своем собственном пространстве памяти** изолированно от других. Параллелизм происходит потому, что каждому процессу абсолютно точно указывается, над какой частью конкретной задачи он должен работать, основываясь исключительно на идентификаторе процесса [78–79]. Одним из достоинств программ, разработанных с использованием библиотеки MPI, является возможность их использования как на специально оборудованном кластере, так и на кластере, состоящем из обычных локальных машин, связанных между собой сетью [80]. Для корректной работы программ, использующих для вычислений

кластер, состоящий из персональных компьютеров, связанных между собой сетью, необходимо выполнить следующие условия:

- на каждой рабочей станции должна быть установлена одна и та же операционная система и одинаковые версии программ для запуска параллельных процессов MPI;
- необходимо выполнять запуск программ на каждой машине от имени администратора.

В модели MPI параллельно выполняющиеся процессы имеют разные адресные пространства. Это может быть как один компьютер, имеющий несколько процессоров, так и несколько компьютеров, связанных между собой сетью. С точки зрения MPI, параллельная программа представляет собой набор обычных последовательных программ, которые выполняют некоторые вычислительные задачи одновременно. Обычно каждая из таких последовательных программ выполняется на своем процессоре и имеет доступ к своей локальной памяти [80].

Для того, чтобы обеспечить корректную согласованную работу частей параллельной программы необходим механизм межпроцессного обмена данными. В модели MPI пересылка управляющих сигналов и данных осуществляется с помощью сообщений. При этом обмен происходит не через общую или разделяемую память, а через другие коммуникационные среды, поэтому данная модель ориентирована на вычислительные системы с распределенной памятью. В рамках модели пересылка сообщений (отправка и прием) реализуются с помощью вызова соответствующих функций из библиотеки MPI.

Программы, использующие при работе библиотеку MPI, имеют так называемую SMPD-структуру (Single Program Multiple Data – одна программа, много данных). Обычно при запуске программ с применением технологии MPI инициализируется фиксированное число процессов, которое задается пользователем при запуске программы. По сути, каждый из процессов выполняет копию одной и той же программы. Каждый из экземпляров программы может определить собственный идентификатор, и как следствие, предпринять различные

действия. Экземпляры программы взаимодействуют, вызывая функции библиотеки MPI, которые поддерживают взаимодействие процессов с другими процессами, группами и окружением.

5.3 Экспериментальная часть

Для решения задачи в рамках распределенной медицинской системы наиболее оптимальным, с точки зрения эффективного потребления ресурсов вариантом, является способ распараллеливания на основе библиотеки MPI. Для проверки гипотезы были проведены экспериментальные расчеты. Все эксперименты проводились на локальной рабочей станции: процессор Intel Core i7 – 11800H (8-ядер по 2,3 ГГц с гиперпоточностью), 16 ГБ ОЗУ [69] с использованием количества файла, разделяемого на 255 частей. В качестве тестовых наборов данных использовались файлы, содержащие 500, 1000, 2000, 4000 и 8000 символов в случайных полях.

Эксперименты «слабого масштабирования» заключались в удвоении количества символов во входном файле при одновременном удвоении количества потоков. Были определены 2 функции в реализации, которые позволили существенно сократить время вычисления долей [69]. Для проверки работы параллельных вычислений использовалась библиотека Miracl и mpi4py.

```
from mpi4py import MPI
from MIRACL import big, mirsys, epoint, ebrick
import random

# Initialize MPI
comm = MPI.COMM_WORLD
rank = comm.Get_rank()
size = comm.Get_size()
```

В первую очередь, удалось распараллелить процесс генерации случайных коэффициентов, используемых в полиномиальной функции.

```
def generate_coefficients(k, secret):
    coefficients = [big(secret)]
```

```
    for _ in range(1, k):
        coefficients.append(big(random.randint(0, p.toInt())))
    return coefficients
```

Во-вторых, удалось распараллелить генерацию долей ключей. Оставив функцию слияния долей нетронутой, реализована проверка корректности процесса распараллеливания, поскольку функция могла повторно собрать ресурсы в исходный файл.

```
def generate_shares(k, n, secret):
    shares = []
    coefficients = generate_coefficients(k, secret)
    for i in range(1, n + 1):
        x = big(i)
        share = (x, evaluate_polynomial(coefficients, x))
        shares.append(share)
    return shares
```

После реализации параллелизма на этапе генерации долей схемы разделения секрета Шамира и проверки корректности этого преобразования принято решение реализовать параллелизм и в функции слияния долей.

```
def reconstruct_secret(shares):
    result = big(0)
    for i in range(len(shares)):
        num = big(1)
        den = big(1)
        for j in range(len(shares)):
            if i != j:
                num *= -shares[j][0]
                den *= shares[i][0] - shares[j][0]
        result += shares[i][1] * (num / den)
    return result % p
```

Проблема с распараллеливанием этапа соединения частей секрета в процессе совместного использования секрета Шамиром заключается в правильном определении области действия переменных MPI (т. е. выборе переменных, которые должны быть видны всем потокам), а также в определении и выделении критических областей. Наиболее важная область — это место, где каждый поток обновляет секрет после вычисления интерполяционных полиномов Лагранжа [93,

92]. Для этого пришлось синхронизировать доступ к такой области с помощью функционала библиотеки MPI, что позволило распараллелить цикл, вычисляющий интерполирующий полином Лагранжа функции, используемой при соединении долей обратно воедино.

```
chunk_size = n // size
    start = rank * chunk_size
    end = start + chunk_size if rank < size - 1 else n
```

Рассмотренный способ продемонстрировал значительное ускорение при разделении и объединение долей секрета в схеме Шамира с помощью библиотеки MPI. Ниже приведены результаты.

В таблице 5.1 показано, что для «сильного масштабирования» время создания общих ключей уменьшается почти **вдвое** каждый раз, когда мы удваиваем количество потоков. Такая реализация обеспечивает близкое к линейному ускорение [69].

Таблица 5.1 – Время необходимое для генерации 255 долей секрета

Количество потоков	1000 – символьный файл	2000 - символьный файл
1	8.42 мс	17.53 мс
2	4.44 мс	9.02 мс
4	2.35 мс	5.23 мс
8	1.3 мс	2.98 мс
16	0.7 мс	1.58 мс

Были получены аналогичные результаты масштабирования при объединении общих ресурсов для восстановления исходного секрета, как показано в таблице 5.2. Здесь важно отметить, что происходит увеличение времени при 16 потоках при объединении долей. Это соответствует количеству гиперпотоков ядер тестовой машины (восемь ядер с шестнадцатью гиперпотоками). На этом этапе увеличение

количества потоков становится контрпродуктивным, поскольку они не могут выполняться параллельно на таком оборудовании [69].

Таблица 5.2 – Время необходимое на восстановление 255 долей секрета

Количество потоков	1000 – символьный файл	2000 - символьный файл
1	1.48 мс	3.02 мс
2	0.75 мс	1.6 мс
4	0.4 мс	0.93 мс
8	0.29 мс	0.64 мс
16	0.33 мс	0.69 мс

В таблице 5.3 продемонстрированы результаты для «слабого масштабирования»: время не увеличивается пропорционально, поскольку были удвоены как размер входных данных, так и количество потоков [69].

Таблица 5.3 – Изменение времени при генерации долей секрета с одновременным удвоением количества символов и потоков

Количество потоков	Количество символов	Время
1	500	4.53 мс
2	1000	4.65 мс
4	2000	5.17 мс
8	4000	5.66 мс
16	8000	7.13 мс

5.4 Выводы

При выполнении экспериментальных исследований выполнено распараллеливание функций генерации случайных коэффициентов, разделения секрета на фрагменты и восстановление фрагментов воедино для протокола разделения секрета на основе схемы Шамира и получены практические результаты. Подход на основе использования библиотеки MPI продемонстрировал сокращение времени для генерации долей секрета в 1,87 раза, для задачи восстановления долей в 1,74 раза – при работе в многопоточном режиме. Подход демонстрирует ускорение на локальной машине (Intel Core i7 – 11800H: 8-ядер по 2,3 ГГц с гиперпоточностью до 16, 16 ГБ ОЗУ).

Можно сделать предположение, что в распределенной системе, состоящей из N рабочих станций, удастся добиться большего ускорения для рассмотренных операций, поскольку будет использоваться большое количество вычислительных ресурсов – количество ядер процессоров на распределенных серверах, а как следствие возможность использовать больше потоков. Также стоит отметить, что библиотека MPI имеет необходимый функционал для распараллеливания облачных в контексте облачных вычислений.

ЗАКЛЮЧЕНИЕ

В результате диссертационного исследования было предложено решение актуальной научной задачи и достигнута поставленная цель, заключающаяся в повышении эффективности обеспечения безопасности медицинских информационных систем за счет разработки и внедрения метода обеспечения защиты конфиденциальных данных пациентов на основе метода с использованием протокола разделения секрета, реализованного по схеме Шамира. В ходе работы были получены следующие результаты:

1. Разработан метод защиты медицинских данных на основе протокола разделения секрета, реализованного по схеме Шамира, который применяется в рамках облачной информационной системы для обеспечения безопасности файлов обследований. Метод может быть применим к различным форматам файлов медицинских обследований: DICOM, Nrrd, Nifti.

2. Использование разработанного метода защиты медицинских данных позволяет уменьшить количество угроз в медицинской информационной системе (основываясь на модели угроз Долева-Яо) на величину 45%.

3. Разработана архитектура облачной медицинской распределенной системы, обладающая наличием средств защиты конфиденциальных данных. Разработанный метод защиты медицинских данных на основе протокола разделения секрета, реализованного по схеме Шамира может быть интегрирован с большинством существующих программно-аппаратных комплексов и средств диагностики посредством использования протокола health level 7.

4. Разработано программное решение с применением библиотеки MPI для моделирования распараллеливания операций разделения и слияния долей секрета по схеме Шамира, чтобы оценить время разделения файла на фрагменты и слияния фрагментов воедино. Результаты экспериментов продемонстрировали, что при параллельной реализации схемы Шамира время для разделения файла на 255

частей сократилось в 1,87 раза, а для восстановления долей в 1,74 раза при работе в многопоточном режиме.

5. Получено свидетельство о регистрации программы для ЭВМ из Роспатента.

6. Результаты проведённых диссертационных исследований могут найти применение:

- при разработке и проектировании распределенных медицинских информационных систем, систем обеспечения безопасного хранения персональных данных;

- при обеспечении защиты информации, в распределенной медицинской системе.

СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

МИС – медицинская информационная система

КТ – компьютерная томография

ЭЭГ – электроэнцефаллография

ЭМГ – электромиография

DICOM - digital imaging and communications in medicine

NIFTI - neuroimaging informatics technology initiative

EDF – european data format

CSV – comma-separated values

BMP – bitmap picture

NRRD – nearly raw raster data

ЕГИСЗ – единая государственная информационная система в сфере здравоохранения

HL7 – health level 7

РИС – распределенная информационная система

БД – база данных

IDL – interface definition language

FDDI – fiber distributed data interface

AES – advanced encryption standard

DCC – distributed credit chain

LSB – least significant bit

OTP – one-time password

СРС – схема разделения секрета

MPI - message passing interface

SPMD –single program multiple data

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Шумилин, А. С. Метод обеспечения защиты персональных данных в медицинской облачной системе / А. С. Шумилин // Вопросы кибербезопасности. – 2023. – № 4(56). – С. 53-64. – DOI 10.21681/2311-3456-2023-4-53-64.

2. Российская Федерация. Законы. О персональных данных : Федеральный закон № 152-ФЗ [принят Государственной думой 8 июля 2006 года : одобрен Советом Федерации 14 июля 2006 года]. – Москва. – Текст : непосредственный.

3. Российская Федерация. Законы. Об основах охраны здоровья граждан в Российской Федерации: Федеральный закон № 323-ФЗ [принят Государственной думой 1 ноября 2011 года : одобрен Советом Федерации 9 ноября 2011 года]. – Москва. – Текст : непосредственный.

4. Российская Федерация. Законы. О безопасности критической информационной инфраструктуры Российской Федерации: Федеральный закон № 187-ФЗ [принят Государственной думой 12 июля 2017 года : одобрен Советом Федерации 19 июля 2017 года]. – Москва. – Текст : непосредственный.

5. Бабенко, Л. К. Алгоритм обеспечения защиты конфиденциальных данных облачной медицинской информационной системы / Л. К. Бабенко, Д. М. Алексеев, А.С. Шумилин // Известия ЮФУ. Технические науки. – 2021. – № 5(222). – С. 120-134. – DOI 10.18522/2311-3103-2021-5-120-134.

6. ЕМИАС упрощает жизнь пациентам / [Электронный ресурс] // MOS.ru : [сайт]. — URL: https://www.mos.ru/news/item/105828073/ (дата обращения: 03.06.2023).

7. Эксперимент центра диагностики и телемедицины / [Электронный ресурс] // Центр диагностики и телемедицины : [сайт]. — URL: https://mosmed.ai/ai/ (дата обращения: 31.05.2023).

8. Утечка данных в России / [Электронный ресурс] // TADVISER : [сайт]. — URL: https://www.tadviser.ru/index.php/Статья:Утечки_данных_в_России (дата

обращения: 31.05.2023).

9. Статистика Роскомнадзора по утечкам данных россиян / [Электронный ресурс] // TADVISER : [сайт]. — URL: https://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%A3%D1%82%D0%B5%D1%87%D0%BA%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85_%D0%B2_%D0%A0%D0%BE%D1%81%D1%81%D0%B8%D0%B8#.D0.A0.D0.BE.D1.81.D0.BA.D0.BE.D0.BC.D0.BD.D0.B0.D0.B4.D0.B7.D0.BE.D1.80_.D0.B7.D0.B0.D1.84.D0.B8.D0.BA.D1.81.D0.B8.D1.80.D0.BE.D0.B2.D0.B0.D0.BB_.D1.83.D1.82.D0.B5.D1.87.D0.BA.D0.B8_600_.D0.BC.D0.BB.D0.BD_.D0.B7.D0.B0.D0.BF.D0.B8.D1.81.D0.B5.D0.B9_.D0.BE_.D1.80.D0.BE.D1.81.D1.81.D0.B8.D1.8F.D0.BD.D0.B0.D1.85 (дата обращения: 31.05.2023).

10. Утечки данных в медицинских учреждениях / [Электронный ресурс] // ZDRAV.EXPERT Медтех-портал : [сайт]. — URL: https://zdrav.expert/index.php/Статья:Утечки_данных_в_медицинских_учреждениях (дата обращения: 02.06.2023).

11. Население России / [Электронный ресурс] // Wikipedia : [сайт]. — URL: https://ru.wikipedia.org/wiki/Население_России (дата обращения: 13.06.2023).

12. Утечка данных Shields Healts Care Group / [Электронный ресурс] // Xakep : [сайт]. — URL: https://xakep.ru/2022/06/08/shields-leak/ (дата обращения: 28.05.2023).

13. Утечка данных Managed Care of North America / [Электронный ресурс] // ZDRAV.EXPERT Медтех-портал : [сайт]. — URL: https://zdrav.expert/index.php/Компания:MCNA_Dental (дата обращения: 03.06.2023).

14. Особенности сбора медицинских данных для создания систем ИИ / [Электронный ресурс] // Webiomed : [сайт]. — URL: https://webiomed.ru/blog/osobennosti-sbora-meditsinskikh-dannykh-dlia-sozdaniia-sistem-iskusstvennogo-intellekta-s-tochki-zreniia-zakonodatelstva-o-zashchite-personalnykh-dannykh (дата обращения: 04.06.2023).

15. Защита данных в облачной МИС N3.Health / [Электронный ресурс] // N3.Health - официальный оператор передачи данных в ЕГСИЗ : [сайт]. — URL: https://n3health.ru/bezopasnost (дата обращения: 04.06.2023).

16. Особенности современных информационных распределённых систем / [Электронный ресурс] // Studfiles : [сайт]. — URL: https://studfile.net/preview/17161478/ (дата обращения: 18.07.2023).

17. Распределенные системы : учебное пособие для студентов, обучающихся по направлению подготовки 38.03.05 Бизнес-информатика / [авт.-сост. А.В. Демина, О.Н. Алексенцева]. – Саратов : Саратовский социально-экономический институт (филиал) РЭУ им. Г.В. Плеханова, 2018. – 108 с.

18. Jathanna, R International Journal of Engineering Research and Application / Jathanna, R [Электронный ресурс] // Ijera : [сайт]. — URL: https://ijera.com (дата обращения: 21.02.2022). – Vol. 7. – Issue 6. – (Part - 5). –2017. – P. 31-38.

19. Selvaraj, J. Cryptographic Encryption and Optimization for Internet of Things Based Medical Image Security / Jeeva Selvaraj, Wen-Cheng Lai, Balasubramanian Prabhu Kavin, Kavitha C. and Gan Hong Seng // Electronics 2023. – Vol. 12, Art №1636. – 2023.

20. Shobana Pritha, P. Healthcare information system using cloud security / P. Shobana Pritha, Dr. A. Sasi Kumar // International Journal of Engineering & Technology. – Vol. 7 (2.33). – 2018.

21. Tebaa, M. Secure Cloud Computing through Homomorphic Encryption / M. Tebaa, S. EL Hajii // International Journal of Advancements in Computing Technology. – Volume 5. – Art. №16. – 2019.

22. Awadallah, R. Homomorphic Encryption for Cloud Computing and Its Challenges / R. Awadallah, A. Samsudin // IEEE 7th International Conference on Engineering Technologies and Applied Sciences (ICETAS). Vol 7. – 2020.

23. Izang, A. Overview of Cloud Computing and Recent Addendum / A. A. Izang, Y. A. Mensah, O. J. Omotosho, and C. P. Obioma // Journal of Communications Technology, Electronics and Computer Science. – Vol. 5. – 2016. – P. 26-32.

24. Muhammad, K. A survey on top security threats in cloud computing / K. Muhammad, Y. Shao // International Journal of Advanced Computer Science and Applications (IJACSA). – Vol. 6. – Art № 3. – 2015. – P.109-113.

25. Acar, A. A survey on homomorphic encryption schemes: Theory and implementation / A. Acar, H. Aksu, A. S. Uluagac, M. Conti // ACM Computing Surveys (CSUR). – Vol. 51. – № 4. – 2018. – P. 1-35.

26. Liu, X. Efficient and Privacy-Preserving Outsourced Calculation of Rational Numbers / X. Liu, K. Choo, R. Deng, R. Lu, and J. Weng // IEEE Trans. Dependable Secur. Comput. – Vol. 15. – Art № 1. – 2018. – P. 27–39.

27. Patel, R. Review on Data Security on Cloud using Homomorphic Encryption over Big Data / R. Patel // International Research Journal of Engineering and Technology (IRJET). – Vol. 4. – Art № 4. – 2017. – P. 1807-1809.

28. Котяшичев, И. А. Защита информации в «Облачных технологиях» как предмет национальной безопасности / И. А. Котяшичев, Е. А. Бырылова // Молодой ученый. — 2015. — № 6.4 (86.4). — С. 30-34.

29. Керейтова, М. Р. Информационная безопасность в медицинских информационных системах / М. Р. Керейтова, В. Н. Малыш // НиКа. – 2012.

30. Бойченко И. В. Построение ИТ-инфраструктуры здравоохранения на основе парадигмы облачных вычислений / И. В. Бойченко // Врач и информационные технологии № 3. – 2011.

31. Кривошеева, Д. Модель угроз безопасности в системах дистанционного мониторинга состояния человека / Д. Кривошеева // Правовая информатика № 3. – 2016. – С 22-28.

32. Sambyal, S. Hybrid security model for securing healthcare data on cloud / S. Sambyal // 15th International Conference on Computer Modelling and Simulation. – Vol. 4. – Art № 6. – 2023.

33. Sharma, Y. A security model for the enhancement of data privacy in cloud computing /Y. Sharma, H. Gupta, S.K. Khatri // 2019 Amity International Conference on Artificial Intelligence (AICAI). – 2019. – P.898–902. – DOI:

10.1109/AICAI.2019.8701398.

34. Kartit, A. New Approach Based on Homomorphic Encryption to Secure Medical Images in Cloud Computing / A. Kartit // Trends Sci. – Vol. 19. – Art № 9. – 2022. – P. 3970. – DOI: 10.48048/tis.2022.3970

35. Marwan, M. A cloud-based solution for collaborative and secure sharing of medical data / M. Marwan, A. Kartit, H. Ouahmane // Int. J. Enterprise Information System. – Vol. 14.– 2018. – P.128.

36. Farokhi, F. Secure and private cloud-based control using semihomomorphic encryption / F. Farokhi, I. Shames, N. Batterham // IFAC-PapersOnLine. – Vol. 46. – 2016. – P.163-168.

37. Yang, P. An efficient secret key homomorphic encryption used in image processing service / P. Yang, X. Gui, J. An, F. Tian // Secure Communication Network. – 2018.

38. Chinnasamy, A. Design of secure storage for health-care cloud using hybrid cryptography / A. Chinnasamy, P. Deepalakshmi // 2018 second international conference on inventive communication and computational technologies (ICICCT). – 2018. – P. 1717–1720. –DOI: 10.1109/ICICCT.2018.8473107.

39. Siju John, S.N. Kumar Role of medical image encryption algorithms in cloud platform for teleradiology applications, Advances in Cyber Security and Intelligent Analytics, Edition: 1, Chapter: 7, Publisher: CRC Press, DOI:10.1201/9781003269144-7

40. Alsalmany, Abdulrahman Cloud system for encryption and authentication medical images, IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661, p-ISSN: 2278-8727, Volume 20, Issue 1, Ver. II (Jan.- Feb. 2018), pp. 65-75, DOI: 10.9790/0661-2001026575.

41. Vengadapurvaja, A. M. An Efficient Homomorphic Medical Image Encryption Algorithm For Cloud Storage Security / A. M. Vengadapurvaja, G. Nisha, R. Sasikaladevi // Procedia Computer Science. – Vol. 115. – 2017. – P.643–650.

42. Sumathi, S. Chaotic Map Based Encryption Algorithm for Secured Medical Data Analytics / S. Sumathi, S. Gopika, S. Nivedha, R. Kalaimathi // Data Intelligence

and Cognitive Informatics. – 2024. – P.59-68. – DOI: 10.1007/978-981-99-7962-2_5.

43. Blesswin, AJ. An improved gray scale visual secret sharing scheme for visual information security / AJ. Blesswin, P. Visalakshi // Fifth international conference on advanced computing (ICoAC). – 2013. – P.560–564. – https://doi.org/10.1109/ICoAC.2013.6922012.

44. Batra, M. Secure file storage in cloud computing using hybrid encryption algorithm / M. Batra, P. Dixit, L. Rawat, R. Khalkar // International Journal of Computer Engineering and Application. – Vol. 3. – 2018.

45. Biswas, C. An efficient algorithm for confidentiality, integrity and authentication using hybrid cryptography and steganography / C. Biswas, U. D. Gupta, M. M. Haque, // 2019 international conference on electrical, computer and communication engineering (ECCE). – Vol. 4. – 2019. – P.1–5. – DOI: 10.1109/ECACE.2019.8679136.

46. Hegui, Z. Two-dimensional sine-tentbased hyper chaotic map and its application in image encryption / Z. Hegui, P. Baoming, Zhu. Zhiliang // Chinese Computer Systems. – Vol. 7. – 2019. – P.1510–1518. – DOI:1000-1220 07-1510-09.

47. Viswanath, G. Hybrid encryption framework for securing big data storage in multi-cloud environment / G. Viswanath, P. V. Krishna // Evolutionary Intelligence. – Vol.14(2). – 2021. – P.691–698. – DOI: https://doi.org/10.1007/s12065-020-00404-w.

48. Ahsan, Md. A Secure medical record sharing scheme based on blockchain and two-fold encryption // Sci. – Vol. 4. – Art № 6. – 2023. – P.335–339.

49. Kumar, V. Complex entropy-based encryption and decryption technique for securing medical images / V. and P. Kumar, V. and B. Neelendra, P. Mishra, R. amd G., Dr Sachin // Multimedia Tools and Applications. – 2022. – DOI:10.1007/s11042-022-13546-z.

50. Soman, V. An enhanced hybrid data security algorithm for cloud / V. K. Soman, V. Natarajan // International 125 Conference Networks Adv. Computer Technologies NetACT. – 2017. – P.416–419. – DOI:10.1109/NETACT.2017.8076807.

51. Когелов, Д. Н. Пороговая схема протокола Диффи - Хеллмана / Д. Н.

Когелов, Ю. Р. Халниязова [Электронный ресурс] // ПДМ. Приложение : [сайт]. — URL: https://cyberleninka.ru/article/n/porogovaya-shema-protokola-diffi-hellmana-79 (дата обращения: 03.09.2022).

52. Одноразовые пароли / [Электронный ресурс] // Академик, словари и энциклопедии : [сайт]. — URL: https://dic.academic.ru/dic.nsf/ruwiki/666057 (дата обращения: 12.07.2022).

53. Арзиева, Ж. Т. Анализ методов генерации одноразовых паролей и высокая степень случайности генерируемых паролей / Ж. Т. Арзиева, А. Т. Арзиев // Бюллетень науки и практики. – 2022. – Т. 8, № 7. – С. 382-388. – DOI 10.33619/2414-2948/80/35.

54. Бабаш, А. В. Криптографические и теоретико-автоматные аспекты современной защиты информации / А. В. Бабаш // Криптографические методы защиты. – Москва. – 2008.

55. Разделение секрета / [Электронный ресурс] // Википедия, свободная энциклопедия : [сайт]. — URL: https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D1%8C_%D0%A4%D0%B5%D0%B9%D1%81%D1%82%D0%B5%D0%BB%D1%8F (дата обращения: 12.07.2022).

56. Схема разделения секрета Шамира / [Электронный ресурс] // Хабр журнал : [сайт]. — URL: https://habr.com/ru/post/431392/ (дата обращения: 16.07.2024).

57. Cryptographic algorithms – Shamir Secret Sharing [Электронный ресурс]: - Режим доступа: https://cryptography.fandom.com/wiki/Shamir%27s_Secret_Sharing (дата обращения 16.07.2022).

58. Носиров, З. Анализ криптографических схем разделения секрета для резервного хранения ключевой информации / З. А. Носиров, О. В. Щербинина // Прикаспийский журнал: управление и высокие технологии – №2 (46) – 2019. – С.62-66.

59. Схема Блэкли [Электронный ресурс]: Википедия, свободная

энциклопедия. – Режим доступа: https://ru.wikipedia.org/wiki/Схема_Блэкли (дата обращения: 16.07.2022).

60. Лавров, Д. Н. Принципы построения протокола гарантированной доставки сообщений / Лавров, Д. Н. [Электронный ресурс] // МСиМ : [сайт]. — URL: https://cyberleninka.ru/article/n/printsipy-postroeniya-protokola-garantirovannoy-dostavki-soobscheniy (дата обращения: 03.09.2022).

61. Blakley, G. R. Safeguarding cryptographic keys / G. R. Blakley // Proc. Of AFIPS Nasiona1 Computer Conference. – Vol. 48. – 1979. – P.313-317.

62. Бабенко, Л. К. Алгоритм обеспечения безопасности конфиденциальных данных медицинской информационной системы хранения и обработки результатов обследований / Л. К. Бабенко, А. С. Шумилин, Д. М. Алексеев // Известия ЮФУ. Технические науки. – 2020. – № 5(215). – С. 6-16. – DOI 10.18522/2311-3103-2020-5-6-16.

63. Алексеев, Д. М. Разработка и описание структуры и функционала облачной платформы хранения, систематизации и обработки медицинских данных: интеграция системы автоматического поиска участков эпилептической активности / Д. М. Алексеев, А. Н. Минюк, З. А. Понимаш, А. С. Шумилин // Системы управления и информационные технологии – №3(77) – 2019. – С. 52-55.

64. ГОСТР НСО/HL7 27931-2015 Информатизация здоровья // Health Level Seven Version 2.5. Прикладной протокол электронного обмена данными в организациях здравоохранения.

65. Алексеев, Д. М. Обеспечение защиты конфиденциальной информации в медицинской облачной системе с использованием пороговой гомоморфной криптосистемы с открытым ключом / Д. М. Алексеев, А. С. Шумилин // Вестник современных исследований. – 2021. – № 5-9(43). – С. 4-8.

66. Шумилин, А. С. Метод обеспечения безопасности конфиденциальной информации в распределенной медицинской облачной системе / А. С. Шумилин // Проблемы передачи информации в инфокоммуникационных системах: Сборник докладов и тезисов XIII Всероссийской научно-практической конференции,

Волгоград, 26 мая 2023 года. – Волгоград: Волгоградский государственный университет, 2023. – С. 82-86.

67. Ансамбль классификаторов: реализация, оценка эффективности и интеграция в облачную платформу хранения, систематизации и обработки медицинских данных / Д. М. Алексеев, А. Н. Минюк, З. А. Понимаш, А. С. Шумилин // Современные наукоемкие технологии. – 2019. – № 9. – С. 20-25.

68. Babenko, L. Development of the algorithm to ensure the protection of confidential data in cloud medical information system / L. Babenko, A. Shumilin, D. Alekseev // Proceedings – 2021 14th International Conference on Security of Information and Networks. – 2021. – P. 1-4. – DOI 10.1109/SIN54109.2021.9699356.

69. Бабенко, Л. К. Использование параллельных вычислений для реализации метода обеспечения безопасности на основе схемы Шамира в медицинской информационной системе / Л. К. Бабенко, А. С. Шумилин // Известия ЮФУ. Технические науки. – 2023. – №4(234). – С. 6-13. – DOI 10.18522/2311-3103-2023-4-6-13

70. Basavegowda, R. Electronic Medical Report Security Using Visual Secret Sharing Scheme / R. Basavegowda, S. Seenapp // UKSim 15th International Conference on Computer Modelling and Simulation. – 2013.

71. Bystrov, O. Cost and Performance Analysis of MPI-Based SaaS on the Private Cloud Infrastructure / O. Bystrov, A. Kačeniauskas, R. Pacevic // Parallel Processing and Applied Mathematics. – 2023. – P.171-182. – DOI:0.1007/978-3-031-30442-2_13.

72. Титов, К. Е. Исследование поведения фреймворка Openmp в программах с параллельными вычислениями / К. Е. Титов // E-Scio – №4 (55) – 2021.

73. Мартыненко, С. И. Параллельное решение краевых задач с помощью технологии Openmp / С. И. Мартыненко, В. А. Бахтин, Е. В. Румянцев, Г. А. Тарасов // Вестник МГТУ им. Н. Э. Баумана. Сер. Естественные науки – №2 (101) – 2022.

74. Аксенов, С. В. Применение технологии параллельных вычислений OpenMP для поиска образующих полиномов / С. В. Аксенов, А. Н. Мальчуков, Е. А. Мыцко // Вестник евразийской науки – №6 (19) – 2013.

75. Тузко, Я. Н. Организация параллельных вычислений в многоядерных процессорах / Я. Н. Тузко, О. О. Соколова, Б. А. Акишин // Молодой исследователь Дона – №5(14) – 2018 (14).

76. Бабенко, Л. К. Параллельные алгоритмы для решения задач защиты информации / Л. К. Бабенко, Е. А. Ищукова, И. Д. Сидоров //. – М.: Горячая линия-Телеком. – 2014. – 304 с.

77. Гервич, Л. Р. Об автоматизации применения размещения данных с перекрытиями в распределенной памяти / Л. Р. Гервич, Б. Я. Штейнберг // Вестник ЮУрГУ. Серия: Математическое моделирование и программирование – №1. – 2023.

78. Nikolaevskaya, E. Parallel Calculations Using MPI. Studies / E. Nikolaevskaya, A. Khimich, T. Chistyakova // Computational Intelligence. – Vol. 397. – 2012. – P.99-121. – DOI:10.1007/978-3-642-25673-8_6.

79. Francis, P. An Introduction to Parallel Programming Using MPI / P. Francis, J. Whiteley // Guide to Scientific Computing in C++. – Vol. 2. – 2017. – DOI:10.1007/978-3-319-73132-2_11.

80. Moreira, J. Optimizing Infrastructure for MPI Applications / J. Moreira // High Performance Computing in Clouds. – Vol 1. – 2023. – P.147-161. – DOI:10.1007/978-3-031-29769-4_8.

81. Babenko, L. Development and Testing of the Information Security Protocol in a Medical Cloud Platform / L. Babenko, D. Alekseev, E. Ishchukova, A. Shumilin // Proceedings of the International Workshop on Advanced in Information Security Management and Applications. – 2021. – Vol. 3094. – Article № 3. – P. 35-40.

82. Babenko, L. Algorithm of ensuring confidential data security of the cloud medical information system / L. Babenko, A. Shumilin, D. Alekseev // E3S Web of

Conferences. – 2020. – Vol. 224. – Article № 03023. – P. 1-8. – DOI 10.1051/e3sconf/202022403023.

83. Development and Testing of the Information Security Protocol in the Medical Cloud Platform / L. Babenko, D. Alekseev, E. Ishchukova, A. Shumilin // CEUR Workshop Proceedings : Advanced in Information Security Management and Applications 2021. 2022. – Vol. 3094. – Article № 3. – P. 35-40.

84. Система автоматического поиска участков эпилептической активности в составе облачной платформы хранения, систематизации и обработки медицинских данных / Д. М. Алексеев, А. Н. Минюк, З. А. Понимаш, А. С. Шумилин // Современные наукоемкие технологии. – 2019. – № 1. – С. 14-19.

85. Шумилин, А. С. Разработка алгоритма для обеспечения защиты конфиденциальных данных в МИС с облачной архитектурой / А. С. Шумилин, Д. М. Алексеев // Проблемы передачи информации в инфокоммуникационных системах : сборник докладов и тезисов XII Всероссийской научно-практической, г. Волгоград, 20 мая 2022 г. – Волгоград : Издательство Волгоградского государственного университета, 2022. – С. 15-19.

86. Алексеев, Д. М. Метод защиты информации в распределенной облачной информационной системе здравоохранения / Д. М. Алексеев, А. С. Шумилин // Развитие правового сознания в образовательном пространстве : материалы Международной 9-ой научно-практической конференции, Махачкала, 22 февраля 2022 г. Ч. 1. – Махачкала : Дагестанский государственный педагогический университет, 2022. – С. 92-103.

87. Алексеев, Д. М. Использование пороговой гомоморфной криптосистемы с открытым ключом для защиты информации в медицинской облачной платформе / Д. М. Алексеев, А. С. Шумилин // Интеграция науки, общества, производства и промышленности: проблемы и перспективы : сборник статей по итогам Международной научно-практической конференции, г. Волгоград, 29 мая 2021 г. – Стерлитамак : Агентство международных исследований, 2021. – С. 86-88.

88. Алексеев, Д. М. Облачная медицинская информационная система: защита конфиденциальной информации с использованием порогового гомоморфного шифрования / Д. М. Алексеев, А. С. Шумилин // Проблемы и перспективы разработки инновационных технологий : сборник статей Международной научно-практической конференции, г. Магнитогорск, 1 июня 2021 г. – Магнитогорск ; Уфа : Аэтерна, 2021. – С. 6-8.

89. Алексеев, Д. М. Обзор методов обеспечения безопасности конфиденциальных данных в медицинских информационных системах / Д. М. Алексеев, А. С. Шумилин // Внедрение результатов инновационных разработок: проблемы и перспективы : сборник статей Международной научно-практической конференции, г. Челябинск, 12 января 2021 г. : [в 2 ч.]. Ч. 1. – Челябинск ; Уфа : МЦИИ Омега Сайнс, 2021. – С. 47-49.

90. Алексеев, Д. М. Методы защиты информации при передаче медицинских обследований в облачной платформе / Д. М. Алексеев, А. С. Шумилин // Совершенствование методологии и организации научных исследований в целях развития общества : сборник статей по итогам Международной научно-практической конференции, г. Новосибирск, 29 декабря 2020 г. : [в 2 ч.]. Ч. 2. – Стерлитамак : Агентство международных исследований, 2020. – С. 117-120.

91. Алексеев, Д. М. Методы и подходы к обеспечению конфиденциальности персональных данных в медицинских информационных системах / Д. М. Алексеев, А. С. Шумилин // Научно-технический прогресс как механизм развития современного общества : сборник статей Всероссийской научно-практической конференции, г. Тюмень, 13 января 2021 г. – Тюмень ; Уфа : Аэтерна, 2021. – С. 19-21.

92. Алексеев, Д. М. Защита конфиденциальной информации в облачной медицинской информационной системе / Д. М. Алексеев, А. Н. Минюк, А. С. Шумилин // Инновационная наука. – 2020. – № 6. – С. 32-33.

93. Шифрование изображении обследовании в медицинской облачной платформе хранения и обработки данных / Д. М. Алексеев, А. Н. Минюк, А. С. Шумилин, Л. К. Бабенко // Студенческая наука для развития информационного общества : Сборник материалов X Всероссийской научно-технической конференции с международным участием, Ставрополь, 07–08 ноября 2019 года. Том Часть 1. – Ставрополь: Северо-Кавказский федеральный университет, 2019. – С. 27-36.

94. Модель многоканального безопасного обслуживания в Private Cloud системе / А. Н. Волокита, Д. Т. Ву, А. В. Щербина [и др.] // Вестник Брестского государственного технического университета. Физика, математика, информатика. – 2013. – № 5(83). – С. 26-29.

ПРИЛОЖЕНИЕ 1 – Акты о внедрении и использовании результатов диссертационной работы

«УТВЕРЖДАЮ»

Технический директор

ООО «СиВижинЛаб»

В.В. Губарев

17.01.2024

АКТ

об использовании результатов, полученных в рамках диссертационной работы Шумилина Александра Сергеевича, в коммерческом проекте ООО «СиВижинЛаб»

Настоящим актом подтверждается, что результаты диссертационной работы Шумилина Александра Сергеевича «Метод обеспечения безопасности конфиденциальной информации в распределенной медицинской облачной системе» были использованы в ходе работы над проектом по разработке программного обеспечения в ООО «СиВижинЛаб».

Шумилиным А.С. была предложена архитектура облачной медицинской информационной системы, поддерживающая возможность проведения диагностических обследований по протоколам DICOM и HL7, а также процедуру обмена и хранения медицинских данных. Особенностью предложенной архитектуры является интеграция с разработанным в ходе диссертационного исследования методом защиты медицинских данных, позволяющим обеспечить безопасность файлов при помощи механизмов шифрования и разделения файла на фрагменты для последующего децентрализованного хранения.

Руководитель группы разработки _____ Д.В. Лапин

«УТВЕРЖДАЮ»

Директор ООО НМФ «Нейротех»

В. Л. Сахаров

«26» января 2024 г.

АКТ

**об использовании результатов,
полученных в рамках диссертационной работы
Шумилина Александра Сергеевича,
в рабочих процессах ООО НМФ «Нейротех»**

Настоящий акт подтверждает использование результатов диссертационной работы «Метод обеспечения безопасности конфиденциальной информации в распределенной медицинской облачной системе» Шумилина Александра Сергеевича в рабочих процессах ООО НМФ «Нейротех».

Шумилиным Александром Сергеевичем был предложен метод обеспечения защиты медицинских данных на основе схемы разделения секрета, который был апробирован в пилотном проекте организации. В ходе апробации предложенного метода было отмечено, что особое внимание уделено вопросу безопасности хранения данных и предусмотрена возможность самостоятельно задавать пороговое значение фрагментов, которое необходимо иметь в наличии коалиции пользователей для получения доступа к исходному файлу.

Технический директор
ООО НМФ «Нейротех» А. Н. Минюк

Интегра
ИНЖЕНЕРНЫЙ ЦЕНТР

347927, г. Таганрог, Ростовская обл.,
ул. Поляковское шоссе, 16, оф.310
e-mail: info@ec-integra.ru
от 12.02.2024 г. № 14-И

УТВЕРЖДАЮ

Директор ООО «Инженерный центр «Интегра»

Басан А.

АКТ

в внедрении результатов диссертационной работы

Настоящим актом подтверждается, что результаты диссертационной работы «Метод обеспечения безопасности конфиденциальной информации в распределенной медицинской облачной системе» Шумилина Александра Сергеевича использованы в научно-производственной деятельности ООО «Инженерный центр Интегра» (г. Таганрог), а именно апробация результатов проведенных исследований в рамках внутренних задач компании, включающих проведение предварительных испытаний реализации предложенного метода на стендовой медицинской информационной системе с целью определения ее работоспособности и готовности к опытной эксплуатации.

Технический директор Д.В. Шахов

«УТВЕРЖДАЮ»

Заведующий кафедрой

Безопасность информационных технологий

имени О.Б. Макаревича ИКТИБ ЮФУ

_____ Е.С. Абрамов

А К Т

об использовании результатов, полученных в рамках диссертационной работы Шумилина Александра Сергеевича на кафедре безопасности информационных технологий имени О.Б. Макаревича ИКТИБ ЮФУ

Настоящим актом подтверждается, что результаты диссертационной работы Шумилина Александра Сергеевича «Метод обеспечения безопасности конфиденциальной информации в распределенной медицинской облачной системе» были использованы в учебно-исследовательском процессе на кафедре безопасности информационных технологий имени О.Б. Макаревича ИКТИБ ЮФУ.

Шумилиным А.С. была предложен метод защиты медицинских данных, позволяющий обеспечить безопасность файлов при помощи механизмов шифрования и разделения файла на фрагменты с применением схемы разделения секрета Шамира для последующего децентрализованного хранения. Материалы диссертации планируется использовать в учебном процессе при подготовке студентов по специальности 10.05.03 «Информационная безопасность автоматизированных систем» в дисциплине «Криптографические методы защиты информации».

Научный руководитель

доктор технических наук, профессор _____ Л.К. Бабенко

ПРИЛОЖЕНИЕ 2 – Свидетельство о государственной регистрации программы для ЭВМ

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2023665287

«Программа для реализации алгоритма защиты результатов обследований пациентов в медицинской информационной системе на основе схемы разделения секрета»

Правообладатель: ***Шумилин Александр Сергеевич (RU)***

Авторы: ***Шумилин Александр Сергеевич (RU), Алексеев Дмитрий Михайлович (RU), Бабенко Людмила Климентьевна (RU), Салманов Вячеслав Дмитриевич (RU)***

Заявка № **2023663429**
Дата поступления **28 июня 2023 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***13 июля 2023 г.***

Руководитель Федеральной службы по интеллектуальной собственности

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ
Сертификат 429b6a0fe3863164aaf96f83b73b4aa7
Владелец Зубов Юрий Сергеевич
Действителен с 10.05.2023 по 02.08.2024

Ю.С. Зубов

ПРИЛОЖЕНИЕ 3 – Листинг программной реализации алгоритма защиты результатов обследований

Основной модуль main.py

```
import os
import pyAesCrypt
import ast
from random import shuffle

import SliceFile
from Diffi_Helman import DiffiHelman
from Diseases import Disease
from Indications import Indication
from Patients import Patient
from Servers import Server
from Diffi_Helman import print_with_time

import MIRACL
from MIRACL import big, mirsys, epoint, ebrick

diffi_helman = DiffiHelman()
patient = Patient(main_path)
indication = Indication(main_path)
disease = Disease(main_path)
server = Server(main_path)

def send_file(file_name, server_list, NameFrom, NameTo, pathFrom):
    server.delete_file(file_name)
    server_path = str(server.get_path()) + str(server_list[0]) + "/" +
file_name + ".txt"

    diffi_helman.send_file(NameFrom, NameTo, pathFrom, server_path)
    SliceFile.cut(len(server_list), server_path)

  pyAesCrypt.encryptFile(server_path.replace(".txt", "_1.txt"),
server_path.replace(".txt", "_1.txt.aes"), server.get_code(server_list[0]))
  os.remove(server_path.replace(".txt", "_1.txt"))

  for i in range(1, len(server_list)):

  server_end_path = str(server.get_path()) + str(server_list[i]) + "/"
+ file_name + "_" + str(i + 1) \ + ".txt"

diffi_helman.send_file("Сервер" + str(server_list[0]), "Сервер" +
str(server_list[i]), server_path.replace(".txt", "_" + str(i + 1) +
".txt"),server_end_path)
```

```
    pyAesCrypt.encryptFile(server_end_path,  server_end_path  +  ".aes",
server.get_code(server_list[i]))
    os.remove(server_end_path)

    def get_file(file_name, server_list):


        server_path = str(server.get_path()) + str(server_list[0]) + "/" +
file_name + "_1.txt"


        pyAesCrypt.decryptFile(server_path   +   ".aes",   server_path,
server.get_code(server_list[0]))
        os.remove(server_path + ".aes")

        for i in range(1, len(server_list)):  # перебор серверов
            from_path = str(server.get_path()) + str(server_list[i]) + "/"
+ file_name + "_" + str(i + 1) \ + ".txt.aes"


            to_path = str(server.get_path()) + str(server_list[0]) + "/" +
file_name + "_" + str(i + 1) + ".txt"
            diffi_helman.send_file("Сервер"    +    str(server_list[i]),
"Сервер" + str(server_list[0]), from_path, to_path + ".aes")
            pyAesCrypt.decryptFile(to_path    +    ".aes",    to_path,
server.get_code(server_list[i]))
            os.remove(to_path + ".aes")  # удаление


        SliceFile.combine(len(server_list),   str(server.get_path())   +
str(server_list[0]) + "/" + file_name + ".txt")


    def patients_options():
        print("Выберете действие\n")
        print("0)Выйти")
        print("1)Добавить пациента")
        print("2)Удалить пациента")
        print("3)Список пациентов")
        print("4)Информация о пациенте")
        print("5)Добавить жалобу")
        print("6)Удалить жалобы")
        try:
            option = int(input())
           except ValueError:
               print("Неправильно введено число")
            return
        if option == 0:
```

```
    return
elif option == 1:  # Добавление пациента


    print("Введите информацию о пациенте")
    info_massive = []
    local_text = input("Введите ФИО:").strip()
    info_massive.append("name:" + local_text)
    local_text = input("Введите возраст:").strip()
    info_massive.append("age:" + local_text)
    local_text = input("Введите пол:").strip()
    info_massive.append("gender:" + local_text)
    patient.add_patient(info_massive)  # добавление нового пациента
elif option == 2:  # Удаление пациента
    print("Введите id пациента")
    try:
        option = int(input())
            except ValueError:
        print("Неправильно введено число")
        return
    patient.delete_patient(option)
elif option == 3:
    patient.list()
elif option == 4:
    print("Введите id пациента")
    try:
        option = int(input())
             except ValueError:
        print("Неправильно введено число")


  info = patient.get_info(str(option))

    for key in info:
        print(key + ' ' + info[key])
elif option == 5:
    print("Введите id пациента")
    try:
        patient_id = int(input())
          except ValueError:
        print("Неправильно введено число")
        return
    patient_info = patient.get_info(patient_id)
    if len(patient_info) == 0:
        return
    print("Введите id заболевания")
    try:
        disease_id = int(input())
             except ValueError:
```

```
            print("Неправильно введено число")
            return
        disease_info = disease.get_info(disease_id)
        if len(disease_info) == 0:
            return
        patient.add_diseases(patient_id, disease_id)
    elif option == 6:
        print("Введите id пациента")
        try:
            patient_id = int(input())
                 except ValueError:
            print("Неправильно введено число")
            return
        patient_info = patient.get_info(patient_id)
        if len(patient_info) == 0:
            return
        patient.dell_diseases(patient_id)  # Удаляем жалобы
    else:
        print("Нет такого действия")

def disease_options():
    print("Выберете действие\n")
    print("0)Выйти")
    print("1)Добавить заболевание")
    print("2)Удалить заболевание")
    print("3)Список заболеваний")
    print("4)Информация о заболевании")
    print("5)Добавить симптомы")
    print("6)Список симптомов")
    print("7)Удалить симптомы")
    try:
        option = int(input())
           except ValueError:
                print("Неправильно введено число")
       return
    if option == 0:
        return
    elif option == 1:
        print("Введите информацию о заболевании")
        info_massive = []
        local_text = input("Введите название:").strip()
        info_massive.append("name:" + local_text)
        local_text = input("Введите описание:").strip()
        info_massive.append("description:" + local_text)
        disease.add_disease(info_massive)
    elif option == 2:
        print("Введите id заболевания")
        try:
            option = int(input())
```

```
            except ValueError:
        print("Неправильно введено число")
    disease.delete_disease(option)
elif option == 3:
    disease.list()
elif option == 4:
    print("Введите id заболевания")
    try:
        option = int(input())
            except ValueError:
        print("Неправильно введено число")
    info = disease.get_info(str(option))
    for key in info:
        print(key + ' ' + info[key])
elif option == 5:
    print("Введите id заболевания")
    try:
        disease_id = int(input())
            except ValueError:
        print("Неправильно введено число")
        return
    disease_info = disease.get_info(disease_id)
    if len(disease_info) == 0:
        return
    print("Введите id показания")
    try:
        indication_id = int(input())
            except ValueError:
        print("Неправильно введено число")
        return
    indication_info = indication.get_info(indication_id)
    if len(indication_info) == 0:
        return
    disease.add_indications(disease_id, indication_id)
elif option == 6:
    print("Введите id заболевания")
    try:
        disease_id = int(input())
            except ValueError:
        print("Неправильно введено число")
        return

    disease.dell_indications(disease_id)
elif option == 7:
    print("Введите id заболевания")
    try:
        disease_id = int(input())
            except ValueError:
        print("Неправильно введено число")
```

```
            return
        disease_info = disease.get_info(disease_id)
        if len(disease_info) == 0:
            return
        disease.dell_indications(disease_id)
    else:
        print("Нет такого действия")

def indications_options():
    print("Выберете действие\n")
    print("0)Выйти")
    print("1)Добавить показание")
    print("2)Удалить показание")
    print("3)Список показаний")
    print("4)Информация о показании")
    try:
        option = int(input())
           except ValueError:
        print("Неправильно введено число")
        return
    if option == 0:
        return
    elif option == 1:
        print("Введите информацию о показании")
        info_massive = []
        local_text = input("Введите наименования:").strip()
        info_massive.append("name:" + local_text)
        local_text = input("Введите минимум:").strip()
        info_massive.append("min:" + local_text)
        local_text = input("Введите максимум:").strip()
        info_massive.append("max:" + local_text)
        indication.add_indication(info_massive)
    elif option == 2:
        print("Введите id показания")
        try:
            option = int(input())
                except ValueError:
            print("Неправильно введено число")
        indication.delete_indication(option)
    elif option == 3:
        indication.list()
    elif option == 4:
        print("Введите id показания")
        try:
            option = int(input())
        except ValueError:
            print("Неправильно введено число")
        info = indication.get_info(str(option))
        for key in info:
```

```
            print(key + ' ' + info[key])
    else:
        print("Нет такого действия")

def servers_options():
    print("Выберете действие\n")
    print("0)Выйти")
    print("1)Добавить сервер")
    print("2)Удалить сервер")
    print("3)Список серверов")
    print("4)Информация о сервере")
    try:
        option = int(input())
           except ValueError:
        print("Неправильно введено число")
        return
    if option == 0:
        return
    elif option == 1:
        print("Введите информацию о сервере")
        info_massive = []
        local_text = input("Введите наименования:").strip()
        info_massive.append("name:" + local_text)
        local_text = input("Введите статус(0/1):").strip()
        info_massive.append("status:" + local_text)
        server.add_server(info_massive)
    elif option == 2:
        print("Введите id сервера")
        try:
            option = int(input())
                except ValueError:
            print("Неправильно введено число")
        server.delete_server(option)
    elif option == 3:
        server.list()
    elif option == 4:
        print("Введите id сервера")
        try:
            option = int(input())
                except ValueError:
            print("Неправильно введено число")
        info = server.get_info(str(option))
        for key in info:
            print(key + ' ' + info[key])
    else:
        print("Нет такого сервера")


# запуск
```

```
    def play_movie():
        if len(patient.list(True)) == 0:
            print("Начало")
            return
        else:
            print("Введите id пациента")
        try:
            patient_id = int(input())
               except ValueError:
            print("Неправильно введено число")
            return
        patient_info = patient.get_info(patient_id)
        if len(patient_info) == 0:
            return

        print_with_time("Пациент \"" + patient_info["name"] + "\" в
клинике")
        diseases_list = patient.get_diseases(patient_id)  # Получаем список
жалоб у пациента
        if len(diseases_list) == 0:
            print_with_time("У пациента \"" + patient_info["name"] + "\"
нет жалоб")
            return

        for element in diseases_list:
            disease_info = disease.get_info(element)
            print_with_time("Пациента \"" + patient_info["name"] + "\"
думает, что у него \"" + disease_info["name"] + "\"")

        server_list = server.get_active_servers()
        if len(server_list) == 0:
            print("Сервера недоступны")
            return

        shuffle(server_list)
        for diseases_id in diseases_list:    # Проходимся по списку
заболеваний
            disease_info = disease.get_info(diseases_id)
            indication_list = disease.get_indications(diseases_id)
            if len(indication_list) == 0:  # Если вдруг появилось жалоба
без симптомов
                print_with_time("не указан симптом" + disease_info["name"]
+ ")")
                return

            for indication_id in indication_list:
                indication_info = indication.get_info(indication_id)
                print_with_time("Пациент отправлен к специалисту \"" +
indication_info["name"] + "\"")
```

```
                indication.take_test(patient_id,    indication_id)       #
Получаем анализ пациента
                print_with_time("Лаборант     взял     анализ     \""     +
indication_info["name"] + "\"")

                # записываем имя файла
                file_name  =  "indication_"  +  str(patient_id)  +  "_"  +
str(indication_id)
                send_file(file_name=file_name, server_list=server_list,
      NameFrom="Лаборант"    +    str(indication_id),    NameTo="Сервер"    +
str(server_list[0]), pathFrom=indication.get_path() + str(indication_id) +
"/" + file_name + ".txt")

        print_with_time("Специалист узнаёт об анализах")
        print_with_time("Специалист просматривает анализы")

        specialist_file  =  open(indication.get_path()  +  "specialist_"  +
str(patient_id) + ".txt", "w")
        specialist_file.write('{')


        specialist_dictionary = indication.get_dictionary()
        count = 0
        {key:value,<--(Я имею в ввиду эту) key2:value2}
        for diseases_id in diseases_list:
            indication_list = disease.get_indications(diseases_id)
            for indication_id in indication_list:
                file_name  =  "indication_"  +  str(patient_id)  +  "_"  +
str(indication_id)
                get_file(file_name=file_name, server_list=server_list)
                indication_file     =     open(str(server.get_path())     +
str(server_list[0]) + "/" + file_name + ".txt", "r")
                indication_value = float(indication_file.read())

                if count != 0:
                    specialist_file.write(',')
                if   specialist_dictionary[int(indication_id)]['high']   >=
indication_value:
                    specialist_file.write(str(indication_id) + ":'high'")
                elif  specialist_dictionary[int(indication_id)]['low']  <=
indication_value:
                    specialist_file.write(str(indication_id) + ":'low'")
                else:
                    specialist_file.write(str(indication_id) + ":'good'")
                    indication_file.close()
                os.remove(str(server.get_path())  +  str(server_list[0])  +
"/" + file_name + ".txt")
```

```
                count += 1
            specialist_file.write('}')
            specialist_file.close()
            print_with_time("Специалист сделал выводы")
            file_name = "specialist_" + str(patient_id)

            shuffle(server_list)
            send_file(file_name=file_name, server_list=server_list,
       NameFrom="Специалист",    NameTo="Сервер"    +    str(server_list[0]),
pathFrom=indication.get_path() + "specialist_" + str(patient_id) + ".txt")
            print_with_time("Доктор узнаёт об результате исследований")
            print_with_time("Доктор просматривает результат исследований")

            file_name = "specialist_" + str(patient_id)
            get_file(file_name=file_name, server_list=server_list)
            specialist_file       =       open(str(server.get_path())       +
str(server_list[0]) + "/" + file_name + ".txt", 'r')
            specialist_rezult                                                 =
ast.literal_eval(specialist_file.read().strip())
            specialist_file.close()
            os.remove(str(server.get_path())  +  str(server_list[0])  +  "/"  +
file_name + ".txt")
            doctor_file = open(disease.get_path() + "doctor_" + str(patient_id)
+ ".txt", "w")
            doctor_file.write("{")
            count = 0
            for diseases_id in diseases_list:
                if count != 0:
                    doctor_file.write(",")
                is_ill = True
                indication_list = disease.get_indications(diseases_id)

                for indication_id in indication_list:
                    if specialist_rezult[int(indication_id)] != 'high':
                        doctor_file.write(str(diseases_id) + ":0")
                        is_ill = False
                        break
                if is_ill:
                    doctor_file.write(str(diseases_id) + ":1")
                count += 1
            doctor_file.write("}")
            doctor_file.close()
            print_with_time("Доктор сделал выводы")

            file_name = "doctor_" + str(patient_id)
            shuffle(server_list)
            send_file(file_name=file_name,            server_list=server_list,
NameFrom="Специалист",      NameTo="Сервер"      +      str(server_list[0]),
pathFrom=disease.get_path() + "doctor_" + str(patient_id) + ".txt")
```

```
        print_with_time("Пациент узнаёт об результатах")
        print_with_time("Пациент просматривает результаты")
        file_name = "doctor_" + str(patient_id)
        get_file(file_name=file_name, server_list=server_list)
        doctor_file = open(str(server.get_path()) + str(server_list[0]) +
"/" + file_name + ".txt", 'r')
        doctor_rezult = ast.literal_eval(doctor_file.read().strip())
        doctor_file.close()
        for diseases_id in diseases_list:
            disease_info = disease.get_info(diseases_id)
            if doctor_rezult[int(diseases_id)] == 0:
                print_with_time("Пациент узнаёт, что он не болен \"" +
disease_info['name'] + "\"")
            else:
                print_with_time("Пациент  узнаёт,  что  он  болен  \""  +
disease_info['name'] + "\"")

        diffi_helman.send_file("Сервер" + str(server_list[0]), "Пациент",
     str(server.get_path())  +  str(server_list[0])  +  "/"  +  file_name  +
".txt", str(patient.get_path()) + str(patient_id) + "/" + file_name + ".txt")
     while True:
        mod = 0
           print("Выберете опцию\n")
           print("0)Выйти")
           print("1)Пациенты")
          print("2)Заболевания")
          print("3)Показания")
          print("4)Сервера")
          print("5)Запуск эмуляции")
        try:
            mod = int(input())
        except ValueError:
            print("Неправильно введено число")
            continue

        if mod == 0:
            break
        elif mod == 1:
            patients_options()
        elif mod == 2:
            disease_options()
        elif mod == 3:
            indications_options()
        elif mod == 4:
            servers_options()
        elif mod == 5:
            play_movie()
            print("Сюжет окончен")
```

```
    input("Press enter...")
else:
    print("Нет такой опции")
print()
```

Модуль diseases.py

```
import os
class Disease:
    path = ""
    def get_path(self):
        return self.path
    def __init__(self, main_path):
        self.path = main_path + "diseases/"
        if not os.path.exists(self.path):
            os.mkdir(self.path)
    def add_disease(self, info_massive):
        local_path = self.path + str(self.get_next_id()) + "/"
        os.mkdir(local_path)
        info_file = open(local_path + "info.txt", "w")
        for info in info_massive:
            info_file.write(info+"\n")

    def delete_disease(self, disease_id):
        if os.path.exists(self.path + disease_id):
            os.remove(self.path + disease_id)
        else:
            print("Нет такой болезни")

    def list(self, get_list=False):
        if not get_list:
            for file in sorted(os.listdir(self.path)):
                disease_info = self.get_info(file)
                print(file, disease_info['name'])
        else:
            return sorted(os.listdir(self.path))

    def get_next_id(self):
        massive = sorted(os.listdir(self.path))
        if len(massive) > 0:
            return int(massive[len(massive)-1]) + 1
        else:
            return 0

    def get_info(self, file):
        if os.path.exists(self.path + str(file) + "/info.txt"):
            local_file = open(self.path + str(file) + "/info.txt", "r")
            disease_info = {}
```

```
                for line in local_file.readlines():
                    disease_info[line[:line.index(':')]] =
line[line.index(':')+1:].strip()
                return disease_info
            else:
                print("Нет такой болезни")
                return {}

        def add_indications(self, file, indication_id):
            local_file = open(self.path + str(file) + "/indications.txt",
"a")
            local_file.write(str(indication_id) + "\n")

        def dell_indications(self, file):
            if os.path.exists(self.path + str(file) + "/indications.txt"):
                os.remove(self.path + str(file) + "/indications.txt")

        def get_indications(self, file):
            if os.path.exists(self.path + str(file) + "/indications.txt"):
                local_file = open(self.path + str(file) +
"/indications.txt", "r")
                indications_list = []
                for indication in local_file.readlines():
                    indications_list.append(indication.strip())
                return indications_list
            else:
                print("У пациента нет жалоб")
                return []
```

Модуль Diffi_Helman.py

```
import os
import random
import pyAesCrypt
import time

class DiffiHelman:
    animation_time = 0

    def send_file(self, User1, User2, file_from, file_to):
        print_with_time("Генерация секретного ключа " + str(User1), 2
* self.animation_time)

        for i in range(1, 4):
            print_with_time("." * i, self.animation_time // 2)

        Sender_SecretKey = random.randint(0, 20)
        print_with_time("Секретный ключ сгенерирован " + str(User1), 2
* self.animation_time)

        Sender_Key = (7 ** Sender_SecretKey) % 71
        print_with_time("Передача публичного ключа " + str(User1), 2 *
self.animation_time)

        for i in range(1, 4):
            print_with_time("." * i, self.animation_time // 2)

        print_with_time("Генерация секретного ключа " + str(User2), 2
* self.animation_time)
        for i in range(1, 4):
            print_with_time("." * i, self.animation_time // 2)

        Receiver_SecretKey = random.randint(0, 20)
        print_with_time("Секретный ключ сгенерирован " + str(User2), 2
* self.animation_time)

        Receiver_Key = (7 ** Receiver_SecretKey) % 71
        print_with_time("Передача публичного ключа " + str(User2), 2 *
self.animation_time)

        for i in range(1, 4):
            print_with_time("." * i, self.animation_time // 2)
```

```
            if (Receiver_Key ** Sender_SecretKey) % 71 == (Sender_Key **
Receiver_SecretKey) % 71:
                Secret_Session_Key = (Receiver_Key ** Sender_SecretKey) %
71
                print_with_time("Общий секретный ключ выработан", 2 *
self.animation_time)

                pyAesCrypt.encryptFile(file_from,   file_to   +   ".aes",
str(Secret_Session_Key))
                print_with_time("Файл передан", 2 * self.animation_time)

                if os.path.isfile(file_from):
                    os.remove(file_from)
                else:
                    print("File", file_from, "dosn't exists!!!")

                pyAesCrypt.decryptFile(file_to    +    ".aes",    file_to,
str(Secret_Session_Key))
                print_with_time("Файл       расшифрован",       2       *
self.animation_time)

                if os.path.isfile(file_to + ".aes"):
                    os.remove(file_to + ".aes")
                else:
                    print("File", file_to + ".aes", "dosn't exists!!!")


    def print_with_time(text, time_sleep=1):
        if time_sleep > 0:
            print(text)
            time.sleep(time_sleep)
```

Модуль Indications.py

```
import os
import random
import ast

class Indication:
    path = ""

    def get_path(self):
        return self.path

    def __init__(self, main_path):
        self.path = main_path + "indications/"
        if not os.path.exists(self.path):
            os.mkdir(self.path)

    def add_indication(self, info_massive):
        local_path = self.path + str(self.get_next_id()) + "/"
        os.mkdir(local_path)
        info_file = open(local_path + "info.txt", "w")
        for info in info_massive:
            info_file.write(info+"\n")

    def delete_indication(self, indication_id):
        if os.path.exists(self.path + indication_id):
            os.remove(self.path + indication_id)
        else:
            print("Нет такого показания")

    def list(self, get_list=False):
        if not get_list:
            for file in sorted(os.listdir(self.path)):
                indication_info = self.get_info(file)
                print(file, indication_info['name'])
        else:
            return sorted(os.listdir(self.path))

    def get_next_id(self):
        massive = sorted(os.listdir(self.path))
        if len(massive) > 0:
            return int(massive[len(massive)-1]) + 1
        else:
```

```
                return 0

        def get_info(self, file):
            if os.path.exists(self.path + str(file) + "/info.txt"):
                local_file = open(self.path + str(file) + "/info.txt", "r")
                indication_info = {}
                for line in local_file.readlines():
                    indication_info[line[:line.index(':')]]=
line[line.index(':')+1:].strip()
                return indication_info
            else:
                print("Нет такого показания")
                return {}

        def take_test(self, patient_id, indication_id):
            indication_info = self.get_info(indication_id)
            indication_value = str(float(random.randint(
                int(float(indication_info["min"])          *          10),
int(float(indication_info["max"]) * 10)
            ))/10)
            file = open(self.path + str(indication_id) + "/indication_" +
str(patient_id) + "_" + str(indication_id) +
                        ".txt", "w")
            file.write(str(indication_value))

        def get_dictionary(self):
            with open(self.path + "info.txt", "r") as info_file:
                return dict(ast.literal_eval(info_file.read().strip()))
```

Модуль patients.py

```
import os

class Patient:
    path = ""

    def get_path(self):
        return self.path

    def __init__(self, main_path):
        self.path = main_path + "patients/"
        if not os.path.exists(self.path):
            os.mkdir(self.path)

    def add_patient(self, info_massive):
        local_path = self.path + str(self.get_next_id()) + "/"
        os.mkdir(local_path)
        info_file = open(local_path + "info.txt", "w")
        for info in info_massive:
            info_file.write(info+"\n")

    def delete_patient(self, id):
        if os.path.exists(self.path + id):
            os.remove(self.path + id)
        else:
            print("Нет такого пациента")

    def list(self, get_list=False):
        if not get_list:
            for file in sorted(os.listdir(self.path)):
                patient_info = self.get_info(file)
                print(file, patient_info['name'])
        else:
            return sorted(os.listdir(self.path))

    def get_next_id(self):
        massive = sorted(os.listdir(self.path))
        if len(massive) > 0:
            return int(massive[len(massive)-1]) + 1
        else:
            return 0
```

```
        def get_info(self, file):
            if os.path.exists(self.path + str(file) + "/info.txt"):
                local_file = open(self.path + str(file) + "/info.txt", "r")
                patient_info = {}
                for line in local_file.readlines():
                    patient_info[line[:line.index(':')]]                =
line[line.index(':')+1:].strip()
                return patient_info
            else:
                print("Нет такого пациента")
                return {}

        def add_diseases(self, file, diseases_id):
            local_file = open(self.path + str(file) + "/diseases.txt", "a")
            local_file.write(str(diseases_id) + "\n")

        def dell_diseases(self, file):
            if os.path.exists(self.path + str(file) + "/diseases.txt"):
                os.remove(self.path + str(file) + "/diseases.txt")

        def get_diseases(self, file):
            if os.path.exists(self.path + str(file) + "/diseases.txt"):
                local_file = open(self.path + str(file) + "/diseases.txt",
"r")
                diseases_list = []
                for disease in local_file.readlines():
                    diseases_list.append(disease.strip())
                return diseases_list
            else:
                print("У пациента нет жалоб")
                return []
```

Модуль servers.py

```
import os
import random

class Server:
    path = ""

    def get_path(self):
        return self.path

    def get_code(self, server_id):
        if os.path.exists(self.path + str(server_id) + "/"):
            code_file = open(self.path + str(server_id) + "/code.txt",
"r")
            return code_file.readline().strip()

    def __init__(self, main_path):
        self.path = main_path + "servers/"
        if not os.path.exists(self.path):
            os.mkdir(self.path)

    def add_server(self, info_massive):
        local_path = self.path + str(self.get_next_id()) + "/"
        os.mkdir(local_path)
        info_file = open(local_path + "info.txt", "w")
        for info in info_massive:
            info_file.write(info+"\n")
        code_file = open(local_path + "/code.txt", "w")
        code_file.write(str(random.randint(0, 999999)))

    def delete_server(self, server_id):
        if os.path.exists(self.path + server_id):
            os.remove(self.path + server_id)
        else:
            print("Нет такого сервера")

    def list(self, get_list=False):
        if not get_list:
            for file in sorted(os.listdir(self.path)):
                server_info = self.get_info(file)
                print(file, server_info['name'])
        else:
```

```
                return sorted(os.listdir(self.path))

        def get_next_id(self):
            massive = sorted(os.listdir(self.path))
            if len(massive) > 0:
                return int(massive[len(massive)-1]) + 1
            else:
                return 0

        def get_info(self, file):
            if os.path.exists(self.path + str(file) + "/info.txt"):
                local_file = open(self.path + str(file) + "/info.txt", "r")
                server_info = {}
                for line in local_file.readlines():
                    server_info[line[:line.index(':')]]                    =
line[line.index(':')+1:].strip()
                return server_info
            else:
                print("Нет такого сервера")
                return {}

        def get_active_servers(self):
            server_List = []
            for file in self.list(True):
                server_info = self.get_info(file)
                if server_info['status'] == '1':
                    server_List.append(file)
            return server_List

        def delete_file(self, name):
            for server in self.list(True):
                for file in os.listdir(self.path + server):
                    if name in file:
                        os.remove(self.path + server + "/" + file)
```

Модуль SliceFile.py

```
import os
import math

def cut(CountOfSlices, SourceToSlice):
    move = 0
    for i in range(1, CountOfSlices + 1):
        if i != CountOfSlices:
            bytes_to_read  =  math.ceil(os.path.getsize(SourceToSlice)
// CountOfSlices)
        else:
            bytes_to_read = os.path.getsize(SourceToSlice) - move
        extension = SourceToSlice[SourceToSlice.index('.'):]
        with open(SourceToSlice, 'rb') as ifile, \
                open(SourceToSlice[:SourceToSlice.index('.')] + '_' +
str(i) + extension, 'wb') as out_file:
            ifile.seek(move)
            move += bytes_to_read
            data = ifile.read(bytes_to_read)
            out_file.write(data)
    os.remove(SourceToSlice)
def combine(CountOfSlices, rezult):
    out_file = open(rezult, 'wb')
    extension = rezult[rezult.index('.'):]
    for i in range(1, CountOfSlices + 1):
        ifile  =  open(rezult[:rezult.index('.')]  +  '_'  +  str(i)  +
extension, 'rb')
        out_file.write(ifile.read())
        ifile.close()
        os.remove(rezult[:rezult.index('.')]   +   '_'   +   str(i)   +
extension)
    out_file.close()
```