Казахский национальный университет имени аль–Фараби

УДК 004.056.5

На правах рукописи

**УСАТОВА ОЛЬГА АЛЕКСАНДРОВНА**

# Разработка и исследование алгоритма аутентификации пользователей информационно-коммуникационных систем

6D100200– Системы информационной безопасности

Диссертация на соискание степени
доктора философии (PhD)

Научные консультанты:
Нысанбаева С.Е.,
д.т.н., профессор.
Wojcik Waldemar,
д.т.н., профессор

Республика Казахстан
Алматы, 2020

**СОДЕРЖАНИЕ**

**НОРМАТИВНЫЕ ССЫЛКИ**……..3
**ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ**……4
**ВВЕДЕНИЕ**……5
**1 АНАЛИЗ ИЗВЕСТНЫХ МЕТОДОВ И СРЕДСТВ ЗАЩИТЫ ИНФОРМАЦИОННЫХ СИСТЕМ НА ОСНОВЕ ДВУХФАКТОРНОЙ АУТЕНТИФИКАЦИИ**……11
1.1 Принципы построения защиты информации в информационной системе при аутентификации пользователя на основе второго фактора ……...11
1.2 Алгоритмы и протоколы аутентификации с использованием одноразового пароля……..19
1.3 Атаки и средства защиты информационных систем……24
1.4 Выводы по первому разделу……...32
**2 РАЗРАБОТКА АЛГОРИТМА ДВУХФАКТОРНОЙ АУТЕНТИФИКАЦИИ** ……...34
2.1 Система аутентификации пользователя на основе генерации одноразового пароля…….…34
2.2 Разработка алгоритма двухфакторной аутентификации……...……..35
2.3 Создание генераторов для формирования одноразового пароля двухфакторной аутентификации……..39
2.4 Разработка модели аутентификации пользователя на основе второго фактора……49
2.5 Выводы по второму разделу…….....50
**3 РЕАЛИЗАЦИЯ И АНАЛИЗ ЗАЩИТЫ ИНФОРМАЦИОННОЙ СИСТЕМЫ ПРИ ПОМОЩИ ДВУХФАКТОРНОЙ АУТЕНТИФИКАЦИИ** ……..52
3.1 Разработка архитектуры системы аутентификации…….52
3.2 Программная реализация защиты информации при аутентификации пользователя на основе одноразового ключа……..59
3.3 Практическая реализация системы защиты информации при аутентификации пользователя на основе одноразового ключа…….…68
3.4 Производительность разработанной программной реализации……..77
3.5 Выводы по третьему разделу …….....78
**ЗАКЛЮЧЕНИЕ**……...79
**СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**……81
**ПРИЛОЖЕНИЕ А** Список публикаций……89
**ПРИЛОЖЕНИЕ Б** Полученные авторские свидетельства……..91
**ПРИЛОЖЕНИЕ В** Акты о внедрении результатов диссертационной работы..93
**ПРИЛОЖЕНИЕ Г** Программное обеспечение системы защиты информации при аутентификации пользователя на основе одноразового ключа…….95

## НОРМАТИВНЫЕ ССЫЛКИ

В данной диссертации приведены ссылки на следующие стандарты:

1. СТ РК 34.026–2006 «Защита информации. Основные термины и определения»;
2. СТ РК ISO/IEC 27001–2015 «Информационная технология. Методы и средства обеспечения безопасности. Системы менеджмента информационной безопасностью. Требования»;
3. СТ РК ИСО/МЭК 10118–1–2006 «Информационная технология. Методы защиты информации. Хеш–функции. Часть 1. Общие положения
4. СТ РК ИСО/МЭК 10118–3–2006 (ISO/IEC 10118–3–2004, IDT) «Информационная технология. Методы защиты информации. Хеш–функции. Часть 3. Специализированные хеш–функции;
5. Информационная технология. Криптографическая защита информации. Функция хеширования. ГОСТ Р 34.11–2012. – Москва, Стандартинформ, 2012.

## ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

КН МОН РК – Комитет науки Министерства образования и науки Республики Казахстан;

ИИВТ – Республиканское государственное предприятие на праве хозяйственного ведения (РГП на ПХВ) «Институт информационных и вычислительных технологий» КН МОН РК;

OTP – One – TimePassword (Одноразовый пароль);

SMS – Short Message Service (Служба коротких сообщений);

DdoS – Distributed Denial of Service(Распределённая атака типа «отказ в обслуживании»);

API – Программный интерфейс приложения;

SSL – Secure Socket Layer (Уровень защищенных сокетов);

TLS – Transport Layer Security (Безопасность транспортного уровня);

ASCII – American standard code for information interchange(Американский стандартный код для обмена информацией);

2FA – Двухфакторная аутентификация;

АСУ– Автоматизированные системы управления;

БД – База данных;

ЛИБ – Лаборатория информационной безопасности;

ЗИ – Защита информации;

ИС – Информационные системы;

ИКС – Информационно – коммуникационные системы;

НИР – Научно – исследовательские работы;

НСД – Несанкционированный доступ;

ОС – Операционная система;

СОЦ – Система обеспечения целостности;

СУБД – Система управления базами данных.

# ВВЕДЕНИЕ

С момента приобретения независимости Республики Казахстан ее первый президент Нурсултан Абишевич Назарбаев неоднократно акцентировал внимание на необходимость защиты интересов граждан в Республике Казахстан. Так, в Послании Президента страны народу Казахстана от 10 октября 1997 года «Казахстан – 2030. Процветание, безопасность и улучшение благосостояния всех казахстанцев» долговременным приоритетом установлена государственная защищенность, где одним из важных направлений является информационная безопасность [1].

Комплексный механизм процессов обеспечения информационной безопасности нашей Республики охватывает организационные, социальные, технические и программные подходы, способные осуществлять конституционные права и свободу человека, гражданина в области получения информации, пользования ею в целях защиты конституционного строя, суверенитета и территориальной целостности Республики Казахстан, финансовой, а также общественной устойчивости, формирование выгодного интернационального партнерства в сфере информативной защищенности [2].

Принятый 6 января 2012 года Закон «О национальной безопасности Республики Казахстан» содержит необходимые статьи, в них даны чёткие определения с позиции государства, затрагивающих вопросы информационной безопасности страны в целом и граждан в частности [3].

На законодательном уровне в Республике Казахстан формируется и закрепляется национальная концепция обеспечения информационной безопасности, электронного документооборота, автоматизированных информационных систем, ресурсов, ИКТ, а также важных объектов.

При построении и эксплуатации телекоммуникационных сетей связи необходимо учитывать требования Республики Казахстан о соблюдении национальной безопасности в области связи. Об этом свидетельствует последняя действующая поправка статьи «О полномочиях государственных органов Республики Казахстан» от 27 декабря 2017 года [4].

21 мая 2013 года был принят закон Республики Казахстан (РК) № 94–V «О персональных данных и их защите» (с изменениями и дополнениями по состоянию на 28.12.2017 г.), в нем регулируются отношения, связанные со сбором и обработкой, а также защитой персональных данных [5]. Не маловажными являются Закон РК от 24 ноября 2015 года № 418–V «Об информатизации» (с изменениями по состоянию на 01.01.2020 г.) [6] и Закон РК от 7 января 2003 года № 370–II «Об электронном документе и электронной цифровой подписи» (с изменениями по состоянию на 25.11.2019г.) [7].

Развитие информационно–телекоммуникационных технологий позволяют использовать новые возможности в сети Интернет. Появилась возможность производить удалённые покупки товаров из любой точки страны, отслеживать состояние и исполнение работ, не присутствуя в местах их проведения. В настоящее время существует возможность оформить практически любую

справку, не выходя из дома, благодаря электронной цифровой подписи и информационному порталу государственных услуг.

В Республике Казахстан 12 декабря 2017 года, Постановлением Правительства №827 была утверждена программа государства «Цифровой Казахстан» [8]. В данной программе основным аспектом является развитие экономики страны и повышение жизнедеятельности населения, основанной на совершенствовании и ускорении развития инфокоммуникационных технологий и также создание цифровой экономики. Основное внимание уделено обеспечению информационной безопасности в сфере информационно-коммуникационных технологий и консолидации кибербезопасности автоматизированных информационных систем в нашей стране.

С развитием технологического прогресса и общего уровня информатизации появляются также и новые угрозы. Киберпреступность позволяет злоумышленникам совершать противоправные и незаконные действия, находясь в тысячах километрах от цели их атаки. В Послании народу Казахстана «Третья модернизация Казахстана: Глобальная конкурентоспособность» Президент Республики Казахстан отмечал, что все большую актуальность приобретает борьба с киберпреступностью [9]. В связи с этим была разработана и утверждена Постановлением Правительства Республики Казахстан от 30 июня 2017 года №407 Концепция кибербезопасности («Киберщит Казахстана») [10], в ней определены основные направления реализации государственной политики в сфере защиты электронных информационных ресурсов, информационных систем и сетей телекоммуникаций, обеспечения безопасного использования информационно-коммуникационных технологий

Информационные технологии стали неотъемлемой частью нашей жизни, в связи с этим осуществляется автоматизация многих процессов жизнедеятельности человека. Большая часть информации хранится в информационных системах, которую необходимо защищать. В атаках на информационные системы злоумышленники используют как ошибки в написании и администрировании программ, так и методы социальной психологии для получения желаемой информации. Разработчики ресурсов, на которых предполагается работа с данными пользователей, обязаны эти данные защищать и предотвращать возможность их утечек. Одной из основных проблем РК является слабое развитие отечественной индустрии информационной безопасности, в частности, в разработке средств криптографии. В Казахстане, в настоящее время, в действующих системах защиты электронных данных используются зарубежные криптографические алгоритмы и стандарты. Исследования, касающиеся безопасности данных, непосредственно объединены с государственными секретами и применение зарубежных готовых решений весьма рискованно, в связи с этим необходимо формировать собственные ресурсы для безопасности информации [11].

В ИИВТ КН МОН РК проводятся научно–исследовательские работы (НИР) по криптографической защите информации и разграничению доступа.

Этими НИР занимаются сотрудники Лаборатории информационной безопасности (заведующий лабораторией, д.т.н., проф. Бияшев Р.Г., академик НАН РК Калимолдаев М.Н. [12], д.т.н., асс. проф. Нысанбаева С.Е., ВНС, к.т.н. Капалова Н.А., PhD Бегимбаева Е.Е., научные сотрудники О.А.Рог, Д.С.Дюсембаев, К.Е.Алгазы, А.В.Варенников и другие). На осуществление этой деятельности в ИИВТ КН МОН РК выдана государственная лицензия № 549 от 03 июля 2017 г. Комитетом национальной безопасности РК.

Одними из основных средств защиты информационных систем от постороннего вмешательства являются идентификация и аутентификация, так как механизмы защиты информации рассчитаны на работу с поименованными субъектами и объектами. Аутентификация и идентификация пользователя являются взаимозависимыми действиями распознавания и проверки подлинности.

Основной целью аутентификации пользователя информационной системы является снижение угроз безопасности, а именно нарушение конфиденциальности и целостности информации. Несанкционированный доступ – один из самых распространенных видов нарушений, представляющий непосредственную угрозу работоспособности системы.

Аутентификация используется для доступа к социальным сетям, электронной почте, интернет магазинам, интернет–банкингу, платежным системам и т.д. Аутентификация пользователя классифицируется по следующим типам [13]:

– аутентификация на основе пароля: она проводится по одноразовым и многоразовым паролям. Многоразовый пароль задает пользователь, а система хранит его в базе данных. Он является одинаковым для каждой сессии. К ним относятся PIN–коды, слова, цифры, графические ключи. Одноразовые пароли для каждой сессии являются разными.

– комбинированная аутентификация, которая происходит с использованием нескольких методов, например, парольных и криптографических сертификатов. Она требует специальное устройство для считывания информации.

– биометрическая аутентификация: она предотвращает утечку или кражу персональной информации. Проверка проходит по физиологическим характеристикам пользователя, например, по отпечатку пальца, сетчатке глаза, распознавание лица и тембру голоса.

В настоящее время использование парольной аутентификации является доступной и распространенной из–за простоты применения. Этот метод защищенности ведет к увеличению прочности концепции защиты информации. Одним из эффективных методов защиты информации является двухфакторная аутентификация для входа в систему. Она предполагает двойную защиту данных посредством привязки аккаунта к системе защиты. После привязки пользователю необходимо будет взаимодействовать с этой системой для верификации данных.

Задачами безопасности информации, разграничение доступа и аутентификации на основе второго фактора, занимались зарубежные ученые Эдна Элизабет, С. Нивета [14], Фазех Садат Бабамир, Мурвет Кирчи [15], И Юй, Цзинша Хе, Нафей Чжу, Фанбо Цай, Мухаммед Салман Патан [16] и другие. Важной частью является также защита информации, хранящаяся в базах данных и программно–аппаратные средства для обработки и передачи информации. Существенный вклад в развитие этого направления был внесен также зарубежными учеными Мещеряковым Р. В. [17], Исхаковым А.Ю. [18], Полтавцевой М.А. [19] и другие.

В связи с вышеуказанным, тема данной диссертационной работы по разработке и исследованию алгоритма аутентификации пользователей информационно-коммуникационных систем на основе второго фактора является актуальной. В данной диссертации в качестве второго фактора является одноразовый пароль.

**Актуальность исследования** заключается в необходимости:

– реализации задач, поставленных в Государственной программе правительства Республики Казахстан и Концепции кибербезопасности («Киберщит Казахстана»), направленных на развитие государственной политики в сфере защиты электронных информационных ресурсов, систем и сетей телекоммуникаций, обеспечения безопасного использования информационно–коммуникационных технологий;

– разработки отечественных Казахстанских систем обеспечения информационной безопасности;

– применения политики парольной двухфакторной аутентификации с целью повышения надежности систем защиты информации.

**Цель диссертационной работы:** разработка, исследование и реализация алгоритма двухфакторной аутентификации для обеспечения защиты информации в информационно-коммуникационных системах.

**Задачи исследования,** реализующие цель диссертационного исследования:

1. Проведение обзора и анализа существующих систем защиты информации в информационно-коммуникационных системах и алгоритмов многофакторной аутентификации.
2. Разработка алгоритма аутентификации пользователей информационно-коммуникационных систем с использованием одноразового пароля.
3. Создание информационной системы для обеспечения целостности и защиты информации в информационно-коммуникационных системах.

**Объект исследования**: система защиты информации от несанкционированного доступа при аутентификации пользователя на основе второго фактора.

**Предметом исследования** являются процессы информационного взаимодействия пользователей информационно-коммуникационных систем и их аутентификация с помощью цифрового одноразового пароля.

**Научная новизна** проведенных исследований и полученных в работе результатов:

– разработан алгоритм двухфакторной аутентификации пользователя, основанный на генерации тригонометрических функций путем усложнения масштабирования функций при вычислении одноразового пароля. Масштабирование выполняется матричным представлением вариантов тригонометрических функций и использованием хеш-функций для вычисления координат и параметров, генерируемой тригонометрической функцией по текущему времени, секретной строке, логину и паролю первого аутентификационного кода;

– разработана модель процесса двухфакторной аутентификации пользователя на основе второго фактора, отличающаяся от известных тем, что она открытая и может генерировать наборы функций получения второго аутентификационного кода для каждой отдельной системы;

– предложена схема информационной системы программной реализации двухфакторной аутентификации с использованием мобильного устройства для ее внедрения и использования в закрытой сети.

**Личный вклад исследователя.** Разработан алгоритм аутентификации пользователя в информационной системе на основе второго фактора. Проведены численные исследования и экспериментальная оценка предлагаемых моделей и алгоритмов. Разработана архитектура клиент–серверной системы аутентификации и осуществлена программная реализация предложенной системы аутентификации пользователя при генерации одноразового пароля с использованием компьютерной программы аутентификатора и мобильного телефона.

**Апробация работы.** Результаты диссертационной работы докладывались и обсуждались на семинарах и конференциях ИИВТ и на кафедре «Информационные системы» КазНУ им.аль–Фараби, в том числе на международной научно-практической конференции «Инновационные технологии на транспорте: образование, наука, практика» в рамках реализации Послания Президента РК Н. Назарбаева «Новые возможности развития в условиях четвертой промышленной революции» (Каз АТК, Казахстан, Алматы, 2018); научной конференции «Современные проблемы информатики и вычислительной технологий» (ИИВТ КН МОН РК, Казахстан, Алматы, 2018); международной научной конференции «Информатика и прикладная математика» (ИИВТ КН МОН РК, Казахстан, Алматы, 2018); международной научно-методической конференции посвященной 90-летнему юбилею Казахского национального педагогического университета имени Абая (Каз НПУ, Казахстан, Алматы, 2018); International Conference on Wireless Communication, Network and Multimedia Engineering (WCNME2019) (Guilin, China, 2019); научной конференции «Инновационные IT и Smart–технологии» (ИИВТ КН МОН РК, Казахстан, Алматы, 2019); научной конференции «Современные проблемы информатики и вычислительных технологий» (ИИВТ КН МОН РК, Казахстан, Алматы, 2019); международной

научно–практической конференции «Информатика и прикладная математика» (ИИВТ КН МОН РК, Казахстан, Алматы, 2019); международной научно-практической конференции «Актуальные проблемы информационной безопасности в Казахстане» (ИИВТ КН МОН РК, Казахстан, Алматы, 2020).

**Связь темы с планами научно – исследовательских программ.** Представленные результаты получены при выполнении следующих проектов ИИВТ КН МОН РК (источник финансирования Комитет науки МОН РК):

– программно–целевого финансирования (ПЦФ) КН МОН РК «Разработка программных и программно–аппаратных средств для криптографической защиты информации при ее передаче и хранении в инфокоммуникационных системах и сетях общего назначения» в 2018–2019 годы;

– грантового финансирования (ГФ) КН МОН РК «Разработка казахстанского сегмента защищенного трансграничного информационного взаимодействия» в 2020 году.

**Публикации.** Основные результаты, проведенных исследований по теме диссертации, представлены в 15 публикациях, из которых 5 – в научных изданиях, рекомендуемых КН МОН РК, 2 – в международных научных изданиях, входящих в базу данных Scopus и Web of Science, 8 – в материалах международных научно практических конференций.

**Структура и объем диссертации.** Общий объем работы – 123 страницы. Диссертация состоит из введения, 3 разделов, заключения, списка используемых источников из 104 наименований, 4 приложений, включает 37 рисунков и 5 таблиц.

**Во введении** дано обоснование актуальности выбранной темы диссертационного исследования. Сформулированы цель, объект, предмет и задачи исследования. Описаны результаты проведенных исследований, показана их научная новизна и практическая значимость. Представлены данные об апробации результатов диссертационной работы.

**Первый раздел** посвящен исследованию известных методов и средств защиты информационных систем на основе двухфакторной аутентификации. Описаны принципы построения защиты информации в базе данных при аутентификации пользователя.

Представлена классификация распространенных методов двухфакторной аутентификации, используемых в информационных системах, рассмотрены недостатки и достоинства этих методов.

Рассмотрены алгоритмы и протоколы аутентификации с использованием одноразового кода. HOTP (HMAC – Based One – Time Password Algorithm)– алгоритм защищенной аутентификации с помощью использования одноразового кода, основанного на SHA–1 и TOTP (Time – based One – Time Password Algorithm) – алгоритм создания одноразовых паролей для защищенной аутентификации, которые являются фундаментом для разработки одноразовых паролей защищенной аутентификации. Проведен анализ систем защиты информации и их характеристики на основе двухфакторной

аутентификации. Приведены статистические данные компаний, специализирующихся в области обеспечения информационной безопасности. Проведен анализ кибератак и рассмотрены некоторые компании, предоставляющие услуги по защите информации, хранящейся в базах данных. Описаны проблемы, которые возникают при использовании готовых решений аутентификаторов.

**Во втором разделе** представлены результаты, полученные при разработке модели защиты информационной системы при идентификации пользователя на основе двухфакторной аутентификации. Рассмотрены методы аутентификации с использованием одноразового пароля. Разработан и описан алгоритм генерации одноразового пароля с использованием программы аутентификатора и мобильного телефона, который основан на модели генерации одноразового ключа для аутентификации пользователя на основе второго фактора. Разработанная модель основана на использовании комбинации двух факторов: постоянного и временного паролей.

Описана хеш–функция SHA256 [20], которая используется в качестве входного параметра для генерации набора функций и вычисления одноразового пароля. Для формирования хеш–функции SHA256 учитываются такие данные, как логин и пароль пользователя, текущий момент времени/даты и секретная строка. Рассмотрены созданные генераторы случайных секретных слов и тригонометрических функций для формирования одноразового пароля двухфакторной аутентификации.

**В третьем разделе** приведены результаты программной реализации предложенного алгоритма двухфакторной аутентификации. Разработана информационная система, состоящая из 3 взаимодействующих модулей: пользователя, мобильного приложения и серверной части. Рассмотрены структуры каждого из этих модулей.

Описан алгоритм Base64 и схема его использования для защиты информации, хранящейся в базе данных с приведенным программным кодом на языке реализации JavaScript. Рассмотрены стандартные протоколы TLS (Transport Layer Security) и SSL (Secure Socket Layer) для защиты трафика веб–сайтов и обмена файлами по сети.

Описана работа СУБД MongoDB, в которой хранятся и обрабатываются данные. Описан объектно–ориентированный подход, используемый при реализации алгоритма.

Поэтапно изложена структура работы приложения. Осуществлена компьютерная реализация информационной системы защиты информации при аутентификации пользователя на основе одноразового ключа и проведено исследование корректности выполнения предложенного алгоритма.

В заключении изложены основные результаты и выводы диссертации. Результаты исследования включены в отчеты указанных выше проектов ПЦФ за 2018, 2019 годы и ГФ за 2020 годы, выполняемые в Лаборатории информационной безопасности ИИВТ КН МОН РК.

# 1 АНАЛИЗ ИЗВЕСТНЫХ МЕТОДОВ И СРЕДСТВ ЗАЩИТЫ ИНФОРМАЦИОННЫХ СИСТЕМ НА ОСНОВЕ ДВУХФАКТОРНОЙ АУТЕНТИФИКАЦИИ

Защита информационных систем является актуальной задачей, так как они используются для учета, обработки и хранения данных в таких системах различных компаний. Одним из основных методов защиты информационной системы является идентификация пользователя на основе дополнительного средства защиты при аутентификации.

## 1.1 Принципы построения защиты информации в информационной системе при аутентификации пользователя на основе второго фактора

Современное общество использует информационные ресурсы, которые являются главным инструментом для обмена, передачи и хранения информации. В век глобальной информатизации, почти каждая компания любого направления использует информационные системы для электронного документооборота [21].

Информация – это главный объект, который используется для обработки и хранения сведений. Сведения имеют конкретную цену, следовательно, приобретение данных злоумышленниками дает конкретный доход, ослабив вероятность соперничества с конкурентами. Главной целью злоумышленника является получение данных о предметной деятельности, являющихся конфиденциальными. Обычно злоумышленникам интересна информация, содержащая экономические сведения, персональные сведения сотрудника, клиента, покупателя, собственности объекта, исследование деятельности соперников, методы и средства платежных систем.

Основная цель конкурентов или злоумышленников заключается во внесении изменения в структуру данных, которые циркулируют в предметах секретной заинтересованности. Подобные воздействия могут привести к дезинформации в конкретной области при работе с данными. Наиболее небезопасными считаются ликвидирование собранного массива данных в программных приложениях. В связи с рассмотренными инцидентами огромную значимость приобретает формирование результативных концепций информационной защищенности [22].

Каждая организация, используя информационные ресурсы, проводит меры по защите данных, применяя политику безопасности компании. Защита и безопасность информационной системы заключается в сохранении целостности и конфиденциальности данных, а также мониторинге доступности к информации, для минимизации последствий, которые могут возникнуть в случае модификации или разрушения данных.

В рамках защищенной системы информации нужно учитывать действия, которые предотвратят доступ, обработку или кражу информации, когда к ней осуществляется доступ или она передается по сети [23–25].

Представленные данные подтверждают необходимость проведения мероприятий, связанных с защитой таких информационных ресурсов, как хранилища данных, операционные системы, программное обеспечение и другие компоненты инфраструктуры.

Базы данных являются основным компонентом любой информационной системы, в которых структурируется, хранится и обрабатывается информация.

База данных является структурой, состоящая из поименованных объектов, имеющих высокую структуризацию данных [26]. Базы данных классифицируются и подразделяются по образу хранимой информации (документальные, фактографические), способу при сбережении данных (распределенные, централизованные), системе организации данных (сетевые, иерархические, табличные). В настоящее время данные могут, храниться в объектно – реляционных и облачных хранилищах. Для формирования, руководства, а также для коллективного применения баз данных многочисленными пользователями используется система управления базами данных (СУБД) [27].

Угроза взлома секретных данных, хранящихся в БД, способна послужить причиной к её невозвратной потере, вследствии чего организации имеют все шансы потерять важную информацию о финансах, клиентах и т.д. В связи с этим, необходимо обеспечить защиту информационной системы с использованием аппаратных, программных, процедурных, структурных или организационных методов защиты [28, 29] (рисунок 1.1).


Рисунок 1.1– Методы защиты

Организационные методы защиты используются для ограничения числа лиц, которые получают право доступа к информационной системе (ИС). Эти меры включают организацию режима доступа в ИС, мероприятия по обеспечению надежного хранения носителей информации, регламентируют

технологические схемы автоматизированной обработки защищаемой информации, процесс взаимодействия пользователей с информационной системой, задачи и обязанности обслуживающего персонала ИС и пользователей БД и т.д.

Процедурные методы защиты делают возможным доступ к данным и передачу их только тем пользователям, которые имеют соответствующие полномочия. К указанным методам относится установление различного рода паролей пользователей, присвоение документам грифов секретности, проведение занятий с персоналом с целью повышения уровня ответственности.

Наиболее часто процедурные методы защиты используются на этапах первичной обработки данных, управления процессом функционирования системы и на этапе выдачи информации пользователям.

Структурные методы защиты применяются на этапах проектирования структур БД (канонических и логических). Они призваны обеспечить такую структуризацию данных, при которой распределение данных по группам и логическим записям, а также установление между ними соответствующих взаимосвязей позволяет повысить уровень защищенности хранимых данных. Процедуры анализа и синтеза структур БД и соответствующие им механизмы защиты должны обеспечивать:

– разделение всей хранимой информации БД на общедоступные и индивидуальные (или конфиденциальные) данные;

– идентификацию прав пользователей;

– защиту данных и взаимосвязей между ними.

Аппаратные средства защиты информации представляют собой различные электронные устройства, встраиваемые в состав технических средств вычислительной системы или сопрягаемые с ними с помощью стандартного интерфейса. К аппаратно реализованным методам защиты, предназначенным для контроля обращения к данным, хранящимся в оперативной памяти, относятся блоки защиты памяти, схемы прерываний и др.

В настоящее время разработаны устройства распознавания пользователей, основанные на сравнении аудиограмм их голосов, отпечатков пальцев и других индивидуальных характеристик человека (однозначно его идентифицирующих), с тем, что записано в памяти терминала. Следует отметить, что эти устройства отличаются высокой надежностью исполнения функций идентификации. Криптографическая защита информации БД может быть реализована с помощью специальной аппаратуры шифрования или кодирования.

Программные методы играют важнейшую роль при создании эффективных систем защиты информационных ресурсов баз данных от несанкционированного доступа. Под программными средствами защиты понимаются специальные программы, предназначенные для выполнения функций обеспечения безопасности данных. Программные методы защиты могут быть реализованы путем включения разработанных программ в состав используемых операционных систем и СУБД, либо выделения их в

специальные самостоятельные пакеты программ, которые инициируются перед началом процесса обслуживания запросов пользователей [30].

Аппаратные и программные методы защиты используются в основном на этапах обработки данных. Они обеспечивают обслуживание только пользователей, успешно прошедших идентификацию в системе; доступ к защищаемым объектам в соответствии с установленными правилами и правами; возможность изменения правил, установленных для взаимодействия пользователей с объектами защиты; возможность получения информации о сохранности и безопасности объектов защиты [31].

При разработке современных информационных систем возникает необходимость в создании новых подходов и метода для защиты информации в базах данных. Это связано также и с тем, что постоянное ускорение роста объема данных является неотъемлемым элементом современных реалий – появились «Большие данные», обработка и анализ которых требует новых подходов, инструментов и методов, которые могут существенно отличаться от классических [32]. При этом данные могут быть структурированными, слабоструктурированными и неструктурированными, что не позволяет эффективно управлять ими и обрабатывать традиционным образом. Большие данные, которые служат источниками для анализа, могут содержать конфиденциальную информацию [33]. Нарушение конфиденциальности работы с такими данными может обернуться серьезными проблемами.

Согласно закону Республики Казахстан «О персональных данных и их защите» [2], три основных свойства обеспечивают действия по хранению персональных данных – её конфиденциальность, доступность и целостность.

Первым барьером в инфокоммуникационных системах является парольная аутентификация, которая начала свое развитие с появлением операционных систем. Первым этапом для входа в операционную систему является аутентификация пользователя, позволяющая разграничивать права доступа. Несмотря на свою простоту применения, эта методика защиты обеспечивает безопасностьво многих организациях. Но с развитием процессов информатизации и электронного документооборота стало необходимо усложнять методы и средства защиты. Для этого стала использоваться двухфакторная аутентификация пользователя, основанная на генерации одноразового пароля, который действует определенный промежуток времени.

Вопросами по созданию двухфакторной аутентификации начали заниматься в 1980–х годах, когда компания Security Dynamics Technologies запатентовала «Метод и устройство для точной идентификации личности». К 2000–м годам инфраструктура и производственные возможности были доступны для обеспечения разработки собственных средств двухфакторной аутентификации. В 2005 году пользователи стали больше задумываться о конфиденциальности и целостности своих данных, и использование двухфакторной аутентификации стала набирать обороты. Применение двухфакторной аутентификация и сейчас является актуальным и служит дополнительным барьером защиты информационной системы от

несанкционированного доступа. Двухфакторная аутентификация используется в банковских системах РК, а также в автоматизированных информационных системах. Преимуществом созданной системы двухфакторной аутентификации является открытость кода реализации, так как готовые приложения закрыты и, что конкретно происходит при подключении их к нашим системам неизвестно. При выборе двух разных каналов для аутентификации появляется возможность защиты пользовательских логинов от удаленных атак, цель которых – использование чужих личных или идентификационных данных [34]. 2FA требует не только ввода имени пользователя и пароля, но и использование такой информации, которую знает только автор или информация, которая незамедлительно будет доступна только этому автору. Такие данные могут включать то, что известно вам (например, уникальное имя пользователя и пароль), принадлежит вам (к примеру, смартфон с приложением для подтверждения запроса аутентификации) или то, что является частью пользователя (например, биометрические данные – отпечаток пальца или скан сетчатки). Так, первым фактором может быть пароль, а вторым фактором то, что отправляется через приложение или уведомление на смартфон для подтверждения.

В современном мире используется также более 5 миллиардов мобильных устройств, и применение телефона в качестве средства аутентификации помогает быстро решить задачи по усиленной защите, сокращению дополнительных расходов и задержкам доставки. Проблема утечки информации актуальна во всем мире и применение двухфакторной аутентификации для защиты информации послужит дополнительным барьером для злоумышленников. Методы двухфакторной аутентификации рассматриваются как механизмы усиления стойкости аутентификаторов. Двухфакторная защита достаточно надежный барьер, серьезно усложняющий доступ к чужим данным и в какой–то степени нивелирующий недостатки классической парольной защиты [35].

В диссертации к связкам классического использования логин/пароль применяется дополнительный барьер защиты, именуемый вторым фактором, владение которым необходимо подтвердить, с целью принятия прав доступа к учётным и другим данным [36].

Двухфакторная аутентификация используется для осуществления доступа к информационным системам, аккаунтам в социальных сетях, к почте и другим сервисам. На рисунке 1.2 представлена схема двухфакторной аутентификации, где описаны типы, способы реализации и стойкость.


Рисунок 1.2– Схема двухфакторной аутентификации

Рассмотрим наиболее распространенные типы аутентификации пользователей информационных систем и отметим их достоинства и недостатки [37]:

1. SMS–код. После успешной авторизации пользователя, на его номер телефона поступает SMS с кодом, который действует определенный промежуток времени и вводится в аккаунт системы. Повторный вход возможен, при отправке нового SMS с кодом.

Преимуществом такого метода аутентификации, является доступность, так как идет привязка к телефонному номеру пользователя, и сессия для новой генерации пароля повторяется.

Недостатком является отсутствие сотовой связи, а также подмена номера.

Отказ от использования SMS для обработки второго фактора заключается в небезопасности данного метода. Сотовые сети используют «SignalSystem 7» для взаимодействия между собой. Но в этой системе обнаружены серьезные

уязвимости, позволяющие перехватывать входящие вызовы и SMS абонентов [38].

2. Проверка входа с помощью мобильных приложений. Вход в систему осуществляется на смартфоне с установленным специальным приложением. Смартфон хранит ключ и обеспечивает вход в информационную систему.

Преимущества: нет необходимости вводить пароль, не нужны сотовая связь для получения SMS сообщений и Интернет.

Недостатки: если пройдет перехват частного ключа, вероятна фальсификация личного номера.

3. Аппаратные (физические) токены. Считается наиболее прочным и надежным методом двухфакторной аутентификации – USB ключ. Он имеет свой процессор, который производит генерацию одноразового пароля для аутентификации пользователя при присоединении к компьютеру. Подбор ключа зависит от определенной услуги. Преимущество: целиком независимый прибор не требует смартфона. Недостатки: прибор берется в отдельности; не все устройства обладают возможностью использовать этот способ; на один аккаунт, один токен.

4. Приложения – аутентификаторы. Производится генерация пароля на устройстве с помощью разработанной специальной программы. В период опции user приобретает первичный ключ для генерации одноразового пароля при помощи криптографического алгоритма, с конкретным сроком времени, обычно до 1 минуты. Преимущества: необходим Интернет с целью открытия сессии. Недостатки: вероятность перехвата первичного ключа, тогда правонарушитель может производить генерацию дальнейших паролей.

5. Биометрическая аутентификация. Аутентификация пользователя по его уникальным биометрическим характеристикам, таким как отпечаток пальца, структура сетчатки глаза, черты лица, голос и т.д. При регистрации биометрического аутентификатора происходит считывание и запись образца соответствующей биометрической характеристики пользователя с помощью специального считывающего устройства. Затем программный алгоритм обрабатывает полученный образец, и система сохраняет его в качестве шаблона в базе данных. При последующей аутентификации, когда пользователь предъявляет биометрический идентификатор, система сравнивает предоставленный идентификатор с шаблоном с помощью алгоритма сопоставления. Пользователь признается легитимным и получает доступ только в том случае, если степень схожести предоставленного идентификатора с сохраненным в базе данных шаблоном удовлетворяет установленному пороговому значению.

Преимущества: идентификатор неотделим от человека, его нельзя забыть, потерять, передать. Проверив идентификатор, можно говорить о том, что был идентифицирован именно этот человек. Недостатки: необходимость наличия определенных окружающих условий для проведения биометрической идентификации, могут возникать ситуации, когда биометрические

идентификаторы повреждены или недоступны для считывания, а так же не дешевая стоимость сканеров.

Резервные ключи. Это запасной вариант, используемый в случае потери мобильного телефона, который служил элементом защиты, так как принимал одноразовые коды подтверждения для входа в информационную систему. При работе с двухфакторной аутентификацией производится настройка, позволяющая получать резервные ключи для применения их во внеплановых ситуациях.

Анализ современных информационных систем подтверждает необходимость использования двухфакторной аутентификации, как дополнительного барьера защиты доступности, целостности и конфиденциальности хранимых и обрабатываемых данных в информационной системе. Методы приложения – аутентификаторы и проверка входа с помощью мобильных приложений являются более защищенными, практичными и удобными для программной реализации информационной системы двухфакторной аутентификации. В связи с этим, разработке этих методов посвящены исследования диссертационной работы. Использование этих методов позволит усилить защиту информации, хранящейся в информационной системе.

## 1.2 Алгоритмы и протоколы аутентификации с использованием одноразового пароля

Двухфакторная аутентификация позволяет обеспечивать более высокую степень защиты по сравнению с однофакторной аутентификацией, при которой пользователь предлагает только один фактор, обычно пароль, который используется для контроля доступа к чувствительным системам и данным.

Предлагаемая в диссертации система двухфакторной аутентификации состоит из двух этапов. На первом этапе аутентификации пользователь вводит свой логин и пароль. При успешном прохождении этого этапа необходимо пройти второй этап, на котором рассматривается генерация одноразового пароля на основе программы аутентификации и смартфона [39].

Рассмотрим основные протоколы, которые используются в диссертационном исследовании.

OAuth–сервер от DnC (OAuthSD) – это сервер аутентификации, который реализует протокол OAuth и систему OpenID Connect [40].

OpenID – открытая децентрализованная система, позволяющая использовать единый аккаунт пользователя для аутентификации web приложений не связанных друг с другом. Благодаря централизации аутентификации приложений и пользователей сервер OpenID Connect позволяет полностью контролировать доступ к конфиденциальной информации [40, p.2].

OAuth – протокол авторизации, предоставляющий конкретному сервису или программе полномочия на допуск к пользовательскому ресурсу на другом сервисе или программе. Протокол OAuth дает возможность без указания данных для авторизации использовать стороннюю программу, а также

позволяет выдавать набор прав при настройке. Он базируется на применении веб – технологий, запросов и редиректив HTTP, это дает возможность использования на разных платформах, если имеется глобальная сеть Интернет и браузер. Протокол OAuth имеет общую структуру работы:

– доступ для авторизации;
– применение к ресурсам защиты.

Итогом авторизации считается «токен доступа» – конкретный ключ, позволяющий предоставить доступ к защищенным ресурсам. Протокол HTTPS дает возможность обратиться к ним, указав его в заголовках или в качестве одного из параметров полученного токена доступа.

Виды авторизации представлены на рисунке 1.3.

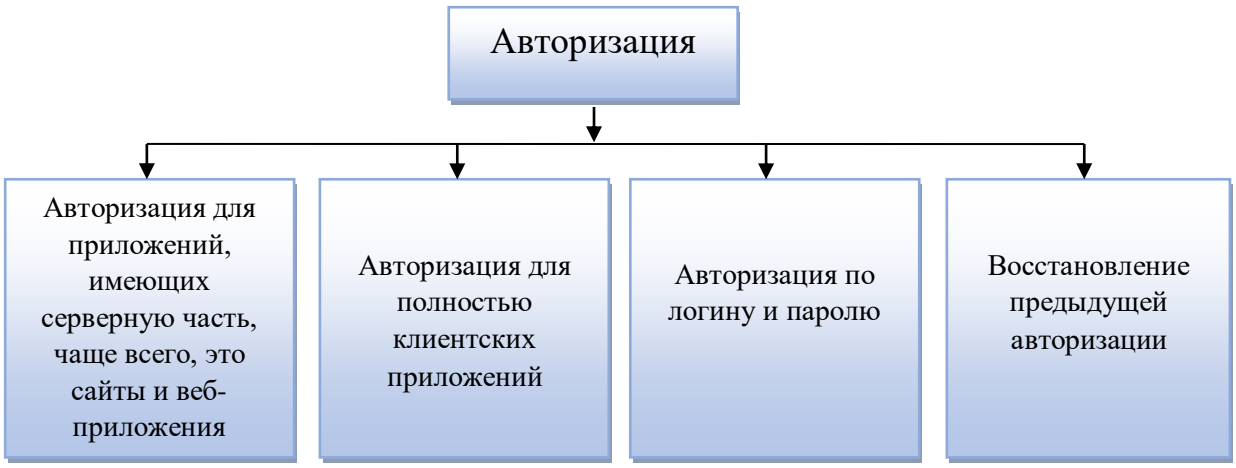


Рисунок 1.3 – Виды авторизации

Общий принцип работы с протоколом OAuth заключается в следующих шагах (рисунок 1.4).

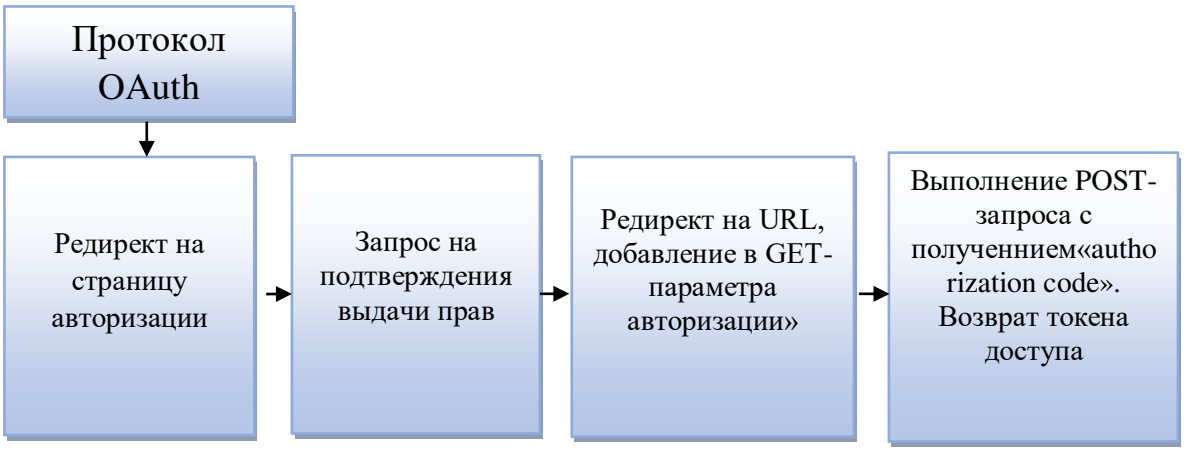


Рисунок 1.4 – Работа с протоколом OAuth

Безопасность OAuth основана на использовании криптографического протокола SSL, который обеспечивает защищенную передачу информации в

Интернете [41]. Защищенность передачи является необходимой в работе высоко нагруженных систем.

Большинство современных систем аутентификации основываются на применение функций вычисления хеш, которые вычисляют хеш на основе входной строки для дальнейшего использования в системе кодирования. В качестве входной строки для хеш–функции могут быть использованы такие данные учетных данных пользователя (логин, пароль), текущее время, дополнительная секретная строка и т.д. Для разработки приложений с использованием функции хеширования в Республики Казахстан используется стандарт СТ РК ИСО/МЭК 10118–1–2006 «Информационная технология. Методы защиты информации. Хеш–функции. Часть 1. Общие положения, Часть 3. Специализированные хеш–функции» [42-43].

Хеш–функция является односторонней функцией, реализуемой средствами симметричного шифрования путем связывания блоков электронного сообщения. Результат шифрования результирующего блока (зависящего от всех предыдущих) является результатом выполнения хеш–функции (или хеширования), т.е. хеш–значением [44].

При идентификации/аутентификации пользователь вводит пароль, а по каналу связи высылается его хеш–значение (далее хеш). Проверяющая система сравнивает этот введенный хеш с хранящимся в информационной системе хешом этого пользователя. В случае совпадения их значений разрешается доступ к системе. В системе не хранятся пароли для повышения ее защищенности, но в этом случае возможен перехват хеш для аутентификации.

Для реализации алгоритма двухфакторной аутентификации используется функция SHA256. Эта функция использовалась для вычисления хеша в виду ее особенностей. Изменение одного бита входной строки влияет на все биты вычисляемого хеша, что позволяет исключить вычисляемую зависимость получаемого хеша от входной строки [45].

Рассмотрим основные алгоритмы аутентификации, которые используются в диссертационном исследовании.

HOTP (HMAC – Based One – Time Password Algorithm) – алгоритм защищенной аутентификации с помощью использования одноразового кода, основанного на SHA–1. Он является алгоритмом односторонней аутентификации, который подразумевает аутентификацию клиента на стороне сервера [46]. Алгоритм впервые формально описан командой IETF (Internet Engineering Task Force), которая занимается развитием глобальной сети Интернет, в декабре 2005 года. Параметром, отвечающим за динамику генерации паролей, является факт генерации или само событие. Каждый раз при создании нового пароля счетчик событий увеличивает значение на единицу. Именно такое монотонное возрастающее значение используется как основной параметр алгоритма. Вторым параметром, отвечающим за генерацию одноразовых паролей, является симметричный ключ, которому необходимо быть уникальным для каждого клиента и в тоже время закрытым от всех, кроме сервера и самого клиента. Системы защиты, построенные с использованием

HOTP, обладают высокой степенью надёжности. HOTP устойчив к широко распространённым криптографическим атакам.

Алгоритм HOTP основан на увеличении значения счетчика и статического симметричного ключа, публичному токену и валидации предоставления услуг. Для того, чтобы создать значение HOTP, используется HMAC Алгоритм SHA–1– алгоритм защищенной аутентификации с помощью использования одноразового кода, это требование, которое установлено в RFC 2104(Request for Comments), документ, содержащий стандарт работы глобальной сети Интернет [47]. Так как значение вычисления HMAC–SHA–1 составляет 160 бит, нужно уменьшить значение, установленное пользователем (формула 1.1).

$$HOTP\ (K, C) = Truncate\ (HMAC\text{-}SHA\text{-}1\ (K, C)), \quad (1.1)$$

где Truncate – функция, которая преобразует HMAC-SHA-1значение в значение HOTP; K – ключ; C – счетчик.

Значения, выдаваемые генератором HOTP, рассматриваются как большой обратный порядок байтов.

Генерация значения HOTP включает 3 этапа, представленные в выражениях (1.2) – (1.5).

1: Генерация значения HMAC-SHA-1:

$$HS = HMAC\text{-}SHA\text{-}1\ (K, C), \quad (1.2)$$

где HS является 20–байтовой строкой.

2: Генерация 4–байтовой строки (динамическое усечение):

$$Sbits = DT\ (HS), \quad (1.3)$$

где DT возвращает 31–битную строку.

3: Вычислить значение HOTP:

$$Snum = StToNum\ (Sbits), \quad (1.4)$$

где Snum – преоразование в число в 0 ... 2 в степени 31 -1

$$Return\ D = Snum\ mod\ 10\ ^\wedge\ Digit\ , \quad (1.5)$$

где D – число в диапазоне 0 ... 10 в степени {Разряд} -1.

Функция усечения выполняется во 2–м и 3–м этапах, то есть производится динамическое усечение и сокращение по модулю 10. Цель этих преобразований состоит в том, чтобы извлечь 4–байтовый динамический двоичный код из 160 – битного (20 – байтового) результата HMAC–SHA–1.

Необходимость маскировки самого значительного бита P состоит в том, чтобы правильно производились вычисления по модулю. Некоторые процессоры могут производить данные операции различно, отмечая подписанный бит, очищая расплывчатость. Шестизначный код считается наименьшим.

TOTP (Time–based One–Time Password Algorithm) – алгоритм создания одноразовых паролей для защищенной аутентификации (2008 г.) [48]. Он также, как и HOTP, является алгоритмом односторонней аутентификации, в котором сервер удостоверяется в подлинности клиента. В отличие от HOTP в нем параметром, отвечающим за динамику генерации паролей, является время. Используется временной интервал с установленными границами, а не конкретное время.

Концепция одноразовых паролей вкупе с современными криптографическими способами применяется с целью осуществления достоверных систем удаленной аутентификации. Алгоритм создания одноразовых ключей является стабильным к криптографическим атакам, предположение взлома есть. Например, возможен такой вариант атаки как «человек посередине», это когда злоумышленник изменяет связь между передающими сторонами, которые считают, что они непосредственно общаются друг с другом.

Реализация TOTP может использовать функции HMAC–SHA–256 или HMAC–SHA–512,основанные на хеш-функциях SHA–256 или SHA–512.

Алгоритм TOTP является вариантом алгоритма HOTP, основанный на представлении счетчика, как фактора времени (формула 1.6).

$$TOTP = HOTP (K, T), \quad (1.6)$$

где T – целое число, которое представляет промежуток времени между начальным значением и текущим системным временем; K – ключ.

Тогда формула 1.7 будет следующая:

$$T = (\text{Текущее время ОС} - T0) / X, \quad (1.7)$$

где T – представляет количество временных шагов X между начальным T0 и текущим временемоперационной системы (ОС).

Например, если текущее время ОС составляет 59 секунд и значение T0 = 0, а временный шаг X = 30 то получим, T = 1. Если текущее время ОС составляет 60 секунд, с теми же параметрами T0 = 0 и временным шагом X = 30, тогда получим T = 2. Реализация данного алгоритма должна поддерживать значение времени T, большего, чем 32–разрядное целое число. Значения системных параметров X и T0 предварительно устанавливаются в течение процесса обеспечения, и передается между проверяющим верификатором, как часть шага инициализации. Поток обеспечения выходит за рамки этого

документа. По умолчанию предлагается шаг = 30 секундам в качестве баланса между безопасностью и удобством использования.

Описанные алгоритмы задают фундамент для разработки одноразовых паролей защищенной аутентификации, и используется для реализации.

### 1.3 Атаки и средства защиты информационных систем

Согласно статистике компании InfoWatch [49], количество утечек данных в мире беспрерывно растет, за 2019 год 1748 случаев утечек конфиденциальной информации – что на 22,5% больше, чем за аналогичный период 2018 г. (рисунок 1.5). Объемы скомпрометированных данных растут из–за увеличения «мощности» внешних и внутренних утечек. При этом утечки по всем каналам, кроме сетевых, зачастую просто не фиксируются.

Группа компаний InfoWatch (Россия) специализируется по защите корпораций от утечек информации и от целевых атак извне, а также контролирует российский рынок систем защиты конфиденциальных данных.

В мире за 2019 год каналом утечки доминирует сеть 70%, а в России 62%.

Из организаций финансового сегмента по типу инцендентов за 2019 год в мире неквалифицированная утечка составила 79,8%, мошенничество с использованием данных 19,3%, превышение прав доступа 0,9%.

Наружные атаки, приведшие к десяти из двадцати зафиксированных «мега–утечек», из них 8,74 миллиардов дискредитированных записей (97% общего количества). В 41 вариантах размер дискредитированных данных превысил 1 миллион записей. В 44,8% утечек виновными оказались работники организаций, в 2% руководители организаций, а также избранные пользователи.

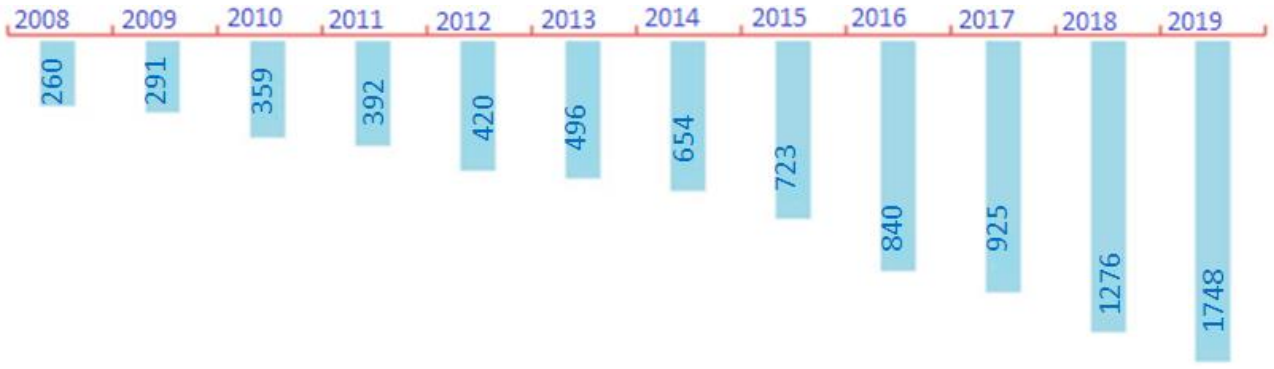


Рисунок 1.5 – Количество утечек информации в 2008 – 2019 гг.

Защита баз данных начинается с тщательного анализа неучтенных уязвимостей при их осуществлении. Нередко фактом потери или взлома имеют все шансы быть неучтенные или не функционирующие на надлежащем уровне стандартные методы обеспечения безопасности информации, хранящейся в базах данных, в связи с тем, что СУБД устанавливается в совокупности с другими приложениями (операционными системами и серверами). Вопросы

обеспечения информационной безопасности в БД привлекают внимание исследователей как в Казахстане, так и за рубежом.

Из отчета компании TAdviser выявление инциндентов информационной безопасности за 2018 – 2019 год: 54% программы – вымогатели, 52% DDOS, 39% целеноправленные атаки, 39% фишинг, 31% мобильное мошенничество, 23% хищение финансов через интернет – банкинг, 16% криптомайнеры [50].

По данным компании PositiveTechnologies в 2019 году для хакеров наиболее привлекательной является информация о персональных данных (60%), учетные записи и парольная информация для доступа к различным сервисам и системам, в том числе и к онлайн – банкам частных лиц [51]. По данным этой же компании выделены распространенные уязвимости: использование устаревших версий программного обеспечения и отсутствие актуальных обновлений безопасности для операционных систем; множественные ошибки конфигурации (в том числе избыточные привилегии пользователей и программного обеспечения, а также установку паролей локальных администраторов через групповые политики); использование словарных паролей привилегированными пользователями; отсутствие двухфакторной аутентификации для доступа к критически важным системам.

Исследования по практической безопасности за 2019 г. показали, что механизмы использование двухфакторной аутентификации в онлайн – банках и веб – приложениях недостаточно надежны, выявлены 77% уязвимостей [52]. Эти данные подтверждают необходимость использования надежной проверки подлинности пользователя на базе двухфакторной аутентификации для повышения уровня безопасности.

По данным исследования инцидентов безопасности Verizon’s Data Breach Investigations Report (DBIR) в 2019 году 95% нарушений включают использование украденных персональных данных [53]. Стандартные процедуры безопасности (особенно онлайн) требуют простого ввода имени пользователя и пароля, и преступники могут с легкостью завладеть персональными данными пользователя – личной и финансовой информацией – с целью дальнейшего ее использования для совершения мошеннических действий, в основном в сфере финансов.

По данным компании «СёрчИнформ» в 2019 году 59% российских компаний столкнулись с утечками информации и 50% страны СНГ (рисунок 1.6) [54]. Как видно из приведенных данных, эта проблема актуальна для всего мира. Большой процент утечки приходится на информацию о клиентах и сделках, техническую информацию, а так же персональные данные. Некоторые компании скрывают подобные инциденты и не делают никаких оповещений об утечках информации в их компаниях.

Таким образом, возникает проблема защиты конфиденциальной информации от атак злоумышленников. Одним из средств защиты информации является парольная защита с использованием второго фактора.

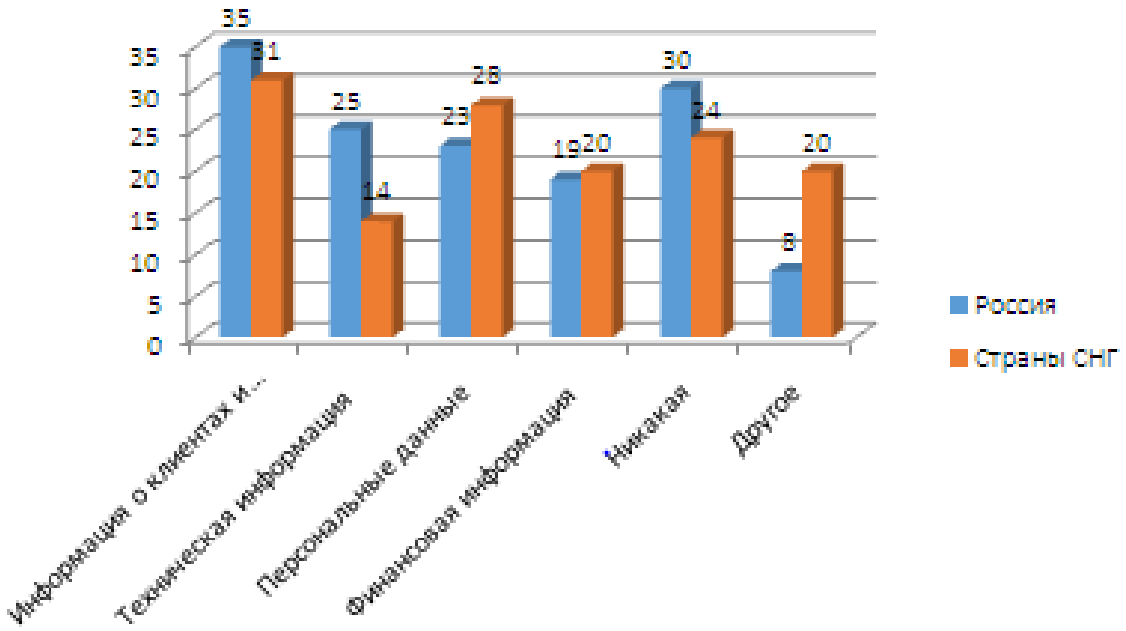


Рисунок 1.6 – Данные об утечках информации за 2019 год

Двухфакторная аутентификация от Infobip решает данную проблему с помощью SMS – сообщения, отправляемого на мобильный телефон, и Голосовых технологий. Использование в качестве второго фактора авторизации SMS – сообщение не является самым безопасным решением. Телефонный номер может быть привязан всего один, а значит и подтверждающее устройство будет одно:

– при нахождении в зоне неуверенного приема SMS – сообщение от сервиса может не прийти;

– номер могут украсть вместе с телефоном или попытаться сделать его дубликат для авторизации вместо хозяина телефона;

– могут быть проблемы с авторизацией в заграничных поездках, например, SMS – сообщение долго доходит, роуминг не работает или вместо родной карты решили использовать местную.

Всех этих неудобств можно избежать при использовании специальных сервисов и приложений, которые могут выступать вторым фактором авторизации [55].

Любая компания, хранящая и обрабатывающая информацию в базах данных (БД), может столкнуться с хищением конфиденциальных сведений.

Для проведения атак на информационные системы злоумышленники используют широкий спектр «технологий» – DDoS атаки, получение несанкционированного доступа путем компрометации учетных записей пользователей, вредоносное программное обеспечение различных типов, взлом службы доменных имен и приложений.

Значительная доля успешных внешних атак на информационные системы была проведена при помощи SQL–инъекций [56]. Проведен анализ кибератак за 2018–2019 год в мире, результаты которого приведены в таблице 1.1 и на рисунке 1.7.

Таблица 1.1 –Мировые кибератаки за 2018–2019 год

| Месяц | Кибер–преступ–ность 2018 | Кибер шпио–наж 2018 | Хакерс–кие атаки 2018 | Кибер–преступ–ность 2019 | Кибер Шпионаж 2019 | Хакерс–кие атаки 2019 |
|---|---|---|---|---|---|---|
| Январь | 70 | 12 | 6 | 94 | 14 | 5 |
| Февраль | 49 | 17 | 6 | 115 | 12 | 1 |
| Март | 56 | 6 | 3 | 75 | 19 | 3 |
| Апрель | 63 | 18 | 3 | 81 | 11 | 3 |
| Май | 51 | 14 | 0 | 95 | 12 | 6 |
| Июнь | 44 | 6 | 4 | 85 | 12 | 1 |
| Июль | 58 | 7 | 1 | 108 | 19 | 3 |
| Август | 70 | 9 | 5 | 66 | 16 | 2 |
| Сентябрь | 65 | 9 | 1 | 93 | 10 | 0 |

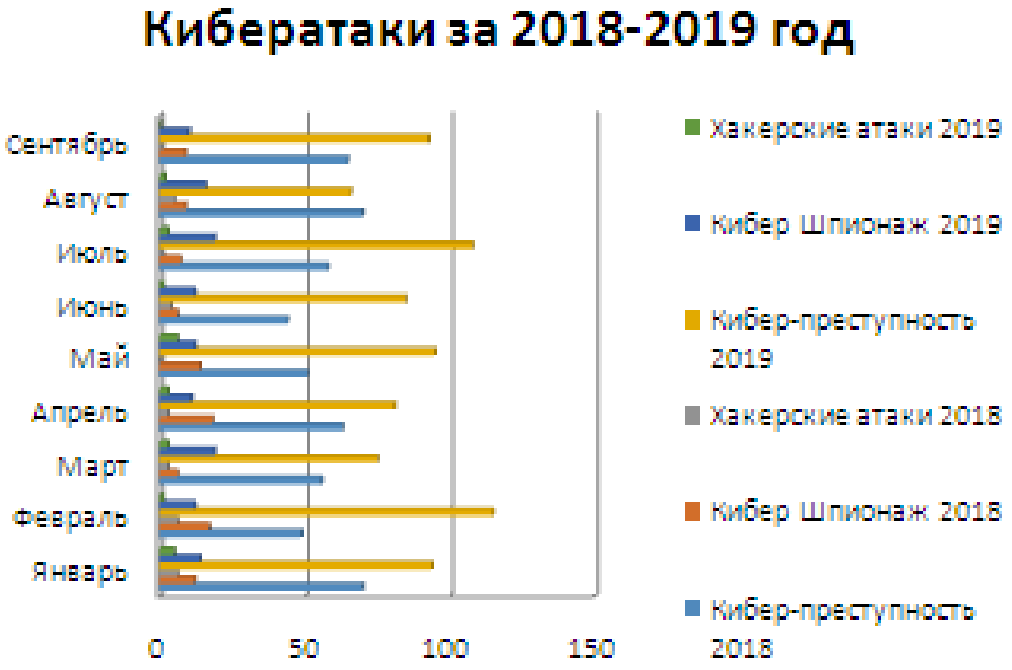


Рисунок 1.7 – Кибератаки за 2018–2019 год

По ОСИ «X» указано количество кибератак, а по ОСИ «Y» число месяцев. Как видно из приведенных данных, эта проблема кибератак является актуальной. Киберпреступность усиливается с каждым годом, в связи, с чем возникает необходимость также увеличении применения средств защиты.

В данной таблице представлены данные за 2018-2019 годы, так как отчет за 2020 год выходит в первой декаде 2021 года.

Рассмотрим атаки на хеш – функции. Существуют различные виды атак на хеш – функции: например, словарная атака – метод вскрытия информации путем перебора, где за основу множества паролей взяты слова, которые загружаются в словари; метод грубой силы – перебор хешируемых значений; пародокс «день рождения»; метод поиска коллизий; метод встречи посередине и т.д.

Отметим также, что хеш – функция в данной работе используется только для определения координат тригонометрической функции и её параметров. При вычислении значения этой функции генерируется одноразовый пароль.

Рассмотрим некоторые компании, которые предоставляют услуги по защите информации, хранящейся в БД.

– IBM SecurityGuardium. Предотвращение утечки информации из баз данных, хранилищ данных и сред больших данных, таких как Hadoop, обеспечивает целостность информации и автоматизирует контроль соответствия в гетерогенных средах. Решение защищает структурированные и неструктурированные данные в базах данных, средах больших данных и файловых системах от угроз и обеспечивает соответствие требованиям. Также предоставляет масштабируемую платформу, которая обеспечивает непрерывный мониторинг структурированного и неструктурированного трафика данных, а также применение политик для доступа к конфиденциальным данным в масштабах всего предприятия [57].

– Компания Imperva. Известная компания, занимающаяся защитой веб приложений и систем управления базами данных. Основная цель работы компании заключается в проведении мониторинга и контроля данных, которая осуществляет проверку потока информации от хранилища на сервере до приложения обработки [58].

– Imperva Database Security Secure Sphere. Основной вид деятельности – организация защиты информации хранящейся в базе данных, цель– обеспечение доступности и целостности работы с данными в информационной системе. Компания предоставляет услуги по оптимизации решений для обеспечения защиты базы данных, за счет проведения анализа и сканирования сетевой активности при помощи агентов приложений [59].

– McAfee DataCenter Security Suite for Databases. Предоставляет возможность получать полную информацию об общем состоянии и уровне защищенности баз данных. Это программное решение для защиты баз данных не требует ни внесения изменений в архитектуру, ни установки дорогостоящего оборудования. В режиме реального времени решение обнаруживает и

блокирует попытки атак и вторжений без необходимости отключать базы данных и тестировать приложения [60].

– McAfee Vulnerability Manager for Databases. Осуществляет автоматическое обнаружение базы данных, имеющейся в вашей сети, определение установок новейших пакетов исправлений, проведение проверки на наличие распространенных уязвимостей [61].

– Trustwave AppDetectivePRO. Сканер хранилищ данных и больших данных, предоставляет возможность обнаружить ошибки в настройках, выявить трудности контроля доступа и идентификации пользователя, отсутствующие патчи и небезопасные композиции опций, могут являться основой для DoS–атак, распределения привилегий пользователя и недопустимому изменению информации. Простота установки и интуитивно понятный интерфейс AppDetectivePRO не требует от пользователя экспертных знаний в области баз данных. Всего за несколько минут можно мгновенно проанализировать состояние безопасности, получить оценку рисков и составить отчет о соответствии требованиям для любых БД и хранилищ BigData – как локальных, так и облачных. AppDetectivePRO является прекрасным дополнением к другим сканерам сетей и приложений [62].

– DbProtect: платформа для обеспечения безопасности данных, позволяющая мгновенно выявлять ошибки в конфигурации, проблемы идентификации и контроля доступа, недостающие патчи и опасные комбинации настроек, способные привести к атакам типа «повышение привилегий» или «отказ в обслуживании» (DoS–атакам), утечкам и несанкционированному изменению данных. За счет использования многопользовательской ролевой модели доступа и распределенной архитектуры с инструментами аналитики и отчетности корпоративного класса DbProtect защищает все реляционные СУБД и хранилища BigData в инфраструктуре компании, развернутые как локально, так и в облаке [63].

– FUDO SECURITY – ведущая польская компания, поставщик инновационных решений в области ИТ–безопасности. Компания специализируется на управлении привилегированным доступом, аутентификации и авторизации пользователей, а так же проверке зашифрованного SSL/TLS трафика.

– FUDO PAM – эффективное управление и контроль привилегированных пользователей. FUDO PAM – передовое решение, доступное в виде физического или виртуального устройства. Запись сессии, проактивный мониторинг на основе искусственного интеллекта, современное хранилище паролей и бизнес-аналитика – все это в одном устройстве, которое устанавливается за пару часов [64].

Все перечисленные компании предоставляют недешёвые услуги, в связи с этим ни каждая компания может себе позволить ихиспользовать. Для решения проблем, связанных с безопасностью данных компании, обычно находят более выгодный и удобный вариант решения проблемы.

В настоящее время практически все наиболее значимые ресурсы и сервисы используют 2FA. На первом этапе аутентификации пользователь вводит свой логин и пароль, при успешном прохождении этого этапа необходимо пройти второй этап. В качестве второго этапа используется OTP аутентификация по средствам SMS или email рассылки, либо с использованием программного генератора паролей, установленного на мобильное устройство.

В связи с этим рассмотрим несколько наиболее популярных аутентификаторов и их возможности. Несмотря на то, что базовая функция у всех приложений одна и та же – создание одноразовых кодов по одному и тому же алгоритму, некоторые аутентификаторы обладают дополнительными функциями или особенностями интерфейса.

1. Google Authenticator. Поддерживаемые платформы: Android, iOS. Google Authenticator является приложением, созданным для аутентификации пользователя на основе второго фактора. Google Authenticator не имеет настроек, что облегчает работу данного приложения [65].

2. Duo Mobile. Поддерживаемые платформы: Android,iOS. Duo Mobile имеет ряд преимуществ: таких как простота в использовании, минималистичен и не требует дополнительных настроек [66]. Для отображения кода необходим токен.

3. Microsoft Authenticator. Поддерживаемые платформы: Android, iOS. Данное приложение имеет возможность производить настройки токена, так что при запуске токен может быть скрыт. Microsoft Authenticator – прост в использовании и является высокофункциональным [67].

4. FreeOTP. Поддерживаемые платформы: Android, iOS. Программное обеспечение с открытым кодом [68]. Размер приложения равен 750 Кбайт для платформы iOS, является минимальным из всех. Приложение Google Authenticator имеет размер равный 14 Мбайт, Authy равен 44 Мбайта. У приложения FreeOTP есть возможность производить настройку токена в ручную, что у других отсутствует.

5. Authy. Работает на платформах Android, iOS, Windows, macOS, Chrome [69]. Это приложение позволяет использовать облачные сервисы для хранения токенов, что является преимуществом. Облачные сервисы позволяют из любых устройств предоставить доступ к токену. Обязательным является то, что приложение использует аккаунт, привязанный к телефонному номеру и без этого работа невозможна.

6.Яндекс.Ключ. Поддерживаемые платформы: Android, iOS. «Яндекс. Ключ» также как и приложение Google Authenticatorне имеет необходимости производить регистрацию при входе [70]. Приложение имеет возможности такие как использование облака Яндекс для хранения резервных токенов.

Основными характеристиками приложений аутентификаторов являются:

1.Независимость – предполагает существование канала связи, которым может являться интернет или оператор сотовой связи;

2.Потребность синхронизации – данное качество подразумевает потребность декодирование с сервером аутентификации. Устройство генерации

одноразовых паролей (OTP) функционирует самостоятельно, однако возможно нарушениево времени, в этом случае полученный пароль не будет достоверным и необходимо сгенерировать его заново;

3.Восстановление автоматизации – позволяет определить доступность возобновления к сервису аутентификационного прибора, не прибегая к помощи IT специалистов, это будет возможно в случае автоматического кода восстановления;

4.Вспомогательный метод аутентификации – демонстрирует, имеется ли услуга безубыточного способа аутентификации, кроме применения приложения на смартфоне. К примеру, не имея мобильный телефон при себе, можно воспользоваться списком одноразовых кодов для входа в сервис определенной информационной системы;

5.Защита приложения паролем – отражает, реализован ли механизм парольной защиты для самого мобильного приложения;

6.Ввод данных с клавиатуры – наличие этого свойства говорит о том, что сервис требует введения каких– либо данных с помощью клавиатурного ввода. Например, для использования какого–либо сервиса необходимо сначала ввести свой логин и пароль, а потом ввести код аутентификации, полученный из мобильного приложения;

7.Применимость к другим системам – этот пункт говорит о том, что одно приложение может использоваться для аутентификации на различных ресурсах. Так же подразумевается возможность добавления новых систем;

8. Необходимость регистрации – если присутствует данное свойства, то сервис, предоставляющий услугу, подразумевает предварительную регистрацию на нем. Другими словами, создание аккаунта с заданием логина и пароля и, возможно, с заполнением других данных;

9. Передача секретов сервису – этот параметр говорит о том, что сервис хранит секретную информацию пользователя, используемую для его аутентификации.

Характеристики приложений двухфакторной аутентификации описанные выше представлены в таблице 1.2.

Таблица 1.2– Характеристики приложений двухфакторной аутентификации

| Аутентификатор | Google Authenticator | Duo Mobile | Microsoft Authenticator | Free OTP | Authy | Яндекс. Ключ |
|---|---|---|---|---|---|---|
| Независимость | + | + | + | + | + | + |
| Потребность синхронизации | + | + | + | + | + | + |
| Восстановление автоматизации | – | + | + | + | – | – |

Продолжение таблицы 1.2

| Аутентификатор | Google Authenticator | Duo Mobile | Microsoft Authenticator | Free OTP | Authy | Яндекс. Ключ |
|---|---|---|---|---|---|---|
| Вспомогательный метод аутентифик. | + | – | – | – | – | + |
| Защита приложения паролем | – | – | – | + | – | + |
| Ввод данных с клавиатуры | + | + | + | + | + | + |
| Применимость | – | – | – | – | – | + |
| Необходимость регистрации | + | + | + | + | + | + |
| Передача секретов сервису | + | + | + | + | + | + |

Рассмотрев готовые решения по двухфакторной аутентификации, можно сделать вывод, что главным недостатком использования готовых решений является использование одного набора генерации для всех подключений, а также вероятность контроля и доступ к информации фирмой разработчиком.

Готовые решения имеют закрытый код и проследить за работой алгоритма невозможно, есть вероятность того, что все данные пользователя могут быть доступны. При подключении готового решения 2FA с привязкой к аккаунтам, нам не дает полной защиты, в связи с этим, необходимо разрабатывать отечественные системы защиты 2FA, в которых код алгоритма будет открыт и его можно просмотреть. Это позволит усилить защиту данных в информационной системе. Большинство компаний, имеющих доступ к конфиденциальной информации, личным данным, особенно, финансовым, стремятся максимально достоверно определять личность своих пользователей.

### 1.4 Выводы по первому разделу

В данном разделе исследованы методы и средства защиты информации в базах данных и двухфакторная аутентификация. Изложены структурные методы защиты баз данных. Рассмотрены методы и типы двухфакторной аутентификации. Описаны достоинства и недостатки каждого из приведенных типов. Рассмотрены некоторые известные протоколы, алгоритмы авторизации и аутентификации. Алгоритм HOTP защищенной аутентификации с помощью использования одноразового кода, основанного на SHA–1. Алгоритм TOTP – алгоритм создания одноразовых паролей для защищенной аутентификации. Рассмотренные алгоритмы послужили основой, для разработки представленного в диссертационной работе алгоритма.

Приведены статистические данные по атакам и средствам защиты информационных систем. Рассмотрены исследования некоторых известных

компаний по утечке конфиденциальной информации и кибератакам. Проведен анализ внешних и внутренних атак, который показал, что большой процент утечки приходится на информацию о клиентах и сделках, техническую информацию, а также персональные данные. Одним из средств защиты информации в информационных системах является парольная защита с использованием второго фактора, которая является актуальной на сегодняшний день, так как весь банковский сектор и компании, занимающиеся вопросами безопасности, используют двухфакторную аутентификацию, как дополнительный метод защиты. Описаны организации, которые предоставляют возможность использования двухфакторной аутентификации на основе одноразовых кодов для защиты информации. Рассмотрены некоторые имеющиеся приложения двухфакторной аутентификации и приведен сравнительный анализ готовых приложений. Выделены основные характеристики приложений, основанные на обзоре средств двухфакторной аутентификации.

# 2 РАЗРАБОТКА АЛГОРИТМА ДВУХФАКТОРНОЙ АУТЕНТИФИКАЦИИ

В данном разделе описан разработанный алгоритм защиты информационной системы при идентификации пользователя на основе двухфакторной аутентификации.

## 2.1 Система аутентификации пользователя на основе генерации одноразового пароля

В ходе исследования были рассмотрены методы аутентификации пользователя на основе генерации одноразового пароля для проведения эффективности работы алгоритмов.

Одноразовые пароли OTP (One – Time Passwords) – генерируются для однократного входа в информационную систему с использованием программных или аппаратных методов защиты [71]. Неуязвимость этих паролей основана на использовании заданного интервала времени. В связи с этим решается проблема перехвата сгенерированного одноразового пароля.

Даже если злоумышленнику удастся перехватить одноразовый пароль, воспользоваться им, в лучшем случае, он сможет лишь в весьма ограниченный промежуток времени, поскольку пароль действителен только один раз в течение короткого промежутка времени.

В качестве возможных устройств для генерации одноразовых паролей используются различные программные методики. В общем случае для применения одноразовых паролей пользователь должен обладать чем–либо (мобильный телефон, список паролей и т.д.), что является аутентификацией «на основе обладания чем–либо».

Для применения аутентификации на основе одноразового пароля используются криптографические алгоритмы для усиления его стойкости. Данные аутентификации являются секретным ключом пользователя, основанным на использовании метода шифрования. Секретный ключ хранится определенное время на сервере и доступен клиенту при его успешной инициализации в информационной системе [72]. По истечению времени функционирования пароля осуществляется его повторная генерация.

Для обеспечения двухфакторной аутентификации в информационных системах одноразовые пароли часто используются в группе с хранимыми паролями. В общем случае данный алгоритм выглядит следующим образом (рисунок 2.1).


Рисунок 2.1 – Схема двухфакторной аутентификации с использованием одноразовых паролей

Использование одноразовых паролей для прохождения двухфакторной аутентификации осуществляется следующим образом:

1. производится ввод учетных данных пользователя (логин/пароль);
2. учетные данные посылаются на сервер аутентификации;
3. производится проверка введенных пользователем данных на сервере аутентификации;
4. на стороне сервера формируется запрос, который генерирует одноразовый код и отправляет его на устройства;
5. одноразовый пароль вводится пользователем и передает его на сервер;
6. сервер производит проверку сходства паролей;
7. проверка удовлетворительная, аутентификация считается успешной.

Основными источниками одноразовых паролей, используемых в информационных системах являются следующие:

1. Программные токены, которые генерируют одноразовые пароли, используя секретный ключ, и текущий момент времени (системное время ОС). Секретные ключи пользователей обычно хранятся на сервере, но при необходимости можно использовать дополнительное хранилище.
2. Коды, которые генерируются случайным образом, отображаются пользователю либо в качестве SMS сообщения или с использованием другого канала связи.
3. Запись заранее сформированных одноразовых паролей или Scratch card. При каждом входе в систему потребуется введение нового сгенерированного одноразового пароля с указанным номером.

Самыми распространенными методами аутентификации являются программные токены, которые более востребованы и не требуют дополнительных затрат.

### 2.2 Разработка алгоритма двухфакторной аутентификации

Разработан алгоритм двухфакторной аутентификации для защиты информации в информационно-коммуникационных системах, позволяющий использовать комбинацию постоянного и временного (одноразового) паролей. Постоянный пароль пользователь выбирает сам. Временный пароль

генерируется по определенным алгоритмам и действует в течение определенного времени.

Для получения одноразового пароля используется дополнительное программное обеспечение. Генерация одноразового пароля возможна в режиме онлайн. В данном режиме дополнительное программное обеспечение посылает запрос на сервер авторизации для генерации временного пароля.

Для получения одноразового пароля происходит его генерация на стороне сервера и представляется пользователю в дополнительном программном обеспечении. Одноразовый пароль может содержать определенный срок действия до одной минуты.

Генерация одноразового пароля выполняется на основе результата выбранной определенной тригонометрической функции, которая будет иметь ряд переменных параметров.

Функции будут объединены в массив значений размерностью 256 на 256, их количество равно 65536. Выбор тригонометрической функции и ее начальных параметров осуществляется на основе результата вычисления хеш–функции SHA256. Также для вычисления тригонометрической функции и ее начальных параметров может быть использован любой тип хеш–функции.

Входной строкой для вычисления хеш–функции является комбинация учетных данных пользователя, текущего момента времени и дополнительной секретной строки.

Полученная хеш–функция разделяется на отдельные числа, которые будут являться индексами для выбора тригонометрической функции и начальными данными [73].

Перед получением одноразового числового пароля необходимо провести единоразовую инициализацию алгоритма с помощью следующих параметров: f(t) и w, где f(t) – функция получения системного времени/даты, которое выдает значение t, а w – сгенерированное секретное слово.

Вычисление одноразового числового пароля производится по формуле:

$$Pas = S\ (L,P), \tag{2.1}$$

где Pas – результат вычисленного числового пароля; S – это функция вычисления одноразового числового пароля на основе идентификации пользователя по введенным параметрам L и P: L – логин пользователя; P – пароль пользователя.

Раскрываем функцию S (L,P) следующим образом:

$$S\ (L,P) = T[i, j\ ]\ (a,b,c,x,y,p1,p2), \tag{2.2}$$

где T[i,j] – это массив функций $t_{i,j}$ для вычисления одноразового числового пароля:

$$T[i,j] = \begin{pmatrix} t_{11} & t_{12} & \dots t_{1j} \\ t_{21} & \dots & \dots \dots \\ \dots & & \\ t_{i1} & \dots & \dots t_{ij} \end{pmatrix}, \quad (2.3)$$

где i,j – позиция функций в массиве T[i,j]; a,b,c,x,y,p1,p2 – параметры функции для вычисления из массива T[i,j]. Позиция функции в массиве T[i,j] и её параметры являются результатом получения определенного числа из сгенерированного хеша.

Затем находим значения параметров функции (2.2):

$$i = K(H, z_1), \quad (2.4)$$
$$j = K(H, z_2), \quad (2.5)$$
$$a = K(H, z_3), \quad (2.6)$$
$$b = K(H, z_4), \quad (2.7)$$
$$c = K(H, z_5), \quad (2.8)$$
$$x = K(H, z_6), \quad (2.9)$$
$$y = K(H, z_7), \quad (2.10)$$
$$p1 = K(H, z_8), \quad (2.11)$$
$$p2 = K(H, z_9), \quad (2.12)$$

где $K(H,z_k)$ – является функцией получения; $z_k$ – значения из хеша H.

Вычисляем хеш из полученных результатов:

$$H=F(L, P, t, w), \quad (2.13)$$

где F (L, P, t, w) – функция вычисления хеша на основе входных параметров L, P, t, w; t – текущее системное время/дата, w – сгенерированное секретное слово.

Предложенный в диссертационной работе алгоритм генерации одноразового пароля с использованием программы аутентификатора и мобильного телефона основан на модели генерации одноразового ключа для аутентификации пользователя на основе второго фактора.

Для реализации описанного алгоритма используется хеш функция SHA256 [20, с.38]. Этот алгоритм хеширования создан в Агентстве Национальной Безопасности США (АНБ, National Security Agency). Хеш–функции превращают некоторый набор элементов в значение фиксированной длины и применяются для аутентификации пользователя [74]. SHA256 применяется для сжатия цифровых данных до длины 2,31 экзабайт.

Алгоритм Меркла – Дамгарда служит базой выполнения семейства хеш–функции SHA–2, задачей которой является деление исходного сообщения произвольной длины на блоки заданной длины. Действия с этими блоками осуществляются последовательно с использование функции сжатия [75].

Рассмотрим пример получения одноразового пароля. Входными данными будут следующие значения:

– логин пользователя: user16;
– пароль пользователя: pass17word;
– текущий момент времени/даты: 21 ноябрь 201915:19:31;
– секретная строка: saLte.

Входная строка для вычисления хеша примет вид:

user16pass17word20191121151931saLte

Результом хеш-функции будет являться следующее выражение:

90CC9939B3C05AA8D36A16B95EE5416B19632332CFCF46B30C4D37DFDE3F7DB7

Для выбора тригонометрической функции используются первые символы результата. Полученные числа «90,CC» необходимо перевести из шестнадцатеричной системы счисления в десятичную систему, полученные значения «144, 204» будет являться индексами для выбора тригонометрической функции из массива размерностью 256x256. Пусть, для примера, по данному индексу будет выбрана следующая функция:

$$\frac{sin^3(x)+\sin(x^2)}{p1/_{p2}}, \tag{2.14}$$

где p1 и p2 –начальные параметры.

В качестве начальных параметров возьмем два шестнадцатеричных числа с конца значения хеш–функции, а в качестве значения x возьмем шестнадцатеричное число с позиции 10. Тогда они примут следующие значения:

p1 = 125 (7D);  p2 = 183 (B7);  x = 60 (3C).

Тогда результат функции будет следующий:

$$\frac{sin^3(60)+\sin(60^2)}{125/_{183}} = -0{,}42515111329414546, \tag{2.15}$$

В качестве одноразового пароля берем числа после запятой, начиная с 5 позиции и длиной 6 цифр. Тогда одноразовый пароль есть число 511132.

Использование рассмотренного алгоритма двухфакторной аутентификации, основано на разработке приложения – аутентификатора и смартфона для генерации одноразового пароля позволит усилить защиту информации в ИС и избежать несанкционированного взлома. Данная система

2FA позволяет обеспечить защиту информации, как от внутренних, так и от внешних вторжений.

Предложенная система двухфакторной аутентификации может применяться в информационной системе компании, как дополнительный барьер защиты при электронном документообороте. Она основана на совместном использовании двух факторов аутентификации, что значительно повышает безопасность использования информации со стороны пользователей, подключающихся к информационным системам по защищенным и незащищенным каналам коммуникаций [76].

### 2.3 Создание генераторов для формирования одноразового пароля двухфакторной аутентификации

На первом этапе создания одноразового пароля использовались входные данные, описанные в главе 2.2, где секретное слово вводилось пользователем с клавиатуры самостоятельно. При исследовании алгоритма возникла необходимость создания генератора секретных слов. Данный метод реализации позволит усилить защиту вводимых данных при генерации одноразового пароля.

Для формирования секретной строки разработан генератор, который позволяет рандомно формировать слова. Словари слов не были использованы, так как слова, хранящиеся в словаре, легче взломать.

Генератор секретных слов основан на использовании латинского алфавита заглавных и прописных символов в общей сумме, равной 52. Длина сгенерированного слова – 5 символов. Генератор реализован на языке JavaScript и имеет следующий вид:

```
getWord
letfuncVariblesList = []
let funcComponents = []
let expressions = ['+', '-', '/', '*']
module.exports.getWord = function() { // генератор слов
  let chars =
'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz'
  let wordLength = getRandomInt(5, 10)
  let word = ''
  for (let i = 0; i <= wordLength; i++) {
    let charIndex = getRandomInt(0, chars.length - 1)
    word += chars[charIndex]
  }
  return word
}
```

Для проведения оптимизации, можно использовать такие методы как [77]:

– полный перебор;
– применение математических моделей;
– аналитическое исследование системы;
– метод комбинаторики;
– использование программных средств.

Для проведения анализа стойкости генератора использовался метод полного перебора. По данному методу учитывается длина строки и скорость перебора в секунду. На скорость перебора влияют характеристики программного обеспечения и оборудования. К характеристикам программного обеспечения относятся антивирусные программы, защита экранов, операционные системы и т.д. К характеристикам оборудования относятся, пропускная способность шины данных, скорость работы накопителей информации, пропускная способность сети, наличие виртуализации и т.д. Данные характеристики существенно влияют на скорость работы алгоритма и не могут превышать скорости перебора 1000000 в секунду. 64 ядерный процессор может перебрать 1000000 слов в сукунду.

В разработанном алгоритме аутентификации пользователя на основе второго фактора, генератор секретных слов работает по описанному выше алгоритму и генерирует слова, состоящие из 5 символов. Скорость перебора была взята 100000 слов в секунду, которую может перебрать 4 ядерный процессор, что для использования обработки на персональном компьютере является оптимальным [78].

Количество вариантов рассчитывается по формуле (2.16), а время поиска рассчитывается по формуле (2.17):

$$S = A^n \tag{2.16}$$

где S– количество вариантов; A – количество символов; n– длина строки.

$$ST = S/P \tag{2.17}$$

где ST– время поиска; S– количество вариантов; P– скорость поиска.

Рассчитаем время перебора:

$S = 52^5 = 380204032$,

ST= 380204032/100000= 3802, 04032 секунд /60 = 63 минуты.

Пример анализа генератора представлен в таблице 2.1.

Стойкостью определяется мера оценки времени, которая используется для подбора (распознавания) пароля, при этом учитывается количество проделанных попыток для того, чтобы угадать пароль. Также стойкостью определяется зависимость функции от длины пароля в символах, его случайность и непредсказуемость.

Таблица 2.1 – Параметры генератора секретной строки

| Количество знаков | Количество вариантов | Длина | Время перебора |
|---|---|---|---|
| 1 | 52 | 5 бит | Меньше секунды |
| 2 | 2704 | 10бит | Меньше секунды |
| 3 | 140608 | 15 бит | 1 секунда |
| 4 | 7311616 | 21бит | 1 минута 22 секунды |
| 5 | 380204032 | 26 бит | 63 минуты |
| 6 | 19770609664 | 31бит | 55 часов |

Случайные пароли формируются путем подбора случайным способом любого количества символов и типа шрифта с возможным использованием регистров текста. Тогда выбор любого символа из некоторого набора символов равновероятен. Надежность случайного пароля находится в зависимости от энтропии применяемого генератора случайных чисел, при этом часто применяются генераторы псевдослучайных чисел. Применение ограниченной энтропии используется в библиотеках программных продуктов, которые применяются для генерации случайных чисел, так же могут быть применимы стандартные словари. Приложения для создания одноразовых паролей имеют в результате сгенерированный пароль, состоящий из определенной длины символов [79]. Для определения сложности случайного пароля, измеренная в терминах информационной энтропии, вычисляется по формуле 2.18:

$$H = \log_2 N^L = L \log_2 N = L\ (\log N / \log 2) \qquad (2.18)$$

где H – результат, измеряется в битах, N – это количество возможных символов; L – количество символов в пароле.

В разработанном алгоритме аутентификации пользователей информационно-коммуникационных систем, основанным на генерации одноразового пароля, создан генератор секретных строк, который является одним из входных параметров. Данный генератор секретных слов использует 52 символа из 26 прописных и 26 – заглавных латинских букв, рассчитан в соответствии с формулой 2.18.

$$H = L\ (\log N / \log 2) = 5(\log 52 / \log 2) = 5{,}7004 \text{ бит}$$

Из таблицы 2.1 видно, что 5 символов достаточно для генерации секретного слова, поскольку секретное слово является всего лишь одним из параметров для генерации хеша. В связи с тем, что в соответствии с разработанным алгоритмом двухфакторной аутентификации генерация одноразового пароля происходит каждые 20 секунд, вероятность взлома сгенерированного секретного слова очень мала. Этим подтверждается эффективность работы предлагаемого генератора.

Рассмотрим также влияние длины пароля на время его вскрытия равному 1000000 слов в секунду. Если использовать метод полного перебора для проверки стойкости алгоритма, то с увеличением числа символов в пароле увеличивается количество вариантов перебора. За основу возьмем 52 символа, а скорость 1000000 слов в секунду (это характеристики выше, чем у обычно используемых персональных компьютеров, например в ИИВТ). Результаты приведенных изменений показаны в таблице 2.2.

Таблица 2.2 –Таблица зависимости пароля от времени вскрытия

| Длина пароля в символах | Время вскрытия в секундах |
|---|---|
| 1 | 0,000052 |
| 2 | 0,002704 |
| 3 | 0,140608 |
| 4 | 7,311616 |
| 5 | 380,204032 |
| 6 | 19770,609664 |
| 7 | 1028071,702528 |
| 8 | 53459728,531456 |
| 9 | 2779905883,63571 |
| 10 | 144555105949,057 |

График построения на основе результатов изменения по времени представлен на рисунке 2.2.


Рисунок 2.2 – Результат зависимость длины пароля (по вертикали) от времени вскрытия в секундах

По ОСИ «X», указано время вскрытия в минутах, а по ОСИ «Y» представлены значения длины пароля в символах.

Далее проводится генерация тригонометрических функций. При генерации должны отсутствовать закономерности или предсказуемости в событиях. Согласно теории Рамсея идеальная случайность невозможна, особенно для больших структур. Профессор Теодор Моцкин отметил, что «хотя беспорядок в целом более вероятен, полный беспорядок невозможен» [80].

Случайная последовательность событий, символов или шагов часто не имеет порядка и не следует понятному шаблону или комбинации.

В математике определением случайности являются теория вероятности и статистики. В статистике случайная величина является присвоением числового значения каждому возможному результату пространства событий. Данное событие облегчает идентификацию и расчет вероятностей. Случайные переменные могут возникать в случайных последовательностях. Случайным процессом является некоторая последовательность случайных величин, описываемая распределением вероятностей.

Случайный выбор, когда он тесно связан с простой случайной выборкой, представляет собой метод выбора предметов из совокупности, где вероятность выбора конкретного предмета является долей этих предметов в совокупности. В ситуациях, когда набор состоит из различимых элементов, механизм случайного выбора требует равных вероятностей для любого элемента, который будет выбран [81].

Существует три механизма, ответственных за случайное поведение в системах:

– случайность, исходящая из окружающей среды;

– случайность, исходящая из начальных условий. Данный подход исследуется с использоанием концепциий беспорядка и также прослеживается в концепциях, действия которых весьма заметны к незначительным переменам первоначальных обстоятельств;

– случайность, внутренне порожденная системой. Случайность, внутренне порожденная системой используется в генераторах случайных чисел. Для генерации случайных чисел разработаны алгоритмы, которые позволяют определить поведение ситемы, используя её начальное состояние. Данные методы работают быстрее, чем получение «истинной» случайности из окружающей среды;

Это привело к созданию разных способов генерации случайных данных, называемых псевдослучайными. Данные способы различаются тем, насколько они непредсказуемы или статистически случайны, и как много времени затрачивают на генерирацию случайных чисел.

До появления генераторов псевдослучайных чисел много времени затрачивалось для формирования большого количества случайных чисел [82]. Результаты сгенерированных чисел сохранялись в таблицах.

На рисунке 2.3 представлено распределение на плоскости массива тригонометрических функций. Для получения этих данных в диссертационной

работе применяется генератор, основанный на рандомной генерации и выборе тригонометрической функции из сформированного массива.

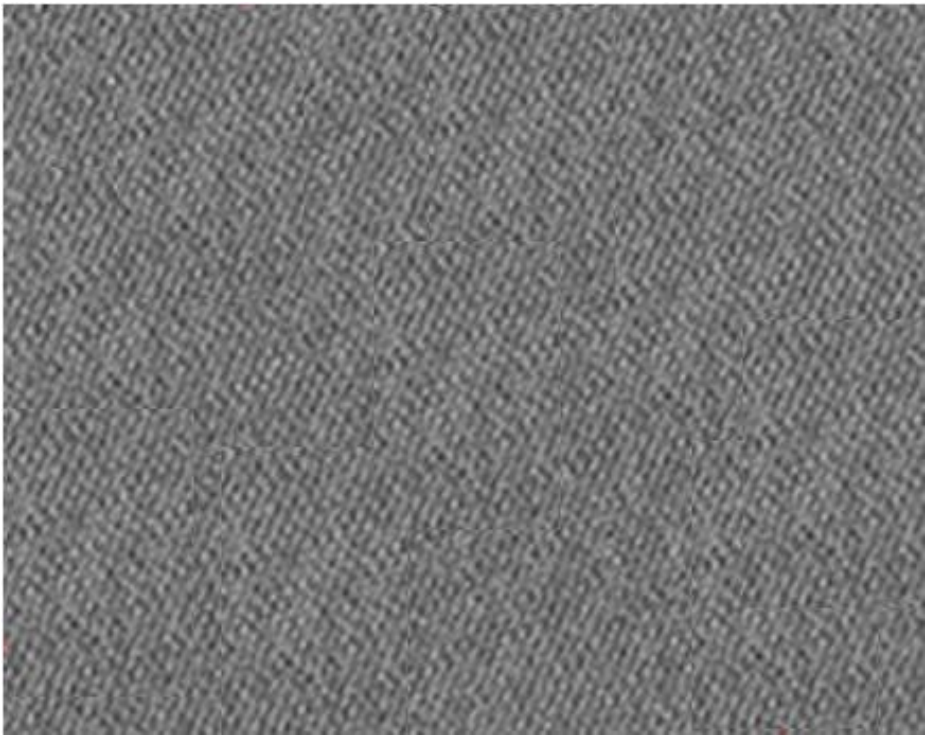

Рисунок 2.3 – Распределение массива тригонометрических функций

На рисунке 2.3 показано равномерное распределение тригонометрических функций и параметров на плоскости.

Как указано выше, генерация одноразового пароля осуществляется на основе результата выбранной тригонометрической функции, которая имеет ряд переменных параметров. Выбор осуществляется в соответствии с результатом полученной хеш–функции стандартов SHA256, где используются первые символы, которые будут являться индексами в таблице размерностью 256x256. По данному индексу будет выбрана функция и определены её параметры. По итогам вычисления в качестве одноразового временного пароля берутся цифры после запятой, начиная с 5 – й позиции и длиною в 6 цифр. Полученное число и будет временным паролем, которое необходимо ввести в приложение. Для реализации данного метода разработан генератор тригонометрических функций, использование которого значительно облегчит формирование этих функций [83].

Схема алгоритма генератора тригонометрической функции представлена на рисунке 2.4–2.6. Для генерации тригонометрической функции за основу берется количество переменных. В данном генераторе их 7: a, b, c, x, y, p1, p2. Изначально формируется список переменных, в результате чего получаем случайное число variablesCount от 1 до количества переменных минус 1. Затем проходит перебор по массиву с определенными переменными N–раз, основываясь на случайном числе от 0 до длины массива минус 1. Считывается

из массива переменная, которая добавляется в новый массив и удаляется из старого. После завершения цикла формируется список переменных для функции. Основываясь на данном списке, формируются составные части (Math.sin(a), 1/Math.tan(p2)) формата – “['Math.sin()', 'Math.cos()', 'Math.tan()', '(1/Math.tan())', '()']”. Функция ComponentsCount (количество элементов минус 1) запускает цикл по массиву со сформированными переменными. В цикле на каждом шаге формируется случайное число componentIndex от 0 до componentsCount. Элемент с соответствующим значением componentIndex индекса преобразуется, заменив символ на переменную из списка и добавляется в новый массив. В итоге формируется список составных частей с переменными.

Дальше формируются ряды, основываясь на случайном числе от 1 до 3. В цикле случайным образом сливаются составные части, разделенные знаками математических выражений. После получения строк происходит их объединение, разделенное математическими выражениями.

В результате работы генератора получаем строчную функцию на языке JavaScript, которая будет обрабатываться на стороне сервера.

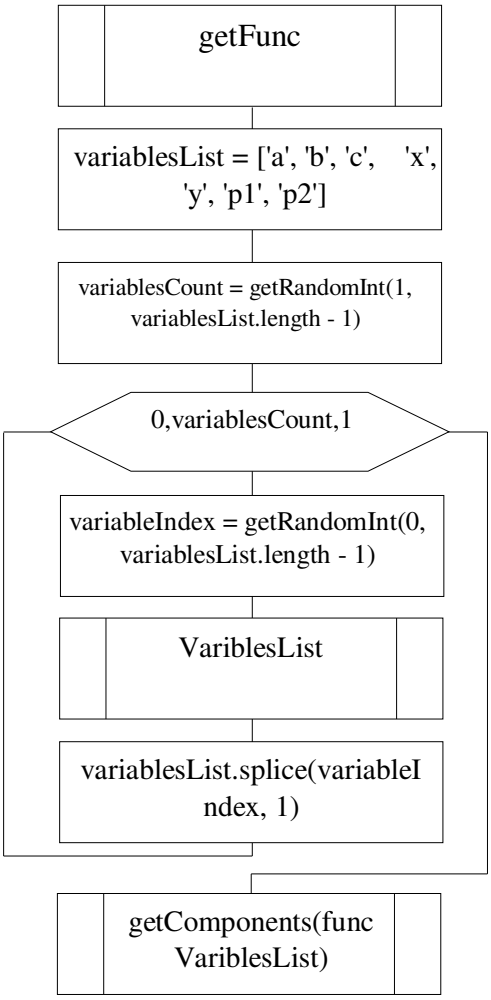


Рисунок 2.4 – Алгоритм функции getFunc

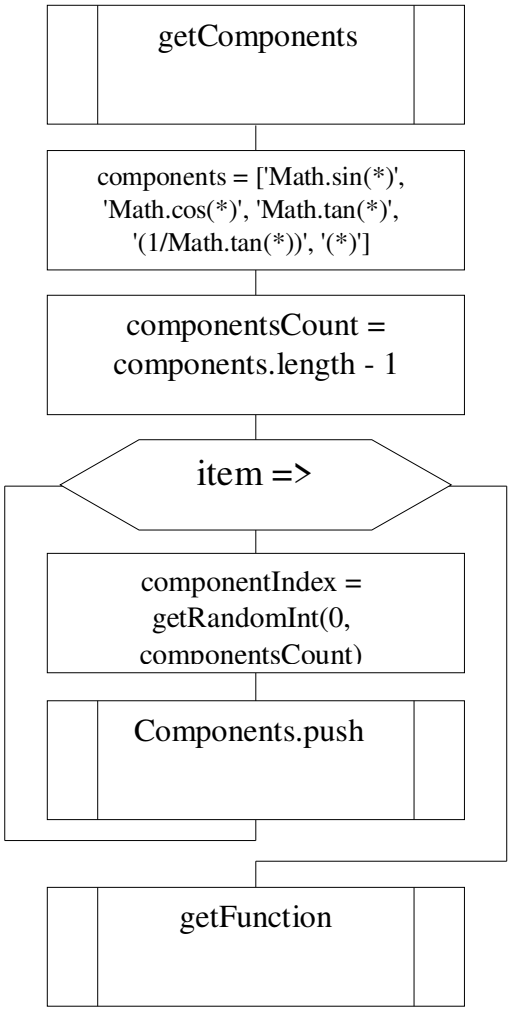


Рисунок 2.5 – Алгоритм функции getComponents

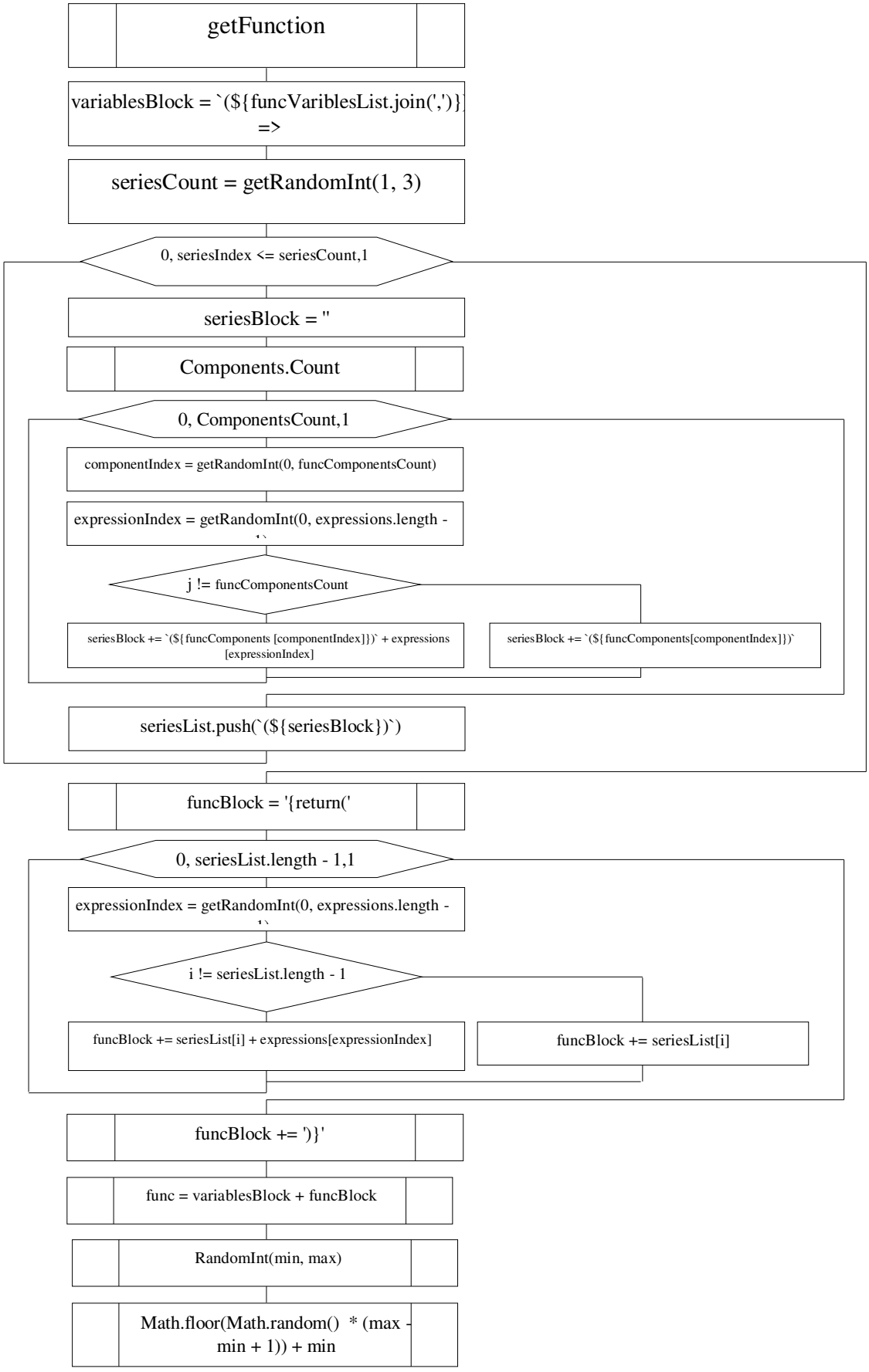


Рисунок 2.6 – Алгоритм функции getFunction

Программа для  реализации генератора имеет следующий вид:

```
module.exports.getFunc = function() { // генератор функций
funcVaribles List = []
  let variablesList = ['a', 'b', 'c', 'x', 'y', 'p1', 'p2']
  let variablesCount = getRandomInt(1, variablesList.length - 1)
  for (let i = 0; i <= variablesCount; i++) {
    let variableIndex = getRandomInt(0, variablesList.length - 1)
    funcVariblesList.push(variablesList[variableIndex])
    variablesList.splice(variableIndex, 1)
  }
  return getComponents(funcVariblesList)
}
function getComponents(variablesList) {
  funcComponents = []
  let components = ['Math.sin(*)', 'Math.cos(*)', 'Math.tan(*)', '(1/Math.tan(*))',
'(*)']
  let componentsCount = components.length - 1
  variablesList.forEach(item => {
    let componentIndex = getRandomInt(0, componentsCount)
    funcComponents.push(components[componentIndex].replace('*', item))
  })
  return getFunction()
}
function getFunction() {
  // формирование блока переменных
  let variablesBlock = `(${funcVariblesList.join(',')}) => `
  // формирование тела функции
  let seriesCount = getRandomInt(1, 3)
  let seriesList = []

for (let seriesIndex = 0; seriesIndex <= seriesCount; seriesIndex++) { // будет 2
ряда
    let seriesBlock = ''
    let funcComponentsCount = funcComponents.length - 1
    for (let j = 0; j <= funcComponentsCount; j++) {
      let componentIndex = getRandomInt(0, funcComponentsCount)
      let expressionIndex = getRandomInt(0, expressions.length - 1)
      if (j != funcComponentsCount) {
        seriesBlock += `(${funcComponents[componentIndex]})` +
expressions[expressionIndex]
      } else {
        seriesBlock += `(${funcComponents[componentIndex]})`
      }
```

```
    }
    seriesList.push(`(${seriesBlock})`)
  }
  let funcBlock = '{return('
  for (let i = 0; i<= seriesList.length - 1; i++) {
    let expressionIndex = getRandomInt(0, expressions.length - 1)
    if (i != seriesList.length - 1) {
      funcBlock += seriesList[i] + expressions[expressionIndex]
    } else {
      funcBlock += seriesList[i]
    }
  }
  funcBlock += ')}'
  // строчныйрезультат
  return func = variablesBlock + funcBlock
}
FunctiongetRandomInt(min, max) { // функция для получения рандомного
числа по заданному диапазону
  return Math.floor(Math.random() * (max - min + 1)) + min;
  }
```

В результате получаем генерируемую строчную функцию, которая используется для вычисления одноразового пароля двухфакторной аутентификации.

Применения генераторов для работы в формировании одноразового пароля позволяет усилить уровень защиты описываемой системы. Не все случайности созданы одинаково. Есть два вида случайности: единообразие и непредсказуемость. Генератор случайных чисел обеспечивает «равномерный» вывод, если все числа будут появляться одинаково часто. Это полезно для моделирования случайных процессов, но недостаточно для безопасности.

Для компьютерной безопасности случайные числа должны быть трудно угадываемые: они должны быть непредсказуемыми. Предсказуемость чисел определяется количественно в мере, названной энтропией.

Энтропия отличается от статистической случайности. Просмотр статистических свойств потока чисел не гарантирует, что поток содержит какую–либо энтропию. Например, цифры числа пи выглядят случайными практически по любой статистической мере, но не содержат энтропии, поскольку существует хорошо известная формула для их вычисления и точного прогнозирования следующего значения. Для криптографических ключей количество энтропии, используемой для их создания, связано с тем, насколько трудно их угадать. 128–битный ключ, созданный из источника с 20–битной энтропией, не более безопасен, чем 20–битный ключ. Хороший источник энтропии необходим для создания безопасных ключей [84].

## 2.4 Разработка модели аутентификации пользователя на основе второго фактора

При разработке системы защиты информации предлагается модель двухфакторной аутентификации (рисунок 2.7).

Разработанная модель основана на двух типах двухфакторной аутентификации: приложение – аутентификатор и проверка входа с помощью мобильных приложений.

Описанная модель и алгоритм предложенного способа защиты информации в информационной системе управления основаны на использовании комбинации двух факторов: постоянного и временного пароля. Постоянный пароль (первый фактор) пользователь выбирает сам и использует при регистрации аккаунта (учетная запись).

Перед авторизацией необходимо пройти регистрацию в приложении. После этого необходимо запустить приложение для ввода данных пользователя (логина и пароля), которые должны соответствовать зарегистрированным данным. При успешном вводе данных необходимо войти в приложение на мобильном устройстве и ввести начальные данные для генерации временного пароля.

Одноразовые пароли, используемые как дополнительный фактор, генерируются на сервере по описанному выше алгоритму и действуют на протяжении фиксированного отрезка времени, равного двадцати секундам для одного сеанса аутентификации. Сгенерированный пароль повторно не используется, что является барьером для злоумышленника, перехватившего его [85, 86]. Отметим, что длина пароля равна 6 символам, и он изменяется каждые 20 секунд. В связи с этим для его взлома необходимо будет перебрать 1000000 паролей, что за указанный промежуток времени, является сложной задачей.


Рисунок 2.7 – Модель предложенной двухфакторной аутентификации

Рассмотренный в данной диссертационной работе алгоритм двухфакторной аутентификации на основе генерации одноразового ключа является результатам исследования по анализу возможности применения в информационной системе.

### 2.5 Выводы по второму разделу

В данном разделе рассмотрены полученные результаты по разработке модели и алгоритма защиты информационной системы при помощи двухфакторной аутентификации на основе генерации одноразового пароля.

Описана предложенная математическая модель системы защиты информации при аутентификации пользователя на основе генерации одноразового пароля.

Разработан алгоритм для усиления средств защиты информации в информационной системе. Алгоритм основан на выборе определенной тригонометрической функции и ряда переменных параметров. Выбор параметров основан на результате хеш–функции SHA256. В качестве входной

строки для хеш–функции используются учетные данные пользователя, текущий момент времени по Гринвичу и дополнительная секретная строка. Результат хеш–функции являются индексами для выбора тригонометрической функции и начальных данных из массива 256х256. Приведен и подробно описан пример выполнения алгоритма с подробным описанием.

Проведен анализ стойкости разработанного генератора секретных слов с использованием метода полного перебора, а также генератора тригонометрических функций, основанный на рандомном формировании массива. Из результата вычисления функции формируется временной код, который отображается только в мобильном приложении пользователя, привязанном к логину on–line системы. Для успешной авторизации в системе пользователю необходимо ввести полученный код после попытки авторизации с корректными данными по запросу.

Разработанная модель и алгоритм позволяет усилить средства защиты информационной системы на основе двухфакторной аутентификации.

Возможность дальнейшего развития предложенной системы двухфакторной аутентификации заключается в расширении компонентов (например, используется уже такая терминология, как «Композитная аутентификация»), применении отечественных криптографических алгоритмов, разработке дополнительных инструментов и оптимизации решений с целью повышения производительности.

# 3 РЕАЛИЗАЦИЯ И АНАЛИЗ ЗАЩИТЫ ИНФОРМАЦИОННОЙ СИСТЕМЫ ПРИ ПОМОЩИ ДВУХФАКТОРНОЙ АУТЕНТИФИКАЦИИ

## 3.1 Разработка архитектуры системы аутентификации

Для программной реализации предложенного алгоритма разработана диаграмма развертывания, состоящая из трех компонент:

– сервер – база данных;
– клиент – мобильное приложение;
– генератор ключей.

Архитектура информационной системы представлена на рисунке 3.1.

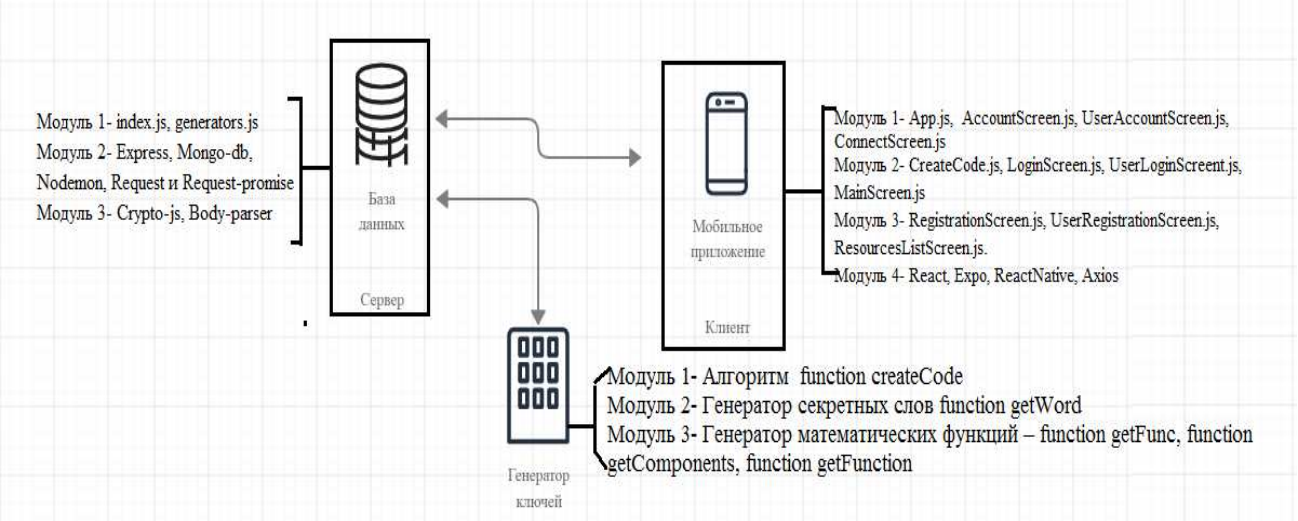


Рисунок 3.1 – Диаграмма развертывания

Пользователь при входе в информационную систему должен пройти регистрацию для осуществления дальнейшего взаимодействия с сервисом.

При регистрации пользователю необходимо создать логин и пароль, пароль желательно создать сильный, следуя рекомендуемым стандартам.

При входе в информационную систему пользователь авторизуется, вводя свои учетные данные и одноразовый пароль. Для получения одноразового пароля пользователь должен иметь мобильный телефон и установленное на него приложение [87].

Мобильное приложение – клиентская часть защиты информационной системы с использованием двухфакторной аутентификации. Оно позволяет пользователю взаимодействовать с системой путем регистрации собственных информационных ресурсов или выбора одного из доступных и генерировать одноразовый временный пароль для входа [88].

Мобильное приложение состоит из следующих основных файлов:

– App.js – начальный файл, в котором происходит настройка маршрутизации приложения.

– AccountScreen.js– страница авторизованного профиля подключенного информационного ресурса. Из этой страницы открывается доступ к просмотру

API серверной части разработанной системы. API представляет собой систему запросов, на которые сервер ответит корректно.

– UserAccountScreen.js – страница авторизированного аккаунта пользователя, из которой открывается доступ к выбору подключенных ресурсов и генерации секретного кода.

– ConnectScreen.js – страница подключения информационного ресурса к своему аккаунту.

– CreateCode.js – страница для осуществления генерации временного пароля и его отображения пользователю.

– LoginScreen.js – страница авторизации профиля подключенного информационного ресурса.

– UserLoginScreent.js – страница авторизации пользовательского аккаунта.

– MainScreen.js – начальная страница с навигационными кнопками.

– RegistrationScreen.js – страница для осуществления регистрации профиля и подключения информационного ресурса.

– UserRegistrationScreen.js – страница регистрации пользовательского аккаунта.

– ResourcesListScreen.js – страница, содержащая список доступных информационных ресурсов, в которых осуществлена защищенная авторизация с помощью разработанной системы двухфакторной аутентификации. Из этой страницы при выборе одного из ресурсов открывается доступ к странице генерации одноразового пароля.

Фреймворки и библиотеки, используемые для реализации мобильного приложения:

– React – фреймворк для разработки клиентской части web приложений.

– Expo – клиентская оболочка, позволяющая проводить компиляцию кода написанного на ReactNative в полноценном мобильном приложении. Также используется для отладки программы во время разработки.

– ReactNative– фреймворк для разработки мобильного приложения, основываясь на JavaScript коде с использованием фреймворка React.

– Axios – подход к построению интерактивных пользовательских интерфейсов веб-приложений, заключающийся в «фоновом» обмене данными браузера с веб–сервером.

Серверная часть – основополагающий модуль в работе проекта, в котором реализован алгоритм генерации временного пароля для осуществления защищенного входа в подключенные информационные ресурсы.

Для безопасного хранения информации используются криптографические алгоритмы защиты баз данных и стандартные средства защиты, такие как подключение к базе данных с помощью логина и пароля администратора базы данных [89].

В диссертационной работе для защиты информации используется механизм кодирования Base64. Это механизм кодирования, используемый для

представления и потоковой передачи двоичных данных через носители, ограниченные только печатными символами [90]. Основная идея технологии кодирования Base64 состоит в том, чтобы взять три символа, каждый из которых представлен в 8 битах, и превратить их в четыре символа, каждый из которых представлен в 6 битах. Результатом является три символа в ASCII.

Каждый символ отображается в 8–битное число от 0 до 255 на основе таблицы ASCII: представление трех символов из 8 битов каждый и их соединение для получения 24 бит. Следующим шагом является разделение 24 бит на четыре равные части по 6 бит в каждой части и перевод каждой части с использованием таблицы Base64. Каждые 6 бит имеют 64 варианта представления из символов, доступны следующие символы: цифры, строчные и прописные буквы, а также символы «+» и «/». Кодировка Base64 разбивает входной текст на части по три символа и кодирует эти три символа.

В конце процесса может возникнуть проблема в том, что не хватает одного или двух символов для завершения последнего трио. Для решения этой проблемы при кодировании добавляют один или два символа «0» в конце, чтобы создать последнюю 3–байтовую группу. Затем кодировка Base64 преобразует последние символы в '='. В итоге получим кодированный в Base64 текст, который заканчивается одним или двумя символами '='.

Независимо от того, какая строка закодирована, после многократного кодирования в Base64 всегда получаем один и тот же фиксированный префикс, который начинается с «Vm0wd». Причина этого явления заключается в том, как работает кодировка, т.е. как буква «V» ведет себя при кодировании. Кодируется буква «V» с помощью Base64. В ASCII буква «V» равна 86, которое в 8–битном представлении имеет вид 01010110. После кодирования и игнорирования заполнения, поскольку нужен только префикс, будут использованы только первые 6 бит представления, что означает 010101. В Base64 это 21, что также является «V». В результате получается, что каждый раз при кодировании всего, что начинается с буквы «V», получим закодированную строку, которая также начинается с «V», что означает, цикл бесконечен.

Результат перекодировки Base64 для каждого ASCII – читаемого символа и цифры представлен на графике рисунка 3.2. Каждый цвет представляет расстояние кодирования до буквы V: синий – четыре итерации кодирования; зеленый – три итерации кодирования; желтый – две итерации кодирования; оранжевый – одна итерация кодирования.

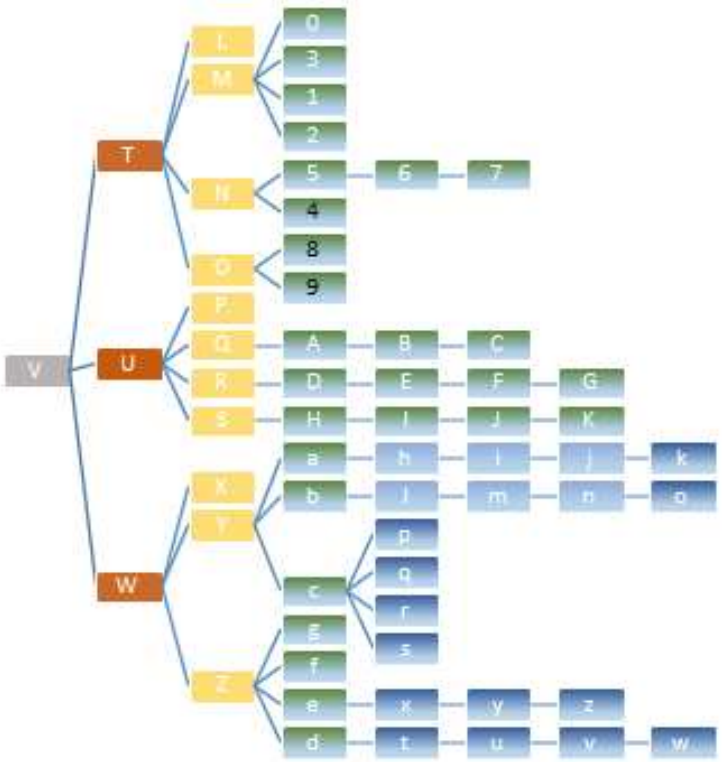

Рисунок 3.2 – График результата перекодировки

После кодирования первых трех букв фиксированного префикса остается остаток в 6 бит. Эти 6 бит будут определять следующую букву префикса. Фактически, для каждых трех букв, добавляемых к фиксированному префиксу, после кодирования остаются дополнительные 6 битов, которые будут определять дополнительный символ префикса. Это означает, что фиксированный префикс будет увеличиваться в каждой дополнительной кодировке на количество букв в префиксе, деленное на три.

Например, если в фиксированном префиксе девять символов, то после другой кодировки в фиксированном префиксе будет двенадцать символов [91].

Base64 подходит для безопасного хранения информации разрабатываемой системы, так как надежно зарекомендовал себя в шифрование передаваемых данных. Схема использования представлена на рисунках 3.3 и 3.4.


Рисунок 3.3 – Кодирование в Base64

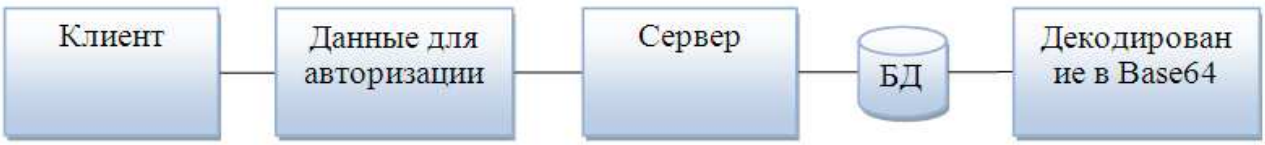


Рисунок 3.4 – Декодирование в Base64

Программный код Base64 реализован на языке JavaScript и имеет следующий вид:

```
(function (undefined) {
    // Shortcuts
    var C = CryptoJS;
    var C_lib = C.lib;
    var Base = C_lib.Base;
    var X32WordArray = C_lib.WordArray;
    var C_x64 = C.x64 = {};
    var X64Word = C_x64.Word = Base.extend({
        init: function (high, low) {
            this.high = high;
            this.low = low;
        }
        init: function (words, sigBytes) {
            words = this.words = words || [];
            if (sigBytes != undefined) {
                this.sigBytes = sigBytes;
            } else {
                this.sigBytes = words.length * 8;
            }
        }
        toX32: function () {
            // Shortcuts
            var x64Words = this.words;
            var x64WordsLength = x64Words.length;
            // Convert
            var x32Words = [];
            for (var i = 0; i < x64WordsLength; i++) {
                var x64Word = x64Words[i];
                x32Words.push(x64Word.high);
                x32Words.push(x64Word.low);
            }
            return X32WordArray.create(x32Words, this.sigBytes);
        },
        clone: function () {
            var clone = Base.clone.call(this);
            // Clone "words" array
```

```
            var words = clone.words = this.words.slice(0);
            // Clone each X64Word object
            var wordsLength = words.length;
            for (var i = 0; i < wordsLength; i++) {
                words[i] = words[i].clone();
        }
        return clone; } }); }()).
```

Организация защиты базы данных с хранимой в ней информацией с применением программной реализации стандарта кодирования данных Base64 существенно усиливает защиту разрабатываемой системы. Данный стандарт широко применяется в транспортном кодировании.

Для защиты пересылаемых данных MongoDB, которая в диссертационной работе используется в качестве СУБД, поддерживает протоколы TLS и SSL для приема и передачи данных по сети. В протоколах TLS и SSL используют шифрование для зашиты трафика веб–сайтов и обмена файлами [92].

Протокол SSL (Secure Socket Layer) – уровень защищенных сокетов [93]. Протокол TLS (Transport Layer Security) – безопасность транспортного уровня [94]. SSL является более ранней системой. TLS появился позднее и основан на спецификации SSL 3.0, разработанной компанией Netscape Communications. Задача у этих протоколов одна – обеспечение защищенной передачи данных между двумя компьютерами в сети Интернет. Такую защиту при передаче любой информации используют для различных сайтов, электронной почты, обмена сообщениям. Безопасная передача обеспечивается при помощи аутентификации и шифрования передаваемой информации.

Принципиальных различий между протоколами TLS и SSL нет. TLS является преемником SSL, хотя они и могут использоваться одновременно, причем даже на одном и том же сервере. Такая поддержка необходима для того, чтобы обеспечить работу как с новыми клиентами (устройствами и браузерами), так и с устаревшими, которые TLS не поддерживают.

Задачи SSL/TLS соединения:

- обеспечение приватности с помощью шифрования передаваемых данных;
- аутентификация с помощью сертификатов;
- обеспечение достоверности данных с помощью проверки целостности данных.

Эти протоколы шифрования для защиты данных, пересылаемых из одной точки в другую. Однако данные перед отправлением и после прибытия в место назначения, не зашифрованы. MongoDB предоставляет возможность настроить протоколы TLS и SSL с использованием сертификатов и пары «открытый и закрытый» ключи, также именуемой системой асимметричных ключей.

Серверная часть состоит из следующих файлов:

– index.js – основной файл, в котором происходит подключение базы данных, настройка сессий и обработка защищенного входа для подключенных информационных ресурсов;

– generators.js – файл, содержащий генератор слов и генератор тригонометрических функций.

Для реализации серверной части используются следующие фреймворки и библиотеки:

– Express – фреймворк для Node.js, упрощающий ведение разработки.

– Mongo–db – библиотека для подключения MongoDB.

– Nodemon – библиотека, позволяющая следить за обновлениями кода и перезагружать сервер после появления таковых.

– Request и Request-promise – библиотеки, осуществляют отправку запросов напрямую с сервера на сервер.

– Crypto–js – данная библиотека позволяет использовать модуль шифрации SHA256.

– Body–parser – библиотека, позволяющая обрабатывать данные приходящие на сервер в виде POST – запросов.

Реализация связи клиентской и серверной части позволяет осуществить передачу данных между мобильным приложением и серверной частью. Для решения данной проблемы были использованы стандартные библиотеки:

– Axios – связь клиента с сервером.

– Request – связь сервера с сервером.

Данные библиотеки позволяют гибко сформировать запрос, учитывая различные параметры и способы передачи данных (рисунок 3.5).

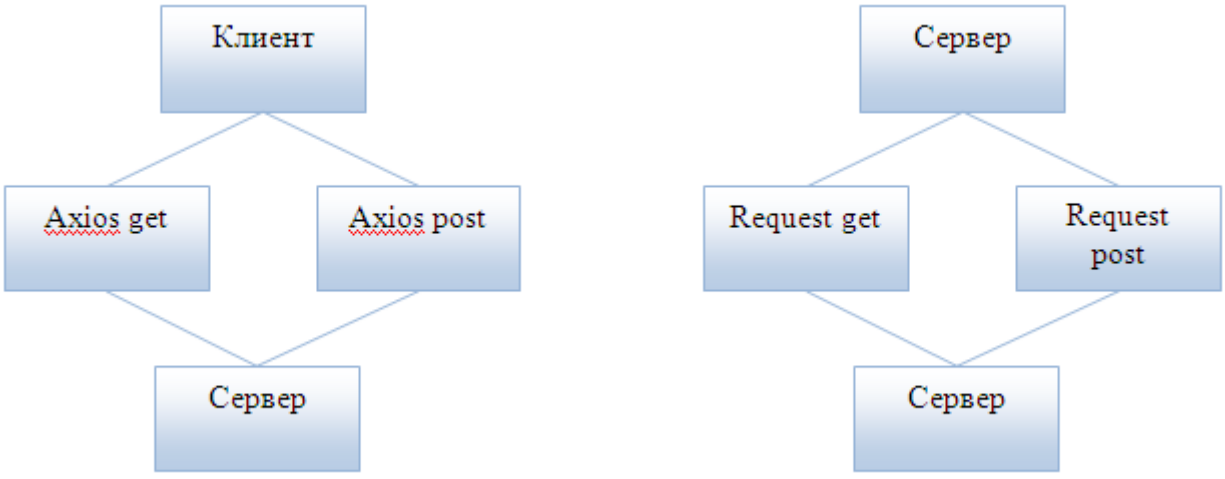


Рисунок 3.5 – Библиотеки для связи между клиентской и серверной частью

Разработка и отладка программного обеспечения для исследования научных работ использует современные методы, которые позволяют эффективно реализовывать разрабатываемые алгоритмы. Одним из решений является использование объектно–ориентированного программирования [95].

Для реализации разработанного в диссертационной работе алгоритма необходимо создать функции генерации одноразового пароля с использованием необходимых конфигураций алгоритма. Разработанная модель реализована на языке программирования JavaScript. На диаграмме классов представлена работа приложения для более детального описания имени класса и их атрибутов (рисунок 3.6) [96].

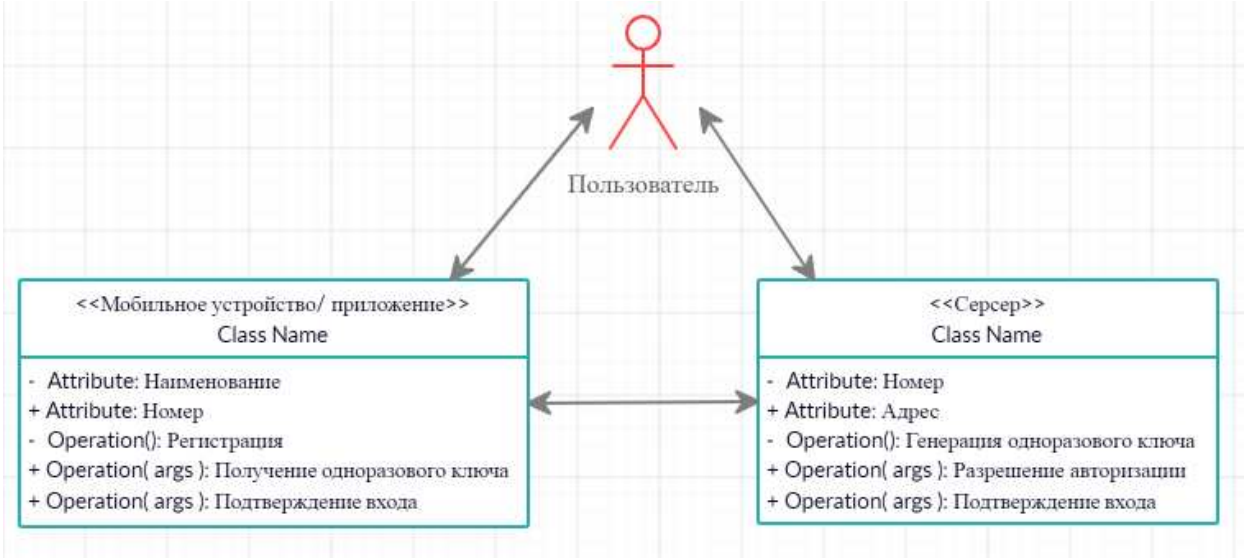


Рисунок 3.6 – Диаграмма классов

На рисунке 3.6 представлена диаграмма классов, на которой визуально описана взаимосвязь пользователя и приложения, предложенного в диссертационной работе.

### 3.2 Программная реализация защиты информации при аутентификации пользователя на основе одноразового пароля

На первом этапе по предложенной модели был разработан алгоритм генерации временного пароля для двухфакторной аутентификации [36, p.103].

Данный алгоритм реализован на языке JavaScript в консольном режиме с использованием библиотеки CryptoJS.SHA256 для шифрования данных [97]. Для проверки правильности работы предложенного алгоритма используется массив функций размерностью 10x10.

Работа первого этапа программной реализации диссертационного исследования представим в виде описанных ниже шагов.

Шаг1. Ввод данных. Начальными данными для входной строки будут следующие значения:

- логин: olga;
- пароль:ussatovaolga;
- секретная строка: nikdar.

Шаг2. Считывание системной даты и времени (19 декабря 2018 12:21:18). Текущий момент времени считывается автоматически и зависит от настройки операционной системы.

Шаг3. Использование библиотек CryptoJS.SHA256 для шифрования данных. Разбитие строки входных данных на слова. Вычисление хеш–значения.

Входная строка примет вид:olgaussatovaolga20181219122118nikdar.

Результат представляется в следующем виде:
8CD63646C6EE48DD3C542121A146144547E1B6D7DFA0423CE753B8C695CC7D58.

Шаг4. Формирование переменных на основании hashStr. Задание переменных для вычисления тригонометрических функций.

В качестве начальных параметров выбираются два шестнадцатеричных числа с конца результата хеш–функции, а в качестве значений «X» и «Y» берутся шестнадцатеричные числа с 10–11 позиции. Тогда они примут следующие значения:

1. a: "58";
2. b: "7D";
3. c: "CC";
4. p1: "95";
5. p2: "C6";
6. x: "6E";
7. y: "E4".

Шаг5.:

1. a: 88;
2. b: 125;
3. c: 204;
4. p1: 149;
5. p2: 198;
6. x: 110;
7. y: 228.

Шаг6. Определение индекса функции (mod 10), т.к. размер массива равен (10x10).

Шаг7. Заполнение массива размером 10x10, состоящего из строчных тригонометрических функций.

Шаг8. Выбор функции из массива по определенному индексу. В данном примере была взята таблица размером 10x10, в связи с этим выбирается следующая функция:

$$\mathrm{P1} * \cos(\mathrm{y})^2 - \sin(2\mathrm{C}) - \cos(\mathrm{P2})^3. \qquad (3.5)$$

Шаг 9. Передача в *f*– строчную функцию в виде программного кода.

Шаг10-11. Определение границ изменения аргументов функции.

Шаг12. Получение аргумента функции в виде строки.

Шаг13. Преобразование объекта values по заданным переменным, затем результат вывода функции с заданными переменными представляется в виде аргументов. В результате будет получено следующее значение функции:

$$149 * \cos(228)^2 - \sin(2 * 204) - \cos(198)^3 = 9.43306576524. \quad (3.6)$$

Шаг14. Объявление временного пароля.

Шаг15. Проверка результата на целостность и нужное количество цифр после запятой.

Шаг16. Временный пароль определяется числами после запятой.

Шаг17. Проверка длины пароля: после запятой не менее 8 символов.

Шаг18. Циклический перебор.

Шаг19. Если пароль имеет менее 7 цифр после запятой, то происходит дозапись пароля последовательностью цифр.

Шаг20. Вывод результата.

Шаг21. Начиная с 1–й позиции – длина 6 цифр.

Шаг22. Начиная с 5–й позиции – длина 6 цифр.

Шаг23. Проверка результата, равного 6 цифрам.

Шаг24. Временный пароль определяется числами после запятой, начиная с 5-й позиции, длина – 6 цифр. Тогда временный пароль принимает значение 657652.

Шаг25. Вывод временного пароля.

Применение двухфакторной аутентификации позволяет избежать несанкционированного взлома системы, снизить риск утечки персональных данных и другой важной информации в информационной системе. К преимуществам двухфакторной аутентификации можно отнести также возможность защиты информации от внутренних и внешних вторжений. Описанный алгоритм реализован на языке JavaScript в консольном режиме. Данное приложение позволяет наглядно отобразить генерацию временного пароля по описанному выше алгоритму. Реализация алгоритма представлена на рисунке 3.7. Секретная строка для генерации одноразового пароля на первом этапе вводилась вручную, и её длина была не фиксирована.

Для оценки сгенерированного временного пароля были взяты разные входные данные, которые показывают эффективность реализации алгоритма.

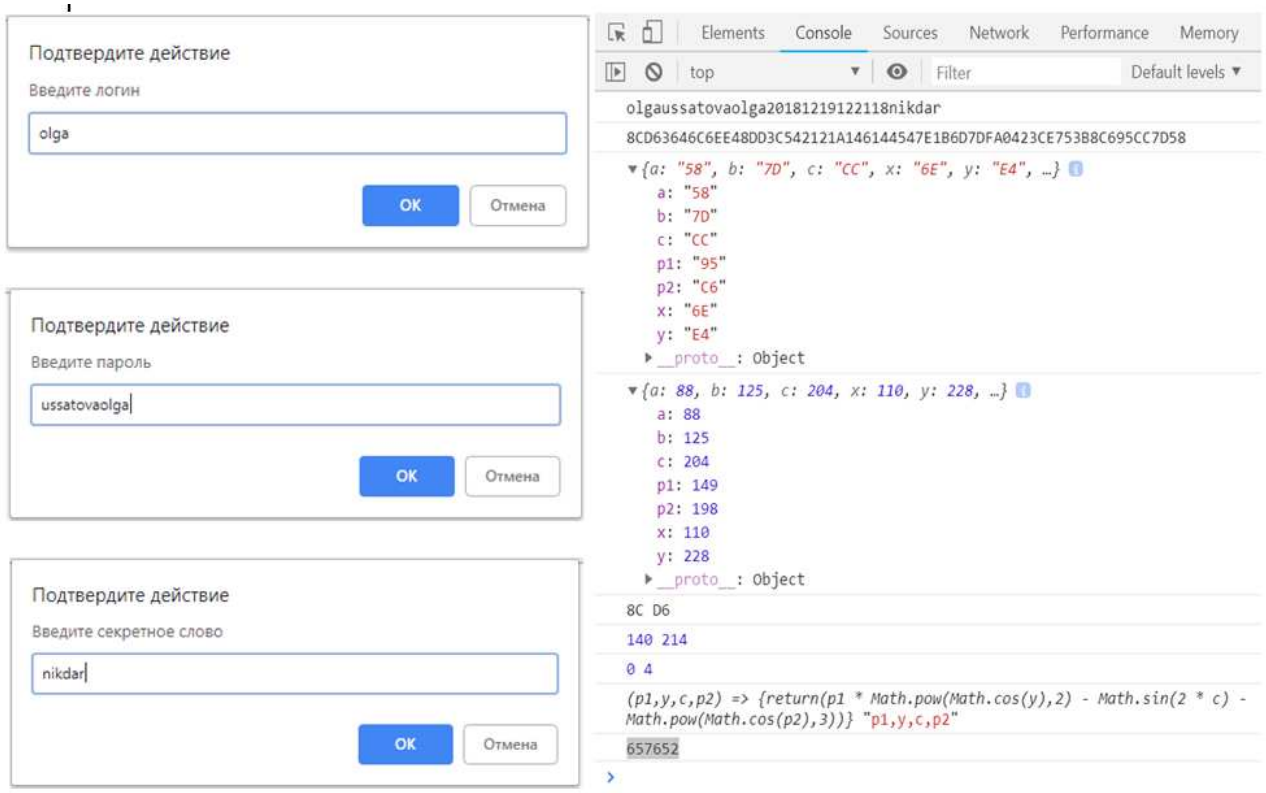


Рисунок 3.7 – Реализация описанного алгоритма на языке разработки JavaScript

При проведении анализа формирования одноразового пароля был выявлен недостаток, который в дальнейшем был устранен. Он заключался в том, что при вычислении тригонометрической функции её дробная часть равна 2 или 5 символам, а для выбора одноразового пароля по разработанному алгоритму необходимо было брать числа после запятой, начиная с 5 позиции длиной 6 цифр. Для решения этой проблемы на программном уровне была осуществлена проверка длины дробной части и дополнении её «0 и 1» по определенному алгоритму, что позволило правильно реализовать разработанный алгоритм.

Вторым этапом является разработка предложенного метода аутентификации, реализованного с использованием сетевой архитектуры клиент–сервер. Технология клиент–сервер является сетевой архитектурой с вероятным распределением нагрузки систем.

Разработанное клиент–серверное приложение, реализовано на программной платформе Node.js с использованием языка программирования JavaScript, а также фреймворков и подключенных системных библиотек. Для реализации данного приложения была выбрана программная платформа Node.js, так как она является серверной платформой для работы с JavaScript через движок V8. JavaScript выполняет действие на стороне клиента, а Node – на сервере [98]. Для языка JavaScript фреймворком является jQuery – библиотека с готовыми функциями визуальных эффектов.

Фреймворк определяет принципы организации архитектуры приложения, создавая основу при разработке, которую необходимо расширять, и вносить корректировки согласно указанным требованиям.

Для реализации приложения необходим смартфон, который отобразит сгенерированный одноразовый пароль для аутентификации пользователя в системе. Приложение разработано под операционную систему Android. По статистическим данным ОС Android использует 85,9 %пользователей [99], что определило его выбор для реализации предложенного метода.

Схема взаимодействия программных модулей предложенного алгоритма предтавлена на рисунке 3.8.

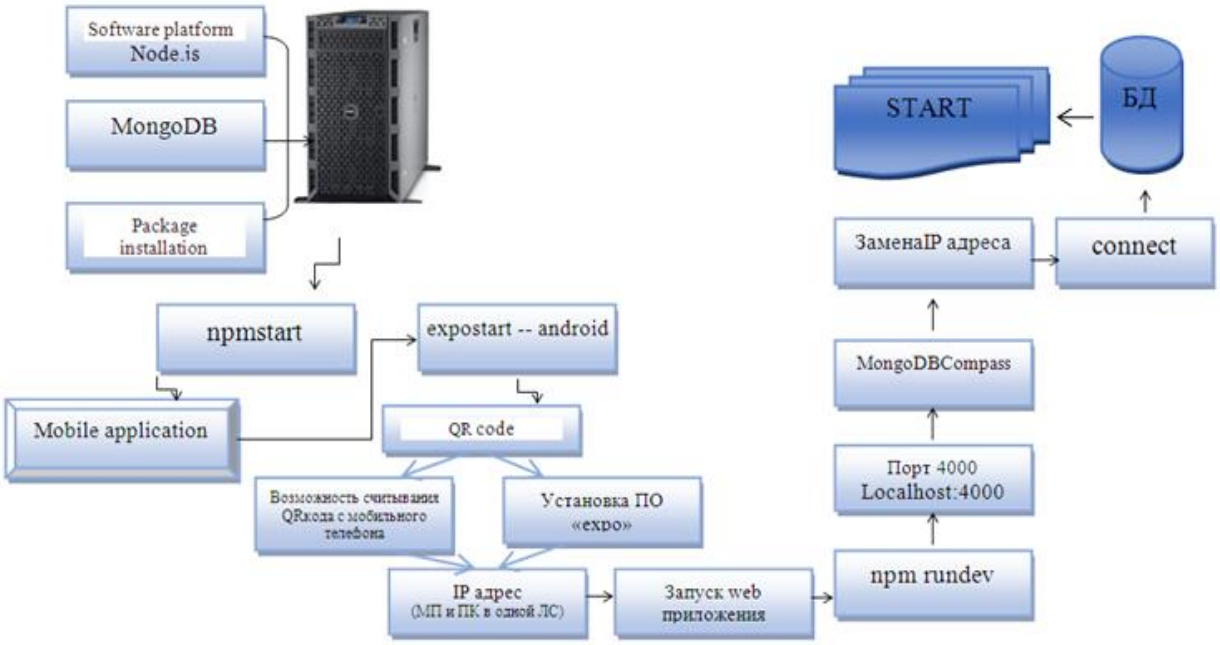


Рисунок 3.8 – Схема взаимодействия программных модулей

Для реализации системы необходимо установить Node.js. Эта программная платформа отвечает за написание серверной части на языке программирования JavaScrript. Также вместе с ней поставляется пакетный менеджер npm, который используется для установки различных библиотек и фреймворков. Дополнительно нужно установить.

MongoDB – документоориентированную систему управления базами данных с открытым исходным кодом, не требует описания схемы таблиц. При возникновении сложных запросов они обычно решаются на стороне приложения, что позволяет облегчить работу с данными и ссылками на них. MongoDB – часто применяемая нереляционная база данных. Использование системы управления базами данных (СУБД) MongoDB обусловлено тем, что в данную систему встроена достаточно простая масштабируемость с использованием технологии шардинга, которая производит разделение (партиционирование) базы данных на логические части. Парционирование базы

данных используется для того, чтобы каждая из частей могла функционировать на отдельном сервере [100].

Преимуществом использования СУБД MongoDB является:

– увеличение скорости разработки;

– отсутствие необходимости синхронизировать схему в базе данных и приложении;

– понятен принцип к маштабируемости;

– использование внутреннего метода шифрования данных;

– простота предписанных решений.

Если в реляционных БД содержимое составляют таблицы, то в MongoDB база данных состоит из коллекций.

Коллекция – это группа документов MongoDB, являющаяся эквивалентом простой таблицы в реляционной базе данных. Коллекция помещена внутри одной БД. Документ в коллекции может иметь различные поля (рисунок 3.9). Чаще всего, все документы в коллекции созданы для одной цели, либо относящихся друг к другу целей.

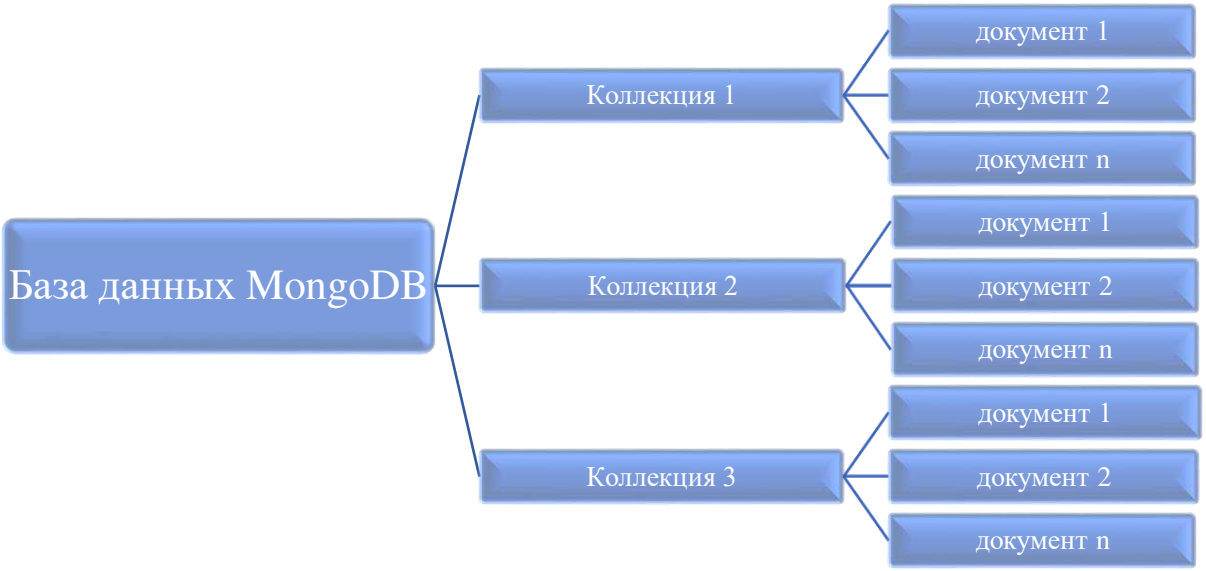


Рисунок 3.9 – Структура MongoDB

Для работы с БД была выбрана СУБД MongoDB, так как в отличие от реляционных баз данных она не использует табличное устройство с четко заданным количеством столбцов и типов данных. Документ в MongoDB можно представить как объект, хранящий некоторую информацию. В некотором смысле он подобен строкам в реляционных СУБД, где строки хранят информацию об отдельном элементе.

Выбор документноориентированной системы обусловлен тем, что база данных MongoDB, в которой хранятся данные для входа в систему и временная информация для организации аутентификации, использует внутренний метод шифровки данных, основанный на криптографическом алгоритме. В коде разрабатываемого приложения администратор сам задает логин и пароль:

MongoClient("mongodb+srv://admin:Mdb12812122424@@cluster0-o83lo.mongodb.net/test?retryWrites=true", { useNewUrlParser: true }).

Данная настройка позволит организовать защиту базы данных, т.к. это является важнейшей частью в хранении пользовательских данных. Такие результаты хакерских атак, как неавторизированный доступ и потеря данных пользователя, являются следствием ее недостаточной защиты. Для определения методов защиты нужно определить объекты защиты. Такими объектами могут выступать значения определенных полей, таблиц или записей, сами записи в таблицах, отдельные таблицы и целые базы данных. Для защиты этих объектов можно использовать такие методы как представления, триггеры и встроенные функции шифрования данных.

Представлением является динамическая выборка данных из нескольких таблиц. Также это можно назвать виртуальной таблицей, формирование происходящее в соотношении с пользовательским запросом, направленным на доступность представления данных. Используя данный подход, пользователь, получает доступ только к совокупности столбцов и записей, которые определены используемым представленным алгоритмом. Таким образом, можно ограничить доступные данные и контролировать набор информации с доступом для редактирования. Исходя из этого, метод представления можно использовать для защиты конфиденциальности и целостности данных.

Триггер – это слушатель, который вызывается во время исполнения определенных событий. Такими событиями являются добавление, удаление или редактирование данных в баз данных. Используя триггеры, можно определять пользовательский доступ для взаимодействия с данными в БД. Также они позволяют фиксировать все события, связанные с модификацией данных в защищаемой таблице. Таким образом, данный метод необходимо использовать для эффективного контроля целостности данных и слежения за всеми событиями, возникающими во время изменения информации.

Функции шифрования доступны далеко не во всех системах управления базами данных. В СУБД MySQL они предоставляют шифрование данных с помощью алгоритмов AES и DES. Также доступно хеширование с помощью алгоритмов MD5 и SHA1. Использование этих доступных функций значительно облегчает задачу защиты информации в базах данных.

Также немаловажное значение имеет защита базы данных от SQL инъекций. SQL инъекция – распространенный способ взлома сайтов и приложений. Целью такой атаки является нарушение работы или получение данных БД, без права доступа к ней. В тоже время внедрение инъекции, в зависимости от используемой системы управления базами данных и условий внедрения, может позволить атакующему выполнение произвольных запросов к базе данных. Атака типа внедрения SQL может быть возможна из–за некорректной обработки входных данных, используемых в SQL-запросах [101].

Используемая в диссертационной работе MongoDB защищена от SQL инъекций, так как это не реляционная система управления базами данных

документоориентированного типа и использует подобные документы и схему базы данных JSON ().

JSON (JavaScript Object Notation) –параметр текста, применим для записи и чтения. В JSON данные выводятся без разрывов строк для экономии места. Для дальнейшего расширения возможностей отладки была добавлена расширенная проверка JSON в соответствии с описанием, в json.org в RFC 4627 [102] . Обновление JSON необходимо для обеспечения возможности проверки нескольких стандартов, включая текущие спецификации RFC 8259 и ECMA–404 . Добавлена возможность исправления распространенных ошибок JSON. Если этот параметр включен, он заменяет неправильные кавычки, добавляет пропущенные кавычки, экранирует неэкранированные символы, удаляет комментарии и запятые. JSON не зависит от языка реализации.

В JSON могут использоваться такие типы данных, как строки, числа, объекты, массивы, логический тип, принимаемый значения (истина или ложь), значение null.

JSON используется в JavaScript сценариях, которые включают в себя:

– хранилище данных;

– конфигурацию и проверку данных;

– генерацию структур данных из пользовательского ввода;

– передачу данных с клиента на сервер, с сервера на клиент и между серверами.

Для работы с MongoDB на диске «C» создается папка «data», а в ней – папка «db».

Для установки пакетов, необходимых для запуска определенных частей проекта («Server»,«devschacht»,«dip») нужно открыть консоль в папке с файлом package.json и в ней выполнить команду «npm install». После установки всех пакетов проект запускается на выполнение частями. Вначале запускается «Server». Для этого нужно зайти в папку, открыть консоль и выполнить команду «npm start». После запуска появится сообщение «сервер запущен». Далее запускается мобильное приложение через папку «devschacht», в которой открывается консоль и запускается команда «expostart -- android». После запуска в консоли появится QR код: это код быстрого реагирования. Также в браузере откроется вкладка с системной информацией и QR кодом (рисунок 3.10).

Поле с IP адресом над QR кодом необходимо запомнить. Для успешной работы заменить IP адреса в коде на тот, который написан во вкладке в следующих файлах:

– Dip—> src—> server.js (строка 79).

– Server—>index.js (строка 225).

– Devschacht—>App.js (строка 17).

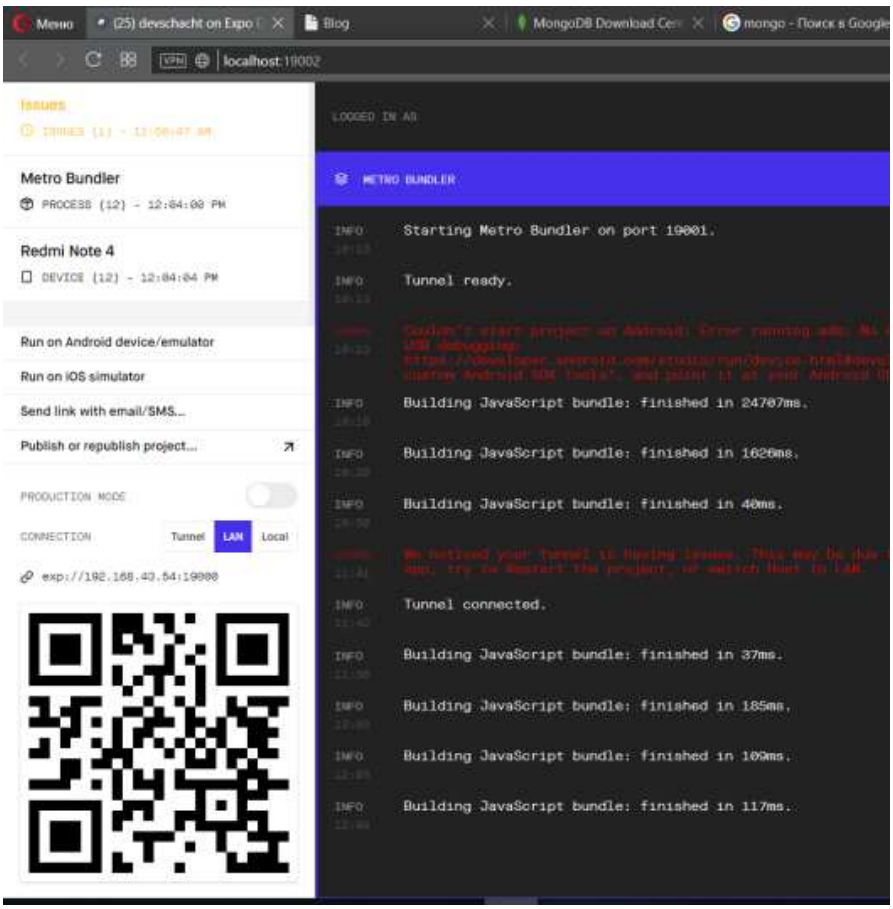


Рисунок 3.10 – Окно QR кода

Для работы мобильного приложения необходимо считывание QR кода, которое можно произвести при помощи установленного сканера на мобильном устройстве по умолчанию, либо установить дополнительное программное обеспечение, например «EXPO». Для запуска смартфона и персонального компьютера производится настройка локальной сети, таким образом, чтобы они находились в одной сети. Далее происходит запуск web сайта, для этого открывается папка «dip» и производится запуск приложения. Далее открывается консоль и выполняется команда «npmrundev». Для работы web приложения используется порт 4000 «localhost:4000». Для просмотра содержимого базы данных нужно устанавливать MongoDBCompass и произвести подключения нажатием кнопки «connect» рисунок 3.11.

После сохранения изменений всех настроек части приложения перезапустятся и готовы к работе [103].

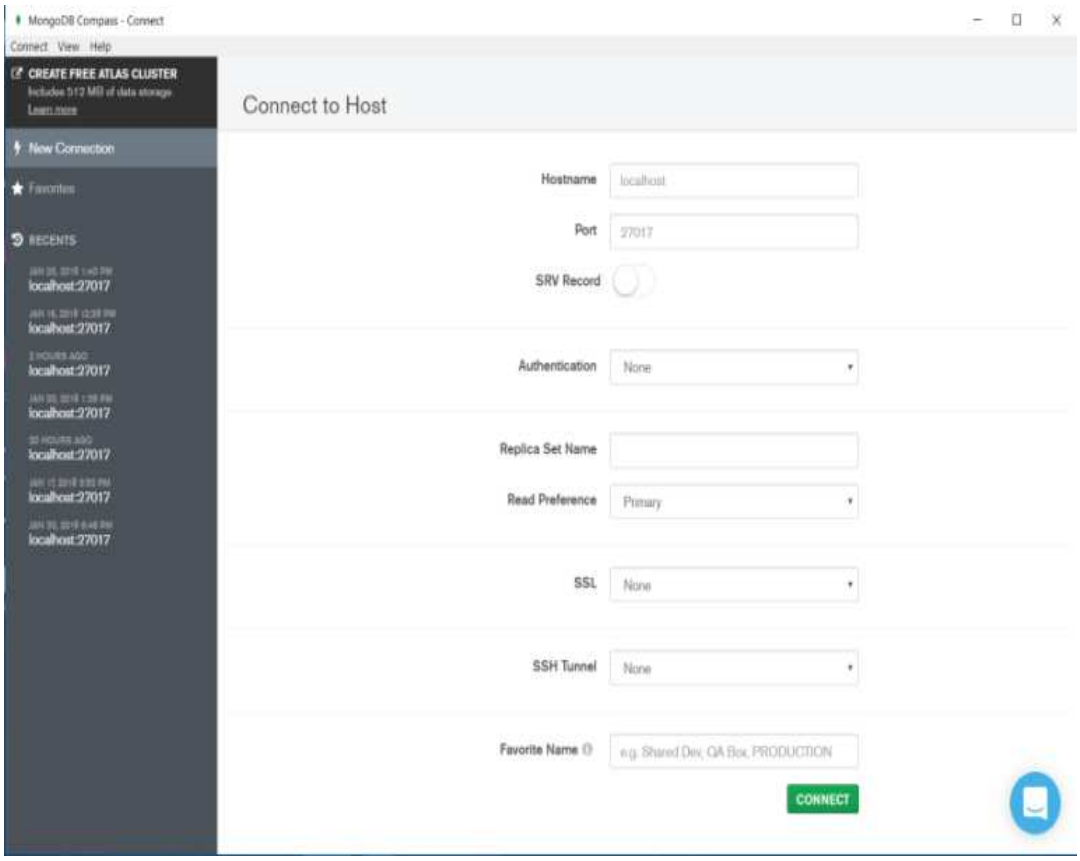


Рисунок 3.11 – Окно работы MongoDBCompass

### 3.3 Практическая реализация системы защиты информации при аутентификации пользователя на основе одноразового пароля

Программная реализация рассмотренного алгоритма представлена на рисунке 3.12.

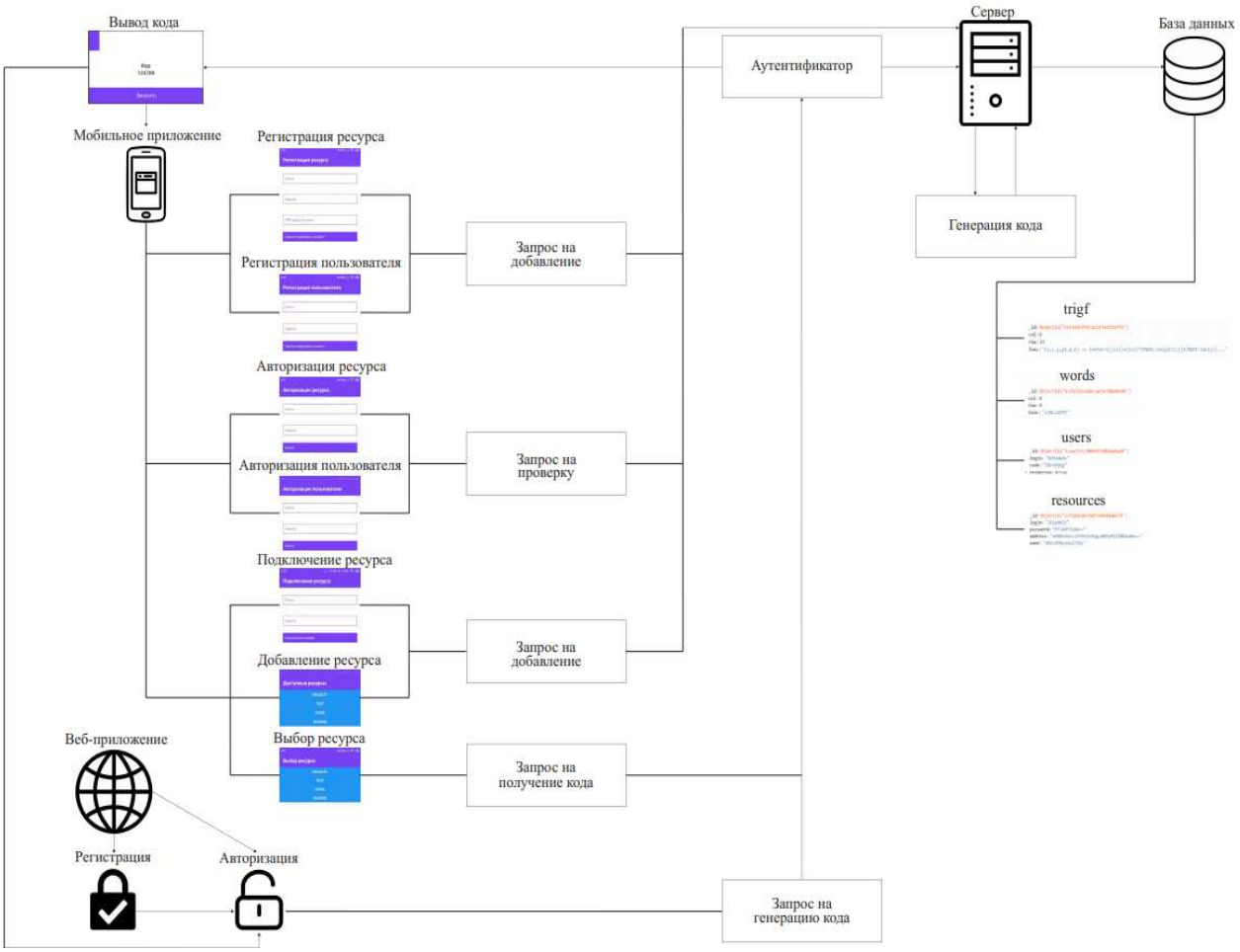


Рисунок 3.12 – Модель реализации предложенной двухфакторной аутентификации

Для запуска мобильного приложения необходимо выполнить следующие действия:

– установить приложение на телефон с помощью установочного «apk» файла;

– запустить приложение.

Реализуется данный алгоритм на стороне сервера. Со стороны клиента необходимо лишь взаимодействие с API приложением.

Для реализации клиентской части приложения была выбрана платформа для разработки мобильных приложений – ReactNative [104]. Она позволяет использовать ряд веб–технологий привычных для разработки веб–приложений и ставит расчет на использование компонентов для написания универсальных мобильных приложений.

После запуска разработанного приложения Security Code of the 2FA открывается главный экран [78, p.10], рисунок 3.13.

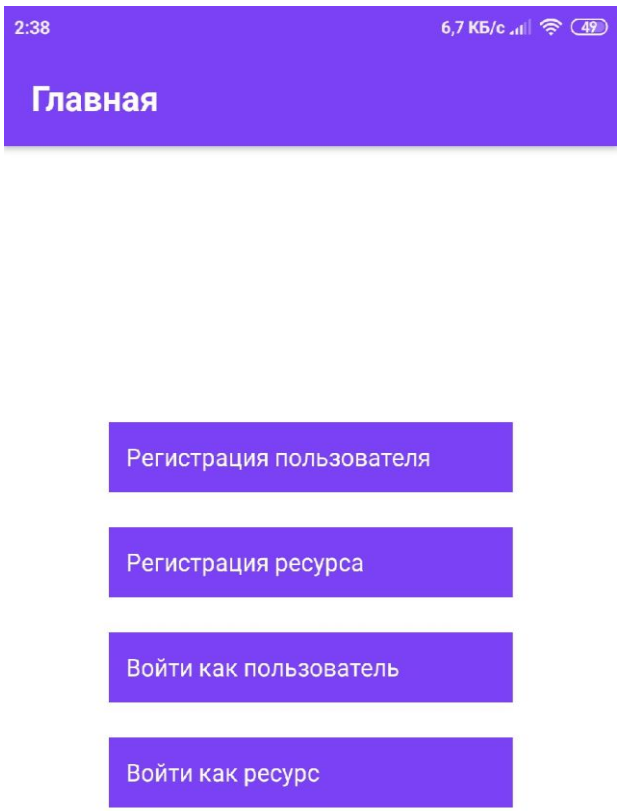


Рисунок 3.13 – Главный экран

Для начала работы с приложением пользователю необходимо зарегистрировать аккаунт. Для регистрации пользовательского аккаунта необходимо нажать на кнопку «Регистрация пользователя», после этого откроется экран с формой для регистрации пользовательского аккаунта, рисунок 3.14.

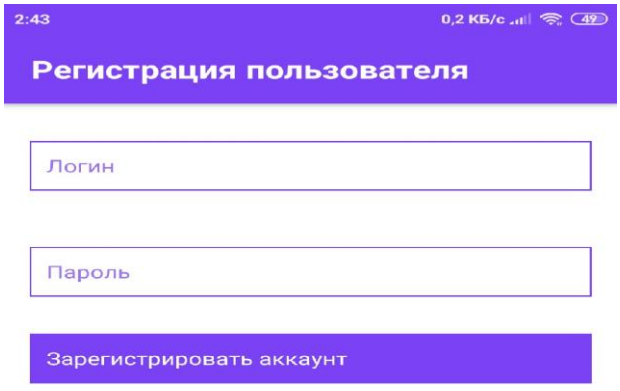


Рисунок 3.14 – Регистрация пользователя

Для регистрации необходимо заполнить форму и нажать на кнопку «Зарегистрировать аккаунт». После успешной регистрации пользователь может осуществить вход в свой аккаунт. Для этого на главном экране необходимо

нажать на кнопку «Войти как пользователь» и в открывшемся окне заполнить форму и нажать на кнопку «Войти», рисунок 3.15.

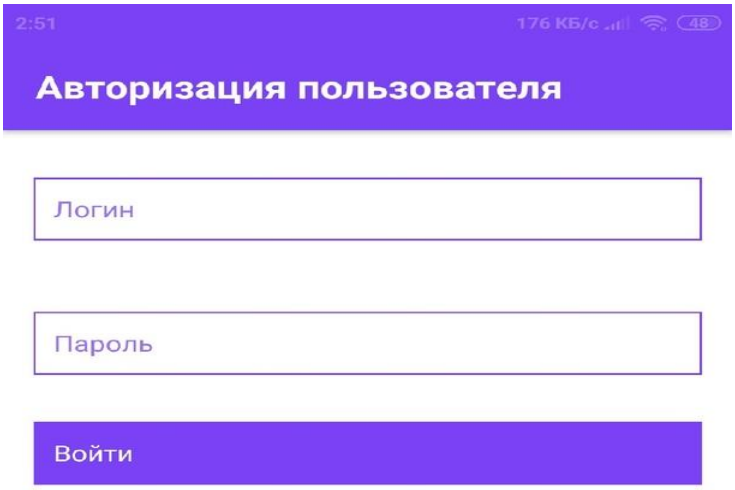


Рисунок 3.15 – Авторизация пользователя

После успешного входа открывается экран пользовательского аккаунта. На этом экране пользователю предоставляются функционалы «Подключить ресурс» и «Генерация кода», рисунок 3.16.

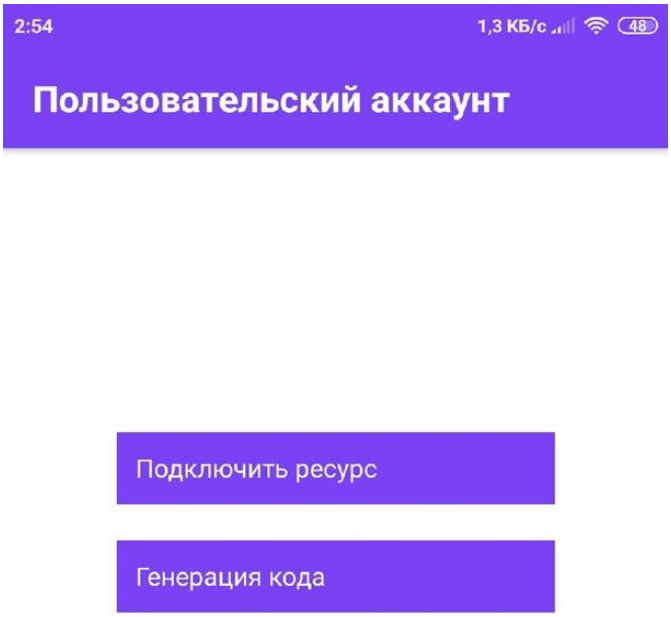


Рисунок 3.16 – Пользовательский аккаунт

Для генерации секретного кода при осуществлении авторизации необходимо подключить ресурс кнопкой «Подключить ресурс». Открывается экран со всеми доступными ресурсами, рисунок 3.17.

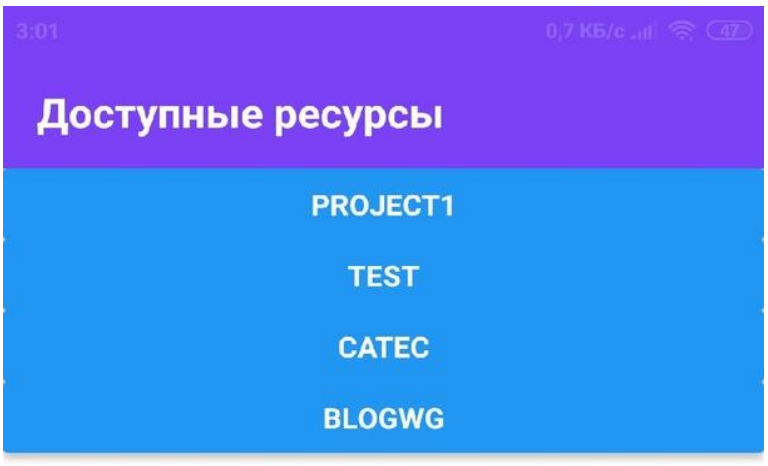


Рисунок 3.17 – Доступные ресурсы

Далее необходимо нажать на нужный ресурс. Тогда откроется экран подключения ресурса с формой, которую необходимо выбрать и нажать на кнопку «Подключить ресурс», рисунок 3.18.

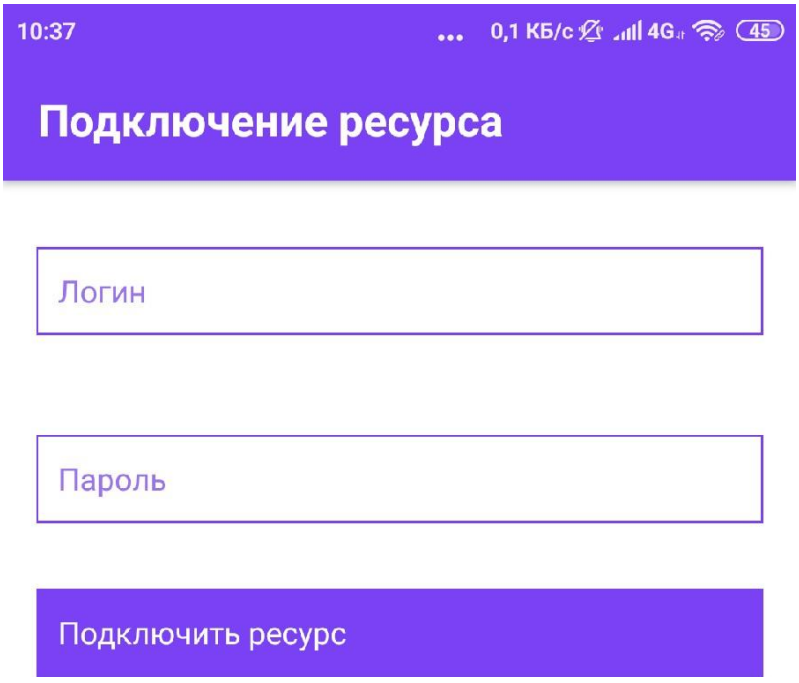


Рисунок 3.18 – Подключение ресурса

Далее после успешного подключения ресурса на экране пользовательского аккаунта следует нажать кнопку «Генерация кода». Откроется окно выбора ресурса, рисунок 3.19.

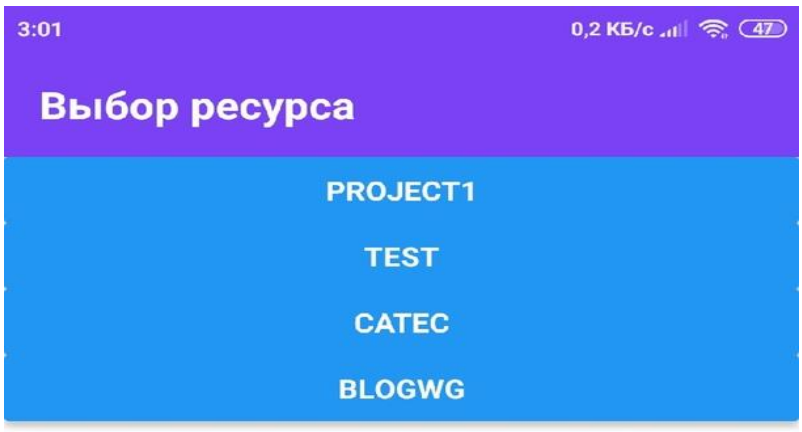


Рисунок 3.19 – Выбор ресурса

После выбора ресурса будет открыто модальное окно с тремя элементами, рисунок 3.20:

– прогрессбар – данный элемент визуально указывает, через какой период времени будет осуществлено обновление кода. После того как прогрессбар начнет заполняться заново, ранее отображаемый временный пароль будет более недействительным;

– текст с кодом – с помощью данного элемента выводится временный пароль, который необходимо ввести в информационном ресурсе во время осуществления авторизации;

– кнопка «Закрыть» – данный элемент позволяет закрыть модальное окно и прекратить генерацию временных секретных паролей.

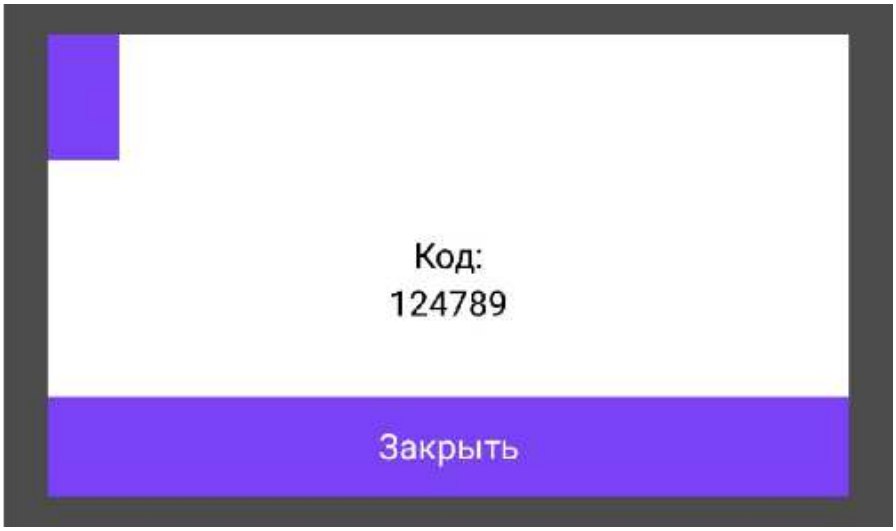


Рисунок 3.20 – Модальное окно

Для подключения информационного ресурса к системе защиты необходимо на главном экране нажать на кнопку «Регистрация ресурса»: откроется экран с формой, которую требуется заполнить и нажать на кнопку «Зарегистрировать аккаунт», рисунок 3.21.

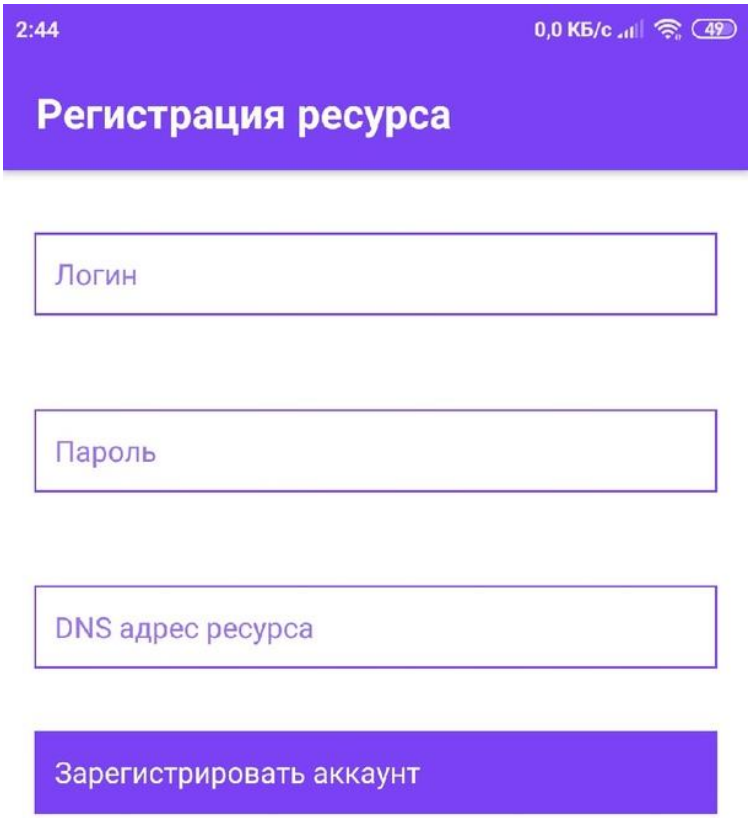


Рисунок 3.21 – Регистрация ресурса

После успешной регистрации необходимо авторизоваться в системе. Для этого пользователю нужно нажать на кнопку «Войти как ресурс» и заполнить форму в открывшемся экране, затем необходимо нажать на кнопку «Войти», рисунок 3.22.

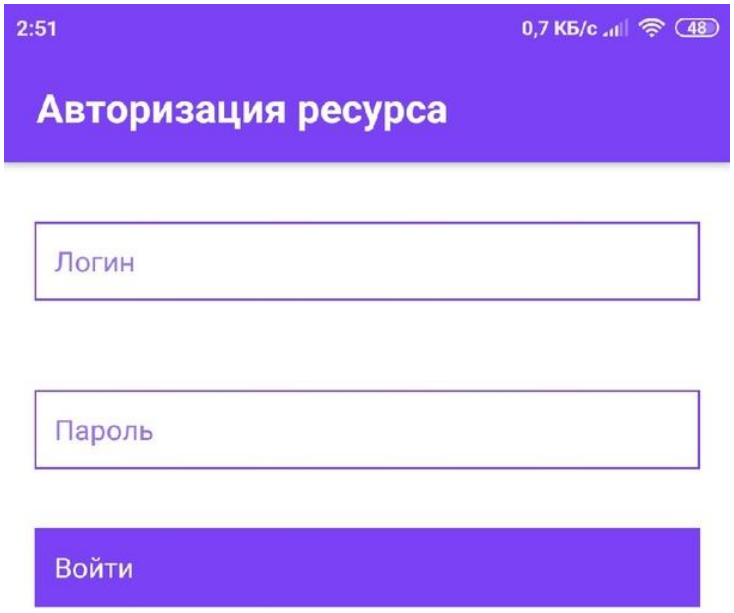


Рисунок 3.22 – Авторизация ресурса

После успешной авторизации откроется экран аккаунта, на котором можно просмотреть API для взаимодействия с системой двухфакторной аутентификации.

Рассмотрим влияние генерации временного пароля на программную реализацию средств защиты информации на основе двухфакторной аутентификации. Для оценки сгенированного временного пароля были взяты разные входные данные, которые показывают эффективность реализации алгоритма.

В таблице 3.1 приведены результаты исследования генерации временного пароля по описанному выше алгоритму.

Таблица 3.1 состоит из 5 столбцов:

– столбец «№ п/п» – порядковый номер столбца;

– столбец «Входные данные» – учетные данные для авторизации пользователя (Логин, Пароль, Текущий момент даты/времени, Секретная строка);

– столбец «Хеш-значение» – полученное хеш–значение, которое используется для работы алгоритма, описанного выше;

– столбец «Тригонометрическая функция» – это тригонометрическая функция и ряд переменных, которые генерируются из массива 256x256 случайным образом;

– столбец «Пароль сгенерирован» – получен одноразовый пароль из результата расчета тригонометрической функции.

Таблица 3.1 – Генерация временного пароля на основе описанного алгоритма

| № п/п | Входные данные | Хеш–значение | Тригонометрическая функция | Сгенери–рованный пароль |
|---|---|---|---|---|
| 1. | user16<br>pass17word<br>20191121151931<br>saLte | 90CC9939B3C05AA8D36A16B95EE5416B19632332CFCF46B30C4D37DFDE3F7DB7 | (p1,p2,x) => {return((Math.pow(Math.sin(x),3) + Math. sin (Math.pow(x),2)) /(p1/p2)} " p1,p2,x " | 511132 |
| 2. | login1<br>pas123<br>2019121716362<br>seret | 73BD8C18ED83B1171DC437C71A762210DE52080EF9DF64C1178967BCEEDC6CAD | (y,p1,x,c) => {return ((y*Math.pow(Math.sin(p1),2) + 4 * Math.pow(Math.tan(x), 4)) * c)} "y,p1,x,c" | 711633 |
| 3. | login1<br>pas123<br>20191217163947<br>seret | 3CF82991E4FB8B0D2C748013662A0C20FCCAE7D871A60A6E06F634F67900C81A | (y,c,b,x) => {return ((Math.cos(y) * (x * Math.sin(c)))/Math.tan (Math.sqrt(b)))} "y,c,b,x" | 101010 |
| 4. | user123<br>password456<br>20191217164158<br>Werol | 2F37FB8B50E6D2E5611F8CA6B56CEF60D2F78B399C0E3DBD720CC447C3D45018 | (p1,y,c,p2) => {return(p1 * Math.pow(Math.cos(y),2) - Math.sin(2 * c) - Math.pow(Math.cos(p2),3))} "p1,y,c,p2" | 005996 |
| 5. | pinokio<br>qwert<br>2019121716449<br>asdfA | BBA39EFC927C6BBA86AD45FE524DDAC6BCEE9F0A0830BE62F67A5474B0B8BBD9 | (a,b,y) => {return(Math.sqrt(a) * Math.sin(b + Math.PI/y))} "a,b,y" | 296145 |
| 6. | olga<br>pass20line<br>20191217165354<br>saKlt | 985465A101B8CFD070FE44CB16CA7D4F5CB5B9FAA0741051B286FBA6B910AAD | (p1,y,c,p2) => {return(p1 * Math.pow(Math.cos(y),2) - Math.sin(2 * c) - Math.pow(Math.cos(p2),3))} "p1,y,c,p2" | 124789 |

Результаты анализа показывают, что сгенерированный временный пароль не повторяется и изменяется даже при вводе повторяющихся данных. Это обусловлено тем, что во входных данных учитываются не только логин и пароль, но системная дата/время и секретная строка. Предложенный метод позволяет использовать двухфакторную аутентификацию для обеспечения защиты информации в информационной системе [78, p.11].

### 3.4 Производительность разработанной программной реализации

Исследование производительности разработанного приложения позволяет значительно уменьшить период обработки алгоритма и расходы на поддержание работоспособности. Автоматизация процесса тестирования позволит уменьшить время обработки данных в информационной системе.

Одной из важных характеристик применения программного продукта является скорость его выполнения. В связи с этим рассмотрены результаты исследования скорости выполнения программной реализации алгоритма двухфакторной аутентификации на основе генерации одноразового пароля.

Результаты анализа выполнения алгоритмов двухфакторной аутентификации для операционной системы Android, представлены на рисунке 3.23.

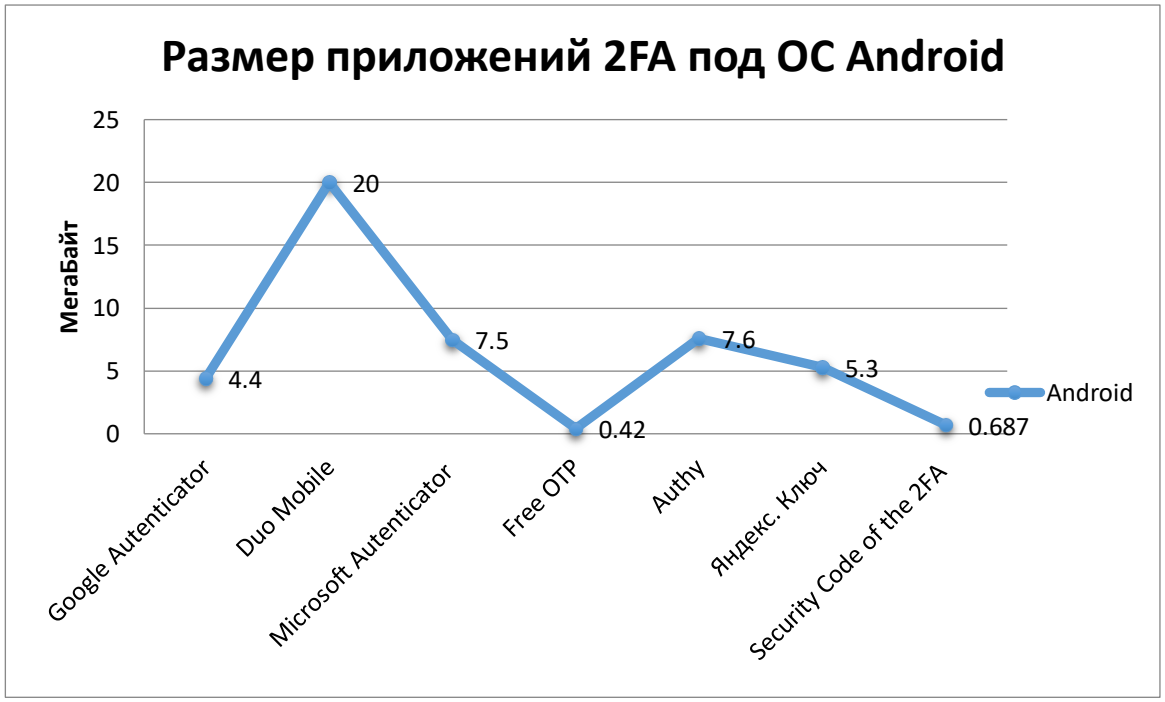


Рисунок 3.23 – Размер приложения 2FA

По ОСИ «X» представлены приложения двухфакторной аутентификации, а по ОСИ «Y»размер приложений в МегаБайтах.

Из графика видно, что рассмотренное в диссертационной работе клиент-серверное приложение Security Code of the 2FA имеет размер 0,687 Мбайт, что является оптимальным для реализации предложенного алгоритма.

### 3.5 Выводы по третьему разделу

В данном разделе приведены результаты по программной реализации предложного алгоритма. Разработанное приложение является клиент–серверной архитектурой и позволяет полностью реализовать разработанный алгоритм.

Рассмотрены компоненты системы: пользователь, мобильное приложение и серверная часть, описаны используемые файлы, библиотеки и фреймворки для реализации данного алгоритма.

Описаны механизмы криптографической защиты баз данных Base64 и работа алгоритмов TLS и SSL.

Для разработанной информационной системы выбрана программная платформа Node.js, язык программирования JavaScript и документоориентированная система управления базами данных MongoDB.

Новизна результатов: реализованы модели генерация и последующего обновления блоков функций и секретных слов с помощью генераторов, реализованных на стороне сервера.

Для проверки корректности одноразового пароля были рассмотрены как разные, так и одинаковые учетные данные пользователя, которые генерировались по описанному алгоритму. Результаты проверки показали правильность работы исследуемой системы.

# ЗАКЛЮЧЕНИЕ

Основные результаты, полученные в диссертационной работе.

Диссертационное исследование заключалось в разработке и исследовании алгоритма аутентификации пользователей информационно-коммуникационных систем на основе второго фактора. Алгоритм двухфакторной аутентификации создан на основе генерации одноразового пароля (второго фактора), который вычисляется с использованием выбранной тригонометрической функции для усиления защиты информационной системы.

Проведен анализ нормативно-правовых актов по вопросам и проблемам обеспечения информационной безопасности по теме диссертационного исследовании.

Осуществлен анализ известных алгоритмов и протоколов двухфакторной аутентификации, криптографических алгоритмов и приложений для защиты информации в информационной системе.

Разработан алгоритм двухфакторной аутентификации на основе программы – аутентификатора и мобильного телефона, который генерирует усложненный набор функций одноразового пароля для каждой отдельной информационно-коммуникационной системы. Предложенная система может быть внедрена также и в закрытую сеть

Для предложенного алгоритма аутентификации пользователей информационно–коммуникационных систем на основе второго фактора разработано и реализовано клиент – серверное приложение.

По результатам проведенных исследований опубликовано 15 научных работ в журналах и престижных международных конференциях, из них 2 в базе данных Scopus и Web of Science.

Получены 2 авторских свидетельства:

– Сертификат регистрации авторского права на алгоритм «Двухфакторная аутентификация в автоматизированной системе управления» №712144534 от 2018.10.01, компании «WORKS COPYRIGHT», это цифровая сертификация, юридически признанная во всем мире для регистрации авторских прав авторов, New York – NY–USA.

– Авторское свидетельство о внесении сведений в государственный реестр прав на объекты, охраняемые авторским правом РК, № 4330 от 28 июня 2019г., «Система аутентификации с использованием второго фактора для контроля доступа к данным –Security Code of the 2FA».

Выполнение диссертационной работы проведено в соответствии с календарными планами научно-исследовательских работ на 2018–2020 годы проекта программно-целевого финансирования (ПЦФ) КН МОН РК «Разработка программных и программно-аппаратных средств для криптографической защиты информации при ее передаче и хранении в инфокоммуникационных системах и сетях общего назначения» и проекта грантового финансирования (ГФ) КН МОН РК «Разработка казахстанского сегмента защищенного трансграничного информационного взаимодействия»

Института информационных и вычислительных технологий КН МОН РК. Результаты исследования по данной диссертационной работе включены в отчеты указанных проектов ПЦФ за 2018–2019 годы и ГФ за 2020.

Полученные результаты являются новыми, имеют теоретическую и практическую значимость. Проведенные научно-исследовательские работы по разработке и анализу систем защиты информации при идентификации и аутентификации пользователя на основе двухфакторной аутентификации направлены на решение задач по обеспечению информационной безопасности.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Казахстан – 2030. Процветание, безопасность и улучшение благосостояния всех казахстанцев. Послание Президента страны народу Казахстана – 1997.http://adilet.zan.kz/rus/docs/K970002030_.30.10.2018.

2 Концепция Применение известных алгоритмов криптографической защиты информации по новому информационно-технологическому назначению для обеспечения персональной идентификации/аутентификации и управления доступом пользователей в едином информационном пространстве глобальной компьютерной сети Интернет.https://www.zakon.kz/4744402-koncepcija-primenenie-izvestnykh.html.30.10.2018.

3Закон Республики Казахстан «О национальной безопасности Республики Казахстан»// Астана, Аккорда, 2012. – №527–IV ЗРК//http:// continent-online.com/document/?doc_id=31106860#pos=511;-34/11.11.2018.

4 О полномочиях государственных органов Республики Казахстан вступила в силу 27 декабря 2017 года. https://online.zakon.kz/ document/? doc_id = 31106860#pos=3;-155. 11.11.2018.

5 Закон «О персональных данных и их защите» (с изменениями и дополнениями по состоянию на 28.12.2017г.). https://online.zakon.kz/ document/? doc_id=31396226. 11.11.2018.

6 Закон Республики Казахстан Об информатизации (с изменениями по состоянию на 01.01.2020г.). https://online.zakon.kz/document/? doc_id = 33885902 &show_di=1. 25.01.2020.

7 Закон Республики Казахстан от 7 января 2003 года № 370-II Об электронном документе и электронной цифровой подписи (с изменениями и дополнениями по состоянию на 25.11.2019 г.). https://online.zakon.kz/ document/?doc_id=1035484#pos=41;-155. 25.12.2019.

8 Постановление Правительства Республики Казахстан от 12 декабря 2017 года № 827, Об утверждении Государственной программы Цифровой Казахстан. https://online.zakon.kz/document/?doc_id=37168057. 15.02.2019.

9 Третья модернизация Казахстана: Глобальная конкурентоспособность. Послание Главы государства народу Казахстана. – Астана. https://online.zakon. kz/ Document/?doc_id=35676318#pos=1;-119. 12.12.2018.

10 Постановление Правительства Республики Казахстан. Об утверждении Концепции кибербезопасности (Киберщит Казахстана): 30 июня 2017 года № 407.http://adilet.zan.kz/rus/docs/P1700000407. 12.12.2018.

11 Digital.Report/ Все о цифровой экономике и ИКТ в политике. https:// digital.report/kibershhit-kazahstana-ideologiya-konkretika. 15.02.2019.

12 Калимолдаев М.Н., Бияшев Р.Г., Рог О.А. Анализ атрибутных методов разграничения доступа // Прикладная дискретная математика. – 2019. – № 44. – С. 43–58.

13 Ding Wang, Wenting Li, Ping Wang. Measuring Two-Factor Authentication Schemes for Real-Time Data Access in Industrial Wireless Sensor Networks// Industrial Informatics IEEE Transactions. – 2018. – Vol.14, № 9. – P. 4081-4092.

14 Edna Elizabeth N., S. Nivetha. Design of a two-factor authentication ticketing system for transit applications//IEEE Region 10 Conference (TENCON), Singapore. – 2016.– P. 2496–2502.

15 Faezeh Sadat Babamir, Murvet Kirci. Dynamic digest based authentication for client–server systems using biometric verification// Future Generation Computer Systems.2019. -Vol.101. – P.112– 126.

16 Yi Yu, Jingsha He, Nafei Zhu, Fangbo Cai, Muhammad Salman Pathan. A new method for identity authentication using mobile terminals// Procedia Computer Science. – 2018. – Vol.131.– P. 771– 778.

17 Исхаков А.Ю., Мещеряков Р.В., Шелупанов А.А., Исхаков С.Ю. Современные методы и способы идентификации // Теория и практика. – Томск: ТГУСУР, 2016. – 114 с.

18 Исхаков А.Ю. Система двухфакторной аутентификации на основе QR кодов // Безопасность информационных технологий. –2014. – № 3. – С. 97–101.

19 M. A. Poltavtseva, Evolution of Data Management Systems and Their Security// International Conference on Engineering Technologies and Computer Science (EnT). – M., Russia, 2019. –P. 25–29.

20 Гапанович Д.А., Чубариков В.Н. Хеш-алгоритм с управляющей древовидной структурой и метод его реализации на параллельных архитектурах// Современные информационные технологии и ИТ-образование. – М., 2017, – С.35–42.

21 Послание Президента Республики Казахстан Н. Назарбаева народу Казахстана. Новые возможности развития в условиях четвертой промышленной революции: 10 января 2018 г. http://www.akorda.kz/ru/addresses/addresses_of _president/poslanie-prezidenta-respubliki-kazahstan-n-nazarbaeva-narodu-kazahstana -10-anvarya–2018–g. 15.06.2019.

22 Национальный доклад по науке. Тенденции развития в науке, выявление позитивных и негативных факторов, влияющих на развитие казахстанской науки. http://nauka-nanrk.kz/ru/ assets /%D0 % 94%D0% BE%D0% BA%D0%BB%D0%B0%D0%B4/%D0%9D%D0%B0%D1%86%D0%B4%D0%BE %D0%BA%D0%BB%D0%B0%D0%B4%20.pdf.15.06.2019.

23 Атаками и угрозы информационной безопасности. https://www.zakon. kz /4985176-kolichestvo-intsidentov-svyazannyh-s.html. 21.07.2019.

24 СТ РК 34.026-2006. Защита информации. Основные термины и определения. https://online.zakon.kz/document/?doc_id=31507774. 21.07.2019.

25 СТ РК ISO/IEC 27001-2015. Информационная технология. Методы и средства обеспечения безопасности. Системы менеджмента информационной безопасностью. Требования. https://online.zakon.kz/Document/?doc_id= 36588184 #pos=0;0. 25.07.2019.

26 Ramez Elmasri, Shamkant B. Navathe.Database Systems.– Boston: Addison-Wesley, 2010. – P.1201.

27 Тарасов С. В. СУБД для программиста. База данных изнутри. – М.: Солон-Пресс, 2017.– 320с.

28 Шульц В.Л., Рудченко А.Д., Юрченко А.В. Комплексное противодействие атакам на информационные ресурсы.– М.: АСТ, 2016.– 87с.

29 Нысанбаева С.Е., Усатова О. А. Способы обеспечения безопасности информации в базах данных // Вестник КазНИТУ им.Сатпаева.–Алматы, 2018. – № 2. – С. 66–70.

30 Кульба В.В., Курочка Н.П. Математическая модель обеспечения безопасности информации в базах данных// Интернет-журнал «НАУКОВЕДЕНИЕ». – 2015. – Т. 7, №3. – С. 1–6.

31 Нысанбаева С.Е., Усатова О.А. Криптографическая защита в автоматизированных системах// Науч. конф. «Современные проблемы информатики и вычислительной технологий». – Алматы, 2018. – С. 220–223.

32 Nathan Martz, James Warren. Big Data: Principles and Best Practices of Scalable Realtime Data Systems. – United States of America: Manning, 2017. – P.330.

33 Нысанбаева С.Е., Усатова О.А. Возможное применение больших данных в системе образования// Матер. междунар. науч. практ. конф. «Инновационные технологии на транспорте: образование, наука, практика» в рамках реализации Послания Президента РК Н. Назарбаева «Новые возможности развития в условиях четвертой промышленной революции» – Алматы, 2018. – С. 95–99.

34 Aloul, F.; Zahidi, S.; El–Hajj, W.Two factor authentication using mobile phones//Computer Systems and Applications. ACS International Conference on AICCSA 2009 IEEE. –P.641–644.

35 Нысанбаева С.Е., Усатова О.А. Двухфакторная аутентификация в автоматизированной системе управления// III Междунар.науч. конф. «Информатика и прикладная математика». – Алматы, 2018. –С. 239–242.

36 S. Nysanbayeva, W. Wojcik, O. Ussatova. Algorithm for generating temporary password based on the two-factor authentication model// Przegl№d Elektrotechniczny. – Polan,–2019. –№ 5. – P. 101–106.

37 Юрьев Д.Р., Рогова О.С.Сравнительный анализ двухфакторной аутентификации// LXXI междунар. науч. – практ. конф. Технические науки – от теории к практике.– Новосибирск: СибАК, 2017, № 6(66). – С. 46–51.

38 М.А. Шнепс-Шнеппе. Система сигнализации SS7 и ее уязвимость // International Journal of Open Information Technologies. – 2015.- Vol. 3, №5.– С. 1 – 11.

39 С. П. Евсеев В. Г. Абдуллаев. Алгоритм мониторингу метода двухфакторной аутентификации на основе системы PASSWINDOW // Восточно-Европейский журнал передовых технологий, 2015. –№ 2/2 (74).– С. 9–16.

40 OpenID Connect. https://auth0.com/docs/protocols/oidc.18.08.2019.

41 Ильина, Л.А., ПавловД.В. Авторизация в системе PolyAnalyst с использованием протокола OAuth 2.0 // Аллея науки. 2017. № 5. –С. 481–484.

42 СТ РК ИСО/МЭК 10118-1-2006 «Информационная технология. Методы защиты информации. Хеш–функции. Часть 1. Общие положения». https://online. zakon. kz/Document/?doc_id=31071245#pos=0;0.15.06.2019.

43 СТ РК ИСО/МЭК 10118-3-2006 (ISO/IEC 10118-3-2004, IDT) «Информационная технология. Методы защиты информации. Хеш–функции. Часть 3. Специализированные хеш–функции». https://online.zakon.kz/ Document / ?doc_id = 30461868#pos=0;0.10.08.2019.

44 Информационная технология. Криптографическая защита информации. Функция хеширования. ГОСТ Р34.11–2012.–Москва, Стандартинформ, 2012. http://docs.cntd.ru/document/gost-r-34-11-201210.08.2019.

45 Л.К. Бабенко, Е.А. Ищукова. Дифференциальный криптоанализ упрощенной функции хеширования sha// Известия ЮФУ, Технические науки, – 2010.– С 203–220.

46 Configuring an HOTP one-time password mechanism. https://www. ibm. com/support/knowledgecenter/SSPREK_9.0.3/com.ibm.isam.doc/config/task/Config uringOneTimePassword HOTP.html.10.08.2019.

47 HMAC: Keyed- Hashing for Message Authentication. https:// tools. ietf. org/html/rfc 2104. 23.08.2019.

48 HUSEYNOV, Emin, SEIGNEUR, Jean-Marc. Enhancing RADIUS based multifactor-factor authentication systems with RESTful API for self-service enrolment// Conference Proceedings. – M.:IEEE Xplore / AICT, 2017. – P. 1–4.

49 За 2019 год в мире скомпрометировано 13,7 млрд записей персональных данных. https://www.infowatch.ru/company/presscenter/news/26270. 23.05.2020.

50 Исследование TAdviser и Microsoft. http://www.tadviser.ru/index.php 28.08.2019.

51 Компания PositiveTechnologies, Актуальные киберугрозы: итоги 2019 года. https://www.ptsecurity.com/ru-ru/research/analytics/cybersecurity-threatscape-2019/. 19.03.2020.

52 Сборник исследований по практической безопасности PositiveResearch. 2019 https://www.ptsecurity.com/upload/corporate/ru-ru/analytics/Positive-Research-2019-rus.pdf 19.03.2020.

53 Data Breach Investigations Report 2019. https://www.key4biz.it/wp-content/uploads/2019/05/2019-data-breach-investigations-report.pdf. 21.02.2020.

54 Исследование уровня информационной безопасности в компаниях России и СНГ за 2019 год. https://searchinform.ru/research-2019/. 10.04.2020.

55 5 простых шагов для успешной двухфакторной– Infobip. https://www. infobip.com/blog/five-easy-steps-guide-for-successful-two-factor-authentication. 10.09.2019.

56 Соколин Д.Т., Тимохович А.С.Методы комплексного обеспечения безопасности SQL – сервера от атак пита SQL – инъекций// Автоматика. Вычислительная техника. Academy, 2017.- Т. 3 (18). –С. 7–19.

57 IBM Security Guardium. https://www.ibm.com/security/data-security/ guardium. 10.09.2019.

58 Компания Imperva. https://www.pacifica.kz/catalog/zashchita-bd/imperva-database-security. 10.09.2019.

59 Imperva Secure Sphere Data Security. https://www.imperva.com/ resources /datasheets/DS_SecureSphere_Data-Security.pdf. 18.09.2019.

60 McAfee DataCenter Security Suite for Databases. https://www.mcafee. com/enterprise/en-us/assets/data-sheets/ds-data-center-security-suite-databases.pdf. 18.09.2019.

61 McAfee Vulnerability Manager for Databases. http://b2b-download. mcafee.com/products/evaluation/database_security/vulnerability_manager_for_databases/vmd_4.5.0/mcafee_vulnerability_manager_for_databases_product_guide_4_5.pdf.25.09.2019.

62 Trustwave AppDetectivePRO. https://www.trustwave.com/en-us/resources/ library/documents/trustwave-appdetectivepro. 25.09.2019.

63 DbProtect. https://www3.trustwave.com/software/Database-Security/DbProtectUserGuide_649.pdf. 30.09.2019.

64 FUDOSECURITY. https://www.fudosecurity.com.30.09.2019.

65 Install Google Authenticator. https://support.google.com/ accounts/ answer/ 1066447?co=GENIE.Platform%3DAndroid&hl=en. 30.09.2019.

66 Secure Authentication With the Duo Mobile App. https://duo.com/ product/multi-factor-authentication-mfa/duo-mobile-app. 13.10.2019.

67 Использование приложения Microsoft Authenticator. https:// support. microsoft.com/ru-kz/help/4026727/microsoft-account-how-to-use-the-microsoft-authenticator-app. 13.10.2019.

68 FreeOTP Authenticator. https://www.securitylab.ru/software/498773.php. 13.10.2019.

69 Authy: 2FA and Password less Login. https://www.twilio.com/docs/authy. 21.10.2019.

70 Включить двухфакторную аутентификацию. https://yandex.kz /support/ passport/authorization/twofa-on.html/23.10.2019.

71 Evseev S. P., Tomashevskyy B. P. Two–factor authentication methods threats analysis// Radio Electronics, Computer Science, Control, e–ISSN 2313–688X –2015, № 1. –P.52–59.

72 Byung Rae Cha, Kyungjun Kim, Hyun Shik Na. «Random password generation of OTP system using changed location and angle of finger print features»// Computer and Information Technology CIT 200S. Sth IEEE International Conference, 2009.–P.420–425.

73 O. Ussatova, S. Nyssanbayeva, W. Wojcik. Development of an authentication model based on the second factor in an automated control system// Вестник КБТУ. –Алматы, 2019.–Т.16. –С.115–118.

74 Fatih Sulak «Cryptographic Random Testing of Block Ciphers and Hash Functions»// Publication of the Middle East Technical University Ph.D Examinations, 2011. – P. 7–39.

75 Авезова Я. Э. «Современные подходы к построению хеш–функций на примере финалистов конкурса sha-3»// Вопросы гипербезопасности. –М., 2015. –№3 (11). –С.60–67.

76 P. Prisha, H. Neo, T. Ong and C. TeoE–Commerce Security and Identity Integrity: The Future of Virtual Shopping// Advanced Science Letters, –2017. Vol. 23, № 8 – P. 7849–7852.

77 Alata E., Nicomette V., Kaaniche M., Dacier M., Herrb M.: Lessons learned from the deployment of a high–interaction honeypot, in Proc. Dependable Computing Conference (EDCC06). – Portugal: Coimbra. 2006. –P. 39 – 46.

78 Begimbayeva Yenlik, Ussatova Olga, Biyashev Rustem, Nyssanbayeva Saule Development of an automated system model of information protection in the cross-border exchange// Cogent Engineering Journal. – 2020. DOI: 10.1080/ 23311 916. 2020.1724597. –P. 1–13.

79 Folasade Ayankoya, Blaise Ohwo. Brute-Force Attack Prevention in Cloud Computing Using One-Time Password and Cryptographic Hash Function// International Journal of Computer Science and Information Security (IJCSIS). – 2019,– Vol. 17, № 2. – P. 7–19.

80 Купавский А. Б., Титова М. В. Дистанционные числа Рамсея// Докл. РАН., 2013, Т. 449, № 3. – С. 267–270.

81 Шмаков, М. С. Разработка алгоритма генерации временных кодов для идентификации подлинности продукта / М. С. Шмаков, А. Н. Кошевая // Труды БГТУ. Сер. 4, Принт и медиатехнологии. – Минск: БГТУ, 2017. – № 2 (201). – С. 44–48.

82 Randomness.https://www.wikiwand.com/en/Randomness. 18.12.2019.

83 Olga Ussatova, Saule Nyssanbayeva. Generators of one-time two-factor authentication passwords// Informatyka, Automatyka, Pomiary w Gospodarcei Ochronie Środowiska. – Poland, 2019. № 2. –P. 60–64.

84 Какой пароль защитит от взлома, или энтропия на службе секретности. http://samag.ru/uart/more/53. 18.12.2019.

85 Усатова О.А., Нысанбаева С.Е. Обеспечение защиты информационной системы с помощью двухфакторной аутентификации// Науч. конф. «Современные проблемы информатики и вычислительных технологий» – Алматы, 2019. –С.337–343.

86 Бегимбаева Е.Е., Усатова О.А.,Бияшев Р.Г., Нысанбаева С.Е., Вуйцик В. Разработка модулей для защиты информации в автоматизированной системе с применением разграничения доступа// IV междунар. науч. – практ.конф. «Информатика и прикладная математика», посвященная 70-летнему юбилею профессоров Биярова Т.Н., Вальдемара Вуйцика и 60-летию профессора Амиргалиева Е.Н. – Алматы, 2019. – С. 595–602.

87 Усатова О.А. Клиент-серверная система защиты информации на основе двухфакторной аутентификации// междунар. науч. – практ. конф. «Актуальные проблемы информационной безопасности в Казахстане». – Алматы, 2020. – С. 243–248.

88 S. V. Diasamidze, E. Yu. Kuzmenkova, D. A. Kuznetsov, A. R. Sarkisyan. V Implementation of the Role Based Access Control in Application for Mobile Device on the Android OS Platform// Интеллектуальные технологии на транспорте, 2016, № 1. С. 21–26.

89 Усатова О.А., Нысанбаева С.Е., Исследование и разработка модели защиты базы данных информационной системы// Вестник КазНУим.аль–Фараби, Алматы, 2019. – № 4 (104), – С.95–106.

90 Sholeh A. T., Gunadhi E. and Supriatna A. D. Mengamankan Skrip Pada Bahasa Pemograman Php Dengan Menggunakan Kriptografi Base64 J. Algorithm. Sekol// Tinggi Teknol. –Garut, 2013. –P.1–9.

91 The Catch 22 of Base64: Attacker Dilemma from a Defender Point of View. https://www.imperva.com/blog/the-catch-22-of-base64-attacker-dilemma-from-a-defender-point-of-view.23.12.2019.

92 Усатова О.А., Науменко В.В. «Статистические исследования инфраструктурной платформы с использованием систем защиты данных»//VIII междунар. науч. – метод. конф. посвященной 90-летнему юбилею Казахского национального педагогического университета имени Абая. – Алматы, 2018. – С. 113-116.

93 А.А. Акулов, В.В. Галушка. Подмена ssl–сертификатов как средство перехвата зашифрованного трафика//«Символ науки» международный научный журнал. –Уфа, 2018. –№ 1–2. –С.19–21.

94 М.Я. Беккер, А.О. Терентьев, Ю.А. Гатчин, Н.С. Кармановский Использование цифровых сертификатов и протоколов ssl/tls для шифрования данных при облачных вычислениях// Научно-технический вестник Санкт–Петербургского государственного университета информационных технологий, механики и оптики. – Санкт–Петербург, 2011. – № 4 (74). – С.125–130.

95 Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. Приемы объектно–ориентированного проектирования. Паттерны проектирования. Санкт–Петербург: Питер, 2007. – 366 с.

96 Фаулер М. UML. Основы. Краткое руководство по стандартному языку объектного моделирования. Санкт-Петербург: Питер, 2011. – 192 с.

97 CryptoJS. https://cryptojs.gitbook.io/docs. 18.01.2020.

98 Руководство по Node.js.https://metanit.com/web/nodejs/. 18.01.2020.

99 iOS and Android already occupy 99.9% of the market for mobile operating systems,2018г. https://www.ixbt.com/news/2018/02/24/ios-android-99-9.html.28.02.2019.

100 O. Ussatova, S. Nyssanbayeva, W. Wojcik Two-factor authentication algorithm implementation with additional security parameter based on mobile application// International Conference on Wireless Communication, Network and Multimedia Engineering (WCNME2019). – Guilin, China, 2019. – Vol. 89. – P. 84–86.

101 MySQL and MongoDB – when and what is better to use. https: //habr. com/ru/ post/ 322532/. 20.02.2020.

102 Шмидт И.А., Васенев Н.В. Выбор оптимальной JSON–модели для хранения результатов испытаний// Фундаментальные исследования. – 2016. – № 11 (3). – С. 620–625.

103 O. Ussatova, S. Nyssanbayeva, W. Wojcik Software implementation of two-factor authentication to ensure security when accessing an information system//Вестник КазНУ им.аль-Фараби. – Алматы, 2019. – С.87–95.

104 Eric Windmill, Ray Rischpater. Flutter in Action. – United States of America: Manning, 2020. – P. – 368.

## ПРИЛОЖЕНИЕ А
## Список публикаций

1 Нысанбаева С.Е., Усатова О. А. «Способы обеспечения безопасности информации в базах данных»// Вестник КазНИТУ им. Сатпаева.– Алматы, 2018. – № 2. – С. 66–70.

2 Нысанбаева С.Е., Усатова О.А. «Возможное применение больших данных в системе образования»// Матер. междунар. науч. практ. конф. «Инновационные технологии на транспорте: образование, наука, практика» в рамках реализации Послания Президента РК Н. Назарбаева «Новые возможности развития в условиях четвертой промышленной революции» – Алматы, 2018. – С. 95–99.

3 Нысанбаева С.Е., Усатова О.А. «Криптографическая защита в автоматизированных системах»// Науч. конф. «Современные проблемы информатики и вычислительной технологий». – Алматы, 2018. – С. 220–223.

4 Нысанбаева С.Е., Усатова О.А. «Двухфакторная аутентификация в автоматизированной системе управления»// III Междунар. науч. конф. «Информатика и прикладная математика». – Алматы, 2018. – С. 239–242.

5 Усатова О.А., Науменко В.В. «Статистические исследования инфраструктурной платформы с использованием систем защиты данных»// VIII междунар. науч.-метод. конф. посвященной 90-летнему юбилею Казахского национального педагогического университета имени Абая. – Алматы, 2018. – С. 113–116.

6 O. Ussatova, S. Nyssanbayeva, W. Wojcik. «Development of an authentication model based on the second factor in an automated control system»// Вестник КБТУ. – Алматы, 2019. –Т.16. – С.115–118.

7 S. Nyssanbayeva, W. Wojcik, O. Ussatova. «Algorithm for generating temporary password based on the two–factor authentication model»// Przegląd Elektrotechniczny. – Polan, 2019. –№ 5. – P. 101–106.

8 O. Ussatova, S. Nyssanbayeva, W. Wojcik. «Two-factor authentication algorithm implementation with additional security parameter based on mobile application »// International Conference on Wireless Communication, Network and Multimedia Engineering (WCNME2019). –Guilin, China, 2019. – Vol. 89. – P. 84–86.

9 O. Ussatova, S. Nyssanbayeva, W. Wojcik. « Software implementation of two-factor authentication to ensure security when accessing an information system» // Вестник КазНУ им.аль-Фараби. –Алматы,2019. – С.87–95.

10 Olga Ussatova, Saule Nyssanbayeva. «Generators of one-time two-factor authentication passwords»// Informatyka, Automatyka, Pomiary w Gospodarcei Ochronie Środowiska. – Poland, 2019. № 2. – P. 60–64.

11 Усатова О.А., Нысанбаева С.Е. «Обеспечение защиты информационной системы с помощью двухфакторной аутентификации»// науч. конф. «Современные проблемы информатики и вычислительных технологий» – Алматы, 2019. –С. 337–343.

12 Begimbayeva Yenlik, Ussatova Olga, Biyashev Rustem, Nyssanbayeva Saule. «Development of an automated system model of information protection in the cross-border exchange»// Cogent Engineering Journal. – 2020. DOI: 10.1080/ 23311 916. 2020.1724597. –P. 1–13.

13 Бегимбаева Е.Е., Усатова О.А., Бияшев Р.Г., Нысанбаева С.Е., Вуйцик В., «Разработка модулей для защиты информации в автоматизированной системе с применением разграничения доступа»// IV междунар. науч.– практ. конф. «Информатика и прикладная математика», посвященная 70–летнему юбилею профессоров Биярова Т.Н., Вальдемара Вуйцика и 60–летию профессора Амиргалиева Е.Н. – Алматы, 2019. – С. 595–602.

14 Усатова О.А., Нысанбаева С.Е. «Исследование и разработка модели защиты базы данных информационной системы»// Вестник КазНУ им.аль–Фараби, Алматы, 2019. – № 4 (104), – С.95–106.

15 Усатова О.А. «Клиент-серверная система защиты информации на основе двухфакторной аутентификации»// междунар. науч.– практ. конф. «Актуальные проблемы информационной безопасности в Казахстане». – Алматы, 2020. – С. 243–248.

## ПРИЛОЖЕНИЕ Б
## Полученные авторские свидетельства

«WORKSCOPYRIGHT» – Цифровая сертификация, юридически признанная во всем мире, для регистрации авторских прав авторов, New York – NY-USA, №712144534 от 2018.10.01.Сертификат регистрации авторского права алгоритма «Двухфакторная аутентификация в автоматизированной системе управления».

Certificate of Registration

TECHNICAL WORK

| | |
|---|---|
| REGISTRATION NUMBER: | 712144534 |
| TIMESTAMP: | 2018-10-01 15:21:25 GMT |
| TITLE: | ДВУХФАКТОРНАЯ АУТЕНТИФИКАЦИЯ В АВТОМАТИЗИРОВАННОЙ СИСТЕМЕ УПРАВЛЕНИЯ |
| WORK'S FILE: | autentifikacia-v-asu-usatova-oa.docx[20181001_152125].zip |
| REGISTERED BY: | USSATOVA OLGA ALEKSANDROVNA (AUTHOR) |
| YEAR OF CONCLUSION: | 2018 |
| LANGUAGE: | RUSSIAN |
| AUTHOR (BIRTH / COUNTRY): | USSATOVA OLGA ALEKSANDROVNA (1979-04-22 / KAZAKHSTAN - ҚАЗАҚСТАН) |

:: WORK'S eDNA - ELECTRONIC IDENTIFIER ::

SHA512: 8fec3666690655c3a308a5f7e3befac5cbac956ef327e4cdf4182f27f7b942537b3d32eae06325de dcd1bd0a9cb41f04b2b9d209439da3d855a8e611f7de3eac

WORKS COPYRIGHT
YOUR TALENT PROTECTED

Generated on 2018-10-01 15:23:18 GMT

Рисунок Б.1 – Сертификат регистрации авторского права

Авторское свидетельство о внесении сведений в государственный реестр прав на объекты, охраняемые авторским правом РК, № 4330 от 28 июня 2019г., Система аутентификации с использованием второго фактора для контроля доступа к данным –Security Code of the 2FA.

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ

РЕСПУБЛИКА КАЗАХСТАН

**СВИДЕТЕЛЬСТВО**

**О ВНЕСЕНИИ СВЕДЕНИЙ В ГОСУДАРСТВЕННЫЙ РЕЕСТР ПРАВ НА ОБЪЕКТЫ, ОХРАНЯЕМЫЕ АВТОРСКИМ ПРАВОМ**

**№ 4330 от «28» июня 2019 года**

Фамилия, имя, отчество, (если оно указано в документе, удостоверяющем личность) автора (ов):
**УСАТОВА ОЛЬГА АЛЕКСАНДРОВНА, НЫСАНБАЕВА САУЛЕ ЕРКЕБУЛАНОВНА**

Вид объекта авторского права: **программа для ЭВМ**

Название объекта: **Система аутентификации с использованием второго фактора для контроля доступа к данным – Security Code of the 2FA**

Дата создания объекта: **10.05.2019**

Подписано ЭЦП

Оспанов Е.К.

Рисунок Б.2 –Авторское свидетельство

# ПРИЛОЖЕНИЕ В
## Акты о внедрении результатов диссертационной работы

Акт о внедрении результатов диссертационной работы в Институт информационных и вычислительных технологий Комитета науки МОН РК.

УТВЕРЖДАЮ
Зам. генерального директора
Института информационных
и вычислительных технологий
PhDдоктор
_____ Мамырбаев О.Ж.
«_21_» _июля_ 2020

**АКТ**
о внедрении результатов диссертационной работы
Усатовой Ольги Александровны

Экспертная комиссия «Института информационных и вычислительных технологий» (ИИВТ) Комитета науки МОН РК в составе
председателя: д.т.н., ассоциированный профессор Калижанова А.У. заместитель генерального директора по связям с общественностью;
членов: д.т.н., ассоциированный профессор Нысанбаева С.Е. ГНС ИИВТ;
к.т.н., Капалова Н.А.ВНС ИИВТ;
к.т.н., доцент Юничева Н.Р. ученый секретарь ИИВТ.

составили настоящий акт о том, что результаты диссертационной работы «Разработка и исследование алгоритма аутентификации пользователей информационно-коммуникационных систем» Усатовой О.А. были получены при выполнении проектов РГП на ПХВ «ИИВТ» КН МОН РК, № гос. регистрации - 0115РК00534), источник финансирования Комитет науки МОН РК:
– программно-целевого финансировании (ПЦФ) КН МОН РК «Разработка программных и программно-аппаратных средств для криптографической защиты информации при ее передаче и хранении в инфокоммуникационных системах и сетях общего назначения» на 2018-2020 годы;
– грантового финансирования (ГФ) КН МОН РК «Разработка казахстанского сегмента защищенного трансграничного информационного взаимодействия» на 2018-2020 годы.
Результаты включены в отчёты проектов ПЦФ за 2018, 2019 и ГФ за 2020 годы.
Краткое содержание внедренных результатов:
1. Проведены исследования методов и средств защиты информации в информационно-коммуникационных системах;
2. Разработан алгоритм аутентификации пользователей на основе одноразового пароля в информационно-коммуникационной системе;
3. Разработана и внедрена программная реализация системы двухфакторной аутентификации в информационно-коммуникационной системе.

Материалы к настоящему акту были рассмотрены на Ученом Совете института (протокол №9 от 21 июля 2020 год).

Председатель комиссии _____ Калижанова А.У.
Члены комиссии _____ Нысанбаева С.Е.
_____ Капалова Н.А.
_____ Юничева Н.Р.

Рисунок В.1 –Акт внедрения ИИВТ КН МОН РК

Акт о внедрении результатов диссертационной работы в ТОО «Digital Media Center».

ТОО "Digital media center"
050051, г.Алматы
ул.Кокшинаки 5, офис 16
тел: +7 (727) 354 16 20

DMC
Digital Media Center

ТОО "Digital media center"
050051, г.Алматы
ул.Кокшинаки 5, офис 16
тел: +7 (727) 354 16 20

**АКТ**

о внедрении результатов диссертационной работы

Усатовой Ольги Александровны

Настоящим подтверждаем, что результаты диссертационного исследования Усатовой О.А. на тему: «Разработка и исследование алгоритма аутентификации пользователей информационно-коммуникационных систем» являются актуальными и имеют практическое значение. Идентификация и аутентификация субъектов является одним из основных средств защиты информационного объекта от постороннего вмешательства.

Разработанный алгоритм двухфакторной аутентификации позволил повысить уровень защиты электронной информации и надежность функционирования информационной системы ТОО «Digital Media Center». Подходы и архитектура созданного клиент-серверного приложения Усатовой О.А. под названием «Security Code of the 2FA» построены грамотно и отвечает требованиям практического применения. Данное приложение соответствует современным средствам защиты от несанкционированного доступа.

Клиент-серверное приложение «Security Code of the 2FA» успешно подключенно как дополнительный модуль защиты и используется в информационной системе управления ТОО «Digital Media Center».

Директор ТОО «Digital Media Center» Тихонова Л.В.

Рисунок В.2 – Акт внедрения ТОО «Digital Media Center»

## ПРИЛОЖЕНИЕ Г

### Программное обеспечение системы защиты информации при аутентификации пользователя на основе одноразового ключа

```
constexpress = require('express');
const session = require('express-session');
const bodyParser = require('body-parser');
var CryptoJS = require("crypto-js");
const SHA256 = require('crypto-js/sha256');
const request = require('request-promise');
const generator = require('./generators/generators.js')
const mongo = require('mongodb');
const MongoClient = require('mongodb').MongoClient;
const mongoClient = new
MongoClient("mongodb+srv://admin:Mdb12812122424@@cluster0-
o83lo.mongodb.net/test?retryWrites=true", { useNewUrlParser: true });
let dbClient;
function getRandomInt(min, max) { // функция для получения рандомного числа
по заданному диапазону
return Math.floor(Math.random() * (max - min + 1)) + min;
}
const app = express(); /* подключение модулей */
app.use(session({ // настройка сессий
secret: 'newSecret',
resave: false,
saveUninitialized: false,
}));
let wordsArr = []

for (let i = 0; i < 256; i++) {
        wordsArr[i] = []
        for (let j = 0; j < 256; j++) {
                wordsArr[i][j] = generator.generateWord()
        }}
let arrFunc = []

for (let i = 0; i < 256; i++) {
        arrFunc[i] = []
        for (let j = 0; j < 256; j++) {
                arrFunc[i][j] = generator.generateFunction()
        }
}
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
```

```
app.use((req, res, next) => {
console.log(`${req.method} ${req.url}`);
next();
});
mongoClient.connect((err, client) => { // подключение БД и запуск сервера
if (err) return console.log(err);
dbClient = client;
  app.locals.usersCollection = client.db('mobileDB').collection('users');
  app.locals.resourcesCollection = client.db('mobileDB').collection('resources')
  app.locals.wordsCollection = client.db('mobileDB').collection('words')
  app.locals.funcCollection = client.db('mobileDB').collection('trigf')
app.listen(process.env.PORT || 3000, () => {
console.log('Сервер проекта запущен');
});
});
function createCode(login,apps) {
const systemDate = new Date();
  const date =
"+systemDate.getFullYear()+(systemDate.getMonth()+1)+systemDate.getDate()+sy
stemDate.getHours()+systemDate.getMinutes()+systemDate.getSeconds();
let i = getRandomInt(0, 9);
let j = getRandomInt(0, 9);
const secretStr = wordsArr[i][j]; //выбор слова из массива по индексу
  const hash = SHA256(login+systemDate+secretStr); // разбиение на слова
строки состоящей из входных данных
hashStr = hash.toString(CryptoJS.enc.SHA256).toUpperCase();
let values = { // формирование переменных на основании hashStr
a: hashStr.substr(hashStr.length-2,2),
b: hashStr.substr(hashStr.length-4,2),
c: hashStr.substr(hashStr.length-6,2),
x: hashStr.substr(9,2),
y: hashStr.substr(11,2),
    p1: hashStr.substr(hashStr.length-8,2),
    p2: hashStr.substr(hashStr.length-10,2),
}
  values = { // перевод в десятичное значение
    a: parseInt(values.a,16),
b: parseInt(values.b,16),
c: parseInt(values.c,16),
x: parseInt(values.x,16),
y: parseInt(values.y,16),
    p1: parseInt(values.p1,16),
    p2: parseInt(values.p2,16),
  }
```

```
  i = parseInt(hashStr.substr(0,2),16)%10;
  j = parseInt(hashStr.substr(2,2),16)%10;  // определение индекса функции (mod
10 т.к. массив 10x10)
let selectedFunction = arrFunc[i][j]; // выбор функции из массива по
определенному индексу
  let f = eval(selectedFunction); // передаю в f строчную функцию в виде
программного кода
  let firstBorder = selectedFunction.indexOf('(')+1; // определение границы
указания аргументов функции
let secondBorder = selectedFunction.indexOf(')');
let arguments = selectedFunction.substring(firstBorder,secondBorder); //
получаюаргументыфункцииввидестроки
let res;
eval('const {' + arguments + '} = values;' + 'res = f('+ arguments +')');
let tempPass; // объявление временного пароля
if (res.toString().indexOf('.') > 0) { // проверка на целостность результата и на
нужное количество цифр после запятой
    let pointIndex = res.toString().indexOf('.'); // временный пароль определяется
числами после запятой
if (res.toString().substring(pointIndex+1).length < 8) {
for(let i = 0; i < 8; i++) {
res += "1";
      }
    } else {
tempPass = res.toString().substr(pointIndex+1,6);
    }
tempPass = res.toString().substr(pointIndex+5,6); // начинаяс 5-йпозиции, длина 6
цифр
  } else {
tempPass = 101010;
  }
return tempPass;
} // функция генерации пароля
app.post('/createCode', (req, res) => { // генерация кода
const { login,address } = req.body;
const options = { // настройка запроса
method: 'POST',
uri: address + '/checkUser',
body: {
login: login,
    },
json: true,
  }
request(options) // проверка наличия пользователя на сайте
```

```
.then((response) => {
if (response.userAvailable) {
// console.log('Такой аккаунт есть на сайте');
const collection = app.locals.usersCollection;
collection.findOne({login: login}, (err, result) => { // поиск в собственной БД
if (result) {
let newCode = createCode(login);
collection.updateOne({_id: new mongo.ObjectId(result._id)}, {$set: {code:
newCode}}) // измененияbазы
res.send({secretCode: newCode}); // отправка кода
        }
else {
let newCode = createCode(login); // генерация кода
let newUser = {
login: login,
code: newCode,
          }
collection.insertOne(newUser, (err, result) => { // дабавление в базу
if (err) return console.log(err);
res.send({secretCode: newCode}); // отправка кода
          })        }
       })
     } else
{
res.send({secretCode: false});
     }
   })
   .catch((err) => {
if (err) return console.log(err);
   })
})
app.post('/checkCodeWithData', (req, res) => { // проверка кода в базе и отправка
const { login } = req.body;
const collection = app.locals.usersCollection;
collection.findOne({login: login}, (err, result) => {
if (err) return console.log(err);
res.send({codeS: result.code});
  });
})
app.post('/registration', (req, res) => {
const { login,password,address } = req.body;
const resource = { login: login, password: password, address: address }
const collection = app.locals.resourcesCollection;
collection.insertOne(resource, (err, result) => { // добавление в БД
```

```
if (err) return console.log(err);
res.send({ registrationSuccess: true });
  });
})
app.post('/autentification', (req,res) => {
const { login,password } = req.body;
const user = { login: login, password: password }
const collection = app.locals.resourcesCollection;
collection.findOne(user, (err, result) => { // поиск в БД
if (err) return console.log(err);
if (result) {
      req.session.authorized = true;
      req.session.userlogin = login;
console.log(req.session)
res.send({ userExists: true, userLogin: req.session.userlogin });
    }
else
{
res.send({ userExists: false });
    }
    })
})
app.get('/checkSession', (req, res) => { // проверка сессии
if (req.session.authorized) {
console.log('Сессия есть')
res.send({ userExists: true, userLogin: req.session.userlogin });
  } else {
res.send({ userExists: false });
console.log('Сессиинет')
  }
});
app.get('/getResources', (req, res) => {
const collection = app.locals.resourcesCollection;
collection.find({}, {login: false, password: false}).toArray((err, result) => {
if (err) return console.log(err);
console.log(result)
res.send({ resourcesList: result})
})})
process.on("SIGINT", () =>{ // вывод информации о запросах
dbClient.close();
process.exit();
});
let funcVariblesList = []
let funcComponents = []
```

```
let expressions = ['+', '-', '/', '*']
function getRandomInt(min, max) { // функция для получения рандомного числа по заданному диапазону
return Math.floor(Math.random() * (max - min + 1)) + min;
}
module.exports.generateWord = function() { // Генератор слов
	let chars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz'
	let wordLength = getRandomInt(5, 10)
	let word = ''

	for (let i = 0; i <= wordLength; i++) {
		let charIndex = getRandomInt(0, chars.length - 1)
		word += chars[charIndex]
	}
	return word
}
module.exports.generateFunction = function() {
	funcVariblesList = []
	let variablesList = ['a', 'b', 'c', 'x', 'y', 'p1', 'p2']
	let variablesCount = getRandomInt(1, variablesList.length - 1)

	for (let i = 0; i <= variablesCount; i++) {
		let variableIndex = getRandomInt(0, variablesList.length - 1)
		funcVariblesList.push(variablesList[variableIndex])
		variablesList.splice(variableIndex, 1)
	}
	return getComponents(funcVariblesList)
}
function getComponents(variablesList) {
	funcComponents = []
	let components = ['Math.sin(*)', 'Math.cos(*)', 'Math.tan(*)', '(1/Math.tan(*))', '(*)']
	let componentsCount = components.length - 1

	variablesList.forEach(item => {
		let componentIndex = getRandomInt(0, componentsCount)
		funcComponents.push(components[componentIndex].replace('*', item))
	})
		return getFunction(funcComponents)
}
function getFunction() {
	// формированиеблокапеременных
	let variablesBlock = `(${funcVariblesList.join(',')}) => `
```

```
        // формированиетелафункции
        let seriesCount = getRandomInt(1, 3)
        let seriesList = []
                for (let seriesIndex = 0; seriesIndex <= seriesCount; seriesIndex++) { //
будет 2 ряда
                let seriesBlock = ''
                let funcComponentsCount = funcComponents.length - 1

                for (let j = 0; j <= funcComponentsCount; j++) {
                        let componentIndex = getRandomInt(0, funcComponentsCount)
                        let expressionIndex = getRandomInt(0, expressions.length - 1)

                        if (j != funcComponentsCount) {
                                seriesBlock += `(${funcComponents[componentIndex]})`
+ expressions[expressionIndex]
                        } else {
                                seriesBlock += `(${funcComponents[componentIndex]})`
                        }
                }
                seriesList.push(`(${seriesBlock})`)
        }
        let funcBlock = '{return('
                for (let i = 0; i<= seriesList.length - 1; i++) {
                let expressionIndex = getRandomInt(0, expressions.length - 1)

                if (i != seriesList.length - 1) {
                        funcBlock += seriesList[i] + expressions[expressionIndex]
                } else {
                        funcBlock += seriesList[i]
                }
        }

        funcBlock += ')}'
        return func = variablesBlock + funcBlock
}
Function getRandomInt(min, max) { // функция для получения рандомного
числа по заданному диапазону
        return Math.floor(Math.random() * (max - min + 1)) + min;
}
import React from 'react';
import { View, Text, Button, ScrollView } from 'react-native';
import { createStackNavigator, createAppContainer } from "react-navigation";
import { MainScreen } from './components/MainScreen.js'
import { RegistrationScreen } from './components/RegistrationScreen.js'
```

```
import { LoginScreen } from './components/LoginScreen.js'
import { AccountScreen } from './components/AccountScreen.js'
import { ResourcesListScreen } from './components/ResourcesListScreen.js'
import { CreateCode } from './components/CreateCode.js'
import { Polygon } from './components/Polygon.js'
const AppNavigation = createStackNavigator({
  Main: {
screen: MainScreen
  },
  Registration: {
screen: RegistrationScreen
  },
  Login: {
screen: LoginScreen
  },
  Account: {
screen: AccountScreen
  },
  ResourcesList: {
screen: ResourcesListScreen
  },
  GenerateCode: {
screen: CreateCode
  }
}, {
initialRouteName: 'MainScreen'
})
const AppContainer = createAppContainer(AppNavigation)
export default class App extends React.Component {
render() {
return(
<AppContainer />
    )  }
}
import React from 'react';
import { StyleSheet, View, ScrollView, Text, Button, TextInput, Alert,
TouchableOpacity} from 'react-native';
import axios from 'axios';
class AccountScreen extends React.Component {
static navigationOptions = {
title: 'Профиль',
headerStyle: {
backgroundColor: '#7a42f4',
    },
```

```
headerTintColor: '#fff',
headerTitleStyle: {
fontWeight: 'bold',
    },
  }
state = {
userLogin: '',
userExists: false,
  }
componentDidMount = () => { // проверка сессии пользователя
axios.get('http://192.168.43.22:3000/checkSession').then(response => {
if (response.data.userExists)
{
this.setState((prevState) => {
return { userExists: response.data.userExists, userLogin: response.data.userLogin }
        })
      }
    }).catch(err => {
Alert.alert(
        'Ошибка',
        `${err}`,
      )     })
  }
render() {
return(
<View>
<Text>Добропожаловать, {this.state.userLogin}!</Text>
</View>
    )  }
}
export { AccountScreen }
import React from 'react';
import { TouchableOpacity, StyleSheet, Text, View, Button, Alert, TextInput,
Modal, Animated, Dimensions } from 'react-native';
import axios from 'axios';
let { width } = Dimensions.get('window')
let available_width = width * 0.9
let progress = new Animated.Value(0)
class CreateCode extends React.Component {
static navigationOptions = {
title: 'Генерациякода',
headerStyle: {
backgroundColor: '#7a42f4',
    },
```

```
headerTintColor: '#fff',
headerTitleStyle: {
fontWeight: 'bold',
    },  }
state = { // глобальные переменные для компонентов (состояние)
login: '',
password: '',
modalVisible: false,
code: ''
  }
openModal = () => {
this.setState((prevState) => {
return { modalVisible: true }
    })
  }
closeModal = () => {
this.setState((prevState) => {
return { modalVisible: false }
    })
  }
checkModalStatus = () => {
return this.state.modalVisible
  }
getProgressStyles = () => {
var animated_width = progress.interpolate({
inputRange: [0, 50, 100],
outputRange: [0, available_width / 2, available_width]
    });
return {
width: animated_width,
height: 50, //height of the progress bar
backgroundColor: '#7a42f4'
    }
  }
getCode = () => { // нажатие на кнопку "отправка"
const { login,modalVisible } = this.state;
if (login) {
axios.post('http://192.168.43.22:3000/createCode', { // запрос на получение кода
login: login,
address: this.props.navigation.state.params.address
        })
        .then((response) => {
this.setState({ code: response.data.secretCode })
progress.setValue(0)
```

```
this.openModal()
Animated.timing(progress, {
duration: 20000,
toValue: 100
        }).start(() => {
console.log(modalVisible)
if (this.checkModalStatus()) {
this.getCode()
          }
        });
       })
       .catch((error) => {
Alert.alert(
          'Ошибка',
          `${error}`
        )
return console.log(error);
       })
   }
else
{
Alert.alert(
       'Ошибка',
       'Сперва нужно ввести данные'
     )
   }
  }
render() { // отрисовка компонентов
const { login,code } = this.state;
return (
<View style={styles.container}>
<TextInput

style={styles.input}
underlineColorAndroid = "transparent"
autoCapitalize = "none"
onChangeText={(login) => {this.setState({login})}}
placeholder="Логин"
placeholderTextColor = "#9a73ef"
value={login}
       />
<TouchableOpacity
style = {styles.submitButton}
onPress={this.getCode}
```

```
>
<Text style = {styles.submitButtonText}>Отправить</Text>
</TouchableOpacity>
<Modal
animationType="fade"
transparent={true}
visible={this.state.modalVisible}
onRequestClose={() => {
Alert.alert('Modal has been closed.');
        }}
>
<View style={styles.modalWrapper}>
<View style = {styles.modalContainer}>
<View>
<View>
<Animated.View
style={[this.getProgressStyles.call(this)]}
>
</Animated.View>
</View>
</View>
<View style={styles.modalText}>
<Text>Код:</Text>
<Text>{code}</Text>
</View>
<TouchableOpacity
style = {styles.modalButton}
onPress={this.closeModal
            }>
<Text style = {styles.submitButtonText}>Закрыть</Text>
</TouchableOpacity>
</View>
</View>
</Modal>
</View>
    );
  }
}
const styles = StyleSheet.create({ // стили компонентов
mainContainer: {
backgroundColor: '#7a42f4',
  },
container: {
backgroundColor: 'white'
```

```
  },
input: {
margin: 15,
marginTop: 30,
padding: 10,
height: 40,
borderColor: '#7a42f4',
borderWidth: 1
  },
submitButton: {
backgroundColor: '#7a42f4',
padding: 10,
margin: 15,
height: 40,
  },
modalButton: {
backgroundColor: '#7a42f4',
height: 40,
padding: 10,
width: '100%',
alignItems: 'center',
  },
modalText: {
flex: 1,
justifyContent: 'center',
alignItems: 'center',
  },
submitButtonText:{
color: 'white',
  },
modalWrapper: {
flex: 1,
flexDirection: 'column',
justifyContent: 'center',
alignItems: 'center',
height: '100%',
backgroundColor: 'rgba(0,0,0,0.7)',
  },
modalContainer: {
flexDirection: 'column',
justifyContent: 'space-between',
width: '90%',
height: '30%',
backgroundColor: 'white',
```

```
},
});
export { CreateCode }
import React from 'react';
import { StyleSheet, View, ScrollView, Text, Button, TextInput, Alert,
TouchableOpacity} from 'react-native';
import { AccountScreen } from './AccountScreen.js'
import axios from 'axios';
class LoginScreen extends React.Component {
static navigationOptions = {
title: 'Вход',
headerStyle: {
backgroundColor: '#7a42f4',
    },
headerTintColor: '#fff',
headerTitleStyle: {
fontWeight: 'bold',
    },
  }
state = {
login: '',
password: '',
userExists: false,
userLogin: '',
  }
componentDidMount = () => { // проверка сессии пользователя
console.log(this.navigationOptions.title)
axios.get('http://192.168.137.1:3000/checkSession').then(response => {
console.log('Это')
console.log(response.data)
console.log(response.data.userExists)
if (response.data.userExists) {
this.setState((prevState) => {
return
{
userExists: response.data.userExists, userLogin: response.data.userLogin }
        })
      }
    }).catch(err => {
Alert.alert(
        'Ошибка',
        `${err}`,
      )
    })
```

```
  }
authorizationtionHandle = () => {
const { login,password } = this.state
axios.post('http://192.168.43.22:3000/autentification', {
login: login,
password: password,
    }).then(response =>
 {
if (response.data.userExists)
{
this.props.navigation.replace('Account')
} else {
        Alert.alert(
          'Ошибка',
          `Такой пользователь не зарегистрирован`,
        )
      }
    }).catch(err => {
      Alert.alert(
        'Ошибка',
`${err}`,
      )
    })
  }
render() {
const { login,password } = this.state
return(
<ScrollView>
<TextInput
style={styles.input}
underlineColorAndroid = "transparent"
autoCapitalize = "none"
onChangeText={(login) => {this.setState({login})}}
placeholder="Логин"
placeholderTextColor = "#9a73ef"
value={login}
        />
<TextInput

style={styles.input}
underlineColorAndroid = "transparent"
autoCapitalize = "none"
onChangeText={(password) => {this.setState({password})}}
placeholder="Пароль"
```

```
placeholderTextColor = "#9a73ef"
value={password}
      />
<TouchableOpacity
style = {styles.submitButton}
onPress={this.authorizationtionHandle}
>
<Text style = {styles.submitButtonText}>Войти</Text>
</TouchableOpacity>
</ScrollView>
   )
  }
}
const styles = StyleSheet.create({
input: {
margin: 15,
marginTop: 30,
padding: 10,
height: 40,
borderColor: '#7a42f4',
borderWidth: 1
  },
submitButton: {
backgroundColor: '#7a42f4',
padding: 10,
margin: 15,
height: 40,
  },
submitButtonText:{
color: 'white',
  },
})
export { LoginScreen }
import React from 'react';
import { StyleSheet, View, Text, Button, TouchableOpacity, Alert } from 'react-
native';
import { NavigationEvents } from 'react-navigation'
import axios from 'axios';
class MainScreen extends React.Component {
static navigationOptions = {
title: 'Главная',
headerStyle: {
backgroundColor: '#7a42f4',
   },
```

```
headerTintColor: '#fff',
headerTitleStyle: {
fontWeight: 'bold',
    },
  }
state = {
userLogin: '',
userExists: false,
}
  checkSession = () =>{ // проверка сессии пользователя
axios.get('http://192.168.43.22:3000/checkSession').then(response => {
console.log('Данные с сервера')
console.log(response.data.userExists)
if (response.data.userExists)
{
this.setState((prevState) => {
return { userExists: response.data.userExists, userLogin: response.data.userLogin }
        })
      }
    }).catch(err => {
Alert.alert(
        'Ошибка',
        `${err}`,
      )
    })
  }
checkUserStatus = () => {
return this.state.userExists
  }
loginHandle = () => {
console.log(this.checkUserStatus())
if (!this.checkUserStatus()) {
return this.props.navigation.navigate('Login')
    } else {
return this.props.navigation.navigate('Account')
    }
  }
render() {
let regBtn = null

if (!this.state.userExists)
{
regBtn = <TouchableOpacity style={styles.mainButtons}><Text
style={styles.mainButtonText} onPress={() =>
```

```
this.props.navigation.navigate('Registration')}>Регистрация</Text></TouchableOp
acity>
    }
return(
<View style={styles.mainContainer}>
<NavigationEvents
onDidFocus={this.checkSession}
       />
<View style={styles.buttonsContainer}>
<TouchableOpacity style={styles.mainButtons}>
<Text style={styles.mainButtonText}
onPress={this.loginHandle}>Аккаунт</Text>
</TouchableOpacity>
         {regBtn}
<TouchableOpacity style={styles.mainButtons}>
<Text style={styles.mainButtonText} onPress={() =>
this.props.navigation.navigate('ResourcesList')}>Генерация кода</Text>
</TouchableOpacity>
</View>
</View>
    )
  }
}
const styles = StyleSheet.create({
mainContainer: {
flex: 1,
justifyContent: 'center',
alignItems: 'center',
padding: 15,
height: '100%'
  },
buttonsContainer: {
flexDirection: 'row',
justifyContent: 'center',
flexWrap: 'wrap',
alignItems: 'center',
height: '30%',
width: '90%',
paddingLeft: 20,
paddingRight: 20,
  },
mainButtons: {
backgroundColor: '#7a42f4',
padding: 10,
```

```
margin: 10,
height: 40,
width: 'auto',
  },
mainButtonText:{
color: 'white',
  },
border: {
borderWidth: 1,
borderColor: 'red',
}
})
export { MainScreen }
import React from 'react';
import { StyleSheet, View, ScrollView, Text, Button, TextInput, Alert,
TouchableOpacity} from 'react-native';
import axios from 'axios';
class RegistrationScreen extends React.Component {
static navigationOptions = {
title: 'Регистрация',
headerStyle: {
backgroundColor: '#7a42f4',
    },
headerTintColor: '#fff',
headerTitleStyle: {
fontWeight: 'bold',
    },
  }
state = {
login: '',
password: '',
address: '',
  }
registrationHandle = () => {
const { login,password,address } = this.state

if (login && password && address)
{
axios.post('http://192.168.43.22:3000/registration', {
login: login,
password: password,
address: address,
      }).then(response => {
Alert.alert(
```

```
          'Успех',
          `Я все могу`,
        )
      }).catch(err => {
Alert.alert(
          'Проверка',
          `${err}`,
        )
      })
    } else {
Alert.alert(
'Ошибка',
        `Для регистрации необходимо заполнить все поля`,
      )
    }
    Alert.alert(
      'Проверка',
      `${this.state.login} ${this.state.password} ${this.state.address}`,
)
  }
render() {
const { login,password,address } = this.state
return(
<ScrollView>
<TextInput
style={styles.input}
underlineColorAndroid = "transparent"
autoCapitalize = "none"
onChangeText={(login) => {this.setState({login})}}
placeholder="Логин"
placeholderTextColor = "#9a73ef"
value={login}
        />
<TextInput
style={styles.input}
underlineColorAndroid = "transparent"
autoCapitalize = "none"
onChangeText={(password) => {this.setState({password})}}
placeholder="Пароль"
placeholderTextColor = "#9a73ef"
value={password}
        />
<TextInput
style={styles.input}
```

```
underlineColorAndroid = "transparent"
autoCapitalize = "none"
onChangeText={(address) => {this.setState({address})}}
placeholder="DNS адресРесурса"
placeholderTextColor = "#9a73ef"
value={address}
      />
<TouchableOpacity
style = {styles.submitButton}
onPress={this.registrationHandle}
>
<Text style = {styles.submitButtonText}>Зарегистрировать аккаунт</Text>
</TouchableOpacity>
</ScrollView>    )
  }}
const styles = StyleSheet.create({
input: {
margin: 15,
marginTop: 30,
padding: 10,
height: 40,
borderColor: '#7a42f4',
borderWidth: 1
  },
submitButton: {
backgroundColor: '#7a42f4',
padding: 10,
margin: 15,
height: 40,
  },
submitButtonText:{
color: 'white',
  },
})
export { RegistrationScreen }
import React from 'react';
import { StyleSheet, View, ScrollView, Text, Button, TextInput, Alert,
TouchableOpacity} from 'react-native';
import { NavigationEvents } from 'react-navigation'
import axios from 'axios';
class ResourcesListScreen extends React.Component {
static navigationOptions = {
title: 'Выборресурса',
headerStyle: {
```

```
backgroundColor: '#7a42f4',
    },
headerTintColor: '#fff',
headerTitleStyle: {
fontWeight: 'bold',
    },
  }
state = {
resourcesList: '',
address: ''
  }
getResourcesList = () => {
axios.get('http://192.168.43.22:3000/getResources').then(response => {
let resources = []
response.data.resourcesList.forEach(item => {
console.log(item.name)
resources.push({id: item._id, title: item.name, address: item.address})
      })
this.setState((prevState) => {
return {resourcesList: resources}
      })
    }).catch(err => {
Alert.alert(
        'Ошибка',
        `${err}`,      )
    })
  }
generateCodeNavigate = (address) => {
this.props.navigation.navigate('GenerateCode', { address: address})
  }
render() {
let list = null

if (this.state.resourcesList) {
list = this.state.resourcesList.map(item => {
return<Button title={item.title} key={item.id} onPress={() =>
this.generateCodeNavigate(item.address)}/>
      })
    }
return(
<View>
<NavigationEvents
onDidFocus={this.getResourcesList}
/>
```

```
        {list}
</View>
    )
  }}
export { ResourcesListScreen }
import React, { Component } from 'react'; // подключение react
import $ from 'jquery'; // подключение jquery
class App extends Component {
state = { // глобальные переменные для компонентов (состояние)
login: '',
password: '',
profileName: '',
authorized: '',
userExists: '',
code: '',
}  componentDidMount = () =>{ // проверка сессии пользователя
$.ajax({
url: '/checkSession',
type: 'GET',
success: (res) => {
const { authorized,profileName } = res;
if (authorized) {
this.setState({ authorized: authorized, profileName: profileName });
        } else {
this.setState({ authorized: authorized });
}}
    })
  }
  onInputChange = (e) =>{ // при изменении полей ввода изменять общий state
const { id,value } = e.currentTarget;
this.setState({ [id]: value });
  }
onAuthorizationBtnClick = (e) => { // При нажатии на кнопку авторизации
отправляю логин и пароль на сервер
e.preventDefault();
const { login,password,code } = this.state;
if (login && password) {
if (!code) { // проверка ввода кода
        $.ajax({
url: '/authorization',
type: 'POST',
data: {
login: login,
password: password,
```

```
        },
success: (res) => {
const { userExists } = res;

if ( userExists )
{
this.setState({ userExists: userExists });
          }
else
{
this.setState({ authorized: false });
alert('Такая учетная запись не зарегестрирована')
}
        }
       });
     } else { // если код введен, отправляется запрос на сервер для проверки
кода
$.ajax({
url: '/checkCodeWithData',
type: 'POST',
data: {
login: login,
password: password,
code: code,
        },
success: (res) => {
const { authorized,profileName } = res;
if (authorized)
{
this.setState({ authorized: authorized, profileName: profileName, login: '',
password: '' })
          }
else
{
alert('Неверныйкод');
          }        }
       })
     }    }
Else
 {
alert('Нужно ввести данные');
   }
  }
onRegistrationBtnClick = (e) => { // регистрация пользователя
```

```
e.preventDefault();
const { login,password } = this.state;
if (login && password)
{
     $.ajax({
url: '/registration',
type: 'POST',
data: {
login: login,
password: password,
       },
success: (res) => {

if (res.registrationSuccess) {
this.setState({ login: '', password: '' })
alert('Регистрация прошла успешно');
         } else {
alert('Что-то пошло не так');
         }       }
     });
   } else {
alert('Нужно ввести данные');
   }  }
  onLogoutBtnClick = (e) =>{ // Выход из аккаунта
   e.preventDefault();
$.ajax({
url: '/logout',
type: 'GET',
success: (res) => {
const { authorized,profileName } = res;
this.setState({ authorized: authorized, profileName: profileName, code: '',
userExists: '' });
     }    });
  }
render() { // отрисовка компонентов
const { login,password,profileName,authorized,userExists,code } = this.state;
if (!authorized)
{
return(
<div className="main-block">
<form className="form-block">
<input onChange={this.onInputChange} id="login" placeholder="login"
value={login}/>
```

```
<input onChange={this.onInputChange} id="password" type="password"
placeholder="password" value={password}/>
          {
userExists&&<input onChange={this.onInputChange} id="code"
placeholder="Введикод" value={code}/>
}
<div className="buttons-block">
<button onClick={this.onAuthorizationBtnClick}>Вход</button>
<button onClick={this.onRegistrationBtnClick}>Регистрация</button>
</div>
</form>
</div>
     )
   } else {
return (
<div className="main-block">
<div className="account-block">
<p>Добропожаловать, {profileName}</p>
<button onClick={this.onLogoutBtnClick}>Выход</button>
</div>
</div>
     )
   }
  }
}
export default App;
const express = require('express');
const session = require('express-session');
const MongoStore = require('connect-mongo')(session);
const bodyParser = require('body-parser');
const request = require('request-promise');
const mongo = require('mongodb');
const MongoClient = require('mongodb').MongoClient;
const mongoClient = new
MongoClient("mongodb+srv://admin:Mdb12812122424@@cluster0-
o83lo.mongodb.net/test?retryWrites=true", { useNewUrlParser: true });
const app = express();
let dbClient;
app.set('view engine', 'ejs');
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true })); /* подключение модулей */
app.use(session({ // настройка сессий
secret: 'newSecret',
resave: false,
```

```
saveUninitialized: false,
}));
app.use((req, res, next) => { // вывод информации о запросах
console.log(`${req.method} ${req.url}`);
next();
});
app.get('/src/dist/bundle.js', (req, res) => { // отправка скомпилированного файла
res.sendFile(__dirname + '/dist/bundle.js');
});
app.get('/src/dist/bundle.js.map', (req, res) => { //
отправкаскомпилированногофайла
res.sendFile(__dirname + '/dist/bundle.js.map');
});
mongoClient.connect((err, client) => { // Подключение БД и запуск сервера
if (err) return console.log(err);
dbClient = client;
  app.locals.usersCollection = client.db('testDip').collection('users'); // база testDip,
"таблица" users
app.listen(process.env.PORT || 4000, () => {
console.log('Серверзапущен');
  }); });
app.post('/checkUser', (req, res) => { // Проверка на наличие аккаунта в базе для
отправки кода
const { login } = req.body;
let user = {
login: login,
  }
console.log(user);
const collection = app.locals.usersCollection;
collection.findOne(user, (err, result) => {
if (err) return console.log(err);
if (result)
{
res.send({userAvailable: true});
    } else
 {
res.send({userAvailable: false});
    }  })
})
app.post('/checkCodeWithData', (req, res) => { // проверка введенного кода
const { login,password,code } = req.body;
const collection = app.locals.usersCollection;
let user = {
login: login,
```

```
password: password,
  }
collection.findOne(user, (err, result) => {

if (err) return console.log(err);
    //console.log(result);

if (result)
{
const options = { // настройка запроса на сервер с генерацией пароля
method: 'POST',
uri: 'http://192.168.43.22:3000/checkCodeWithData',
body: {
login: login
          },
json: true
        }
request(options) // отправка запроса на сервер с генерацией пароля
.then((response) => {
console.log(response);
const { codeS } = response;
if (codeS == code)
{
             req.session.authorized = true;
             req.session.userlogin = login;
res.send({ authorized: true, profileName: req.session.userlogin })
           }
else
{
res.send({ authorized: false })
           }          })
         .catch((err) =>
{if (err) return console.log(err);
         })      }
    })})
app.post('/authorization', (req, res) => { // авторизация пользователя
  //console.log(req.body);
const { login,password } = req.body;
const collection = app.locals.usersCollection;
let user = {
login: login,
password: password
  }collection.findOne(user, (err, result) => { // поиск в БД
if (err) return console.log(err);
```

```
console.log(result);
if (result) {
res.send({ userExists: true });
      } else {
res.send({ userExists: false });
      }
    })
});
app.post('/registration', (req, res) => { // регистрация пользователя
console.log(req.body);
const { login,password } = req.body;
const user = { login: login, password: password };

const collection = app.locals.usersCollection;
collection.insertOne(user, (err, result) => { // добавление в БД

if (err) return console.log(err);
console.log('Пользователь добавлен');
res.send({ registrationSuccess: true });
  });
});
app.get('/checkSession', (req, res) => { // проверка сессии
if (req.session.authorized) {
res.send({ authorized: true, profileName: req.session.userlogin });
  }
else {
res.send({ authorized: false });
}
console.log(req.session);
});
app.get('/logout', (req, res) => { // выход из аккаунта
delete req.session.authorized;
delete req.session.userlogin;
res.send({ authorized: false, profileName: '' });
})
app.get('/', (req, res) => { // html файл
console.log(req.session);
res.render('index.ejs');
});
process.on("SIGINT", () => {
dbClient.close();
process.exit();
});
```