УДК 004.9

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДВУХФАКТОРНОЙ АУТЕНТИФИКАЦИИ ДЛЯ ЗАЩИТЫ ДАННЫХ В ГОСУДАРСТВЕННЫХ ОРГАНАХ КР

Калбердиев А.А., Орозобекова А.К.
КГТУ им. И. Раззакова

В условиях активного развития цифровых технологий перед государственными структурами встает необходимость надежной защиты учетных данных граждан. В настоящее время в государственных органах Кыргызской Республики отсутствуют локальные решения для двухфакторной аутентификации, что вынуждает использовать зарубежные сервисы. Однако такие решения не всегда соответствуют требованиям безопасности и могут привести к утечке конфиденциальной информации, а также усложняют централизованное управление доступом к государственным системам.

В статье проводится анализ существующих методов аутентификации, включая использование одноразовых паролей (OTP), аппаратных токенов, биометрической аутентификации, а также рассматриваются их преимущества и ограничения. В качестве альтернативного решения предлагается разработка локального аутентификатора, который обеспечит надежную защиту учетных данных, удобство в использовании и возможность интеграции с государственными информационными системами.

Описаны архитектурные принципы и технологический стек предлагаемого решения, включая использование микросервисного подхода, языка программирования Go, базы данных PostgreSQL, а также механизмы взаимодействия через REST API и WebSocket. Также рассматриваются алгоритмы генерации одноразовых паролей, управление учетными записями пользователей, а также меры защиты системы от возможных атак.

Представленный прототип демонстрирует возможность организации безопасной аутентификации без зависимости от зарубежных сервисов, что делает его перспективным для внедрения в государственных учреждениях.

Ключевые слова: защита, двухфакторной аутентификации, государственные органы, учетные записи, Backend, язык программирования Go.

КЫРГЫЗСТАНДА МАМЛЕКЕТТИК ОРГАНДАРЫНДА МААЛЫМАТТАРДЫ КОРГОО ҮЧҮН ЭКИ ФАКТОРЛУУ АУТЕНТИКАЦИЯЛЫК ТИРКЕМЕЛЕРДИ ИШТЕП ЧЫГУУ

Калбердиев А.А., Орозобекова А.К.
И. Раззаков атындагы КМТУ

Санариптик технологиялардын тездик менен өнүгүшү мамлекеттик органдар үчүн жарандардын аккаунттарын коргоонун ишенимдүү механизмдерин түзүү зарылдыгын жаратты. Учурда Кыргыз Республикасынын мамлекеттик мекемелеринде эки факторлуу аутентификация үчүн локалдык чечимдер жок болгондуктан, алар чет элдик кызматтарга таянууга аргасыз. Бирок мындай системалар мамлекеттик коопсуздуктун талаптарына дайым эле жооп бере бербейт жана маалыматтын сыртка чыгуу коркунучун жаратып, колдонуучулардын каттоо жазууларынын борборлоштурулган башкаруусун татаалдаштырат.

Бул макалада учурдагы аутентификация ыкмалары талданат, анын ичинде бир жолку сырсөздөр (OTP), аппараттык белгилер, биометрикалык аутентификация, жана алардын артыкчылыктары менен чектөөлөрү каралат. Алардын альтернативасы катары мамлекеттик мекемелердин маалыматтык тутумдары менен интеграцияланып, каттоо жазууларын коргоону камсыздай турган жергиликтүү аутентификатор иштеп чыгуу сунушталат.

Сунушталган чечимдин архитектуралык принциптери жана технологиялык негизи баяндалат, атап айтканда, микросервис архитектурасын, Go программалоо тилин, PostgreSQL маалымат базасын колдонуу, ошондой эле REST API жана WebSocket аркылуу өз ара аракеттенүү механизмдери каралат. Андан тышкары, бир жолку сырсөздөрдү жаратуу алгоритмдери, колдонуучулардын каттоо жазууларын башкаруу, ошондой эле тутумду мүмкүн болгон чабуулдардан коргоо чаралары талданат.

Көрсөтүлгөн прототип чет элдик аутентификация кызматтарына көз карандылыксыз коопсуз аутентификацияны ишке ашыруу мүмкүнчүлүгүн тастыктайт жана аны мамлекеттик мекемелерде колдонуу келечектүү багыт экенин көрсөтөт.

Баштапкы сөздөр: коргоо, эки фактордук аутентификация, мамлекеттик органдар, аккаунттар, Backend, Go программалоо тили.

DEVELOPMENT OF A TWO-FACTOR AUTHENTICATION APPLICATION FOR DATA PROTECTION IN GOVERNMENT AGENCIES OF THE KR

Kalberdiev A.A., Orozobekova A.K.
KSTU named after I. Razzakov

With the rapid advancement of digital technologies, government institutions face an urgent need to implement reliable mechanisms for protecting user credentials. Currently, government agencies in the Kyrgyz Republic lack local solutions for two-factor authentication, forcing them to rely on foreign services. However, such solutions do not always meet national security requirements, pose a risk of data leakage, and complicate centralized management of user accounts.

This paper analyzes existing authentication methods, including one-time passwords (OTP), hardware tokens, and biometric authentication, highlighting their advantages and limitations. As an alternative, the paper proposes the development of a local authenticator that ensures secure authentication, seamless integration with government information systems, and centralized user management.

The paper describes the architectural principles and technological stack of the proposed solution, including the use of a microservices architecture, the Go programming language, PostgreSQL for data storage, and interaction mechanisms through REST API and WebSocket. Additionally, it explores algorithms for generating one-time passwords, user account management, and security measures to protect against potential attacks.

The presented prototype demonstrates the feasibility of implementing secure authentication without reliance on foreign authentication services, making it a promising solution for adoption in government institutions.

Keywords: protection, two-factor authentication, government agencies, accounts, Backend, Go programming language.

В наши дни, когда цифровые технологии развиваются семимильными шагами, государственным учреждениям приходится работать с огромным объёмом конфиденциальной информации – от личных данных граждан до финансовых отчётов. Надёжная защита таких данных – первостепенная задача. В Кыргызстане пока отсутствуют собственные решения для двухфакторной аутентификации.

В итоге госструктуры вынуждены использовать зарубежные сервисы, такие как Google Authenticator и Microsoft Authenticator, что несёт в себе риски утечки данных, зависимости от иностранных компаний и сложности с централизованным управлением.

Проблемы и недостатки при использовании:

- Google Authenticator- этот сервис прост в использовании и не требует постоянного подключения к интернету. Однако каждый пользователь

настраивает его самостоятельно, что усложняет контроль со стороны администрации. Кроме того, данные могут передаваться за границу, что вызывает вопросы безопасности.

- Microsoft Authenticator- здесь добавлен удобный функционал push-уведомления, которые ускоряют процесс подтверждения входа. Но и тут есть недостатки: сервис зависит от экосистемы Microsoft, а данные также хранятся за рубежом, а это не всегда подходит для государственных систем.
- OTP-методы (TOTP и HOTP)
- Одноразовые пароли – надёжное решение, ведь каждый код действует лишь короткое время или ограниченное число раз. Но для их правильной работы требуется синхронизация серверного времени, а в масштабных системах это может стать проблемой.

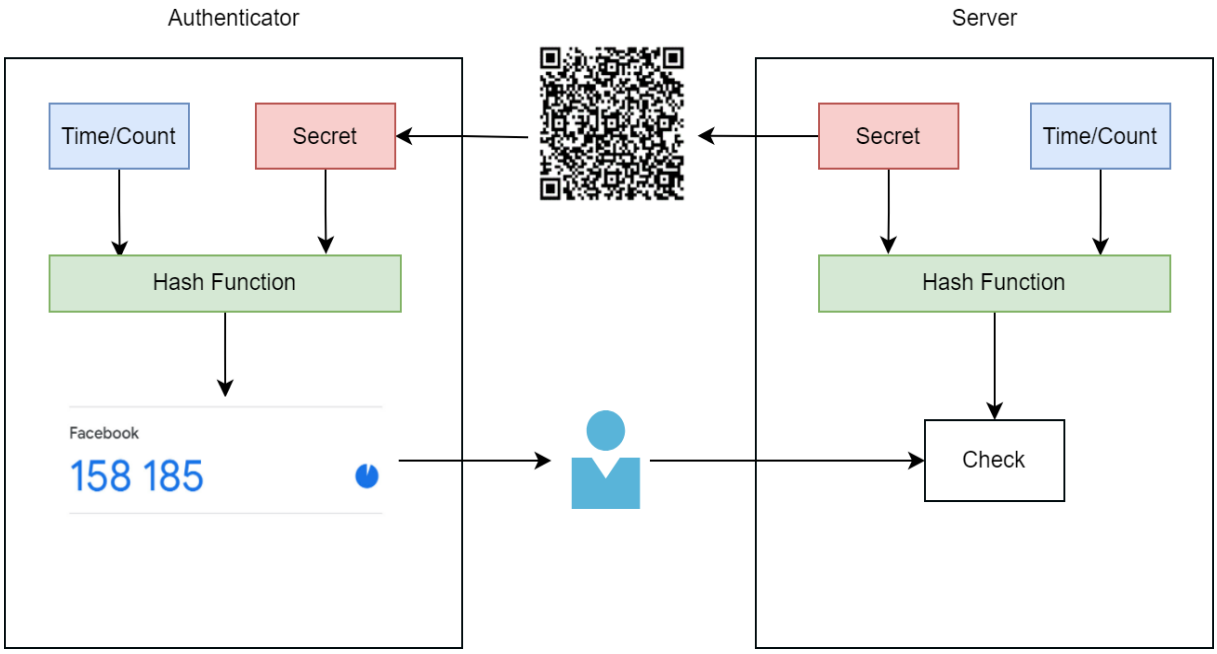

Рис 1. Структура системы

Нами разработано собственное решение, которое будет адаптировано под нужды государственных структур. Вот основные преимущества такого подхода:

- Безопасность: Все данные остаются на локальных серверах, что исключает их передачу за границу.
- Централизованный контроль: Администраторы легко управляют пользователями и настраивают политику безопасности.
- Удобная интеграция: Новый аутентификатор будет легко внедряться в уже существующие государственные сервисы.
- Простота использования: Интерфейс будет интуитивно понятным как для обычных пользователей, так и для специалистов.

На текущем этапе создан прототип, который успешно демонстрирует возможность защиты учетных данных без передачи их за границу. Кроме того, система показала высокую гибкость при интеграции с госуслугами.

В будущем это решение может стать основой для единой системы аутентификации в государственных органах Кыргызстана, значительно повышая безопасность и снижая зависимость от зарубежных сервисов.

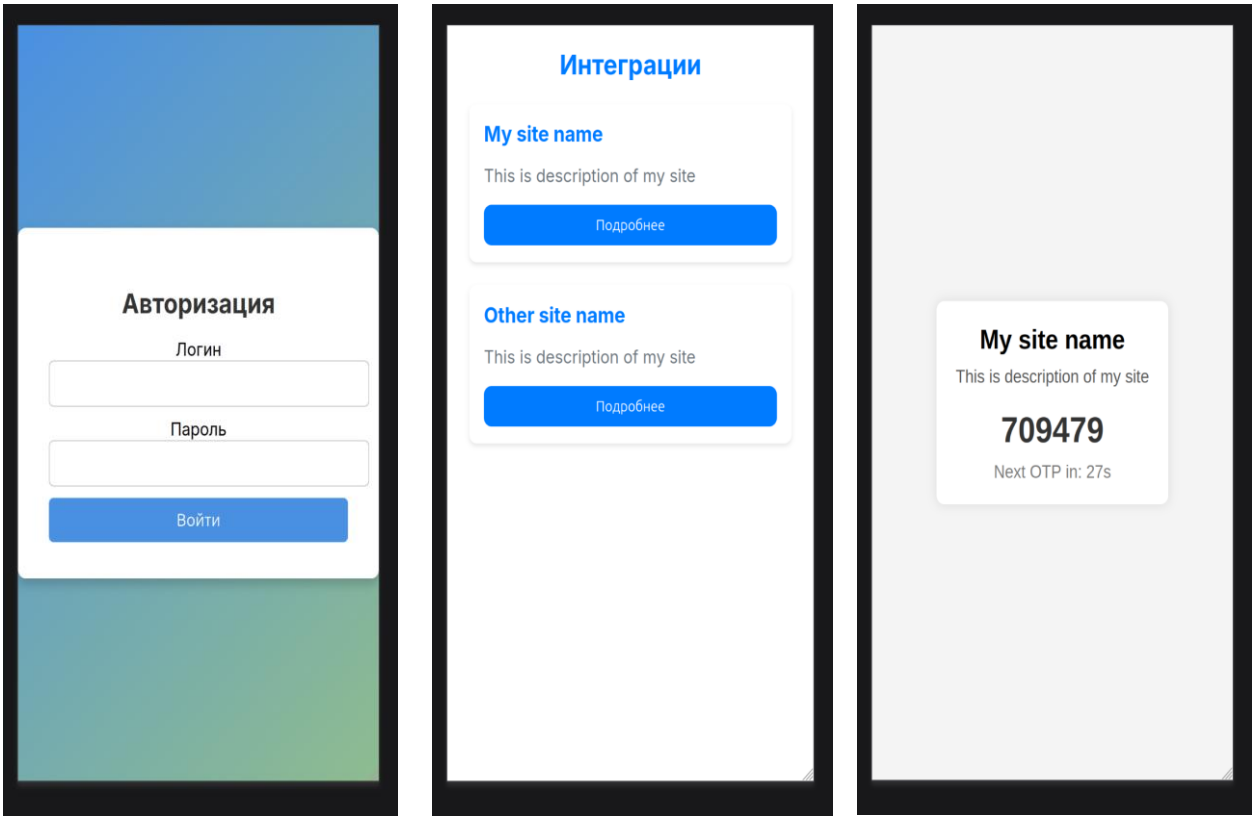

Рис 2. Авторизация

В условиях растущей угрозы утечек данных и компрометации учетных записей использование одного лишь пароля становится недостаточной мерой безопасности. В этом контексте двухфакторная аутентификация (2FA) играет важную роль, обеспечивая дополнительный уровень защиты. Однако стандартные методы двухфакторной аутентификации, такие как OTP-коды и push-уведомления, также могут быть уязвимы при целенаправленных атаках. Например, злоумышленник может перехватить SMS-код или использовать методы социальной инженерии, чтобы обманом получить OTP от пользователя.

В рамках разработки локального аутентификатора особое внимание уделяется устойчивости к потенциальным атакам и удобству интеграции с государственными сервисами. Для повышения уровня защиты предлагается несколько ключевых механизмов:

– Аппаратная привязка устройств – учетные записи пользователей можно привязывать не только к их логину и паролю, но и к

доверенным устройствам, предотвращая доступ с незарегистрированных терминалов.

- Контроль аномальной активности – анализ частоты входов, геолокации и используемых IP-адресов позволяет выявлять подозрительные попытки авторизации.
- Динамическое изменение уровня защиты – в зависимости от риска система может запрашивать дополнительный фактор аутентификации. Например, если пользователь пытается войти в систему с нового устройства или из другой страны, система автоматически усилит требования безопасности.

Для защиты учетных данных и удобного управления аутентификацией в разработанном решении применяется микросервисная архитектура, обеспечивающая гибкость и масштабируемость системы.

1. **Инфраструктура и серверная часть**

Серверная часть аутентификатора построена на языке программирования Go, что позволяет обрабатывать большое количество запросов с высокой производительностью. В качестве базы данных используется PostgreSQL, обеспечивающая надежное хранение учетных записей и журналов событий аутентификации.

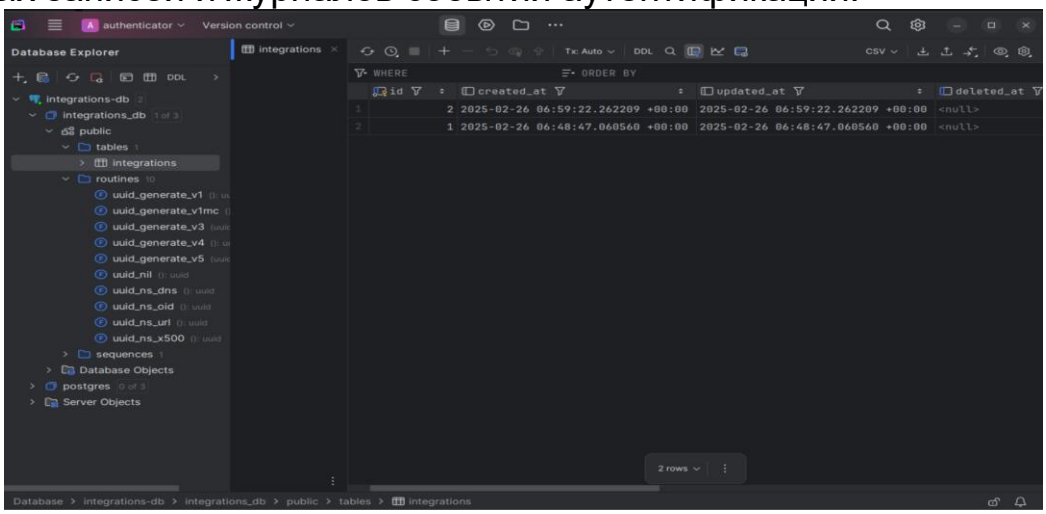

Рис 2. Таблица integrations

Взаимодействие между компонентами системы реализовано через **REST API и WebSocket**, что позволяет организовать гибкую интеграцию с различными государственными информационными системами.

Фрагмент кода одного из REST контроллеров)

```
package controllers
import (
"net/http"
"integrations-service/app/commands"
"integrations-service/app/queries"
"integrations-service/infrastructure/repositories"
```

```
"github.com/gin-gonic/gin"
"gorm.io/gorm"
"github.com/google/uuid"
)
type IntegrationController struct {
createHandler *commands.CreateIntegrationHandler
getAllHandler *queries.GetAllIntegrationsHandler
getByUUIDHandler *queries.GetIntegrationByUUIDHandler
}
func NewIntegrationController(db *gorm.DB) *IntegrationController
{
repo := repositories.NewIntegrationRepository(db)
createHandler := commands.NewCreateIntegrationHandler(repo)
getAllHandler := queries.NewGetAllIntegrationsHandler(repo)
getByUUIDHandler := queries.NewGetIntegrationByUUIDHandler(repo)

return &IntegrationController{
     createHandler:  createHandler,
     getAllHandler:  getAllHandler,
     getByUUIDHandler: getByUUIDHandler,
}
}
func (c *IntegrationController) CreateIntegration(ctx
*gin.Context) {
var cmd commands.CreateIntegrationCommand
if err := ctx.ShouldBindJSON(&cmd); err != nil {
     ctx.JSON(http.StatusBadRequest, gin.H{"error": "Неверный
формат JSON"})
     return
}
if err := c.createHandler.Handle(cmd); err != nil {
     ctx.JSON(http.StatusInternalServerError, gin.H{"error":
err.Error()})
     return
}

ctx.JSON(http.StatusCreated, gin.H{"message": "Интеграция
создана"})
}
func (c *IntegrationController) GetAllIntegrations(ctx
*gin.Context) {
integrations, err := c.getAllHandler.Handle()
if err != nil {
     ctx.JSON(http.StatusInternalServerError, gin.H{"error":
"Ошибка получения интеграций"})
     return
}
ctx.JSON(http.StatusOK, integrations)
}

func (c *IntegrationController) GetIntegrationByUUID(ctx
*gin.Context) {
```

```
idStr := ctx.Param("uuid")

id, err := uuid.Parse(idStr)
if err != nil {
     ctx.JSON(http.StatusBadRequest, gin.H{"error": "Неверный
формат UUID"})
     return
}
integration, err := c.getByUUIDHandler.Handle(id)
if err != nil {
     ctx.JSON(http.StatusNotFound, gin.H{"error": err.Error()})
     return
}
ctx.JSON(http.StatusOK, integration)
}
func SetupRoutes(r *gin.Engine, db *gorm.DB) {
controller := NewIntegrationController(db)
r.POST("/integrations", controller.CreateIntegration)
r.GET("/integrations", controller.GetAllIntegrations)
r.GET("/integrations/:uuid", controller.GetIntegrationByUUID)
}
```

2. Генерация и верификация OTP-кодов

Аутентификатор поддерживает алгоритмы **TOTP (основанный на времени) и HOTP (основанный на счетчике)**, которые обеспечивают безопасную генерацию одноразовых паролей. Секретные ключи пользователей хранятся в зашифрованном виде, а сам процесс верификации OTP проходит на сервере без передачи чувствительных данных в открытом виде.

3. Фронтенд и пользовательский интерфейс

Для удобства взаимодействия с пользователем разработан **веб-интерфейс на Angular 19**. В проекте используется **NX и Module Federation**, что позволяет легко разделять модули системы и встраивать их в различные государственные веб-порталы.

```
(Фрагмент кода одного из компонентов фронтенда приложения)

import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { AuthService } from '../services/auth.service';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';
import { HttpClientModule } from '@angular/common/http';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.scss'],
  providers: [HttpClientModule, AuthService],
  imports: [
CommonModule,
FormsModule,
```

```
HttpClientModule,
  ],
})
export class LoginComponent {
  username = '';
  password = '';
  errorMessage: string | null = null;

  constructor(private authService: AuthService, private router:
Router) {}

  onSubmit() {
this.authService.login(this.username, this.password).subscribe({
   next: (token) => {
     localStorage.setItem('auth_token', token);
     this.router.navigate(['/']);
   },
   error: () => {
     this.errorMessage = 'Ошибка авторизации. Проверьте логин и
пароль.';
   }
});
  }
}
```

4. Администрирование и мониторинг аутентификаций

Система предусматривает возможность **централизованного управления пользователями**. Администраторы госучреждений могут:

- Просматривать логи аутентификации.
- Настраивать политику безопасности (например, требовать 2FA только при входе с новых устройств).
- Управлять списком доверенных устройств пользователей.

Разработанное решение протестировано на предмет устойчивости к атакам, включая попытки подбора паролей, фишинг, MITM-атаки и попытки компрометации учетных данных. В ходе тестирования система продемонстрировала высокий уровень защиты и готовность к интеграции в государственные сервисы.

Ключевыми компонентами системы являются:

- **Сервис авторизации:** Управляет пользователями, генерирует и проверяет JWT-токены, а также позволяет настраивать политики безопасности.

Сервис интеграций: Обеспечивает подключение к внешним сервисам и управление API-ключами.

```
(Фрагмент кода по созданию новой интеграции)
package commands
import (
"errors"
"integrations-service/domain/aggregate"
"integrations-service/domain/repositories"
```

```
"github.com/google/uuid"
)
type CreateIntegrationCommand struct {
Name    string
Description string
}
type CreateIntegrationHandler struct {
repo repositories.IntegrationRepository
}
func NewCreateIntegrationHandler(repo
repositories.IntegrationRepository) *CreateIntegrationHandler {
return &CreateIntegrationHandler{repo}
}
func (receiver *CreateIntegrationHandler) Handle(cmd
CreateIntegrationCommand) error {
if cmd.Name == "" {
  return errors.New("название интеграции не может быть пустым")
}
integration, err := aggregate.NewIntegration(cmd.Name,
cmd.Description, uuid.New().String())
if err != nil {
  return err
}
return receiver.repo.Create(integration.Integration)
  }
```

- **Модуль OTP:** Генерирует и проверяет одноразовые пароли, защищая систему от попыток подбора кодов.

Выводы: Создание локального аутентификатора – это не просто технологическая инновация, а важный шаг к обеспечению безопасности личных данных граждан и независимости государственных структур.

Разработанное решение уже сегодня демонстрирует потенциал для внедрения в государственных учреждениях, а его дальнейшее развитие позволит ещё больше укрепить доверие к государственным цифровым сервисам.

ЛИТЕРАТУРА

1. Стандарты и исследования по двухфакторной аутентификации: RFC 6238. TOTP: Time-Based One-Time Password Algorithm. [Электронный ресурс]: https://datatracker.ietf.org/doc/html/rfc6238
2. RFC 4226. HOTP: An HMAC-Based One-Time Password Algorithm. [Электронный ресурс]: https://datatracker.ietf.org/doc/html/rfc4226, Национальный институт стандартов и технологий США (NIST).
3. Спецификация для двухфакторной аутентификации. [Электронный ресурс]: https://pages.nist.gov/800-63-3/
4. Исследование угроз фишинга и атак на двухфакторную аутентификацию. ACM Journal of Cybersecurity, 2022.

5. Документация Go: https://golang.org/doc/
6. PostgreSQL Documentation: https://www.postgresql.org/docs/
7. WebSocket API – MDN Web Docs: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API
8. OpenID Connect, спецификация аутентификации: https://openid.net/specs/openid-connect-core-1_0.html
9. Практические исследования и государственные проекты
10. Криптографическая безопасность двухфакторной аутентификации. Journal of Information Security, 2021.
11. Применение TOTP и HOTP в государственных системах: опыт Эстонии и Германии. European Journal of Digital Governance, 2023.
12. "Внедрение многофакторной аутентификации в правительственных сервисах" – доклад Министерства цифрового развития РФ, 2022.
13. Публикации по кибербезопасности и аутентификации
14. Schneier B. "Applied Cryptography", 2nd edition. John Wiley & Sons, 1996.
15. Ross J. "Implementing Two-Factor Authentication", O’Reilly Media, 2018.
16. Biryukov A., Khovratovich D., Perrin L. "Security Analysis of Authentication Protocols". Journal of Cryptology, 2021.