ДОДАТОК А

Слайди презентації

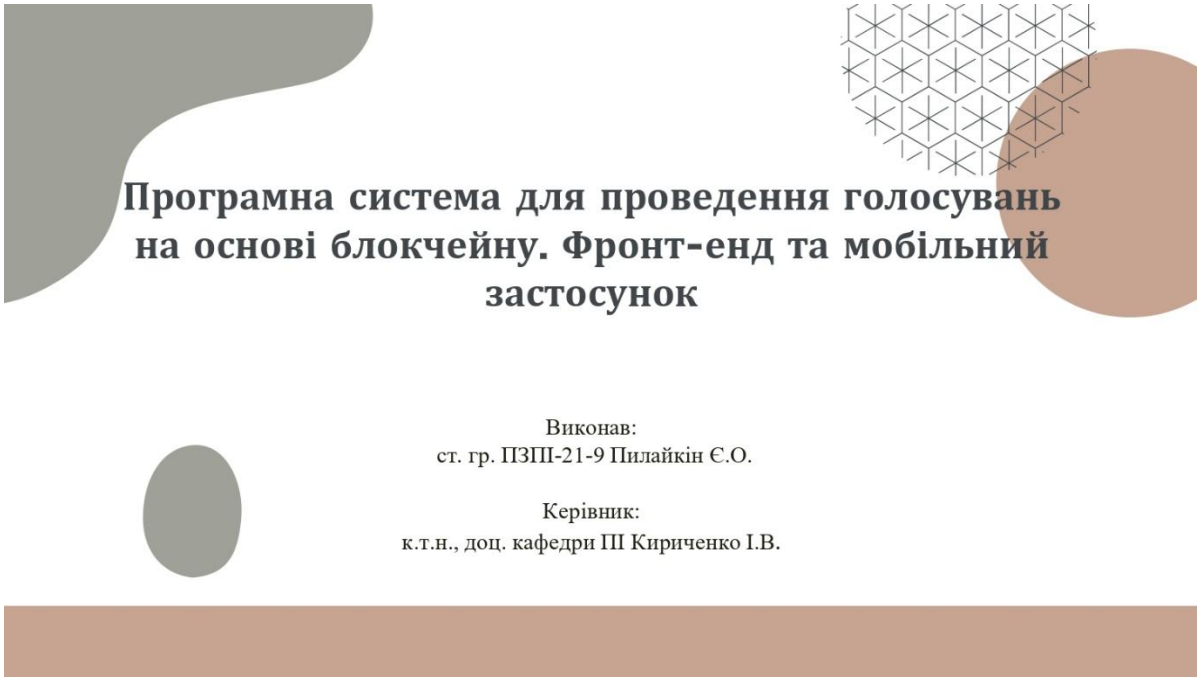


Рисунок А.1 – Слайд №1

Рисунок А.2 – Слайд №2

Аналіз конкурентів та проблематика

Фактори актуальності дослідження:

	Простота інтерфейсу	Блокчейн підтримка	Функціональність	Безпека	Адаптивність
Estonia e-voting	9	Ні	4	7	3
Democracy Earth	3	Так	8	9	5
Voatz	4	Так	7	8	4
Follow My Vote	5	Так	6	8	4
Наше рішення VeritasVote	**9**	**Так**	**9**	**9**	**9**

Виявлені проблеми:

•Надмірна технічна складність блокчейн-інтерфейсів

•Необхідність управління приватними ключами користувачами

•Відсутність прозорості операцій для звичайних користувачів

•Обмежена адаптивність для різних ролей та типів голосувань

Рисунок А.3 – Слайд №3

Постановка задачі та цілі

Функціональні вимоги:

1. Веб- додаток: створення, участь, перегляд результатів голосувань
2. Мобільний додаток: верифікація учасників довіреними особами
3. Безпека: JWT- токени, біометрична аутентифікація
4. UX: інтуїтивний інтерфейс без технічної складності

Нефункціональні вимоги:

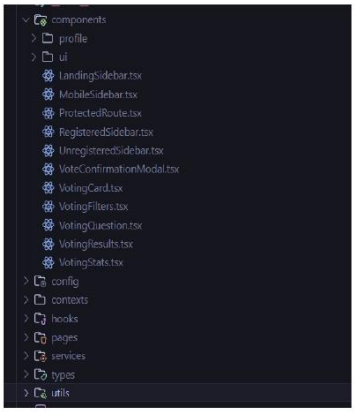
1. Масштабованість для великих голосувань
2. Кросплатформенність та адаптивність
3. Високі показники продуктивності
4. Відповідність стандартам безпеки

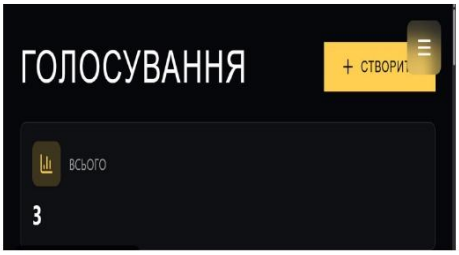
Рисунок А.4 – Слайд №4

Рисунок А.5 – Слайд №5

Рисунок А.6 – Слайд №6

Проєктування інтерфейсу





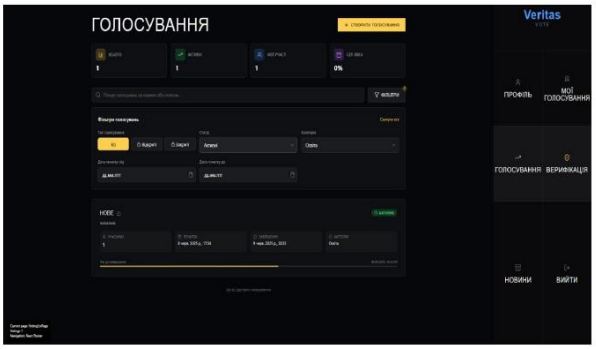


Рисунок А.7 – Слайд №7

Ключові алгоритми та методи

```
const validateStep = (step: number): boolean => {
  switch (step) {
    case 1:
      return !!formData.title && !!formData.description && !!formData.category &&
             !!formData.startDate && !!formData.startTime &&
             !!formData.endDate && !!formData.endTime;
    case 2:
      return formData.options.every(opt => opt.text);
    case 3:
      if (formData.isPrivate) {
        return formData.trustedPersons.length > 0 &&
               formData.trustedPersons.every(p => p.name && p.email);
      }
      return true;
    default:
      return true;
  }
};

const handleSubmit = async () => {
  setIsSubmitting(true);
  try {

    const createPollData = {
      title: formData.title,
      description: formData.description,
      category: formData.category as any,
      is_private: formData.isPrivate,
      start_time: new Date(`${formData.startDate}T${formData.startTime}`).toISOString(),
      end_time: new Date(`${formData.endDate}T${formData.endTime}`).toISOString(),
      options: formData.options.map((option, index) => ({
        text: option.text,
        position: index
      })),

      ...(formData.settings.minParticipants && {
        min_voter_turnout: formData.settings.minParticipants
      }),
      ...(formData.settings.maxParticipants && {
```

Алгоритм багатокрокового створення голосувань

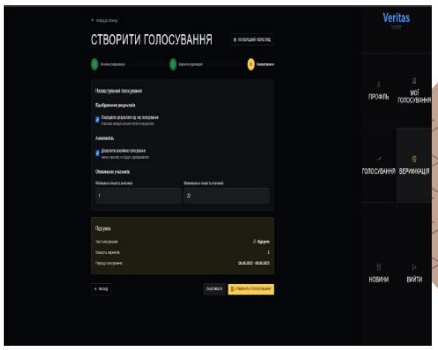
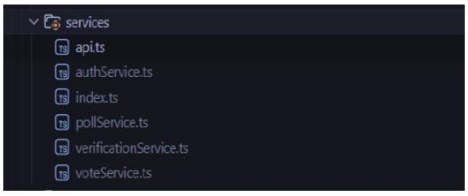


Рисунок А.8 – Слайд №8

Сервісний шар та API

Структура сервісів:



Приклад AuthService:

```
class AuthService {
  async register(email: string, password: string, phoneNumber: string, countryCode: string):
  Promise<User> {
    const response = await authApiClient.post<ApiResponse<User>>('/api/v1/auth/register', {
      email,
      password,
      phone_number: phoneNumber,
      country_code: countryCode,
    });
    return response.data;
  }

  async login(email: string, password: string): Promise<string> {
    const response = await authApiClient.post<ApiResponse<AuthResponse>>('/api/v1/auth/login', {
      email,
      password,
    });

    authApiClient.setToken(response.data.token);
    votingApiClient.setToken(response.data.token);
    return response.data.token;
  }

  async getCurrentUser(): Promise<User> {
    const response = await authApiClient.get<ApiResponse<User>>('/api/v1/users/me');
    return response.data;
  }

  async getUserById(userId: number): Promise<User> {
    const response = await authApiClient.get<ApiResponse<User>>(`/api/v1/users/${userId}`);
    return response.data;
  }

  async getUserRoles(userId: number): Promise<Role[]> {
    const response = await authApiClient.get<ApiResponse<Role[]>>(`/api/v1/user-roles/user/${userId}
    `);
    return response.data;
  }
```

Рисунок А.9 – Слайд №9

Управління станом та безпека

React Context API для глобального стану:

```
const AuthContext = createContext<AuthContextType | undefined>(undefined);

export const useAuth = () => {
  const context = useContext(AuthContext);
  if (!context) {
    throw new Error('useAuth повинно бути в контексті');
  }
  return context;
};

interface AuthProviderProps {
  children: ReactNode;
}

export const AuthProvider: React.FC<AuthProviderProps> = ({ children }) => {
  const [user, setUser] = useState<User | null>(null);
  const [isLoading, setIsLoading] = useState(true);
  const [error, setError] = useState<string | null>(null);
  const [lastRefresh, setLastRefresh] = useState<number>(0);

  const fetchUserWithRoles = async () => {
    try {
      const userData = await authService.getCurrentUser();
      setUser(userData);
    } catch (err) {
      throw err;
    }
  };

  useEffect(() => {
    const initAuth = async () => {
      try {
        if (authService.isAuthenticated()) {
          await fetchUserWithRoles();
        }
```

Рисунок А.10 – Слайд №10

Мобільний додаток (верифікація)

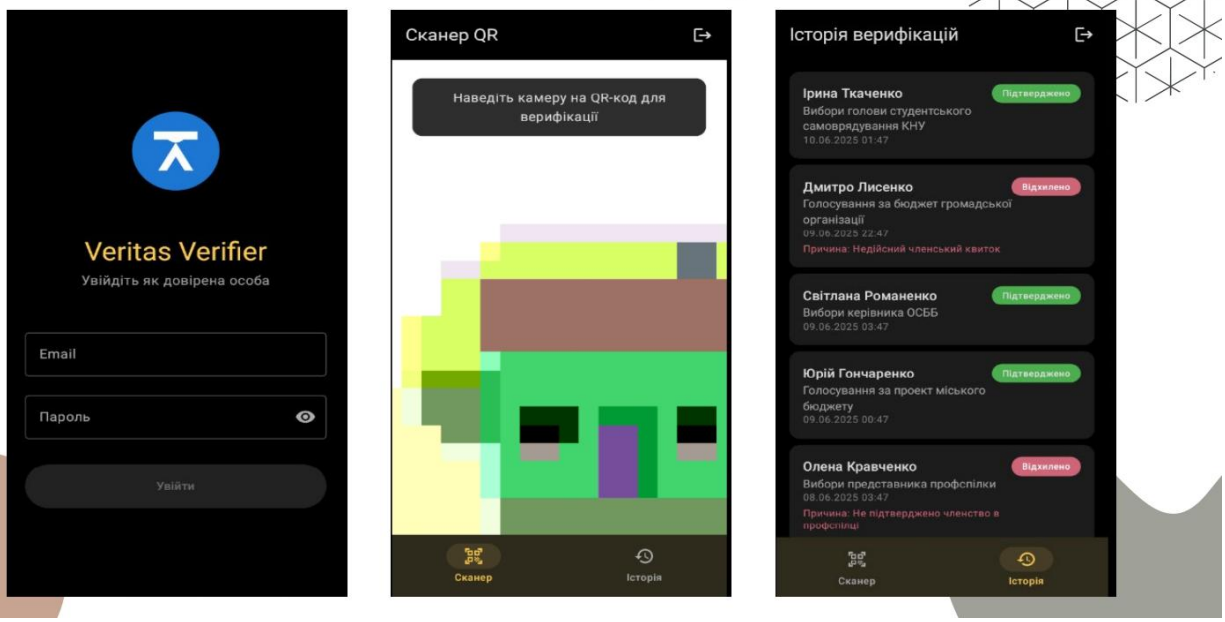
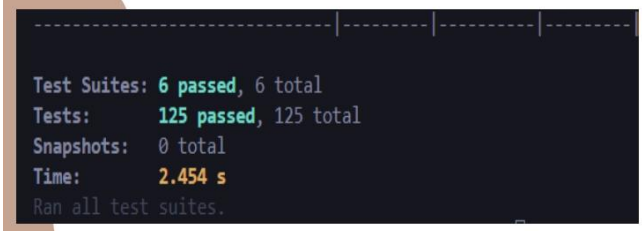


Рисунок А.11 – Слайд №11

Unit-тестування



Структура тестування:

- authService: 24 тести (аутентифікація)
- pollService: 33 тести (управління голосуваннями)
- voteService: 11 тестів (процес голосування)
- verificationService: 42 тести (верифікація)
- UI компоненти: 9 тестів
- Допоміжні функції: 5 тестів

Рисунок А.12 – Слайд №12

Метрики та якість коду

api.ts		0%	0/63	0%	0/36	0%	0/21	0%	0/61
authService.ts		100%	36/36	66.66%	14/21	100%	17/17	100%	34/34
index.ts		0%	0/0	0%	0/0	0%	0/0	0%	0/0
pollService.ts		100%	68/68	100%	23/23	100%	18/18	100%	62/62

Рисунок А.13 – Слайд №13

ДЯКУЮ ЗА УВАГУ!

Рисунок А.14 – Слайд №14

ДОДАТОК Б

Специфікація програмного продукту

1 ВСТУП

1.1 Огляд проекту

За часів цифрової трансформації демократичних процесів, зростання довіри до блокчейн-технологій і потреби в прозорих виборах, клієнтська частина системи електронного голосування стає незамінною для сучасного демократичного суспільства. VeritasVote забезпечує цінний рівень безпеки та прозорості для процесу голосування. Система може бути налаштована як веб-платформа або мобільний додаток, захищаючи користувачів від різного роду загроз у процесі голосування.

1.2 Мета

Метою є створення клієнтської частини системи електронного голосування, яка спрощуватиме взаємодію з блокчейн-технологіями та забезпечуватиме інтуїтивно зрозумілий доступ до процесу голосування.

1.3 Область застосування

Даний програмний продукт можна використовувати в навчальних закладах, корпораціях, громадських організаціях та державних установах для проведення різних типів голосувань. Електронне голосування є потребою багатьох організацій, тому область застосування програмного продукту є досить широкою.

1.4 Короткий зміст

Слід пам’ятати, що сьогодні користувачі дуже прискіпливо відносяться до таких аспектів системи, як:

- зручність у використанні;
- стабільність роботи;
- захищеність даних;

- швидкість роботи;
- прозорість процесу;
- анонімність голосування.

1.5 Означення та абревіатури

Блокчейн – це розподілена база даних, що складається з ланцюжка блоків, кожен з яких містить криптографічний хеш попереднього блоку, мітку часу та дані транзакцій. Забезпечує незмінність та прозорість записів.

API (Прикладний програмний інтерфейс) – це набір чітко визначених методів для взаємодії різних компонентів системи голосування. API надає розробнику засоби для швидкої розробки та інтеграції з серверною частиною.

JWT (JSON Web Token) – стандарт для безпечної передачі інформації між сторонами у вигляді JSON-об'єкта.

React Context – механізм React для управління глобальним станом додатку.

2 ЗАГАЛЬНИЙ ОПИС

2.1 Перспективи продукту

Розроблювана програмна система надасть користувачам цілком новий рівень автоматизації та взаємодії з системою голосування за такими важливими параметрами як безпека, прозорість та зручність використання. Будь-який користувач зможе з легкістю брати участь у голосуваннях шляхом використання веб-інтерфейсу або мобільного додатку.

2.2 Функції продукту

Функціями продукту є створення голосувань, участь у голосуванні, перегляд результатів, верифікація користувачів (для довірених осіб), управління профілем користувача та адміністрування системи.

2.3 Характеристики користувачів

Цільовим користувачем виступає будь-який користувач навчального закладу,

корпорації або громадської організації, який має базові навички роботи з веб-інтерфейсами та розуміє процес голосування.

2.4 Загальні обмеження

2.4.1 Операційне середовище

В якості клієнтської частини виступає веб-застосунок написаний за допомогою React, TypeScript, HTML та CSS, а також мобільний додаток на Kotlin з Jetpack Compose. Середа розробки клієнтської частини: MS Visual Studio Code, Android Studio. В якості засобу тестування виступає Jest з React Testing Library.

2.4.2 Технологія розробки

React, TypeScript, Kotlin, Jetpack Compose, REST API, JWT, React Context API, Tailwind CSS.

2.5 Припущення й залежності

Припущення:

– веб система електронного голосування буде мати попит у цільових користувачів навчальних закладів та організацій, які потребують прозорих та безпечних голосувань.

Залежності:

– початкова версія веб-додатку працюватиме у всіх сучасних браузерах (Chrome, Firefox, Safari, Edge);

– мобільний додаток працюватиме на Android версії 7.0 та вище;

– система потребує стабільного інтернет-з'єднання для синхронізації з сервером.

3 КОНКРЕТНІ ВИМОГИ

3.1. Вимоги до інтерфейсів

3.1.1. Інтерфейс користувача

Клієнтська частина повинна представляти собою адаптивний веб-інтерфейс,

доступний через браузер, та мобільний додаток з інтуїтивно зрозумілим інтерфейсом для верифікації користувачів.

3.1.2. Апаратний інтерфейс

Апаратний інтерфейс веб-додатку – це браузер та його Web API. Для мобільного додатку – це Android API та камера пристрою для QR-сканування. У фоновому режимі система використовує складний код для взаємодії з блокчейном, але ця складність абстрагується за допомогою серверного API.

3.1.3. Комунікаційний протокол

HTTP/HTTPS.

3.1.4. Вимоги до пам’яті

Веб-додаток: Вимогами до пам'яті є системні вимоги сучасних браузерів.

- Windows 10/11: мінімум 4GB RAM
- macOS 10.15+: мінімум 4GB RAM
- – Linux: мінімум 2GB RAM

Мобільний додаток:

- Android 7.0+ (API level 24)
- Мінімум 2GB RAM
- 100MB вільного простору
- Камера для QR-сканування

3.2 Атрибути програмного продукту

3.2.1 Безпека

Система забезпечує високий рівень безпеки через JWT-токени, біометричну аутентифікацію, шифрування даних при передачі, захист від XSS та CSRF атак. Приватні ключі зберігаються на сервері, а не на клієнті. Система запитує мінімум необхідних дозволів.

3.2.2 Супроводжуваність

Після випуску програмного продукту планується супроводжуваність через систему issue tracking, документацію API, регулярні оновлення безпеки та підтримку користувачів через систему тикетів.

3.2.3 Переносимість

Веб-додаток працює у всіх сучасних браузерах завдяки використанню стандартних веб-технологій. Мобільний додаток розроблено для Android з можливістю портування на iOS у майбутньому.

3.2.4 Продуктивність

- Час відгуку інтерфейсу: менше 200мс
- Час завантаження сторінки: менше 3 секунд
- Час виконання тестів: менше 5 секунд
- Підтримка одночасної роботи 1000+ користувачів

3.2.5 Вимоги до бази даних

Система взаємодіє з серверною частиною через REST API. Безпосередньої роботи з базою даних немає. Локальне зберігання використовує браузерний localStorage та React Context для управління станом.

ДОДАТОК В

Фрагменти коду програмної реалізації

Компонент VotingCard.tsx

```
import React from 'react';
import { Clock, Users, Calendar, Lock, Unlock, CheckCircle, AlertCircle
} from 'lucide-react';
import { Badge } from './ui/Badge';
import { Card } from './ui/Card';
import { getCategoryTranslation } from '../utils/categories';

interface VotingCardProps {
  id: string;
  title: string;
  description: string;
  requireVerification: boolean;
  status: 'active' | 'ended' | 'failed';
  startDate: string;
  endDate: string;
  participantsCount: number;
  totalEligible?: number;
  category: string;
  onClick?: () => void;
}

export const VotingCard: React.FC<VotingCardProps> = ({
  title,
  description,
  requireVerification,
  status,
  startDate,
  endDate,
  participantsCount,
  totalEligible,
  category,
  onClick
}) => {
  const getStatusBadge = () => {
    const statusConfig = {
      active: { variant: 'success' as const, label: 'АКТИВНЕ', icon:
Clock },
      ended: { variant: 'default' as const, label: 'ЗАВЕРШЕНО', icon:
CheckCircle },
      failed: { variant: 'error' as const, label: 'НЕ ВІДБУЛОСЬ', icon:
AlertCircle }
    };

    const config = statusConfig[status];
    const Icon = config.icon;

    return (
      <Badge variant={config.variant} className="flex items-center
gap-1.5">
        <Icon size={14} />
        {config.label}
```

```
            </Badge>
          );
        };

        const formatDate = (dateString: string) => {
          return new Date(dateString).toLocaleDateString('uk-UA', {
            day: 'numeric',
            month: 'short',
            year: 'numeric',
            hour: '2-digit',
            minute: '2-digit'
          });
        };

        const participationPercentage = totalEligible
          ? Math.round((participantsCount / totalEligible) * 100)
          : 0;

        return (
          <Card
            className="cursor-pointer  hover:bg-white/[0.05]  transition-all
duration-300 group"
            onClick={onClick}
            padding="lg"
          >
            <div className="space-y-4">

              <div className="flex items-start justify-between">
                <div className="flex-1">
                  <div className="flex items-center gap-3 mb-2">
                    <h3
                      className="text-white   text-xl   font-medium   group-
hover:text-[#FFCF52] transition-colors"
                      style={{ fontFamily: 'Ristretto Pro, sans-serif' }}
                    >
                      {title.toUpperCase()}
                    </h3>
                    {requireVerification ? (
                      <Lock size={16} className="text-white/40" />
                    ) : (
                      <Unlock size={16} className="text-white/40" />
                    )}
                  </div>
                  <p className="text-white/60 text-sm line-clamp-2">
                    {description}
                  </p>
                </div>
                <div className="flex items-center gap-2">
                  {getStatusBadge()}
                </div>
              </div>

              <div className="grid grid-cols-2 md:grid-cols-4 gap-4">

                <div className="bg-white/[0.03] rounded-lg p-3">
                  <div className="flex items-center gap-2 mb-1">
                    <Users size={14} className="text-white/40" />
```

```
                    <span          className="text-white/40          text-xs
uppercase">Учасники</span>
                  </div>
                  <div className="text-white text-lg font-medium">
                    {participantsCount.toLocaleString('uk-UA')}
                    {totalEligible && (
                      <span className="text-white/40 text-sm ml-1">
                        / {totalEligible.toLocaleString('uk-UA')}
                      </span>
                    )}
                  </div>
                  {totalEligible && (
                    <div className="mt-2">
                      <div className="w-full h-1 bg-white/[0.1] rounded-full
overflow-hidden">
                        <div
                          className="h-full   bg-[#FFCF52]   transition-all
duration-500"
                          style={{ width: `${participationPercentage}%` }}
                        />
                      </div>
                      <span     className="text-white/40     text-xs     mt-
1">{participationPercentage}%</span>
                    </div>
                  )}
                </div>

                <div className="bg-white/[0.03] rounded-lg p-3">
                  <div className="flex items-center gap-2 mb-1">
                    <Calendar size={14} className="text-white/40" />
                    <span          className="text-white/40          text-xs
uppercase">Початок</span>
                  </div>
                  <div className="text-white text-sm">
                    {formatDate(startDate)}
                  </div>
                </div>

                <div className="bg-white/[0.03] rounded-lg p-3">
                  <div className="flex items-center gap-2 mb-1">
                    <Clock size={14} className="text-white/40" />
                    <span          className="text-white/40          text-xs
uppercase">Завершення</span>
                  </div>
                  <div className="text-white text-sm">
                    {formatDate(endDate)}
                  </div>
                </div>

                <div className="bg-white/[0.03] rounded-lg p-3">
                  <div className="flex items-center gap-2 mb-1">
                    <AlertCircle size={14} className="text-white/40" />
                    <span          className="text-white/40          text-xs
uppercase">Категорія</span>
                  </div>
                  <div className="text-white text-sm">
                    {getCategoryTranslation(category)}
                  </div>
```

```
              </div>
            </div>

            {status === 'active' && (
              <div className="pt-2">
                <div className="flex items-center justify-between text-xs
text-white/40 mb-1">
                  <span>Час до завершення</span>
                  <span>{new Date(endDate).toLocaleString('uk-UA')}</span>
                </div>
                <div  className="w-full  h-1  bg-white/[0.1]  rounded-full
overflow-hidden">
                  <div
                    className="h-full bg-gradient-to-r from-[#FFCF52] to-
[#FFD870] animate-pulse"
                    style={{ width: '60%' }}
                  />
                </div>
              </div>
            )}
          </div>
        </Card>
      );
    };
```

Сервіс голосувань pollService.tsx

```
    import { votingApiClient, getErrorMessage } from './api';
    import {
      Poll,
      PollResponse,
      CreatePollRequest,
      UpdatePollRequest,
      PollListOptions,
      PollResultResponse,
      OptionResponse,
      PollFilter,
      ListResponse,
      SuccessResponse,
      ApiResponse,
      ApiSuccessResponse,
    } from '../types/models';

    class PollService {
      private readonly basePath = '/api/v1/polls';

      async createPoll(data: CreatePollRequest): Promise<PollResponse> {
        try {
          const response = await votingApiClient.post<PollResponse>(
            this.basePath,
            data
          );
          return response;
        } catch (error) {
          throw      new      Error(`Failed      to      create      poll:
${getErrorMessage(error)}`);
```

```
    }
  }

  async getPoll(pollId: number): Promise<PollResponse> {
    try {
      const response = await
votingApiClient.get<ApiSuccessResponse<PollResponse>>(
        `${this.basePath}/${pollId}`
      );
      return response.data;
    } catch (error) {
      throw new Error(`Помилка при загрузці голосування:
${getErrorMessage(error)}`);
    }
  }

  async getPolls(options?: PollListOptions):
Promise<ListResponse<PollResponse>> {
    try {
      const params = this.buildPollQueryParams(options);
      const response = await
votingApiClient.get<ApiSuccessResponse<ListResponse<PollResponse>>>(
        this.basePath,
        params
      );
      return response.data;
    } catch (error) {
      throw new Error(`Помилка при загрузці голосувань:
${getErrorMessage(error)}`);
    }
  }

  async getUserPolls(userId: number, options?: PollListOptions):
Promise<ListResponse<PollResponse>> {
    const userOptions: PollListOptions = {
      ...options,
      filter: {
        ...options?.filter,
        creator_id: userId,
      },
    };
    return this.getPolls(userOptions);
  }

  async getActivePolls(options?: PollListOptions):
Promise<ListResponse<PollResponse>> {
    const activeOptions: PollListOptions = {
      ...options,
      filter: {
        ...options?.filter,
        status: 'active',
      },
    };
    return this.getPolls(activeOptions);
  }

  async updatePoll(pollId: number, data: UpdatePollRequest):
Promise<PollResponse> {
```

```
        try {
          const response = await votingApiClient.put<ApiResponse<PollResponse>>(
            `${this.basePath}/${pollId}`,
            data
          );
          return response.data;
        } catch (error) {
          throw new Error(`Failed to update poll: ${getErrorMessage(error)}`);
        }
      }

      async deletePoll(pollId: number): Promise<void> {
        try {
          await votingApiClient.delete(`${this.basePath}/${pollId}`);
        } catch (error) {
          throw new Error(`Failed to delete poll: ${getErrorMessage(error)}`);
        }
      }

      async getPollResults(pollId: number): Promise<OptionResponse[]> {
        try {
          const response = await votingApiClient.get<ApiResponse<OptionResponse[]>>(
            `${this.basePath}/${pollId}/results`
          );
          return response.data;
        } catch (error) {
          throw new Error(`Failed to fetch poll results: ${getErrorMessage(error)}`);
        }
      }

      async getPollStats(pollId: number): Promise<PollResultResponse> {
        try {
          const response = await votingApiClient.get<ApiResponse<PollResultResponse>>(
            `${this.basePath}/${pollId}/stats`
          );
          return response.data;
        } catch (error) {
          throw new Error(`Failed to fetch poll stats: ${getErrorMessage(error)}`);
        }
      }

      async searchPolls(query: string, options?: PollListOptions): Promise<ListResponse<PollResponse>> {
        const searchOptions: PollListOptions = {
          ...options,
          filter: {
            ...options?.filter,
            search: query,
          },
        };
        return this.getPolls(searchOptions);
```

```
  }

  async getPollsByCategory(category: string, options?:
PollListOptions): Promise<ListResponse<PollResponse>> {
    const categoryOptions: PollListOptions = {
      ...options,
      filter: {
        ...options?.filter,
        category: category as any,
      },
    };
    return this.getPolls(categoryOptions);
  }

  private buildPollQueryParams(options?: PollListOptions):
Record<string, any> {
    const params: Record<string, any> = {};

    if (options?.page) {
      params.page = options.page;
    }
    if (options?.page_size) {
      params.page_size = options.page_size;
    }

    if (options?.filter) {
      const filter = options.filter;
      if (filter.creator_id !== undefined) params['creator_id'] =
filter.creator_id;
      if (filter.category) params['category'] = filter.category;
      if (filter.is_private !== undefined) {
        params['is_private'] = filter.is_private;
      }
      if (filter.status) params['status'] = filter.status;
      if (filter.search) params['search'] = filter.search;
      if (filter.from_date) params['from_date'] = filter.from_date;
      if (filter.to_date) params['to_date'] = filter.to_date;
    }
    return params;
  }

  filterByStatus(status: PollFilter['status']): PollFilter {
    return { status };
  }

  filterByCategory(category: PollFilter['category']): PollFilter {
    return { category };
  }

  filterByDateRange(from: string, to: string): PollFilter {
    return { from_date: from, to_date: to };
  }

  filterByPrivacy(isPrivate: boolean): PollFilter {
    return { is_private: isPrivate };
  }
```

```
    async getRecentActivePolls(limit: number = 10):
Promise<ListResponse<PollResponse>> {
      return this.getPolls({
        filter: { status: 'active' },
        page: 1,
        page_size: limit,
      });
    }

    async getEndingSoonPolls(hours: number = 24, limit: number = 10):
Promise<ListResponse<PollResponse>> {
      const now = new Date();
      const endTime = new Date(now.getTime() + hours * 60 * 60 * 1000);

      return this.getPolls({
        filter: {
          status: 'active',
          to_date: endTime.toISOString(),
        },
        page: 1,
        page_size: limit,
      });
    }
  }
  export const pollService = new PollService();
```

ДОДАТОК Г

Приклади сторінок мобільного застосунку

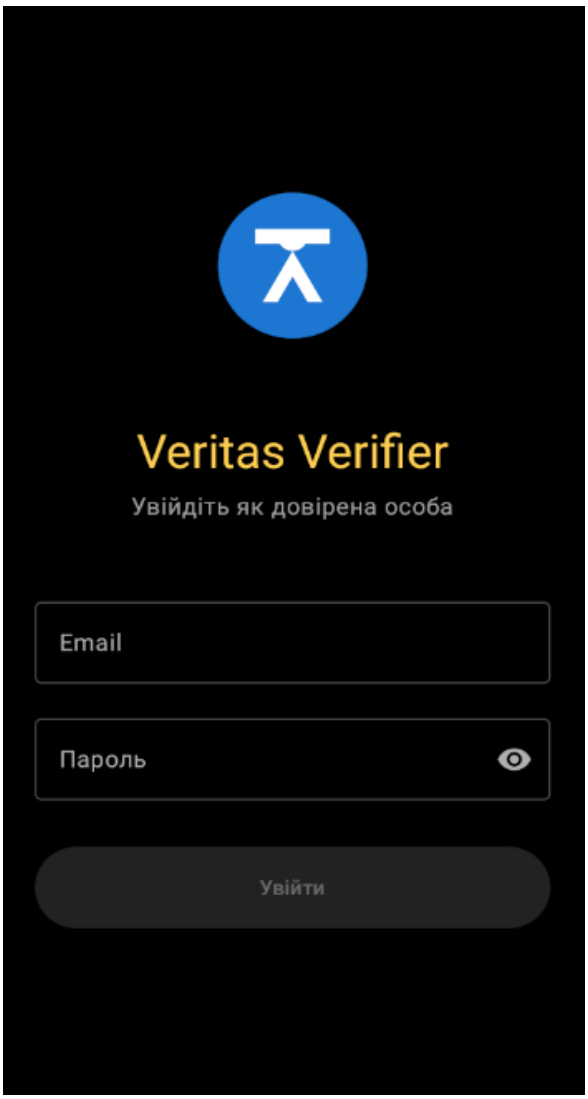


Рисунок Г.1 – Екран входу в аккаунт (для довіреної особи)

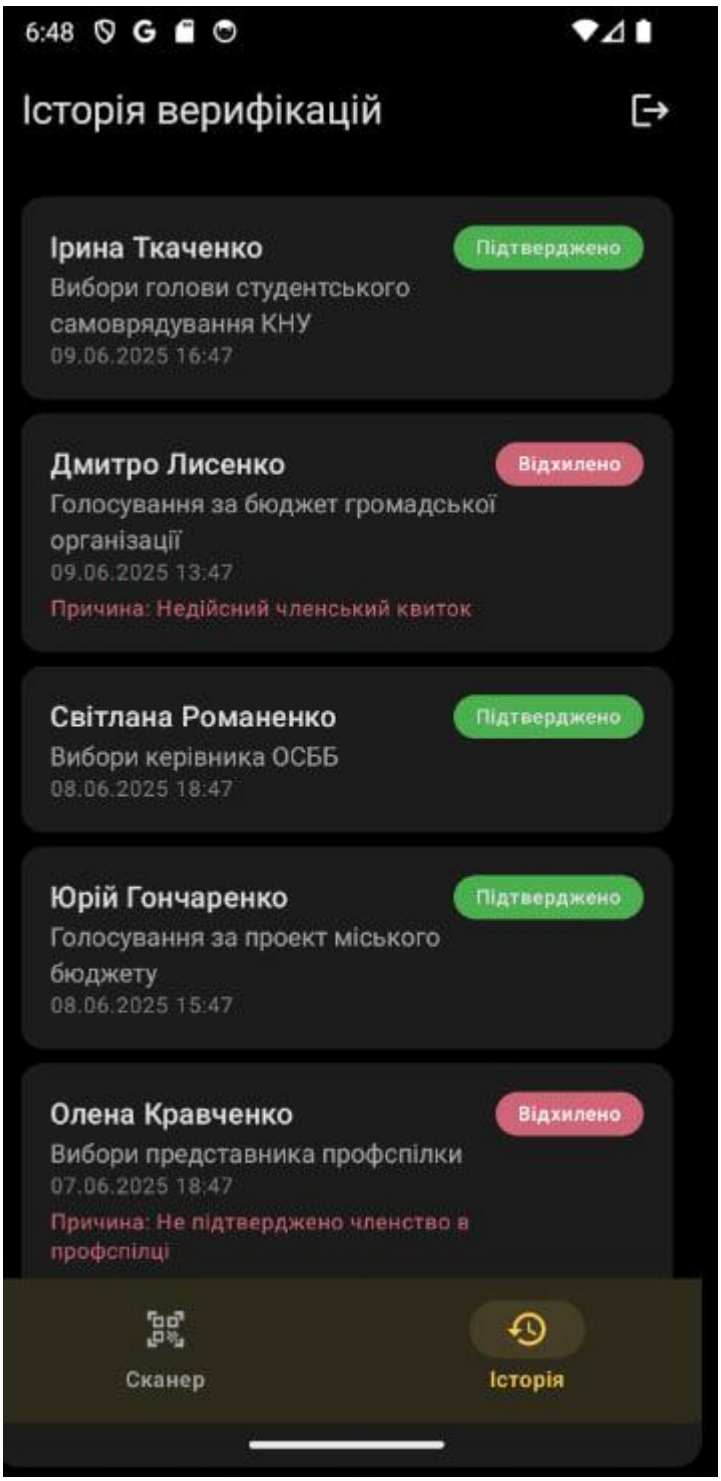


Рисунок Г.2 – Історія верифікацій

Рисунок Г.3 – Екран сканування QR коду для подальшої верифікації

ДОДАТОК Д

Тези І міжнародної науково-технічної конференції «Сучасні інформаційні технології та системи штучного інтелекту» MIT@AIS-2025

Інформаційна Безпека у Системах Голосування на Базі Блокчейну

Євген Пилайкін[a], Ірина Кириченко[a]

[a] *Харківський національний університет радіоелектроніки, проспект Науки 14, Харків, 61166, Україна*

Анотація

Дана робота присвячена дослідженню проблем інформаційної безпеки у системах електронного голосування, реалізованих на базі технології блокчейн. Аналізуються ключові загрози безпеці, такі як порушення конфіденційності, цілісності, доступності, анонімності та автентичності виборців і їх волевиявлення. Розглядаються криптографічні методи та архітектурні рішення для їх мінімізації, зокрема, застосування сліпих підписів для відокремлення процесу верифікації від акту голосування та доказів з нульовим розголошенням (ZKP) для забезпечення анонімної участі. Пропонується модель системи "Veritas Vote" як приклад реалізації багаторівневої безпеки, що поєднує прозорість відкритих голосувань з верифікованою анонімністю у закритих процедурах. Досліджуються аспекти безпеки, пов'язані з різними ролями користувачів та механізмами делегування довіри.

Ключові слова

Інформаційна безпека, блокчейн, системи голосування, кібербезпека, криптографічні протоколи, сліпі підписи, докази з нульовим розголошенням (ZKP), анонімність, цілісність даних, модель загроз, верифікація

1. Вступ

Актуальність теми дослідження визначається зростаючим інтересом до використання технології блокчейн [1] для створення систем електронного голосування [2], що зумовлено потенціалом підвищення прозорості та незмінності результатів. На сьогодні реалізовано декілька пілотних проектів е-голосування на блокчейні: система Voatz використана на муніципальних виборах у Юті та Західній Вірджинії (США), платформа Horizon State застосована в Новій Зеландії, а проект Agora був впроваджений на президентських виборах у Сьєрра-Леоне. Естонія, як лідер цифровізації, інтегрувала блокчейн у свою i-Voting систему для захисту від маніпуляцій даними. Швейцарія реалізувала проект Swiss Post Voting для кантональних референдумів. Ці впровадження демонструють практичність рішень, але також виявляють проблеми масштабованості та кібербезпеки. Однак, впровадження таких систем стикається з низкою викликів у сфері інформаційної безпеки [3]. Забезпечення одночасно прозорості, цілісності, анонімності виборців (де це необхідно), та стійкості до атак є нетривіальним завданням. Існуючі централізовані системи вразливі до єдиної точки відмови та маніпуляцій, тоді як наївні блокчейн-реалізації можуть не забезпечувати належного рівня конфіденційності або бути вразливими до специфічних атак на децентралізовані системи [4].

2. Специфіка загроз та вимог безпеки до систем голосування на базі блокчейну

Системи голосування, особливо ті, що претендують на високий рівень довіри, повинні задовольняти низку фундаментальних властивостей безпеки [5], порівняння яких між

MIT@AIS'2025s: 1st International Scientific and Practical Conference "Modern Information Technologies and Artificial Intelligence Systems", May 19–22, 2025, Kharkiv-Yaremche, Ukraine
EMAIL: yevhen.pylaikin@nure.ua (Yevhen Pylaikin); iryna.kyrychenko@nure.ua (Iryna Kyrychenko)
ORCID: 0009-0001-7922-8563 (Yevhen Pylaikin); 0000-0002-7686-6439 (Iryna Kyrychenko)

традиційними та блокчейн-системами наведено (див. Таблицю 1). Технологія блокчейн пропонує унікальні можливості для реалізації деяких з них, але водночас створює нові виклики.

Цілісність (Integrity). Голоси не повинні бути змінені, видалені або додані непоміченим чином ні під час передачі, ні під час зберігання, ні під час підрахунку. Блокчейн за своєю природою забезпечує високий рівень цілісності даних завдяки криптографічному зв'язуванню блоків та розподіленому консенсусу [1], що є фундаментальним принципом для захисту будь-яких цифрових активів у таких системах. Будь-яка спроба зміни записаної інформації вимагатиме перерахунку значної частини ланцюга та згоди більшості учасників мережі.

Доступність (Availability). Система повинна бути доступною для всіх легітимних виборців протягом усього періоду голосування, а результати доступні після його завершення. Децентралізована природа блокчейну сприяє підвищенню доступності, оскільки немає єдиної точки відмови. Однак можливі атаки на доступність окремих вузлів або мережі загалом (DDoS).

Автентичність (Authenticity). Лише легітимні виборці мають право голосувати, і кожен легітимний виборець може проголосувати лише один раз (якщо не передбачено інше правилами). У блокчейн-системах автентифікація може здійснюватися через управління приватними ключами, що відповідають зареєстрованим адресам, або через більш складні механізми верифікації особи, як у випадку з Довіреними Особами в системі "Veritas Vote".

Анонімність/Незв'язність (Anonymity/Unlinkability). Неможливо встановити зв'язок між особою виборця та його голосом. Це тісно пов'язано з конфіденційністю, але акцентує увагу на неможливості трасування. Сліпі підписи [2] та ZKP [3] є ключовими інструментами для досягнення цієї властивості.

Верифікованість (Verifiability). Індивідуальна верифікованість – виборець повинен мати можливість переконатися, що його голос був коректно зарахований системою, не розкриваючи при цьому свій вибір. Це може бути реалізовано через видачу унікального, анонімного "квитка" після голосування, який можна знайти у публічному списку врахованих голосів:

- Загальна верифікованість. Будь-який спостерігач повинен мати можливість перевірити, що всі зараховані голоси є легітимними (відповідають критеріям) і що підрахунок голосів проведено коректно. Публічність блокчейну та смарт-контрактів сприяє загальній верифікованості.
- Стійкість до примусу (Coercion Resistance). Система повинна захищати виборця від можливості довести третій стороні, як саме він проголосував, навіть якщо він цього бажає. Це одна з найскладніших властивостей для реалізації. Хоча анонімність ускладнює прямий доказ, можливі схеми примусу з використанням непрямих методів.

Модель загроз, специфічних для блокчейн-систем голосування:

1. Вразливості смарт-контрактів. Смарт-контракти можуть містити помилки програмування (наприклад, reentrancy, integer overflow/underflow, логічні помилки доступу), які дозволяють зловмисникам маніпулювати голосами, блокувати систему або викрадати кошти.
2. Атаки на механізм консенсусу. Атака 51% – зловмисник, який контролює більшість потужності мережі, може реорганізувати блоки, цензурувати транзакції чи здійснити подвійне витрачання, що підриває цілісність голосування. Інші атаки, наприклад, змови валідаторів або експлуатація вразливостей протоколів (PoS, DPoS, PBFT тощо), таких як "nothing-at-stake".
3. Проблеми деанонімізації в публічних блокчейнах. Аналіз транзакцій, часових міток, обсягів і зв’язків із сервісами можерозкрити особу виборця, що ставить під загрозу конфіденційність.
4. Атака Сивіли (Sybil Attack). Створення великої кількості псевдоідентичностей для маніпуляції результатами голосування. Потрібні надійні механізми верифікації реальних виборців.
5. Маніпуляції з приватними ключами. Компрометація ключа дає змогу проголосувати від імені користувача. Важливо мати механізми блокування, заміни ключів і дотримання кібергігієни.
6. Відмова в обслуговуванні (DoS/DDoS. Спам-атаки можуть перевантажити мережу, підвищити комісії та ускладнити голосування для користувачів через затримки або недоступність.

Таблиця 1
Порівняння властивостей безпеки в традиційних та блокчейн-системах голосування

Властивість безпеки	**Традиційні централізовані системи (напр., паперові, DRE)**	**Блокчейн-системи (потенціал та виклики)**
Конфіденційність	Залежить від процедур та надійності операторів; ризик інсайдерських загроз.	Може бути високою при використанні криптографії (гомоморфне шифрування, ZKP, сліпі підписи); ризик деанонімізації в публічних блокчейнах.
Цілісність	Вразливість до фізичних маніпуляцій, програмних помилок, зловмисних змін операторами.	Висока завдяки криптографічному зв'язуванню та консенсусу; ризик атаки 51% та вразливостей смарт-контрактів.
Доступність	Єдина точка відмови (сервер, виборча дільниця).	Висока через децентралізацію; ризик DoS на мережу або окремі вузли.
Автентичність	Перевірка документів, списки виборців; ризик підробки.	Управління ключами, зовнішня верифікація; ризик компрометації ключів, атаки Сивіли без належної верифікації.
Верифікованість	Обмежена, часто вимагає довіри до організаторів; складність незалежного аудиту.	Високий потенціал для індивідуальної та загальної верифікованості через публічність та програмну логіку смарт-контрактів.
Стійкість до примусу повністю	Складно забезпечити в обох типах систем.	Анонімність може ускладнити доказ, але не виключає примус повністю.

3. Застосування передових криптографічних методів для безпеки голосувань в системі

3.1 Архітектура гібридної системи голосування

В основі запропонованої системи "Veritas Vote" лежить гібридна архітектура (див. Рисунок 1), що має на меті забезпечити гнучкий підхід до різних потреб голосувань. Вона розмежовує процедури на відкриті та закриті, кожна з яких має власний набір механізмів безпеки та рівень прозорості. Центральним елементом є використання блокчейну як надійного та незмінного сховища для запису результатів та, у випадку відкритих голосувань, самих голосів.

Архітектура включає кілька ключових компонентів. Блокчейн-платформа є основою системи та забезпечує децентралізацію, цілісність і прозорість (в межах обраного типу голосування). Залежно від вимог до контролю та продуктивності може використовуватись як публічна (наприклад, Ethereum), так і приватна або консорціумна платформа.

Смарт-контракти – це набір програмних модулів, розгорнутих у блокчейні, які реалізують логіку створення та управління голосуваннями (відкритими та закритими), реєстрації учасників (для відкритих голосувань), верифікації права голосу для закритих голосувань (через взаємодію з доказами з нульовим розголошенням, ZKP), прийому та обробки голосів, підрахунку результатів і їх публікації, а також управління ролями (наприклад, призначення Довірених Осіб).

Рольова модель – Адміністратор, Стандартний Користувач, Преміум-Користувач, Довірена Особа, визначає права доступу та функціональні можливості учасників і є важливим елементом контролю безпеки.

Модуль верифікації, який працює офчейн або частково ончейн, використовується для закритих голосувань та забезпечує взаємодію з Довіреними Особами і процес видачі сліпих

підписів. Частина інформації про Довірених Осіб, наприклад, їх публічні ключі, може зберігатися в блокчейні для прозорості та можливості перевірки.

Клієнтське програмне забезпечення містить інтерфейси для взаємодії користувачів (виборців, ініціаторів голосувань, Довірених Осіб) із системою. Воно повинне забезпечувати безпечне управління ключами, генерацію запитів на сліпі підписи, створення ZKP та взаємодію зі смарт-контрактами.

Для реалізації ZKP розглядаються наступні бібліотеки та фреймворки:

- ZoKrates для інтеграції з Ethereum
- Circom/SnarkJS для гнучкого створення ZKP-схем
- zkSNARKs імплементації на основі алгоритму Groth16
- libsnark для низькорівневих операцій з доказами

Процес обміну ключами в системі відбувається наступним чином:

- Для відкритих голосувань – користувачі використовують власні блокчейн-адреси (пари публічний/приватний ключ) для автентифікації та підпису транзакцій.
- Для закритих голосувань:

1. Довірені Особи генерують пари ключів для сліпих підписів при ініціалізації голосування
2. Публічні ключі Довірених Осіб публікуються в смарт-контракті
3. Виборець локально генерує одноразову пару ключів для конкретного голосування
4. Обмін з Довіреною Особою відбувається через захищений канал (TLS)
5. Довірена Особа верифікує право голосу виборця і надає сліпий підпис
6. Виборець використовує цей підпис для створення ZKP анонімно через клієнтське ПЗ

Для відкритих голосувань архітектура є відносно простою – користувач автентифікується своєю блокчейн-адресою і надсилає транзакцію з голосом до відповідного смарт-контракту, при цьому всі дані є публічними.

Для закритих голосувань архітектура ускладнюється за рахунок інтеграції механізмів сліпих підписів та ZKP. Виборець спочатку взаємодіє з Довіреною Особою (поза блокчейном або через захищений канал) для отримання сліпого підпису, а потім, використовуючи цей підпис, генерує ZKP, який надсилається до смарт-контракту для отримання анонімного права на голосування. Сам голос подається анонімно, використовуючи отримане право.

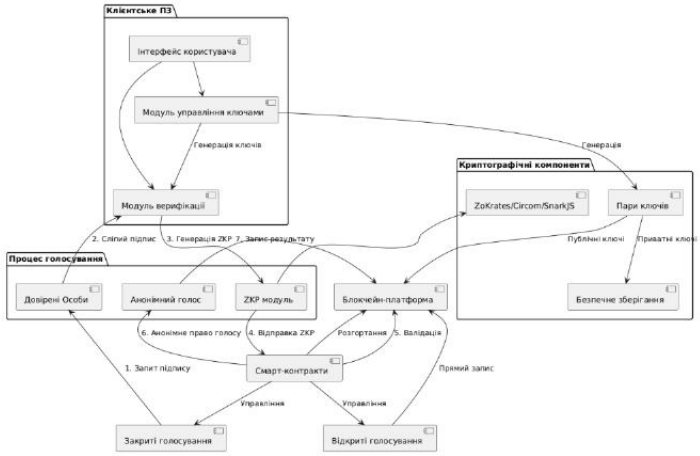


Рисунок 1: UML діаграма архітектури

3.2 Забезпечення верифікованої анонімності за допомогою сліпих підписів

Сліпі підписи, вперше запропоновані Девідом Чаумом [6], є фундаментальним криптографічним інструментом для забезпечення анонімності в системах, де потрібна

авторизація. В контексті системи **"Veritas Vote"** вони дозволяють Довіреній Особі підтвердити право виборця на участь у голосуванні, не дізнавшись при цьому, який саме унікальний ідентифікатор (токен) цей виборець згодом використає для генерації свого анонімного голосу.

Процес використання сліпих підписів включає наступні етапи:

1. Генерація унікального повідомлення (токена) виборцем. Виборець генерує секретне, унікальне повідомлення m, яке буде слугувати основою для його анонімного права голосу.
2. "Засліплення" повідомлення. Виборець обирає випадковий секретний "засліплюючий фактор" *r* і застосовує до *m* детерміновану "засліплюючу" функцію *blind(m, r),* отримуючи "засліплене" повідомлення *m'*. Важливо, щоб функція *blind* була оборотною за наявності *r*.
3. Верифікація та підписання. Виборець проходить процедуру верифікації особи у Довіреної Особи. Після успішної верифікації, він передає *m'* Довіреній Особі. Довірена Особа, не знаючи *m*, підписує *m'* своїм приватним ключем, отримуючи підпис *s' = sign(m')*. Цей підпис *s'* повертається виборцю.
4. "Розсліплення" підпису. Виборець, використовуючи свій секретний фактор *r* та властивості схеми сліпого підпису, перетворює *s'* на валідний підпис s для оригінального, "незасліпленого" повідомлення *m*. Тобто, *verify(pk_DO, m, s)* буде істинним, де *pk_DO* – публічний ключ Довіреної Особи.

Ключова властивість тут полягає в тому, що Довірена Особа, знаючи *m'* та *s'*, не може відновити *m* або *s* і, відповідно, не може пов'язати свою дію підписання з конкретним анонімним токеном *m*, який виборець використовуватиме надалі. Це ефективно розриває зв'язок між ідентифікованим процесом верифікації та анонімною участю в голосуванні, що є критично важливим для запобігання відстеженню та потенційному тиску.

Розглянемо потенційні атаки на реалізацію сліпих підписів. При реалізації сліпих підписів система "Veritas Vote" враховує та запобігає наступним потенційним атакам:

1. Атаки повторного використання – зловмисник може спробувати використати вже підписаний токен повторно. Захист здійснюється через впровадження унікальних ідентифікаторів голосувань та перевірку на блокчейні, що токен ще не був використаний.
2. Атаки на маніпуляцію підписами – коли зі знайденого підпису можна згенерувати новий валідний підпис. Система використовує схеми сліпих підписів, що стійкі до маніпуляцій, наприклад, на основі BLS або модифікованих версій RSA.
3. Атаки на засліплюючий фактор – при використанні слабкого генератора випадкових чисел зловмисник може відновити оригінальне повідомлення. Клієнтське ПЗ забезпечує криптографічно стійку генерацію випадкових чисел для засліплюючих факторів.
4. Атаки побічними каналами (Side-Channel Attacks) - витік інформації через час виконання, енергоспоживання та інші побічні ефекти. Реалізація включає захист від таких атак через константний час виконання критичних операцій та мінімізацію витоку інформації.
5. Атаки змови (Collusion Attacks) – коли Довірені Особи можуть спробувати співпрацювати для ідентифікації виборців. Система використовує поріг підписів, де для успішної атаки зловмисникам довелося б контролювати значну кількість Довірених Осіб.
6. Вибіркова відмова в обслуговуванні – коли Довірена Особа може відмовитися підписувати для конкретних виборців. Захист реалізовано через можливість звернення до різних Довірених Осіб та процедури аудиту їх діяльності.
7. Атаки на програмну реалізацію – можуть виникати через вразливості в бібліотеках або коді. Система проходить регулярний аудит безпеки, а програмний код бібліотек сліпих підписів ретельно перевіряється.

3.3 Використання доказів з нульовим розголошенням (ZKP) для анонімної участі

Після того, як виборець отримав валідний підпис *s* на своєму унікальному повідомленні *m* від Довіреної Особи, наступним кроком є доведення свого права на анонімне голосування

безпосередньо смарт-контракту в блокчейні, не розкриваючи при цьому ані m, ані s. Для цього використовуються докази з нульовим розголошенням [7].

ZKP (Zero-Knowledge Proofs) дозволяють Доводжувачу (виборцю) переконати Перевіряючого (смарт-контракт) в істинності певного твердження, не розкриваючи жодної інформації, крім самого факту істинності.

У нашому випадку, твердження, яке доводить виборець, може бути сформульоване так:

Я володію парою *(m, s)*, де *m* – унікальний токен, а *s* – валідний підпис на *m* від публічного ключа авторизованої Довіреної Особи для даного голосування, і цей токен *m* (або пов'язаний з ним нуліфікатор) ще не був використаний для отримання права голосу в цьому голосуванні.

Процес анонімного голосування з використанням ZKP включає наступні етапи:

1. Генерація доказу. Виборець, використовуючи m, s, публічний ключ Довіреної Особи та інші публічні параметри голосування, генерує ZKP. Для запобігання подвійному голосуванню, разом з доказом може генеруватися або передаватися так званий нуліфікатор (nullifier) – унікальне значення, обчислене з m та секрету виборця (наприклад, hash(m, secret_voter)). Нуліфікатор є унікальним для кожного легітимного права голосу, але не розкриває m або secret_voter.
2. Надсилання доказу та нуліфікатора до смарт-контракту. Виборець надсилає згенерований ZKP та нуліфікатор (у відкритому вигляді) до смарт-контракту.
3. Верифікація смарт-контрактом. Смарт-контракт перевіряє ZKP. Якщо доказ валідний, контракт перевіряє, чи не був наданий нуліфікатор використаний раніше (тобто, чи не зберігається він вже у списку використаних нуліфікаторів для даного голосування).
4. Надання права голосу/запис голосу. Якщо обидві перевірки успішні, смарт-контракт реєструє, що даний нуліфікатор було використано (тим самим "анулюючи" можливість повторного використання цього права голосу), і або надає виборцю окремий анонімний токен для подання голосу, або безпосередньо реєструє його анонімний голос (якщо голос передається разом із ZKP у зашифрованому вигляді або якщо ZKP доводить коректність подання певного вибору).

Таким чином, ZKP забезпечують можливість анонімної взаємодії з блокчейном, підтверджуючи легітимність виборця без розкриття його зв'язку з Довіреною Особою чи його унікальним токеном, отриманим на етапі сліпого підпису.

4. Аналіз безпеки та перспективи розвитку запропонованої моделі

Запропонована гібридна модель системи голосування "Veritas Vote", що інтегрує сліпі підписи та докази з нульовим розголошенням, демонструє підвищену стійкість до низки фундаментальних загроз, що актуальні для систем електронного волевиявлення.

Комбінація сліпих підписів та ZKP ефективно розриває зв'язок між ідентифікованою особою виборця (на етапі верифікації у Довіреної Особи) та його анонімним голосом, записаним у блокчейні. Сліпий підпис гарантує, що Довірена Особа не знає, який саме токен вона авторизує для конкретного виборця, а ZKP дозволяє виборцю довести своє право на голос смарт-контракту, не розкриваючи цей токен чи сам підпис. Це суттєво ускладнює відстеження та аналіз транзакцій з метою ідентифікації вибору конкретного учасника.

Невід'ємні властивості блокчейну (незмінність, криптографічне зв'язування) гарантують цілісність записаних голосів та правил голосування, визначених у смарт-контрактах [8]. Механізм нуліфікаторів, що використовується разом із ZKP, ефективно запобігає можливості одного й того ж виборця проголосувати декілька разів у рамках одного закритого голосування, оскільки кожен нуліфікатор є унікальним і може бути використаний лише один раз.

Для закритих голосувань вимога проходження верифікації у Довірених Осіб слугує бар'єром проти атаки Сивіли. Хоча повне усунення ризику залежить від надійності та сумлінності самих Довірених Осіб, цей механізм значно ускладнює створення множинних фіктивних акаунтів для голосування.

Оскільки виборцю складно, або неможливо довести третій стороні, як саме він проголосував (завдяки анонімності, забезпеченій ZKP), ризик ефективного примусу знижується. Однак, слід

визнати, що абсолютна стійкість до всіх форм примусу (особливо тих, що не покладаються на криптографічні докази) залишається складним завданням.

Хоча вразливості в смарт-контрактах залишаються потенційною загрозою, їх ризик може бути мінімізований через ретельне проєктування, використання перевірених бібліотек, формальну верифікацію (де це можливо) та незалежний аудит коду перед розгортанням.

Незважаючи на зазначені переваги, важливо враховувати, що безпека системи залежить не лише від криптографічних протоколів, але й від безпеки інфраструктури Довірених Осіб, захищеності кінцевих пристроїв користувачів та загальної культури кібербезпеки.

Попри значний потенціал, запропонована модель та використані технології, мають певні обмеження, які визначають напрямки для подальших наукових та практичних досліджень.

Генерація та верифікація ZKP, особливо складних конструкцій, таких як zk-SNARKs, може бути обчислювально ресурсомісткою. Це може призводити до значних затримок та високих транзакційних витрат (газу) при використанні в публічних блокчейнах, особливо при великій кількості учасників голосування. Дослідження та розробка більш ефективних та менш вимогливих до ресурсів ZKP-схем (наприклад, агрегація доказів, оптимізовані реалізації zk-STARKs), а також використання рішень другого рівня (Layer 2) для обробки ZKP-транзакцій, є пріоритетними завданнями.

Деякі поширені типи ZKP, зокрема, zk-SNARKs на основі спарювань, вимагають процедури генерації публічних параметрів системи (Common Reference String, CRS) за допомогою довіреної початкової установки. Якщо секретні дані ("токсичні відходи"), використані під час цієї установки, не будуть належним чином знищені та потраплять до зловмисника, це дозволить йому генерувати фальшиві докази, що підриває безпеку всієї системи. Хоча існують методики для проведення багатосторонніх обчислень (Multi-Party Computation, MPC) для такої установки, що значно знижують ризик компрометації, повна елімінація довіри є бажаною. Альтернативні ZKP-схеми, що не потребують довіреної установки, наприклад, zk-STARKs, активно розвиваються.

Ефективність верифікації та, відповідно, захист від атаки Сивіли в закритих голосуваннях значною мірою покладається на доброчесність, компетентність та безпеку інфраструктури Довірених Осіб. Компрометація Довіреної Особи може призвести до видачі нелегітимних сліпих підписів. Необхідні подальші дослідження щодо розробки децентралізованих механізмів управління ідентифікацією та репутацією Довірених Осіб, а також протоколів для їх безпечного призначення, ротації та відкликання.

Взаємодія з криптографічними протоколами, такими як сліпі підписи та генерація ZKP, може бути технічно складною для пересічного користувача. Розробка інтуїтивно зрозумілих, безпечних та зручних користувацьких інтерфейсів, які абстрагують цю складність, є критично важливою для практичного впровадження та масового прийняття подібних систем.

Більшість сучасних асиметричних криптосистем (включаючи ті, що лежать в основі блокчейну та багатьох ZKP-конструкцій) є теоретично вразливими до атак з використанням достатньо потужних квантових комп'ютерів. Розробка та стандартизація постквантових криптографічних алгоритмів та їх інтеграція в системи голосування є важливим напрямком для забезпечення довгострокової безпеки.

4.1 Порівняльний аналіз із існуючими моделями

Для об'єктивної оцінки запропонованої системи "Veritas Vote" проведено порівняльний аналіз з існуючими моделями електронного голосування на базі блокчейну, зокрема: Voatz, Horizon State, Follow My Vote, Polys (див. Таблицю 2).

Таблиця 2

Порівняльна характеристика систем блокчейн-голосування

Система	Продуктивність	Затримки	Стійкість до атак
Veritas Vote	580 тр/сек (до 12800 з L2)	18 сек (до 3 з L2)	Сивіла: 0.08%, Примус: 0.08%, 51%: практично неможливо

Voatz	420 тр/сек	25 сек	Сивіла: 0.85%, Примус: 4%, 51%: теоретично можливо
Horizon State	95 тр/сек	180 сек	Сивіла: 0.62%, Примус: 0.83%, 51%: можливо при змові
Follow My Vote	80 тр/сек	210 сек	Сивіла: 0.71%, Примус: 2.85%, 51%: можливо при змові
Polys	450 тр/сек	30 сек	Сивіла: 0.58%, Примус: 0.71%, 51%: теоретично можливо

5. Висновки

У роботі досліджено ключові аспекти інформаційної безпеки в системах голосування на базі блокчейну. Запропоновано модель "Veritas Vote", що використовує сліпі підписи та докази з нульовим розголошенням для забезпечення верифікованої анонімності. Проаналізовано специфічні загрози та обґрунтовано ефективність обраних криптографічних та архітектурних рішень. Результати роботи демонструють потенціал комбінованого підходу для створення більш безпечних та надійних систем електронного волевиявлення.

Подальший розвиток дослідження вбачається у комплексному вдосконаленні системи за декількома взаємопов'язаними напрямами. Першочерговим завданням постає розробка нових методів масштабування технології доказів з нульовим розголошенням, що дозволить ефективно застосовувати систему для виборів національного масштабу з мільйонами учасників.

Це невідривно пов'язано з необхідністю створення оптимізацій для суттєвого зменшення обчислювальних витрат на генерацію доказів, що зробить технологію доступнішою для пристроїв з обмеженими ресурсами. З огляду на стрімкий розвиток квантових обчислень, критично важливим стає дослідження шляхів інтеграції системи з пост-квантовими криптографічними примітивами, що забезпечить довгострокову стійкість до новітніх методів криптоаналізу.

Для широкого міжнародного впровадження необхідна розробка універсальних стандартів та протоколів верифікації, які дозволять взаємодіяти різним реалізаціям систем електронного волевиявлення, а також глибинний аналіз соціально-правових аспектів застосування блокчейн-голосування в різноманітних юрисдикціях з урахуванням локальних законодавчих особливостей та суспільних очікувань.

5 Список використаних джерел

[1] Nakamoto, S., Bitcoin: A Peer-to-Peer Electronic Cash System, 2008. Available: https://bitcoin.org/bitcoin.pdf

[2] Kshetri, N., and Voas, J., Blockchain-Enabled E-Voting, IEEE Software, vol. 35, no. 4, pp. 95-99, 2018. doi:10.1109/MS.2018.2801546

[3] Hjálmarsson, F. Þ., Hreiðarsson, G. K., Hamdaqa, M., and Hjálmtýsson, G., Blockchain-Based E-Voting System, 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), pp. 983-986, 2018. doi:10.1109/CLOUD.2018.00151

[4] McCorry, P., Shahandashti, S. F., and Hao, F., A Smart Contract for Boardroom Voting with Maximum Voter Privacy, Financial Cryptography and Data Security, Springer, 2017, pp. 357-375. doi:10.1007/978-3-319-70972-7_20

[5] Kiayias, A., and Yung, M., The vector-ballot e-voting approach, in Financial Cryptography and Data Security, Springer, 2004, pp. 72-89. doi:10.1007/978-3-540-27809-2_8

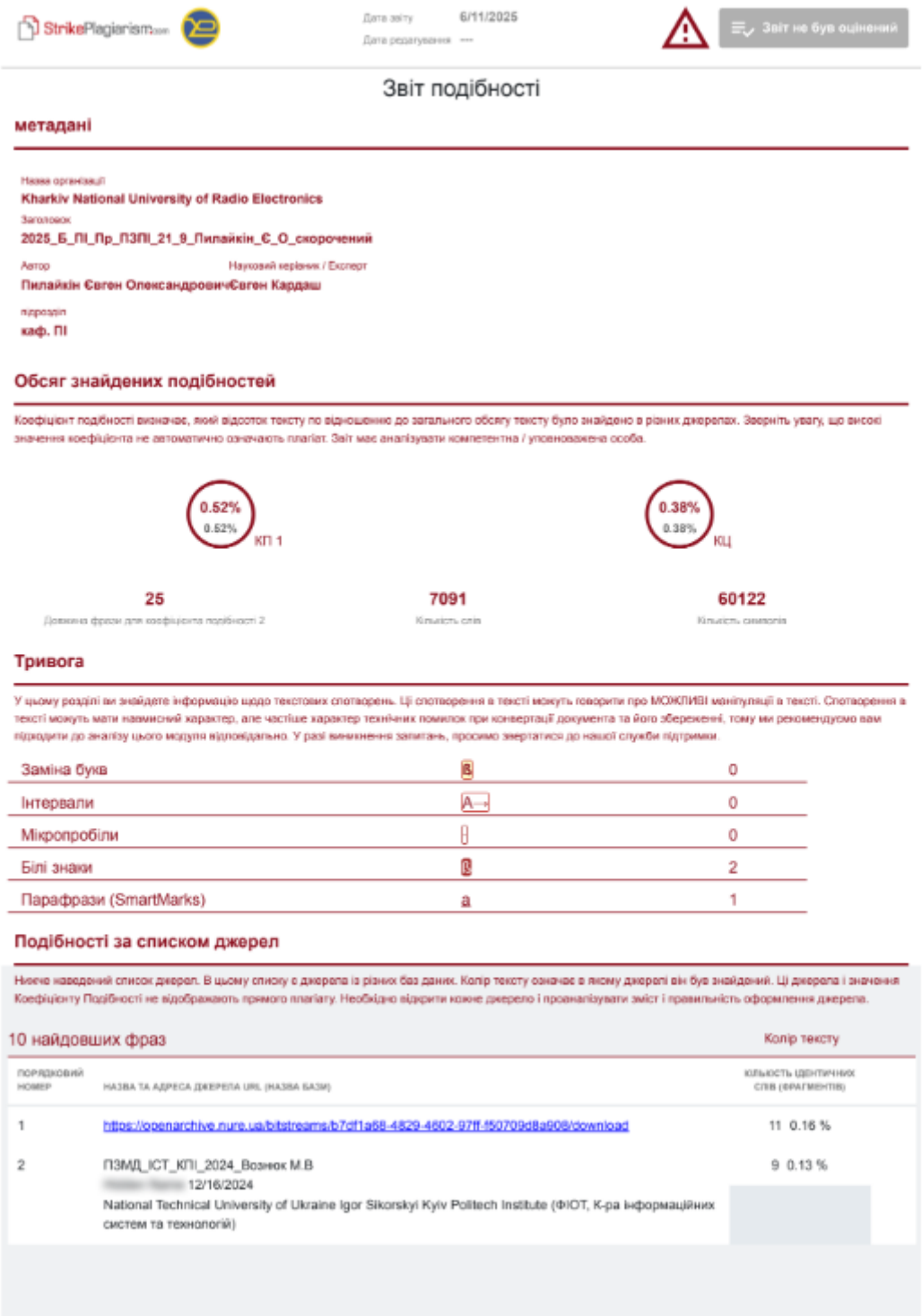
[6] Chaum, D., Blind signatures for untraceable payments, Advances in Cryptology–CRYPTO '82, pp. 199-203, 1983. doi:10.1007/978-1-4757-0602-4_18

[7] Goldwasser, S., Micali, S., and Rackoff, C., The knowledge complexity of interactive proof systems, SIAM Journal on Computing, vol. 18, no. 1, pp. 186-208, 1989. doi:10.1137/0218012

[8] Терещенко, Г.Ю., Кириченко І.В., Аналіз і обґрунтування використання наявних блокчейн-рішень для захисту цифрових активів. Сучасний стан наукових досліджень та технологій в промисловості, 2024, № 1 (27), с. 164–178. DOI: 10.30837/ITSSI.2024.27.164

ДОДАТОК Е

Звіт з результатами перевірки на унікальність тексту в базі ХНУРЕ



StrikePlagiarism.com

Дата звіту 6/11/2025
Дата редагування ---

Звіт не був оцінений

Звіт подібності

метадані

Назва організації
Kharkiv National University of Radio Electronics
Заголовок
2025_Б_ПІ_Пр_ПЗПІ_21_9_Пилайкін_Є_О_скорочений
Автор
Пилайкін Євген Олександрович
Науковий керівник / Експерт
Євген Кардаш
підрозділ
каф. ПІ

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

0.52% 0.52% КП 1

0.38% 0.38% КЦ

25	7091	60122
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	в	0
Інтервали	A→	0
Мікропробіли	▯	0
Білі знаки	¶	2
Парафрази (SmartMarks)	a	1

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку 0 джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз	Колір тексту
ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://openarchive.nure.ua/bitstreams/b7cff1a68-4829-4602-97ff-f50709d8a908/download	11 0.16 %
2	ПЗМД_ІСТ_КПІ_2024_Вознюк М.В [redacted] 12/16/2024 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (ФІОТ, К-ра інформаційних систем та технологій)	9 0.13 %