КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ РАДІОФІЗИКИ, ЕЛЕКТРОНІКИ ТА КОМП'ЮТЕРНИХ СИСТЕМ

Кафедра комп'ютерної інженерії

До захисту допущено: «На правах рукопису»
Завідувач кафедри _____ Юрій Бойко
« _ » _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему:

«СИСТЕМА ОНЛАЙН ГОЛОСУВАНЬ ДЛЯ ВИБОРІВ В ОРГАНАХ СТУДЕНТСЬКОГО САМОВРЯДУВАННЯ»

Виконав:
студент 4-го курсу бакалаврату
денної форми навчання
спеціальності 123 Комп'ютерна інженерія
ОНП «_____»
Андрій Петровський _____

Науковий керівник:
кандидат технічних наук, асистент
Євген Слюсар _____

Рецензент:
кандидат фізико-математичних наук, доцент
В'ячеслав Борецький _____

Засвідчую, що у цій бакалаврській роботі
немає запозичень з праць інших авторів без
відповідних посилань
Студент _____

Робота допущена до захисту в ЕК рішенням кафедри _____
від «__»_____ 2023 р., протокол № __.

Завідувач кафедри _____,
кандидат фізико-математичних наук, доцент
Бойко Юрій Володимирович

(підпис)

Київ – 2023

РЕФЕРАТ

Об’єктом роботи є процес розробки веб-застосунку для проведення онлайн виборів в органах студентського самоврядування.

Метою роботи є створення протоколу онлайн виборів який задовольняє всі вимоги прямих таємних онлайн виборів.

Методи розроблення: визначення вимог до веб-застосунку, дослідження існуючих протоколів онлайн виборів, створення протоколу який задовольнятиме вимоги, технічна реалізація із застосуванням існуючих механізмів шифрування та передачі інформації.

Результат роботи: створено протокол онлайн виборів для ОСС КНУ, технічна реалізація протоколу. Інтерфейс для користувача та для Центральної виборчої комісії студентів.

Розроблене рішення може бути використано у якості демонстрації процедури онлайн виборчого процесу, та стати в основу застосунку для онлайн виборів в ОСС КНУ та діджиталізації бюрократичних процесів в ОСС КНУ.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

ОСС - органи студентського самоврядування

КНУ - Київський національний університет

СП - студентський парламент

ЦВКс - центральна виборча комісія студентів

ВКС - виборча комісія студентів

ВСТУП

Оцінка сучасного стану розробки. Органи студентського самоврядування інших ЗВО вже мають свої системи, проте немає універсального рішення для всіх ОСС ЗВО, яке би відповідало всім вимогам до онлайн виборів.

Актуальність роботи та підстави для її виконання. Спочатку через SARS-CoV-2 (2019-nCoV), потім – через повномасштабне вторгнення Російської Федерації в Україну студенти втратили можливість проводити повноцінні вибори на посади голови Студентського парламенту структурного підрозділу (факультету або інституту). У 2021 році в Київському національному університеті імені Тараса Шевченка було проведено останні, на даний момент, вибори, які показали дуже низьку явку в порівнянні з попередніми роками, 2019 р. – близько 2 500 студентів, 2018 р. – близько 3 500 студентів, так як навчання відбувалось дистанційно, з використанням інформаційно-комунікаційних технологій, тому не всі студенти мали можливість відвідати виборчі дільниці. При такій низькій явці з урахуванням того, що в КНУ навчались на той момент близько 30 000 студентів денної форми навчання, постає питання, наскільки вибори є коректними, адже на деяких маленьких структурних підрозділах перемогти міг той, хто просто матиме більше зв’язків зі свого структурного підрозділу у Києві. Під час повномасштабного вторгнення вибори неможливо провести, так як не всі студенти мають можливість навіть приїхати в Київ, що позбавляє їх права на волевиявлення. Система розробляється задля забезпечення можливості проведення виборів при будь-яких обставинах.

Мета й завдання роботи. Метою є розробка системи для проведення онлайн-виборів. Для досягнення цієї мети поставлено такі завдання.

- Дослідити існуючі протоколи онлайн голосування.
- Дослідити існуючі засоби шифрування та передачі даних
- Дослідити існуючі способи підтвердження особи та верифікування даних.
- Дослідити можливості запобігання втручання сторонніх осіб у виборчий процес.

● Розробити протокол онлайн виборів на основі досліджених який відповідатиме поставленим вимогам.
● Розробити інтерфейс користувача та ЦВК

Можливі сфери застосування. Даний застосунок може стати основою для розробки мобільного додатку для ОСС ЗВО, який забезпечить можливість волевиявлення студентів у будь яких питаннях незалежно від ситуації в країні та світі і також пришвидшить бюрократичні процеси всередині ОСС та збільшить прозорість.

Об’єкт методи та засоби розроблення. Об’єктом розробки є веб-застосунок для проведення онлайн-виборів. Для виконання поставленої задачі фреймворк Django для серверної частини та для Інтерфейсу ЦВК. Бібліотека React для інтерфейсу користувача.

ОГЛЯД ЛІТЕРАТУРИ ТА ЗАСОБІВ РОЗРОБКИ

Огляд фреймворку Django

Django є одним з найпопулярніших веб-фреймворків для розробки веб-додатків на основі мови програмування Python. Його основна мета полягає у спрощенні процесу створення складних веб-додатків, шляхом надання широкого спектру готових інструментів та розширюваної архітектури. Django розроблений на основі принципів моделі-подання-контролер (MVC) та моделі-подання-контролер-шаблон (MTV), що дозволяє розподіляти відповідальність у розробці веб-додатків.

Основні компоненти Django включають:

1. Моделі (Models)

Моделі Django використовуються для визначення структури бази даних. Це означає визначення таблиць, полів та взаємозв'язків між ними. Цей фреймворк також надає ORM (об'єктно-реляційне відображення), яке дозволяє здійснювати взаємодію з базою даних, використовуючи об'єктно-орієнтований підхід, замість написання складних SQL-запитів.

2. Подання (Views)

Подання в Django відповідають за обробку HTTP-запитів та визначення, які дані передавати в шаблони для відображення. Вони взаємодіють з моделями та шаблонами, виконуючи бізнес-логіку та здійснюючи різні операції.

3. Контролери (Controllers)

Контролери в Django відповідають за маршрутизацію URL-адрес і визначення, які подання обробляють певні HTTP-запити. Вони встановлюють зв'язок між URL-адресами та функціями-поданнями.

4. Шаблони (Templates)

Шаблони використовуються для відображення даних на стороні клієнта. Вони містять HTML-код разом з шаблонними тегами та фільтрами Django, які дозволяють динамічно відображати дані, виконувати цикли, умовні оператори та інші операції.

5. URL-маршрутизація

Django має простий та потужний механізм маршрутизації URL-адрес. Ви можете визначати шаблони URL-адрес, які відповідають конкретним маршрутам та вказувати, яке подання повинне обробляти цей маршрут.

6. Адміністративний інтерфейс

Django постачається з вбудованим адміністративним інтерфейсом, який автоматично генерується на основі ваших моделей. Ви можете використовувати його для додавання, редагування та видалення даних з вашої бази даних без необхідності писати додатковий код.

7. Статичні файли

Django дозволяє легко керувати статичними файлами, такими як зображення, CSS-стилі та JavaScript. Ви можете вказати директорію для зберігання статичних файлів та посилатися на них у своїх шаблонах.

8. Розширення функціональності

Django має велику кількість розширень та активну спільноту розробників, які надають підтримку та створюють нові розширення. Ви можете використовувати ці розширення для додавання нових функціональних можливостей до вашого веб-додатку, таких як автентифікація користувачів, робота з API, розсилки електронної пошти.

Огляд бібліотеки React

React є однією з найпопулярніших бібліотек для розробки інтерфейсів користувача (UI) веб-додатків. Розроблена компанією Facebook, вона надає потужні інструменти для побудови ефективних та масштабованих інтерфейсів. React використовує концепцію компонентів, що дозволяє розбити складні UI на невеликі, перевикористовувані блоки, спрощуючи розробку та підтримку додатків.

Основні особливості React:

1. Компонентна архітектура:

React базується на концепції компонентів, що дозволяє розбити користувацький інтерфейс на незалежні, перевикористовувані блоки. Компоненти можуть містити в собі як HTML-код, так і логіку, що дозволяє створювати динамічні та інтерактивні інтерфейси.

2. Віртуальний DOM:

React використовує віртуальний DOM, який є абстракцією над реальним DOM. Він дозволяє React оптимізувати процес оновлення інтерфейсу шляхом ефективного розподілу змін на стороні віртуального DOM та мінімізації фактичних оновлень реального DOM.

3. Одностороннє зв'язування даних:

React використовує одностороннє зв'язування даних, що означає, що дані потокуються вниз по дереву компонентів. Коли дані змінюються, React автоматично оновлює відповідні частини інтерфейсу, що спрощує синхронізацію стану даних зі зображенням на екрані.

4. JSX:

JSX є розширенням синтаксису JavaScript, який дозволяє вбудовувати HTML-подібний код безпосередньо в JavaScript. Це дозволяє писати компоненти React

більш зрозуміло та зручно, забезпечуючи можливість створення динамічного контенту.

5. Універсальна розробка:

React може бути використаний як на стороні клієнта, так і на стороні сервера. Це дозволяє створювати універсальні додатки, які працюють як на сервері, так і в браузері, що забезпечує поліпшену швидкість завантаження та індексацію пошуковими системами.

SQLite

SQLite - це легкий вбудовуваний реляційний база даних, який пропонує повний набір функцій із масово використовуваними системами управління базами даних (СУБД), але не вимагає окремого сервера для роботи. Він є одним з найпоширеніших та найпопулярніших варіантів бази даних, особливо в областях, де пріоритетом є невеликі розміри, простота використання та низькі вимоги до ресурсів.

Особливості SQLite включають:

1. Легкість використання: SQLite простий у встановленні та використанні. Він не потребує окремого сервера, і база даних зберігається в одному файлі, що робить його зручним для вбудовування в різні додатки та пристрої з обмеженими ресурсами.

2. Компактність: Файли бази даних SQLite зазвичай дуже компактні, що забезпечує ефективне використання дискового простору. Це особливо корисно для мобільних пристроїв та додатків, які мають обмежені ресурси.

3. Кросплатформеність: SQLite підтримується на багатьох операційних системах, включаючи Windows, macOS, Linux та інші. Це дозволяє розробникам створювати додатки, які працюють на різних платформах без необхідності переписувати код.

4. Підтримка стандартних SQL-запитів: SQLite підтримує велику частину стандартного мови запитів SQL, що дозволяє використовувати звичні SQL-операції для створення, зчитування, оновлення та видалення даних.

5. Транзакційна безпека: SQLite має вбудовану підтримку транзакцій, що дозволяє виконувати групу операцій як одну атомарну одиницю, забезпечуючи цілісність даних та відновлення у випадку збою.

6. Підтримка широкого спектру додаткових можливостей: SQLite надає розширення для виконання складних операцій, таких як шифрування бази даних, виконання SQL-функцій та тригерів, використання повнотекстового пошуку тощо.

Цифровий підпис

Цифровий підпис - це криптографічний механізм, який використовується для підтвердження автентичності та цілісності цифрової інформації. Він дозволяє особі, яка підписується, підтвердити, що повідомлення або документ належить їй і не було змінено після підписання.

Основний алгоритм роботи цифрового підпису включає наступні кроки:

1. Ключі: Особа, яка підписується, генерує два криптографічних ключі - приватний ключ і публічний ключ. Приватний ключ слід зберігати в секреті, тоді як публічний ключ може бути розповсюджений

2. Хешування: Повідомлення або документ, який потрібно підписати, піддається хешуванню. Хеш-функція перетворює цей вхідний текст на унікальний хеш-код фіксованої довжини.

3. Криптографічний підпис: Приватний ключ використовується для зашифрування хеш-коду повідомлення або документу. Це створює криптографічний підпис, який є унікальним для даного повідомлення та приватного ключа.

4. Передача: Підписане повідомлення або документ разом з публічним ключем може бути передане іншим особам.

5. Перевірка підпису: Особа, яка отримала підписаний документ, витягує публічний ключ підписуючої сторони. Вона використовує публічний ключ для розшифрування криптографічного підпису і отримує хеш-код повідомлення або документу. Потім вона обчислює хеш-код отриманого повідомлення або документу і порівнює його з отриманим

6. Порівняння хеш-кодів: Особа, яка перевіряє підпис, порівнює хеш-код, отриманий з розшифрованого підпису, з хеш-кодом, обчисленим з отриманого повідомлення або документу. Якщо хеш-коди співпадають, це означає, що повідомлення або документ не змінювалися після підписання і що підпис є дійсним.

Цифровий підпис забезпечує такі важливі властивості:

1. Автентичність: Цифровий підпис дозволяє перевірити автентичність підписувача. Оскільки приватний ключ відомий лише підписувачу, отримання вірного підпису підтверджує, що повідомлення було підписане власником ключа.

2. Цілісність: Перевірка хеш-коду дозволяє виявити навіть незначні зміни в підписаному повідомленні або документі. Якщо хеш-коди не співпадають, це свідчить про зміни в документі після підписання.

3. Незаперечність: Оскільки кожен підпис має унікальний приватний ключ, підписувач не може заперечувати факт підпису. Це може бути важливо в правових або комерційних ситуаціях.

Алгоритм RSA

RSA (Rivest-Shamir-Adleman) - це асиметричний криптографічний алгоритм, який використовується для шифрування, розшифрування та цифрового підпису. Алгоритм базується на математичних властивостях факторизації цілих чисел.

Основний алгоритм RSA включає наступні кроки:

1. Генерація ключів:
   - Вибір двох великих простих чисел, скажімо p і q.
   - Обчислення їх добутку n = p * q, який називається модулем RSA.
   - Обчислення значення функції Ойлера (euler's totient function) φ(n) = =(p-1) * (q-1), яка представляє кількість чисел, менших за n, і взаємно простих з n.
   - Вибір цілого числа e, такого що 1 < e < φ(n) та є взаємно простим з φ(n). Це значення e використовується як публічний ключ.

2. Шифрування:
   - Повідомлення або дані, які потрібно зашифрувати, представляються у вигляді цілого числа m, такого що 0 ≤ m < n.
   - Зашифрування повідомлення за допомогою публічного ключа (n, e) з використанням наступної формули: c = m^e mod n, де c - зашифроване повідомлення.
3. Розшифрування:
   - Зашифроване повідомлення c отримується.
   - Використовуючи приватний ключ (d), розшифроване повідомлення m обчислюється за допомогою формули: m = c^d mod n.
4. Генерація цифрового підпису:

- Підписувач обчислює хеш-код повідомлення або документу, який потрібно підписати, за допомогою хеш-функції.

- Хеш-код піддається шифруванню за допомогою приватного ключа (d), щоб отримати криптографічний

5. Підпис перевірки:

- Отримане зашифроване повідомлення підписується приватним ключем (d) за допомогою формули: s = c^d mod n, де s - цифровий підпис.

- Цифровий підпис s додається до повідомлення або документу.

6. Перевірка підпису:

- Особа, яка перевіряє підпис, отримує підписане повідомлення та відкритий ключ (n, e).

- Розшифровує цифровий підпис s за допомогою відкритого ключа (n, e): c = s^e mod n. - Отриманий результат c порівнюється з отриманим раніше зашифрованим повідомленням. Якщо вони співпадають, це означає, що підпис є дійсним і повідомлення не змінювалося після підписання.

Цей алгоритм базується на складності факторизації великих чисел. Зламати RSA, знаходячи прості числа p і q, потрібно обчислювально надзвичайно витратними методами факторизації. Тому RSA є одним з найбільш використовуваних алгоритмів для забезпечення безпеки у криптографії.

Важливо зазначити, що безпека RSA вимагає достатньої довжини ключа. Зазвичай використовуються ключі довжиною 2048 біт або більше для надійного захисту від атак. Крім того, RSA також може бути використаний для обміну ключами, як частина протоколу, такого як TLS (Transport Layer Security) або SSH (Secure Shell).

Алгоритм AES

AES (Advanced Encryption Standard) є симетричним алгоритмом шифрування, який використовується для захисту конфіденційності даних. Він став стандартом

шифрування, прийнятим Національним Інститутом Стандартів і Технологій (NIST) у 2001 році. AES замінив попередні стандарти шифрування, такі як DES (Data Encryption Standard).

Основний алгоритм AES використовується для шифрування та розшифрування блоків фіксованого розміру (128 бітів) даних. Алгоритм має декілька основних етапів:

1. Ключова розширення:

Початковий ключ, який вводиться користувачем, розширюється до серії підключів. Ключове розширення використовує певні математичні операції, такі як підстановки та циклічні зсуви, для генерації підключів, які використовуються на кожному етапі шифрування.

2. Початкове додавання ключа:

Кожен блок даних розділяється на 16 байтів (128 бітів) та початково додається до блоку даних ключ шифрування.

3. Рундові трансформації:

AES складається з декількох рундів, залежно від розміру ключа. Кожен раунд включає в себе кроки, такі як підстановка байтів, зсуви рядків, змішування стовпців та додавання ключа.

- Підстановка байтів (SubBytes): Кожен байт в блоку замінюється на відповідний байт зі заздалегідь визначеної таблиці заміни (S-Box).
- Зсув рядків (ShiftRows): Байти в кожному рядку блоку зсуваються циклічно вліво на певну кількість позицій.
- Змішування стовпців (MixColumns):

Кожен стовпець блоку множиться на певну матрицю, що змінює його значення.

- Додавання ключа (AddRoundKey): Кожен байт в блоку XOR-ується з відповідним байтом з підключа.

4. Фінальний раунд:

Останній раунд виконується без кроку змішування стовпців.

Після закінчення всіх раундів, шифрування або розшифрування завершується, і результатом є зашифрований або розшифрований блок даних.

AES може використовувати різні розміри ключів (128, 192 або 256 бітів), із збільшенням розміру ключа збільшується його стійкість до зламу. Більшість реалізацій AES оптимізовані для виконання на апаратному рівні, що робить його ефективним і швидким для шифрування великих обсягів даних.

Вибори в ОСС КНУ

В ОСС КНУ є різні виборчі процеси (обрання студентського омбудсмана, голови СП структурного підрозділу, голови СП Університету і т.п).

У виборах голови СП структурного підрозділу беруть участь тільки студенти даного структурного підрозділу, на відміну від виборів голови СП Університету.

Процес голосування відбувається наступним чином: виборець приходить на виборчу дільницю, де показує студентський квиток, котрий перевіряє ВКС. Після цього студенту видають бюлетень, на який вносять ПІБ виборця і номер студентського квитка, відривають корінець бюлетня, залишаючи виборцю тільки частину з ПІБ кандидатів та місця для волевиявлення виборця. Виборець далі йде до кабінки для голосування, після чого кидає бюлетень у корзину, з якої після закриття дільниці ЦВКс дістає бюлетені та веде підрахунок голосів.

– Якщо виборець прийшов після закриття дільниці або вкидає бюлетень після закриття дільниці, його голос не буде врахованим.

– Якщо виборець вийшов за межі дільниці з бюлетнем, його голос також є втраченим.

Всі етапи очного виборчого процесу відтворені та/або адаптовані для онлайн-виборів.

Недоліки прямих таємних виборів

Так як забезпечення чесності виборчого процесу лежить на людях прямі таємні вибори можна сфальсифікувати і такі прецеденти вже не раз були у світовій історії, коли завдяки домовленостям певної кількості людей виборчий процес був сфальсифікований. Також у випадку пандемії, військових дій і т.п виборчий процес унеможливлюється, що ставить під питання демократичні засади держави.

Вимоги для системи онлайн-виборів

1. Анонімність

Як і звичайні вибори мають бути анонімними, система онлайн-виборів також має гарантувати анонімність виборцям, задля того щоб на їх вибір не могли впливати зацікавлені в цьому сторони.

2. Відсутність можливості фальсифікації голосів.

Задля того, щоб обраний очільник був легітимним, система має забезпечити унеможливлення фальсифікації голосів.

3. Збереження особистих даних від доступу сторонніх осіб.

Так як для авторизації виборців використовуються їх особисті дані такі як ПІБ, місце навчання, освітня програму тощо, потрібно унеможливити можливість витоку даних.

4. Забезпечення безпеки виборчого процесу від втручання сторонніх осіб.

Вибори це комплексний процес і задля того щоб зацікавленні у тому щоб процес не відбувся не мали змогу його зірвати необхідні механізми які дадуть змогу виявити порушників.

Протоколи онлайн голосування

Спрощений протокол онлайн голосування

Розглянемо спрощені протоколи голосування та пов'язані з цим проблеми. Найпростіший про-токол голосування виглядатиме таким чином:

1) кожний виборець шифрує свій бюлетень відкритим ключем Центральної виборчої комісії (ЦВК);

2) виборець відправляє свій бюлетень до ЦВК;

3) ЦВК розшифровує бюлетені, підводить підсумки і опубліковує результати голосу-вання.

Цей протокол має недоліки. ЦВК не може дізнатися, звідки отримано бюлетені, і навіть чи належать надіслані бюлетені виборцям. У неї немає можливості визначити, чи не голо-сували виборці більше одного разу. Позитив-ною стороною є неможливість змінити бюле-тень іншої людини, але ніхто і не намагати-меться це зробити, тому що набагато простіше голосувати повторно, добиваючись потрібних результатів виборів.

Розширений спрощений протокол

1) виборець підписує свій бюлетень власним закритим ключем;

2) виборець шифрує свій бюлетень відкри-тим ключем ЦВК;

3) виборець посилає свій бюлетень до ЦВК;

4) ЦВК розшифровує бюлетені, перевіряє підписи, підводить підсумки і опубліковує результати голосування.

Представлений протокол вирішує проблему попереднього. Кожен бюлетень підписаний закритим клю-чем виборця, тому ЦВК знає, хто голосував, а хто ні, і як голосував кожен виборець. Якщо отрима-ний бюлетень не підписано законним користува-чем, або бюлетень підписано виборцем, який вже проголосував, то такий бюлетень ігнорується ко-місією. Крім того, через цифровий підпис ніхто не може змінити бюлетень іншого виборця, на-віть якщо вдасться перехопити

його на етапі (2). Проте даний протокол не задовольняє всі вимоги описані вище, а саме анонімність голосу виборця так як за підписом ЦВКс може ідентифікувати хто як проголосував.

Протокол Нурмі — Салома — Сантіна

V - валідатор

B - виборець

A - ЦВК

M - мітка виборця

K - ключ

1. Валідатор (далі V) відправляє секретні розпізнавальні мітки (Далі M) всім B до голосування.
2. V відправляє весь A набір M, але без інформації про те, кому вони належать.
3. B створює свої ключі КВзак - закритий , КВвід - відкритий та викладає у загальний доступ КВвід, а також створює секретний ключ (КВсек - секретний ключ), який потрібен, щоб ніхто не дізнався вміст бюлетеня до потрібного моменту.
4. B формує повідомлення C, де висловлює свій вибір, підписує КВзак Прикладає до нього отриману M і шифрує КВсек
5. До зашифрованого тексту прикладає M і відправляє A.
6. A отримує зашифрований текст, M визначає, що він прийшов від B, але не знає від кого саме і як B проголосував, згодом публікує його.
7. Опублікований зашифрований текст є інформацією, що B відправив КВсек.
8. A збирає ключі, розшифровує текст, підраховує голоси та приєднує до опублікованого зашифрованого тексту C без M.

На 6 кроці A не зможе заперечувати, що не отримував повідомлення від B. Завдяки публікації зашифрованого тексту та бюлетеня, кожен може перевірити, що його голос був врахований належним чином. Даний протокол має мінуси,

якщо A вступить у таємну змову з V, то він зможе маніпулювати голосуванням, не приймаючи повідомлення від деяких B. Також є проблема "мертвих душ", якщо V спеціально додасть неіснуючих, то A зможе фальсифікувати бюллетени від них.

Протокол Фудзіока-Окамото-Охта

Робота протоколу полягає в заздалегідь обраному способі маскуючого шифрування – це особливий вид шифрування, який дозволяє переконатися, що документ є справжнім і був підписаний авторизованим користувачем, але не дає інформації про дані, що містяться. Алгоритм виглядає так:

1. V затверджує перелік B.
2. B створює свої ключі КВзак, КВвід, КВсек і викладає в загальний доступ КВвід
3. Формує повідомлення C, де висловлює свій вибір, шифрує його КВсек, маскує, підписує КВзак та відправляє V.
4. V створює свої ключі К□зак, К□від та викладає в загальний доступ К□від
5. V засвідчується, що належить B, який ще голосував, підписує його К□зак та відправляє B.
6. Видаляє шар маскуючого шифрування і відправляє повідомлення C до A.
7. A перевіряє підписи B і V і поміщає зашифрований C спеціальний список, який буде опублікований після голосування.
8. Після публікації списку B відправляє A свій КВсек.
9. A збирає ключі, розшифровує і підраховує голоси, при цьому публікує декодуючі ключі разом із зашифрованими C, щоб B змогли самостійно перевірити результати голосування

Протокол Sensus

Лоррі Кранор і Рон Сітрон (Lorrie Faith Cranor, Ron K. Cytron) в 1996 запропонували модифікацію протоколу Фудзіока-Окамото-Охта під назвою Sensus. Відмінність полягає в кроках 5-6. Після того, як **A** отримало зашифроване

повідомлення від **V**, воно не тільки додає його в список, що публікується, а додатково надсилає підписаний бюлетень назад виборцю в якості квитанції. Таким чином **B** не потрібно чекати, поки проголосують всі інші, і він може закінчити голосування за одну сесію. Це не тільки зручно для кінцевого користувача, але ще і надає додатковий доказ, що **B** брав участь у виборах. Крім того, в Sensus регламентовані додаткові допоміжні модулі, які спрощують і автоматизують хід голосування.

Особливості, переваги та недоліки

Тепер навіть якщо агентствам вдасться змовитися, **A** не зможе впізнати виборців до того, як отримає ключ. Хоча воно все ще має можливість не приймати повідомлення, відпадає можливість ігнорувати повідомлення конкретно від «небажаних» виборців. Залишається лише проблема подачі голосів виборців, які не прийшли на вибори.

Протокол HE-SU

1. V затверджує список, створює свої ключі К□зак, К□від і викладає у загальний доступ К□від

2. У створює свої ключі КВзак, КВвід, генерує випадкове число R обчислює хеш-функцію h від КВвід, маскує її R і відправляє V отриману функцію, яка виглядає так: □ = К□від (□) · *h*(КВвід).

3. V засвідчується у праві голосувати, підписує отримане повідомлення, тобто. К□зак (К□від(□) · *h*(КВвід)) = □ · К□зак (*h*(КВвід)) та відправляє В.

4. Видаляє шар маскуючого шифрування, перевіряє справжність

підписи V, т. Е. К□від (К□зак (*h*(КВвід))) = *h*(КВвід) і відправляє А КВвід та підпис V, тобто. К□зак (*h*(КВвід)).

5. А перевіряє справжність підпису V, перевіряє збіг хешфункції від КВвід з тим, що зберігається у підписі V, додає КВвід до списку авторизованих ключів та повідомляє про це Ст.

6. У формує повідомлення С, де висловлює свій вибір, шифрує його створеним КВсек і відправляє А набір що складається з КВвід, зашифрованого КВсек повідомлення С та зашифровану КВзак хеш-функцію від зашифрованого КВсек повідомлення С.

7. А перевіряє КВвід зі списком, створеним раніше, порівнює хешфункцію повідомлення С зашифрованого КВсек і хеш-функцію, отриману за допомогою КВзак та публікує весь набір у відкритому списку.

8. Після публікації списку відправляє А новий набір що складається з КВвід, КВсек та зашифровану КВзак хеш-функцію від КВсек.

9. А перевіряє справжність КВсек, порівнюючи хеш-функцію від КВсек і хеш-функцію отриману за допомогою КВзак Якщо все правильно, то розшифровує отриману раніше С, публікує всі дані і підраховує голоси.

10. Після голосування V публікує затверджений список, а А – список авторизованих ключів.

В представленому протоклі ЦВК та Валідатор не можуть змовитись оскільки публікуються списки відповідно неможливо створити неіснуючих виборців та проголосувати за тих хто не взяв участь. Головною проблемою є вразливість до DoS - атак оскільки потребує багато ресурсів для підтримки працездатності протоколу через його складність.

Підсумовуючи огляд протоколів представлених вище можна охарактеризувати наступні проблеми:

1. Складність реалізації та підтримки

2. Можливість ЦВК та Валідатору домовитись та підробити голоси чи використати голоси тих хто не взяв участь в голосуванні.
3. Можливість ЦВК не зарахувати голос виборця чи недопустити його до виборчого процесу.

У висновку потрібно створити новий протокол який не буде потребувати великої кількості ресурсу, надасть прозорості в діях ЦВК та Валідатора

ТЕОРЕТИЧНА ЧАСТИНА

Протокол проведення онлайн виборів

1. ЦВК реєструє виборців з їх унікальним паролем які виборці надають ЦВК для реєстрації. Виборці реєструються на Валідаторі.
2. Валідатор кожному виборцю надає унікальний ідентифікатор від Валіадатора який відомо тільки виборцю.
3. Виборець через валідатор надсилає ЦВК свій унікальний пароль за яким ЦВК перевіряє чи брав виборець участь в голосуванні, якщо ні, то надсилає бюлетень разом з унікальним ідентифікатором від ЦВК згенерованим при запиті виборця.
4. Виборець зробивши свій голос шифрує унікальний ідентифікатор наданий ЦВК, шифрує унікальний ідентифікатор разом з голосом, та надсилає це все разом з унікальним ідентифікатором від ЦВК
5. Валідатор надає виборцю його шифр та ключ до нього.
6. ЦВК зберігає шифр виборця.
7. ЦВК надає виборцю підтвердження що його голос зарахований разом з його зашифрованим ідентифікатором.
8. По завершенню виборів, валідатор надсилає ЦВК ключ для дешифрування голосів виборців, ЦВК їх розшифровує та виставляє список шифрів і як вони проголосували. Валідатор виставляє список які унікальні ідентифікатори від Валідатора взяли участь у виборах.

Таким чином у випадку якщо ЦВК та Валідатор домовляться та проголосують за виборців які не брали участь в голосуванні виборець зможе перевірити чи його голос не був використаний і відповідно довести це тим що не матиме підтвердження від валідатора та ЦВК. Перевіривши опубліковані списки виборець може бути впевненим у тому що його голос зарахований правильно у випадку шахраювання ЦВК він зможе апелювати це маючи шифр за яким можна буде перевірити чи голос врахований правильно. Завдяки видачі унікального ідентифікатору від ЦВК, ЦВК не зможе визначати як голосував кожен виборець, відповідно анонімність виборця збережена. У випадку якщо стороння людина

втрутиться в базу даних ЦВК, вона не зможе завчасно дізнатись скільки голосів за того чи іншого кандидата оскільки можливість для дешифрування голосів з’являється тільки по завершенню виборів.

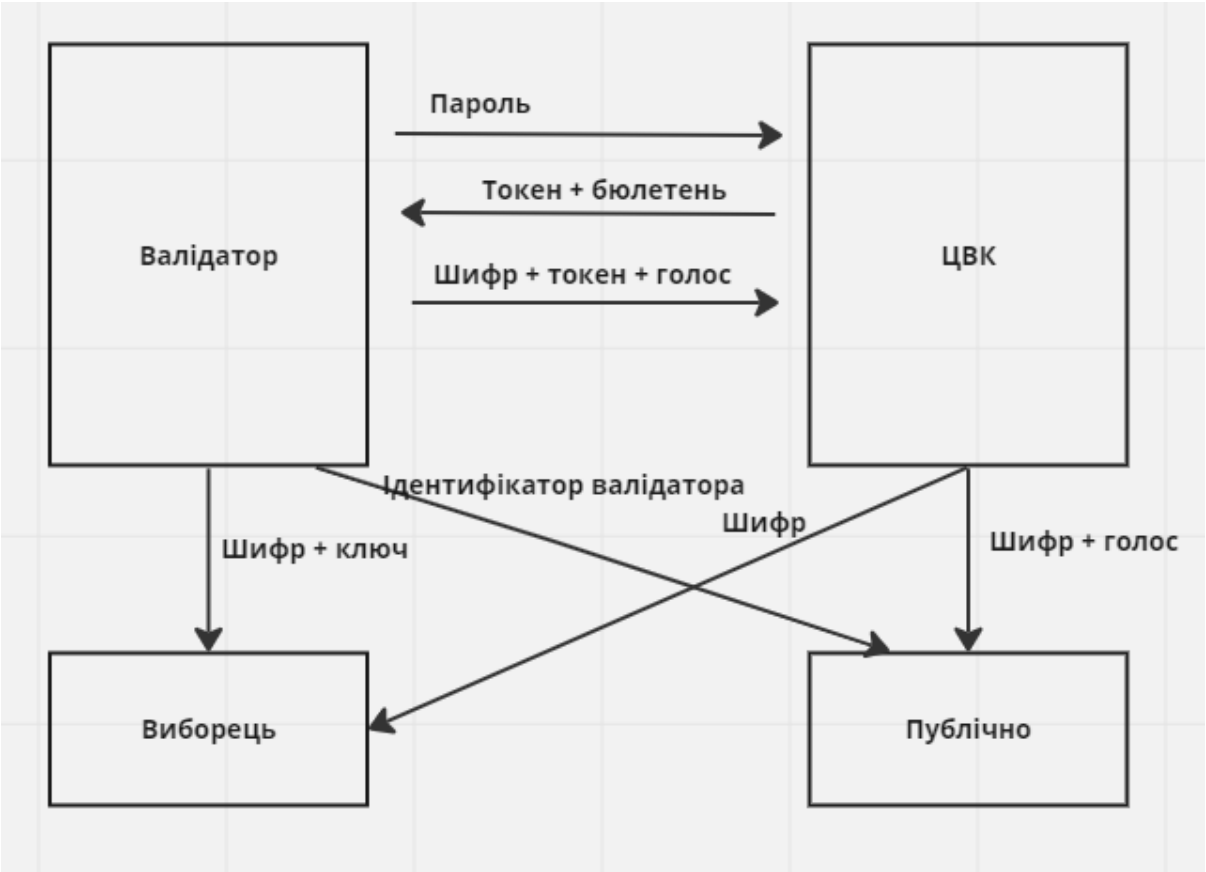

Рис 1. Схема роботи протоколу

ПРАКТИЧНА ЧАСТИНА

В даній частині буде описана реалізація основних етапів протоколу описаного в теоретичній частині.

Реалізація протоколу

Підтвердження права голосу Виборцем

Алгоритм підтвердження права голосу виборця:

1. Виборець надсилає свої дані Валідатору

```
const data = {
  name: name.current.value,
  password: password.current.value,
}
await axios.put('http://127.0.0.1:9583/check/', { name: name.current.value })
```

2. Валідатор перевіряє чи брав виборець участь в голосуванні. Та дає користувачу відповідь, відмітивши в базі даних що користувач взяв участь в голосуванні.

```
def check(request: Request):
    req = request.data
    user = codeVote.objects.get(name=req['name'])
    if user.is_voted == 1:
      return JsonResponse({'value': 'you voted'})
    user.is_voted == 1
    user.save()
    return JsonResponse({'value': 'approved'})
```

3. Після отримання підтвердження від валідатора Дані та пароль виборця підписуються цифровим підписом надсилаються ЦВК.

```
 if (response.data.value == 'aproved') {
        let sign = digit_sign(data)

        const dataToSend = {
          data: sign.data,
```

```
    signature: sign.signature,
    publicKey: sign.publicKeyPem,
   };
   axios.post('http://127.0.0.1:8000/api/send_bulletin', dataToSend)
```

4. ЦВК перевіряє чи брав виборець участь в голосуванні, у випадку успішної перевірки повертає бюлетень та унікальний ідентифікатор сформований ЦВК підписані цифровим підписом, зберігаючи унікальний ідентифікатор та відмічаючи що користувач взяв участь у виборах.

```
def send_bulletin(request: Request):
    reqdata = request.data
    try:
        check_sign(reqdata)
    except Exception as e:
        return JsonResponse({'valid': False, 'error': str(e)})
    req = decodeData(reqdata['data'])
    user = Users.objects.get(name=req['name'])
    vote = Voting.objects.get(faculty=1)
    if not check_time(vote.start, vote.finish):
        return JsonResponse({'value': 'timeoff'})
    if user.uniq == req['password'] and not user.is_voted:
        user.is_voted = 1
        user.save()
        number = generate_unique_string()
        uniqField(number=number).save()
        candidates = list(Candidates.objects.filter(
            faculty=user.faculty).values('name', 'id'))
        vote = list(Voting.objects.filter(
            faculty=user.faculty).values('name', 'faculty', 'id'))
        data_to_sign = {
            'name': req['name'],
```

```
        'candidates': candidates,
        'vote': vote,
        'token': number,
    }
    signat = sign(data_to_sign)
    return JsonResponse({
        'data': data_to_sign,
        'signature': signat['signature'],
        'public_key': signat['public_key']
    })
else:
    return JsonResponse({'valid': 'you voted'})
```

Отримання бюлетеня

1. Користувач отримує бюлетень від ЦВК та надсилає його Валідатору для перевірки цифрового підпису.
2. Валідатор перевіряє підпис та надсилає підтвердження користувачу разом з бюлетенем та кодом для шифрування голосу.

```
axios.put('http://127.0.0.1:9583/counter/', response.data).then(response => {
            setStatus('можете голосувати')
            console.log(response)
            setVoteState(response.data)
            console.log(voteState);
          })
```

3. Бюлетень відображається користувачу

```
return (
    <div style={keyStyle}>
      <p>{voteState.data.vote[0].name}</p>

      <form action="">
```

```
        {Candidate}
        <button onClick={sendVote}>Send Vote</button>
      </form>
      <p>Selected candidate: {choose}</p>
      <p >Статус голосування: {statusv}</p>
      <p >uniq: {uniqc}</p>
      <p >uniq response: {uniqr}</p>
      <p>key: {keyc}</p>
    </div>
  )
```

Надсилання та підтвердження голосу

1. Користувач надсилає свій голос.

2. React створює шифр алгоритмом AES з унікального ідентифікатора від ЦВК і шифрує голос разом з шифром ключем який отримав від Валідатора разом з бюлетнем. Підписує це цифровим підписом та надсилає до ЦВК, разом з унікальним ідентифікатором від ЦВК

```
let sendVote = async (event) => {
  event.preventDefault();
  let code = (voteState.data.token + choose).toString();
  let key = generateRandomString(16)
  let uniq = encryptAES(code, key)
  let ccode = encryptAES(uniq + ' ' + choose, voteState.data.key)
  const data = {
    voting: voteState.data.vote[0].id,
    token: voteState.data.token,
    ccode: ccode,
    uniq: uniq,
    vote: 'yes',
  }
  setUniq(uniq)
```

```
setKey(key)
let sign = digit_sign(data)
const dataToSend = {
  data: sign.encodedData,
  signature: sign.signature,
  publicKey: sign.publicKey,
  codeVote: uniq,
};
await axios.post('http://127.0.0.1:8000/api/vote', dataToSend).then(response => {
  setUnir(response.data.code);
  setStatusv(response.data.status);
})
```

3. ЦВК перевіряє цифровий підпис, перевіряє чи брав унікальний ідентифікатор користувача виданий ЦВК участь у виборах. Якщо ні то зміню запис в базі даних про цей токен та зберігає шифр користувача та зашифрований голос.

```
reqdata = request.data
    req = decodeData(reqdata['data'])
    uniq = req['uniq']
    vote = req['ccode']
    try:
        check_sign(reqdata)
    except Exception as e:
        return JsonResponse({'valid': False, 'error': str(e)})

    token = uniqField.objects.get(number=req["token"])
    voting = Voting.objects.get(id=req["voting"])
    if not check_time(voting.start, voting.finish):
        return ''
    if token.is_voted:
        return JsonResponse({'status': 'ви вже брали участь в голосуванні'})
    codeVote(code=uniq, vote=vote).save()
```

```
    token.is_voted = 1
    token.save()
return JsonResponse({'valid': True, 'code': uniq, 'status': 'ваш голос зарахований'})
```

4. Користувач отримує шифр, ключ до нього, та повторно шифр який надісланий ЦВК як підтвердження що голос правильно зарахований.

```
await axios.post('http://127.0.0.1:8000/api/vote', dataToSend).then(response => {
     setUnir(response.data.code); - шифр який поверає ЦВК
     setStatusv(response.data.status); - оновлення статусу що голос зарахований
    })
```

Отримання результатів

1. Щоб отримати результати ЦВК робить запит до Валідатора щоб отримати ключ для декодування голосів.

```
results_list = []


    response = requests.get('http://127.0.0.1:9583/get_key')
    data = response.json()  # отримати дані у форматі JSON
    key = data['key']
```

2. Після отримання ключа ЦВК розшифровує голоси виборців, підраховує їх та зберігає розшифровані голоси виборців.

```
candidates = Candidates.objects.filter(faculty=v.faculty).values()
            cand_list = []
            codeList = []
            sum = 0
            votes = codeVote.objects.all().values()
            for v in votes:
                voteCode = decrypt_AES(v['vote'], key).split()
                candidate = voteCode[1] + ' ' + voteCode[2]
                goal = Candidates.objects.get(name=candidate)
                goal.vote_for += 1
                goal.save()
```

```
            code = codeVote.objects.get(code=v['code'])
            code.vote = candidate
            code.save()
            codeList.append({'code': code.code, 'vote': code.vote})
```

3. Формує сторінку результатів для ЦВК разом з шифрами та голосами виборців.

```
 for c in candidates:


            cand_list.append(
                {"candidate_name": c['name'], "resultyes": c['vote_for']})
        results_list.append({"candidates": cand_list})
    context = {
        "result": results_list,
        'code': codeList
    }
    print(context)
    return render(request, 'results.html', context=context)
```

Представлена реалізація протоколу є основою для створення повноцінної системи для онлайн голосування, де буде присутня реєстарція виборців у Валідаторі та ЦВК, резервне збереження даних, покращенний інтерфейс користувача, та підтвердження зарахування голосу від ЦВК через пошту чи інші сервіси які дадуть можливість користувачу зберегти це підтвердження.

Приклад роботи

Надсилання паролю та ПІБ виборця з валідатора до ЦВК

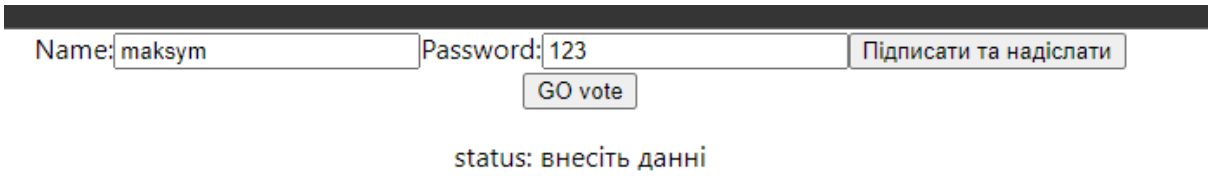

Рис3. Інтерфейс користувача Валідатора

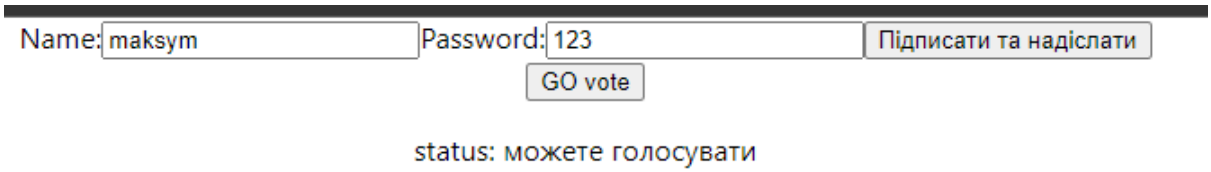

Рис4. Оновлення статусу валідатора

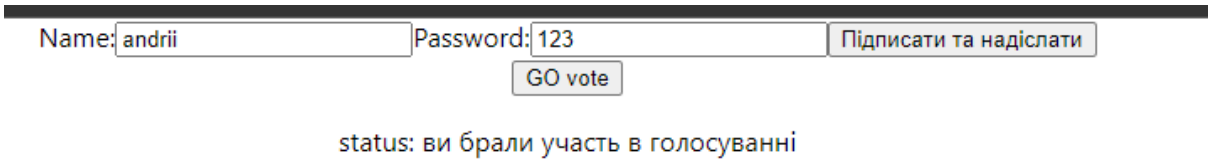

Рис5. Оновлення статусу при повторній спробі голосування

Після підтвердження можливості голосувати переходимо до бюлетеня.

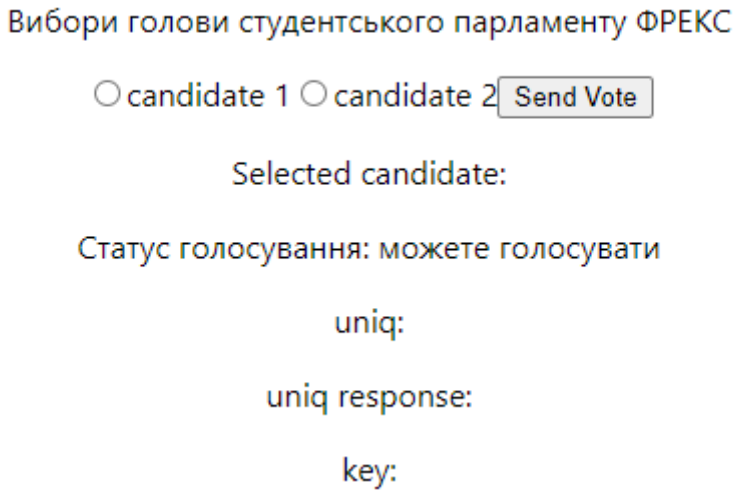

Рис10. Виборчий бюлетень

Підтвердження зарахування голосу від ЦВК

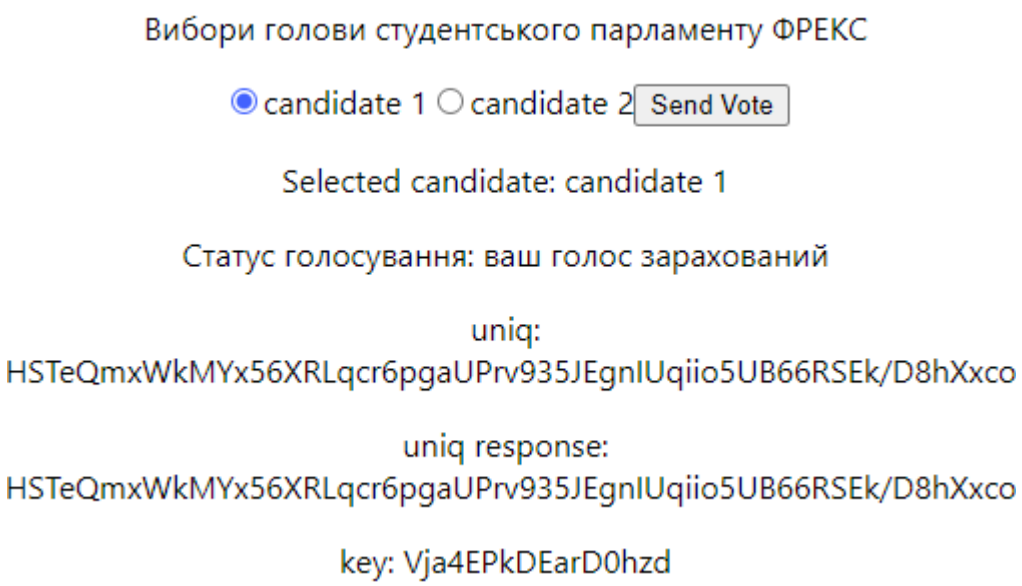

Рис11. Підтвердження зрахування голосу від ЦВК

uniq - шифр який сформував React

uniq response -шифр який повернула ЦВК

key - ключ від шифру

у випадку повторного надсилання голосу виборець отримує сповіщення що він вже брав участь в голосуванні і повторний голос не буде зарахованим.

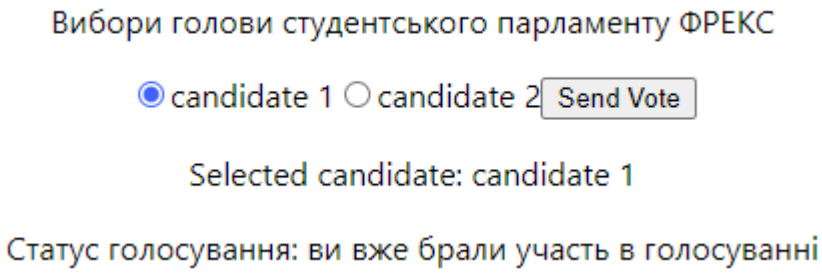

Рис12.Оновлення статусу на бюлетні

ФУНКЦІЇ ДЛЯ ЦВК

Створення голосувань:

- структурний підрозділ, на якому проводиться голосування;
- часовий проміжок, в якому проводиться голосування;
- назва голосування.

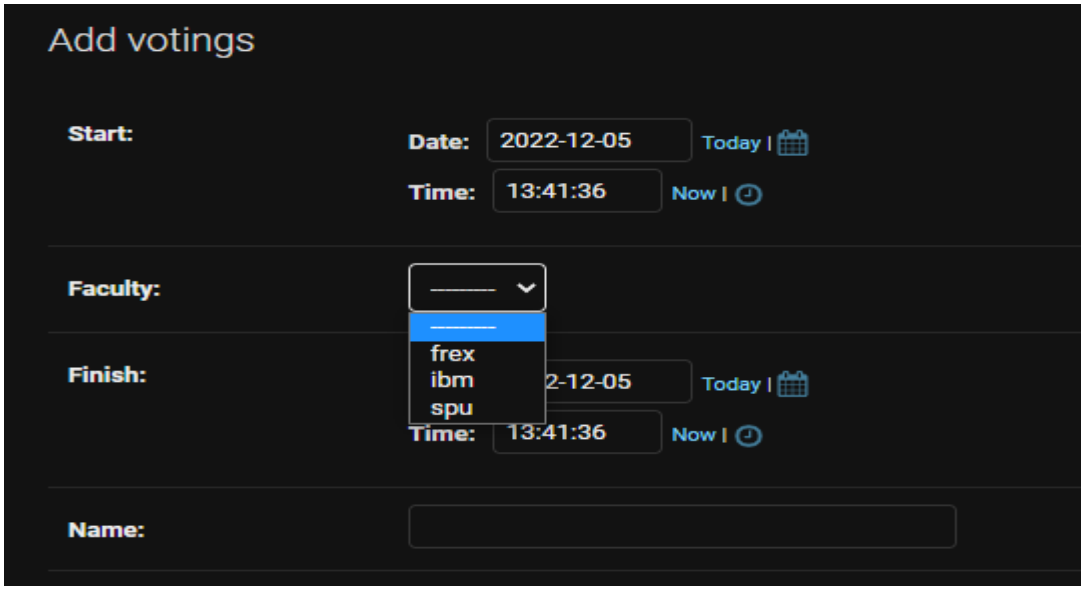

Рис 13. Інтерфейс ЦВК для створення голосування

Для запобігання можливості зірвати вибори членами ЦВК які матимуть доступ до панелі адміністратора під час виборів, можливість редагувати та видаляти вибори в період їх проведення закривається

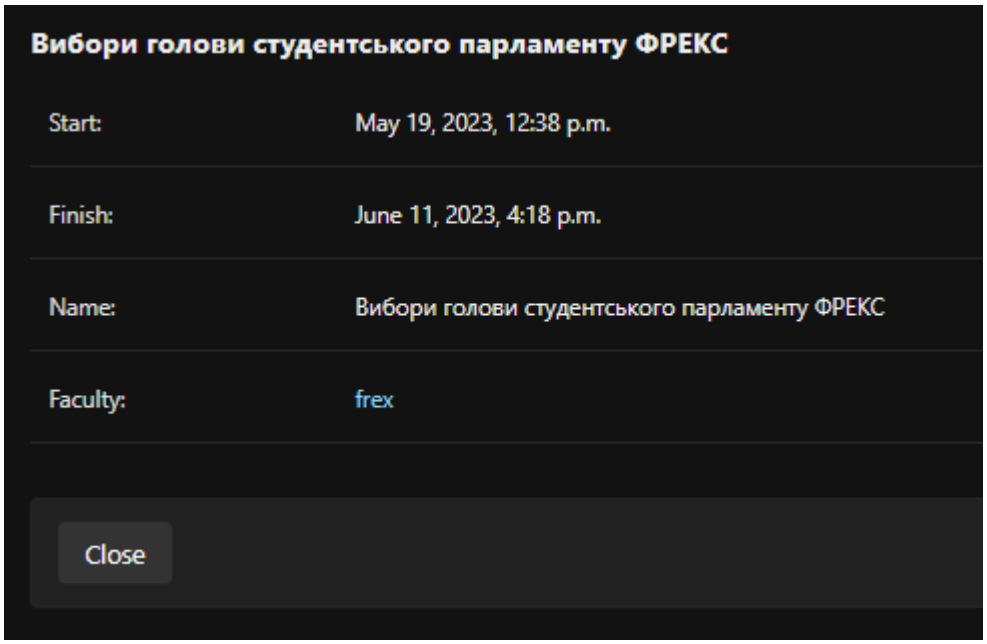

Рис 14. Зміна інтерфейсу ЦВК на час проведення виборів

Реєстрація кандидатів

- ПІБ кандидата
- структурний підрозділ на якому кандидат балотується

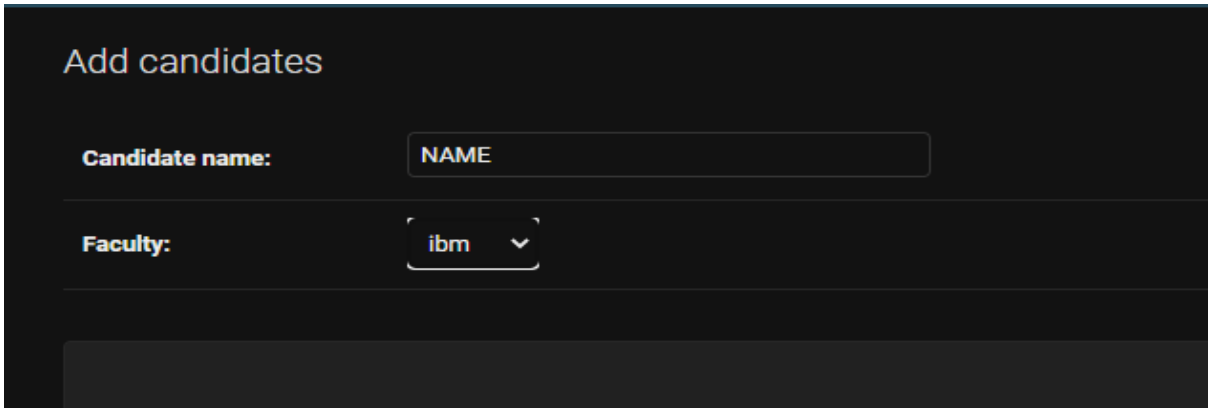

Рис14. Інтерфейс додавання кандидата

Можливість змінювати інформацію про кандидатів чи видаляти їх закривається на період проведення виборів.

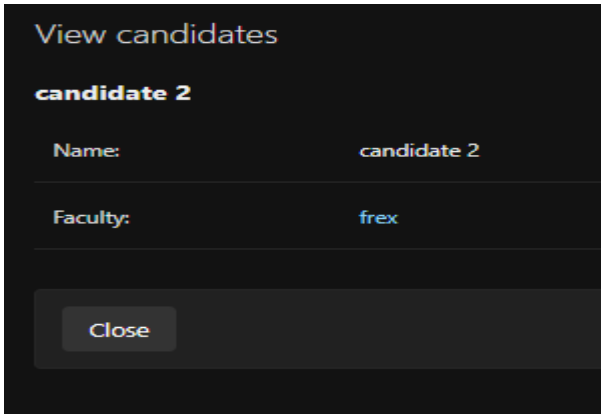

Рис15. Зміна інтерфейсу редагування кандидата

Отримання результатів виборів

Обравши вибори результати яких ЦВК хоче дізнатись, ЦВК отримує ключ для дешифрування від Валідатора, дешифровує голоси та підраховує їх.

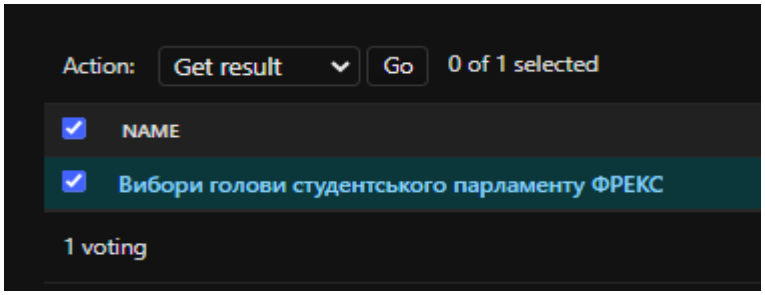

Рис 16. Інтерфейс ЦВК для отримання результатів

Результати

candidate 1

За 1

candidate 2

За 0

foon7PUb3LiqX6IQEMopso1V9J9AphW+7PgyC3Y64SzEbTr3Kplklcxa7nAc74bwLUPeDMdI+

candidate 1

Рис17. Результати виборів разом з шифрами користувачів та їх голосами

ВИСНОВКИ

Аналіз наявних протоколів для онлайн голосування показав, що кожен з них має переваги та недоліки, оскільки головною проблемою є людський фактор та бажання зірвати чи сфальсифікувати вибори, що призвело до створення нового протоколу для онлайн голосування в рамках виборів.

Аналіз наявних алгоритмів шифрування надав можливість реалізувати протокол з достатнім рівнем захисту, при мінімальному використанні ресурсів.

Наявна реалізація протоколу може стати основою для подальшої розробки додатку для виборів, з реєстрацією та аутентифікацією користувачів, та більш гнучкою системою опитувань. Тобто завдяки даній системі можна буде проводити опитування, голосування і подібні форми опитування, що призведе до зростання явки виборців, та більшої прозорості у демократичних процесах.

В подальшій розробці системи потрібно реалізувати покращену верифікацію зарахування голосу від ЦВК, інтерфейс користувача, забезпечити резервне зберігання даних, додати системи аудиту на бази даних, та зробити мобільну версію додатку для більшої доступності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ПРОТОКОЛИ ЕЛЕКТРОННОГО ГОЛОСУВАННЯ/ Анісімова Л.А.б 2013
2. Django documentation [Електронний ресурс]. Режим доступу до ресурсу: https://docs.djangoproject.com/en/4.1/
3. React documentation [Електронний ресурс]. Режим доступу до ресурсу: https://uk.reactjs.org/docs/getting-started.html
4. Python documentation [Електронний ресурс]. Режим доступу до ресурсу: https://www.python.org/doc/
5. Положення про проведення виборів в ОСС КНУ [Електронний ресурс]. Режим доступу до ресурсу: http://sp.knu.ua/wp-content/uploads/2019/04/%D0%9F%D0%BE%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%BD%D1%8F_%D0%BF%D1%80%D0%BE_%D0%B2%D0%B8%D0%B1%D0%BE%D1%80%D0%B8_%D1%81%D1%82%D1%83%D0%B4%D0%B5%D0%BD%D1%82%D1%96%D0%B2_%D0%9A%D0%9D%D0%A3-1.pdfhttp://sp.knu.ua/wp-content/uploads/2019/04/%D0%9F%D0%BE%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%BD%D1%8F_%D0%BF%D1%80%D0%BE_%D0%B2%D0%B8%D0%B1%D0%BE%D1%80%D0%B8_%D1%81%D1%82%D1%83%D0%B4%D0%B5%D0%BD%D1%82%D1%96%D0%B2_%D0%9A%D0%9D%D0%A3-1.pdf
6. A New Practical Secure e-Voting Scheme [Електронний ресурс] // 1997 - Режим доступу до ресурсу : http://www.cs.cmu.edu/~qihe/paper/e_voting/
7. Sensus: A Security-Conscious Electronic Polling System for the Internet [Електронний ресурс] //1997 - Режим доступу до ресурсу: https://dl.acm.org/doi/10.5555/938435.938629

ДОДАТОК А

Шифрування даних алгоритмом AES з використанням бібліотеки CryptoJS мовою Java Script

```
function encryptAES(message, key) {
  // Перетворення ключа на формат, який приймає AES
  const formattedKey = CryptoJS.enc.Utf8.parse(key);

  // Шифрування повідомлення
  const encrypted = CryptoJS.AES.encrypt(message, formattedKey, {
    mode: CryptoJS.mode.ECB, // Режим шифрування
    padding: CryptoJS.pad.Pkcs7, // Схема вирівнювання
  });

  // Повертаємо зашифроване повідомлення у вигляді рядка
  return encrypted.toString();
}
```

ДОДАТОК Б

Підписання даних цифровим підписом мовою Java Script

```
function digit_sign (data) {
   const keyPair = KEYUTIL.generateKeypair('RSA', 2048);
   // Convert data to string
   const dataString = JSON.stringify(data);
   // Encode data to base64
   const encodedData = btoa(dataString);
   // Create a new Signature object with SHA256withRSA algorithm
   const sig = new KJUR.crypto.Signature({ alg: 'SHA256withRSA' });
   // Initialize the signature object with the private key
   sig.init(keyPair.prvKeyObj);
   // Update the signature object with the encoded data
   sig.updateString(encodedData);
   // Generate the signature
   const signature = sig.sign();
   // Public key in PEM format
   const publicKeyPem = KEYUTIL.getPEM(keyPair.pubKeyObj);
   // Data to send in the AJAX request

   return {
     data: encodedData,
     signature: signature,
     publicKey: publicKeyPem,
    }
  }
```

ДОДАТОК В

Перевірка цифрового підпису з використанням бібліотеки cryptography мовою python

```
def sign(data_to_sign):
    # Генерація закритого та відкритого ключів
    private_key = rsa.generate_private_key(
        public_exponent=65537,
        key_size=2048
    )
    public_key = private_key.public_key()
    # Конвертування словника даних у формат JSON
    data_json = json.dumps(data_to_sign).encode()
    # Підписування даних закритим ключем
    signature = private_key.sign(
        data_json,
        padding.PSS(
            mgf=padding.MGF1(hashes.SHA256()),
            salt_length=padding.PSS.MAX_LENGTH
        ),
        hashes.SHA256()
    )
    # Конвертування відкритого ключа у PEM формат
    pem_public_key = public_key.public_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PublicFormat.SubjectPublicKeyInfo
    )
    return {'signature': signature.hex(),
            'public_key': pem_public_key.decode()}
```