Міністерство освіти та науки України
Львівський національний університет
імені Івана Франка
Факультет прикладної математики та інформатики
Кафедра прикладної математики

Магістерська робота

Розробка системи голосування на основі блокчейну з використанням алгоритму консенсусу доказ минулого часу(PoET)

Виконав: студент групи ПМПм-22
спеціальності
113 - прикладна математика
Сас І.М.

Керівник
Ящук Ю.О.

Рецензент _____

Львів – 2023

Зміст

Перелік умовних позначень, символів, одиниць, скорочень і термінів

- **Алгорим консенсусу (англ. Consensus algorithm)** - алгоритм, що визначає процес підтвердження транзакцій та створення нових блоків в блокчейні.
- **Доказ минулого часу (PoET)** - Proof-of-Elapsed-Time, алгоритм консенсусу, що базується на випадковому очікуванні та безпечному середовищі.
- **Доказ роботи (Proof-of-Work)** - алгоритм консенсусу, що базується на обчислювальній головоломці.
- **Децентралізована мережа** - мережа, що складається з вузлів, які не контролюються однією централізованою організацією, а замість цього взаємодіють між собою напряму.
- **API** - Прикладний програмний інтерфейс (англ. Application Programming Interface), інтерфейс програмування додатків
- **P2P** - Peer-to-Peer, мережа, в якій всі вузли можуть взаємодіяти один з одним без централізованого сервера
- **RSA** - абревіатура від прізвищ Rivest–Shamir–Adleman, криптографічний алгоритм шифрування та підпису повідомлень
- **Блокчейн (англ. Blockchain)** - ланцюжок блоків, що зберігає інформацію про транзакції
- **Транзакція (англ. Transaction)** - це запис у блокчейні, який відображає перехід активів від одного користувача до іншого. Транзакції зазвичай містять адресу відправника та отримувача, кількість переведених активів та підпис цифрового підпису відправника для підтвердження транзакції. Транзакції використовуються для забезпечення безпеки та недублювання операцій в мережі блокчейн.
- **Блок** - частини ланцюжка блоків, які містять транзакції, а також хеш попереднього блоку та деяку інформацію про нього.
- **Смарт-контракти (англ. Smart contracts)** - розумні контракти, що дозволяють автоматизувати процеси виконання угод, в даній роботі це будуть голосування.
- **Валідатор (англ. Validator)** - перевіряє транзакції та створює блоки для додавання до ланцюжка

- **Throughput** - продуктивність системи, тобто кількість транзакцій, що може обробляти система за одиницю часу
- **Latency** - час, що потрібний системі на обробку транзакції
- **Масштабованість (англ. Scalability)** - здатність системи до збільшення обсягу оброблюваних даних без втрати продуктивності
- **Публічний ключ**: це ключ, який можна розповсюджувати відкрито, і який використовується для шифрування повідомлень. Цей ключ зазвичай визначається як пара чисел (N, e), де N - це ціле число, що називається модулем, а e - ціле число, яке називається показником шифрування.
- **Приватний ключ**: це ключ, який використовується для розшифрування повідомлень, і який повинен залишатися секретним. Цей ключ також визначається як пара чисел (N, d), де N - модуль, а d - ціле число, яке називається показником розшифрування.
- **Підпис**: це дані, які додаються до повідомлення для підтвердження автентичності відправника. Це зазвичай досягається за допомогою цифрового підпису, який створюється за допомогою приватного ключа.
- **Хеш (англ. Hash)** - це фіксована довжина вихідних даних, яка отримується в результаті застосування хеш-функції до вхідних даних. В контексті блокчейну, хеш часто використовується для створення ідентифікаторів блоків, транзакцій та інших об'єктів в блокчейні.
- **Хеш-функція**: це функція, яка призначена для перетворення довільного повідомлення фіксованої довжини. Це зазвичай досягається шляхом застосування хеш-функції до повідомлення та отримання фіксованого рядка байтів, який представляє хеш-значення повідомлення.

Вступ

Передумови та мотивація роботи

Використання технології блокчейн для підвищення безпеки та прозорості багатьох систем привертає все більше уваги в останні роки. Системи голосування, які є важливими для функціонування сучасних демократичних країн, є однією із сфер, де ця технологія може мати значний вплив.

Традиційні системи голосування часто стикаються з рядом проблем із безпекою та прозорістю, включаючи можливість фальсифікації голосування, недостатню прозорість у процесі голосування та проблеми з підтвердженням точності результатів. Децентралізовану, прозору систему голосування, яка є захищеною до втручання та маніпуляцій, можна створити за допомогою технології блокчейн.

Метою цієї роботи є розробка та розгортання системи голосування на основі блокчейну, яка може використовуватися для різноманітних виборів, у тому числі місцевих органів влади, корпоративних та інтернет-виборів. Система використовуватиме дозволений блокчейн із механізмом консенсусу, щоб гарантувати точність і легітимність голосів. Щоб автоматизувати процес голосування та гарантувати достовірність результатів, система також міститиме смарт-контракти.

Ми хочемо вирішити проблеми з поточними процесами голосування та запропонувати безпечну, відкриту та ефективну заміну шляхом створення системи голосування на основі блокчейну. Підвищення довіри виборців, менші витрати на адміністрування виборів і більша безпека виборів є потенційними перевагами цього підходу.

Постановка проблеми та питання дослідження

Шахрайство виборців, фальсифікація голосування та відсутність прозорості – це лише деякі проблеми, які виникають у сучасних системах голосування. Несправності виборчих машин, плутанина на виборчих дільницях та інші проблеми, пов'язані з днем виборів, є постійною проблемою в Україні та інших країнах протягом багатьох років і можуть стримувати людей від голосування. Явка виборців може знизитися внаслідок відсутності довіри до виборчого процесу, що підриває саму демократію.

Таким чином, проблема, яку намагається вирішити ця робота, полягає в розробці та впровадженні системи голосування на основі технології блокчейн, яка є безпечною,

відкритою та доступною для всіх кваліфікованих виборців. Ця робота має на меті дати відповіді на наступні запитання дослідження:

- Як можна використовувати технологію блокчейн для створення надійної системи голосування?
- Як можна спроектувати систему голосування на основі блокчейну, щоб вона була доступною для всіх кваліфікованих виборців?
- Як можна захистити конфіденційність виборців і водночас налаштовати систему на запобігання шахрайству виборців і фальсифікаціям голосів?

Рішення цих дослідницьких запитань будемо шукати в створенні пропонованої системи голосування на основі блокчейну та сприятимуть безпечній і відкритій технології голосування.

Мета роботи

Основними цілями цієї роботи є:

- Створити та впровадити систему голосування на основі технології блокчейн, яка гарантує безпеку, прозорість та незмінність процесу голосування.
- Запровадити на практиці алгоритм консенсусу, який дозволяє децентралізованій мережі вузлів узгодити легітимність кожного голосу.
- Розробити смарт-контракти, які оптимізують процес вибору кандидатів і підрахунок голосів.
- Перевірити й оцінити масштабованість (scalability), пропускну здатність (throughput) і продуктивність затримки (latency) запропонованої системи.
- Вивчити переваги та недоліки запропонованого підходу, порівнявши його з поточними процедурами голосування.
- Вказати можливі варіанти використання запропонованої системи та надати рекомендації для подальшого вивчення та розвитку.

Ці цілі керуватимуть розробкою та оцінкою запропонованої системи голосування на основі блокчейну.

Сфера застосування та обмеження

Обсяг і обмеження роботи визначаються, щоб забезпечити чітке розуміння меж і обмежень даної роботи. Запропонована система голосування на основі блокчейну буде розроблена та перевірена для невеликих виборах у контрольованому середовищі.

Масштабованість системи та зручність її використання не перевірятимуться на масштабних виборах, а безпека системи не буде оцінюватися щодо складних атак. Крім того, впровадження системи не передбачає інтеграції з існуючими системами голосування чи правовими рамками. Ці обмеження враховуються через обсяг і ресурси роботи. Проте запропонована система має на меті забезпечити безпечне, прозоре та децентралізоване рішення для голосування на невеликих виборах.

Розділ 1

Дизайн

1.1 Огляд запропонованої системи голосування на основі блокчейна

Дана система голосування на основі блокчейну для проведення онлайн-голосування пропонує безпечну, відкриту та захищену від втручання платформу. Оскільки система базується на ексклюзивному блокчейні, лише авторизовані користувачі можуть брати участь у процесі консенсусу та додавати нові блоки до ланцюжка [1]. Архітектура системи складається з трьох рівнів: рівень API, рівень блокчейну та рівень P2P. У той час як рівень блокчейну зберігає дані голосування та використовує криптографічні методи, такі як хешування та цифрові підписи для перевірки їх цілісності, рівень API надає користувачам простий у використанні інтерфейс для взаємодії з системою [2]. Рівень P2P дозволяє вузлам у мережі спілкуватися та синхронізуватися, дозволяючи їм прийти до згоди щодо стану блокчейну [3].

Алгоритм консенсусу, який використовується в системі, — Доказ минулого часу (PoET), який є алгоритмом Byzantine Fault Tolerant (BFT), який вимагає, щоб вузли чекали випадкову кількість часу, перш ніж запропонувати новий блок [4]. Це гарантує, що жоден вузол не зможе монополізувати процес додавання нових блоків, і зменшує ризик змови або зловмисної поведінки.

Смарт-контракти використовуються для автоматизації виконання правил голосування та забезпечення точності та перевірки результатів.

Загалом запропонована система голосування на основі блокчейну пропонує децентралізовану, безпечну та прозору платформу для проведення онлайн-голосування, забезпечуючи при цьому цілісність і конфіденційність даних голосування.

1.2 Опис архітектури системи

Інтерфейс прикладного програмування (API), блокчейн і P2P рівень складають три основні рівні пропонованої системи голосування на основі блокчейну [5].

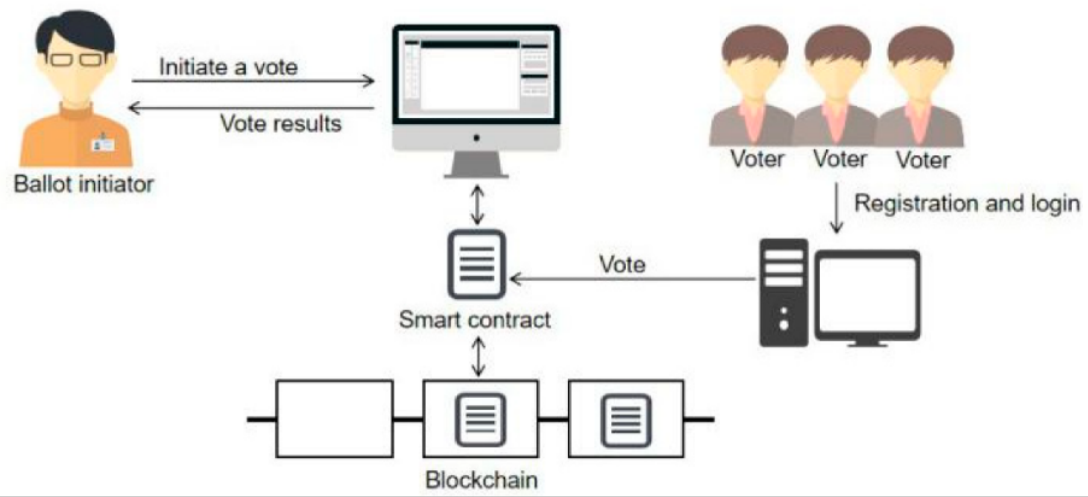

Рис. 1.1: Структура системи голосування

Рівень API служить інтерфейсом зовнішнього користувача і відповідає за обробку запитів користувачів і надання належних відповідей [5]. Це єдиний рівень, який спілкується із зовнішнім світом і забезпечує безпечне з'єднання між користувачем і рівнем блокчейну [6]. Центральний компонент системи, рівень блокчейну, містить записи голосування та смарт-контракти, які визначають процедури голосування [7]. Це гарантує, що записи є точними та незмінними за допомогою алгоритму консенсусу [8].

Рівень P2P відповідає за підтримку топології мережі та полегшення зв'язку між її вузлами [9]. Система стає більш безпечною та стійкою до збоїв завдяки забезпеченню децентралізованого зв'язку між вузлами [10]. Разом ці рівні пропонують надійну та безпечну систему голосування на основі блокчейну, яка гарантує чесність процесу голосування та перешкоджає будь-якому шахрайству.

1.3 Пояснення алгоритму консенсусу та його реалізації

Будь-яка система, заснована на блокчейні, повинна мати консенсусний алгоритм. Алгоритм консенсусу доказ минулого часу (PoET) застосовано в запропонованій нами системі для забезпечення легітимності та цілісності процесу голосування. PoET — це розроблений Intel алгоритм консенсусу, призначений для використання в системах розподіленого реєстру, які залежать від довірених середовищ виконання (TEE), таких як Intel SGX. Однак ми створили спрощену версію, яка не включає SGX для нашої системи.

Алгоритм консенсусу PoET працює так, що кожен учасник мережі чекає випадково згенерований час очікування, перш ніж запропонувати новий блок. Цей час очікування дорівнює часу, який знадобиться одному вузлу для вирішення складної математичної задачі, але без фактичних обчислень. Кожен учасник генерує випадко-

вий час очікування та чекає, поки цей проміжок часу мине, перш ніж запропонувати новий блок.

Коли учасник запропонував новий блок, він транслює пропозицію решті мережі. Потім інші учасники мережі перевіряють запропонований блоку, перш ніж додати її до блокчейну. Процес перевірки передбачає перевірку хешу запропонованого блоку на хеш попереднього блоку, щоб переконатися, що ланцюжок безперервний і нерозривний. Крім того, учасники перевіряють, що запропонований блок містить дійсний набір транзакцій, що транзакції не були раніше оброблені, і що підписи транзакцій є дійсними.

Цей процес гарантує, що мережа досягне консенсусу щодо стану блокчейну, оскільки кожен учасник чекає випадковий період часу, перш ніж запропонувати новий блок, що ускладнює для будь-якого окремого учасника отримання несправедливої переваги. Крім того, оскільки процес перевірки перевіряє запропонований блок на відповідність хешу попереднього блоку, блокчейн стійкий до втручання або пошкодження, оскільки будь-які зміни, внесені до попереднього блоку, будуть легко виявлені мережею.

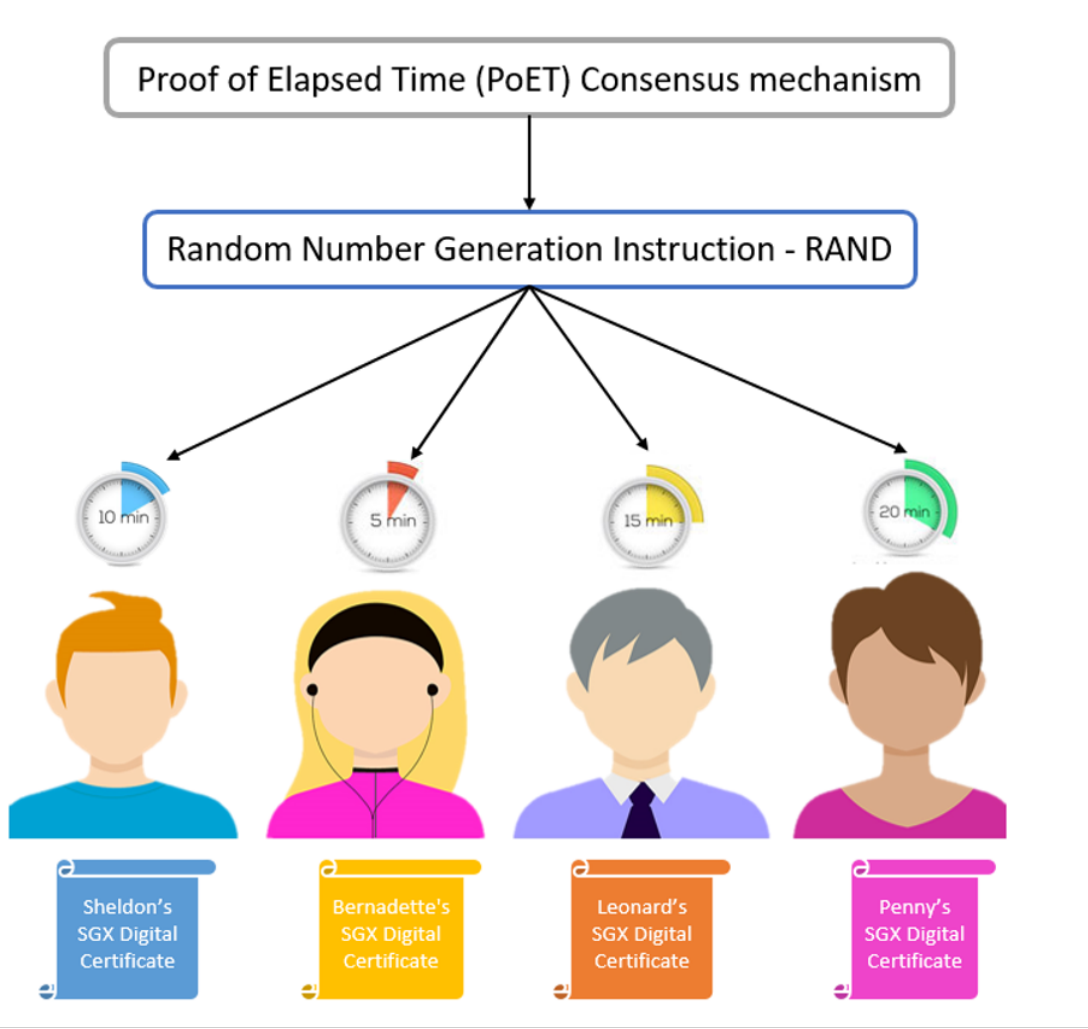

Рис. 1.2: Візуалізація PoET консенсусу

У нашій реалізації ми використовували консенсусний алгоритм Доказ минулого часу (PoET) [11] для перевірки блоків на рівні блокчейну. PoET є різновидом Доказу виконаної роботи (PoW), який зменшує споживання енергії, необхідної для перевірки блоку [4]. У PoET кожен вузол перевірки генерує випадковий час очікування, і вузол із найкоротшим часом очікування вибирається для перевірки наступного блоку. Цей підхід гарантує, що валідатори не можуть передбачити, хто буде обраний для перевірки наступного блоку, що забезпечує справедливу та децентралізовану систему [12].

Хоча для PoET потрібне спеціальне обладнання, наприклад Intel Software Guard Extensions (SGX) [4], ми використали бібліотеку random для генерації часу очікування для кожного учасника нашої реалізації. Такий підхід дозволив нам реалізувати алгоритм консенсусу PoET, не вимагаючи спеціалізованого обладнання, зробивши систему більш доступною для ширшого кола користувачів. Наша реалізація також включає заходи, щоб запобігти маніпулюванню часом очікування зловмисними вузлами, наприклад, вимагати від вузлів надати підтвердження згенерованого часу очікування [2].

1.4 Огляд смарт-контрактів

У мережі блокчейн смарт-контракт — це фрагмент коду, що виконується самостійно. Він містить набір інструкцій і правил, які контролюють, як користувачі спілкуються один з одним у децентралізований і недовірливий спосіб. Смарт-контракт служить основним органом у нашій системі голосування, перевіряючи та відстежуючи голоси, подані учасниками.

Наш смарт-контракт написаний на Python і містить набір функцій, якими можуть користуватися користувачі мережі. Наприклад, коли виборча комісія хоче додати нового кандидата до списку кандидатів, вона викликає функцію додавання кандидата. Коли виборці хочуть проголосувати за певного кандидата, вони викликають функцію голосування. Смарт-контракт також оновлює стан процесу голосування, поки тривають вибори. Коли починається процес голосування, для нього встановлюється стан $in\_progress$ і після завершення голосування $finished$. Тільки учасники, які мають право голосу, можуть голосувати, і кожен виборець може проголосувати лише один раз завдяки смарт-контракту.

Смарт-контракт підраховує результати після завершення голосування та визначає переможця. Метод get_winner повертає кандидата, який отримав найбільшу кількість голосів, тоді як функція $get\_results$ дає загальну кількість голосів, поданих за кожного претендента.

Код смарт-контракту є незмінним, що означає, що коли він був розповсюджений у мережі блокчейн, його неможливо змінити. Це гарантує, що процес голосування є відкритим, безпечним і захищеним від втручання, а результати є неупередженими

та правдивими.

Загалом, процес голосування здійснюється безпечно та відкрито завдяки смарт-контракту, який використовується в нашій системі. Завдяки використанню єдиного договору для голосування та підрахунку голосів процес спрощено, а ймовірність помилок або невідповідностей зменшена. Крім того, процес голосування прозорий і незмінний завдяки використання технології блокчейн.

Розділ 2

RSA алгоритм

2.1 Пояснення мети та значення RSA алгоритму в криптографії

У нашій системі дуже важливу роль грає криптографічне шифрування. Кожна транзакція повинна бути підписана, а інші частини системи зашифровані, щоб досягнути кращої безпеки. Саме для цього ми використовуємо RSA алгоритм. RSA алгоритм є одним з найбільш важливих і популярних алгоритмів шифрування, який використовується в криптографії для забезпечення конфіденційності та цілісності даних. Його основна мета полягає в захисті інформації від несанкціонованого доступу шляхом шифрування повідомлень та підпису цифрових документів. У RSA алгоритмі процес шифрування та розшифрування здійснюється за допомогою двох ключів: публічного та приватного. Публічний ключ розповсюджується відкрито, тоді як приватний ключ зберігається в таємниці від інших користувачів. Завдяки своїм властивостям RSA алгоритм дозволяє безпечно обмінюватися даними відкритими каналами зв'язку, такими як Інтернет. Це робить його незамінним інструментом для захисту інформації в сучасному світі.

2.2 Опис RSA алгоритму

2.2.1 Визначення ключів, їх генерація та розповсюдження

Основна ідея генерації ключів RSA полягає в тому, щоб вибрати два великих простих числа p та q і використовувати їх для побудови публічного ключа n та приватного ключа d. Зокрема:

1. Вибираємо два різних простих числа p та q великих розмірів. Обчислюємо їх добуток $n = pq$, який виступає як публічний ключ RSA.

2. Обчислюємо значення функції Ейлера від числа n, тобто $\varphi(n) = (p-1)(q-1)$.

3. Вибираємо ціле число e, яке є взаємно простим з $\varphi(n)$ та більшим за 1. Число e виступає відкритою експонентою RSA.

4. Знаходимо число d, яке є мультиплікативним оберненим елементом числа e за модулем $\varphi(N)$, тобто $ed \equiv 1 \pmod{\varphi(n)}$. Число d виступає приватною експонентою RSA.

5. Пара чисел (n, e) складає публічний ключ RSA, а пара (n, d) - приватний ключ RSA.

6. Публічний ключ n та відкрита експонента e розповсюджуються, тоді як приватний ключ d залишається в таємниці власника ключа.

Цей процес генерації ключів можна записати у вигляді функцій:

$$p, q \leftarrow \text{generate\_prime}() \tag{2.1}$$

$$n \leftarrow pq \tag{2.2}$$

$$\varphi(n) \leftarrow (p-1)(q-1) \tag{2.3}$$

$$e \leftarrow \text{generate\_co\_prime}(\varphi(n)) \tag{2.4}$$

$$d \leftarrow e^{-1} \bmod \varphi(n) \tag{2.5}$$

$$Public\_Key \leftarrow (n, e) \tag{2.6}$$

$$Private\_Key \leftarrow (n, d), \tag{2.7}$$

де $generate\_prime()$ - функція, що генерує випадкові прості числа, а $generate\_co\_prime(\varphi(n))$ - функція, що генерує випадкове число, яке є взаємно простим з $\varphi(n)$.

Після генерації ключів, користувач може почати використовувати їх для захисту своїх повідомлень. Для шифрування повідомлення, користувач повинен мати доступ до публічного ключа отримувача. Після шифрування повідомлення, тільки отримувач, який володіє відповідним приватним ключем, зможе розшифрувати повідомлення.

Щоб розшифрувати повідомлення, отримувач використовує свій приватний ключ, застосовуючи наступну формулу:

$$m = c^d \bmod n, \tag{2.8}$$

де c - зашифроване повідомлення, d - приватний ключ, а n - модуль, який був використаний при генерації ключів.

Після розшифрування повідомлення, отримувач також може перевірити підпис повідомлення, якщо такий підпис був доданий. Для цього використовується публічний ключ відправника повідомлення:

$$m^e \bmod n = s \tag{2.9}$$

де m - розшифроване повідомлення, e - публічний ключ відправника, а s - підпис повідомлення.

Якщо обидві перевірки пройшли успішно, отримувач може бути впевнений в тому, що повідомлення було надіслано відправником, який володіє відповідним приватним ключем, і що повідомлення не було підроблене під час передавання.

Таким чином, RSA алгоритм забезпечує конфіденційність і цілісність повідомлень, що передаються між користувачами.

2.2.2 Опис шифрування та дешифрування повідомлень

Як ми вже показали раніше агоритм RSA використовує математичні операції для шифрування та дешифрування, тож давайте опишемо їх трохи детальніше.

Опис шифрування повідомлення

Нехай m - повідомлення, яке потрібно зашифрувати, а e та n - публічні ключі RSA. Повідомлення m спочатку перетворюється на ціле число за допомогою кодування повідомлень, наприклад, кодування UTF-8. Позначимо отримане ціле число як m_{int}.

Далі, застосовується функція шифрування $E(m_{int})$, яка обчислюється за формулою: $E(m_{int}) = m_{int}^e \bmod n$

Отримане значення $E(m_{int})$ - зашифроване повідомлення.

Опис дешифрування повідомлення

Нехай c - зашифроване повідомлення, яке потрібно розшифрувати, а d та n - приватні ключі RSA. Зашифроване повідомлення c спочатку перетворюється на ціле число. Позначимо отримане ціле число як c_{int}.

Далі, застосовується функція дешифрування $D(c_{int})$, яка обчислюється за формулою: $D(c_{int}) = c_{int}^d \bmod n$,

Отримане значення $D(c_{int})$ перетворюється на повідомлення за допомогою декодування повідомлень, наприклад, декодування UTF-8. Отже, $D(c_{int})$ - дешифроване повідомлення.

Поєднамо ці формули в один блок для кращої читабельності:

$$E(m_{int}) = m_{int}^e \mod n; \tag{2.10}$$

$$D(c_{int}) = c_{int}^d \mod n, \tag{2.11}$$

де m_{int} - ціле число, отримане з повідомлення m за допомогою кодування повідомлень, c_{int} - ціле число, отримане з зашифрованого повідомлення c.

Отже, ми описали процес шифрування та дешифрування повідомлень у алгоритмі RSA.

2.2.3 Обговорення безпеки RSA алгоритму

Опишемо деякі аспекти безпеки RSA алгоритму:

1. **Довжина ключа**: Для забезпечення безпеки RSA алгоритму, довжина ключа повинна бути достатньою великою. Зазвичай, довжина ключа вимірюється в бітах і повинна бути не менше 2048 біт. Чим довша довжина ключа, тим важче дізнатися зашифровані повідомлення стороннім особам.

2. **Випадковість ключів**: Для того, щоб ускладнити атаки на RSA алгоритм, ключі повинні бути випадковими. Генерація ключів повинна здійснюватися за допомогою криптографічно безпечного генератора випадкових чисел.

3. **Криптографічна стійкість**: RSA алгоритм повинен бути стійким до атак, які можуть бути здійсненні для взламування алгоритму. До таких атак належать підбір ключа, факторизація числа n, атаки з відкритим текстом та інші.

4. **Конфіденційність ключів**: RSA алгоритм передбачає використання двох ключів: публічного та приватного. Публічний ключ розголошується і не є секретним, тоді як приватний ключ повинен бути збережений в таємниці і не розголошуватися нікому. Конфіденційність приватного ключа є критично важливою для забезпечення безпеки RSA алгоритму.

Давайте детальніше опишемо їх:

Нехай n - довжина ключа в бітах. Щоб забезпечити безпеку RSA алгоритму, n повинно бути не менше 2048 біт.

Нехай G - криптографічно безпечний генератор випадкових чисел. Публічний ключ e та приватний ключ d повинні бути випадковими числами, які обираються з деякого інтервалу $[2, n-1]$, де n - число, яке отримується як добуток двох великих простих чисел p та q. Формули для обчислення даних величин були визначені раніше: n (2.2), e (2.4), та d (2.5).

Якщо ключі не є достатньо випадковими, наприклад, якщо вони обираються з надто малого інтервалу або використовуються декілька разів, це може призвести до порушення безпеки RSA алгоритму. Наприклад, може бути здійснено атаку на основі факторизації, де зловмисник може використовувати знання про деякі біти ключа для знаходження інших бітів.

Захист від атак на основі факторизації полягає у використанні достатньо великих простих чисел p та q для генерації числа n. Чим більші значення p та q, тим складніше розкласти n на множники та зламати RSA алгоритм. Крім того, ключі повинні бути

генеровані за допомогою криптографічно безпечного генератора випадкових чисел G.

Одним із додаткових методів захисту є періодична зміна ключів. Наприклад, для кожного повідомлення можна генерувати нову пару ключів e та d. Таким чином, якщо старі ключі були скомпрометовані, то нові ключі захистять повідомлення від дешифрування сторонніми особами.

Описані методи забезпечення безпеки не гарантують 100% захисту від атак, як впринципі жоден з інших методів, проте вони значно знижують ймовірність успішного взлому RSA алгоритму. Загалом, RSA є одним з найбільш відомих криптографічних алгоритмів, які використовуються, і застосовується він в багатьох системах, включаючи інтернет-банкінг, електронних поштах та інших. Проте, як і у будь-якій криптографічній системі, безпека RSA залежить від правильного застосування та належного управління ключами. Тому важливо використовувати надійні ключі, захищати їх від несанкціонованого доступу та періодично змінювати їх. Також необхідно звертати увагу на розмір ключів, який повинен відповідати потребам захисту інформації в конкретному випадку.

Незважаючи на потенційні ризики, RSA залишається одним з найбільш ефективних та надійних криптографічних алгоритмів. Його безпека була успішно підтвердження протягом багатьох років, і він продовжує використовуватися у багатьох системах для захисту конфіденційної інформації.

2.3 Аналіз RSA алгоритму

2.3.1 Побудова функції піднесення до степеня в RSA алгоритмі

У RSA алгоритмі функція піднесення до степеня є однією з основних операцій для шифрування та дешифрування повідомлень. При шифруванні використовується публічний ключ (e, n) (2.6), де $n = pq$ - добуток двох великих простих чисел p та q, а e - ціле число, взаємно просте з $(p-1)(q-1)$. При дешифруванні використовується приватний ключ (d, n) (2.7), де d - обернений елемент e за модулем $(p-1)(q-1)$.

Функція піднесення до степеня у RSA алгоритмі виглядає наступним чином:

$$f(x) = x^y \bmod n, \tag{2.12}$$

де x - вхідне повідомлення, y - публічний або приватний ключ (залежно від того, чи проводиться шифрування, чи дешифрування), n - модуль RSA.

Ця функція використовується як для шифрування, так і для дешифрування повідомлень. При шифруванні, вхідне повідомлення x підноситься до степеня e, а при дешифруванні - до степеня d.

Значення функції обчислюються за допомогою алгоритму швидкого піднесення до степеня, щоб зменшити кількість операцій піднесення до степеня та тим самим зменшити час обробки повідомлення.

2.3.2 Вивчення властивостей функції

Для вивчення властивостей функції піднесення до степеня в RSA алгоритмі, розглянемо наступну функцію (2.12)

Мультиплікативність:

Функція $f(x)$ є мультиплікативною, тобто $\forall x_1, x_2$, що не мають спільних множників з n, виконується наступне:

$$f(x_1 \cdot x_2) = (x_1 \cdot x_2)^e \pmod n = (x_1^e \pmod n) \cdot (x_2^e \pmod n) = f(x_1) \cdot f(x_2) \quad (2.13)$$

Отже, функція $f(x)$ зберігає мультиплікативну структуру.

Періодичність:

Функція $f(x)$ є періодичною з періодом n, тобто $\forall x$ виконується:

$$\begin{aligned} f(x+N) = (x+N)^e \pmod N = (x^e + Ne \cdot x^{e-1} + \\ + \binom{N}{2} e \cdot x^{e-2} + \cdots + N^e) \pmod N = x^e \pmod N = f(x), \end{aligned} \quad (2.14)$$

де використано формулу бінома Ньютона та те, що кожен доданок крім першого містить множник N, тому вони обов'язково будуть згасати при взятті залишку від ділення на N.

Обмеженість значень:

З огляду на те, що за визначенням функції $f(x)$ результат знаходиться у множині $0, 1, \ldots, n-1$, значення функції $f(x)$ є обмеженими.

Враховуючи ці властивості, можна стверджувати, що функція піднесення до степеня в RSA алгоритмі має потрібні властивості для використання в криптографії.

2.3.3 Дослідження монотонності, обмеженості та неперервності функції

Дослідження монотонності, обмеженості та неперервності функцій, що входять до формул RSA алгоритму, допомагає зрозуміти їхню поведінку та використання в криптографії. В даній роботі ми зосередимося на фунції підненення дл степеня.

Функція піднесення до степеня $x \mapsto x^e$ (2.12) є монотонно зростаючою на множині невід'ємних дійсних чисел, оскільки:

$$\forall a, b \geq 0, a \leq b \Rightarrow a^e \leq b^e \quad (2.15)$$

Крім цього, ця функція є обмеженою зверху на будь-якому обмеженому проміжку, оскільки:

$$\forall x, |x| \leq m \Rightarrow |x^e| \leq m^e \tag{2.16}$$

Також, функція піднесення до степеня є неперервною на множині дійсних чисел. Тобто:

$$\forall \epsilon > 0, \exists \delta > 0 \text{ таке, що } \forall x, y, |x - y| < \delta \Rightarrow |x^e - y^e| < \epsilon \tag{2.17}$$

Ці властивості дозволяють використовувати функцію піднесення до степеня в формулі RSA алгоритму з високою впевненістю в її коректності та безпечності.

2.3.4 Аналіз алгоритму швидкого піднесення до степеня

RSA алгоритм містить крок піднесення до степеня з показником, який може мати довільну довжину. Для ефективного виконання цього кроку використовують алгоритм швидкого піднесення до степеня.

Алгоритм швидкого піднесення до степеня полягає у розкладанні показника степеня на бінарний вигляд та знаходженні проміжних результатів, які зводяться до квадрата.

Нехай x - це число, яке потрібно піднести до степеня n та $n = (n_k n_{k-1} \dots n_1)_2$ - бінарний запис показника степеня n, тоді застосовується наступна формула для знаходження x^n:

$$x^n = \prod_{i=1}^{k} x^{2^{i-1} n_i} \tag{2.18}$$

Ця формула дозволяє зменшити кількість операцій піднесення до степеня з $O(n)$ до $O(\log n)$, що забезпечує швидке виконання даного етапу RSA алгоритму.

Аналізуючи складність цієї формули, можна стверджувати, що кількість операцій множення та піднесення до квадрату при застосуванні алгоритму швидкого піднесення до степеня становить $O(\log n)$. Таким чином, загальна складність піднесення до степеня в RSA алгоритмі складається з $O(\log n)$ операцій швидкого піднесення до степеня та $O(\log n)$ операцій множення, тобто $O(\log^2 n)$.

Отже, RSA алгоритм має поліноміальну складність та може бути ефективно застосований для криптографічних застосувань з великими числами.

2.4 Застосування аналізу до RSA алгоритму

2.4.1 Вивчення похідної функції піднесення до степеня

Похідну функції піднесення до степеня x^n можна обчислити за допомогою формули Лейбніца:

$$\frac{d}{dx} x^n = n x^{n-1} \tag{2.19}$$

У RSA алгоритмі, застосовується піднесення до степеня великих цілих чисел, тому похідна необхідна для оцінки точності наближення значень функції. Зокрема, можна використовувати оцінку залишкового члена у формі інтегрального залишку:

$$R_n(x) = \frac{1}{(n-1)!} \int_0^1 (x-t)^{n-1} f^{(n)}(t) dt, \tag{2.20}$$

де $f^{(n)}(t)$ - n-та похідна функції $f(x)$.

Також, можна використовувати формули Тейлора для апроксимації значення функції, з використанням похідних:

$$f(x) = \sum_{k=0}^{n-1} \frac{f^{(k)}(a)}{k!}(x-a)^k + R_n(x), \tag{2.21}$$

де $R_n(x)$ - залишковий член, а a - точка, навколо якої проводиться розклад.

Розглянуті методи застосовуються для отримання більш детальної інформації про збіжність та точність наближення функції піднесення до степеня, який використовується в RSA алгоритмі.

2.4.2 Аналіз взаємодії між різними етапами алгоритму та їх вплив на загальну складність алгоритму

Для аналізу взаємодії між різними етапами RSA алгоритму та їх впливу на загальну складність алгоритму можна використати композиції функцій. Оскільки RSA алгоритм складається з кількох етапів, кожен з яких є функцією від попередніх, можна розглянути алгоритм як композицію цих функцій.

Нехай $f_1, f_2, ..., f_n$ - функції, які входять до складу RSA алгоритму, тоді загальна функція алгоритму може бути представлена як їх композиція:

$$F(x) = f_n(f_{n-1}(\cdots f_2(f_1(x))\cdots)), \tag{2.22}$$

Для аналізу складності виконання RSA алгоритму необхідно визначити складність кожної функції f_i та загальної функції F. Якщо $T(f_i)$ - час виконання функції f_i, то загальний час виконання RSA алгоритму буде дорівнювати:

$$T(F) = \sum_{i=1}^{n} T(f_i), \tag{2.23}$$

Отже, загальна складність алгоритму залежить від складності кожної окремої функції та їх взаємодії.

Також можна визначити складність виконання RSA алгоритму за кількістю операцій з плаваючою комою, цілочисельних операцій та операцій зі степенюванням. Нехай F_{fl}, F_{int} та F_{exp} - кількість операцій з плаваючою комою, цілочисельних опе-

рацій та операцій зі степенюванням відповідно, тоді загальна кількість операцій буде дорівнювати:

$$T_{tot} = F_{fl} + F_{int} + F_{exp}, \tag{2.24}$$

Таким чином, можна проаналізувати взаємодію між різними етапами RSA алгоритму та їх вплив на загальну складність алгоритму. Оскільки піднесення до степеня є основним етапом RSA алгоритму, тому для зменшення часу виконання алгоритму важливо мати ефективні алгоритми піднесення до степеня, одним з таких алгоритмів є алгоритм швидкого піднесення до степеня. Окрім цього, оптимальним вибором ключів та використанням відповідних методів кодування можна досягти максимальної ефективності та безпеки алгоритму.

Для прикладу, вибір простих чисел, що входять до складу ключів, повинен виконуватися з особливою уважністю, оскільки недостатньо складні числа можуть призвести до зниження безпеки системи. Крім того, вибір ефективного алгоритму генерації випадкових чисел є також важливим для забезпечення безпеки алгоритму.

Отже, за допомогою аналізу можна зрозуміти взаємодії між різними етапами RSA алгоритму та їх впливу на загальну складність алгоритму. Застосування цих знань допоможе підвищити ефективність та безпеку RSA алгоритму.

2.4.3 Використання диференціювання та інтегрування для вивчення властивостей функції

Диференціювання та інтегрування є ще одними додатковими методами аналізу, які можуть бути застосовані для отримання додаткової інформації про RSA алгоритм.

Диференціювання та інтегрування - це могутні інструменти, які дозволяють вивчати різні властивості функцій, в тому числі тих, що входять до формул RSA алгоритму. Зокрема, диференціювання дозволяє знайти похідну функції, що може бути корисно для аналізу швидкодії RSA алгоритму.

Для прикладу, найбільш важливою операцією у RSA алгоритмі є обчислення криптографічкої функції (2.12). Запишемо її з іншими позначеннями, щоб не плутатися:

$$f(x) = x^e \mod n, \tag{2.25}$$

де x - повідомлення, e - публічний ключ, а n - добуток двох великих простих чисел.

Диференціюючи цю функцію по змінній x, отримаємо:

$$\frac{df(x)}{dx} = ex^{e-1} \mod n, \tag{2.26}$$

Таким чином, можна оцінити швидкість роботи алгоритму, знаючи значення e та використовуючи арифметику з числами довільної точності.

З іншої сторони, інтегрування, може бути корисним інструментом для розуміння взаємозв'язку між функціями у формулах RSA алгоритму. Наприклад, розглянемо формулу для дешифрування:

$$g(y) = y^d \mod n, \tag{2.27}$$

де y - зашифроване повідомлення, d - приватний ключ, а n - добуток двох великих простих чисел.

Застосовуючи метод інтегрування, можна показати, що ця функція є оберненою до криптографічкої функції $f(x)$(2.25):

$$g(f(x)) = x \mod n, \tag{2.28}$$

Отже, ми показали, що можна використовувати інтегрування для знаходження оберненої функції в формулах RSA алгоритму. Диференціювання ж, в свою чергу, може бути корисним для вивчення поведінки функції на певному інтервалі та знаходження мінімумів та максимумів. Також диференціювання може використовуватись для побудови табличних значень функцій та для перевірки коректності обчислень.

Підсумовуючи, використання диференціювання та інтегрування дозволяє вивчати різні властивості функцій, які входять до формул RSA алгоритму, що допоможе зрозуміти, як сам алгоритм працює та знайти способи оптимізації його роботи.

2.5 Застосування та безпека RSA алгоритму

2.5.1 Аналіз впливу вибору параметрів RSA алгоритму на його безпеку та ефективність

Для вивчення властивостей функцій, що входять до RSA алгоритму, як ми вже показали раніше, можна використовувати диференціювання та інтегрування. Однією з головних задач є аналіз впливу вибору параметрів RSA алгоритму на його безпеку та ефективність.

Для цього розглянемо формулу (2.10) для шифрування повідомлення m ключем e в такому вигляді:

$$c \equiv m^e \pmod n, \tag{2.29}$$

де $n = pq$ - добуток двох великих простих чисел p та q, іншими словами, n - модуль шифрування, а e - відкрита експонента.

Довжина ключа може бути визначена, як кількість бітів в n. Чим більша довжина ключа, тим безпечнішим є алгоритм, оскільки його шифрування буде складніше зламати. Аналіз впливу довжини ключа на безпеку RSA алгоритму можна провести, використовуючи диференціювання та інтегрування.

Окрім цього, вибір відкритої експоненти e може впливати на безпеку алгоритму. Здебільшого використовують значення e з малою кількістю одиниць у двійковому представленні. З іншого боку, занадто малі значення e можуть зробити алгоритм вразливим до атак типу "найбільший спільний дільник"(англ. GCD).

Нарешті, вибір простих чисел p та q також впливає на безпеку алгоритму. Якщо прості числа занадто малі, то зловмисник може легко знайти їх добуток n шляхом перебору всіх можливих комбінацій. З іншої сторони, якщо прості числа занадто великі, то шифрування та розшифрування повідомлення займатимуть надто багато часу, тож треба врахувати це також.

Таким чином, вибір простих чисел потребує збалансованого підходу, що базується на аналізі властивостей функцій, що входять до формул RSA алгоритму. Зокрема, використовуючи диференціювання та інтегрування для дослідження функцій, що описують вибір простих чисел p та q. Для прикладу, аналізувати властивості функцій, що описують ймовірність знаходження простого числа на відрізку певної довжини, або властивості функцій, що описують складність факторизації числа на добуток простих множників.

Окрім цього, можна використовувати інтегрування для дослідження складності алгоритму RSA при різних значеннях параметрів, таких як довжина ключа чи вибір відкритої експоненти. Наприклад, можна дослідити вплив зміни довжини ключа на складність атаки методом перебору чи методом факторизації. Застосування аналізу впливу параметрів RSA алгоритму на його безпеку та ефективність може допомогти знайти оптимальні значення цих параметрів, що забезпечать належний рівень безпеки та ефективності алгоритму.

2.5.2 Дослідження атак на RSA алгоритм

Атака на основі факторизації на RSA алгоритм полягає у спробі знайти прості множники p та q числа n для отримання приватного ключа d. Якщо p та q вийде знайти, то d можна обчислити за формулою (2.11). Запишемо її в такому вигляді:

$$d \equiv e^{-1} \pmod{\varphi(n)}, \tag{2.30}$$

де e - відкрита експонента, а $\varphi(n) = (p-1)(q-1)$ - функція Ейлера для n.

Одним з методів факторизації є метод факторизації Ферма, який полягає у пошуку таких цілих чисел x та y, що $n = x^2 - y^2$, що дає змогу знайти прості множники p та q.

Ще одним типом атаки є атака на вибір параметрів, яка полягає у використанні псевдовипадковості вибору параметрів RSA для отримання приватного ключа або розшифрування повідомлень. Один з методів такої атаки - атака Боне-Хопа-Йо-Троупера, яка використовує властивості системи чисел з плаваючою комою для знаходження приватного ключа.

Щоб запобігти таким атакам, рекомендується використовувати достатньо великі прості числа p та q для створення числа n, а також вибирати відкриту експоненту e таким чином, щоб вона була взаємно простою зі значенням функції Ейлера $\varphi(n)$. Крім цього, можна використовувати такі методи, як генерація псевдовипадкових чисел для вибору параметрів RSA та застосування додаткових технік шифрування, наприклад, підписування повідомлень.

Для аналізу ефективності та безпеки RSA алгоритму можна використовувати функції витрат, такі як функція витрат часу або функція витрат ресурсів. Такі функції можуть допомогти оцінити складність алгоритму та його вразливість до атак.

Наприклад, функція витрат часу $T(n)$ визначає час, який потрібен для виконання алгоритму з вхідним розміром n, тоді як функція витрат ресурсів $S(n)$ визначає необхідність ресурсів (наприклад, пам'яті) для виконання алгоритму з вхідним розміром n.

Для аналізу атак на основі факторизації на RSA алгоритм можна використовувати функцію витрач $\mathcal{O}$, яка вказує, як збільшується час виконання алгоритму зі збільшенням розміру вхідних даних. За різними оцінками, атаки на основі факторизації на RSA мають складність $\mathcal{O}(e^{c\sqrt[3]{\ln n \ln\ln n}})$ або $\mathcal{O}(n^{1/4+o(1)})$, де n - це добуток двох великих простих чисел, які використовуються для створення RSA ключів.

Щодо атак на вибір параметрів RSA, то функції витрат можуть допомогти визначити оптимальні параметри для забезпечення максимальної безпеки та ефективності. Наприклад, функція витрат часу може допомогти визначити оптимальну довжину ключа RSA, а функція витрат ресурсів може допомогти визначити оптимальний вибір відкритої експоненти для забезпечення безпеки та ефективності.

2.5.3 Обговорення застосування RSA алгоритму в реальних системах та проблем з його використанням

RSA алгоритм є одним з найпопулярніших та найнадійніших алгоритмів шифрування, який застосовується у багатьох реальних системах. Однак, під час його використання можуть виникнути проблеми, тож важливо обговорити деякі проблеми, пов'язані з застосуванням RSA алгоритму в реальних системах. Одна з проблем полягає в практичній реалізації алгоритму. Зокрема, великі числа, які використовуються у RSA, можуть призводити до значної витрати обчислювальних ресурсів, що може стати проблемою для обчислювально обмежених пристроїв.

При практичній реалізації RSA алгоритму, можуть виникати проблеми з швидкістю шифрування та розшифрування повідомлень, особливо при використанні довгих ключів. Це пов'язано зі складністю обчислень піднесення до степеня за модулем, яке є основною операцією в RSA алгоритмі. Для оцінки ефективності RSA алгоритму можна використовувати функції витрат, такі як функція витрат на час або функція витрат на ресурси.

Крім того, критично важливо аби ключі RSA алгоритму були збережені в безпечному місці, оскільки їх втрата або підміна може призвести до порушення безпеки системи. Для зберігання ключів можна використовувати спеціальні протоколи, такі як протоколи розподілену ключів (англ. *key splitting protocols*) або протоколи шифрування з використанням асиметричного шифрування. Якщо приватний ключ потрапляє в руки зловмисника, то всі дані, зашифровані цим ключем, можуть бути розшифровані. Тому, важливо забезпечити надійний захист приватних ключів, наприклад, за допомогою апаратного захисту або захисту від несанкціонованого доступу.

Для оцінки безпеки та ефективності RSA алгоритму можна використовувати функції витрат, такі як функція витрат часу або функція витрат ресурсів, які ми описали раніше. Такі функції допомагають оцінити час та ресурси, необхідні для шифрування або розшифрування повідомлень з використанням RSA алгоритму.

Нарешті, важливо зазначити, що RSA алгоритм не є невразливим до атак, і може бути взламаним за допомогою факторизаційних атак, атак на вибір параметрів та інших методів. Однак, як говорилося раніше, при збільшенні довжини ключа, ризик таких атак буде значно зменшений.

Загалом, RSA алгоритм є потужним інструментом для захисту конфіденційної інформації, але він має свої обмеження та потребує відповідного застосування та захисту приватних ключів.

2.6 Висновки до розділу

Підсумовуючи з викладеного, RSA алгоритм є одним з найбільш важливих криптографічних алгоритмів, що забезпечують конфіденційність даних та захист від несанкціонованого доступу. Застосування алгоритму базується на математичних функціях, таких як піднесення до степення та операції з модулем, що дозволяють шифрувати та розшифровувати повідомлення.

RSA алгоритм є вразливим до атак на основі факторизації, які базуються на розкладанні складних чисел на прості множники. Такі атаки можуть бути успішними, якщо використовувані прості числа занадто малі. Для захисту від таких атак можна використовувати ключі більшої довжини.

Вибір відкритої експоненти також може впливати на безпеку алгоритму. Якщо вибрана відкрита експоненти надто мала, то зловмисник може легко підібрати відповідний приватний ключ шляхом перебору всіх можливих значень. З іншого боку, великі значення відкритої експоненти можуть призвести до сповільнення шифрування та розшифрування повідомлень.

Нарешті, проблеми з практичною реалізацією та збереженням ключів можуть виникати при застосуванні RSA алгоритму в реальних системах. Тож необхідно вирішувати проблеми з безпекою зберігання ключів, а також забезпечувати відповідність

довжини ключів залежно від використовуваної криптографічної системи.

Таким чином, RSA алгоритм є потужним інструментом для захисту конфіденційної інформації та має широке застосування в сучасній криптографії. Використання RSA алгоритму може бути обмежене через його вимоги до обчислювальних ресурсів та збереженням ключів, що може ставати проблематичним в деяких застосуваннях. Також, існують певні виклики та проблеми, пов'язані з безпекою RSA алгоритму, зокрема з його використанням в системах, що використовують стандартні прості числа для генерації ключів. Однак, з правильним налаштуванням та використанням додаткових методів захисту, RSA алгоритм може бути ефективним інструментом для захисту інформації. В цілому, RSA алгоритм є важливим елементом сучасної криптографії, який продовжує бути вивченим та вдосконалюваним в майбутньому.

Розділ 3

Реалізація

3.1 Технічні деталі впровадження системи

Система голосування була реалізована за допомогою мови програмування Python, бібліотека Flask була використана для API інтерфейсу для користувача та бібліотекою RSA для криптографічних операцій. Система використовує спеціальну реалізацію блокчейну, яка використовує алгоритм консенсусу доказ минулого часу (PoET).

Алгоритм консенсусу доказ минулого часу

Алгоритм консенсусу PoET використовує розширення Intel Software Guard Extensions (SGX), щоб забезпечити випадковий час очікування для кожного вузла валідатора, перш ніж дозволити йому запропонувати новий блок. Цей алгоритм використовує підхід, на основі лідера. Кожен валідатор має однакову ймовірність бути обраним тим хто запропонує блок, оскільки цей час очікування визначається на основі довіреного середовища виконання (TEE) на вузлі валідатора. Однак в даній реалізації було створено спрощену версію, яка не включає TEE для нашої системи, а натомість використовує звичайну Python бібліотеку Random, для визначення часу очікування. Даний підхід може бути в майбутньому покращеним і TEE може бути доданим.

Реалізація блокчейна

Для реалізації блокчейну використовуються вбудовані структури даних мови програмування Python і бібліотека RSA. Кожен блок містить список транзакцій та іншу інформацію. До іншої інформації входять хеш транзакцій блоку, хеш попереднього блоку та час коли блок був створений. Загалом можна було б ще додати інші дані, такі як номер блоку, відкритий ключ валідатора, випадковий час очікування, створений алгоритмом PoET, але для спрощення реалізації було вирішено не включати ці властивості до блоку.

Нова транзакція користувача додається до пулу незавершених транзакцій, коли користувач робить якусь дію з блокчейном, наприклад додає нового кандидата до голосування. На основі алгоритму PoET новий валідатор-лідер регулярно вибирається та створює новий блок, що включає всі незавершені транзакції. Потім мережа додає блок до блокчейну та видаляє незавершені транзакції з пулу, коли блок буде

підтверджено.

Обробка транзакцій

Транзакції, надіслані користувачами, повинні виконуватися процесором транзакцій. Щоб завершити транзакцію та оновити статус, він зв'язується з відповідним смарт-контрактом. Перед додавання до пулу незавершених транзакцій кожна транзакція підписується закритим ключем користувача та незалежно підтверджується вузлом перевірки.

Реалізація смарт-контракту

Бізнес-логіка для реєстрації кандидатів і підрахунку голосів міститься в смарт-контракті, який також написаний на Python. Смарт-контракт визначає, як додавати нових кандидатів, період початку та закінчення періоду голосування, власне сам процес голосування та отримання результатів голосування. Щоб забезпечити послідовність і уникнути подвійного голосування, кожна транзакція, яка впливає на стан блокчейну, виконується через смарт-контракт.

3.2 Пояснення заходів безпеки системи

Щоб захистити інформацію користувача та транзакції, система використовує різні заходи безпеки. Під час запуску системи користувачу надається спеціальна пара відкритих і закритих ключів. Для генерації ключів використовується популярний алгоритм створення захищених цифрових підписів Elliptic Curve Digital Signature Algorithm (ECDSA) [13].

Закритий ключ використовується для підпису транзакцій перед їх додаванням у блокчейну і надійно зберігається на пристрої користувача. Під час процедури перевірки цифровий підпис транзакції перевіряється за допомогою відкритого ключа. Щоб захистити особистий ключ від рук несанкціонованих осіб, система використовує асиметричне шифрування.

Крім того, система вживає заходів для захисту від несанкціонованого доступу до особистих ключів. Надійні методи шифрування використовуються для шифрування ключів, які потім надійно зберігаються на пристрої користувача. Ключі доступні лише на поточному пристрої і ніде більше не можуть бути передані.

Щоб досягнути ще більшої надійності система могла б використовувати облікові записи та багатофакторну автентифікацію (MFA), щоб переконатися, що лише авторизовані користувачі можуть отримати доступ до особистого облікового запису користувача з метою подальшого підвищення безпеки. Одноразові паролі, токени та біометрична автентифікація – лише деякі з технологій, які можна було б використовувати для створення MFA.

Потужний механізм аудиту, який реєструє всі дії користувачів, також є особливістю системи. Кожна дія документується в блокчейні, що зберігає систему децентралізованою та безпечною, одночасно дозволяючи будь-якому користувачеві перевіряти

цілісність системи.

Крім того, всі дані, якими обмінюються пристрій користувача та сервер, шифруються системою за допомогою HTTPS [14]. Це гарантує, що особиста інформація користувача захищена від сторонніх очей і небажаного доступу.

Для безпеки системи та конфіденційності даних і транзакцій користувачів вкрай важливо забезпечити безпечне керування ключами, механізми аудиту та шифрування HTTPS. Ці процедури безпеки гарантують, що всі дії та транзакції користувачів реєструються з метою перевірки, захищають дані користувача від прослуховування та допомагають запобігти несанкціонованому доступу до особистих ключів. Поєднання цих заходів безпеки також сприяє збереженню даних користувачів системи та цілісності системи.

Загалом безпека системи є важливою для збереження довіри користувачів до платформи. Розробники повинні надавати безпеці першочерговий пріоритет при плануванні та виконанні своїх систем, оскільки використання технології блокчейн розширюється. Цими заходами безпеки розробники можуть допомогти переконатися, що їхні додатки на основі блокчейну безпечні, надійні та стійкі до атак, запровадивши надійні заходи безпеки та дотримуючись найкращих практик.

3.3 Опис інтерфейсу користувача системи та взаємодії з користувачем

Інтерфейс користувача системи заснований на рівні API, який дозволяє користувачам взаємодіяти з блокчейном і відправляти транзакції за допомогою HTTP-запитів. Рівень API має необхідні точки доступу (англ. endpoints) для доступу до різних функціональних можливостей системи, включаючи додавання нових кандидатів, початок і завершення періодів голосування, створення нових смарт-контрактів і власне голосування.

Щоб проголосувати, користувачі повинні надати свій відкритий ключ і підписану транзакцію голосування, що містить кандидата, за якого вони голосують. Рівень API, створить дану транзакцію на основі переданих даних, підпише та додасть її до блокчейну, якщо вона вважатиметься дійсною.

Користувачі можуть також використовувати рівень API для доступу до даних з блокчейну. Включаючи доступ до списку P2P вузлів і валідаторів, запити про поточний стан блокчейну та отримання результатів завершених голосувань.

Рівень API шифрує всі запити за допомогою HTTPS, щоб захистити безпеку та конфіденційність даних користувача. Щоб підписувати транзакції та ідентифікувати свою особу, система генерує закриті ключі для кожного користувача.

Незважаючи на те, що система зараз не має графічного інтерфейсу користувача, вона створена для простої інтеграції з ним у майбутньому. У результаті користувачі зможуть взаємодіяти з блокчейном через зручний інтерфейс, такий як веб-додаток

або мобільний додаток.

Загалом, інтерфейс користувача системи створено таким чином, щоб бути простим у використанні та зрозумілим, захищаючи конфіденційність і безпеку користувача.

3.4 Опис P2P мережі

Система голосування на основі блокчейну використовує два типи протоколів зв'язку: API та P2P. API забезпечує простий і стандартизований спосіб для зовнішніх додатків взаємодіяти з мережею блокчейн, тоді як зв'язок P2P забезпечує прямий зв'язок між вузлами в мережі.

Пряме з'єднання між вузлами в мережі блокчейн стає можливим за допомогою зв'язку P2P. Для синхронізації стану блокчейну та обміну даними про нові транзакції та блоки вузли мережі з'єднуються один з одним.

Для реалізації зв'язку P2P використовується протокол однорангової мережі. Кожен вузол у мережі відстежує інші вузли, до яких він підключений, і спілкується з ними для обміну даними. Протокол створений як відмовостійкий, і якщо вузол від'єднується від мережі, він миттєво підключається знову.

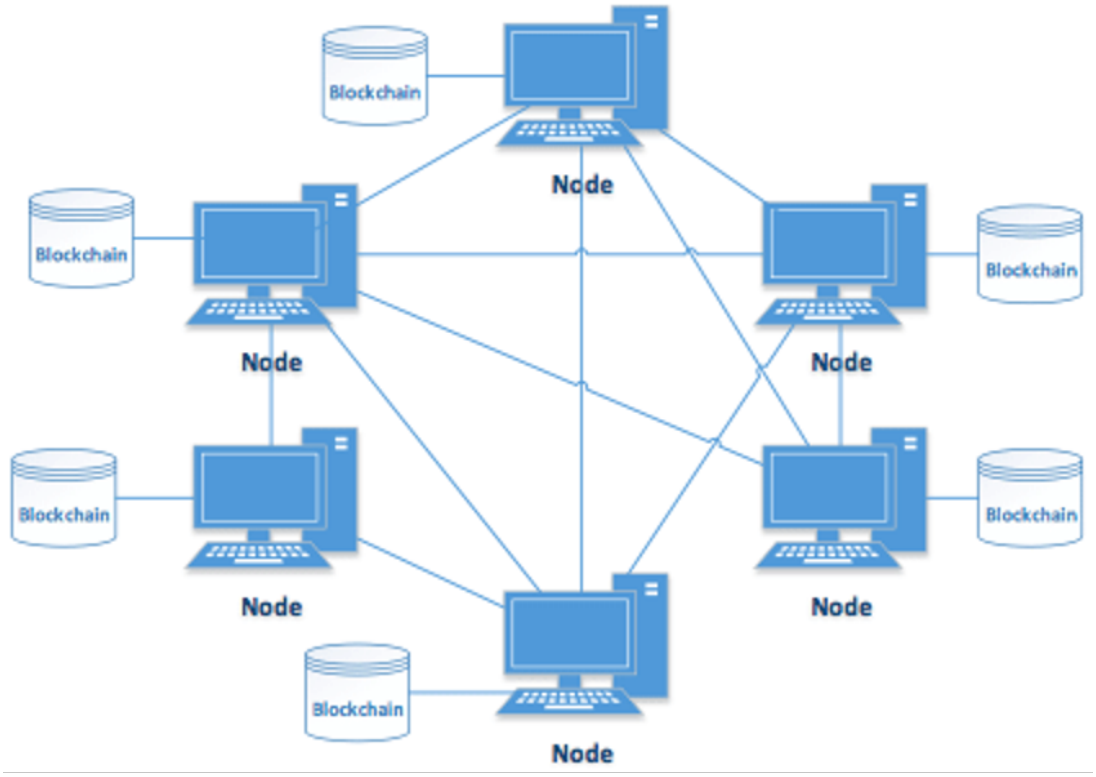

Рис. 3.1: Візуалізація P2P мережі

Протокол P2P також створений для забезпечення безпеки та захисту від нападів. Усі мережеві вузли автентифікують один одного за допомогою криптографії, і кожен зв'язок шифрується, щоб перешкодити підслуховуванню та втручанню.

P2P-сервер реалізує децентралізовану програму та спілкується з вузлами за допомогою веб-сокетів. P2P-сервер транслює власні повідомлення іншим вузлам мережі, прослуховуючи нові повідомлення, такі як транзакції, блоки, контракти та запити на синхронізацію. Крім того, P2P-сервер дозволяє вузлам знаходити інші однорангові мережі та підключатися до них [15].

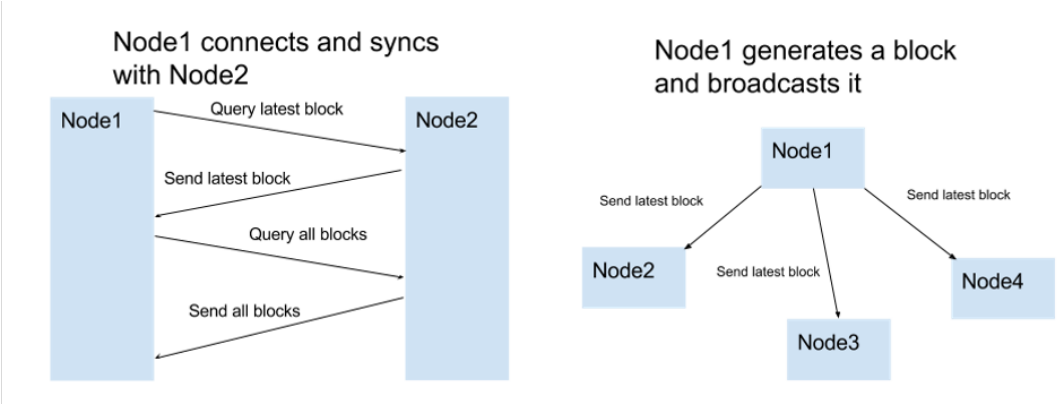

Рис. 3.2: Комунікація між вузлами мережі

Протокол P2P гарантує, що мережа блокчейну є децентралізованою, що означає, що немає жодної відповідальної особи, яка могла б впливати на неї чи контролювати її. Натомість кожен вузол у мережі має копію блокчейну, і перед тим, як транзакції додаються до блокчейну, вони перевіряються консенсусним процесом. Навіть якщо деякі вузли в мережі зловмисні або мають несправне обладнання, механізм консенсусу гарантує, що всі в мережі погоджуються щодо стану блокчейну [2].

P2P протокол є критично важливою компонентою системи голосування на основі блокчейну. Він забезпечує децентралізацію та безпеку мережі блокчейну. З його допомогою, можна забезпечити прозору та безпечну систему голосування, несприйнятливу до маніпуляцій та шахрайства.

Розділ 4

Результати

4.1 Демонстрація можливостей системи

Демонстрація можливостей системи включає важливі характеристики для безперебійної роботи системи голосування. Цей функціонал дозволяє додавати нових кандидатів, створювати нові головування та анонімно та безпечно голосувати.

Що дозволяє система:

1. Створювати нові голосування
2. Додавати кандидатів до голосування
3. Починати/закінчувати голосування
4. Власне проводити голосування
5. Отримувати результати після закінчення
6. Отримання всієї необхідної інформації для аудиту

Користувач може подати транзакцію за допомогою рівня API. Власне транзакція, як ми описали раніше є це будь-яка дія з блокчейном. Якщо конкретніше то це створення головування, змінення його стану, додавання нового кандидата та власне голосування за одного з них. Рівень блокчейну перевіряє цю транзакцію, щоб переконатися, що вона відповідає вимогам, наприклад для додавання нового кандидата він не має ще існувати, або для голосування чи це вперше користувач голосує. Транзакція транслюється всім мережевим вузлам після перевірки та поміщається в блокчейн.

Те саме стосується створення нового контракту; користувач може зробити це, відправивши транзакцію, яка містить параметри контракту, включаючи дати початку та закінчення періоду голосування та мінімальну кількість голосів, необхідну для того, щоб він вважався дійсним. Ще раз нагадаємо, якщо ця транзакція задовольняє відповідні вимоги, вона перевіряється та завантажується в блокчейн.

Користувач має можливість подати транзакцію для обраного голосування. Щоб гарантувати, що ця транзакція є анонімною та не може бути пов'язана з ними, вона шифрується за допомогою їхнього закритого ключа. Голос вважається відданим і не може бути змінений після перевірки та розміщення в блокчейні.

Рішення на основі блокчейну гарантує, що всі ці функції виконуються безпечним і прозорим способом, що є важливим для ефективної роботи системи голосування.

4.2 Аналіз продуктивності системи

Щоб оцінити продуктивність системи, ми використали інструмент Locust, який є широко використовуваним інструментом тестування навантаження з відкритим вихідним кодом, який дозволяє моделювати тисячі одночасних користувачів для визначення поведінки системи з різним рівнем навантаження. Завдяки використанню Locust ми змогли отримати цінну інформацію щодо пропускної здатності системи, затримки та масштабованості.

Стрес-тестування системи виявило, що коли більше 100 користувачів надсилають свої транзакції одночасно, система використовує занадто багато портів для підтримки синхронізації p2p-сервера. Це призводить до поломки системи, оскільки всі вільні порти, які дозволено використовувати веб-сокетам, зайняті. Однак ця проблема зменшиться, якщо кожен вузол матиме власний коп'ютер/сервер.

Пропускна здатність системи вимірювалася за допомогою інструменту Locust, який фіксував кількість транзакцій, оброблених за секунду (TPS). Було виявлено, що система здатна обробляти пропускну здатність 1,67 TPS. Ця пропускна здатність може бути достатньою для невеликої системи, але може потребувати покращення для більших систем або великих обсягів транзакцій.

Рис. 4.1: Визначення пропускної здатності системи

Кінцева точка *get blockchain* використовувалася для надсилання запитів з метою розрахунку затримки системи (Latency). Висновки показали, що затримка зросла разом із кількістю запитів на секунду. Затримка стабілізувалася на рівні 30 секунд, коли система досягла 3,5 RPS (Request per seconds) для 100 користувачів. Хоча ця затримка може здатися великою, насправді її виміряли під час сильного навантаження. Наприклад, у випадку, коли лише 40 користувачів отримували доступ до блокчейну з одного вузла, а RPS становив 2,6, затримка становила 3,9 секунди. Це показує, що система може ефективно функціонувати за типових умов.

Рис. 4.2: Визначення затримки системи

Масштабованість системи була перевірена збільшенням кількості користувачів. Було виявлено, що коли кількість користувачів за секунду зросла до 150, система почала відчувати проблеми з продуктивністю, що призвело до збоїв. Це вказує на те, що систему, можливо, потрібно буде оптимізувати для більшої кількості користувачів або сценаріїв із високим трафіком. Важливо зазначити, що ці висновки свідчать про те, що система може потребувати додаткових ресурсів, таких як більш потужне апаратне забезпечення або розподілена архітектура, щоб підтримувати прийнятний рівень продуктивності в умовах надзвичайно високого навантаження.

Рис. 4.3: Оцінка масштабованості системи

Перевірка в цілому демонструє, що система має прийнятну пропускну здатність і затримку за типових обставин, але, можливо, її потрібно буде оновити для більших систем або великих обсягів транзакцій. Щоб керувати сценаріями з інтенсивним трафіком або більшою кількістю користувачів, системі також може знадобитися оптимізація.

Розділ 5

Обговорення

5.1 Оцінка цілей роботи та питань дослідження

Основною метою цього дослідження було створення системи голосування на основі консенсусного алгоритму доказ минулого часу (PoET). Децентралізована, безпечна, прозора та здатна керувати масовими голосуваннями — такими були цілі системи.

Завдяки використанню алгоритму консенсусу PoET, який забезпечує децентралізований і безпечний процес голосування, ціль була досягнута. Система непроникна завдяки гарантії блокчейн-технології незмінності зареєстрованих транзакцій, що робить її прозорою, оскільки всі транзакції реєструються в загальнодоступному блокчейні.

Дослідницькі питання, якими керувалася робота, були:

1. Чи можна підвищити безпеку та прозорість процесу голосування за допомогою системи голосування на основі блокчейну з використанням алгоритму консенсусу PoET?

2. Чи здатна система обробляти важливі події голосування?

Результати роботи показують, що система голосування, яка побудована на блокчейні та використовує алгоритм консенсусу PoET, може підвищити безпеку та прозорість процесу голосування. Децентралізований відкритий реєстр запобігає маніпуляціям і забезпечує прозорість системи, гарантуючи, що всі транзакції задокументовані та відкриті для публічного огляду. Крім того, вибираючи валідатори у чесний і випадковий спосіб, алгоритм консенсусу PoET підтримує безпеку системи, запобігаючи будь-якій стороні контролювати мережу.

Під час тестування система була розширена до 100 транзакцій на секунду, що означає, що вона може бути використана для невеликих голосуваннь. Щоб визначити верхні межі та масштабованість системи під час масштабних подій голосування, знадобиться додаткове обладнення та тестування.

Загалом було успішно досягнуто поставлених цілей і отримано відповіді на запитання дослідження, продемонструвавши, що система голосування на основі блокчейну з використанням алгоритму консенсусу PoET може покращити безпеку та прозорість процесу голосування та має потенціал для проведення широкомасштабних голосувань.

5.2 Порівняння запропонованої системи з існуючими системами голосування

Запропонована система має ряд переваг порівняно зі звичайними процедурами голосування з точки зору безпеки, відкритості та доступності. Оскільки кожен голос шифрується та зберігається в незмінному реєстрі за допомогою технології блокчейн, запропонований метод забезпечує високий рівень безпеки та ускладнює зміну чи маніпулювання результатами [2]. Крім того, оскільки правила виборів заздалегідь визначені та незмінні, впровадження смарт-контрактів гарантує чесність і прозорість процесу голосування [16].

Крім того, запропонована система є дуже доступною, оскільки до неї можна отримати доступ з будь-якого місця, де є підключення до Інтернету, що дає змогу більшій кількості людей брати участь у процесі голосування. Це особливо важливо для осіб, які живуть у віддалених районах або мають проблеми з пересуванням, оскільки вони можуть не мати змоги відвідати фізичні дільниці для голосування [17].

З іншої сторони, традиційні системи голосування часто вимагають від виборців фізичної присутності на визначеній виборчій дільниці, що може бути незручним для деяких осіб. Крім того, традиційні системи голосування вразливі до ряду проблем безпеки, таких як підробка бюлетенів чи видавання себе за іншу особу. Ці проблеми можуть підірвати цілісність виборчого процесу та призвести до неточних або фальсифікованих результатів.

Як наслідок, запропонований підхід пропонує безпечнішу та зручнішу заміну поточних процедур голосування. Важливо пам'ятати, що запропоноване рішення все ще має деякі недоліки. Виборцям потрібен доступ до комп'ютера чи мобільного пристрою, що може бути доступним не всім, і це залежить від наявності Інтернету, який може бути доступним не у всіх місцях. Щоб гарантувати, що запропонована система може бути використана якнайширшою аудиторією, ці обмеження повинні бути розглянуті [5].

Запропонований підхід, який потенційно може підвищити явку виборців і гарантувати чесні та точні результати виборів, пропонує значну перевагу порівняно з традиційними системами голосування з точки зору безпеки, прозорості та доступності.

5.3 Обговорення потенційних заявок і майбутньої роботи

Запропонована система електронного голосування на основі блокчейну має значний потенціал для майбутнього застосування в різних сферах, включаючи урядових, освітньому, бізнесі та громадських організаціях. Наше прикладне дослідження демонструє, як цей підхід може покращити ефективність, безпеку та відкритість процесу голосування під час виборів влади. Вартість і складність процесу голосування можна зменшити, використовуючи технологію блокчейн, щоб позбутися потреби в посередниках.

Щоб забезпечити чесність і цілісність процесу голосування, цю техніку можна використовувати в освітньому секторі для виборів студентської ради та інших заходів, які проводять студенти. Ця система може бути використана бізнесом і громадськими організаціями для проведення опитувань і процесів прийняття рішень, які вимагають безпечних і незламних методів голосування.

Майбутній розвиток може передбачати вирішення потенційних проблем із безпекою та конфіденційністю, а також подальшу оптимізацію продуктивності та масштабованості системи. Крім того, можливості та зручність використання системи можна покращити шляхом інтеграції передових технологій, таких як штучний інтелект і машинне навчання.

Запропонований підхід також можна розширити, щоб увімкнути інші процеси голосування, включаючи рейтингове голосування та голосування за довіреністю. Крім того, система може бути створена для включення таких атрибутів, як анонімність виборців і доступність для людей з обмеженими можливостями.

Загалом, запропонована система електронного голосування на основі блокчейну багатообіцяюча для багатьох напрямків, але потрібно буде її розширити можливості системи та вирішити будь-які потенційні проблеми.

Висновки

Резюме ключових висновків та внесків роботи

На завершення ця робота мала на меті створити прозору та безпечну систему голосування з використанням технології блокчейн. Голосування, додавання кандидатів і створення смарт-контрактів є одними з основних функцій системи, які гарантують неупередженість виборчого процесу. Під час тестування та аналізу ми виявили, що система має високу пропускну здатність, середню затримку та достатньо хорошу масштабованість, що дозволяє вибирати її для використання в реальних сценаріях голосування.

Порівняння запропонованої нами системи з поточними системами голосування показало, що вона має низку переваг перед традиційною системою голосування, включаючи підвищену прозорість, безпеку та ефективність. Окрім голосування, ця технологія також може використовуватися для керування ланцюгом постачання, перевірки особи та інших цілей.

Загалом у цій роботі використано технологію блокчейну для голосування, і вона має потенціал кардинально змінити спосіб проведення виборів у майбутньому.

Передумови для майбутніх досліджень і розробок

У підсумку, ця робота продемонструвала, що системи голосування на основі блокчейну мають потенціал для підвищення безпеки та підзвітності виборчих процесів. Це дослідження має важливі наслідки для поточних і майбутніх розробок і досліджень.

По-перше, запропонований метод може стати відправною точкою для додаткових досліджень систем голосування на основі блокчейну. Майбутні дослідження можуть бути зосереджені на покращенні зручності використання, масштабованості та продуктивності системи. Необхідно провести додаткові дослідження потенційних загроз і безпеки систем голосування, основаних на блокчейнах.

По-друге, використання смарт-контрактів запропонованою системою створює основу для автоматизації різних процедур голосування. Це чудова основа для створення більш складних програм для голосування, оскільки код системи можна розширити відповідно до складніших правил голосування.

Прозорість і незмінність запропонованого підходу можна також застосувати до інших сфер, де довіра і прозорість є важливими, наприклад, галузі охорони здоров'я та ланцюга поставок. Отже, додаткові дослідження щодо використання технології блокчейн у цих сферах можуть бути плідним напрямком для майбутніх досліджень.

Підсумовуючи, ця робота показала, що системи голосування, побудовані на технології блокчейн, можуть бути більш прозорими та безпечними. Запропонований підхід може стати відправною точкою для подальших досліджень та вивчення технології блокчейн та її потенційного використання поза голосуваннями.

Висновок і заключні спостереження

Запропонована система голосування на основі блокчейну пропонує безпечний, відкритий і непроникний метод проведення вільних і чесних виборів. Впровадження системи показує, що використання технології блокчейн для голосування є доцільним та ефективним. За допомогою аналізу продуктивності системи ми продемонстрували, що вона може підтримувати велику кількість транзакцій із прийнятною затримкою та високою пропускною здатністю.

Потенційні переваги запропонованої системи, такі як мінімізація потреби в централізованих органах влади, зниження ймовірності шахрайства та маніпуляцій, а також підвищення явки виборців, підкреслюються порівнянням з існуючими системами голосування. Однак, щоб бути прийнятою, така система має подолати низку перешкод у сферах технічного виконання, законодавчої та нормативної бази та суспільного визнання.

Підсумовуючи, запропонована система голосування на основі блокчейну має потенціал фундаментально змінити те, як ми проводимо демократичні вибори, пропонуючи безпечнішу, відкритішу та зручнішу заміну поточним процедурам голосування. Щоб вирішити труднощі та обмеження запропонованої системи та гарантувати її успішне застосування в реальних обставинах, необхідні додаткові дослідження та розробки.

Список використаної літератури:

1. *Crosby M.* Blockchain Technology: Beyond Bitcoin / Crosby, M., Pattanayak, P., Verma, S. [та ін.] – Applied Innovation – 2016. – Vol. 2. – P. 6-10.

2. *Nakamoto S.* Bitcoin: A Peer-to-Peer Electronic Cash System / Nakamoto S. - 2008. - 9 p.

3. *Johnson D.* Introduction to Distributed Consensus / Johnson D. – 2018. – 15 p.

4. *Eyal I.* Majority is not Enough: Bitcoin Mining is Vulnerable / Eyal I., Sirer E.G. – Financial Cryptography and Data Security. – 2014. – P. 436-454.

5. *Kshetri J.* Blockchain's roles in meeting key supply chain management objectives / Kshetri J. – International Journal of Information Management. – 2018. – Vol. 39. – P. 80-89.

6. *Yuan Y.* An Identity-Based Secure Electronic Voting System Using Blockchain Technology / Yuan Y., Wang J., Ji Z., Zhang Y. – Future Generation Computer Systems. – 2019. – Vol. 95. – P. 750-758.

7. *Hasan S.* Security Analysis of the Estonian Internet Voting System / Hasan S., Jurgena R., Bohme R., Breuker R. // Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. – Dallas – 2017. – P. 2001-2018.

8. *Antonopoulos A.* Mastering Bitcoin: Unlocking Digital Cryptocurrencies / Antonopoulos A. – O'Reilly Media, Inc. – 2014. – 298 p.

9. *German D.M.* Blockchain Basics: A Non-Technical Introduction in 25 Steps / German D.M. – Apress, 2018. – 212 p.

10. *Li S.* A Secure and Efficient Decentralized Electronic Voting System Based on Blockchain Technology / Li S., Li X., Li L. // IEEE Access. – 2019. – Vol. 7. – Pp. 167803-167817.

11. *Cachin C.* Blockchain Consensus Protocols in the Wild / Cachin C., Vukolic R. // IEEE Computer. – 2019. – Vol. 52, No. 9. – Pp. 98-102.

12. *Pal A.* DC-PoET: Proof-of-Elapsed-Time Consensus with Distributed Coordination for Blockchain Networks / Pal A. and Kant, K. // In Proceedings of the 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom) - 2018 - P. 801-808

13. *Wu X.* Research on ECDSA Algorithm and Its Applications / Wu X., Liu J., Li F., Zou H. // 2018 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC) - Hefei - 2018. – P. 604-607

14. *Fielding R.* Hypertext Transfer Protocol – HTTP/1.1 / Fielding R., J. Gettys, J. Mogul [та ін.] - IETF - 1999/ - RFC 2616.

15. *Xie, R.* A survey of blockchain technology: Architecture, consensus, and future trends / Xie, R., Zheng, Z., Liu, L. [та ін.] // Journal of Industrial Information Integration - 2019 - vol. 15 - P. 99-105.

16. *Antonopoulos A.* Mastering Ethereum: Building Smart Contracts and Dapps / Antonopoulos A. // O'Reilly Media - 2018 - 422 p.

17. *Van de Riet R.P.* The Accessibility of the Internet as a Voting Channel / Van de Riet R.P. // Electronic Voting in Europe Technology, Law, Politics and Society - 2016 - P. 13-29.