НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

«До захисту допущено»
В.о. завідувача кафедри

_____М.В.Грайворонський
(підпис)
“___”_____2019 р.

Дипломна робота

на здобуття ступеня бакалавра

з напряму підготовки 6.170101 «Безпека інформаційних і комунікаційних систем»

на тему: Двофакторна аутентифікація на основі Blockchain у Web-застосунках

Виконала:студентка _4_ курсу, групи ФБ-51
(шифр групи)

Барковська Вероніка Георгіївна _____ _____
(прізвище, ім’я, по батькові) (підпис)

Керівник к.т.н, доцент Коломицев М.В. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (посада, вчене звання,науковий ступінь, прізвище, ініціали) (підпис)

Рецензент провідний наук. співробітник, к.ф.м.н. Фаль О.М. _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____
(підпис)

Київ - 2019 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.170101 «Безпека інформаційних і комунікаційних систем»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ М.В.Грайворонський
(підпис)
«___»_____2019 р.

ЗАВДАННЯ
на дипломну роботу студенту

Барковській Вероніці Георгіївні
(прізвище, ім'я, по батькові)

1. Тема роботи Двофакторна аутентифікація на основі Blockchain у Web-застосунках,
науковий керівник роботи Коломицев Михайло Володимирович, кандидат технічних наук, доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від «_____» 2019 р. № _____

2. Термін подання студентом роботи 10 червня 2019 р.

3. Вихідні дані до роботи

Web-застосунок має такі функції:

1.Перший рівень аутентифікації виду логін/пароль;

2.Другий рівень аутентифікації:

А)Відправка даних до смарт-контракту;

Б)Перевірка даних смарт-контрактом;

3.Аутентифікація користувача.

4. Зміст роботи

1. Дослідити загрози безпеки та механізми захисту в розподілених системах;

2.Дослідити основні принципи роботи технології Blockchain;

3.Проаналізувати існуючі рішення використання технології Blockchain;

4.Розробити методику двофакторної аутентифікації на основі Blockchain у Web-застосунках.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

1. Мережа Blockchain;

2. Блок-схема реалізації системи двофакторної аутентифікації

на основі Blockchain;

3.Скріншоти роботи Web-застосунка.

6. Дата видачі завдання 01.02.2019

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів дипломної роботи	Примітка
1	Отримання завдання	01.02.2019	
2	Збір інформації	15.02.2019	
3	Дослідження предметної області на існуючих рішень	28.02.2019	
4	Розробка back-end частини	26.04.2019	
5	Розробка інтерфейсу	04.05.2019	
6	Тестування прототипу	20.05.2019	
7	Оформлення дипломної роботи	26.05.2019	
8	Отримання допуску до захисту	28.05.2019	

Студентка _____ Барковська В.Г.
(підпис) (ініціали, прізвище)

Науковий керівник роботи _____ Коломицев М.В.
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Метою дипломної роботи є дослідження існуючих методів аутентифікації та використання технології Blockchain в інформаційній безпеці. На сьогодні існує багато застосунків, що потребують захисту від несанкціонованого доступу та саме технологія Blockchain має властивості, що можуть допомогти створити надійний захист від несанкціонованого доступу. У роботі розглядаються існуючі рішення впровадження Blockchain у сферу інформаційної безпеки. Та описано методику двофакторної аутентифікації на основі Blockchain у Web-застосунках.

Загальний обсяг роботи: 61 сторінки, 11 ілюстрацій, 1 таблиця та 13 бібліографічних найменувань.

Ключові слова: Blockchain, Ethereum, Smart Contract, двофакторна аутентифікація.

ABSTRACT

The purpose of the thesis is based on existing authentication methods and used of Blockchain technology in information security. Today, there are many applications that need protection from unauthorized access, and Blockchain technology has properties that can help to provide reliable protection against unauthorized access. The article considers existing solutions to implement Blockchain in the field of information security. The two-factor authentication methodology based on Blockchain in Web applications is described.

The total amount of work: 61 pages, 11 illustrations, 1 table and 13 bibliographic titles.

Keywords: Blockchain, Ethereum, Smart Contract, Two-factor Authentication

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

RFID - Radio-Frequency Identification;

ACL - Access Control List;

CA - Certificate Authority;

SSO - Single Sign-On;

IP - Identity Provider;

SP - Service Provider;

SWT - Simple Web Token;

JWT - JSON Web Token;

SAML - Security Assertion Markup Language;

STS - Secure Token Service;

OAuth - Open Authorization;

PoW - Proof of Work;

PoS - Proof of Stack;

DPoS - Delegated Proof of Stack;

PBFT - Practical Byzantine Fault Tolerance;

DApp – Decentralized Applicattion;

PKI - Public key infrastructure;

DPKI - Decentralized public key infrastructure;

DNS - Domain Name System;

DDoS - Distributed denial-of-service;

EVM - Ethereum Virtual Machine;

JSON - JavaScript Object Notation;

ВСТУП

З кожним роком в світі з'являється все більше і більше сучасних технологій і росте величезна кількість нових Web-застосунків. І в більшості Web-застосунків потрібні методи аутентифікації від несанкціонованого доступу. Але більшість існуючих методів аутентифікації недостатньо, щоб повною мірою захистити облікові записи від несанкціонованого доступу зловмисниками. У кожного існуючого методу аутентифікації є ряд своїх недоліків. І для кращого захисту від атаки несанкціонованого доступу, була придумана двухфакторна аутентифікація.

І в зв'язку з революційним відкриттям такої технології як Blockchain, яка має всі властивості для кращого захисту Web-застосунків і всіляких транзакцій і даних від зловмисників. Стало можливо нове рішення для методів аутентифікації, використовуючи Blockchain.

Для захисту Web-застосунків та облікових записів від несанкціонованого доступу застосовують механізми аутентифікації. Але більшість відомих механізмів аутентифікації не гарантують повний захист від несанкціонованого доступу. Тому актуальним є дослідження методики аутентифікації, двофакторної аутентифікації на основі технології Blockchain.

Метою і завданням даної роботи є дослідження існуючих методів аутентифікації та використання технології Blockchain в інформаційній безпеці.

Об’єктом дослідження є технологія Blockchain, принципи її роботи, компоненти та зв’язок з інформаційною безпекою.

Предметом дослідження є надійність методів аутентифікації у Web-застосунках.

Методом дослідження є ознайомлення та опрацювання електронних джерел на різних мовах про загрози безпеки несанкціонованого доступу, методів аутентифікації та технології Blockchain .

Наукова новизна дослідження полягає у тому, що була запропонована методика аутентифікації на основі Blockchain.

Запропоноване рішення може використовуватися у реальних програмних рішеннях розробників Web-застосунків для забезпечення аутентифікації.

1 АНАЛІЗ ЗАГРОЗ БЕЗПЕКИ І МЕХАНІЗМІВ ЗАХИСТУ В РОЗПОДІЛЕНИХ СИСТЕМАХ

У цьому розділі буде описано можливі загрози безпеки, засновані на несанкціонованому доступі до системи, а так само представлені механізми аутентифікації в розподілених системах.

1.1 Переваги організації безпеки розподілених систем

Розподілені системи мають багато загроз для безпеки, але і розподілені системи мають свої переваги для підвищення безпеки системи. У розподіленої системі можуть бути різні вимоги безпеки. Одною з переваг розподілених систем є те, що вона не обмежує всю систему на один єдиний режим безпеки. Якщо середовище розподілене на окремі домени безпеки, кожен домен може мати різні аспекти політики безпеки. Загальний контроль досягається або узгодженням політик взаємодії безпеки між адміністраторами доменів, або ієрархічними структуруванням доменів, при цьому один менеджер бере на себе відповідальність за координацію взаємодії всіх.

1.2 Основи безпеки

Властивості безпеки в розподілених системах можна розділити на рівні, де високий рівень це захист активів компанії, а низький рівень надійність пароля облікового запису та ієрархічна структура між ними. Властивості нижчого рівня допомагають прийти до властивостей більш

високого рівня. Ці властивості можуть бути досягнуті за допомогою механізмів захисту на декількох різних архітектурних рівнях в розподіленоій системі. Комбінація властивостей безпеки і архітектурних рівнів, на яких вони можуть підтримуватися разом, утворюють основу для опису безпеки.

1.2.1 Властивості безпеки

Основні властивості відповідають таким загрозам, як розкриття інформації, корупція, втрата даних, відмова в обслуговуванні, відмова від прав. Вторинні властивості призводять до специфікації послуг для підтримки первинних. Існують три основні властивості безпеки, які застосовуються, як до збережених даних, так і до переданих даних:

Конфіденційність - збереження конфіденційності інформації, що зберігається в системах або передається між ними. Зазвичай це означає запобігання несанкціонованому доступу до збережених файлів даних і запобігання прослуховування повідомлень при передачі. Однак в застосунках з високим рівнем безпеки може також існувати вимога захисту від розкриття інформації, яка може бути виведена виключно з того факту, що дані передаються, а не з їх змісту. Ця інформація може бути отримана з аналізу трафіку, аналізу джерела, місця призначення та обсягу повідомлень. Класичний випадок аналізу трафіку - військовий, в якому підготовка до пересувань військ може бути виявлена завдяки збільшенню обсягу зв'язку між підрозділами.

Цілісність - підтримка цілісності даних, що зберігаються в системах або переданих між системами. Це запобігає втрати або зміни інформації через, наприклад, несанкціонований доступ, збоїв компонентів або помилок зв'язку. При передачі даних, також, може бути важливо, запобігти повторенню повідомлення. Наприклад, повідомлення в системі електронного

переказу коштів, що дозволяє переказ коштів з одного рахунку на інший, не повинно відправлятися і оброблятися двічі. Захист від цього ризику відомий як запобігання повторного відтворення. Цілісність може бути досягнута двома різними способами: або взагалі запобігти виникненню збоїв, або виявити його і усунути після нього. Профілактика може бути досягнута декількома способами; фізичним захистом, контролем доступу від несанкціонованих дій і процедурними заходами щодо запобігання помилок. Виявлення та відновлення вимагають своєчасності в поєднанні з засобами резервного копіювання, які дозволяють почати заново з ситуації відомої цілісності.

Доступність - підтримка доступності інформації, що зберігається в системах або передається між системами, що забезпечує доступність послуг, які забезпечують доступ до даних, і відсутність втрати даних. Загрози доступності можуть існувати на кількох рівнях. Файл даних недоступний для його користувача, якщо комп'ютер, який надає послугу, фізично знищений пожежею, або якщо файл був безповоротно видалений або стався збій зв'язку між користувачем і комп'ютером. Як і у випадку з цілісністю, доступні два різних режиму захисту: запобігання, виявлення і відновлення з використанням коштів резервного копіювання.

Спострежливість - підтримка спостережливості ресурсу, системи, яка дозволяє реєструвати всі дії об'єктів і суб'єктів однозначно встановлювати імена об'єктів / суб'єктів, причетних до певних дій, а також реагувати на всі дії з метою мінімізації втрат.

Але є так само і інша основна властивість безпеки, що відноситься конкретно до взаємодії між користувачами і / або програмами і що є однією з основних компонент вивчення в даній роботі:

Аутентифікація - аутентифікація особистості при комунікації партнерів і аутентифікація походження і цілісності даних, які передаються між ними. Це важливо для кількох цілей. Аутентифікація особистості відправника даних дає

впевненість в тому, що повідомлення є справжніми. Це також забезпечує основу для аудиту і бухгалтерського обліку. Ця вимога для систем контролю доступу, заснованих на ідентичності користувачів системи. Аутентифікація змісту повідомлення дозволяє виявляти порушення цілісності в повідомленнях.

Вторинні властивості, які визначаються архітектурою безпеки:

Контроль доступу - забезпечення контролю доступу до сервісів або їх компонентів, щоб користувачі могли отримувати доступ тільки до сервісів і даних, на які у них є права. Це одна з властивостей, яка використовується для досягнення конфіденційності, цілісності та доступності. Це може бути забезпечено фізичними та / або логічними механізмами. Несанкціонований доступ до персонального комп'ютера може бути відвернений блокуванням клавіатури. Доступ до спільно використовуваної системі може контролюватися системою логічного контролю доступу з використанням політики доступу, заснований на аутентифікації ідентичності користувачів.

Аудит журнал - надання дій аудит журналу в системі для забезпечення підзвітності користувачів. Журнал аудиту надає докази того, хто що зробив і коли.

Властивості безпеки, викладені вище, є взаємозалежними і не повинні розглядатися окремо. Аутентифікація є основою для досягнення багатьох інших цілей. Аутентифіковані, призначені для користувача, ідентифікатори необхідні для контролю доступу на основі ідентифікаційних даних, відстеження помилок і аудиту, але щоб ідентифікувати себе на основі паролю потрібно: як контроль доступу для захисту файлу паролів, так і конфіденційність на основі шифрування для подальшого захисту в разі збою контролю доступу. Контроль доступу, крім вимоги і підтримки аутентифікації, є основою для конфіденційності, цілісності та доступності. Контрольні журнали і аварійні сигнали безпеки підтримують інші властивості і залежать від них.

1.3 Загрози безпеки в розподілених системах

Всі загрози інформаційній безпеці можна класифікувати як порушення одної з 4 властивостей інформаційної безпеки, таких як конфіденційність, доступність, цілісність і спостережливість.

Існуючі розподілені системи дають зловмисникам значні можливості для здійснення загроз, що порушують властивості інформаційної безпеки. Навіть ті розподілені системи що призначені для бізнесу і мають малі ризики так само вимагають захисту. Зростаюче число взаємопов'язаних програм і технологій також призводить до збільшення числа вразливостей, які можна використовувати. Організації впроваджують кілька заходів безпеки для виявлення і усунення таких вразливостей і це нескінченна робота для фахівців інформаційної безпеки. Проте, найбільш вразливі місця можна визначити за пріоритетами, розсортувавши їх по потенційним загрозам, але для цього потрібен високий рівень практики аналізу загроз.

У роботі будуть розглянуті загрози спрямовані на несанкціонований доступ в розподілені системи, що порушує конфіденційність інформації. Це призводить до компрометації організації, втрати репутації у клієнтів і значної втрати прибутку в подальшому. Так само загроза несанкціонованого доступу в незахищені системи можуть призвести до того, що буде проводитися атака в інші незахищені системи. Існує прямий ризик розкриття конфіденційної інформації при несанкціонованому доступі до системи зловмисником.

Загрози несанкціонованого доступу можна класифікувати на прямий доступ і віддалений доступ. У разі прямого доступу зловмисник використовує програмні та програмно-апаратні засоби для отримання доступу в систему. При віддаленому доступі зловмисник використовує мережеві протоколи взаємодії. Загроза може бути реалізована як і на робочому місці без доступу в загальну мережу, так само як і в системах з підключенням до загальної мережі.

Зловмисників, які здійснюють несанкціонований доступ до системи можна категоризувати, як:

- зовнішній порушник;
- внутрішній порушник.

Зовнішні порушники - це зловмисники, що не мають доступу до системи і здійснюють несанкціонований доступ, поза контрольованої зони (КЗ) через зовнішню мережу. Внутрішні порушники мають доступ до системи і реалізують загрози, перебуваючи безпосередньо в системі.

До загроз, які надходять від зовнішніх зловмисників можна віднести:

- можливість здійснювати несанкціонований доступ через робочі місця, підключені до зовнішньої мережі;
- можливість здійснювати несанкціонований доступ через канали зв'язку, що знаходяться поза службових приміщень;
- можливість здійснювати несанкціонований доступ, скориставшись елементами інформаційної інфраструктури, що перебували поза КЗ під час ремонту, переміщення та утилізації.
- можливість здійснювати несанкціонований доступ через інформаційні системи організацій, які співпрацюють з обраним підприємством і підключаються до їх систем.

Можливості внутрішнього порушника залежать від режимних і організаційно-технічних методів задіяних на підприємстві.

Причинами виникнення вразливостей, що призводять до несанкціонованого доступу, можуть бути:

- помилки розробників при проектуванні програмного забезпечення;
- зловмисні дії по впровадженню вразливостей на етапі розробки програмного забезпечення;
- неправильні настройки програмного забезпечення;
- ненавмисне зміна режимів роботи пристроїв і програм;

- несанкціонована установка і використання нових програм з подальшою необґрунтованою витратою ресурсів;
- зловмисне впровадження шкідливих програм, що створюють відповідні уразливості;
- ненавмисне створення вразливостей в програмному забезпеченні;
- збої в роботі програмного і програмно-апаратного забезпечення;

1.3.1 Перспективи кіберзахисту

Після вивчення виникаючих загроз і деяких найбільш ефективних кібератак, важливо працювати над власним захистом. У цих груп загроз є все, що їм потрібно, щоб виявити активи організації, а потім знайти уразливості для створення своєї зброї відповідним чином. Це викликає величезну стурбованість щодо організацій, які неадаптивні, іноді більш десятиліть, але існує велика кількість організацій, які блискуче досягли кіберзахищенності.

1.4 Механізми безпеки

Для досягнення збереження властивостей безпеки використовуються наведені нижче механізми захисту:

- ✓ фізична і електронна безпека компонентів системи;
- ✓ механізми аутентифікації;
- ✓ механізми контролю доступу;
- ✓ механізми безпеки зв'язку

1.4.1 Аутентифікація як механізм захисту

Аутентифікація як механізм захисту використовується в сучасних системах як компонент підтвердження автентичності особистості. Для підтвердження автентичності особистості використовуються такі механізми, які ідентифікують особу:

- біометричний фактор аутентифікації;
- аутентифікація, тим, що володієш;
- аутентифікація, тим, що знаєш;

Біометричні характеристики людини - це, те що є унікальним для кожної людини, такі як відбиток пальця, відбиток сітківки, голос та інше. Але даний спосіб аутентифікації є не найкращим так як має свої недоліки. Якщо сканер, що проводить аналіз біометричних даних, зробити занадто точним, то при, навіть незначних, змінах структури біометричних характеристик, наприклад трохи пошкоджений відбиток пальця, то сканер не ідентифікує дані як істинні. Але знову ж таки якщо послабити точність сканера, то буде зворотна сторона проблеми і ідентифікуючий пристрій може пропустити людину зі схожими біометричними даними. І тоді про безпеку роботи механізму аутентифікації не може йтися розмова. Але даний вид аутентифікації використовуються найчастіше коли потрібна вища міра безпеки з неймовірною точністю ідентифікації особистості на секретних об'єктах і в інших важливих організаціях;

Підтвердження достовірності особи за допомогою чогось, чим володієш, в деяких джерелах її називають предметною аутентификацией. У цю категорію можна віднести програмно-апаратні пристрою системи аутентифікації. Сюди можна включити різні пристрої введення / виведення ідентифікаційних даних та інші пристрої ідентифікації, наприклад різні зчитувачі, смарт карти тощо.

Пристрої ідентифікації зберігають і обробляють конфіденційні унікальні ідентифікаційні дані людини, що в подальшому використовується для аутентифікації, у вигляді цифрового коду. Пристрої вводу / виводу використовуються для взаємодії між ідентифікаторами та системою проведення контролю доступу. Пристрої введення / виведення системи аутентифікації і ідентифікації можна класифікувати на деякі категорії, такі як:

- ✓ смарт-карти;
- ✓ інформаційна «таблетка» або по-іншому iButton;
- ✓ USB-ключі;
- ✓ RFID-ідентифікатори (radio-frequency identification) - радіочастотні ідентифікатори;
- ✓ безконтактні.

Безконтактні ідентифікатори не вимагають прямого контакту між ідентифікатором і пристроєм вводу / виводу на відміну від контактних. Для коректної роботи безконтактного ідентифікатора, для читання і запису даних, досить піднести ідентифікатор на деяку відстань від пристрою введення / виведення.

Якщо аналізувати безпеку даного механізму захисту, то варто враховувати такі фактори як стійкість ідентифікатора до різних механічних і хімічних чинників, впливаючих на нього. До таких факторів можна віднести: вологу, прямий фізичний вплив з важкими предметами, падінням з висоти, вплив температури і різка її зміна, а також не менш важливо це можливості ідентифікатора протистоянню до атак спрямованих на розкриття та вилучення чіпа з даними з нього.

Спосіб аутентифікації заснований на принципі: «те, що ти знаєш». Це найпопулярніший і часто використовуваний механізм аутентифікації на сьогоднішній день. У соціальних мережах, хмарах, інших Web-застосунках і сайтах найчастіше можна зустріти саме цей вид аутентифікації. Він являє собою введення пароля або іншої інформації, яку знаєш тільки ти. У теорії цей

механізм вважається одним з найпростіших і надійних, через криптостійкість. Але на практиці не завжди це твердження вірно. Це пояснюється тим, що для надійності пароля для підтвердження автентичності особистості в ідеалі повинні використовуватися складні і довгі паролі, довжиною не менше 8 символів, змінюватися паролі з періодичністю хоча-б раз на півроку. Але терміни і довжину пароля для кожної системи адміністратор безпеки може прописати сам в політиці безпеки. Проте, більшість людей не дотримуються рекомендацій при створенні пароля і подальшій зміні його для безпеки системи. Людський фактор відіграє не малу роль, адже запам'ятовувати паролі 8-12 символів, причому ще і кожні півроку, наприклад, є важливим аспектом при створенні паролів. Так само варто зауважити і те, що людям легше запам'ятати паролі, які складаються з слів, що мають деяку асоціацію в їх пам'яті. Це так само не додає надійності паролям. Адже в наш час величезна кількість зловмисників можуть підібрати пароль простим перебором по словнику. Це те що стосується механізму аутентифікації, такого як введення пароля, але є ще інший вид механізму аутентифікації, який найчастіше використовують якщо був забутий пароль, це секретне питання. Ще при реєстрації людина вибирає питання, на яке, в разі втрати пароля, можна дати відповідь, відповідь вказується при реєстрації. І в разі коректної відповіді на секретне питання, система дає можливість відновити пароль, найчастіше створивши новий. Але є недолік в механізмі з секретним питанням і відповіддю на нього, у такого механізму криптостійкость ще менше ніж у звичайного пароля. Адже більшість секретних питань, це те на що може відповісти не тільки власник пароля, а й люди, що добре знають власника ідентифікаційних даних.

1.4.2 Контроль логічного доступу

Логічний контроль доступу використовується найчастіше в багатокористувацьких системах, де фізичний контроль доступу не представляється можливим. Модель для логічного контролю доступу надається контрольним монітором, який перехоплює всі спроби доступу і дозволяє їх, тільки якщо доступ дозволений. Існують дві основні форми логічного контролю доступу:

- мандатний контроль доступу, що заснований на певній політиці;
- дискреційний контроль доступу, що заснований на політиці доступу, що призначається кожному об'єкту окремо.

Для дискреційного контролю доступу зазвичай використовують механізм аутентифікації. Користувачі автентифіковані при вході в систему і контрольний монітор дає доступ чи ні, виходячи з політики доступу.

Існує дві основні реалізації правил доступу:

- Access Control List (ACL) (Список контролю доступу) прикріплюється до об'єктів, визначаючи користувачів, яким дозволений доступ до них і операції можливі для даного користувача;
- Користувачі отримують аутентифіковані можливості, які діють як доступ до певних ресурсів.

Багато персональних обчислювальних систем надають доступ на основі паролів. Так як це простий і примітивний спосіб захисту в системах, де низький рівень безпеки прийнятний.

1.5 Механізм захисту аутентифікації у Web- застосунках

Таблиця 1.1 – Методи аутентифікації у Web-застосунках

Спосіб	Основне застосування	Протоколи
За паролем	Аутентифікація користувачів	HTTP, Forms
За сертифікатами	Аутентифікація користувачів в безпечних застосунках; аутентифікація сервісів	SSL / TLS
За одноразовими паролями	Додаткова аутентифікація користувачів	Forms
За ключами доступу	Аутентифікація сервісів і застосунків	-
За токенами	Делегована аутентифікація користувачів; делегована авторизація додатків	SAML, WS-Federation, OAuth, OpenID Connect

1.5.1 Аутентифікація за паролем

Це метод аутентифікації, що базується на логіні та паролі, що відноситься до класифікації «те, що ти знаєш» наведенної вище. Якщо розглядати цей метод аутентифікації у Web-застосунках, то існує декілька стандартних протоколів аутентифікації за паролем:

- HTTP аутентифікація;
- Forms аутентифікація.

HTTP - це протокол, що описаний у стандартах HTTP 1.0/1.1. У Web-сайтах його роботу можно описати так:

1. Коли неавторизований користувач надсилає запит до захищеного ресурсу, сервер надсилає HTTP статус “401 Unauthorized” і додає заголовок “WWW-Authenticate” та вказує параметри аутентифікації.

2. Після цього браузер відкриває автоматично вікно або сторінку з вводом логіну та пароля.

3. У подальших запитах до цього Web-сайту браузер автоматично додає HTTP заголовок “Authorization” в якому знаходяться дані користувача, що його аутентифікують.

4. Сервер аутентифікує користувача по даних цього заголовку. Рішення про доступ до ресурсу надається окремо контролюючим логічним доступом, аналізуючи роль користувача.

При використанні HTTP-аутентифікації у користувача немає стандартної можливості вийти з Web-застосунку, крім як закрити всі вікна браузера.

Розглянемо декілька схем аутентифікації на базі HTTP-аутентифікації:

1.Basic authentication- механізм аутентифікації, оснований на логіні/паролі та визначений у стандарті HTTP / 1.0.

«Веб-сервер запитує у веб-клієнта аутентифікацію користувача. Як частина запиту, веб-сервер передає область (рядок), в якій користувач повинен пройти аутентифікацію. Рядок області базової перевірки автентичності не повинна відображати будь-який конкретний домен політики безпеки. Веб- клієнт отримує ім'я користувача і пароль від користувача і передає їх на веб- сервер. Потім веб-сервер аутентифікує користувача в зазначеній галузі…»[12]

Логін та пароль передаються у незахищенному вигляді у заголовку Authorization, використовуючи просте кодування base64 і тому данний метод є небезпечним, але при використані безпечного транспортного механізму HTTPS та додаткової мережевої безпеки, наприклад протокол IPSEC або VPN, більш безпечно.

2. Digest authentication – більш безпечна схема аутентифікації на основі логіну та паролю, при якому сервер надсилає значення nonce, завдяки якому здійснюється хешування пароля браузером. Є більш безпечною альтернативою Basic authentication при незахищенному з'єднанні, але має свої недоліки:

- можливість атаки «людина посередині» з подальшою заміною на Basic authentication;
- непідтримка сучасних хеш-функцій для збереження паролів на сервері;

Forms не має стандарту на якому він базується. Працює даний протокол по принципу: у Web-застосунку з'являється HTML-форма, у якій користувач має ввести свої логін та пароль і відправити їх на сервер через HTTP POST для здійснення аутентифікації. Далі, якщо аутентифікація була пройдена Web-застосунок створює токен сессії, який зберігається у cookies браузера. Токен сессії буде передаватися автоматично на сервер при кожному наступному запиті. Токен сессії може бути створений двома шляхами, як ідентифікатор аутентифікованої сессії користувача або як зашифрований об'єкт. Якщо зловмисник перехопить токен сессії, то буде мати доступ до облікового запису користувача. Тому протокол forms повинен виконуватися по захищенному каналу HTTPS.

1.5.2 Загрози безпеки для методу аутентифікації за паролем

Найбільш частіше зустрічаються такі вразливості у Web-застосунках:

- можливість підбору пароля (брут-форс);
- можливість створення легких паролів;

- можливість використання пароля, що був створенний застосунком, без вимоги зміни на новий;
- можливість передачі аутентифікаційних данних по незахищенному каналу;
- можливість використання небезпечної хеш-функції;
- неможливість зміни пароля;
- можливість несанкціонованого доступу до облікового запису через небезпечний метод відновлення пароля;
- можливість використання сессії користувача, через відсутність виходу користувача з аутентифікованої сессії.

1.5.3 Аутентифікація за сертифікатами

«Сертифікат являє собою набір атрибутів, що ідентифікують власника, підписаний certificate authority (CA). CA виступає в ролі посередника, який гарантує справжність сертифікатів. Також сертифікат криптографічно пов'язаний з закритим ключем, який зберігається у власника сертифіката і дозволяє однозначно підтвердити факт володіння сертифікатом…».[7]

Сертифікат може зберігатися у операціонній системі, в браузері або на окремому фізичному пристрої, такому як смарт картка або токен USB. А закритий ключ захищається, зазвичай, ще паролем.

Для аутетифікації використовують сертифікати стандартів X.509. Завдяки сертифікату аутентифікація відбувається у момент з’єднання з сервером і є частиною протоколу SSL/TLS.

«Після успішної аутентифікації Web-застосунок може виконати авторизацію запиту на підставі таких даних сертифіката, як subject (ім'я власника), issuer (емітент), serial number (серійний номер сертифіката) або thumbprint (відбиток відкритого ключа сертифіката)…».[7]

Цей спосіб є більш надійним ніж аутентифікації за паролем, але в зв’язку з важкістю розповсюдження сертифікатів, цей метод аутентифікації є менш популярним.

1.5.4 Аутентифікація за одноразовими паролями

Цей метод аутентифікації використовується, як другий рівень аутентифікації для двофакторної аутентифікації. Цей метод базується на принципі, що користувач для аутентифікації спочатку використовує аутентифікацію за паролем, а потім аутентифікацію за тим, що він має, наприклад фізичний пристрій для генерації одноразового ключа.

Другий спосіб для використання одноразових паролів є використання їх для підтвердження особистості при виконанні якихось важливих дій, наприклад при покупці.

Для створення одноразових паролів використовують різні программні та програмно-апаратні рішення. До них відносяться апаратні та програмні токени, що генерують одноразові паролі на основі секретного ключа введеного до них, наприклад апаратний засіб- RSA SecurID, а програмний - Google Authenticator. Також рандомно генеруючі паролі, що передаються через канал зв’язку, наприклад SMS або телефонний дзвінок.

Зазвичай такий метод аутентифікації використовують, як доповнення до аутентифікації forms. Після створення токену сессії, користувач не буде мати доступ до Web-застосунку поки не виконає аутентифікацію за одноразовим паролем.

Але цей метод аутентифікації є досить ненадійним через можливість сніффінгу, тобто перехоплення мережевих пакетів з одноразовим паролем, або якшо це апаратне рішення, то поблизу може не бути зчитувача.

1.5.5 Аутентифікація за ключами доступу

Цей метод аутентифікації використовується для застосунків та сервісів при їх звернені до Web-сервісів. Для аутентифікаціїї в даному методі використовюються ключі доступу, наприклад access key, API key. Ключі доступу це довгі унікальні строки, що являють собою рандомний набір символів.

Зазвичай користувачі, які хочуть отримати доступ до Web-сервісу роблять запит на створення ключа доступу і в подальшому зберігають його у клієнтському застосунку. Цей ключ може давати не повний доступ до ресурсу після аутентифікації, це може бути задано при створенні ключа доступу.

Так, як ключі доступу являються рандомно підібраними символами, його складніше буде підібрати, на відміну від звичайного пароля. І у випадку компрометації ключа, його можна аннолювати і створити новий.

Для захисту від перехоплення ключів зазвичай з'єднання з сервером захищене протоколом SSL/TLS. Але нема єдиного протоколу для цього методу. Ключі можуть бути передані у різних частинах HTTP-запиту.

Але буває, що аутентифікація відбувається по незахищенному каналу, тоді для аутентифікації використовують публічну та секретну частини ключа доступу. Публічна частина використовується для ідентифікації користувача, а секретна використовується для підпису. Наприклад сервер може надіслати якейсь значення, а користувач повертає хеш цього значення, використовуючи секретну частину ключа, що аутентифікує користувача.

1.5.6 Аутентифікація за токенами

Даний метод аутентифікації використовується зазвичай для розподілених систем Single Sign-On (SSO), де аутентифікація відбувається за рахунок іншого сервісу, наприклад здійснення аутентифікації через обліковий запис соціальних мереж. Соціальні мережі виступають в ролі сервіса аутентифікації.

Ідея цього методу заключається в тому, що identity provider (IP), наприклад соціальні мережі, надають аутентифікаційні дані у вигляді токену до service provider (SP), що являється нашим застосунком, в якому здійснюється аутентифікація. І застосунок використовує даний токен для аутентифікації і авторизації користувача.

В загальному вигляді процесс аутентифікації для активного клієнта, наприклад IOS або Android, які можуть виконувати запрограмовану послідовність дій, виглядає так: клієнт аутентифікується у IP способом який цей застосунок підтримує, далі клієнт просить IP надати йому токен для аутентифікації для SP- застосунку. IP створює токен та надсилає його клієнту і клієнт аутентифікується, завдяки токену.

А для браузера, що являється пасивним клієнтом, тобто може лише відображати сторінки, на які користувач здійснив запит. Процесс аутентифікаціїї виконується автоматично, переправлючи браузер між IP та SP.

Токен –це структура даних, що складається з інформації: срок дії токена , відправник, можливий отримувач токену та деяка інформація о користувачі, що здійснив запит на створення токену. Токен для збереження цілістності даних і захисту від несанкціонованого змінення даних додатково підписується.

Для підтвердження аутентифікації SP-застосунок спочатку здійснює перевірки для токену. Серед яких перевірка IP-застосунка, який видав токен, перевірка можливого отримувача токену, срок дії та цілісність токену. Після успішної перевірки SP-застосунок аутентифікує користувача.

Одними з найрозповсюджених форматів токенів є:

- Simple Web Token (SWT);

- JSON Web Token (JWT);
- Security Assertion Markup Language (SAML);

SWT є одним з наспростіших форматів, що складається з Issuer, Audience, ExpiresOn и HMACSHA256 та їх значень у форматі кодування HTML form. Токен підписується симметричним ключем і обидва застосунки IP та SP повинні мати цей ключ, щоб створити та перевірити токен відповідно.

JWT складається з трьох блоків, що розділяються точками. Блоки представляють собою заголовок, набір полей та підпис. Перші два блока записуються у форматі JSON і кодуються у формат base64 додатково. Набір полей зазвичай має значення таких параметрів, як з Issuer, Audience, ExpiresOn та інших. Підпис генерується завдяки симетричної криптографії та асиметричної.

SAML складається з інформації о емітенті, суб'єкті, що потрібна для створення та перевірки токену. Інформація записана у XML-форматі. Підпис здійснюється завдяки асиметричній криптографії. У даному токені є особливість, що підтверджує власника токену і допомагає уникнути атаки «людини по середині» у незахищенних каналах.

Для даного методу аутентифікації використовуються стандарти, що описують протокол взаємодії між клієнтами та IP і SP застосунками, також, як і формат надсилаємих токенів. До таких стандартів належать:

- стандарт SAML;
- стандарт WS-Trust;
- стандарт WS-Federation;
- стандарт OAuth;
- стандарт OpenID Connect.

Стандарт SAML складається з assertions, protocols, bindings, profiles, де assertions формат токенів стандарту SAML у форматі XML, protocols являється

набором повідомлень, наприклад запит на створення нового токену, отримання існуючих токенів, вихід з системи, управління ідентифікаторами користувачів тощо, bindings – це механізми передачі повідомлень через транспортні протоколи, profiles – типові сценарії стандарту, що визначають набір перших трьох пунктів assertions, protocols та bindings.

Стандарт WS-Trust описує інтерфейс сервісу авторизації Secure Token Service (STS). Стандарт працює по протоколу SOAP і підтримує створення ті анулювання токенів та використовує зазвичай SAML-токены.

Стандарт WS-Federation відноситься до механізму обміну токенами. При цьому WS-Federation розширює функції сервіса STS. Він вирішує ті ж самі задачі, що й SAML, але має інші підходи і реалізацію.

Стандарт OAuth (Open Authorization) на відміну від інших стандартів не описує протокол аутентифікації користувача. Він визначає механізм отримання доступа одного застосунка до іншого від імені користувача. Роботу стандарта можна описати так:

1.Користувач дає згоду застосунку на доступ до певного ресурсу у виді гранту.

2.Застосунок отримує токен доступу до ресурсу від сервера авторизації в обмін на свій грант.

3.Застосунок використовує токен для отримання даних від сервера ресурсів.

Гранти бувають чотирьох видів:

- Authorization Code – цей грант надається після аутентифікації і підтвердження про надання доступу до ресурсу.
- Implicit – використовується при неможливості безпечно отримати токен від серверу авторизації, при такому розкладі грант є токеном отриманим від серверу авторизації без другого кроку описаного вище.

- Resource Owner Password Credentials – грант являється логіном і паролем користувача, шо використовується, якщо застосунок являється інтерфейсом для сервера ресурсів.
- Client Credentials – грант використовується при відсутності користувача і застосунок отримує доступ завдяки ключів доступу і тоді перший крок виключається з процессу описаного вище.

Стандарт OpenID Connect розробленний як доповнення до облікових даних OAuth. Сервер авторизації надає додатковий токен формату JWT ідентифікації на другому кроці.

1.5.7 Двофакторна аутентифікація

У зв'язку з перерахованими недоліками простих методів аутентифікації у першому розділі, людство знайшло рішення в більш надійному механізмі захисту системи, в двофакторній аутентифікації.

Двофакторна аутентифікація - це механізм аутентифікації, заснований на двох різних методах аутентифікації для більш ефективного захисту системи від несанкціонованого входу. Двофакторна аутентифікація не забезпечує повний захист від крадіжки аккаунта, але дає більш надійний захист, ніж аутентифікація тільки одним типом аутентифікації. Недоліки одного типу аутентифікації, в цьому випадку можуть бути нівельовані перевагами іншого типу, що значно збільшує надійність даного методу захисту облікового запису від зловмисників.

У двофакторній аутентифікації перший рівень аутентифікації - це метод аутентифікації з паролем та логіном, а для другого рівня аутентифікації частина інформації надається з центрального сховища. Це центральне сховище відповідає за зберігання всієї інформації, необхідної для автентифікації користувача. Хоча двофакторна аутентифікація підвищує рівень безпеки з

другим рівнем аутентифікації, він все ще стикається з недоліком, що централізована база даних зберігає список секретної інформації про користувача. Центральна база даних може бути підроблена або пошкоджена атаками зловмисника, і це може призвести до масових порушень даних.

У розділі вище було розказано про аутентифікацію за токенами, що активно використовується у двофакторній аутентифікаціїі у даній дипломній роботі буде досліджена нова модель двофакторної аутентифікації на основі технології, що має децентралізовану природу і знижує вразливість, яка притаманна централізованим застосункам.

Висновок до розділу 1

Кожен метод аутентифікації має свої переваги і недоліки. На сьогоднішній день для більш ефективного захисту систем використовують комбінацію механізмів захисту, що знижує ризик з можливих недоліків і підвищує рівень безпеки системи, так званий двофакторна аутентификація. Так само в останні роки набирає популярності технологія Blockchain, що стала революційним проривом в світі технологій. І далі в роботі, я збираюся дослідити як технологія Blockchain може допомогти поліпшити існуючі механізми захисту аутентифікації.

2 АНАЛІЗ ВИКОРИСТАННЯ ТЕХНОЛОГІЇ BLOCKCHAIN В ІНФОРМАЦІЙНІЙ БЕЗПЕЦІ

В даному розділі буде розглядатися технологія Blockchain і її використання в інформаційній безпеці, які рішення вже існують і в цілому буде ознайомлення з технологією Blockchain для більш докладного її розуміння.

2.1 Технологія Blockchain

2.1.1 Що таке Blockchain?

На сьогоднішній день не рідкість, коли зловмисники отримують облікові записи користувачів і використовують їх у своїх цілях. І не завжди користувач, чий обліковий запис був вкрадений, може відразу виявити даний факт. Через цю обставину, не дивно, що безліч людей стурбовані конфіденційністю їх даних і є всі причини не довіряти таких систем. Всі існуючі профілактичні засоби захисту, не можуть гарантувати запобігання подальших інцидентів. Але, якщо вчасно виявити інцидент, це допоможе запобігти несанкціонованому використанню облікових записів зловмисником.

Ось якраз нова революційна технологія Blockchain допомагає вирішити цю проблему. Дана технологія забезпечує можливість перегляду, хто і що змінює в акаунті і з'являється можливість відстежити, що дані про людину не використовуються не за призначенням і забезпечує повну прозорість діяльності. Технологію Blockchain можна охарактеризувати по суті як розумну, безпечну і ту, що постійно поповнюється, базу даних. Дана технологія це хронологічний регістр, який зберігає транзакції будь-якої суми і величини між незалежними сторонами. Спочатку Blockchain передбачався, як технологія для забезпечення

можливості проводити фінансові транзакції без участі третьої сторони, такої як банки. Однак пізніше деякі галузі змогли впровадити цю технологію для інших цілей, про що буде більш докладно розказано в розділі нижче.

2.1.2 Основи технології Blockchain

Blockchain є децентралізованою базою даних, що зберігає транзакції в безпеці і тільки в блоках, що об'єднуються в подальшому в ланцюжок блоків. Популярність в інших галузях технологія набула, як раз завдяки своїй децентралізованої природі. Для організацій для яких єдина точка відмови неприйнятна, технологія бази даних з ланцюжком блоків є відмінним рішенням. Крім цього Blockchain можуть управляти не тільки розробники і адміністратори, а й довірені особи.

На малюнку нижче представлена схема мережі Blockchain:

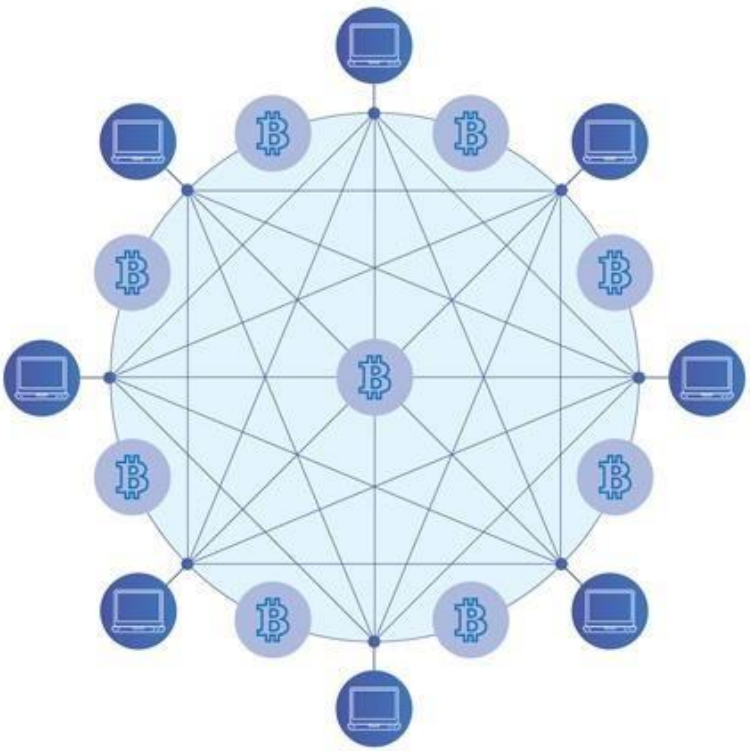

Рисунок 2.1 – Мережа Blockchain

На кожному комп'ютері, що має доступ до Інтернету, має бути встановлено програмне забезпечення, необхідне для роботи з вузлами Blockchain і застосунок, специфічні для середовища технології. Деякі комп'ютери можуть не бути задіяні, в залежності від різних випадків. Наприклад, заснований на Blockchain банківський ланцюг дозволяє банкам запускати тільки клієнтську програму вузла банківського ланцюга.

2.1.3 Принцип роботи технології Blockchain

Щоб зрозуміти принцип роботи Blockchain в простій формі, важливо розібратися в наступних етапах технології:

1. Підготовка транзакції
2. Перевірка транзакції
3. Генерація блоку
4. Перевірка блоку
5. Ланцюжок блоків

Підготовка транзакції включає в себе створення транзакції, що буде складатися з таких компонентів: відкрита адреса одержувача, цифровий підпис відправника і повідомлення транзакції. Далі ця транзакція буде доступна для всіх вузлів ланцюга.

Перевірка транзакції заснована на принципі, де вузли, тобто комп'ютери, на яких встановлено програмне забезпечення Blockchain, не довіряють один одному і ці вузли перевіряють транзакції на достовірність, шляхом перевірки цифрового підпису через відкритий ключ відправника. Після перевірки автентичності відправника, транзакція поміщається в чергу для додавання в блок і чекає, поки всі вузли перевірять транзакцію.

Генерація блоку здійснюється шляхом створення блоку з перевірених транзакцій, що знаходяться в черзі. Блок створюється одним з вузлів мережі Blockchain.

Перевірка блоку відбувається після успішної його генерації та вузли мережі обробляються на подальший ітеративний процес перевірки для досягнення консенсусу між більшістю вузлів.

На даний момент одні з популярних методів перевірки на досягнення консенсусу є: Proof of Work (доказ роботи) (PoW), Proof of Stack (доказ стека) (PoS), Delegated Proof of Stack (делеговане підтвердження стека) (DPoS) і Practical Byzantine Fault Tolerance (практична візантійська стійкість до збоїв) (PBFT). Цей механізм впливає на фінансові аспекти і забезпечує безпеку всіх транзакцій.

Ланцюжок блоків створюється після механізму консенсусу, перевірені блоки додаються в ланцюжок блоків.

У момент підключення вузла до мережі Blockchain, в нього завантажується оновлена база даних, що дійсна на даний момент. Кожен вузол веде реєстр, організований у вигляді блоків і зав'язаний на алгоритмі хешування.

Розглянемо більше блок. Блок складається з заголовка блоку і тіла блоку. Заголовок блоку допомагає ідентифікувати блок серед інших блоків. Заголовок містить в собі версію програмного забезпечення та протоколів, мітка часу, що вказує на час створення блоку в секундах, хеш попереднього блоку в ланцюжку, nonce (тимчасова змінна), що використовується для відстеження лічильника алгоритму консенсусу. Тіло блоку складається зі списку транзакцій, де кожна транзакція підписана цифровим підписом відправника.

2.2 Технологія Blockchain в Інформаційній Безпеці

Як і було сказано на початку розділу, технологія Blockchain своїми перевагами перевернула і інші сфери діяльності. В даному випадку нас цікавить вплив Blockchain на інформаційну безпеку. Загалом Blockchain можна охарактеризувати такими властивостями, як децентралізація, прозорість і незмінність. Що дає відмінні можливості для впровадження технології в сферу інформаційної безпеки. Децентралізація знижує можливість підробки бази даних Blockchain. Зазвичай зловмисники використовують мейнфрейм, де зібрані всі дані для отримання інформації. У Blockchain таке неможливо, так як всі дані лежать розподілено по всій мережі, і це означає, що зловмисники не будуть мати одну ціль для атаки. У разі технології Blockchain порушники повинні будуть змінити дані в усій мережі однаково в кожному вузлі, у всіх блоках. Ще потрібно врахувати, що будь-яка зміна в базі буде відмічена іншими вузлами і має бути погоджена з більшістю вузлів, тому така атака б вимагала величезних обчислювальних потужностей. Так само існують, як публічні, так і приватні Blockchain мережі, вміст мережі видно або всім або колу осіб приватної мережі відповідно. У публічний мережах, те що вміст є надбанням громадськості, є безперечною перевагою. Так як будь-який зміни буде видно, будь-кому хто зверне на це увагу. З цього випливає, що володіючи сотнями спостерігачами, зміни в публічній мережі не можуть пройти непоміченими. Так само ще однією перевагою яку дає властивості мереж Blockchain, пов'язана з хешем, адже кожен блок хешується і його хеш закладений в наступний блок, що унеможливлює зміну або видалення блоку, без подальших за ним змін всіх блоків в ланцюжку.

2.3 Blockchain і його зв'язок з основними властивостями безпеки

2.3.1 Конфіденційність в існуючій моделі Blockchain

Конфіденційність в існуючій моделі Blockchain полягає в тому, що учасники транзакції можуть залишатися анонімними, здійснюючи транзакції. Особливо важлива конфіденційність в бізнес сфері, адже дуже важлива довіра між компанією і клієнтами та іншими зацікавленими особами. Коли йдеться про взаємодію двох бізнес сторін, важливо вирішити якою кількістю інформації поділитися, хто буде мати доступ до неї і на яких умовах. У Blockchain при кожній транзакції важливо знати, в якому обсязі одержувач буде володіти інформацією, а саме це можна прописати в смарт контракті (докладніше про смарт-контракті в розділі 3).

Щоб докладніше розібратися в конфіденційності технології Blockchain потрібно розглянути, які дані зберігає в собі сервіс, при реєстрації і використанні технології Blockchain. Інформацію можна розділити на два типи: інформацію, що користувач може надати сам і інформацію, що про користувача збирає система при реєстрації і використанні. Інформацію, що користувач може надати сам це те, що користувач заповнює форми в додатку або веб-сайті, листуючись з службою підтримки по телефону, електронній пошті або іншим способом. Далі інформація, яку сервіс збирає про користувачів:

- Інформація для входу: при вході в сервіс проводиться збір інформації про тип браузера і його версії, час останнього доступу до кабінету Blockchain, IP-адреса, що була використана при створенні кабінету (гаманця) і сама остання IP-адреса, що була використана при вході.
- Інформація про пристрій: це інформація про пристрої, які користувач використовує для доступу до кабінету, сюди

відноситься інформація про моделі пристрою, операційну систему і версію пристрою і його унікальні ідентифікатори, але ця інформація є анонімною і не прив'язана до якоїсь конкретної особи.

- Інформація про кабінет (гаманець / обліковий запис): При реєстрації облікового запису створюється пара відкритого і закритого ключів, а при виході з кабінету історія транзакцій і ця пара ключів буде зашифрована і сервіс не має доступу до не зашифрованого закритого ключа в будь-якому випадку.

2.3.2 Цілісність в існуючій моделі Blockchain

На сьогоднішній день, багато користувачів не бажають використовувати хмарні сховища. Зазвичай до даних в хмарі застосовується шифрування, що забезпечує тільки конфіденційність даних, але не гарантує їх цілісність. Шифрування не може захистити від атак, що можуть пошкодити захищені дані. Незважаючи на те, що в технології Blockchain є рішення для досягнення цілісності даних, шляхом хешування блоків. Однією з ключових характеристик даного хешування є те, що функція завжди одностороння. Це призводить до того, що неможливо отримати дані назад з хешу або хеш-суми. Невелика зміна у вихідних даних спричинить за собою повну зміну хешу. Як вже було сказано вище, кожен вузол зберігає регістр у вигляді ланцюжка блоків, де кожен блок пов'язаний хешем попереднього блоку. Це ускладнює зловмисникам порушити цілісність даних в Blockchain. Це допомагає різним сферам діяльності підвищити свій рівень цілісності безпеки і на основі технології Blockchain побудувати новий застосунок, захищений від несанкціонованого доступу.

2.3.3 Доступність в існуючій моделі Blockchain

Програма є доступною через мережі і являють собою програмних код, який щось означає, поки до нього маєш доступ. Blockchain працює на програмному забезпеченні, яке може бути корисно в тому випадку якщо користувачі мають до нього доступ і поки воно функціонує коректно. Для того щоб децентралізований застосунок (dApp) був доступний, сервер і інтерфейс Blockchain повинні працювати без збоїв. На практиці кібератаки, спрямовані на відмову в обслуговуванні, наприклад DDoS, роблять веб-сайти недоступними для користувачів, що призводить в результаті до великих збитків. Децентралізована природа Blockchain ускладнює виконання атак. Через відсутність єдиної точки відмови, навіть якщо один вузол вийде з ладу, інформація може бути доступна з інших вузлів і для інших користувачів. І так як вузли при вході в систему автоматично оновлюють точну копію регістра, то інформація завжди залишається актуальною.

2.4 Використання Blockchain у інформаційній безпеці

2.4.1 Аутентифікація на основі PKI з Blockchain

Public key infrastructure (PKI) (Інфраструктура відкритого ключа) - це відкритий фреймворк, створений для вирішення питань довіри між користувачами, підключеними до Інтернету.

До PKI належать компоненти, що забезпечують три основних властивості безпеки : конфіденційність, цілістність та доступність. До них належать:

- шифрування асиметричного ключа;
- сертифікат - це електронний ідентифікатор, який представляє ідентифікацію користувача або пристрою, зацікавленого у спілкуванні через мережу і гарантує, що до мережі підключений законний користувач;
- certificate authority (CA) є довіреною третьою стороною, яка засвідчує, що користувачі, сервери, бази даних і адміністратори є тими, кого вони вважають. CA перевіряє облікові дані користувачів і надає сертифікат, підписуючи його секретним ключем;
- реєстраційний орган підтримує інформацію про місцеві реєстраційні дані та ініціює процес оновлення та скасування старих сертифікатів;
- репозиторій сертифікатів - це база даних сертифікатів, доступна всім вузлам середовища PKI;

Існуючі проблеми у сучасній системі PKI:

1.Потреба в додатковій безпеці. Якщо центральний репозиторій сертифікатів буде скомпрометований, це призведе до масових порушень даних і крадіжки рахунку.

2. Централізованість.

PKI має велику уразливість через централізовану систему управління. Blockchain, однак, принципово децентралізований і дозволяє здійснювати зв'язок між кількома сторонами без участі третьої сторони. Decentralized public key infrastructure (DPKI) (Децентралізована інфраструктура відкритих ключів) - це інноваційна концепція, яка створює системи аутентифікації над загальнодоступними системами, не залежачи від однієї третьої сторони, яка може поставити під загрозу цілісність і безпеку системи. PKI реалізується як функція в смарт контракті в Ethereum.

Наступні набори операцій PKI надаються шляхом програмування смарт-контракту:

Реєстрація суб'єкта-власника: Користувачі або системи додаються до системи PKI, викликаючи подію реєстрації зі смарт-контракту. Сутність може бути такою ж простою, як адреса Ethereum, відкритий ключ, ідентифікатор атрибутів, хеші даних і дані. Конфігурована подія на смарт контракті збирає об'єкт і пересилає його як транзакцію в Ethereum.

Підписання атрибутів: Сутність може бути охарактеризована за допомогою реєстраційної події. Кожен атрибут суб'єкта може бути підписаний системою PKI через смарт контракт, і транзакція буде видана. Цей підписаний об'єкт пізніше буде доступний іншим особам або користувачам.

Отримання атрибутів: Атрибути об'єктів можуть бути розміщені шляхом застосування фільтра до блочного ланцюга з використанням відповідних ідентифікаторів подій, які були налаштовані на смарт контракт.

Відкликати підпис: це одна з найбільш критичних функцій, необхідних для будь-якого рішення PKI: відкликання цифрового підпису на атрибути або сутності. Анулювання стає надзвичайно важливим, коли користувач втрачає свій ключ або він скомпрометований. Смарт контракти можуть бути налаштовані на виклик події скасування та скасування підпису на конкретному об'єкті.

2.4.2 Платформа безпеки DNS, заснована на Blockchain

Domain Name System (DNS) (Система доменних імен) призначена для отримання IP-адреси по імені хоста. У найпростішій формі існує три основних компоненти DNS:

- простір імен;
- сервер імен;
- резольвер.

Простір імен - це структура бази даних DNS. Вона представлена у вигляді інвертованого дерева з кореневим вузлом у верхній частині. Кожен вузол дерева має мітку, а кореневий вузол має нульову мітку.

Сервери відповідають за збереження інформації про простір імен у вигляді зон.

Резольвер (розпізнавач) імен допомагає серверу імен знаходити дані в просторі імен. DNS вразлива і часто буває об'єктом атак.

DNSChain є одним з популярніших проектів для удосконалення DNS і захисту від спуфінга. DNSChain - це програмний набір на основі блочного каналу DNS, який замінює інфраструктуру відкритого ключа X.509 (PKI) і надає MITM докази автентифікації. Це дозволяє користувачам Інтернету встановлювати загальнодоступний сервер DNSChain для запитів DNS і отримувати доступ до сервера з доменами, що закінчуються на .bit. X.509 - це стандартна структура, що визначає формат PKI для ідентифікації користувачів і об'єктів через Інтернет. Це допомагає користувачам інтернету знати, чи є з'єднання з певним веб-сайтом безпечним чи ні. DNSChain має можливість надавати масштабовану та децентралізовану заміну, яка не залежить від третіх сторін.

2.4.3 Розгортання DDoS-захисту на основі Blockchain

Distributed denial-of-service (DDoS) (розподілена відмова в обслуговуванні) – це атака направлена на відмову в обслуговувані системи. Цей вид атаки здійснюється за допомогою величезної кількості ботів. Які надсилають величезну кількість запитів до системи, де система не витримує і припиняє свою роботу. Щоб захистити мережі від DDoS-атак, організації можуть бути розподілені між декількома вузлами сервера, які забезпечують високу стійкість і видаляють єдину точку відмови. Існує дві основні переваги

використання Blockchain: технологія Blockchain може бути використана для розгортання децентралізованої системи для зберігання чорних списків. Технологія Blockchain усуває ризик виникнення однієї точки відмови.

Висновок до розділу 2

У даному розділі було розглянуто можливості технології Blockchain в інформаційній безпеці і існуючі рішення використання Blockchain у інформаційній безпеці. Було досліджено, як завдяки технології Blockchain можна вирішити деякі питання інформаційної безпеки, такі як аутентифікація на основі PKI, захищенність DNS та можливість захисту від DDoS атак, вкиористовуючи технологію Blockchain. Також було введення до самої технології Blockchain, його принцип роботи та компоненти з яких вона складається. Після дослідження властивостей Blockchain та їх ролі у інформаційній безпеці, можна дійти до висновку, що технологію Blockchain можна використати для аутентифікації користувачів у Web-застосунках.

3 ДВОФАКТОРНА АУТЕНТИФІКАЦІЯ НА ОСНОВІ BLOCKCHAIN У WEB-ЗАСТОСУНКАХ

3.1 Ethereum

Ethereum є відкритою платформою на основі технології Blockchain, що була створена для розробників з метою створення ними децентралізованих застосунків. Ідея створення таких застосунків полягає в тому, щоб не було залежності від єдиної точки відмови для зберігання особистих та корпоративних даних. У класичних системах баз даних користувач не знає про те, як дані зберігаються, яка політика безпеки розроблена для захисту даних тощо. Але в децентралізованих застосунках користувач за всім цим може спостерігати і контролювати.

«З Ethereum будь-які централізовані послуги можуть бути перетворені в децентралізовані служби з його унікальними можливостями програмування…»[1].

Криптовалюта яку використовує Ethereum називається ефір, що є аналогом біткойнів у платформі Bitcoin і потрібна для здійснення транзакцій у мережі Ethereum. Ethereum використовує PoS для досягнення консенсусу у створенні блоку:

«PoS: Це метод для досягнення консенсусу в блокових ланцюгах серед вузлів і для перевірки транзакцій. На відміну від PoW, з PoS, генератор блоків не буде вибиратися на основі його поточного багатства. Блоки ніколи не будуть винагороджені в цьому механізмі, а майнер в PoS називається валідатором. Ethereum використовує PoS, і метою вибору цього було уникнути будь-яких

впливів на навколишнє середовище, які приходять з величезним обсягом споживання електроенергії…За допомогою механізму PoS вузли повинні приєднатися до пулу валідаторів, який буде обраний як підробник. Каспер, консенсусний PoS протокол Ethereum, працює як гібридна версія з існуючим механізмом PoW…»[1]

Так як Ethereum це платформа на основі Blockchain, то вона має ті ж перевагами, що і Blockchain:

- незмінність, що забезпечується хешем блоків і не дає іншим користувачам вносити несанкціоновані зміни;
- немає єдиної точки відмови, застосунки завжди будуть доступні, якщо не з одного комп'ютера, так з іншого;
- захист від корупції, адже для досягнення консенсусу для проведення транзакцій, потрібна велика частина вузлів;

Є й недоліки в децентралізованих застосунках. Наприклад, як це було зі створенням Ethereum Classic, через неправильно написаного смарт-контракту, була проведена крадіжка величезної кількості ефірів у користувачів і було рішення відкотити систему до атаки, що суперечило принципу Blockchain і думки розділилися і був створений Ethereum Classic, для тих хто був проти відкату, а для інших був проведений відкат системи і з тих пір існують дві паралельні мережі.

3.1.1 Smart Contract

Smart contract (смарт-контракт) – це електроний контракт, що являється комп’ютерним алгоритмом, написаним для передачі даних, криптовалюти та виконання всіх вимог контракту між двома сторонами. Смарт-контракт слідкує за виконанням вимог обох сторін, що були запрограмовані у контракті та автоматично штрафує сторони при невиконанні вимог. Ці контракти безпечні,

бо немає неоднозначної трактовки вимог, а тільки чітко прописаний комп'ютерний алгоритм, оснований на криптографії. На відміну від звичайних контрактів, для цього виду контракту не потрібна третя сторона для складання та виконання вимог контракту. Смарт-контракт оснований на технології Blockchain. Є багато платформ для работи з смарт-контрактами, але в даній роботі буде використана платформа Ethereum, що є кращим варіантом для роботи з смарт-контрактами, оскільки забезпечує значні можливості розробки. Код смарт-контракту виконується у момент коли приходить транзакція або повідомлення. Це можна зробити, відправивши транзакцію, або через інший контракт, що відправив повідомлення.

Перевагами смарт контрактів є:

- прозорість;

Смарт контракт є повніст'ю прозорим для всіх зацікавлених сторін, усі вимоги контракту повині бути оговорені до його активації.

- однозначність;

Всі вимоги чітко прописані у алгоритмі контракту і не може бути ніяких непорозумінь і недомовок між сторонами.

- ефективність роботи;

Через відсутність посередника та точності, швидкості та автоматизовоності функцій, використання смарт контрактів є ефективним.

- збереження екології;

Через електронну природу смарт контрактів не потрібно оформлювати використовувати папір для контракту, що сприяє збереженню навколишнього середовища.

- резервне копіювання;

Смарт контракт зберігається не в одному екземплярі і тому на відміну від банків немає можливості загубити контракт.

- надійність;

Смарт контракт ніколи не загубить дані, так як вони зберігаються у зашифрованому вигляді у загальному розподіленому реєстрі;

- безпечність;

Не може бути несанкціоновано змінений після його активації.

Але смарт контракти мають ряд своїх недоліків:

- конфіденційність;

Не завжди прозорість є перевагою, іноді у взаємодії між двома сторонами важлива конфіденційність, тоді прозорість Smart contract стає недоліком.

- помилки;

Можуть бути помилки при написані коду.

- гроші;

Для створення специфічного смарт контракту потрібен спеціаліст та потрібно буде оплатити вартість його послуг і для здійснення транзакцій потрібно сплачувати криптовалютою платформи Blockchain.

Для написання смарт-контракту частіше використовується мова програмування Solidity, що має схожий синтаксис з Java-Script мовою програмування.

Смарт-контракти в Ethereum мають вигляд класів і компілюються в байт код для віртуальної машини Ethereum (Ethereum Virtual Machine, EVM), а після відправлються у Blockchain.

EVM підтримують цикли, тому платформа використовує механізм, який називається газом, для обмеження контрактів, які можуть оброблятися надто довго.

У мережі Ethereum є 2 види облікових записів:

- зовнішні облікові записи;

- облікові записи смарт-контрактів.

Зовнішній обліковий запис контролюється за допомогою пари закритих ключів, які зберігються людиною або зовнішніми серверами. Ці акаунти не можуть містити код EVM. Зовнішній обліковий запис має таки характеристики:

- містить баланс ефіру;
- можливість відправляти транзакції;
- управляється парами закритих ключів;
- не пов'язаний з кодом;
- база даних ключ / значення, що міститься в кожному обліковому записі, де ключі і значення є 32-байтними рядками;
- адреса визначається з відкритого ключа;
- обробляються EVM;

Облікові записи смарт-контрактів контролюються кодом. Вони зберігають алгоритми і активуються зовнішніми обліковими записами або іншими обліковими записами смарт-контрактів. Мають такі характеристкии:

- містить баланс ефіру;
- зберігають певний код смарт-контракту в пам'яті;
- активізуються зовнішніми обліковими записами або іншими смарт-контрактами, що відправляють транзакцію;
- можливість виконання складних операцій;
- можливість викликати інші контракти;
- не мають власника після включення в EVM;
- обробляються EVM;
- база даних ключ / значення, що міститься в кожному обліковому записі, де ключі і значення є 32-байтними рядками.

Адреса смарт-контракту визначається на момент створення контракту, створюється з адреси розробника цього смарт-контракту і кількості транзакцій, відправлених з цієї адреси.

3.2 Двофакторна аутентифікації на основі Blockchain

Для створення нової системи двофактроної аутентифікації буде використана платформа Blockchain Ethereum для роботи з смарт-контрактами. Смарт-контракт буде зберігати дані облікового запису користувача для другого рівня аутентифікації. Щоб надіслати дані смарт-контракту користувач повинен мати обліковий запис Ethereum з яким буде пов’язаний обліковий запис Web-застосунка і мати доступ до закритого ключа облікового запису Ethereum.

Для проходження двофакторної аутентифікації потрібно:

- обліковий запис Ethereum, що має свою адресу;
- обліковий запис у Web-застосунку, що зберігає в собі такі дані: логін, пароль та адресу облікового запису Ethereum;
- смарт-контракт, що має функцію аутентифікації authenticate();
- мати ефір, криптовалюту Ethereum, для здійснення транзакції – відправки даних до смарт-контракту;
- плагін Web-браузера, який підключає ваш пристрій до мережі Ethereum;

Web-застосунок використовує бібліотеку Web3 JS, яка надає простий програмний інтерфейс застосунка для роботи зі смарт-контрактами. Браузер надає користувачеві цю бібліотеку, наприклад доповненням до браузеру. До таких розширень можна віднести плагіни web-браузеру Metamask, Mist, Parity.

Є важливий об’єкт даних у Web3 JS, це webeth, що використовується для взаємодії з Blockchain. Його можна порівняти з JSON, у тому як передаються дані між фронтендом та бекендом застосунка на основі Ethereum. Він називається об'єктом JSON-RPC і поставляється з бібліотекою Web3 JS.

Мережа Internet працює за рахунок пересилання HTTP-запитів між пристоями користувачів та Web-серверами. Ethereum в свою чергу не залежить від деяких певних Web-серверів, користувач з'єднується з найближчим вузлом мережі Ethereum, який транслює запит користувача на мережу Ethereum.

Повний вузол Ethereum займає більше 400 Гбайт на пристрої користувача. Раніше для роботи з мережею Ethereum потрібно було зберігати всю базу данних вузла на жорсткому диску. А зараз такі розширення браузеру, як Metamask та Parity дозволяють підключатися до вузла мережі Ethereum на пряму, де плагіни виступають у якості посередника. Функціонал плагіну забезпечується бібліотекою JavaScript Web3 JS та розширенням для браузеру.

Для розробки методики ауетентифікації на основі Blockchain, використовувався плагін Metamask.

Окрім головної мережі Ethereum у Metamask є ще тестові мережі Ropsten Network, Kovan Network та Rinkeby Network, а також можна задати свою приватну мережу Blockchain. Для розробки методики аутентифікації була застосована тестова приватна мережа Ganache Cli, що надає 10 облікових записів Ethereum з відповідними до них секретними ключами, де кожен обліковий запис має на рахунку 100 ефірів, що потім можна підключити до плагіна Metamask з подальшою роботою зі смарт-контрактами.

В наведенній нижче блок-схемі представлений процес реалізації двофакторної аутентифікації у Web-застосунку:

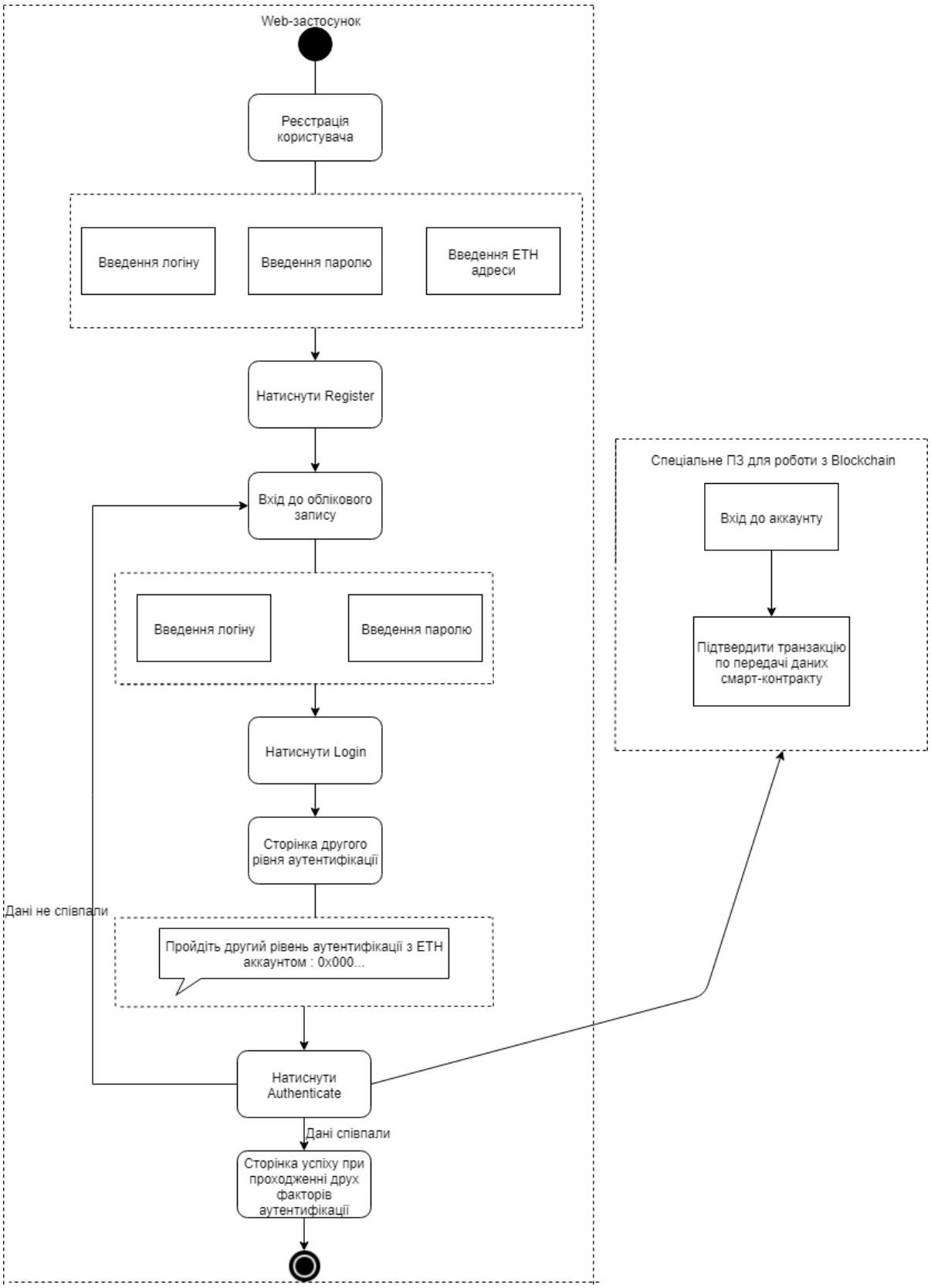

Рисунок 3.1 – Блок-схема реалізації системи двофакторної аутентифікації на основі Blockchain

При вході у Web-застосунок користувач потрапляє до сторінки реєстрації, де вводить дані для нового облікового запису, які складаються з логіну, паролю та адреси облікового запису Ethereum. Після, користувач натискає кнопку Register і в системі з’являється новий обліковий запис. Після цього користувач потрапляє на сторінку для входу до ресурсу. Перед користувачем з’являється два поля для введення даних логіна та пароля. Користувач вводить дані та натискає кнопку Login. Після цього користувач переправляється на сторінку для проходження другого рівня аутентифікації, где пишеться: «Authenticate with your ETH account with:» і вказується адреса Ethereum, яка була вказана користувачем на період реєстрації.

Користувач для здійснення другого рівня аутентифікації переходить до сайту чи плагіну для браузера платформи Ethereum і здійснює вхід до свого облікового запису. При натисненні кнопки Autheticate запускається плагін платформи Ethereum і просить підтвердити проведення транзакції. Транзакція представляє собою смарт-контракт з функцією authenticate(), що перевіряє співпадіння токенів аутентифікації, тобто чи дійсно відправлений токен аутентифікації, а в даному випадку адреса облікового запису Ethereum, що була вказана при реєстрації, співпадає з адресою з якої була здійснена транзакція. Якщо токени аутентифікації співпадають, Web-застосунок переходить до сторінки успіху, що означає проходження другого рівня аутентифікації.

3.2.1 Практична демонстрація моделі системи двофакторної аутентифікації на основі Blockchain у Web-застосунках

Перший етап у системі двофакторної аутентифікації на основі Blockchain, як і було описано вище, це здійснення реєстрації користувача у Web- застосунку. На цій сторінці є три поля для введення даних та одна кнопка Register. Поля з даними складаються з:

- поле логіну;
- поле паролю;
- адреса облікового запису Ethereum;

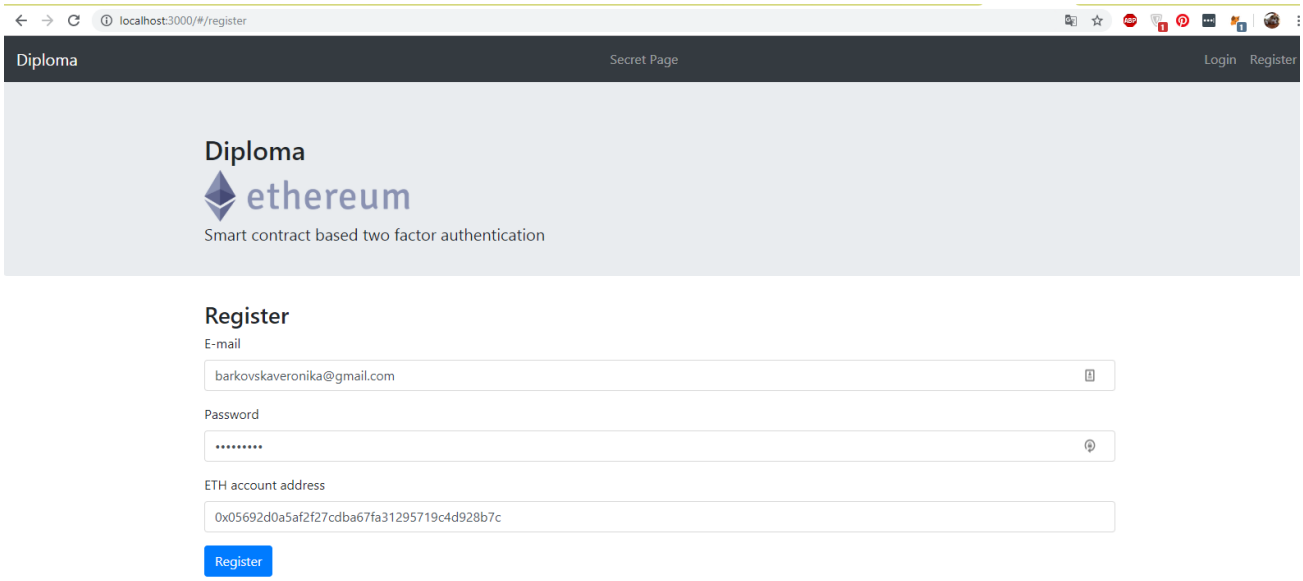

Рисунок 3.2 – Скріншот сторінки реєстрації двофакторної аутентифікації на основі Blockchain

Після реєстрації користувач переходить до сторінки входу. На сторінці входу знаходиться два поля для введення даних та одна кнопка Login. Дані, які потрібно ввести: логін та пароль.

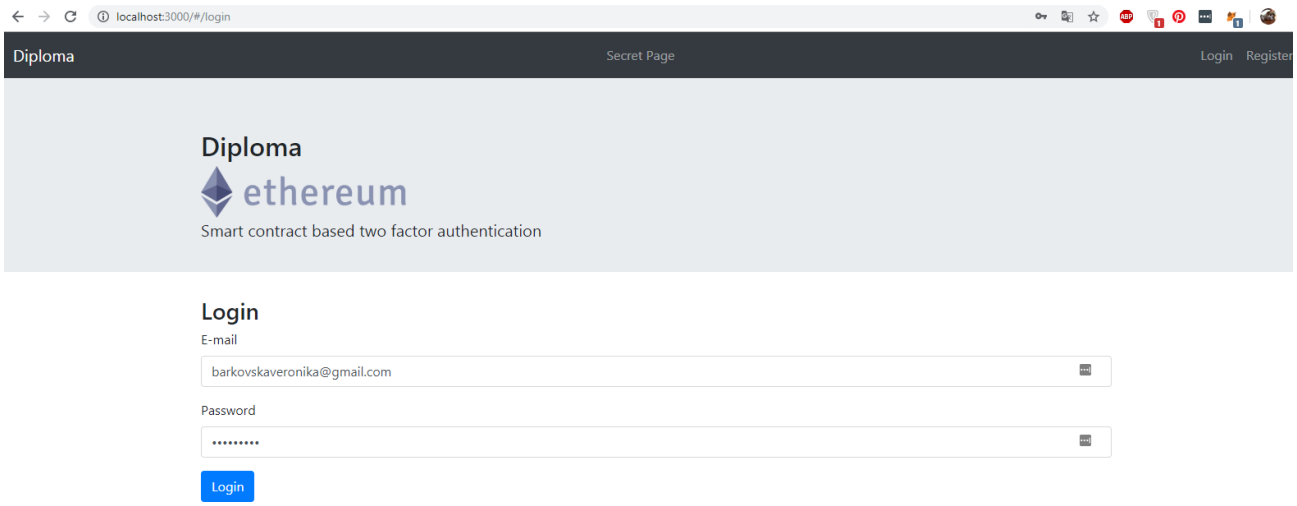

Рисунок 3.3 - Скріншот сторінки входу двофакторної аутентифікації на основі Blockchain

Після натиснення на кнопку Login користувач переходить до сторінки другого рівня аутентифікації, де написано: «Authenticate with your ETH account with» і пишеться адреса облікового запису Ethereum, яка була вказана при реєстрації і кнопка Authenticate.

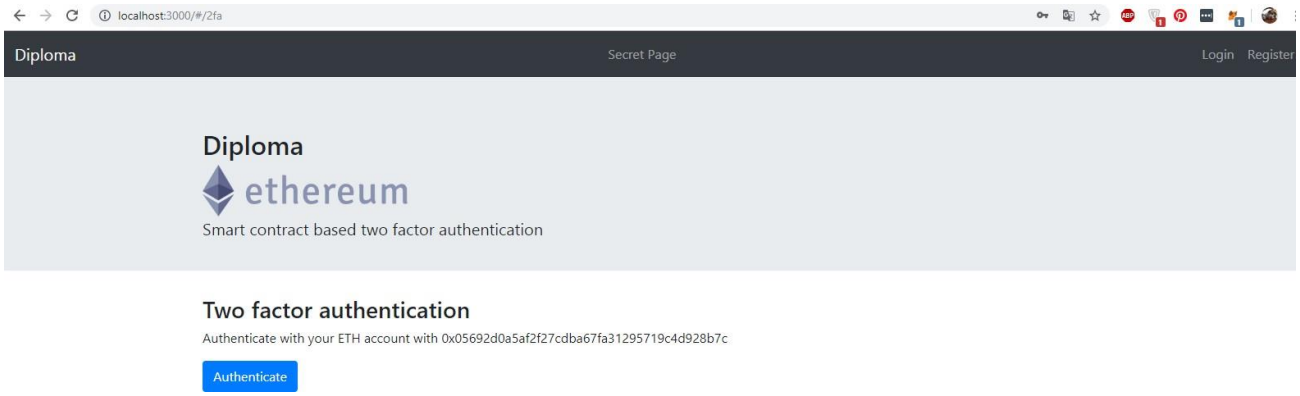

Рисунок 3.4 - Скріншот сторінки двофакторної аутентифікації на основі Blockchain

Перед наступним етапом користувач повинен увійти до свого облікового запису Ethereum, в моєму випадку я використовувала плагін під мій браузер Metamask для роботи з платформою Ethereum. Після того, як користувач увійде до свого облікового запису, він натискає Authenticate і у користувача вспливає вікно плагіну Ethereum, в якому пропонується підтвердити транзакцію. Під транзакцією мається на увазі відправлення даних з Web-застосунка до смарт-контракту. У самому смарт-контракті є функція authenticate(), що перевіряє токен аутентифікації, тобто адресу облікового запису Ethereum з якого здійснюється транзакція та адреса, яка була вказана при реєстрації користувача.

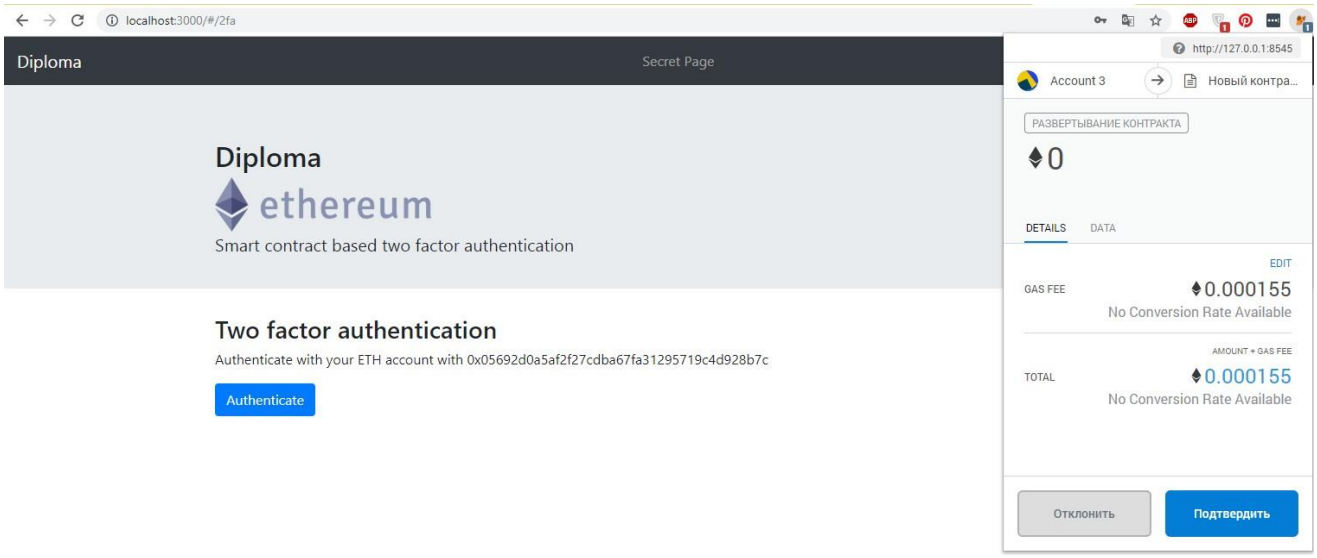

Рисунок 3.5 - Скріншот вікна плагіну для підствердження транзакції

При успішній перевірці і порівнянні токенів аутентифікації, користувач переходить до сторінки успіху Secret Page, що свідчить про те, що другий рівень аутентифікації був пройдений успішно.

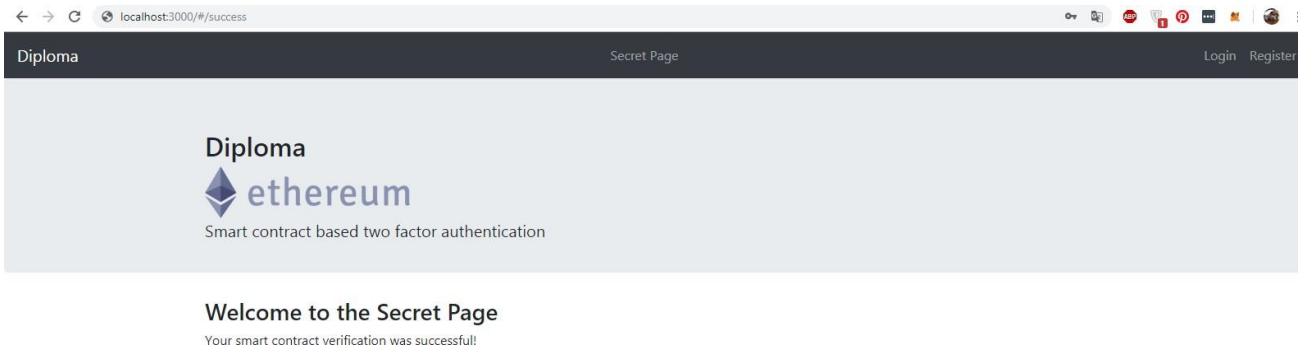

Рисунок 3.6 – Скріншот сторінки secret page, при проходженні двофакторної аутентифікації

Висновок до розділу 3

У даному розділі були досліджені платформа технології Blockchain – Ethereum, що має всі можливості і переваги для роботи з смарт-контрактами. Також були розглянуті смарт-контракти, їх переваги та недоліки. Далі у роботі описується методика двофакторної аутентифікації на основі Blockchain у Web-застосунках. Детально описується, що потрібно для двофактоної аутентифікації, демонструється блок-схема дій при двофакторній аутентифікації і демонструється практична реалізація методики аутентифікації з детальним описом дій. Завдяки цій методиці користувачі, можуть залишатися максимально анонімними, при максимальному захисті від несанкціонованого доступу. Так, як частіше всього для другого рівня аутентифікації використовують облікові записи соціальних мереж, де зберігаються персональні дані про користувача і не кожен користувач хоче довірити свої персональні дані Web-застосунку. Також може статися, що обліковий запис соціальних мереж може бути заблокований та стає неможливим пройти другий рівень аутентифікації. Використання Ethereum і смарт-контрактів для аутентифікації допоможе уникнути таких ризиків.

ВИСНОВКИ

У першому розділі були досліджені загрози несанкціонованого доступу та механізми аутентифікації в роздподілених системах у більш широкій класифікації і більш поглибленно про механізми аутентифікації у Web-застосунках:

- аутентифікація за паролем;
- аутентифікація за сертифікатами;
- аутентифікація за однаразовими паролями;
- аутентифікація за ключами доступу;
- аутентифікація за токенами.

Також було розглянуто переваги розподілених систем, основи безпеки та двофакторну аутентифікацію.

У другому розділі було досліджено технологію Blockchain та її використання у інформаційній безпеці. Була розглянута загальна інформація про технологію Blockchain, її принцип роботи та зв’язок з основними властивосятми інформаційної безпеки. Також було досліджено існуючі рішення використання технології Blockchain у інформаційній безпеці.

Третій розділ розкриває саму ідею методики двофакторної аутентифікації на основі Blockchain у Web-застосунках. Спочатку досліджується платформа Blockchain Ethereum, її особливості та смарт контракти , їх принцип роботи , переваги та недоліки. А потім описується нова модель двофакторної аутентифікації на основі Blockchain у Web-застосунках з зображенням блок-схеми дій користувача та практичною демонстрацією роботи програми у скріншотах.

Нова модель двофакторної аутентифікації дозволить різним сферам діяльності, що будуть створювати застосунки для своєї сфери, бути спокійними

за свою безпеку від несанкціонованого доступу зловмисниками. Другий фактор аутентифікації дозволить користувачам Ethereum чи іншої платформи Blockchain, що підтримує смарт-контракти здійснювати аутентифікацію швидко та зручно, не задіючи свої соціальні мережі з персональними даними для другого рівня аутентифікації, або використовувати номер телефону для відправки SMS-коду. Ця модель двофакторної аутентифікації дозволяє залишатися максимально анонімним у Web-застосунку, будучи максимально захищенним від несанкціонованого доступу до облікового запису.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЬ

1. Gupta R. Hands-On Cybersecurity with Blockchain [Електронний ресурс] / Rajneesh Gupta. – 2018. – Режим доступу до ресурсу: https://subscription.packtpub.com/book/networking_and_servers/9781788990189.
2. Moffett J. Security & Distributed Systems [Електронний ресурс] / Jonathan Moffett – Режим доступу до ресурсу: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.22.2312&rep=rep1&type=pdf.
3. Угрозы несанкционированного доступа [Електронний ресурс]. – 2008. – Режим доступу до ресурсу: https://zakonbase.ru/content/part/1183301.
4. Panda Security. Серьезное влияние блокчейна на информационную безопасность [Електронний ресурс] / Panda Security. – 2019. – Режим доступу до ресурсу: https://www.cloudav.ru/mediacenter/security/blockchain-profound-effect-cybersecurity/.
5. Blockchain Privacy Policy [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://www.blockchain.com/ru/legal/privacy.
6. Malan. Классификация механизмов аутентификации пользователей и их обзор [Електронний ресурс] / Malan. – 2013. – Режим доступу до ресурсу: https://habr.com/ru/post/177551/.
7. Выростков Д. Обзор способов и протоколов аутентификации в веб-приложениях [Електронний ресурс] / Дмитро Выростков. – 2015. – Режим доступу до ресурсу: https://habr.com/ru/company/dataart/blog/262817/.
8. Арянова Т. Ethereum для начинающих: Полное руководство [Електронний ресурс] / Тая Арянова. – 2017. – Режим доступу до ресурсу: https://ru.ihodl.com/tutorials/2017-06-30/ethereum-dlya-nachinayushih-polnoe-rukovodstvo/.

9. Nitish Singh. Смарт-контракты: основное пособие для начинающих [Електронний ресурс] / Nitish Singh. – 2018. – Режим доступу до ресурсу: https://101blockchains.com/ru/%D1%81%D0%BC%D0%B0%D1%80%D1%82-%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%B0%D0%BA%D1%82%D1%8B-%D0%B1%D0%BB%D0%BE%D0%BA%D1%87%D0%B5%D0%B9%D0%BD/#7.
10. Что такое смарт-контракты простым языком [Електронний ресурс] – Режим доступу до ресурсу: https://prostocoin.com/blog/smart-contract.
11. Арянова Т. Смарт-контракты для чайников [Електронний ресурс] / Тая Арянова. – 2017. – Режим доступу до ресурсу: 11. https://ru.ihodl.com/tutorials/2017-11-09/smart-kontrakty-dlya-chajnikov/.
12. Web Application Security [Електронний ресурс] – Режим доступу до ресурсу: http://java.boot.by/wcd-guide/ch05s03.html.
13. adminCraft. Введение в Ethereum и Solidity [Електронний ресурс] / adminCraft. – 2018. – Режим доступу до ресурсу: https://craftappmobile.com/vvedenie-v-%E2%80%8B%E2%80%8Bethereum%E2%80%8B%E2%80%8B-i-s%E2%80%8Bolidity/.

ДОДАТКИ

Додаток А

```
pragma solidity ^0.4.4;

contract Diploma {

   mapping(address => uint) authentications;

    function authenticate(uint authenticationToken) {
    require(authenticationToken > 0);
    address prover = msg.sender;
    authentications[prover] = authenticationToken;
    }

  function getAuthenticationToken(address prover) constant returns (uint) {
    return authentications[prover];
  }
}
```

Рисунок А.1 – Код смарт-контракту з функцією authenticate()

```
myApp.service('Diploma', ['ethClinet', '$q', function (ethClinet, $q) {
  const diplomContract = ethClinet.eth.contract(contractABI).at(contractAddress);

  let account = null;
  $q((resolve, reject) => ethClinet.eth.getAccounts(function (e, r) {
    if (e) {
      reject(e);
    } else {
      account = r[0];
    }
  }));

  const callback = (resolve, reject) => {
    return (e, r) => {
      if (e) {
        reject(e);
      } else {
        resolve(r);
      }
    }
  };

  this.authenticate = (token) => $q((resolve, reject) => {
    diplomContract.authenticate(token, {
      from: account,
      value: 0
    }, callback(resolve, reject));
  });
}]);
```

Рисунок А.2 – Код застосунка з підключенням до смарт-контракту

```
myApp.factory('AuthService', ['$http', '$location', function ($http, $location) {
  const authService = {};
  let loggedUser = null;
  authService.userLoaded = false;
  authService.loadUser = () => $http.get('user').then(response => {
    console.log(response);
    loggedUser = response.data.user;
    authService.userLoaded = true;
    return loggedUser;
  });
  authService.verify = () => $http.get('verify').then(() => loggedUser.tokenVerification = true);
  authService.register = (userData) => $http.post('register', angular.toJson(userData));
  authService.login = (userData) => $http.post('login', angular.toJson(userData)).then((response) => {
    loggedUser = response.data.user;
    return loggedUser;
  });
  authService.getLoggedUser = () => loggedUser;
  authService.isAuthenticated = () => loggedUser && loggedUser.tokenVerification;
  authService.logout = () => $http.post('logout').then(() => {
    loggedUser = null
  });
  return authService;
}]);
```

Рисунок А.3 – Код застосунка з сервісом аутентифікації

```
myApp.controller('2faCtrl', function ($scope, $location, $interval, AuthService, Diploma) {
  if (AuthService.isAuthenticated()) {
    $location.path('/success');
  }
  $scope.loggedUser = AuthService.getLoggedUser();
  $scope.status = 'NotAuthenticated';
  $scope.authenticate = () => {
    Diploma.authenticate($scope.loggedUser.token).then(_ => {
      $scope.status = 'InProgress';
      const stop = $interval(() => {
        $scope.verify().then(() => $interval.cancel(stop));
      }, 500);
    });
  };
  $scope.verify = () => {
    return AuthService.verify().then(() => $location.path('/success'));
  };
```

Рисунок А.4 – Код застосунка з другим фактором аутентифікації