ISSN 2078–5097. Вісн. Львів. ун-ту. Сер. прикл. матем. та інф. 2018. Вип. 26. С. 105–117
Visnyk of the Lviv University. Series Appl. Math. and Informatics. Issue 26. P. 105–117

І Н Ф О Р М А Т И К А

УДК 004.4

БАНКІВСЬКІ ОПЕРАЦІЇ НА БАЗІ ТЕХНОЛОГІЇ БЛОКЧЕЙН. ПРОБЛЕМА КОНСЕНСУСУ В ОДНОРАНГОВИХ МЕРЕЖАХ

Г. Гарбарук, Ю. Щербина

Львівський національний університет імені Івана Франка,
вул. Університетська, 1, Львів, 79000,
e-mail: hryhoriyharbaruk@gmail.com; yshcherbyna@yahoo.com

Описано одну з основних проблем сучасних розподілених баз даних – проблему цілісності даних за умови, що жоден з вузлів не є головним і ніхто один одному в мережі не довіряє. Такі системи називають одноранговими, де функцію сервера може виконувати будь-який клієнт (вузол). Детально описано всі основні поняття технології блокчейну, а саме значення транзакцій і блоків у мережі, що таке одиниця довіри та звідки ця "довіра" береться. Розглянуто як за допомогою використання електронно-цифрових підписів, асиметричного шифрування даних і недовіри досягається надзвичайна стійкість та цілісність даних у мережі. Практично розглянули побудову дерева Меркла для транзакцій, обчислили мінімально потрібну кількість операцій для знаходження хешу SHA-256 у випадку відносно невеликої складності алгоритму.

На практиці довели як працює найпростіший блокчейн застосунок, які ролі є в ньому, які етапи проходить система у своєму циклі. Розглянуто декілька видів атак на блокчейн, які протоколи вразливі й у яких умовах їх варто використовувати.

Провели власний експеримент знаходження хешу блоку у випадку використання протоколу "підтвердження роботи", проаналізували вхідні дані та кінцеві результати, врахувавши усі новітні підходи до моделювання, побудови, підтримки веб проектів у поєднанні з рекомендаціями зі створення блокчейн застосунків, було запропоновано реалізацію власної системи електронного банку зі змішаною трирівневою архітектурою на базі кросплатформеного фреймворку.NET Core. Технічні деталі можна знайти за посиланням на систему контролю версій, де зберігається приклад коду [https://github.com/Harbaruk/Masterwork_LNU].

Ключові слова: блокчейн, однорангова мережа, консенсус, хеш, асиметричне шифрування,.NET Core, банківські операції, дерево Меркла, важкооборотні функції, складність алгоритму, електронно-цифровий підпис, система контролю версій.

1. ВСТУП

У сучасному світі банк є гарантом збереження грошей і цінностей, допомагає безпечно керувати ними. Питання безпеки такої інституції завжди стоїть гостро, особливо з розвитком сучасних технологій. Щороку знаходять нові вразливості мереж, регулярно оновлюють систему захисту інформації та грошей. Використання новітніх девайсів, соціальної інженерії та інших типів засобів у поєднанні з людським чинником створює слабкі місця у системі захисту, які можуть використати зловмисники.

Дані компанії Symantec, яка спеціалізується на захисті інформації [3], такі:

- кожен 13 запит в Інтернеті потенційно небезпечний;
- кількість вразливостей систем збільшилась на 13%;

- кількість зловмисних програм у фінансових установах збільшилась на 15.8 %.

Більшість банків України та світу використовують старі системи для керування та контролю своєї діяльності, що негативно позначається на швидкодії операцій, великій кількості паперової роботи, значному впливу людського чинника, а це призводить до проблем з безпекою: порушення цілісності, втрати даних, програмні перешкоди нормальному функціонуванню системи, крадіжки інформації тощо.

Головний засіб запобігання атак на банківські установи – системи фільтрації запитів та авторизація всіх користувачів у системі. Проте й ці заходи не будуть дієвими при фізичному взломі сервера з конфіденційною інформацією.

У 2008 р. програміст (або група програмістів) під псевдонімом Сатоші Накамото (англ. Satoshi Nakamoto) опублікували статтю про Біткоін-протокол на ресурсі, який присвячений криптографії (http://www.metzdowd.com). Це був перший опис криптовалюти, в основі якої була технологія “блокчейн” або “система розподілених реєстрів”.

Цей підхід розробили для того, щоб виправити головні недоліки наявної банківської системи, створивши стабільну, стійку до атак базу даних, що не потребує постійної підтримки.

Вже є багато реалізацій систем на основі блокчейну, є дослідження великих банківських, наукових установ, інформаційних корпорацій: Німецький Бундесбанк, IBM, The World Bank Group [6] та інших, у сфері використання технології блокчейну для поліпшення внутрішніх і зовнішніх операцій, інтеграції з іншими існуючими системами, створення системи регуляцій тощо [4, 5].

Головна мета – теоретичне обґрунтування використання блокчейну та створення концептуальної моделі реалізації примітивних банківських операцій з використанням згаданого підходу.

2. ФОРМУЛЮВАННЯ ЗАДАЧІ ТА ОСНОВНІ ТЕРМІНИ

Нехай маємо деякий набір вузлів і канал зв’язку між ними. Кожен з вузлів має своє сховище зберігання даних. Припустимо, що один з вузлів передає якесь повідомлення каналами зв’язку іншим, не обов’язково всім. Кожен вузол, який отримує повідомлення, має:

- перевірити цілісність повідомлення;
- перевірити чи правильна адреса адресанта;
- зберегти копію повідомлення собі, якщо воно пройшло перевірки;
- переслати повідомлення далі, якщо потрібно.

Як досягнути цього консенсусу в мережі, тобто такого стану, коли у всіх учасників зберігається копія повідомлення в початковому (не пошкодженому) вигляді? Як гарантувати, що дані не були пошкоджені під час отримання? Як гарантувати стійкість до зламу та маніпулювання даними, якщо один з вузлів буде зловмисником?

Для того, щоб можна було однозначно визначити розв’язок такої задачі, треба розглянути декілька концептуальних теорій і пояснити декілька термінів,якідопоможуть у розумінні підходу вирішення такого класу задач.

Блокчейн – теоретичний концепт побудови баз даних, які надзвичайно стійкі до взлому, підробки та майже унеможливлюють знищення даних. Забезпечуються такі характеристики децентралізацією даних у системі й асиметричною системою

шифрування [1]. Система складається з двох головних компонентів: транзакцій і блоків.

Транзакція – найменша інформаційно цілісна одиниця в блокчейні. Вона може мати будь-яку кількість параметрів, будь-якого типу, проте основними є:

- дані – будь-яка інформація, що має цінність і повинна бути збережена в системі;
- цифровий підпис – послідовність символів, що генерується з хешу транзакції та приватного ключа;
- публічний ключ – ключ для перевірки цілісності транзакції та підтвердження ініціатора.

Блок – це фіксований за розміром набір транзакцій. Саме його використовують для підтвердження збереження даних. Складається з:

- хеш попереднього блоку;
- хеш блоку;
- дерево Меркла, що згенероване з транзакцій;
- список транзакцій.

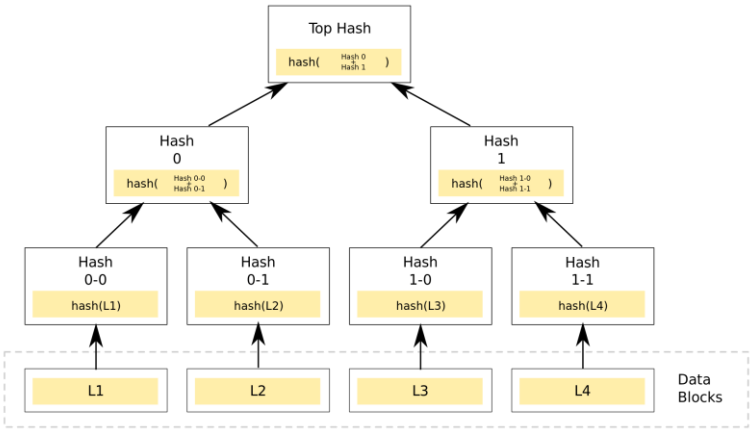

Рис. 1. Дерево Меркла

Дерево Меркла – дерево хешів, що за допомогою алгоритму створює єдине значення хешу, яке ще називають коренем. Використовують для перевірки цілісності даних.

Алгоритм генерації кореня дерева хешів складається з двох етапів.

1. Перетворення всіх записів за допомогою вибраної функції хешування.
2. Рекурсивне виконання хешування за такими правилами:
   а) беремо попарно записи, хешовані записи, об’єднуємо їх в одну стрічку та беремо хеш від цього значення;
   б) якщо є неопрацьоване значення на цій ітерації без пари, то записуємо його без змін як нове.

Розглянемо на прикладі генерацію дерева. Візьмемо непарну кількість транзакцій, алгоритм SHA-256 для хешування:

1) “Гроші:300, Відправник:54445345, Отримувач:8543564564”
2) “Гроші:300, Відправник:12232534, Отримувач:7553645645”
3) “Гроші:300, Відправник:32253578, Отримувач:1543536451”
4) “Гроші:300, Відправник:53231588, Отримувач:1543535645”
5) “Гроші:300, Відправник:53242145, Отримувач:4353645645”

Візьмемо хеш першої та другої функції:

1) 942f2d90662512500503da62e5fb2aaa9732becf41770018047803a9e54ffea0.
2) 7cb510fe778c0c6909105edf88f612d3633699ea0f1890b19a22b9d5fbbcc5e6.

Як бачимо, дані відрізняються лише кількома значеннями, проте хеші зовсім різні. Оскільки хеш функція – це важкооборотна функція, то немає алгоритму, який зможе перетворити хеш назад у дані, які доступні для читання.

Так само потрібно зробити з записами 3, 4 та 5:

1) 9c2a9150acc27c7cd8e85dbbc7bb56cef5761c1aebd68d989002cb5148fe8fea
2) b5d580fbaf888bfa668197c21d37c28b39bd8a7e80160f1204b2c4146d7026f6
3) 23702f6a47645816e56031ec2471197f9842f08d47e454b8176d357bdc37e88a.

Далі потрібно об’єднати 1 та 2, 3 та 4, знову взяти хеш від них. Запис 5 не використовується і приймається без змін. Повторимо ці маніпуляції двічі, доки не отримаємо остаточний результат. Кінцеве значення кореня таке:

7076b11f2d21382022395bb7e85330416216d3915848e3cf1fc5dda8be671ee7.

Ми порахували корінь дерева Меркла. Тепер це значення можна використовувати для підтвердження цілісності набору записів, оскільки зміна у будь-якому місці хоча б одного запису повністю змінить хеш.

Мережа, в якій зберігається інформація у вигляді блокчейну, зазвичай називається “одноранговою” (peer-to-peer). Головна відмінність від звичайної архітектури типу “клієнт-сервер” це те, що кожен вузол мережі одночасно може виконувати роль сервера та клієнта. Також між ними немає ніякої системи пріоритетів під час виконання запитів і завдань.

Децентралізація блокчейну досягається завдяки зберіганню повної історії блоків на кожному вузлі мережі незалежно. Це дає стійкість від пошкодження та знищення даних, оскільки для пошкодження потрібно інфікувати $n / 2 + 1$ вузлів мережі, а для знищення усі n вузлів (де n – кількість учасників мережі).

Найскладніша проблема підтримки стійкості блокчейну – досягнення консенсусу у мережі. Перше правило для системи – повна недовіра вузлів один до одного. Виникає питання досягнення певної згоди у прийнятті рішень щодо підтвердження даних, конфліктів нової чи вже існуючої інформації.

У більшості систем використовують “криптовалюту”, хоча, як ми вважаємо, краще використовувати термін “одиниця довіри”.

Одиниця довіри позначає абстрактну одиницю, яку отримує вузол за вклад у стабільність і стійкість системи. Використовується у так званих протоколах мережі – надбудові над системою, що визначає поведінку та спосіб досягнення консенсусу.

Було розроблено декілька основних надбудов, їх є близько 15, але варто зазначити лише стабільні та найчастіше використовувані:

- підтвердження роботи;
- підтвердження частки;
- підтвердження активності;

- підтвердження згоряння;
- підтвердження місця.

2.1. ПІДТВЕРДЖЕННЯ РОБОТИ

Спочатку розглянемо принцип роботи у випадку протоколу підтвердження роботи.

Кожен учасник (вузол) системи має свою копію усіх даних, а також може створювати власні транзакції. Усі створенні транзакції передаються по мережі і потрапляють у список непідтверджених записів. Коли набирається достатня кількість транзакцій (кількість визначається системою або її розробником), учасники на швидкість намагаються створити блок з транзакцій.

Оскільки не потрібно багато часу, щоб обчислити хеш блока, то запропонували математичну задачу – знайти таке число nonce, яке задовольняє умови

$$i = [1,p] \{ hash(block, sign, nonce)\ i = 0 \},$$

де p – складність.

Коли знаходиться nonce, ми використовуємо результат хешування як унікальний ідентифікатор блоку та передаємо його усім вузлам мережі. Кожен вузол ще раз перераховує хеш, використовуючи публічний ключ з блока, цифровий підпис і знайдений nonce, щоб порівняти його з хешом блока. Якщо вони збігаються – то підписує блок і зберігає його на своєму комп'ютері.

Для читабельності хеш приводять до кодування base64, який набуває вигляду

aHR0cHM6Ly93d3cueW91dHViZS5jb20vd2F0Y2g/dj04Zjd3al9SY3FZaw==

Вигляд хешу для блоку зі складністю p = 10

000000000093d3cueW91dHViZS5jb20vd2F0Y2g/dj04Zjd3al9SY3FZaw==

Отож, ми отримали блок з унікальним, складним хешом, який підписаний вузлом, що створив цей блок. Усі учасники мережі за допомогою публічного ключа можуть перевірити правильність хешу та підпису. Вузол, який створив блок, отримує певну кількість одиниць довіри, що може використовуватись або ні залежно від правил мережі.

2.2. ПРОБЛЕМА КОЛІЗІЙ

Як мережа буде діяти, якщо з різницею в долі секунди буде знайдено 2 блоки, який мають різний nonce, але різний хеш, що задовольняє умови? Якщо в нас буде не 2 блоки, а 2 гілки блоків, що відповідають нашим заданим параметрам?

За замовчуванням протокол визначає правильними блоками такі:

- перший, що прийшов і є правильний, додається в список;
- якщо є гілки, то беремо довшу з них.

2.3. ПІДТВЕРДЖЕННЯ ЧАСТКИ

Proof-of-stake (PoS) (від англ. *proof of stake*, дослівно: "підтвердження частки") – у цьому протоколі головним чинником, який впливає на ймовірність знаходження правильно хешу, буде не швидкість обчислення, а кількість одиниць довіри вузла. Складність хешу, який потрібно знайти, буде обернено-пропорційна кількості довіри вузла.

Наприклад, Node1 має 10% усієї довіри, Node2 має 1%. Обчислення складності таке:

$$+ (98/N * n)\ \%,$$

де N –кількість довіри у всій системі; n –довіра вузла.

Отож, кожен вузол, у тім числі той, що тільки приєднався до мережі, має хоча б мінімальну ймовірність знайти блок, а не лише ті, які вже мають довіру.

Аргументи, які засвідчують спроможність методу:

- для проведення атаки потрібно багато довіри. Атакуючому буде дорого виконати атаку;
- якщо в атакуючого знайдеться багато одиниць довіри, то він сам постраждає від атаки, оскільки це порушить стійкість системи.

Аргументи, що викликають побоювання:

- PoS дає додаткову мотивацію до накопичення довіри в одних руках, що може негативно позначитися на децентралізації мережі;
- якщо утвориться невелика група, яка сконцентрує у себе досить великі ресурси, то вона зможе нав'язувати свої умови функціонування системи.

2.4. ПІДТВЕРДЖЕННЯ АКТИВНОСТІ

Proof-Of-Activity (від англ. *proof of activity*, доказ активності) – гібридна система консенсусу, яка поєднує системи PoW і PoS, що були раніше описані.

Принцип роботи алгоритму.

1. Вузол шукає хеш відповідної складності.
2. Знайдений хеш відправляється в мережу, хоча не є повноцінним блоком, а лише першим кроком, своєрідним шаблоном, який потрібний для його створення.
3. Хеш, який складається з 256 псевдовипадкових біт, інтерпретується як N чисел, до кожного з яких у відповідність ставиться одна умовна одиниця роботи в мережі.
4. Налагоджується взаємно однозначний зв'язок між умовними одиницями та публічними ключами.
5. Як тільки всі N власників поставлять свої підписи на цьому блоці, на виході виходить повноцінний блок.
6. Якщо один з власників недоступний, то інші вузли продовжують генерувати шаблони з різними комбінаціями холдерів-кандидатів.
7. У якийсь момент необхідний блок буде підписаний потрібну кількість разів. Нагорода за блок розподіляється між вузлами, що шукали та підписували блок.

2.5. ПІДТВЕРДЖЕННЯ ЗГОРЯННЯ

Proof-Of-Burn (від англ. *proof of burn*, доказ спалення) – гібридна система консенсусу, яка базується на добровільну витрату якоїсь цінності в обмін на одиниці довіри. Тобто, чим більше витрачено певних ресурсів, тим більше одиниць довіри отримує вузол і, відповідно, складність для нового хешу менша.

2.6. ПІДТВЕРДЖЕННЯ МІСЦЯ

Proof-Of-Storage (від англ. *proof of storage*, доказ вільного місця) – система консенсусу, де одиницею довіри є кількість пам'яті на пристрої (персональний комп'ютер, планшет, телефон), що відкрита для зчитування та запису іншими учасниками мережі.

3. ЕКСПЕРИМЕНТ

3.1. ПЕРЕДУМОВИ

Перевірити стійкість системи на слабкому комп'ютері за умови, що складність хешу значна.

Візьмемо комп'ютер із середньою швидкістю 2000 операцій хешування на секунду. Складність 6, тобто хеш має розпочинатись із 6 нулів.

Хеш має Base64 кодування, яке передбачає символи A-Z, a-z, 0-9, +, /. Загалом 64 можливих символи. Довжина нашого хешу 43 символи і '=' в кінці.

3.2. ТЕОРЕТИЧНІ ОЦІНКИ ЙМОВІРНОСТЕЙ

Отже, якщо мати комп'ютер, що обчислює 2000 хешів в секунду, то щоб перебрати всі можливі комбінації хешів, треба:

(64^43)/2000 ≈ 2.3*10^77 секунд ≈ 7.2^69 років.

Навіть найпотужніший комп'ютер сучасності, який має швидкодію близько 2*10^17 операцій на секунду, буде перебирати хеш близько 7*10^52 років!

Щоб оцінити масштаби цієї оцінки, візьмемо, наприклад, вік Землі, що становить лише 4.5^9 років. Це обмеження лише знизу, оскільки є ненульова ймовірність знайти 2 однакових хеша для різних даних.

Нам не потрібно знаходити всі можливі комбінації, а лише ті, які розпочинаються з певної кількості нулів, а саме 6. Обчислимо ймовірність знаходження такого блоку:

64^6 ≈ 6.8*10^10 перевірок ≈ 1 рік.

Всі оцінки часу – лише нижня межа, оскільки існують різні набори даних, які мають однаковий хеш.

3.3. ВИКОНАННЯ

У зв'язку з незалежними процесами комп'ютера, швидкість обчислень була змінною, 1500-3500 хешів на секунду.

Таблиця 1

Статистика появи хешу

Кількість нулів	Ітерація першого знаходження	Час першого знаходження	Загальна кількість знаходжень
1	236	0.1с	29500555
2	3740	1.5с	472221
3	32578	14.8с	6857
4	5841636	2920с	103
5	1989464116	~ 3 дні	1

Через 5 днів після запуску експерименту ми зупинили програму, не отримавши хеш відповідного розміру. Перераховували хеш більше двох мільярдів разів і знайшли хеш зі складністю 5, на ~ 1.1 * 10^9 спробі. Агрегувавши дані, ми можемо навести статистику появи хешу, що задовольняє наші умови.

3.4. ПІДСУМКИ ЕКСПЕРИМЕНТУ

Виконання такої роботи, а саме знаходження хешу відповідної складності – дуже енергозатратна задача і практично неможлива для одного комп'ютера, навіть потужного. Тому системи, де багато людей паралельно виконують пошук хешу, надзвичайно стійкі до атак.

Наразі навіть корпорації рівня Microsoft і Google не в змозі зробити атаку на найпопулярнішу систему "Біткоін", оскільки немає такої обчислювальної потужності.

Використання важкооборотних функцій хешування в поєднанні з розподіленістю даних в одноранговій мережі дає надзвичайно високий рівень стійкості до будь-яких атак, що у традиційній клієнт-серверній архітектурі зробити неможливо.

4. ПРОЕКТУВАННЯ ПРИМІТИВНОЇ БАНКІВСЬКОЇ СИСТЕМИ

Врахувавши головні тенденції у проектуванні інформаційних систем, для зручності у розширенні та підтримці, використаємо модифіковану трирівневу архітектуру проекту на базі підходу RESTfull API, 2 бази даних для розділення банківських і супутніх даних, додаткову інформацію для інтеграції, систему контролю версій для підтримки командної розробки, Continuous Integration Process для постійної інтеграції з тестовим середовищем.

Оскільки клієнти не мають повноважень підтверджувати транзакції, впливати на рахунки напряму без перевірки банку, то ми можемо використати комбінацію клієнт-сервер та однорангову мережу.

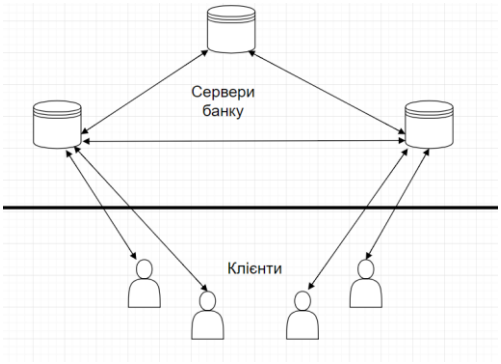

Рис. 2. Однорангова мережа

RESTfull API – підхід до побудови архітектури, що забезпечує доступ до ресурсів за допомогою HTTP запитів, унікальних URL адрес для ресурсів та опрацювання визначеного словника кодів відповіді [11].

Є 4 основних типи HTTP запитів, які ми будемо використовувати в застосунку:

- GET – запит на отримання даних, не має тіла запиту, лише адресу та параметри;
- POST – запит на запис інформації, має тіло та адресу. Дані передаються в адресі та у тілі запиту;
- PUT – запит на оновлення вже існуючої інформації, не відрізняється параметрами від POST;
- DELETE – запит на видалення даних.

Визначимо головні коди відповіді у системі:

- 200 OK – надає інформацію, що запит відбувся успішно. Може містити також додаткову інформацію чи об'єкт, який очікує запит;
- 400 Bad Request – інформація про те, що запит не відбувся успішно. Має інформацію про те, що трапилось у вигляді набору даних "ключ-значення";
- 500 Server Error – непередбачені обставини, запит завершився аварійно. На етапі розробки буде повертати повну інформацію про помилку, на етапі готового проекту лише записувати дані в сховище для того, щоб зберегти інформацію про систему від зловмисників.

Стандартна трирівнева архітектура передбачає три головні функціональні модулі:

1) рівень доступу до бази даних – відповідає за зберігання, цілісність і видобування інформації;
2) рівень бізнес логіки – відповідає за трансформацію, калькуляцію та інші маніпуляції з даними;
3) рівень інтерфейсу проекту – відповідає за передачу інформації у мережі.

Бази даних:

1) класична реляційна база даних для збереження основної інформації про користувача: власний кабінет, відкриті рахунки, відгуки чи інше.
   Містить у собі:
   а) основну інформацію про користувача, яка потрібна для ідентифікації;
   б) банківські рахунки, їхній статус й баланс;
   в) додаткові дані (наприклад, для двофакторної авторизації);
2) блокчейн – для зберігання транзакцій та блоків:
   а) список транзакцій;
   б) список блоків.

Рівень бізнес логіки складається з:

1) сервісів – функціональні модулі, що відповідають за набір задач, пов'язаних з конкретною сутністю бази даних або процесом всередині програми. (LoginService відповідає лише за аутентифікацію користувача);
2) допоміжні модулі (Helpers) – не мають доступу до бази даних, використовуються іншими сервісами для видозміни даних чи пов'язані з зовнішніми програмами (надсилання емейлів, кореляція часу з сервером часу Google).

Рівень інтерфейсу проекту – набір спеціальних методів, які можуть бути викликані через мережу Інтернет за допомогою http/https запитів. Використовують у собі сервіси з рівня бізнес логіки.

Тут використовується основна фільтрація і захист нашої програми, а саме:

- аутентифікація та первинна авторизація користувача;

- фільтрація запитів від неправильних чи заборонених даних;
- базове опрацювання помилок.

4.1. ЗАХИСТ ЗАСТОСУНКУ

Оскільки ми розробляємо банківську систему, головна мета якої – захистити гроші та приватну інформацію, що може надати доступ до рахунку чи будь-як по-іншому зашкодити клієнту, то потрібно мати засоби протидії атакам, крадіжкам даних, маніпуляціям та іншим злочинним діям, які спрямовані на нашу систему.

Оскільки блокчейн сам собою має вже достатній рівень захисту, то треба налаштувати захист на вищому рівні. Основними атаками на систему будуть[2]:

- ін'єкція SQL;
- підбір пароля до акаунта;
- DoS атака (атака відмови обслуговування);
- XSS атака (міжсайтовий скриптинг) [8].

Оскільки ми знаємо, які атаки найпоширеніші, то можемо фундаментально вибудувати захист нашого застосунку. Використавши рекомендації Microsoft для побудови веб додатків, з'ясуємо, які технології варто використовувати для протидії атакам, що наведені вище.

Ін'єкція SQL – тип атак, основним змістом яких є спроба виконати код на сервері за допомогою передачі цього кода у поля, що призначені для введення даних користувачем. Цей код може просто давати інформацію зловмиснику щодо вашого сервера, структури бази даних та інші параметри системи, які дають змогу планувати інші типи атак. Найкращий захист – використання параметризованих запитів або спеціальних засобів контролю (ORM-фрейморків) доступу до бази даних. Отож, будь-які дані, які надсилає користувач, будуть трактуватись як дані, незалежно від змісту і ніколи не виконаються, якщо там зловмисний код. Ми у застосунку використовуватимемо Entity Framework Core, засіб для роботи з базою даних на рівні коду.

Підбір пароля до акаунта – перебір ймовірних паролів або витягнута інформація з бази даних щодо паролю користувача. Захист від цього типу атак передбачає:

- збереження пароля у нечитабельному вигляді, обов'язкове шифрування пароля, відсутність дешифрування;
- обмеження на мінімальну довжину та кількість різних символів у паролі.

Ми використаємо шифрування за допомогою PBKDF2 алгоритму [7]. Він був створений для заповільнення генерації пароля, що зменшить ймовірність знаходження пароля звичайним перебором.

Загальний вигляд алгоритму:

$$DK = PBKDF2\ (Prf, P, S, c, dkLen),$$

де *Prf* – псевдовипадкова функція; *P* – мастер-пароль; *S* – згенерована сіль; *c* – кількість ітерацій; *dkLen* – потрібна довжина вихідного значення.

Сіль – псевдовипадкова стрічка даних, яка надає додатковий захист від перебору, оскільки вона унікальна для кожного користувача [9]. За допомогою цієї солі ми можемо уникнути ризику атаки перебором, коли використовується наперед заданий словник головних правил створення пароля. Такі словники можна придбати у мережі і вони охоплюють статистичні дані найбільш ймовірних паролів. Наприклад,

якщо пароль на сайті обов'язково має містити хоча б одну велику літеру, то найбільш ймовірно, що ця літера буде на початку пароля, оскільки так легше запам'ятати.

Отож, ми зможемо зберегти паролі у зашифрованому вигляді та перевіряти лише збіг згенерованих наборів символів. Також обмежуємо довжину паролю не менше 8 символів.

DoS / XSS атаки – перекладемо цю функцію на сервер, де буде перебувати наш застосунок. Під час реалізації було вибрано сервіс Microsoft Azure, який надає власні сертифікати захисту, має вбудовану підтримку фільтрування великої кількості запитів та зручне налаштування списку дозволених сервісів, на які можна надсилати дані, щоб уникнути XSS атак.

5. ВИСНОВКИ

Отже, ми розглянули основні поняття блокчейну, детальніше дослідили структуру та способи досягнення цілісності даних. Як бачимо, технологія розподілених реєстрів виявила чудові результати у швидкодії, безпеці, конфіденційності. Незважаючи на такі хороші результати, вона має певні недоліки, можуть бути критичними, наприклад, типи атак, від яких ще немає захисту, крім як простого збільшення потужності самої мережі.

Можна з упевненістю сказати, що цей підхід до побудови застосунків має великий потенціал, не лише в фінансовій сфері. Зараз з'являються перші використання блокчейну у сфері медицини (розподілені бази пацієнтів, ліків), управління (системи анонімного оцінювання), туризму (система пошуку житла та відгуки до нього).

Також державні установи розпочинають активно розвивати цю технологію. Національний Банк України оголосив про створення електронної гривні на базі блокчейну [10], земельний реєстр вже переведений на цю технологію.

За допомогою проекту, ми з'ясували головні підходи щодо реалізації та використання технології блокчейн. Розглянули не зовсім класичну архітектуру проекту, що на основі випробувань показує хороші результати, спрощує інтеграцію з наявною системою банків, гнучка щодо розширення. Проаналізували основні типи атак на такі застосунки та побудували систему захисту, використовуючи рекомендації Microsoft. Використали хмарну технологію Azure для запуску проекту, створили базову платформу для подальшої розробки подібних проектів.

У експерименті ми проаналізували наскільки складною є задача знаходження хешу із заданими правилами, якщо брати звичайний комп'ютер. Оцінили кількість можливих комбінацій і відрізок часу, який потрібний на повний перебір цих комбінацій.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. *Drescher D.* Blockchain Basics: A Non-Technical Introduction in 25 Steps / D. Drescher. – 2017.
2. *Sobers R.* 60 Must-Know Cybersecurity Statistics for 2018 / R. Sobers. – Режим доступу: https://www.varonis.com/blog/cybersecurity-statistics/
3. 2018 Internet Security Threat Report. – Режим доступу: https://www.symantec.com/security-center/threat-report
4. *Lang Jo* Three uses for blockchain in banking /Jo Lang. – Режим доступу: https://www.ibm.com/blogs/blockchain/2017/10/three-uses-for-blockchain-in-banking

5. *Pratap M.* How is Blockchain Revolutionizing Banking and Financial Markets / M. Pratap. – Режим доступу: https://hackernoon.com/how-is-blockchain-revolutionizing-banking-and-financial-markets-9241df07c18b
6. *Orcutt M.* The World Bank is a verified blockchain booster, 2018 / M. Orcutt. – Режим доступу: https://www.technologyreview.com/s/612043/the-world-bank-is-a-verified-blockchain-booster/
7. *Dharmadasa K.* How to store passwords securely with PBKDF2 / K. Dharmadasa. – 2017. – Режим доступу: https://medium.com/@kasunpdh/how-to-store-passwords-securely-with-pbkdf2-204487f14e84
8. *Anderson R.* Prevent Cross-site scripting (XSS) in ASP.NET Core 2.1 / R. Anderson. – 2018. – Режим доступу: https://docs.microsoft.com/en-us/aspnet/core/security/cross-site-scripting?view=aspnetcore-2.1
9. Salt (cryptography). – 2018. – Режим доступу: https://en.wikipedia.org/wiki/Salt_(cryptography)
10. *Смолій Я.* НБУ готує електронну заміну гривні / Я. Смолій. – 2018. – Режим доступу: https://m.znaj.ua/society/177150-nbu-gotuye-elektronnu-zaminu-grivni
11. HTTP Status Codes. – 2017. – Режим доступу: https://restfulapi.net/http-status-codes/

Стаття: надійшла до редколегії 10.09.2018
доопрацьовано 24.10.2018
прийнята до друку 31.10.2018

CONSENSUS PROBLEM IN PEER-TO-PEER NETWORKS. IMPLEMENTATION OF BANKS' TRANSACTIONS BASED ON BLOCKCHAIN TECHNOLOGY

H. Harbaruk, Yu. Shcherbyna

Ivan Franko National University of Lviv,
Universytetska Str., 1, Lviv, 79000,
e-mail: hryhoriyharbaruk@gmail.com; yshcherbyna@yahoo.com

One of the main problems of modern distributed databases is described - the problem of data integrity, because of none of the nodes is the main one and nobody trust each other in the network. Such systems are called peer-to-peer, wherefunctions of the server can be performed by any client (node).

We tried to describe in details all the basic concepts of blockchain technology, core role of transactions and blocks in the network, told about the unit of trust and where this "trust" is taken. Considered both through the use of digital signatures, asymmetric data encryption and lack of trust, the extraordinary stability and integrity of data in the network is achieved. Check the example of the construction of the Merkle Tree for transactions, computed the minimum number of operations required to find the hash SHA-256 in the case of relatively small complexity of the algorithm.

In practice, it was shown how the easiest blockchain software application works, what roles are in it, and what stages the system runs in its lifecycle. Several types of attacks on blockchain are considered, which protocols are vulnerable and in which conditions they should be used.

An own experiment was conducted to find the hash of the block in case of use of the "proof-of-work" protocol. The average performance rate with a light complexity of the output hash, spent 5 days in a continuous search and analyzed results.

Taking into account all the latest approaches to modeling, constructing, supporting web projects in conjunction with the recommendations for creating blockchain applications, it was proposed to implement its own system of electronic bank with a mixed three-level architecture based

on the cross-platform framework.NET Core. Technical details can be found in the link to the version control system where the example code is stored [https://github.com/Harbaruk/Masterwork_LNU].

Key words: block cache, peer-to-peer network, consensus, hash, asymmetric encryption,.NET Core, banking operations, Merkel tree, complexity of the algorithm, electronic-digital signature, version control system.