Біткоїн: електронна пірингова система готівки

Сатоші Накамото
satoshin@gmx.com
www.bitcoin.org

Перекладено на українську спільнотою “Що біткоїться”
Маєте зауваження? Пишіть: t.me/wtfbit
wtfbit.media

Анотація. Повністю пірингова версія системи електронної готівки дозволяє надсилати онлайн-платежі напряму від однієї сторони до іншої, оминаючи будь-які фінансові установи. Хоча цифрові підписи частково надають розв’язання цієї проблеми, більшість переваг втрачаються, якщо довіреній третій стороні (посереднику) треба уникнути подвійних витрат. Ми пропонуємо своє рішення задачі подвійних витрат, використовуючи пірингову мережу. Така мережа робить на транзакціях часову позначку, хешуючи їх в неперервний ланцюжок підтвердження виконання роботи (proof-of-work). Таким чином утворюється запис, котрий неможливо змінити без повторного виконання всього ланцюжка обчислень. Найдовший ланцюжок не тільки слугує доказом послідовності подій, а і свідчить про те, що він з’явився в результаті роботи найбільшого сегмента потужності обчислювальної системи. Якщо більшість обчислювальної сили контролюється чесними вузлами, то вони генерують найдовший ланцюжок обчислень, випереджаючи атакуючі вузли. Структура самої мережі доволі проста. Повідомлення надсилаються за методом негарантованої доставки й усі вузли можуть покидати мережу та долучатися до неї в довільний момент часу, приймаючи найдовший ланцюжок для відновлення історії транзакцій за період їх відсутності.

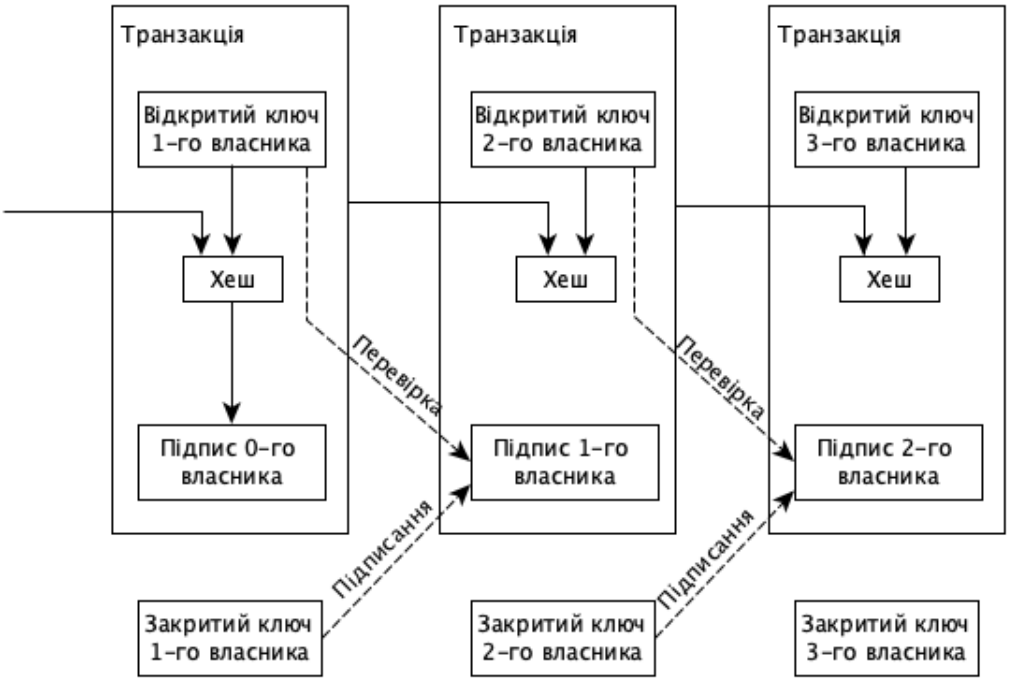
1. Вступ

Інтернет-комерція майже цілковито покладається на фінансові установи, які виступають у якості довірених посередників для обробки електронних платежів. Хоч така система працює доволі добре для більшості транзакцій, вона все ще має недоліки, притаманні всім моделям на основі довіри. Повністю односторонні транзакції неможливі доти, доки фінансові установи не можуть уникнути опосередкування суперечок. Вартість такого опосередкування збільшує вартість транзакцій і встановлює мінімальний розмір транзакції та, відповідно, скорочує можливості невеликих платежів. Це призводить до унеможливлення здійснення незворотних платежів за незворотні послуги. З можливістю повернення платежів збільшується потреба у довірі. Продавці мають бути обережними зі своїми клієнтами та вимагати від них більше інформації, ніж потрібно. Певна частка шахрайства вважається неминучою. Ці всі витрати та невизначеність платежів, звісно, можна уникнути персонально, використовуючи фізичну валюту, але не існує механізму здійснення платежів через канал зв’язку без довіреної сторони.

Що необхідно, так це електронна система платежів на основі криптографії, що виключає фактор довіри та дозволяє сторонам надсилати платежі один одному напряму, без потреби в посередниках. Операції, які є обчислювально непрактичними для зворотних платежів, будуть захищати продавців від шахрайства, а звичайні механізми депонування можуть бути легко реалізовані для захисту покупця. У цьому документі ми пропонуємо рішення задачі подвійних витрат, використовуючи піринговий розподілений сервер позначок часу для генерації обчислюваного підтвердження хронологічного порядку транзакцій. Система захищена доти, доки чесні вузли контролюють більше потужності процесора, ніж будь-яка скоординована група атакуючих вузлів.

2. Транзакції

У нашому уявленні електронна монета – це ланцюжок цифрових підписів. Кожен власник передає монету наступному шляхом цифрового підпису хеша попередньої транзакції та публічного ключа наступного власника, а після цього додає отримані дані в кінець монети. Одержувач може перевірити підписи для підтвердження ланцюжка власників.

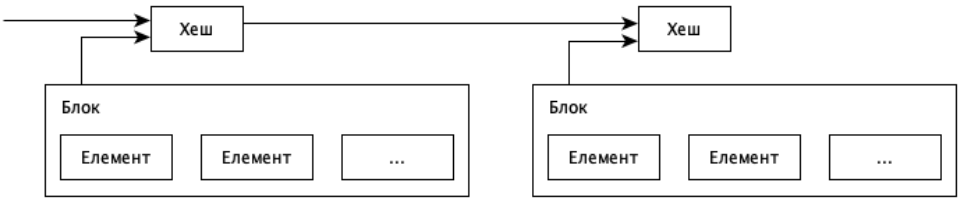


Існує очевидна проблема – отримувач не може дізнатися, чи не витратив один із попередніх власників монету двічі. Поширеним розв'язком даної проблеми є перевірка довіреним центральним органом (емітентом) кожної транзакції на факт подвійної витрати. Після кожної операції монета має бути повернена емітенту для випуску нової монети, і тільки після такої процедури монета буде підтвердженням того, що вона не витрачалася повторно. Недолік такого рішення в тому, що доля усієї грошової системи залежить від керівної компанії емітента, тому що вона, як і банк, контролює всі транзакції, що через неї проходять.

Нам необхідно знайти метод, який надаватиме змогу отримувачу дізнатися, що попередній власник не підписав жодних транзакцій, що передують відправленій транзакції в ланцюжку монети. Для наших цілей важливо врахувати лише попередню транзакцію, а всіма іншими спробами подвійних витрат ми знехтуємо. Єдиний шлях підтвердити відсутність транзакції – знати про всі здійснені платежі. У моделі з довіреною стороною сам емітент знає про всі вхідні транзакції і сам вирішує їх порядок. Виходить, щоб виключити емітента з моделі, необхідно зробити транзакції публічно доступними [1], а також організувати для учасників систему погодження єдиної історії порядку отримання платежів. Отримувачу необхідно впевнитися, що під час кожної транзакції більшість вузлів погоджуються з їх отриманням вперше.

3. Сервер позначок часу

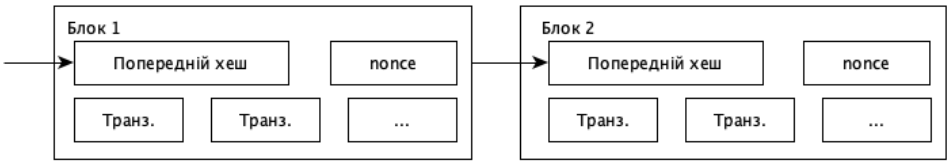
Запропоноване нами рішення бере початок із серверу позначок часу. Суть його роботи полягає в хешуванні блоку елементів, його часовій позначці та наступній публікації позначеного хеша, так само, як у газетах або Usenet-публікаціях [2-5]. Позначка часу свідчить про те, що певні дані існували в певний момент часу, а потім потрапили в хеш. Кожна мітка часу містить попередню в хеші, утворюючи ланцюжок, де наступна ланка підсилює попередню.



4. Доказ виконання роботи

Реалізація розподіленого сервера позначок часу на піринговій основі вимагає від нас використання системи доказу виконання роботи, скоріше схожу на алгоритм Hashcash Адама Бека [6], аніж на газети або Usenet-публікації. Доказ виконання роботи передбачає пошук такого значення, чий хеш, такий як SHA-256, починався би з певної кількості нульових бітів. Необхідна середня кількість задач експоненційно залежить від кількості необхідних нульових бітів, та для верифікації знайденого значення необхідно обчислити всього один хеш.

У нашій мережі часових позначок ми впровадили підтвердження виконаної роботи шляхом інкременту спеціального поля (nonce) в блоці, доки не буде знайдене значення, що видає потрібну кількість нульових бітів. Щойно блок, що задовольняє вимоги, буде знайдений, його неможливо буде змінити без повторного запуску всього ланцюжка. Якщо цей блок буде не останнім у черзі, то наступні блоки також будуть обчислюватися повторно.



Доказ виконання роботи вирішує задачу визначення варіанту прийняття рішень за правилом більшості. Метод «одна IP-адреса – один голос» може бути вразливим, якщо у зловмисника під контролем чималий пул IP-адрес. На відміну від попереднього варіанту, доказ виконання роботи базується на принципі «один ЦП – один голос». Вибір більшості відображається як найдовший ланцюжок, у який вкладено найбільше ресурсів. У випадку контролю чесними вузлами більшості обчислювальної системи, чесний ланцюжок транзакцій, відповідно, буде рости швидше, випереджаючи будь-які конкурентні варіанти. Для внесення змін у попередній блок, атакуючому доведеться повторно виконати роботу над цим блоком та всіма наступними, а вже після цього наздогнати та обійти роботу чесних вузлів. Нижче ми покажемо, що така ймовірність в атакуючого вузла експоненційно знижується з ростом кількості блоків.

Для компенсації зростальної обчислювальної потужності та непостійної кількості активних вузлів у мережі, складність доказу виконання роботи має змінюватися. Рівень складності визначається ковзним середнім, яке наближається до середньої кількості блоків за годину. Згідно з цим, складність виростає, якщо блоки генеруються надто швидко.

5. Мережа

Нижче представлені кроки роботи мережі:

1) Нові транзакції надсилаються всім вузлам в мережі.
2) Вузли збирають їх у блок.
3) Кожен вузол працює над знаходженням доказу виконання роботи для свого блоку.
4) Коли вузол знаходить потрібне значення, він розсилає свій блок усім вузлам.
5) Вузли приймають блок тільки у випадку, якщо всі транзакції в ньому правильні та не витрачені повторно.
6) Про підтвердження прийому блока буде свідчити початок створення нового блоку, який матиме в собі хеш прийнятого блоку в якості попереднього хешу в ланцюжку.

Вузли завжди вважають правильним найдовший ланцюжок та працюють над його розширенням. У випадку, якщо два вузли одночасно розсилають різні версії наступного блока, то деякі вузли отримають першою одну версію, а деякі іншу. У такому випадку кожен почне працювати над першим отриманим блоком, а другий варіант гілки зберігає на випадок, якщо він подовжиться раніше. Ця проблема вирішиться, коли буде отриманий новий блок та одна гілка подовжиться; вузли, які працювали над іншою гілкою, відразу перейдуть на нову (довшу).

Нова розсилка транзакцій не обов'язково має охоплювати всі вузли. Якщо про них буде відомо достатній кількості вузлів, то вони незабаром потраплять у блок. Загублені повідомлення під час розсилки блоків також не є проблемою. Якщо вузол отримує новий блок і виявляє, що один блок пропущений, то він робить на нього запит і заповнює прогалину.

6. Стимули

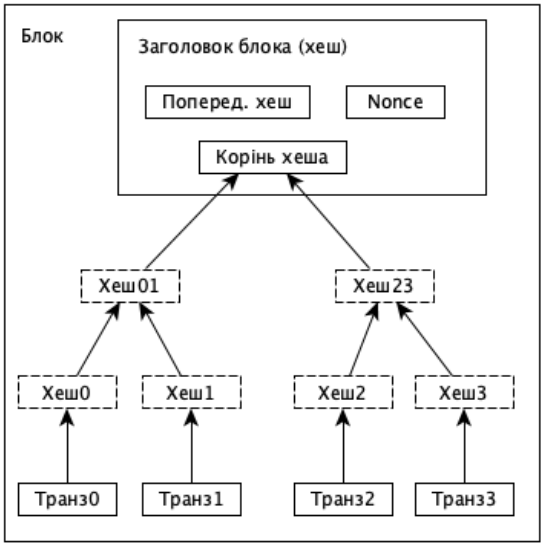
Вважається, що перша транзакція в блоці – це особлива транзакція, яка створює нову монету, яка належить власнику блока. Такий варіант стимулює чесні вузли підтримувати мережу, а також вирішується питання емісії монет, оскільки не існує центрального емітента для їх випуску. Рівномірний приріст числа нових монет можна порівняти із добувачами золота, які вкладають свої ресурси в добування золота та його передачу в обіг. У нашому випадку вклад – це процесорний час та електроенергія.

Стимулом також може бути комісія за транзакцію. Якщо вихідна величина транзакції менша за вхідну, то різниця між ними – це і є комісія за переказ, яка додається до спонукальної винагороди в блок із транзакцією. Щойно встановлена кількість монет увійде в обіг, така винагорода може повністю трансформуватися в комісію від транзакцій, стійку до інфляції.

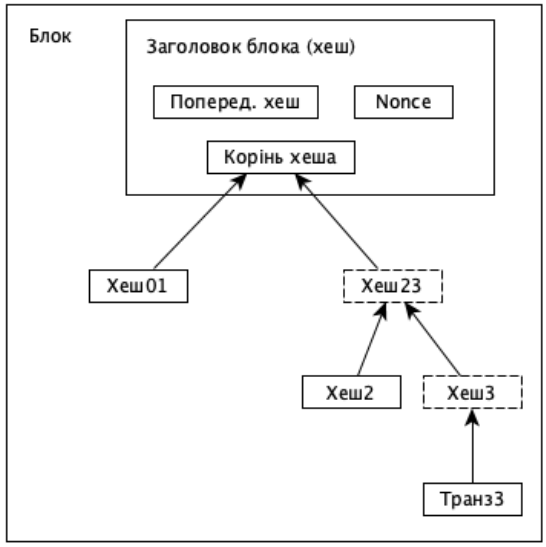
Така система стимулів може заохочувати вузли залишатися чесними. Якщо жадібний зловмисник має змогу виділити більше обчислювальної потужності, аніж чесні учасники, то в нього є два варіанти: обманювати людей, повертаючи назад свої транзакції, або ж використовувати їх для створення нових монет. Він має визнати, що гра за правилами вигідніша для нього, бо вона забезпечує його більшою кількістю монет, ніж варіант підривати систему та перспективу власного багатства.

7. Використання дискового простору

Щойно остання транзакція в монеті буде підтверджена достатньою кількістю блоків, то попередні транзакції можна видалити з метою економії дискового простору. Оскільки всі транзакції хешуються в дереві Меркла [7][2][5], то щоб не розривати хеш блока, в блок записується лише корінь дерева. Після цього старі блоки можна згуртувати, видаляючи зайві гілки дерева. Немає потреби зберігати внутрішні хеші.



Транзакції, хешовані в дерево Меркла

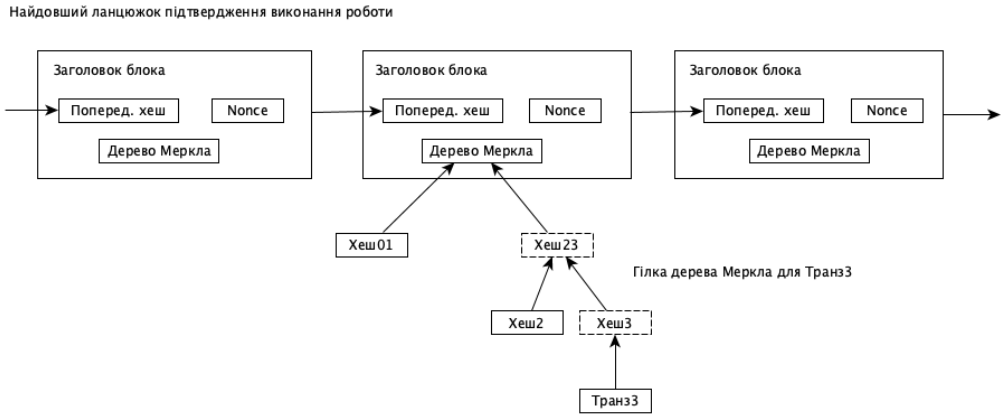


Вигляд дерева після видалення транз. 0–2 з блока

Заголовок блока без транзакцій буде займати приблизно 80 байт. Припустимо, що блоки генеруються кожні 10 хвилин, отримуємо 80 байт * 6 * 24 * 365 = 4.2 Мб за рік. Для типового комп'ютера з 2 Гб оперативної пам’яті станом на 2008 рік із урахуванням закона Мура, який передбачає приріст на 1.2 Гб за рік, зберігання не стане проблемою, навіть якщо доведеться зберігати заголовки блока в пам’яті.

8. Спрощена верифікація платежів

Здійснити перевірку транзакцій без запуску цілого мережевого вузла можливо. Для цього користувачу необхідно зберігати лише заголовок блока найдовшого ланцюжка доказу виконання роботи. Відбувається перевірка усього переліку вузлів у мережі, щоб впевнитися в тому, що користувач отримав найдовший ланцюжок обчислень, та отримати гілки хеш-дерева Меркла для встановлення зв’язку транзакції з відповідним блоком. Користувач не може перевірити транзакцію самостійно, але завдяки попередній операції він може впевнитися, що цей блок і всі наступні прийняті та підтверджені мережевим вузлом.

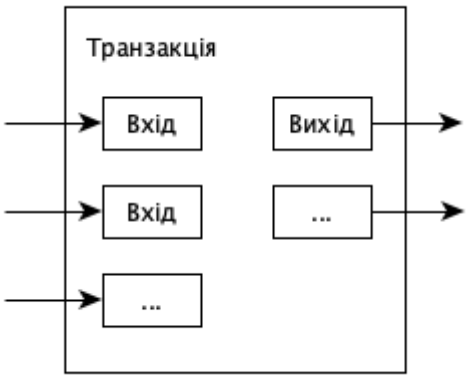


На такий метод верифікації можна покладатися доти, доки чесні вузли в своїй більшості контролюють мережу, у протилежному випадку така схема стане вразливою. Оскільки мережеві вузли можуть самостійно перевіряти транзакції, спрощений метод може бути використаний на користь атакуючих за допомогою сфабрикованих транзакцій у момент, коли чесні вузли будуть у меншості. Одна зі стратегій захисту від таких дій – приймати сповіщення тривоги від мережевих вузлів у випадку знаходження блоку з недостовірними даними. У такому випадку клієнту доведеться завантажувати

неправильний блок повністю, щоб самостійно перевірити його некоректність. Компанії, яким часто доводиться приймати платежі, скоріше за все, захочуть керувати власними вузлами для більш незалежного захисту та прискореної перевірки.

9. З'єднання та розділення суми транзакції

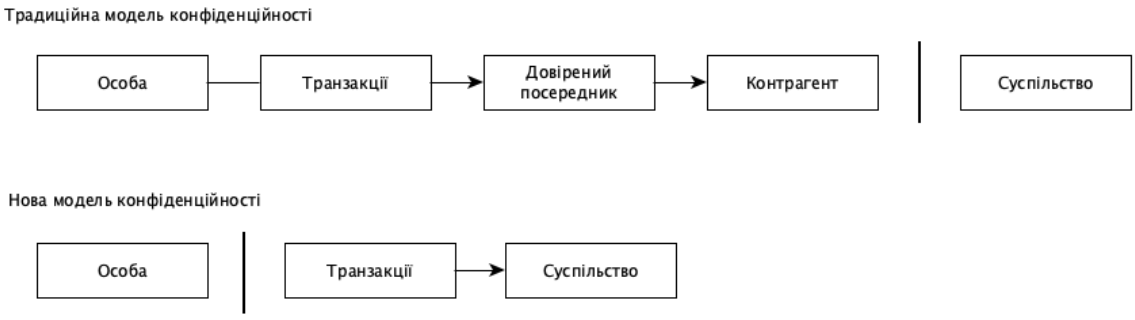
Незважаючи на можливість опрацьовувати окремі монети, виділяти окрему транзакцію на кожен цент – дуже затратний підхід. У складі транзакції є декілька входів та виходів для того, щоб мати змогу її об'єднати чи розділити. Звичайна транзакція буде мати один вхід від більшої попередньої транзакції або ж декілька, які об'єднують менші транзакції та максимум два виходи: один для платежу, інший – для повернення решти відправнику за необхідності.



Слід зауважити, що приріст зв'язків, при якому платіж залежить від декількох інших транзакцій, які, в свою чергу, теж мають багато залежностей, зовсім не проблема, оскільки немає необхідності отримувати повну копію історії транзакції.

10. Конфіденційність

Традиційна банківська модель досягає потрібного рівня конфіденційності, надаючи доступ до інформації тільки для сторін та посередника. Необхідність відкрито публікувати всю історію транзакцій виключає такий варіант, але конфіденційність даних все ще можна зберегти шляхом обмеження інформації з іншого боку. Мова йде про анонімність публічних ключів: кожен може побачити, що хтось комусь відправив платіж, але без прив'язки інформації про транзакцію до конкретних осіб. Приблизно такий самий принцип реалізований на фондових біржах – вони відкривають інформацію про час та об'єм приватних торгів, але без оприлюднення сторін.



У якості додаткового захисту до кожної транзакції має бути застосована нова пара ключів, щоб приховати зв'язок зі спільним власником. Але деяких зв'язків все ще не уникнути в схемі з транзакціями з декількома входами, які показують свою приналежність до одного й того ж власника. Якщо можна буде таким чином виявити власника, то ці зв'язки виведуть на всі транзакції, які йому належать.

11. Розрахунки

Розглянемо сценарій, у якому атакуючий намагається згенерувати альтернативний варіант ланцюга швидше, ніж чесний вузол. Навіть якщо йому вдасться, то це все одно не дає йому можливості вносити довільні зміни в систему, в тому числі робити гроші з нічого або ж отримати кошти, які йому не належали. Вузли не приймуть недостовірну транзакцію-платіж, а також блок, який її містить. Атакуючий зможе лише спробувати змінити одну зі своїх власних транзакцій, щоб повернути назад гроші, які він нещодавно витратив.

Таке змагання між чесним та атакуючим вузлом можна описати як “біноміальне випадкове блукання”. У нашому випадку успіх події – коли чесний ланцюг подовжується на один блок (відрив +1), а невдалою подія називатиметься тоді, коли наступний блок створює атакуючий, чим зменшує розрив із ним на одиницю (-1).

Вірогідність того, що атакуючий набуде перевагу з такого дефіциту аналогічна задачі про розорення гравця. Припустимо, що гравець із нескінченним кредитом починає гру з дефіцитом та використовує нескінченну кількість спроб, намагаючись подолати збитки. Ми можемо розрахувати вірогідність того, що він колись досягне беззбитковості або навіть набуде перевагу в порівнянні з чесними вузлами [8]:

p = ймовірність чесного вузла знайти наступний блок
q = ймовірність атакуючого знайти наступний блок
q_z = ймовірність того, що атакуючий зможе надолужити різницю в z блоків

$$q_z = \begin{cases} 1 & if\ p \le q \\ (q/p)^z & if\ p > q \end{cases}$$

Враховуючи припущення, що p > q, то з ростом числа блоків, на яке відстає атакуючий, ймовірність експоненційно зменшується. Оскільки всі ставки проти нього, то без вдалого відриву з самого початку його шанси на успіх мізерно малі.

Зараз ми розглянемо, скільки часу отримувачу нового платежа необхідно очікувати, щоб переконатися, що колишній власник не може вносити зміни до транзакції. Припустимо, що атакуючий дозволяє отримувачу певний час думати, що платіж насправді був проведений, та після певного періоду часу повернути його собі. Отримувач буде проінформований про такий випадок, але відправник сподіватиметься, що буде запізно.

Отримувач генерує нову пару ключів і надає відкритий ключ прямо перед підписанням. Цей підхід попереджає передчасне виконання ланцюжку блоків, працюючи над ним безперервно, доки не пощастить виконати транзакцію в момент, коли він буде доволі далеко. Щойно транзакція буде відправлена, нечесний відправник починає таємно обчислювати паралельний ланцюжок, що містить альтернативну версію його транзакції.

Отримувач очікує додавання транзакції до блоку та подальшого приєднання блоків z після неї. Йому не відомо, на якому саме етапі знаходиться атакуючий, але якщо знати про середню швидкість генерації чесних блоків, то потенційний прогрес атакуючого буде визначатися як розподіл Пуассона з математичним сподіванням:

$$\lambda = z\frac{q}{p}$$

Для визначення ймовірності того, що атакуючий зможе обійти чесні вузли, ми примножуємо пуассонівський розподіл кожної суми прогресії на ймовірність того, що він зможе надолужити різницю:

$$\sum_{k=0}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} \cdot \begin{cases} (q/p)^{(z-k)} & if\ k \le z \\ 1 & if\ k > z \end{cases}$$

Після перегрупування доданків та виключення нескінченного ряду отримуємо...

$$1 - \sum_{k=0}^{z} \frac{\lambda^k e^{-\lambda}}{k!} \left(1 - (q/p)^{(z-k)}\right)$$

Код мовою C виглядає наступним чином…

```
#include <math.h>
double AttackerSuccessProbability(double q, int z)
{
    double p = 1.0 - q;
    double lambda = z * (q / p);
    double sum = 1.0;
    int i, k;
    for (k = 0; k <= z; k++)
    {
        double poisson = exp(-lambda);
        for (i = 1; i <= k; i++)
            poisson *= lambda / i;
        sum -= poisson * (1 - pow(q / p, z - k));
    }
    return sum;
}
```

При запуску деяких значень ми можемо бачити, що ймовірність експоненційно падає з ростом z.

```
q=0.1
z=0    P=1.0000000
z=1    P=0.2045873
z=2    P=0.0509779
z=3    P=0.0131722
z=4    P=0.0034552
z=5    P=0.0009137
z=6    P=0.0002428
z=7    P=0.0000647
z=8    P=0.0000173
z=9    P=0.0000046
z=10   P=0.0000012
q=0.3
z=0    P=1.0000000
z=5    P=0.1773523
z=10   P=0.0416605
z=15   P=0.0101008
z=20   P=0.0024804
z=25   P=0.0006132
z=30   P=0.0001522
z=35   P=0.0000379
z=40   P=0.0000095
z=45   P=0.0000024
z=50   P=0.0000006
```

Результати для P < 0.1%...

```
P < 0.001
q=0.10   z=5
q=0.15   z=8
q=0.20   z=11
q=0.25   z=15
q=0.30   z=24
q=0.35   z=41
q=0.40   z=89
q=0.45   z=340
```

12. Висновок

Ми запропонували систему для електронних транзакцій, не засновану на довірі. Проект починався зі звичайного фреймворка монет на основі цифрових підписів, який жорстко контролює власника, але неповноцінність такого підходу полягала в задачі повторних витрат. Для її розв'язання ми запропонували пірингову мережу з перевіркою виконання роботи для запису до відкритої історії транзакцій. Якщо чесні вузли контролюватимуть більшість обчислювальної мережі, то змінити цю історію атакуючим вузлом буде майже неможливо.

Сильна сторона мережі полягає в її простоті. Всі вузли працюють одночасно, час від часу обмінюючись інформацією для узгодження роботи. Немає необхідності в проведенні їх ідентифікації, тому що повідомлення надсилаються не конкретним маршрутом, а за принципом негарантованої

доставки. Вузли можуть покидати мережу і підключатися знову в довільний час, приймаючи ланцюжок доказу виконання роботи як підтвердження того, що відбулося за період відсутності вузла. Вузол використовує потужність обчислювальної системи, щоб прийняти правильний блок і продовжити над ним роботу, або щоб відмовитися від блоку у випадку отримання неправильних даних. Будь-які необхідні правила та додаткові установки можуть бути реалізовані за допомогою вищенаведеного механізму.

Перелік посилань

1. [1] W. Dai, "b-money," http://www.weidai.com/bmoney.txt, 1998.
2. [2] H. Massias, X.S. Avila, and J.-J. Quisquater, "Design of a secure timestamping service with minimal trust requirements," In *20th Symposium on Information Theory in the Benelux*, May 1999.
3. [3] S. Haber, W.S. Stornetta, "How to time-stamp a digital document," In *Journal of Cryptology*, vol 3, no 2, pages 99-111, 1991.
4. [4] D. Bayer, S. Haber, W.S. Stornetta, "Improving the efficiency and reliability of digital time-stamping," In *Sequences II: Methods in Communication, Security and Computer Science*, pages 329-334, 1993.
5. [5] S. Haber, W.S. Stornetta, "Secure names for bit-strings," In *Proceedings of the 4th ACM Conference on Computer and Communications Security*, pages 28-35, April 1997.
6. [6] A. Back, "Hashcash - a denial of service counter-measure," http://www.hashcash.org/papers/hashcash.pdf, 2002.
7. [7] R.C. Merkle, "Protocols for public key cryptosystems," In *Proc. 1980 Symposium on Security and Privacy*, IEEE Computer Society, pages 122-133, April 1980.
8. [8] W. Feller, "An introduction to probability theory and its applications," 1957.