НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
ім. Ігоря Сікорського

Навчально-науковий комплекс «Інститут прикладного системного аналізу»
(повна назва інституту/факультету)

Кафедра Системного проектування
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
“___”_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра

з напряму підготовки 6.050101 Комп’ютерні науки
(код і назва)

на тему: Створення Bitcoin гаманця із використанням технології Blockchain_____
_____

Виконав (-ла): студент (-ка) __IV__ курсу, групи _ДА-31__
(шифр групи)

Сидоренко Захар Андрійович
(прізвище, ім’я, по батькові) (підпис)

Керівник ст. викладач, доцент, Романов Валерій Володимирович ____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант Економічний розділ проф. к.е.н. Рощина Н. В. _____ _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Нормоконтроль _____старший викладач Бритов О.А._____ ___
(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.
Студент _____
(підпис)

Київ – 2017 року

Національний технічний університет України
«Київський політехнічний інститут»
ім. Ігоря Сікорського

Інститут (факультет) ННК «Інститут прикладного системного аналізу
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.050101 Комп'ютерні науки
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
«___»_____20__ р.

ЗАВДАННЯ
на дипломну роботу студенту
Сидоренко Захару Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Створення Bitcoin гаманця із використанням технології Blockchain _____
керівник роботи Романов Валерій Володимирович, ст. викладач, доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від «10» травня 2017 р. № 1477-с

2. Термін подання студентом роботи 12 травня 2017 р.

3. Вихідні дані до роботи
Створенний bitcoin wallet має мати наступні функції:
1. Можливість зберігання bitcoin на власному рахунку;
2. Можливість передання bitcoin на будь-який інший гаманець;
3. Можливість обміну bitcoin на гривню та навпаки.

4. Зміст роботи
1. Дослідити та вивчити основні принципи побудови технології Blockchain;
2. Проаналізувати існуючі програми, дослідити існуючі рішення та підходи для створення bitcoin wallet та вибрати ті, що найкраще підходять для реалізації;

3. Розробити план побудови та використання bitcoin wallet відповідно до вимог замовника.
4. Протестувати правильність та зручність використання bitcoin wallet.
5. Аналіз розробленого продукту, оцінка використаних та досліджених підходів, визначення переваг і недоліків створеного bitcoin wallet.

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

1. Архітектура bitcoin wallet – плакат;
2. Загальна схема побудови Blockchain – плакат;
3. Діаграма процесів bitcoin wallet – плакат.

6. Консультанти розділів роботи*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічна частина	Рощина Н. В, проф. д.е.н.		

7. Дата видачі завдання 01.02.2017

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	01.02.2017	
2	Збір інформації	15.02.2017	
3	Дослідження предметної області та існуючих рішень	28.02.2017	
4	Вивчення підходів до побудови bitcoin wallet	10.03.2017	
5	Розробка back-end частини bitcoin wallet	25.04.2017	
6	Розробка інтерфейсу	30.04.2017	
7	Тестування прототипу	15.05.2017	
8	Оформлення дипломної роботи	31.05.2017	
9	Отримання допуску до захисту	10.06.2017	

Студент _____ _____
(підпис) (ініціали, прізвище)

Керівник роботи _____ _____
(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено керівника дипломної роботи.

АНОТАЦІЯ

бакалаврської дипломної роботи Сидоренко Захара Андрійовича на тему «Створення bitcoin wallet на базі технології Blockchain»

Метою дипломної роботи є на основі існуючих варіантів реалізації bitcoin wallet розробити свій варіант електронного гаманця. На сьогодні існує багато статей про Blockchain та його впровадження у різноманітні установи, в яких є транзакції та реєстри. Було виконано огляд існуючих аналогів готового продукту дипломної роботи – електронних гаманців та зроблено висновки щодо майбутньої реалізації продукту. Створено bitcoin wallet з можливістю приймати та передавати bitcoin. Вдосконаливши продукт можна його прив’язати до API українського банку та використовувати його не лише як гаманець, а і як обмінний пункт.

Загальний обсяг роботи: сторінки, 17 ілюстрацій, 6 таблиць та 17 бібліографічних найменувань.

Ключові слова: Blockchain, bitcoin, bitcoin wallet, API, криптовалюта, електронні гроші, SWIFT.

АННОТАЦИЯ

бакалаврской дипломной работы Сидоренко Захара Андреевича на тему «Создание bitcoin wallet на базе технологии Blockchain»

Целью дипломной работы является на основе существующих вариантов реализации bitcoin wallet разработать свой вариант электронного кошелька. Сегодня существует много статей о Blockchain и его внедрение в различные учреждения, в которых существуют транзакции и реестры. Было выполнено обзор существующих аналогов готового продукта дипломной работы - электронных кошельков и сделаны выводы относительно будущей реализации продукта. Создан bitcoin wallet с возможностью принимать и передавать bitcoin. Усовершенствовав продукт можно его привязать к API украинского банка и использовать его не только как кошелек, но и как обменный пункт.

Общий объем работы: страницы, 17 иллюстраций, 6 таблиц и 17 библиографических наименований.

Ключевые слова: blockchain, bitcoin, bitcoin wallet, API, криптовалюта, электронные деньги, SWIFT.

ABSTRACT

to the bachelor thesis by Sydorenko Zakhar Andreyevich on "Creating bitcoin wallet based on Blockchain technology"

The purpose of the thesis is based on the existing options for the implementation of bitcoin wallet to develop its own version of the electronic wallet. Today, there are many articles on Blockchain and its implementation in various institutions in which there are transactions and registries. A review of existing analogs of the finished product of the thesis work - electronic wallets was made and conclusions were made regarding future product realization. Created bitcoin wallet with the ability to receive and transmit bitcoin. Having improved the product it is possible to link it to the API of the Ukrainian bank and use it not only as a purse but also as an exchange office.

The total amount of work: pages, 17 illustrations, 6 tables and 17 bibliographical names.

Keywords: blockchain, bitcoin, bitcoin wallet, API, cryptocurrency, electronic money, SWIFT.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

Blockchain	Побудований за певними правилами ланцюжок з формованих блоків транзакцій.
Bitcoin	Пірінгова платіжна система, яка використовує однойменну розрахункову одиницю і однойменний протокол передачі даних.
Bitcoin wallet	Гаманець для збереження та передавання Bitcoin.
API	(англ. application programming interface) набір готових класів, процедур, функцій, структур і констант, що надаються додатком (бібліотекою, сервісом) або операційною системою для використання у зовнішніх програмних продуктах.
Криптовалюта	Цифрові рахункові одиниці, облік яких децентралізований.
SWIFT	(англ. Society for Worldwide Interbank Financial Telecommunications) міжнародна міжбанківська система передачі інформації та здійснення платежів.
ARIS Objective Diagram	(англ. Architecture of Integrated Information Systems Objective Diagram) діаграма архітектури інтегрованої інформаційної системи.
DFD (IDEF0)	(англ. Data flow diagram) діаграма потоків даних.

ВСТУП

На сьогодні існує дуже велика кількість різноманітних криптовалют, але найпопулярнішою є Bitcoin. Ця валюта вже офіційно визнана в Японії та за допомогою неї можна розрахуватися за комунальні послуги. Наразі в Інтернеті є багато варіантів реалізації біткоін гаманців.

Сьогодні є два методи, як зробити bitcoin wallet:

- Поставити окрему програму-клієнт на свій ПК, налаштувати її і користуватися наданими можливостями;
- Використовувати для зберігання коштів сторонні сервіси.

Програми-клієнти. При виборі цього варіанту варто бути готовим особисто нести відповідальність за свої гроші і забезпечувати збереження коштів. Завантажити необхідний софт можна на офіційному ресурсі криптовалюта - bitcoin.org. Так на вибір пропонується безліч різних сховищ, серед яких MultiBit, Armory, Bitcoin Core та інші.

Онлайн-сервіси. Ще один варіант створення bitcoin wallet - користування послугами сторонніх сервісів. Тут є безліч варіантів - exmo, coinbase, instawallet, blockchain і інші. Перед тим як переводити гроші на новий гаманець біткоіни, необхідно розібратися з принципами роботи сервісу, вжитими заходами безпеки, можливостями відновлення і видалення біткоіни-гаманця. Як це зробити описується в спеціальному розділі сервісу.

Метою дипломної роботи є аналіз існуючих bitcoin wallet, обрання найкращого варіанту та розробка на його базі власного рішення.

Об’єктом дослідження є існуючі онлайн та офлайн рішення:

- Bitcoin Core — це повноцінний клієнт, що становить основу мережі. Для нього характерний високий рівень безпеки, конфіденційності та

стабільності. Однак, у нього менше опцій і він займає багато місця на диску;

- Bitcoin Knots — це повний клієнт Bitcoin і створює основу мережі. Він забезпечує високий рівень безпеки, конфіденційності та стабільності. Також він включає в себе більш складні функції, ніж Bitcoin Core, але вони не так добре перевірені. Він використовує багато місця і пам'яті;
- GreenBits — це швидкий і простий у використанні гаманець. Йому властиві високий рівень безпеки а також двохфакторна аутентифікація. Є мобильна версія цього гаманця;
- MultiBit HD — це дуже простий клієнт, який є швидким і простим у використанні. З вбудованим Трезор і підтримкою Tor, він синхронізується безпосередньо з мережею Bitcoin. Дуже популярний серед недосвідчених користувачів;
- Armory — це просунутий біткойн-клієнт, який розширює функціонал для досвітчених біткойн-користувачів. Він пропонує багато функцій щодо шифрування і створення резервних копій;
- Electrum — швидкий і простий у використанні і вимагає мало ресурсів. Використовує віддалені сервера, які обробляють найбільш складні операції, дозволяє вам відновити гаманець за допомогою пароля;
- mSIGNA — це просунутий гаманець, який поєднує швидкість, простоту і зручність з відмінною захищеністю. Він підтримує транзакції з декількома підписами.

Потенційні застосування та практична цінність результатів дипломної роботи:

1. Біткоін гаманець може використовуватися будь-ким для проведення платіжних операцій з Bitcoin;
2. Сформовано основі концепти, на які потрібно звернути увагу при проектуванні власного біткоін гаманця;

3. При інтеграції цього гаманця із банківським API можливо буде його використовувати як обмінний пункт для обміну біткоіни на гривні та навпаки

1. BLOCKCHAIN

Blockchain - це технологія, яка набрала популярності завдяки її використанню при побудові криптовалюти Bitcoin (рис 1.1).

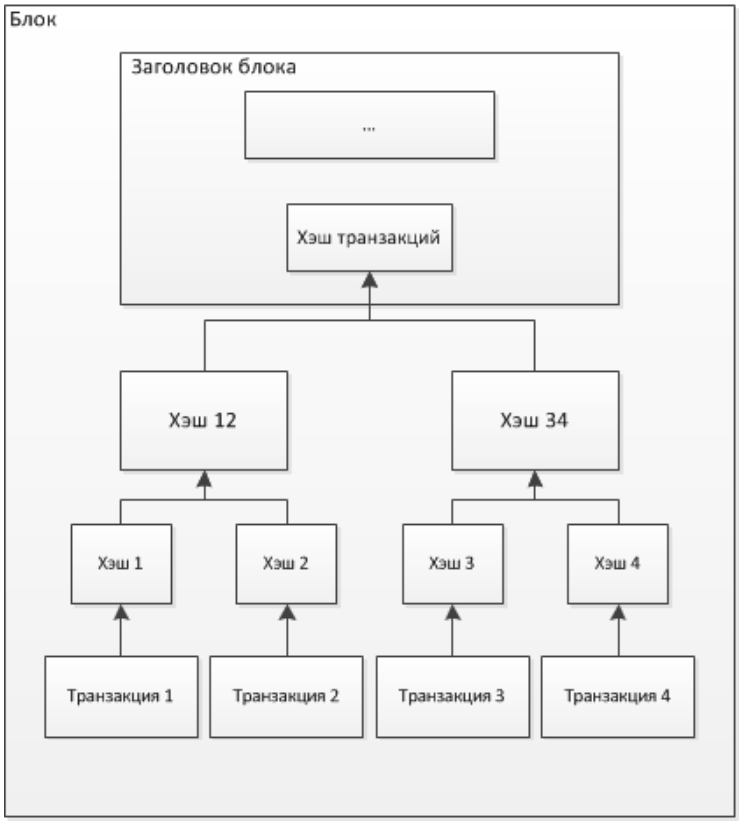


Рис1.1 Схема отримання хешу транзакцій

Основа технології blockchain - в розподіленому зберіганні інформації. Це дозволяє зберігати важливу інформацію одночасно на багатьох серверах (у всіх учасників мережі), при цьому зберігати відкрито і безпечно. Наприклад, на базі цієї технології можна зберігати як історію банківських транзакцій клієнтів, так і базу контрактів, результати голосувань, відбитків пальців або історій хвороб. Інформацію, яка одночасно зберігається у багатьох місцях неможливо підробити, неможливо вкрасти, тому що оригінальні записи тут же можуть бути відновлені з сусідніх джерел.

Провідні американські і європейські банки вже відкрили проекти по використанню даної технології. Також до них приєдналася і міжнародна платіжна система Visa, яка в використання технології бачить можливість

прискорення і здешевлення платежів. Також blockchain може значно знизити ризики шахрайства.

Blockchain - це принципово нова надійна технологія зберігання записів, яка може кардинально змінити підхід до формування і зберігання баз даних.

Blockchain є ланцюжком блоків даних, які створюються і зберігаються на комп'ютерах учасників ланцюжка. Всі учасники мережі діляться на дві категорії: звичайні користувачі, які створюють нові записи, і Майнер, які створюють блоки. Майнер перевіряють записи, які створюють звичайні користувачі, формують з них блоки, а потім розсилають ці блоки по мережі. Звичайні користувачі отримують ці блоки і зберігають їх у себе в комп'ютері. Учасники Blockchain-мережі мають доступ до інших комп'ютерів мережі, завдяки чому можна обмінюватися даними. Кожен користувач перевіряє коректність нових даних. Якщо вони достовірні, він зберігає їх і передає далі по мережі.

Технологія Blockchain має величезний потенціал з точки зору спрощення та підвищення ефективності за рахунок створення принципово нової інфраструктури фінансових сервісів. Ця технологія може успішно використовуватися банками для проведення внутрішніх взаєморозрахунків і здійснення міжбанківських операцій, а також при проведенні мікроплатежів між фізичними особами.

При цьому вона може значно спростити відстеження підозрілих транзакцій і в цілому підвищити прозорість угод. По суті це технологія розподіленого підтвердження транзакцій, яка представляє собою величезну розподілену базу даних. При цьому перевіркою достовірності транзакцій займаються самі учасники, вони ж підтверджують їх справжність та формують блоки записів.

Такий підхід цікавий перш за все тим, що відпадає необхідність в посередниках в особі платіжних систем, що здійснюють процесування транзакцій і, як наслідок, підвищується швидкість обробки операцій і знижується вартість для кінцевого споживача.

За деякими оцінками, використання Blockchain дозволить банкам заощаджувати близько 20 мільярдів доларів за рахунок відмови від послуг посередників при здійсненні транзакцій. Blockchain може стати реальною альтернативою системі SWIFT, яка на даний момент є не дуже гнучкою і досить дорогий.

Але перейти на нову технологію вдасться не так швидко. На це є кілька причин і перш за все - невизначеність у правовій і регуляторної області. Крім того, широкомасштабне впровадження цієї технології зажадає значних зусиль в частині стандартизації.

1.1 Блок транзакцій

Блок транзакцій - спеціальна структура для запису групи транзакцій в системі біткойнів і аналогічних їй.

Щоб транзакція вважалася достовірною («підтвердженої»), її формат і підписи повинні перевірити і потім групу транзакцій записати в спеціальну структуру - блок. Інформацію в блоках можна швидко перевірити ще раз. Кожен блок завжди містить інформацію про попередньому блоці. Всі блоки можна вибудувати в один ланцюжок, яка містить інформацію про всіх скоєних коли-небудь операціях в цій базі. Найперший блок в ланцюжку - первинний блок (англ. Genesis block) - розглядається як окремий випадок, так як у нього відсутня батьківський блок.

Блок складається з заголовка і списку транзакцій. Тема блоку включає в себе свій хеш, хеш попереднього блоку, хеші транзакцій і додаткову службову

інформацію. В системі біткойнів першої транзакцією в блоці завжди вказується отримання комісії, яка стане нагородою користувачеві за створений блок.

Далі йдуть всі або деякі з останніх транзакцій, які ще не були записані в попередні блоки. Для транзакцій в блоці використовується деревоподібна хешування, аналогічне формування хеш-суми для файлу в протоколі BitTorrent. Транзакції, крім нарахування комісії за створення блоку, містять всередині атрибута input посилання на транзакцію з попереднім станом даних (в системі біткойнів, наприклад, дається посилання на ту транзакцію, по якій були отримані витрачаються біткойни). Комісійні транзакції можуть містити в атрибуті будь-яку інформацію (для них це поле зветься англ. Coinbase parameter), оскільки у них немає батьківських транзакцій.

Створений блок буде прийнятий іншими користувачами, якщо числове значення хешу заголовка одно або нижче певного числа, величина якого періодично коригується. Так як результат хешування (функції SHA-256) є незворотним, немає алгоритму отримання бажаного результату, крім випадкового перебору. Якщо хеш не задовольняє умові, то в заголовку змінюється параметр nonce і хеш перераховується. Зазвичай потрібна велика кількість перерахунків. Коли варіант знайдений, вузол розсилає отриманий блок іншим підключеним вузлів, які перевіряють блок. Якщо помилок немає, то блок вважається доданим в ланцюжок і наступний блок повинен включити в себе його хеш.

Величина цільового числа, з яким порівнюється хеш, в системі біткойнів коригується через кожні 2016 блоків. Заплановано, що вся мережа системи біткойнів повинна витрачати на генерацію одного блоку приблизно 10 хвилин, на 2016 блоків - близько двох тижнів. Якщо 2016 блоків сформовані швидше, то мета трохи зменшується і досягти її стає важче, в іншому випадку мета збільшується. Зміна складності обчислень не впливає на надійність мережі біткойнів і потрібно лише для того, щоб система генерувала блоки майже з

постійною швидкістю, що не залежить від обчислювальної потужності учасників мережі.

1.2 Ланцюжок блоків

Блоки одночасно формуються безліччю «Майнерів». Задовольняють критеріям блоки відправляються в мережу, включаючись в розподілену базу блоків. Регулярно виникають ситуації, коли кілька нових блоків в різних частинах розподіленої мережі називають попереднім один і той же блок, тобто ланцюжок блоків може гілкуватися. Спеціально чи випадково можна обмежити ретрансляцію інформації про нові блоках (наприклад, одна з ланцюжків може розвиватися в рамках локальної мережі). У цьому випадку можливо паралельне нарощування різних гілок. У кожному з нових блоків можуть зустрічатися як однакові транзакції, так і різні, що увійшли тільки в один з них. Коли ретрансляція блоків відновлюється, Майнер починають вважати головною ланцюжок з урахуванням рівня складності хешу і довжини ланцюжка. У разі рівного розподілу складності і довжини перевага віддається тій ланцюжку, кінцевий блок якої з'явився раніше. Транзакції, що увійшли тільки в відхилену гілку (в тому числі по виплаті винагороди), втрачають статус підтверджених. Якщо це транзакція з передачі біткойнів, то вона буде поставлена в чергу і потім включена в черговий блок. Транзакції отримання винагороди за створення відсічені блоків не дублюються в іншій гілці, тобто «зайві» біткойни, виплачені за формування відсічені блоків, не отримують подальших підтверджень і «втрачаються».

Таким чином, ланцюжок блоків містить історію володіння, з якою можна ознайомитися, наприклад, на спеціалізованих сайтах.

Розподілена база даних Blockchain формується як безперервно зростаюча ланцюжок блоків із записами про всі транзакції. Копія бази або її частини одночасно зберігається на безлічі комп'ютерів і синхронізуються відповідно до формальних правил побудови ланцюжка блоків. Інформація в блок не

шифрувати і доступна у відкритому вигляді, але захищена від змін криптографически через хеш-ланцюжка.

База публічно зберігає в незашифрованому вигляді інформацію про всі транзакції, що підписуються за допомогою асиметричного шифрування. Для запобігання багаторазової витрати однієї і тієї ж суми використовуються мітки часу, реалізовані шляхом розбиття БД на ланцюжок спеціальних блоків, кожен з яких, в числі іншого, містить в собі хеш попереднього блоку і свій порядковий номер. Кожен новий блок здійснює підтвердження транзакцій, інформацію про яких містить і додаткове підтвердження транзакцій у всіх попередніх блоках ланцюжка. Змінювати інформацію в блоці, який вже знаходиться в ланцюзі, не практично, так як в такому випадку довелося б редагувати інформацію в усіх наступних блоках. Завдяки цьому успішна double-spending атака (повторна трата раніше витрачених коштів) на практиці вкрай малоймовірна.

Найчастіше, умисне зміна інформації в будь-який з копій бази або навіть в досить великій кількості копій нічого очікувати визнано істинним, тому що не буде відповідати правилам. Деякі зміни можуть бути прийняті, якщо будуть внесені в усі копії бази (наприклад, видалення декількох останніх блоків через помилки в їх формуванні).

Для більш наочного пояснення механізму роботи платіжної системи Сатоси Накамото ввів поняття «цифрова монета», визначивши його як ланцюжок цифрових підписів. На відміну від стандартизованих номіналів звичайних монет, кожна «цифрова монета» має свій власний номінал. Кожному біткойн-адресою може зіставлятися будь-яку кількість «цифрових монет». За допомогою транзакцій їх можна ділити і об'єднувати, при цьому зберігається загальна сума їх номіналів за вирахуванням комісії.

1.3 Розгалуження

Блоки одночасно формують безліч «Майнерів». Регулярно виникають ситуації, коли кілька нових блоків вважають попереднім один і той же блок, тобто ланцюжок блоків розгалужується. Цілком можливо обмеження обміну даними із загальною мережею - наприклад, одна з ланцюжків може розвиватися в рамках локальної мережі. У цьому випадку можливо паралельне нарощування різних гілок. Коли ретрансляція блоків відновлюється, мережа автоматично буде вважати основною (істинної) більш довгий ланцюжок. У разі рівного розподілу довжини паралельна робота триватиме до створення нового блоку - в якій з ланцюжків блок з'явиться раніше, та й стане довшим, тобто вона буде визнана основною, а робота над паралельної ланцюжком припиниться.

Транзакції, що увійшли тільки в відхилену гілку, вважаються тепер поза блоком і будуть поставлені в чергу для включення в черговий блок. Транзакції отримання винагороди за створення відсічені блоків не дублюються в іншій гілці, тобто біткойни, отримані за формування відсічені блоків, «зникають».

1.4 Підтвердження транзакцій

Поки транзакція не включена в блок, система вважає, що кількість біткойнів на якомусь адресу залишається незмінним. У цей час є технічна можливість оформити кілька різних транзакцій з передачі з однієї адреси одних і тих же біткойнів різним одержувачам. Але як тільки одна з подібних транзакцій буде включена в блок, інші транзакції з цими ж біткойнів система буде вже ігнорувати. Наприклад, якщо в блок буде включена пізніша транзакція, то більш рання буде вважатися помилковою. Є невелика вірогідність, що при розгалуженні дві подібні транзакції потраплять в блоки різних гілок. Кожна з них буде вважатися правильною, лише при відмирання гілки одна з транзакцій стане вважатися помилковою. При цьому не буде мати значення час здійснення операції.

Таким чином, попадання транзакції в блок є підтвердженням її достовірності незалежно від наявності інших транзакцій з тими ж біткойнів. Кожен новий блок вважається додатковим «підтвердженням» транзакцій з попередніх блоків. Якщо в ланцюжку 3 блоку, то транзакції з останнього блоку будуть підтверджені 1 раз, а поміщені в перший блок матимуть 3 підтвердження. Досить дочекатися декількох підтверджень, щоб звести ймовірність скасування транзакції до мінімуму.

Для зменшення впливу подібних ситуацій на мережу існують обмеження на розпорядження щойно отриманими біткойнів. Згідно сервісу blockchain.info, до травня 2015 року максимальна довжина відкинутих ланцюжків була 5 блоків. Необхідна кількість підтверджень для розблокування отриманого залежить від програми-клієнта або від вказівок приймаючої сторони. Клієнт «Bitcoin-qt» для відправки не вимагає наявності підтверджень, але у більшості одержувачів за замовчуванням виставлено вимогу 6 підтверджень, тобто реально скористатися отриманим зазвичай можна через годину. Різні онлайн-сервіси часто встановлюють свій поріг підтверджень.

Біткойни, отримані за створення блоку, протокол дозволяє використовувати після 100 підтверджень, але стандартна програма-клієнт показує комісію через 120 підтверджень, тобто зазвичай скористатися комісією можна приблизно через 20 годин після її нарахування.

1.5 Деревоподібне хешування

Деревоподібне хешування (Рис 3.1) - тип хеш-функції. Використовується для того, щоб перевіряти цілісність даних (файлів), отримати унікальний ідентифікатор файлу, а також дає можливість відновити файл.

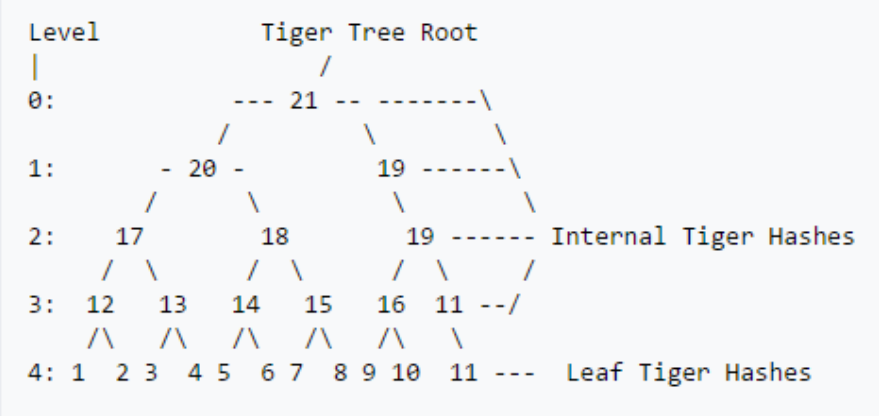


Рис 3.1 Бінарне хеш-дерево

1.6 Обчислення нового блоку

Дані поділяються на маленькі частини - блоки, які індивідуально хешуються за допомогою Leaf Tiger Hash, потім з кожної пари хешів черзі обчислюється Internal Tiger Hash. Якщо хешу немає пари, то він переноситься в нову ланцюжок без змін. Далі в ланцюжку для кожної пари знову обчислюється Internal Tiger Hash. Ця процедура повторюється до тих пір, поки не залишиться один хеш. Цей єдиний залишився Internal Tiger Hash називають Tiger Tree Root. Саме його використовують для однозначної ідентифікації файлу і вказують в різних P2P посиланнях.

1.7 Tiger

Tiger - криптографічний хеш-функція, розроблена Росом Андерсоном і Елі Біхамом в 1995 році. Tiger був призначений для особливо швидкого виконання на 64-розрядних комп'ютерах. Tiger не має патентних обмежень, може використовуватися вільно як з еталонною реалізацією, так і з її модифікаціями. Розмір значення хешу - 192 біта (Tiger / 192), хоча є також більш короткі версії для сумісності з SHA-1 (Tiger / 160) і з MD4, MD5, RIPEMD, Snefru (Tiger / 128). Швидкість роботи - 132 Мбіт / с (перевірено на одному процесорі Alpha 7000, модель 660). На сучасних процесорах значно швидше (навіть при тесті на 32-бітному AMD Sempron 3000+ швидкість близько 225 Мбіт / с).

Кількість використовуваних S-box'ов - 4. S-box виконує перетворення 8 біт в 64 біта. Тобто в кожному з них 256 64-бітних слів і загальний розмір пам'яті, необхідної для зберігання S-box'ов 4 * 256 * 8 = 8192 = 8 Кбайт. Для цього вистачає кеша більшості процесорів, хоча можуть бути складності при реалізації на мікроконтролерах.

Як і в сімействі MD4, до повідомлення додається біт "1", за яким слідують нулі. Вхідні дані діляться на n блоків по 512 біт.

Вибираємо перший 512-бітний блок. Цей блок ділиться на вісім 64-бітових слів x0, x1, ..., x7. Порядок байтів - little-endian.

Беруться три 64-бітових регістра з початковими значеннями (значення хешу h_0):

```
a = 0x0123456789ABCDEF
b = 0xFEDCBA9876543210
c = 0xF096A5B4C3B2E187
```

Тепер для переходу від значення h_i):до значення h_{i+1} виконуються наступні операції:

1. **save_abc**
2. **pass(a, b, c,5)**
3. **key_schedule**
4. **pass(c, a, b,7)**
5. **key_schedule**
6. **pass(b, c, a,9)**
7. **feedforward**

Тут save_abc зберігає значення h_i):

```
aa = a
bb = b
cc = c
```

pass(a, b, c, mul) означає:

```
round(a,b,c,x0,mul)

round(b,c,a,x1,mul)

round(c,a,b,x2,mul)

round(a,b,c,x3,mul)

round(b,c,a,x4,mul)

round(c,a,b,x5,mul)
```

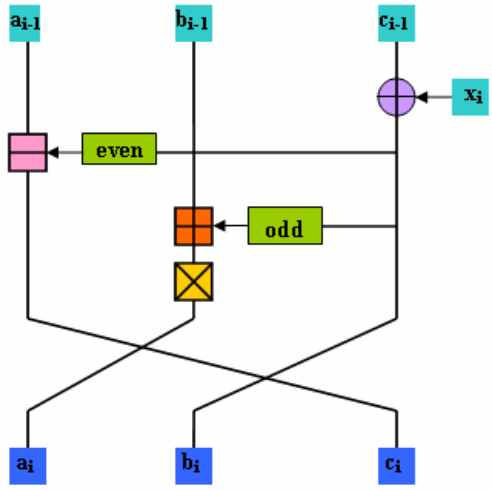


Рис 3.2 Один раунд перетворень Tiger

де round (Рис 3.2) (a, b, c, x, mul):

```
c ^= x
a -= t1[c_0] ^ t2[c_2] ^ t3[c_4] ^ t4[c_6]
b += t4[c_1] ^ t3[c_3] ^ t2[c_5] ^ t1[c_7]
b *= mul
```

c_i - i-й байт c (0 <= i <= 7);

^ - операція XOR;

ti - i-й S-box

key_schedule - генерація ключів, оборотна функція, яка відповідає за те, щоб зміна невеликого числа біт повідомлення x викликало зміна великого числа біт на наступному виконанні pass:

```
x0 -= x7 ^ 0xA5A5A5A5A5A5A5A5
x1 ^= x0
x2 += x1
x3 -= x2 ^ ((~x1)<<19)
x4 ^= x3
x5 += x4
x6 -= x5 ^ ((~x4)>>23)
x7 ^= x6
x0 += x7
x1 -= x0 ^ ((~x7)<<19)
x2 ^= x1
x3 += x2
x4 -= x3 ^ ((~x2)>>23)
x5 ^= x4
x6 += x5
x7 -= x6 ^ 0x0123456789ABCDEF
```

де << і >> - логічні зрушення вліво і вправо, ~ - інвертування

feedforward - зворотний зв'язок:

```
a ^= aa
b -= bb
c += cc
```

Основні аспекти безпеки Tiger:

- нелінійність перетворення S-box'ів;
- використання 64-бітових регістрів прискорює лавиноподібне зміна всіх трьох регістрів при зміні будь-якого біта повідомлення;
- генерація ключів забезпечує значні зміни всіх біт на наступних перетвореннях при незначній зміні повідомлення;
- множення регістра b в кожному раунді також сприяє перемішуванню і збільшує опір атакам на пов'язані ключі;
- зворотний зв'язок перешкоджає проміжним атакам днів народження.

Атака на пов'язані ключі є атакою, при якій криптоаналітик може обчислювати хеш для кількох різних значень ініціюючих векторів, які він не знає, але знає деяку взаємозв'язок між ними (наприклад, що вони відрізняються

на один біт або якась частина всіх векторів одна і та ж). Через атаки такого типу з шифрування WEP довелося перейти на WPA.

Проміжна атака днів народження - атака днів народження, застосована на проміжних раундах для досягнення бажаних хеш-значень. Хоча, на думку авторів, атаки подібного типу навряд чи приведуть до складності менше 2 ^ 96 (відповідно до парадоксом днів народження).

1.8 Висновок

У цьому розділі було проведено загальний огляд, що таке технологія Blockchain. Також було визначено основні компоненти технології, наведено сфери її застосування.

Проведено аналіз аналіз алгоритмів, які використовуються для отримання хешів нових блоків. В результаті дослідження було виявлено, що технологія Blockchain також може використовуватися про створенні платформи для смарт-контрактів. Нагадаємо, що смарт-контракт – це електронний алгоритм, що описує набір умов, виконання яких тягне за собою деякі події в реальному світі або цифрових системах. Для реалізації розумних контрактів потрібно децентралізована середовище, повністю виключає людський фактор, а для можливості використання в розумному контракті передачі вартості потрібно крипто валюта (наприклад Bitcoin).

2. BITCOIN

Біткойн (англ. Bitcoin, від bit - «біт» і coin - «монета») - пірінгова платіжна система, яка використовує однойменну розрахункову одиницю і однойменний протокол передачі даних. Для забезпечення функціонування і захисту системи використовуються криптографічні методи. Вся інформація про транзакції між адресами системи доступна у відкритому вигляді.

Проведені операції незворотні, електронний платіж між двома сторонами відбувається без посередників. Але є можливість залучення третьої сторони-гаранта за допомогою мультіподпісі. Засоби ніхто не може заморозити, навіть тимчасово, за винятком самого власника. Ці та інші розумні контракти можуть бути реалізовані за допомогою спеціальної мови сценаріїв, проте він не доступний з графічного інтерфейсу і не повний по Тьюрингу, на відміну від більш нових блокчейнових систем.

Різні автори по-різному класифікують біткойни. Найчастіше зустрічаються варіанти: криптовалюта, віртуальна валюта, цифрова валюта, електронна готівка.

Біткойни можуть використовуватися для обміну на товари або послуги у продавців, які згодні їх приймати. Обмін на звичайні валюти відбувається через онлайн-сервіс обміну цифрових валют, інші платіжні системи або обмінні пункти.

Комісія за проведення операцій призначається відправником добровільно, розмір комісії впливає на пріоритет при обробці транзакції. Зазвичай програма-клієнт підказує рекомендований розмір комісії. Транзакції без комісії можливі і також обробляються, однак не рекомендуються, оскільки час їх обробки невідомо і може бути досить велике.

Одна з головних особливостей системи - повна децентралізація: немає центрального адміністратора або будь-якого його аналога. Необхідною і достатньою елементом цієї платіжної системи є базова програма-клієнт (має відкритий вихідний код). Запущені на безлічі комп'ютерів програми-клієнти

з'єднуються між собою в однорангові з'єднання, кожен вузол якої рівноправний і самодостатній. Неможливо державне чи приватне управління системою, в тому числі зміна сумарної кількості біткойнів. Заздалегідь відомі обсяг і час випуску нових біткойнів, але розподіляються вони відносно випадково серед тих, хто використовує своє обладнання для обчислень, результати яких є механізмом регулювання і підтвердження правомочності операцій в системі «біткойнів».

2.1 Принцип роботи

Біткойни існують тільки у вигляді записів в розподіленої базі, в якій в загальнодоступному відкритому (нешифрованому) вигляді зберігаються всі транзакції, із зазначенням біткойн-адрес відправників / одержувачів, але без інформації про реального власника цих адрес. У базі немає окремих записів про поточну кількість біткойнів у будь-якого власника. Лише на підставі ланцюжків транзакцій стає зрозумілим поточну кількість біткойнів, пов'язаних з тим чи іншим біткойн-адресою. Тобто можна побачити, що на адресу надійшов 1 біткойн, а по інший транзакції на цю ж адресу надійшло 2 біткойнів, третя транзакція відправила з цієї адреси 1 біткойн. Але в базі не зберігається окремого запису, скільки всього зараз біткойнів числиться за даними адресою - просто надається можливість будь-якої миті це легко підрахувати. Такі підрахунки автоматично роблять клієнтські програми, користувач може і не помічати роздробленості інформації.

2.1.1 Ключі

Кожен користувач системи може генерувати необмежену кількість пар ключів (алгоритм ECDSA з параметром secp256k1). Розмір закритого ключа - 256 біт, а відповідного йому відкритого ключа - 512 біт.

Основне використання ключів - створення біткойн-адреси і підтвердження правомочності формування транзакцій. Але вони можуть використовуватися і для цифрового підпису або шифрування при листуванні.

Створення нової пари ключів автономно і не вимагає підключення до мережі або Інтернетом. Створені ключі зазвичай зберігають в спеціальному зашифрованому файлі wallet.dat («гаманці»). Користувач придумує пароль тільки для доступу до інформації з файлу «wallet.dat», тобто для доступу до своїх парам ключів. Для розпорядження біткойнів наявність цього файлу не є обов'язковим - в більшості випадків буде достатньо будь-яким чином отримати закритий ключ.

Зберігати ключі можна на будь-якому носії, не тільки на карті пам'яті, але і в паперовому вигляді. Існують онлайн гаманці, наприклад, Blockchain.info, Circle Snapcard або Coinbase, які досить прості у використанні. Але проблеми з сайтом такого сервісу можуть призводити до втрат.

2.1.2 Адресація

Адреси створюються за допомогою генерації асиметричною пари криптографічних ключів для чого не потрібне підключення до інтернету. Людина може мати необмежену кількість адрес, створюючи їх за своїм бажанням. Кожному потенційному адресою відповідає баланс, виражений в біткойнів. Всі адреси з ненульовим балансом записані в децентралізовану ланцюжок блоків транзакцій, захищену від змін. При створенні адреси, його баланс завжди нульовий і може бути поповнений або відправкою біткойнів з інших адрес, або шляхом створення нових біткойнів і комісійних зборів за рахунок Майнінг.

Біткойн-адреса є послідовністю байт, отриманих в результаті перетворення відкритого ключа. Найчастіше кодуванням Base58 адресу записують як рядок довжиною до 34 літер латинського алфавіту і цифр. Перший символ адреси є завжди одиницею для звичайних адрес або трійкою для адрес створених з використанням мультипідписі. Частина символів є контрольною сумою, яка перевіряє правильність основної частини адреси, яка, в свою чергу, є повністю випадковим результатом операцій хешування відкритого ключа. Такі адреси як 1111111111111111111114oLvT2 або 1BitcoinEaterAddressDontSendf59kuE є

коректними, проте наявність у кого-небудь відповідного їм приватного ключа обчислювально нездійсненно, тому такі адреси можуть використовуватися наприклад для знищення біткойнів. Адреси з невеликою кількістю невипадкових символів можуть бути отримані в розпорядження шляхом перебору.

Адреси також можуть бути відображені у вигляді QR-кодів і інших штрихкодів, придатних для машинного зчитування, наприклад, мобільними пристроями.

Якщо секретний ключ загублений, біткойн-мережа не прийме ніяких інших доказів права власності. Створити для існуючого адреси новий ключ не вийде, так як унікальної парі ключів завжди відповідає свою адресу. Біткойни, пов'язані з адресою, для якого немає закритого ключа, стають недоступними, фактично втрачаються. В кінці листопада 2013 року на BBC пройшов сюжет про британця, який на місцевому звалищі шукав викинутий ним раніше свій старий комп'ютерний жорсткий диск з секретним ключем до адресою, на якому ще з 2009 року зберігалося 7,5 тис. Біткойнів. З новин британець дізнався про значне зростання курсу біткойнів і «усвідомив, що накоїв». На момент «розкопок» вартість втрачених біткойнів перевищила 7,5 млн доларів.

2.2 Конфіденційність

Традиційна модель досягає секретності шляхом обмеження доступу до інформації. Про операцію можуть знати тільки дві сторони і банк. В системі «біткойнів» всі транзакції публічні, зберігаються у відкритому нешифрованому вигляді, а таємність досягається відсутністю персоніфікації власників адрес. Сатосі Накамото для конфіденційності рекомендує створювати окремі адреси для кожної транзакції. Це ускладнює зіставлення адрес з одним власником.

На думку ряду авторів, біткойн-адреси є псевдонімами користувачів системи. Якщо зв'язати біткойн-адреса з конкретною людиною, то зникає анонімність всіх транзакцій з використанням цієї адреси. У липні 2011 року було показано, що на основі загальнодоступної інформації можливо зв'язати багато

відкриті ключі як один з одним, так і з певною зовнішньої ідентифікуючої інформації. Обмінники, магазини і сховища гаманців, спираючись на e-mail, IP, номери кредитних карт і т. П., Здатні виявляти і персоніфікувати значну частину операцій.

Додаткову анонімність операцій з біткойнів може забезпечити використання мережі Tor, яка приховує справжню IP, але не змінює біткойн-адреси.

Також для збереження конфіденційності може бути застосований «біткойн-міксер», який в одній транзакції змішує на вході біткойни різних користувачів і виробляє одночасно багато платежів. Це ускладнює зіставлення, хто куди платив.

2.3 Транзакції

Біткойни можуть бути передані будь-кому, хто повідомить коректний біткойн-адресу або відкритий ключ. Мінімальна передана величина 10-8 біткойнів отримало назву «Сатоши» - на честь творця Сатоси Накамото, хоча сам він використовував для позначень мінімальної переданої величини слово «цент» [49]. Для передачі біткойнів поточний власник створює нову транзакцію, яка крім вказівок про кількість переданих біткойнів містить підписаний ініціатором хеш попередньої транзакції, по якій біткойни були отримані. Попередня транзакція стає «входом» поточної транзакції. Також вказується публічний ключ або біткойн-адреса нового одержувача («вихід») (див. Схематичну структуру на малюнку). Транзакція широкомовною запитом по відкритих каналах без шифрування відправляється в мережу. Решта вузли мережі, перш ніж прийняти транзакцію до обробки, перевіряють підписи. Правильність підпису свідчить, що ініціатор дійсно є власником секретного ключа для адреси «виходу».

Транзакції підтримують будь-яку кількість «входів» (посилань на попередні транзакції, в тому числі на користь різних адрес) і «виходів» (вказівки

про одержувачів). Значення з усіх «входів» підсумовуються, і сума розподіляється по «виходів».

Особливістю протоколу є неможливість взяти лише деяку частину біткойнів з «входу». Якщо на адресу було передано 2 біткойнів однієї транзакцією, то при наступній операції із зазначенням цієї транзакції в якості «входу» автоматично буде матися на увазі передача 2 біткойнів. Однак їх можна розподілити на кілька «виходів», один з яких може вказувати на цю ж адресу, тобто частина біткойнів будуть передані самому собі («здача»). Але залишок не обов'язково відправляти на адресу з вхідного списку. Наприклад, «Bitcoin-qt» відправляє кожен залишок на новий біткойн-адреса з резерву заздалегідь створених адрес.

Скасувати стандартну транзакцію неможливо, навіть при явній помилці або шахрайстві. Однак передбачено використання мультипідписів, в тому числі для угод за участю арбітра, що може забезпечити повернення біткойнів при невиконанні контрагентами обумовлених умов.

Передача біткойнів зводиться до вказівкою умов подальшого розпорядження ними. Умови формуються із застосуванням відкритих ключів. Для наступної операції з цими біткойнів потрібна відповідна електронний підпис із застосуванням секретних ключів (див. Асиметричні алгоритми шифрування), що і буде виконанням умов. Мережа перевіряє підписи парними відкритими ключами. Таким чином, розпорядитися біткойнів зможе тільки власник секретного ключа. Найбільш типовим умовою є просте зазначення біткойн-адреси, який формують на основі відкритого ключа. Умови можуть бути і іншими. Наприклад, можна зажадати використати більше цифрових підписів (тобто отримати згоду декількох сторін) або вказати відкритий ключ і IP-адреса - тоді цифровий підпис треба буде виконати на комп'ютері з обумовленим IP-адресою.

Окремі транзакції об'єднують разом з іншими транзакціями в спеціальну структуру - блок. Інформація в блоках відкрита, не шифрується, її можна швидко перевірити ще раз.

Кожен блок завжди містить свій порядковий номер і хеш попереднього блоку. Всі блоки можна вибудувати в один ланцюжок, яка містить інформацію про всіх скоєних коли-небудь операціях з біткойнів . З ними можна ознайомитися, наприклад, на спеціалізованих сайтах - браузерах ланцюжків блоків (англ. Blockchain explorer).

Перша транзакція в блоці завжди формується автоматично і передає винагороду за створення блоку. Решта наповнення блоку беруть з черги транзакцій, які ще не були записані в попередні блоки. Створює блок учасник може сам відібрати включаються в блок транзакції, наприклад, не взяти в блок транзакції без комісії.

Не всякий сформований блок буде прийнятий іншими учасниками. Потрібно, щоб числове значення хешу заголовка не перевищувало встановленого значення (параметр «складність»). Чим менше задано значення, тим менше ймовірність виконання умови. У службовій області блоку виділено місце для довільних значень. Якщо хеш заголовка незадовільний, довільні значення замінюються на нові і розрахунок хешу повторюється. Результат хеширования (функції SHA-256) непередбачуваний, тому немає алгоритму цілеспрямованої зміни довільної області для досягнення бажаного результату. Зазвичай потрібна велика кількість перерахунків. Параметр «складність» приблизно раз в два тижні автоматично встановлюється так, щоб підтримувати постійної середню швидкість створення блоків (приблизно 1 блок в 10 хвилин). Якщо блоки формуються швидше, то після перерахунку «складності» досягти мети стає важче, і навпаки. Тому зміна сумарною обчислювальною потужності мережі лише дуже незначно змінює кількість створюваних блоків.

Коли підходящий варіант хешу знайдений, вузол розсилає отриманий блок іншим підключеним вузлів для перевірки. Якщо помилок немає, то кожен вузол мережі отримав блок записує його в свій екземпляр бази.

При формуванні блоків можуть виникнути ситуації, коли кілька нових блоків вважають попереднім один і той же блок. Це явище називається розгалуженням і відбувається через одночасне формування блоків «Майнер».

До включення транзакції в блок є технічна можливість оформлення кількох різних транзакцій з передачі з однієї адреси одних і тих же біткойнів різним одержувачам. Як тільки транзакція буде включена в блок, інші транзакції з цими ж біткойнів система буде вже ігнорувати, тобто в ланцюжку блоків залишиться тільки одна транзакція. Але якщо контролювати більше 50% сумарною обчислювальною потужності мережі, то існує теоретична можливість при будь-якому порозі підтверджень формувати паралельну більш довгий ланцюжок блоків, в якій ті ж біткойни будуть передані іншому одержувачу (проблема «подвійного витрачання»). Коли мережу отримає відомості про другий ланцюжку блоків, вона стане основною, а транзакція в ній - підтвердженої, перша ж транзакція втратить підтвердження і буде вважатися помилковою. В результаті не відбудеться подвоєння біткойнів , але зміниться їх поточний власник, при цьому перший одержувач втратить біткойни без будь-яких компенсацій.

2.4 Комісійні збори

В системі «біткойнів » не передбачено обов'язкової комісії. Користувачі можуть добровільно налаштувати її розмір. Якщо сума «входів» транзакції більше суми «виходів», то різниця вважається комісією і вона дістанеться творцю блоку із даної транзакцією. Різні програми-клієнти мають свої правила і налаштування щодо комісії і найчастіше рекомендований розмір комісії вони обчислюють автоматично.

Той, хто генерує новий блок, може на свій розсуд додавати в нього транзакції з черги. Наприклад, він може відібрати тільки транзакції з комісією. Станом на початок 2015 року зазвичай 50 000 байт в блоці резервується під пріоритетні транзакції незалежно від комісії. За рахунок транзакцій з комісією величина блоку може досягати 750 000 байт. Між комп'ютерами мережі «біткойнів » встановлено обмеження швидкості в 15 кілобайт за хвилину для ретрансляції інформації про транзакції без комісії, які ще включено ні в один блок. Отже, немає гарантії, що транзакція без комісії буде включена в найближчий блок.

2.5 Майнінг

Випуск нових біткойнів децентралізований, не залежить від будь-якого регулюючого органу, обсяг емісії відомий заздалегідь (див. Рис 2.1 - Графік кількості біткойнів до 2033 року). Стандартна порція нових біткойнів додається до суми комісій з транзакцій, включених в черговий блок. Підсумкову суму в якості винагороди отримує той, хто додав черговий блок в базу транзакцій.

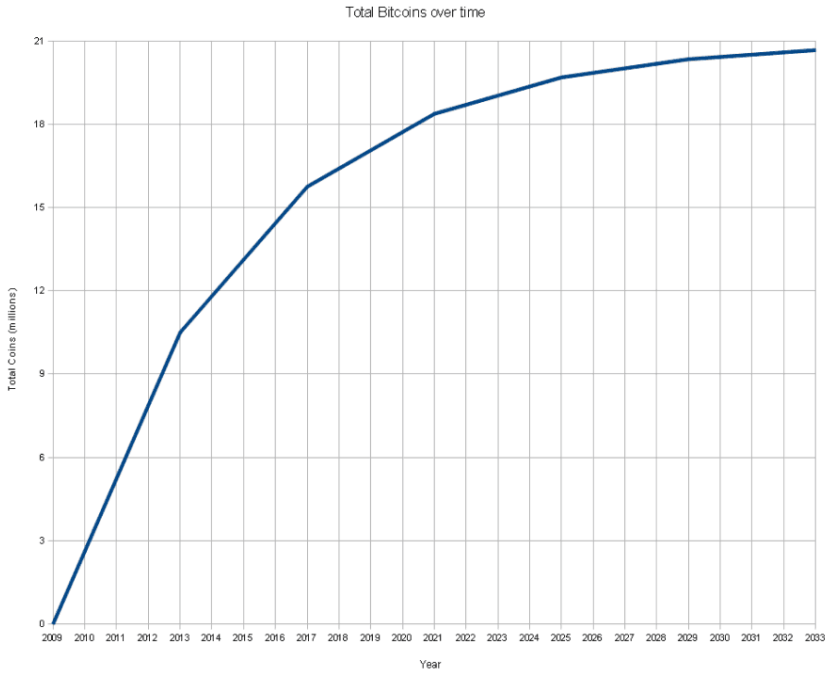


Рис 2.1 - Графік кількості біткойнів до 2033 року

Діяльність зі створення нових блоків заради можливості отримати винагороду в формі емітованих біткойнів і комісійних зборів отримала назву «Майнінг» (від англ. Mining - видобуток корисних копалин).

У перших версіях програми-клієнта була кнопка «згенерувати нові біткойни» . Для пошуку хешу заголовка нового блоку використовувався центральний процесор комп'ютера. Імовірність успішного створення блоку Майнер приблизно дорівнює відношенню його обчислювальної потужності до обчислювальної потужності всієї мережі, і якщо це ставлення дуже мало, то ймовірність отримання нагороди навіть за тривалий проміжок часу буде незначною. Бажають збільшити ймовірність отримання винагороди прагнуть задіяти якомога більші обчислювальні потужності. Особливість завдання перебору хешів дозволяла застосувати максимальне розпаралелювання

обчислень. Для цього добре підійшли багатопотокові графічні процесори (GPU) після появи невеликої додаткової програми (в сотні разів продуктивніше CPU) і плати з FPGA (продуктивність аналогічна відеокарт, але перевершують їх по енергоефективності). Після цього Майнінг за допомогою центрального процесора виявився недоцільним через занадто малу ймовірність отримати винагороду, і кнопку в програмі-клієнті прибрали. Пізніше були випущені спеціалізовані процесори (ASIC), орієнтовані на обчислення хешів для мережі «біткойнів », більш продуктивні, ніж GPU і FPGA. З 2013 року Майнінг без спеціалізованих процесорів (на відкритих або центральний процесор) став нерентабельним: вартість споживаної електроенергії перевищила середній результат.

З 2013 року з'являються репортажі про «фабриках біткойнів » - спеціалізованих безлюдних підприємствах, на яких «працюють» тисячі ASIC-процесорів. Місячний дохід фабрики може перевищувати мільйон доларів (кілька тисяч біткойнів). На початку 2015 року, навіть якщо припустити, що всі Майнер використовують найбільш енергоефективні процесори ASIC, сумарний витрата електроенергії на Майнінг оцінювався в 1,46 терават-годин на рік, що еквівалентно річному споживанню: 135 000 американських будинків (середній рівень близько 10 , 8 МВт·ч за рік).

Після формування кожних 210 000 блоків (приблизно раз в 4 роки) запрограмовано розмір винагороди новими біткойнів зменшувати вдвічі, тобто це значення є спадною геометричною прогресією (розмір винагороди 50 → 25 → 12,5 → ...). Загальний обсяг емісії біткойнів обмежений, тому що є сумою членів спадної

геометричної прогресії, і не перевищить 21 млн. На травень 2014 року в обігу перебувало 12,7 мільйона біткойнів.

Спочатку розмір емісії при створенні блоку становив 50 біткойнів. 28 листопада 2012 відбулося перше зменшення емісійної нагороди з 50 до 25 біткойнів . 9 липня 2016 року стався другий зменшення емісійної нагороди з 25 до 12,5 біткойнів . У 2031 році розмір емісії при створенні блоку складе менше одного біткойнів і продовжить прагнути до нуля. Передбачається, що емісія зупиниться в 2140 році, оскільки нагорода за блок не зможе перевищувати 10-8 BTC, однак задовго до цього поступово основним джерелом винагороди за формування нових блоків стануть комісійні збори.

2.6 Висновок

У цьому розділі був проведений аналіз крипто валюти Bitcoin. Виявлено основні переваги крипто валюти, які наведено нижче.

Повна відсутність інфляції. Монети біткоіни з'являються з величезною швидкістю, при цьому їх максимальна кількість строго обмежена - їх не може бути більше, ніж 21 млн. А так як на дану кріпвалюту немає ніякого економічного чи політичного впливу, то і інфляції не існує. Саме тому, можна сказати, що біткоіни є більш надійним активом, ніж золото.

Біткоіни мають відкритий вихідний код, проте фактичні дані про користувача залишаються недоступними. Іншими словами, ви зможете бачити все транзакції, але не зможете дізнатися, хто і куди їх справив.

Пірінгова мережу. Абсолютно всі протоколи працюють за тимчасової системі, тому ніякої банк або фінансова система не зможуть втрутитися в хід операції.

Відсутня обмеження по переказах, іншими словами, ви можете відправляти біткоіни в будь-яку країну світу, і ніхто не зможе відстежити вашу транзакцію. Комісія при угоді біткоіни становить не більше 0,1 відсотка.

Біткоіни неможливо підробити, ними можна розраховуватися в будь-якому місті світу, а кількість ресурсів, які приймають їх до оплати, стає дедалі більше.

Якщо говорити про недоліки біткоіни, то це велика волатильність валюти. Іншими словами, навіть за місяць ціна біткоіни може здійснювати величезні стрибки.

3. BITCOIN WALLET

Основною задачею Bitcoin Wallet є створення відкритих ключів для прийому біткоінів і використання відповідних закритих ключів, щоб витрачати біткоіни. Гаманець файли зберігає закриті ключі і (за бажанням) іншу інформацію, пов'язану з операціями. Основний принцип роботи Bitcoin Wallet наведено на Рис 3.1.

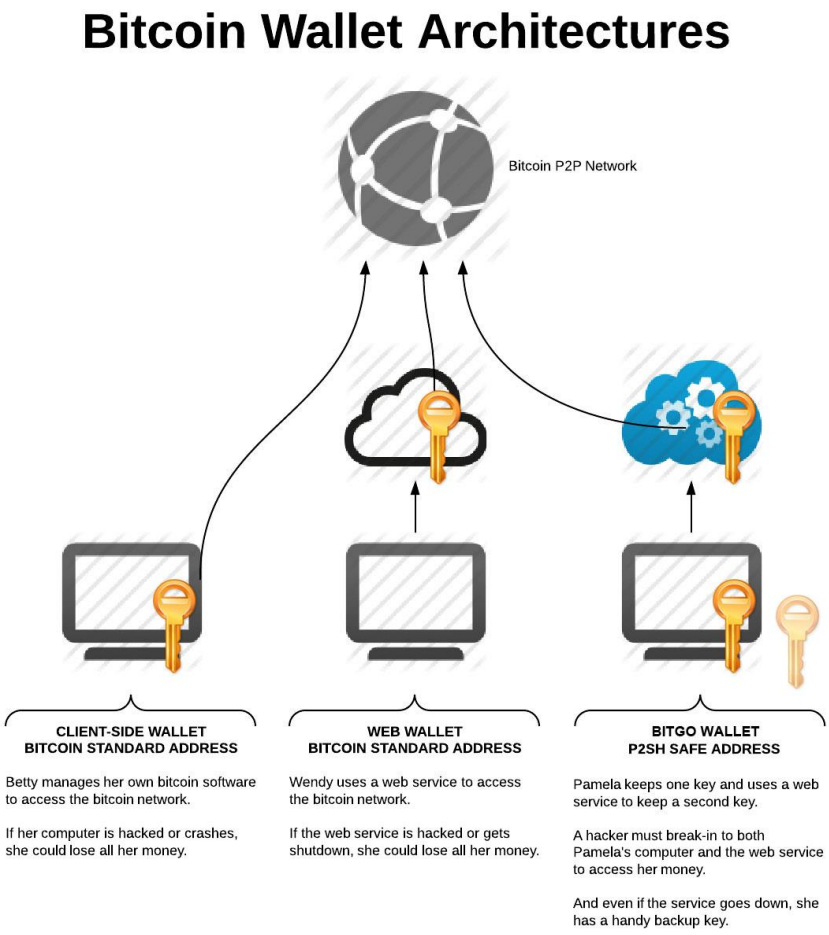


Рис 3.1 – Архітектура Bitcoin Wallet

Дозвіл на витрати та отримання біткоінів є основною ознакою гаманця, але конкретна одна програма може виконувати лише одну з цих ознак. Оскільки різні гаманці можуть працювати в парі. Наприклад, один гаманець може використовуватися для створення відкритих ключів (для отримання біткоінів), а

інший для створення закритих ключів. Гаманець також необхідний для того, щоб взаємодіяти із усією мережею, аби отримувати інформацію про нові транзакції.

Отже, ми маємо три основні частини, які має виконувати Bitcoin Wallet:

- Генерація ключів;
- Можливість підписувати транзакції закритим ключем;
- Постійна взаємодія зі всією мережею.

Найпростіший гаманець це програма, яка виконує всі три функції: вона генерує закриті ключі, отримує відповідні відкриті ключі, допомагає поширювати ці відкриті ключі в міру необхідності, слідкує за біткоінами, які були витрачені за цим закритим ключем.

На момент написання цих рядків майже всі популярні гаманці можна використовувати в якості гаманців з повним набором послуг.

Основною перевагою гаманців з повним набором послуг є те, що вони прості у використанні. Одна програма робить все, що потрібно користувачеві, щоб отримувати і витрачати біткоіни.

Основним недоліком гаманців з повним набором послуг є те, що вони зберігають секретні ключі на пристрої, підключеному до Інтернету. Компроміс таких пристроїв є звичайним явищем, а підключення до Інтернету спрощує передачу секретних ключів від зламаного пристрої зловмисникові.

Для захисту від крадіжки багато програм гаманців пропонують користувачам можливість шифрування файлів гаманця, що містять секретні ключі. Це захищає закриті ключі, коли вони не використовуються, але не може захистити від атаки, призначеної для захоплення ключа шифрування або для читання розшифрованих ключів з пам'яті.

Щоб підвищити безпеку, приватні ключі можуть бути згенеровані і збережені окремою програмою гаманця, діючу у більш безпечному середовищі. Ці гаманці, використовуються тільки для підписання, працюють спільно з мережевим гаманцем, який взаємодіє з усією мережею.

Програми гаманців з підписом зазвичай використовують детерміноване створення ключів (описане в наступному підрозділі) для створення батьківських закритих і відкритих ключів, які можуть створювати дочірні приватні та публічні ключі.

При першому запуску гаманець підписи створює батьківський закритий ключ і переносить відповідний батьківський відкритий ключ в мережевий гаманець.

Мережевий гаманець використовує батьківський відкритий ключ для отримання дочірніх відкритих ключів, необов'язково допомагає їх поширювати, контролює виходи, витрачені на ці відкриті ключі, створює непідписані транзакції, які проводять ці виходи, і переносить непідписані транзакції в гаманець підпису.

Часто користувачам надається можливість переглянути деталі непідписаних транзакцій (зокрема, дані про вихід), використовуючи гаманець підпису.

Після необов'язкового етапу огляду гаманець підпису використовує батьківський закритий ключ для отримання відповідних дочірніх закритих ключів і підписує транзакції, надаючи підписані транзакції назад в мережевий гаманець.

Потім мережевий гаманець транслює підписані транзакції в однорангові з'єднання.

3.1 Опис існуючих рішень

3.1.1 Bitcoin Core

Оригінальний клієнт (Рис 3.2), написаний на базі розробки засновника Bitcoin Сатоси Накамото і підтримуваний групою розробників на чолі з Гевіном Андресеном. Якщо ви не впевнені, який клієнт використовувати, то краще за все почати з нього.

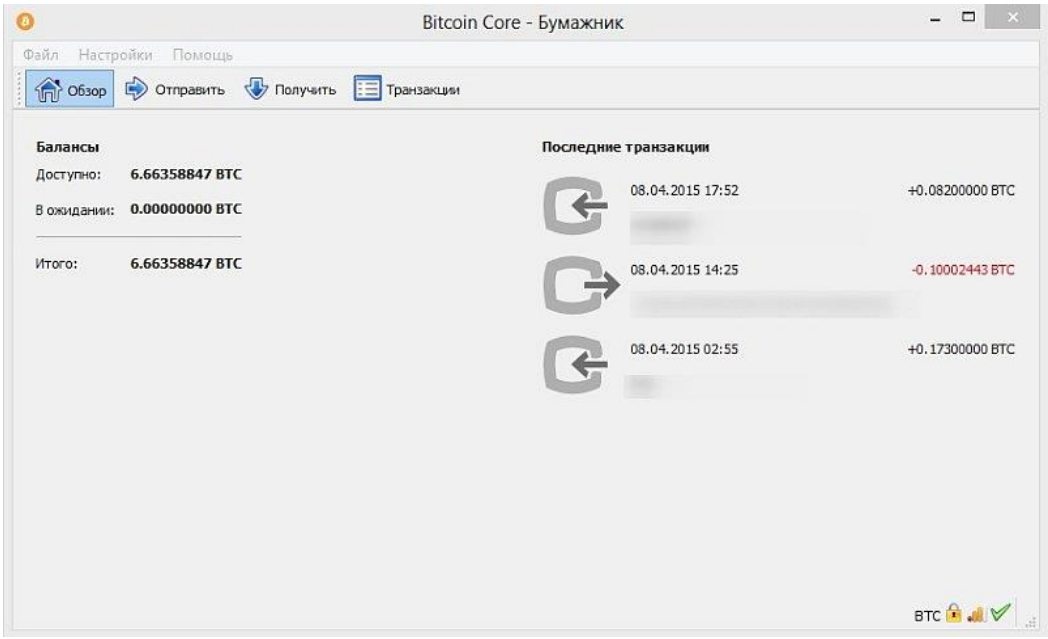


Рис 3.2 – Зовнішній вигляд Bitcoin Core

Додаток має високий рівень безпеки, конфіденційності та стабільності, а його розвиток йде попереду всіх інших клієнтів, так як саме він є офіційним клієнтом Bitcoin. Однак, дане додаток дуже ресурсномістке. Бажано залишати його весь час включеним, підтримуючи тим самим систему Bitcoin, щоб інші вузли могли підключатися до вас. Якщо у вас слабкий комп'ютер, то можливо варто придивитися до інших клієнтів для управління біткоіни.

Особливості гаманця: ваші адреси bitcoin і закриті ключі до них зберігаються в файлі wallet.dat. Клієнт пропонує можливість зашифрувати файл за допомогою пароля. У додатку є російська версія. Можливість імпорту і експорту ключів, підписи повідомлень. А робота через командний рядок

дозволяє скористатися додатковими можливостями клієнта, які не реалізовані в графічній оболонці.

3.1.2 Armory

Працює поверх Bitcoin Core, розширюючи його функціональні можливості. Тому для його роботи буде потрібно встановлений офіційний клієнт Bitcoin Core з синхронізованими блоками. Якщо при запуску Armory не знайде файли баз даних з блоків, то він повідомить про це, вказавши, що програму слід запустити з ключем.

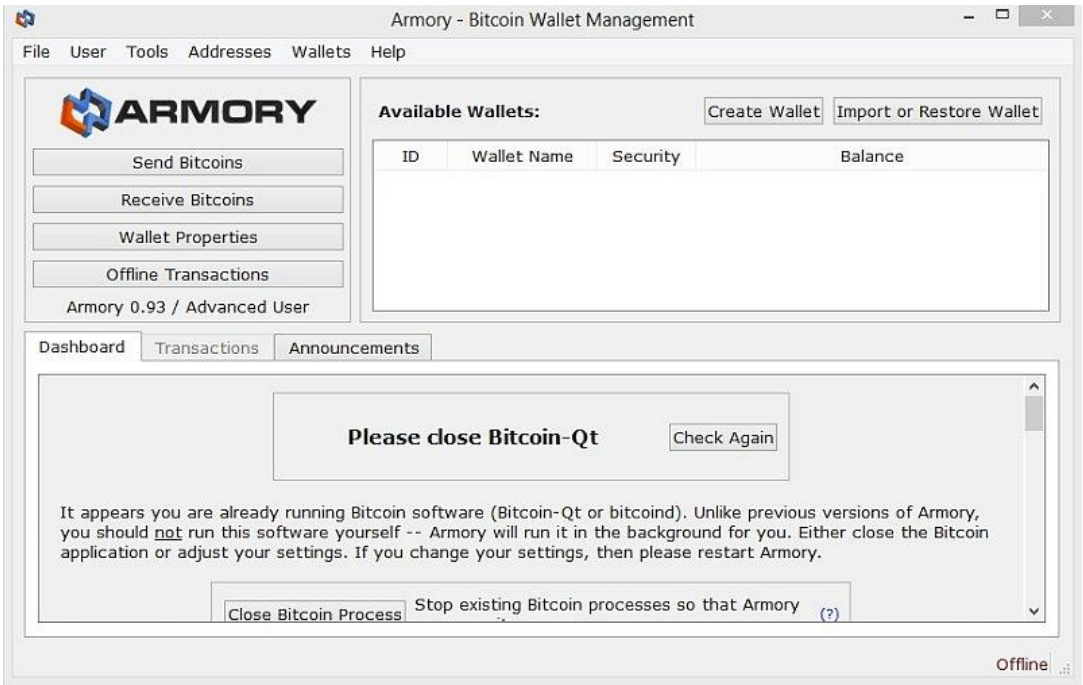


Рис 3.3 – Зовнішній вигляд Armory

Armory (Рис 3.3) простий у використанні навіть для початківців користувачів. Можна керувати кількома гаманцями, покращено безпеку. З його допомогою, для захисту від атак з інтернету, можна зберігати гаманці офлайн. Адреси, згенеровані за допомогою програми VanityGen, легко імпортувати в гаманець.

За допомогою Armory навіть можна створювати повідомлення, які будуть підписані вашим закритим ключем bitcoin адреси, щоб інші могли переконатися в тому, що повідомлення прийшло від вас.

Гаманець Armory підійде для досвідчених користувачів, яким потрібна додаткова гнучкість і безпеку для управління своїми коштами.

Слід врахувати, що Armory буде важко працювати на застарілих комп'ютерах, він вимагає, як мінімум, 2 Гб оперативної пам'яті. І навіть при 2Гб пам'яті, може запускатися хвилин 10-20, а того й більше. Також відсутня російська версія клієнта.

3.1.3 MultiBit

MultiBit - швидке і просте у використанні додаток навіть для людей, які не мають специфічних технічних знань. У нього є канал на YouTube, що допомагає краще познайомитися з даними гаманцем і його функціями. Синхронізація з мережею відбувається досить швидко, зазвичай кілька хвилин. Додаток також переведено на російську мову.

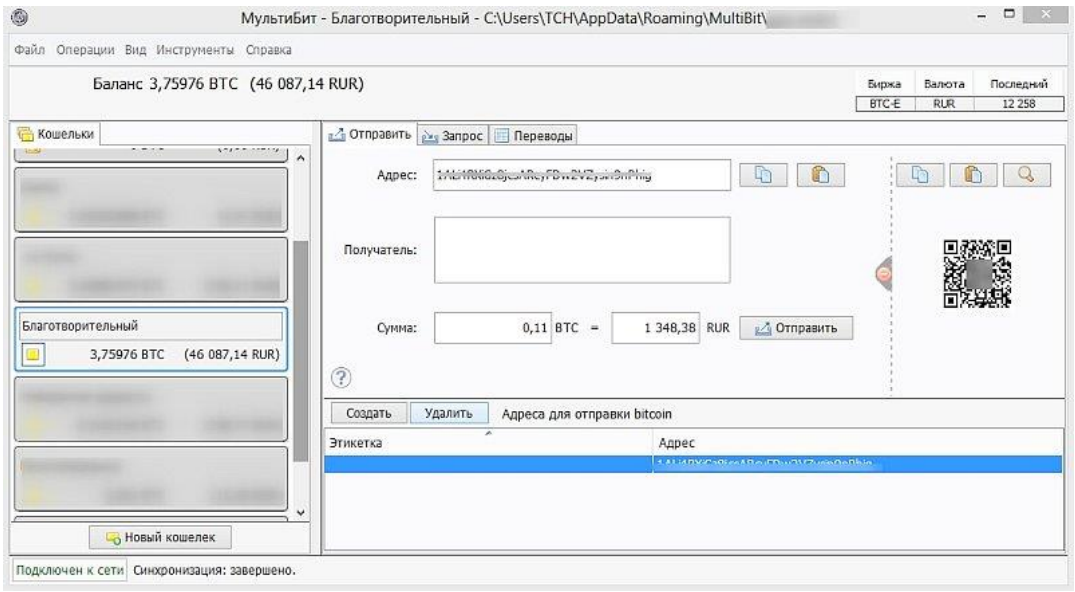


Рис 3.4 – Зовнішній вигляд MultiBit

Розмір комісії задається в настройках програми. Перед відправкою, програма виводить дані з перекладу (адреса, сума, комісія) для підтвердження.

Зараз розробка Multibit (Рис 3.4) кілька загальмувалася, так як зусилля розробників спрямовані на новий продукт - Multibit HD, що знаходиться в стадії початкового тестування.

3.1.4 Electrum

Завдання цієї програми - максимально прискорити роботу з мережею Bitcoin і мінімізувати споживані ресурси. Після установки гаманець пропонує вибрати сервер, до якого буде відбуватися підключення - можна вибрати зі списку або вказати свій. Після цього він згенерує деяку секретну фразу (seed), яке треба або записати, або роздрукувати у вигляді QR коду. При цьому немає необхідності робити резервні копії гаманця.

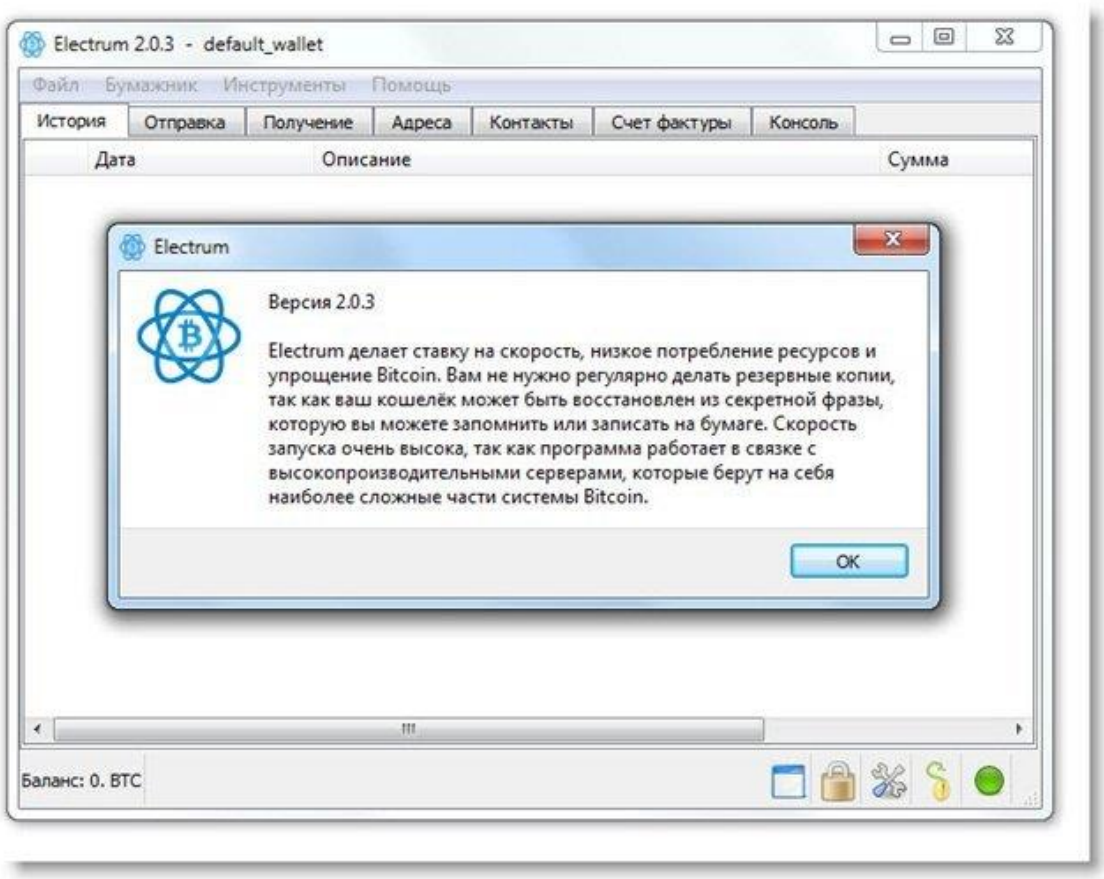


Рис 3.5 – Зовнішній вигляд Electrum

Остання версія Electrum 2.0.3 (Рис 3.5) пропонує новий метод деривації ключів і адрес (BIP32) і підтримку гаманців, які використовують адреси BIP32 і P2SH. Є можливість можливість створення мультіподпісі, а також обробляються платіжні запити BIP70. Взаємодіє з апаратними гаманцями Trezor і Btchip HW1.

3.2 Власна реалізація

На базі технології Bitcoin Core було вирішено розробити власний варіант біткоін гаманця, який виглядає як веб-сайт, а в якості back-end використовує потенціал Bitcoin Core (через API).

Нижче наведено код програми для інтеграції з JSON-RPC протоколом:

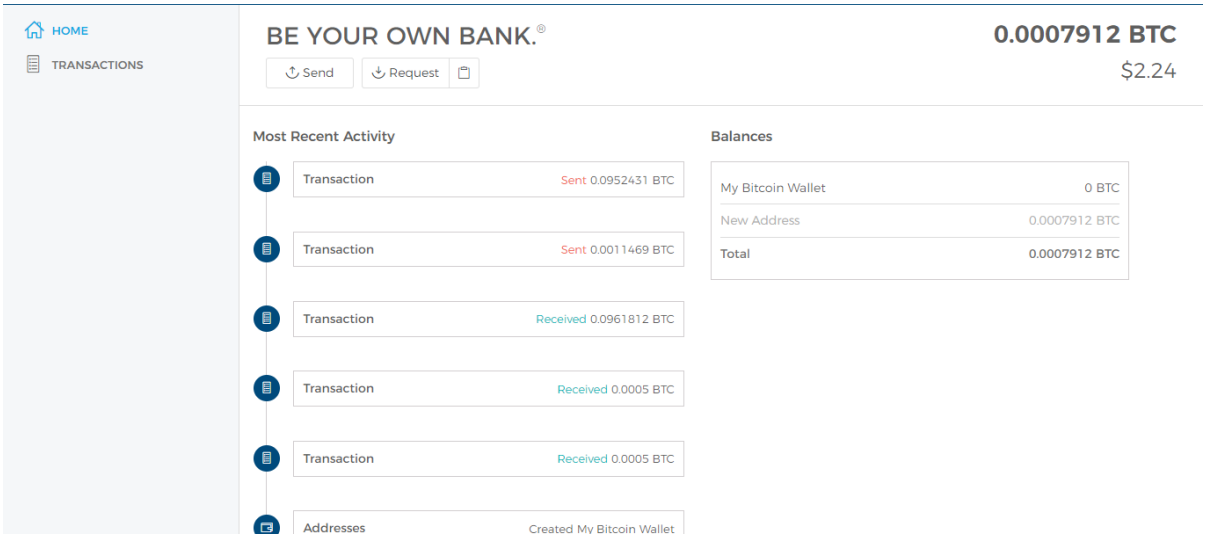
```
#include <QCoreApplication>
#include <QAuthenticator>
#include <QStringList>
#include <QDebug>

#include "qjsonrpchttpclient.h"

class HttpClient : public QJsonRpcHttpClient
{
    Q_OBJECT
public:
    HttpClient(const QString &endpoint, QObject *parent = 0)
        : QJsonRpcHttpClient(endpoint, parent)
    {
        // defaults added for my local test server
        m_username = "bitcoinrpc";
        m_password = "232fb3276bbb7437d265298ea48bdc46";
    }

    void setUsername(const QString &username) {
        m_username = username;
    }

    void setPassword(const QString &password) {
        m_password = password;
    }

private Q_SLOTS:
    virtual void handleAuthenticationRequired(QNetworkReply
*reply, QAuthenticator * authenticator)
    {
        Q_UNUSED(reply)
        authenticator->setUser(m_username);
        authenticator->setPassword(m_password);
    }

private:
    QString m_username;
    QString m_password;

};
```

```
int main(int argc, char **argv)
{
    QCoreApplication app(argc, argv);
    if (app.arguments().size() < 2) {
        qDebug() << "usage: " << argv[0] << "[-u username] [-p
password] <command> <arguments>";
        return -1;
    }

    HttpClient client("http://127.0.0.1:8332");
    if (app.arguments().contains("-u")) {
        int idx = app.arguments().indexOf("-u");
        app.arguments().removeAt(idx);
        client.setUsername(app.arguments().takeAt(idx));
    }

    if (app.arguments().contains("-p")) {
        int idx = app.arguments().indexOf("-p");
        app.arguments().removeAt(idx);
        client.setPassword(app.arguments().takeAt(idx));
    }

    QJsonRpcMessage message =
QJsonRpcMessage::createRequest(app.arguments().at(1));
    QJsonRpcMessage response =
client.sendMessageBlocking(message);
    if (response.type() == QJsonRpcMessage::Error) {
        qDebug() << response.errorData();
        return -1;
    }

    qDebug() << response.toJson();
}
```

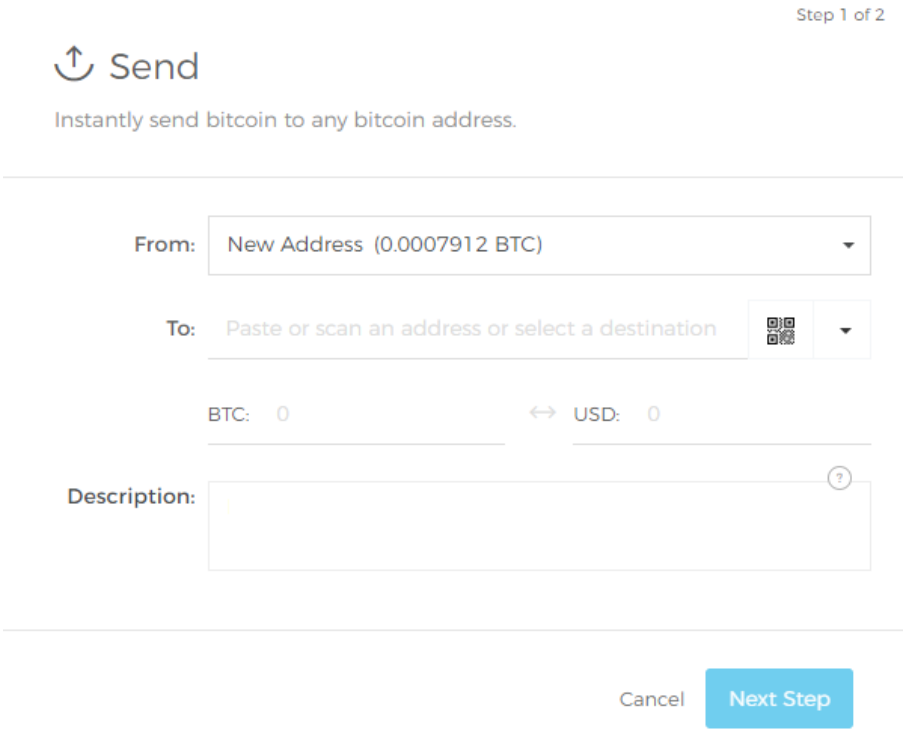
Приклад відповіді, що надходить за допомогою API:

```
{
  "to" : ["1A8JiWcwvpY7tAopUkSnGuEYHmzGYfZPiq", "18fyqiZzndTxdVo7g9ouRogB4uFj86JJiy"],
  "from": ["17p49XUC2fw4Fn53WjZqYAm4APKqhNPEkY"],
  "amounts": [16000, 5400030],
  "fee": 2000,
  "txid": "f322d01ad784e5deeb25464a5781c3b20971c1863679ca506e702e3e33c18e9c",
  "success": true
}
```

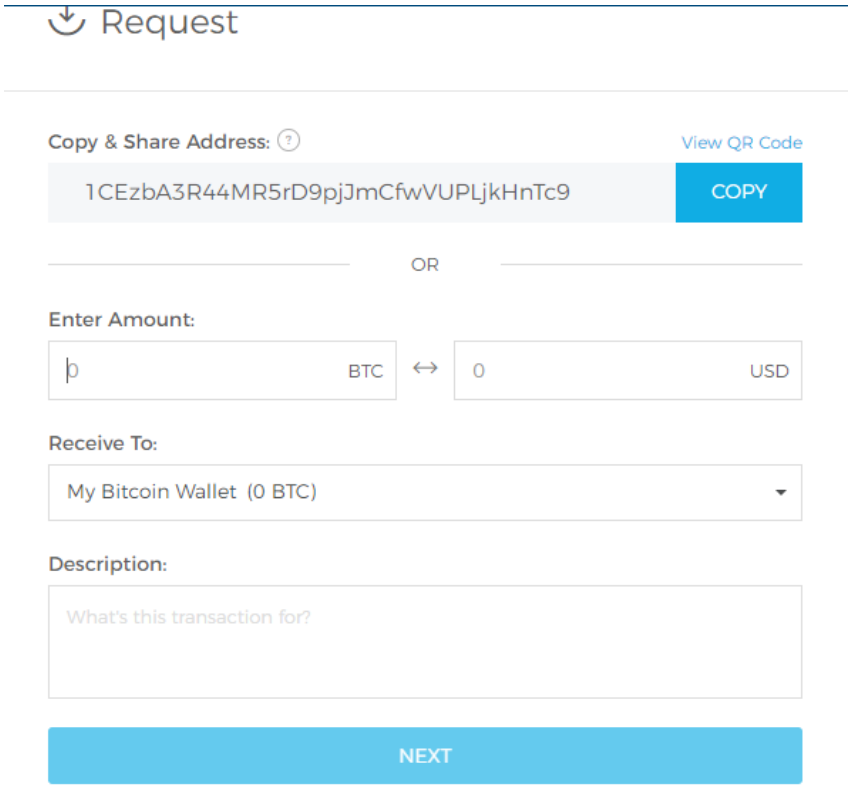
Таким чином виглядає стартове вікно користувача:



Вікно для відправки біткоінів:



Вікно для запиту біткоінів:



3.3 Висновок

В цьому розділі було розглянуто основні принципи побудови біткоін гаманців. Також було детально розглянуто варіанти реалізації, які існують на сьогоднішній день. В ході досліджень було виявлено, що Bitcoin гаманці діляться на 4 основні види в залежності від їх місця розташування (найчастіше це фізичний носій, а гаманець - програмне забезпечення). Слідуючи такій структурі, гаманці діляться на локальні (або ПК), мобільні, онлайнові і апаратні.

Локальние або ПК-гаманці зберігаються на вашому персональному комп'ютері. Файл з ключами в такому випадку зберігається на вашому комп'ютері.

На відміну від локальних, мобільні гаманці («часткові») не вимагають скачки всіх блоків, а подкачивают їх зі сторонніх серверів. Мобільні гаманці мають, як позитивні, так і негативні сторони в експлуатації. До переваг можна

віднести простоту і можливість користуватися ними на портативних гаджетах (смартфонах, планшетах).

Розглянувши усі найпопулярніші варіанти, було вирішено взяти за основу реалізацію Bitcoin Core, оскільки це open source платформа із відкритим API, до якої дуже багато документації.

4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

4.1 Вступ

У даному розділі проводиться оцінка основних характеристик програмного продукту, призначеного для аналізу нелінійних нестаціонарних процесів. Інтерфейс користувача був розроблений за допомогою мови програмування C++ у середовищі розробки Microsoft Visual Studio 2010. Інтерфейс користувача створений за допомогою технології Bootstrap.

Програмний продукт призначено для використання на персональних комп’ютерах під управлінням операційної системи Windows.

Нижче наведено аналіз різних варіантів реалізації модулю з метою вибору оптимальної, з огляду при цьому як на економічні фактори, так і на характеристики продукту, що впливають на продуктивність роботи і на його сумісність з апаратним забезпеченням. Для цього було використано апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) – це технологія, яка дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. Як прямі, так і побічні витрати розподіляються по продуктам та послугам у залежності від потрібних на кожному етапі виробництва обсягів ресурсів. Виконані на цих етапах дії у контексті метода ФВА називаються функціями.

Мета ФВА полягає у забезпеченні правильного розподілу ресурсів, виділених на виробництво продукції або надання послуг, на прямі та непрямі витрати. У даному випадку – аналізу функцій програмного продукту й виявлення усіх витрат на реалізацію цих функцій.

Фактично цей метод працює за таким алгоритмом:

– визначається послідовність функцій, необхідних для виробництва продукту. Спочатку – всі можливі, потім вони розподіляються по двом групам: ті, що впливають на вартість продукту і ті, що не впливають. На цьому ж етапі оптимізується сама послідовність скороченням кроків, що не впливають на цінність і відповідно витрат.

– для кожної функції визначаються повні річні витрати й кількість робочих часів.

– для кожної функції наоснові оцінок попереднього пункту визначається кількісна характеристика джерел витрат.

– після того, як для кожної функції будуть визначені їх джерела витрат, проводиться кінцевий розрахунок витрат на виробництво продукту.

4.2 Постановка задачі техніко-економічного аналізу

У роботі застосовується метод ФВА для проведення техніко-економічний аналізу розробки системи аналізу нелінійних нестаціонарних процесів. Оскільки основні проектні рішення стосуються всієї системи, кожна окрема підсистема має їм задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для збору, обробки та проведення аналізу гетероскедастичних процесів в економіці та фінансах.

Відповідно цьому варто обирати і систему показників якості програмного продукту.

Технічні вимоги до продукту наступні:

– програмний продукт повинен функціонувати на персональних комп’ютерах із стандартним набором компонент;

– забезпечувати високу швидкість обробки великих об’ємів даних у реальному часі;

– забезпечувати зручність і простоту взаємодії з користувачем або з розробником програмного забезпечення у випадку використовування його як модуля;

– передбачати мінімальні витрати на впровадження програмного продукту.

4.2.1 Обґрунтування функцій програмного продукту

Головна функція F_0– розробка програмного продукту, який аналізує процес за вхідними даними та будує його модель для подальшого прогнозування. Виходячи з конкретної мети, можна виділити наступні основні функції ПП:

F_1 – вибір мови програмування;

F_2 – розпізнавання вхідних даних;

F_3 – інтерфейс користувача.

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція F_1:

а) мова програмування C#;

б) мова програмування HTML;

Функція F_2:

а) введення даних вручну (з файлу з певним форматом даних);

б) написання нового модулю розпізнавання даних.

Функція F_3:

а) інтерфейс користувача, створений за технологією Windows Forms;

б) інтерфейс користувача, створений за технологією .NET Framework.

4.2.2 Варіанти реалізації основних функцій

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 4.1). На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (таблиця 4.1).

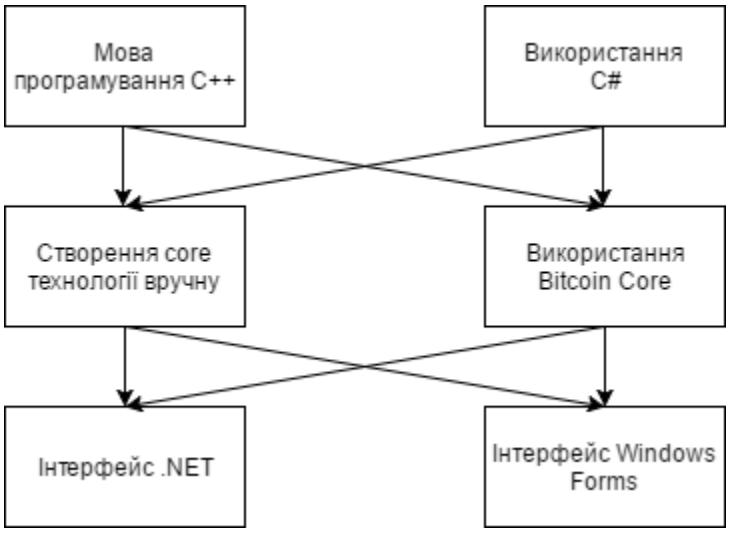


Рисунок 4.1 – Морфологічна карта

Таблиця 4.1 – Позитивно-негативна матриця

Основні функції	Варіанти реалізації	Переваги	Недоліки
F1	*А*	Займає менше часу при написанні коду	Використовує більше ресурсів системи
	Б	Код швидко виконується	Займає більше часу при написанні коду
F2	А	Можливість реалізувати свій протокол	Затрачений час
	Б	Найоптимальніший для використання, оскільки open source	Важко вдосконалити технологію
F3	*А*	Стабільний у використанні	Необхідна додаткова Інсталяція
	Б	Легкий у створенні	Відсутність кросплатформеності

Морфологічна карта відображує всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП.

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція *F1:*

Оскільки розрахунки проводяться з великими об’ємами вхідних даних, то час виконання програмного коду є дуже необхідним, тому варіант а) має бути відкинутий.

Функція *F2:*

Оскільки як показує практика всі open source системи є дуже надійними, а в нашому випадку це дуже важливо, то варіант а) має бути відкинутий

Функція *F3:*

Інтерфейс користувача не відіграє велику роль у даному програмному продукту, тому вважаємо варіанти а) та б) гідними розгляду.

Таким чином, будемо розглядати такі варіанти реалізації ПП:

1. F1б – F2б – F3а

2. F1б – F2б – F3б

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

4.3 Обґрунтування системи параметрів ПП

4.3.1 Опис параметрів

На підставі даних про основні функції, що повинен реалізувати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

– *X1* – швидкодія мови програмування;

– *X2* – об’єм пам’яті для збереження даних;

– *X3* – час обробки даних;

– *X4* – потенційний об’єм програмного коду.

X1: Відображає швидкодію операцій залежно від обраної мови програмування.

X2: Відображає об’єм пам’яті в оперативній пам’яті персонального комп’ютера, необхідний для збереження та обробки даних під час виконання програми.

X3: Відображає час, який витрачається на дії.

X4: Показує розмір програмного коду який необхідно створити безпосередньо розробнику.

4.3.2 Кількісна оцінка параметрів

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у табл. 4.2.

Таблиця 4.2 – Основні параметри ПП

Назва Параметра	**Умовні позначення**	**Одиниці виміру**	**Значення параметра**		
			гірші	**середні**	**кращі**
Швидкодія мови програмування	X1	Оп/мс	19000	11000	2000
Об’єм пам’яті для збереження даних	X2	Мб	32	16	8
Час обробки даних алгоритмом	X3	мс	800	420	60
Потенційний об’єм програмного коду	X4	кількість строк коду	2000	1500	1000

За даними таблиці 4.2 будуються графічні характеристики параметрів – рис. 4.2 – рис. 4.5.

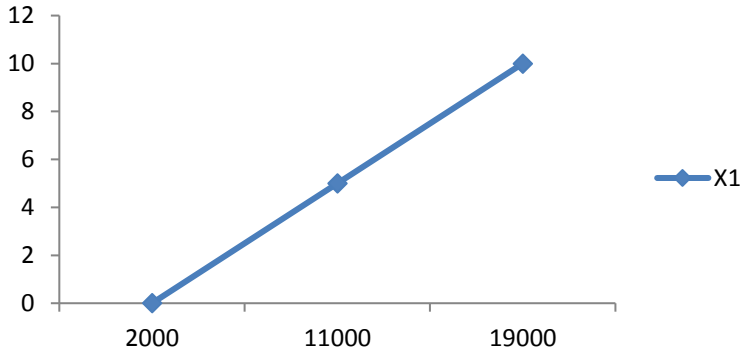


Рисунок 4.2 – X1, швидкодія мови програмування

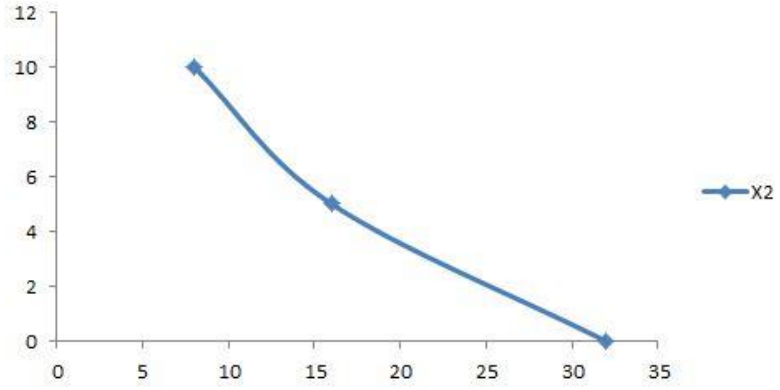


Рисунок 4.3 – X2, об’єм пам’яті для збереження даних

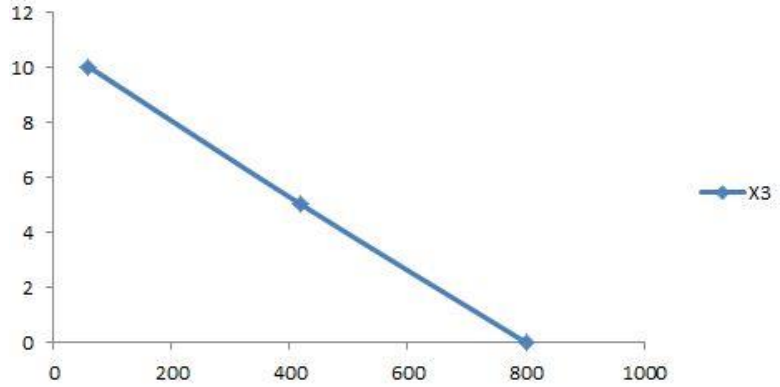


Рисунок 4.4 – X3, час обробки даних алгоритмом

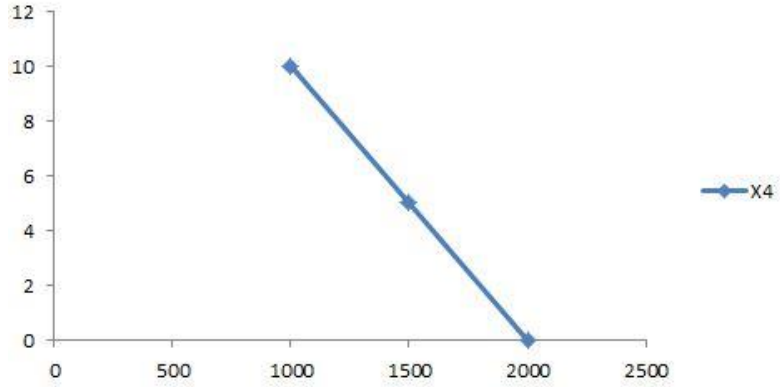


Рисунок 4.5 – X4, потенційний об’єм програмного коду

4.3.3 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

– визначення рівня значимості параметра шляхом присвоєння різних рангів;

– перевірку придатності експертних оцінок для подальшого використання;

– визначення оцінки попарного пріоритету параметрів;

– обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 – Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмування	Оп/мс	4	3	4	4	4	4	4	27	0,75	0,56
X2	Об’єм пам’яті для збереження даних	Мб	4	4	4	3	4	3	3	25	-1,25	1,56
X3	Час обробки даних алгоритмом	Мс	2	2	1	2	1	2	2	12	-14,25	203,06
X4	Потенційний об’єм програмного коду	кількість строк коду	5	6	6	6	6	6	6	41	14,75	217,56
	Разом		15	15	15	15	15	15	15	105	0	420,75

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів за формулою (4.1):

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 105, \quad (4.1)$$

де N – число експертів од;

n – кількість параметрів, од;

R_{ij} – ранг, од.

б) середня сума рангів за формулою (4.2):

$$T = \frac{1}{n} R_{ij} = 26{,}25, \quad (4.2)$$

де n – кількість параметрів, од;

R_{ij} – ранг, од.

в) відхилення суми рангів кожного параметра від середньої суми рангів за формулою (4.3):

$$\Delta_i = R_i - T, \quad (4.3)$$

де Т – середня сума рангів, од;

$R_i -$ сума рангів кожного параметра, од.

Сума відхилень по всім параметрам повинна дорівнювати 0;

г) загальна сума квадратів відхилення за формулою (4.4):

$$S = \sum_{i=1}^{N} \Delta_i^2 = 420{,}75, \quad (4.4)$$

де Δ_i^2 – відхилення, од;

N – кількість експертів.

Порахуємо коефіцієнт узгодженості за формулою (4.5):

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 420{,}75}{7^2(5^3 - 5)} = 1{,}03 > W_k = 0{,}67, \qquad (4.5)$$

де S – загальна сума квадратів відхилення, од;

W_k – нормативний коефіцієнт узгодженості, од.

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 4.4.

Таблиця 4.4 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	=	>	=	<	=	<	<	<	0,5
X1 і X3	<	<	<	<	<	<	<	<	0,5
X1 і X4	>	>	>	>	>	>	>	>	1,5
X2 і X3	<	<	<	<	<	<	<	<	0,5
X2 і X4	>	>	>	>	>	>	>	>	1,5
X3 і X4	>	>	>	>	>	>	>	>	1,5

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij}визначається по формулі (4.6):

$$a_{ij} = \begin{cases} 1{,}5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases} \qquad (4.6)$$

З отриманих числових оцінок переваги складемо матрицю $A = \| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{вi}$ за наступною формулою (4.7):

$$K_{\text{вi}} = \frac{b_i}{\sum_{i=1}^{n} b_i}, \text{де } b_i = \sum_{i=1}^{N} a_{ij}, \qquad (4.7)$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%).На другому і наступних кроках відносні оцінки розраховуються за наступними формулами (4.8, 4.9):

$$K_{\text{ві}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}, \quad (4.8)$$

де $b_i' = \sum_{i=1}^{N} a_{ij} b_j$, (4.9)

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 4.5 – Розрахунок вагомості параметрів

Параметриx_i	Параметриx_j				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	b_i	$K_{\text{ві}}$	b_i^1	$K_{\text{ві}}^1$	b_i^2	$K_{\text{ві}}^2$
X1	1,0	0,5	0,5	1,5	3,5	0,219	22,25	0,216	100	0,215
X2	1,5	1,0	0,5	1,5	4,5	0,281	27,25	0,282	124,25	0,283
X3	1,5	1,5	1,0	1,5	5,5	0,344	34,25	0,347	156	0,348
X4	0,5	0,5	0,5	1,0	2,5	0,156	14,25	0,155	64,75	0,154
Всього:					16	1	98	1	445	1

4.4 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів *X2*(об’єм пам’яті для збереження даних) та *X1* (швидкодія мови програмування) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра *X3 (*час обробки даних) обрано не найгіршим (не максимальним), тобто це значення відповідає або варіанту а) 800 мс або варіанту б) 80мс.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 4.6) за формулою (4.10):

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j}\ , \qquad (4.10)$$

де n – кількість параметрів;

$K_{вi}$– коефіцієнт вагомості i–го параметра;

B_i – оцінка i–го параметра в балах.

Таблиця 4.6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1(X1)	А	11000	3,6	0,215	0,774
F2(X2)	А	16	3,4	0,283	0,962
F3(X3,X4)	А	800	2,4	0,348	0,835
	Б	80	1	0,154	0,154

За даними з таблиці 4.6 за формулою (4.11)

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}], \qquad (4.11)$$

де $K_{\text{ТУ}}[F_{ik}]$ – коефіцієнт рівня якості і функції визначаємо рівень якості кожного з варіантів:

K_{K1} = 0,774 + 0,962 + 0,835 = 2,57

K_{K2} = 0,774 + 0,962 + 0,154 = 1,89

Як видно з розрахунків, кращим є перший варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.5 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка програмної оболонки;

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється за формулою

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТ.М}, \quad (4.12)$$

де T_P – трудомісткість розробки ПП, од;

$K_П$ – поправочний коефіцієнт, од;

$K_{СК}$ – коефіцієнт на складність вхідної інформації, од;

K_M – коефіцієнт рівня мови програмування, од;

$K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм, од;

$К_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення, од.

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $Т_Р$ =90 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $К_П$ = 1.7. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $К_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ}$ = 0.8. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$Т_1 = 90 \cdot 1.7 \cdot 0.8 = 122.4$ людино-днів.

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм другої групи складності, степінь новизни Б), тобто $Т_Р$ =27 людино-днів, $К_П$ =0.9,$К_{СК}$ = 1,$К_{СТ}$ =0.8:

$Т_2 = 27 \cdot 0.9 \cdot 0.8 = 19.44$ людино-днів.

Для третього завдання (використовується алгоритм другої групи складності, степінь новизни Г з використанням перемінної інформації):

$Т_Р$ =14 людино-днів, $К_П$ =0.72, $К_{СТ}$ =0.8:

$Т_о = 14 \cdot 0.72 \cdot 0.8 = 7.05$ людино-днів.

Для четвертого завдання (використовується алгоритм третьої групи складності, степінь новизни Г):

$Т_Р$ =9 людино-днів, $К_П$ =0.6, $К_{СТ}$ =1:

$Т_о = 9 \cdot 0.6 \cdot 1 = 5.4$ людино-днів.

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$T_I = (122.4 + 19.44 + 7.05) \cdot 8 = 1195$ людино-годин;

$T_{II} = (122.4 + 19.44 + 5.4) \cdot 8 = 1181.22$ людино-годин;

Найбільш високу трудомісткість має варіант І.

В розробці беруть участь два програмісти з окладом 16000 грн., один фінансовий аналітик з окладом 19000грн. Визначимо зарплату за годину за формулою (4.13):

$$С_ч = \frac{М}{T_m \cdot t} \text{ грн.}, \qquad (4.13)$$

де М – місячний оклад працівників, грн;

T_m – кількість робочих днів тиждень, од;

t – кількість робочих годин в день, год.

$$С_ч = \frac{16000 + 16000 + 19000}{3 \cdot 21 \cdot 8} = 202{,}38 \text{ грн.}$$

Тоді, розрахуємо заробітну плату за формулою (4.14)

$$С_{ЗП} = С_ч \cdot Т_i \cdot К_Д, \qquad (4.14)$$

де $С_ч$– величина погодинної оплати праці програміста;

$Т_i$ – трудомісткість відповідного завдання;

$К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

І. $С_{ЗП} = 202{,}38 \cdot 1195 \cdot 1.2 = 290214{,}28$ грн.

ІІ. $С_{ЗП} = 202{,}38 \cdot 1181{,}22 \cdot 1.2 = 286866{,}36$ грн.

Відрахування на єдиний соціальний внесок становить 22%:

І. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 290214{,}28 \cdot 0.22 = 63847{,}14$ грн.

ІІ. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 286866{,}36 \cdot 0.22 = 63110{,}6$ грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_М$)

Так як одна ЕОМ обслуговує одного програміста з окладом 16000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 16000 \cdot 0,2 = 38400$ грн.

З урахуванням додаткової заробітної плати:

$С_{ЗП} = С_Г \cdot (1 + К_З) = 38400 \cdot (1 + 0.2) = 46080$ грн.

Відрахування на єдиний соціальний внесок:

$С_{ВІД} = С_{ЗП} \cdot 0.22 = 46080 \cdot 0,22 = 10137,6$ грн.

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 8000 грн за формулою (4.15).

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 8000 = 2300 \text{ грн.}, \quad (4.15)$$

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

$К_А$– річна норма амортизації;

$Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо за формулою (4.16):

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.15 \cdot 8000 \cdot 0.05 = 460 \text{ грн.}, \quad (4.16)$$

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 8 – 16) \cdot 8 \cdot 0.9 = 1706.4$ годин,

де $Д_К$ – календарна кількість днів у році;

$Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів;

$Д_Р$ – кількість днів планових ремонтів устаткування;

t –кількість робочих годин в день;

$К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою (4.17):

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1706,4 \cdot 0,156 \cdot 0,2436 \cdot 1,94 = 125,8 \text{ грн.}, \quad (4.17)$$

де $N_С$ – середньо-споживча потужність приладу;

$К_З$– коефіцієнтом зайнятості приладу;

$Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$С_Н = Ц_{ПР} \cdot 0.67 = 8000 \cdot 0,67 = 5360 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть за формулою (4.18):

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н \quad (4.18)$$

$$С_{ЕКС} = 46080 + 10137,6 + 2300 + 460 + 125,8 + 5360 = 64463,4 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 64463,4 / 1706,4 = 37,78 \text{ грн/час.}$$

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає за формулою (4.19):

$$С_М = С_{М\text{-}Г} \cdot Т \quad (4.19)$$

I. $С_М = 37,78 \cdot 1195 = 45144,02$ грн.;

II. $С_М = 37,78 \cdot 1181,22 = 44626,49$ грн.;

Накладні витрати складають 67% від заробітної плати:

$$С_Н = С_{ЗП} \cdot 0,67$$

I. $С_Н = 290214,28 \cdot 0,67 = 194443,38$ грн.;

II. $С_Н = 286866,36 \cdot 0,67 = 192200,22$ грн.;

Отже, вартість розробки ПП за варіантами становить за формулою (4.20):

$$С_{ПП} = С_{ЗП} + С_{ВІД} + С_М + С_Н \qquad (4.20)$$

I. $С_{ПП} = 290214,28 + 10137,6 + 45144,02 + 194443,38 = 539939,28$ грн.;

II. $С_{ПП} = 286866,36 + 10137,6 + 44626,49 + 192200,22 = 533830,67$ грн.;

4.6 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою (4.21):

$$К_{ТЕРj} = К_{Кj} / С_{Фj}, \qquad (4.21)$$

$К_{ТЕР1} = 2,57 / 539939,28 = 0,47 \cdot 10^{-5}$;

$К_{ТЕР2} = 1,89 / 533830,67 = 0,35 \cdot 10^{-5}$;

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1} = 0,47 \cdot 10^{-5}$.

4.7 Висновок

В даному розділі проведено повний функціонально-вартісний аналіз ПП, який було розроблено в рамках дипломного проекту. Процес аналізу можна умовно розділити на дві частини.

В першій з них проведено дослідження ПП з технічної точки зору: було визначено основні функції ПП та сформовано множину варіантів їх реалізації; на основі обчислених значень параметрів, а також експертних оцінок їх важливості було обчислено коефіцієнт технічного рівня, який і дав змогу визначити оптимальну з технічної точки зору альтернативу реалізації функцій ПП.

Другу частину ФВА присвячено вибору із альтернативних варіантів реалізації найбільш економічно обґрунтованого. Порівняння запропонованих

варіантів реалізації в рамках даної частини виконувалось за коефіцієнтом ефективності, для обчислення якого були обчислені такі допоміжні параметри, як трудомісткість, витрати на заробітну плату, накладні витрати.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР} = 0,47 \cdot 10^{-4}$.

Цей варіант реалізації програмного продукту має такі параметри:

– мова програмування – C#;

– використання технології Bitcoin Core;

– інтерфейс користувача, створений за технологією Windows Forms.

Даний варіант виконання програмного комплексу дає користувачу зручний інтерфейс, непоганий функціонал і швидкодію.

ВИСНОВКИ

В ході роботи було досліджено створення bitcoin wallet на базі технології Blockchain. Складність роботи полягала у тому, що теоретична база хоч і активно розвивається, і є перспективною, проте не надто добре вивчена.

Створення програмних продуктів на базі технології Blockchain є перспективним напрямком сучасних досліджень по децентралізації сховищ даних та створенню смарт-контрактів.

Окремі елементи Blockchain, такі як криптографічні хеш-кодування, розподілені бази даних і побудова консенсусу, самі по собі не нові. Однак їх поєднання створює дуже ефективну нову форму передачі даних і активів, здатну усунути потребу в посередниках, сторонніх центральних органах і дорогих процесах.

Після світової фінансової кризи 2008 року, індустрія ринків капіталу зіткнулася з безпрецедентною лавиною з'їдають доходи проблем, багато в чому зумовлених посиленням регуляторних вимог, зростанням вартості ліквідності і потреби в розподілі капіталу, а також знижуються доходи.

На сьогоднішній день, інвестиційні банки витрачають близько двох третин своїх IT-бюджетів на підтримку старої інфраструктури, щороку вкладаючи додаткові мільярди доларів у проекти зі скорочення витрат.

Іншими словами, банки вкладають занадто багато часу, зусиль, ліквідності і капіталу в підтримку процесів, які не пропонують істотного збільшення прибутковості організації. В результаті банки, центральні банки, біржі та клірингові організації докладають всіх зусиль для якнайшвидшого вивчення можливостей Блокчейн як інструменту впливу на фундаментальні показники витрат, що дозволяє їм повернутися до показників прибутку, достатнім для підвищення рівня прибутковості капіталу.

Слід, однак, внести ясність і підкреслити, що я не вважаю Blockchain панацеєю, здатною вилікувати всі хвороби інвестиційного банкінгу. У багатьох випадках, структури на основі традиційних баз даних або процесів здатні показати схожі результати без необхідності фінансувати розробку блокчейн-рішення і долати пов'язані з нею труднощі. Як приклади можна привести такі області, як внутрішню автоматизацію, скорочення штату, аутсорсинг і офшоринг.

Проте існують наочні свідчення на користь того, що Блокчейн здатний радикально знизити, якщо не повністю усунути, багато існуючих клірингові і взаєморозрахункових процеси.

В майбутньому планую розвивати науково-дослідницьку діяльність в напрямку роботи із технологією Blockchain, оскільки в перспективі можливості створення систем для смарт-контрактів або ICO є необмеженими. На даний момент це ще не до кінця досліджена галузь, поєднавши з іншими технологіями думаю, можна досягти нових, суспільно-корисних результатів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Andreas M. Antonopoulos Mastering Bitcoin: Unlocking Digital Cryptocurrencies / Andreas M. Antonopoulos – K. : NGITS, 2014. – C. 150 – 290
2. Don Tapscott, Alex Tapscott Blockchain Revolution: How the Technology Behind Bitcoin is Changing Money, Business, and the World / Don Tapscott, Alex Tapscott Blockchain – K. : Information Systems, 2016 – C. 100 – 150.
3. Paul Vigna, Michael Casey The Age of Cryptocurrency: How Bitcoin and the Blockchain Are Challenging the Global Economic Order / Paul Vigna, Michael Casey – K. : Economic, 2015 – C. 200 – 210.
4. Chris Skinner Value Web / Chris Skinner – K. Information technologies, 2016 – C. 150 – 175.
5. Roger Wattenhofer The Science of the Blockchain / Roger Wattenhofer – K. : Information technologies, 2016 – C. 94 – 120.
6. Pavan Duggal Blockchain Contracts and Cyberlaw / Pavan Duggal – K. : Information Systems, 2015 – C. 15 – 39.
7. Jacob William Blockchain: The Simple Guide To Everything You Need To Know / Jacob William – K. : Information technologies, 2016 – C. 40 – 50.
8. Tim Harris Bitcoin: Mastering Bitcoin & Cyptocurrency for Beginners — Bitcoin Basics, Bitcoin Stories, Dogecoin, Reinventing Money & Other Digital Currencies / Tim Harris – K. : Economic, 2016 – C. 30 – 47.
9. Eric Sammons Bitcoin Basics: 101 Questions and Answers / Eric Sammons – K. : Information Systems, 2015 – C. 93 – 100.
10. Boyen, X., Carr, C., Haines, T. – Blockchain-Free Cryptocurrencies. A Rational Framework for Truly Decentralised Fast Transactions / Xavier Boyen, Christopher Carr, Thomas Haines K. : Information Systems, 2015 – C. 45 – 90.
11. Churyumov, A. – Byteball: A Decentralized System for Storage and Transfer of Value \ Anton Churyumov – K. : Economic, 2015 – C. 200 – 210.

12. Lerner, S. D. – DagCoin Draft / Sergio Demian Lerner – K. : Information technologies, 2016 – С. 40 – 50.
13. World Wide Web: Біткоін – P2P грощі з відкритим початковим кодом [Електронний ресурс] – Режим доступу: https://bitcoin.org/ - Дата доступу: 15.02.2017.
14. World Wide Web: Біткоін – P2P грощі з відкритим початковим кодом [Електронний ресурс] – Режим доступу: https://bitcoin.org/ - Дата доступу: 15.02.2017.
15. World Wide Web: Blockchain Wallet API: Bitcoin Wallet API – Blockchain [Електронний ресурс] – Режим доступу: https://blockchain.info/api/ - Дата доступу: 30.03.2017.
16. World Wide Web: Bitcoin Wiki [Електронний ресурс] – Режим доступу: http://ru.bitcoinwiki.org - Дата доступу: 30.05.2017.
17. Melanie Swan Blockchain: Blueprint for a New Economy / Melanie Swan - K. : Economic, 2015 – С. 30 – 47.