Demonstracyjna implementacja blockchaina

Autor: Szymon Werk

Wprowadzanie

Blockchain to struktura danych przypominająca listę łączoną. W blockchainie, każdy pojedynczy element (nazywany dalej blokiem) jest połączony z poprzednim blokiem przy pomocy hashu kryptograficznego. Blok zawiera w sobie: hash poprzedniego bloku, własny hash, timestamp, czyli identyfikator, kiedy blok został stworzony oraz dane. Przy tworzeniu nowego bloku tworzony jest nowy hash oparty na powyższych danych, szczególnie używany jest hash poprzedniego bloku. W ten sposób bloki są połączone i odporne na modyfikację danych, bo modyfikacja jednego bloku wymagałaby przeliczenia hashu każdego następnego bloku.
Blockchainy są zazwyczaj obsługiwane na sieci peer-to-peer, jako zdecentralizowana baza danych.
Blockchain został spopularyzowany przez osobę (lub grupę osób) o imieniu Satoshi Nakamoto w 2008 r., który przy użyciu blockchainu stworzył kryptowalutę bitcoin. Tożsamość Satoshiego Nakamoto pozostaje nieznana.

Opis implementacji

W mojej prostej implementacji blockchain to pythonowa lista bloków. Do obliczania hashu używam algorytmu SHA256 (Secure Hash Algorithm 2) z biblioteki hashlib.

```
    def calculate_hash(self):
        """
        funkcja obliczajaca hash bloku na podstawie jego danych
        """
        data_string = str(self.data) + str(self.timestamp) + \
            str(self.previous_hash) + str(self.nonce)
        return sha256(data_string.encode()).hexdigest()
```

SHA256 zwraca hash o długości 256 bitów. Przechowuję ten hash jako string 64 znaków w systemie szesnastkowym. Zagadnieniem znanym z kryptowalut jest „kopanie” bloku. Każdy blok posiada wartość nonce (number used once). Kopanie odnosi się do szukania takiego nonce, aby hash bloku, zaczynał się od zadanej liczby zer (zwane dalej trudnością). Obliczyć hash jest łatwo, ale znalezienie hashu o zadanej trudności może okazać się bardzo trudne. Wykonywane jest to metodą brute-force, a znalezienie odpowiedniego nonce, aby nowy blok został zaakceptowany nazywane jest „proof of work”. To wykorzystywane jest do nagradzania minerów (kopaczy).

```python
    def mine_block(self):
        """
        szuka hashu ktory zgadza sie z zadana trudnoscia blockchainu
        """
        block_hash = self.calculate_hash()

        while not block_hash.startswith('0' * self.difficulty):
            self.nonce += 1
            block_hash = self.calculate_hash()

        return block_hash
```

Uwaga

Przy uruchamianiu programu podajemy trudność, kopanie bloków o trudności max 4 zajmuje zwykle mniej niż sekundę, przy trudności 5 kopanie zajmuje zazwyczaj sekundę lub kilka, przy trudności 6 kopanie zajmuje już nawet kilka minut, przy większych trudnościach trzeba naprawdę długo czekać.

Funkcje

Program uruchamiamy następująco:

python Blockchain.py <difficulty>
gdzie difficulty to trudność blockchaina

Program obsługuję komendę -h

```
usage: Blockchain.py [-h] difficulty

Demonstracja blockchainu.

positional arguments:
  difficulty  Trudnosc blockchainu tzn. ilosc zer wymagana na poczatku kazdego hashu.

optional arguments:
  -h, --help  show this help message and exit
```

Po uruchomieniu wyświetli się informacja o możliwych komendach

```
Blockchain created
Time spent on creating genesis block: 0.19338536262512207 s


Podaj komende:
   add <data> - dodaje blok
   fill <n> - dodaje n blokow o losowych danych (uwaga moze zajac duzo czasu)
   print - wyswietla blockchain
   modify <index> <data> - edytuj dane w bloku
   delete <index> - usun blok
   verify - sprawdz czy blockchain jest poprawny
   clear - wyczysc konsole
   exit - wyjdz z programu
```

Program dla demonstracji wypisuje czas tworzenia każdego bloku.
Blok Genesis to blok zero tworzony przy inicjalizacji blockchaina.

Operacje modyfikacji i usunięcia bloku nie są normalnie dozwolone, ale dodałem je dla demonstracji integralności blockchainu.

Po usunięciu lub modyfikacji bloku, można sprawdzić poprawność blockchainu komendą verify, która wskaże zmodyfikowany blok lub miejsce, w którym brakuje bloku.

```
verify
==============================
Chain is not valid found modified block at index: 2
==============================
Data: zmodyfikowany drugi blok
Timestamp: 2022-02-20 22:02:52
Previous hash: 0000ca5b93c882f1c1ed079387620ef54b9581fe5c839c0d228abd89e4af01ca
Hash: 00002058ba728f4c87df16da6366b751446e667859b230c77d40272028206f9b
Actual recalculated hash:
56831a6c536f64ed5ab45432c71d4ff1195e32b4476ada8e7d7355d2fdf8e673
==============================
```

Po zakończeniu programu zostanie wypisany średni czas kopania bloku, dla celów poglądowych.

Literatura

Artykuł Wikipedii na temat blockchaina

Artykuł Wikipedii na temat Bitcoina

Is it hard to build a Blockchain from scratch?
by Sinai Nday