Bitcoin

Co? Jak? Dlaczego? Po co? Za ile?

Paweł Kraszewski
pawel@kraszewscy.net

Definicja

Bitcoin to:

- kryptowaluta wprowadzona w 2009 roku przez podmiot o pseudonimie *Satoshi Nakamoto*. Do dziś nie wiadomo, czy jest to jedna osoba, grupa osób, czy też jakaś firma.
- używające jej otwarto-źródłowe oprogramowanie
- sieć peer-to-peer, która obsługuje transfery

Bitcoin od strony algorytmów i struktur danych

Hierarchia systemu Bitcoin

- Blockchain to publicznie dostępny łańcuch wzajemnie powiązanych bloków
- Blok zawiera cyfrowo podpisaną listę transakcji
- Transakcja zawiera listę przepływów wartości z wejść na wyjścia
- Wyjście jest definicją, *jaki warunek trzeba spełnić*, aby skorzystać z bitcoina przypisanego temu wyjściu
- Wejście jest cyfrowym *dowodem na prawo dysponowaniem bitcoinem* pochodzącym z wyjścia którejś z wcześniejszych transakcji

Blockchain

- Blockchain jest publiczną bazą, zawierającą listę wszystkich transakcji, jakie kiedykolwiek zostały zaakceptowane w sieci BitCoin.
- Od strony księgowej jest to tzw. *ledger*, czyli globalny rejestr „*winien-ma*”.
- Dystrybucja blockchaina odbywa się za pomocą dedykowanego protokołu P2P.
- Na dziś blockchain ma rozmiar trochę ponad 80 GB
- Integralność blockchaina zapewnia wiązanie skrótów

Kryptografia i kodowanie

- Jako sygnatury z kluczem publicznym używany jest algorytm ECDSA na krzywej secp256k1
  - Jako klucz prywatny może być użyta dowolna liczba 256-bitowa mniejsza od mniej więcej $2^{256}-2^{128}$. *
- Jako funkcji skrótu używa się kaskadowego hashowania:
  - `HASH160` = RIPEMD160 z SHA256
  - `HASH256` = SHA256 z SHA256
- Dane binarne widoczne dla użytkownika kodowane są algorytmem BASE58

* konkretnie - mniejsza od `FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141`

Typy danych 1

```
byte[32] BLOCK_ID_t;
byte[32] TRANS_ID_t;
byte[32] MERKLE_t;

struct {
  varuint size;
  T       data[];
}VARR<T>;
```

Typy danych 2

```
VARR<byte> SCRIPT_t;

struct {
   TRANS_ID_t transaction;
   uint32     index;
} UTXO_t;
```

Blok

```
struct BLOCK_t {
 uint32          size;
 BLOCK_HDR_t     header;
 VARR<TRANS_t> transactions;
};
```

Blok - nagłówek

```
struct BLOCK_HDR_t {
 uint32      version;
 BLOCK_ID_t  prev_block;
 MERKLE_t    merkle_root;
 uint32      time_stamp;
 uint32      difficulty_target;
 uint32      nonce;
};
```

„Kopanie bitcoinów”

- „Kopanie bitcoinów” jest bardzo niefortunnym skrótem myślowym.
- Zadanie w rzeczywistości polega na tym, aby tak manipulować wartościami `nonce`, `time_stamp` i `merkle_root`, aby HASH256 nagłówka miał wartość mniejszą od aktualnie obowiązującej wartości zwanej *trudnością*. (w przybliżeniu - aby hash miał nie mniej niż *ileśtam* zer na początku)
- Trudność dobierana jest raz na 2016 bloków (średnio co 2 tygodnie) tak, aby znalezienie rozwiązania przez całą sieć trwało 10 minut.

„Kopanie bitcoinów”

- Za znalezienie rozwiązania wyznaczona jest nagroda – od 16 lipca 2016 (blok #420 000) jest to 12,5 BTC. Nagroda jest połowiona co 210 000 bloków (~ co 4 lata), czyli następne połowienie będzie na początku lipca 2020.
- Po dojściu do nagrody 1 satoshi, następne połowienie wyzeruje nagrodę. Będzie wtedy wykopane dokładnie 21,000,000 BTC. Szacuje się, że nastąpi to na początku października 2140.
- Do dziś wykopano 75% bitcoinów

Transakcja

```
struct TRANSACTION_t {
 uint32          version;    // obecnie 1
 VARR<INPUT_t>   inputs;
 VARR<OUTPUT_t>  outputs;
 uint32          lock_time;
};
```

Wyjście

```
struct OUTPUT_t {
  uint64    amount;
  SCRIPT_t locking_script;
};
```

- Kwota wyjścia podana jest w *satoshi*, najmniejszej jednostce. Jeden Bitcoin to 100 milionów *satoshi*.
- Skrypt blokujący to zagadka, którą trzeba rozwiązać, aby odblokować bitcoiny.

Wejście

```
struct INPUT_t {
  UTXO_t    UTXO;
  SCRIPT_t unlocking_script;
  uint32    seq; // Nie używ., 0xFFFFFFFF
};
```

- Wejście nie ma kwoty. Zużywa **zawsze** całe wyjście. Jeżeli nie wykorzystujemy całej kwoty, nadmiar odsyłamy na własne konto („reszta”)
- *Skrypt odblokowujący* to rozwiązanie zagadki z wejścia.

Opłaty

- Dla każdej transakcji suma wejść musi być równa lub większa od sumy wyjść (transakcja nie może być debetowa) – z wyjątkiem transakcji tworzącej nowy bitcoin, nie mającej żadnych wejść.
- Różnica sum wejścia i wyjścia to tzw opłata transakcyjna i przypada ona osobie, która „wykopała” blok zawierający daną transakcję.
- Po wykopaniu ostatniego bitcoina, jedynym zyskiem kopalni będą opłaty transakcyjne.
- Kopalnie realizują transakcje zaczynając od najbardziej opłacalnych

Opłaty

(na dzień 17 września 2016)

- Serwis `https://bitcoinfees.21.co` szacuje czas potwierdzeń przy różnych opłatach:
  - >61 satoshi/bajt to realizacja w najbliższym bloku
  - 41...60 satoshi/bajt to realizacja się w ciągu maksymalnie 4 bloków
  - 11-40 satoshi/bajt to czekanie 1-15 bloków
  - 1-10 satoshi/bajt to czekanie 2-21 bloków
  - brak opłat – co najmniej 40 bloków czekania, choć sieć może też całkiem odrzucić transakcję.

Najczęściej płaci się ~60 satoshi/bajt (~130 µBTC = ~30gr za przeciętną transakcję)

Skrypty

- Skrypty Bitcoina wykonywane są przez prostą, stosową maszynę wirtualną.
- Maszyna nie jest zupełna w sensie Turinga. Nie ma żadnego mechanizmu umożliwiającego – nawet pośrednio – realizację pętli.
- Program ma postać ciągu instrukcji.
- Instrukcja składa się 1-bajtowego *opkodu*, po którym występuje 0 lub więcej bajtów argumentu.

Skrypt blokujący i odblokowujący

Rozwiązanie „zagadki” odblokowującej wyjście polega na dopisaniu/wykonaniu takiego skryptu przed skryptem wyjścia, żeby po poprawnym zakończeniu działania całego programu, na stosie pozostała pojedyncza wartość 1 (TRUE).

```
VM.stack.erase()
VM.run(unlocking_script) != E_INVALID
VM.run(locking_script) != E_INVALID
VM.stack.size==1 && VM.stack[0]==1
```

Skrypt *pay-to-pubkey-hash*
(P2PKH)

1)[sig]
2)[pubKey]
3)OP_DUP
4)OP_HASH160
5)[pubKeyHash]
6)OP_EQUALVERIFY
7)OP_CHECKSIG

1-2 – skrypt odblokowania
(część INPUT_t)

3-7 – skrypt blokowania
(część OUTPUT_t)

Skrypt P2PKH

- Do stworzenia transakcji wystarczy hash klucza publicznego odbiorcy. Hash ten przekazywany jest jako „numer konta” bitcoinowego.
- Standardowy adres P2PKH jest konstruowany tak:
  $\mathtt{BASE58}(\ \mathtt{0}_1\ ||\ \mathtt{kHASH}_{20}\ ||\ \mathtt{CHECK}_4\ )$
  gdzie CHECK to pierwsze 4 bajty HASH256(kHASH)
- Adresy P2PKH zaczynają się od cyfry 1
- Do odblokowania BTC trzeba pokazać poprawny podpis hasha wejścia, wyjścia i skryptu kluczem prywatnym, którego klucz publiczny jest zgodny z hashem zaszytym w danym wyjściu.

Skrypt *pay-to-script-hash*
(P2SH)

1)[dane]
2)[skrypt]
3)OP_HASH160
4)[scriptHash]
5)OP_EQUAL
6)OP_EVAL*

1-2 – skrypt odblokowania
(część INPUT_t)

3-5 – skrypt blokowania
(część OUTPUT_t)

Jako niejawny 6-y krok,
[skrypt] jest uruchamiany
tak, jakby został
wstawiony od
punktu 2

* wirtualny opcode

Skrypt P2SH

- Do stworzenia transakcji wystarczy hash skryptu odblokowania. Hash ten przekazywany jest jako „numer konta” bitcoinowego.
- Standardowy adres P2SH jest konstruowany tak: $\texttt{BASE58}(\texttt{5}_1 \texttt{ || } \texttt{sHASH}_{20} \texttt{ || } \texttt{CHECK}_4)$ gdzie CHECK to pierwsze 4 bajty HASH256(sHASH)
- Adresy P2SH zaczynają się od cyfry 3
- Do odblokowania BTC trzeba pokazać skrypt pasujący do hasha i dane wejściowe do tego skryptu, gwarantujące zakończenie go w stanie akceptującym (pojedyncza 1 na stosie)

Skrypt P2SH

Skrypty P2SH używane są głównie do realizacji transakcji multi-signature – do odblokowania wyjścia trzeba N z M podpisów (N≤M).

Przykład skryptu dla multisiga 2-z-3 podpisów:

1) `<sig1>`

2) `<sig2>`

3) `<OP_2 <pk1> <pk2> <pk3> OP_3 OP_CHECKMULTISIG>`

Adresem tej transakcji jest hash skryptu z p. 3

Standardowe typy skryptów

- P2PKH
- P2SH
- P2PK (pay-to-public-key) - jak P2PKH, tylko zamiast hasha klucza publicznego jest cały klucz publiczny). Przestarzały!
- Multisignature (tzw. surowy multisig).
- OP_RETURN metadata (niszczenie bitcoinów, podpisy dokumentów)

Tylko dwa pierwsze typy mają odpowiadające im „adresy bitcoinowe”.

Bitcoin od strony użytkownika

„Zwykły użytkownik”

- Zwykły użytkownik, chcący wymieniać się bitcoinami z innymi użytkownikami, na ogół posługuje się skryptami/adresami P2PKH.
- Do poprawnego funkcjonowania w sieci Bitcoin, użytkownik potrzebuje osobistego klucza prywatnego – który – w kontekście P2PKH jest *myślowym* odpowiednikiem konta w „normalnym” banku.
- To wystarczy, żeby przyjąć, a następnie dalej przesłać bitcoiny.

Pojedyncze konto

banknot albo *czek*

Papierowy „banknot” wygenerowany przez stronę
`https://www.bitaddress.org`
bez żadnych zabezpieczeń

Pojedyncze konto

banknot albo *czek*

Papierowy „banknot” wygenerowany przez stronę
`https://www.bitaddress.org`
zabezpieczony hasłem `Linu.X-Lab`

Pojedyncze konto

moneta

Monety „Denarium” i „Titan”

Pojedyncze konto

portfel pamięciowy

- Nazywany „memory wallet” bądź „brainwallet”
- Klucz prywatny jest generowany z długiego hasła, np. za pomocą funkcji SHA256*
- Dla hasła `Linu.X-Lab-2016` uzyskuje się takie klucze prywatny i publiczny (format WIF):
  - 5J39NvNmdvPWzWxghR7RLahfcHqKMLYPtxwjna5JM18RePXnCMr
  - 1GyUQ3tRUCWxUeoxwdyFjrJ1Y5uD5Do2aF

* Z prawdopodobieństwem 2^{-128} wygeneruje się klucz z „zakazanego” zakresu

Wiele kont („portfel”)

- Aby zapewnić bezpieczeństwo i anonimowość, wskazane jest używanie każdego konta bitcoinowego tylko raz, tworząc nowe konto dla każdej wpłaty.
- Pojawia się problem - jak efektywnie i bezpiecznie zarządzać wieloma kluczami prywatnymi
- Powstały dwa rozwiązania:
  - portfele losowe
  - portfele deterministyczne

Typy portfeli

Klasyczny (losowy)	Deterministyczno-hierarchiczny (HD)
Kolejne adresy prywatne są całkowicie losowe	Kolejne adresy są generowane na podstawie tzw. ziarna („seeda”) w sposób deterministyczny. Dla adresów można generować pod-adresy, pod-pod-adresy – pewnego rodzaju drzewo adresów
Backup ważny do pojawienia się nowych adresów prywatnych (trzeba robić cyklicznie)	Backup obejmuje jedynie ziarno, więc wystarczy zrobić go raz, przy tworzeniu portfela
Adres publiczny jest generowany dla konkretnego adresu prywatnego.	Można maszynowo generować dowolną ilość adresów publicznych na podstawie master-klucza publicznego (xPUB) bez ujawniania żadnego klucza prywatnego.

Typy klientów BitCoin

- Pełny węzeł z kompletną kopią blockchaina - portfele Bitcoin Core i Armory (w sumie nakładka na BTCore)
- Portfel z częściową kopią blockchaina kontaktujący się z pełnym węzłem normalnym protokołem P2P bitcoina („Simple Payment Verification” SPV) – portfele MultiBit, Electrum, Mycelium
- Portfel bez kopii blockchaina, bazujący na usłudze zewnętrznej do weryfikacji sald – portfele webowe na bazie JS

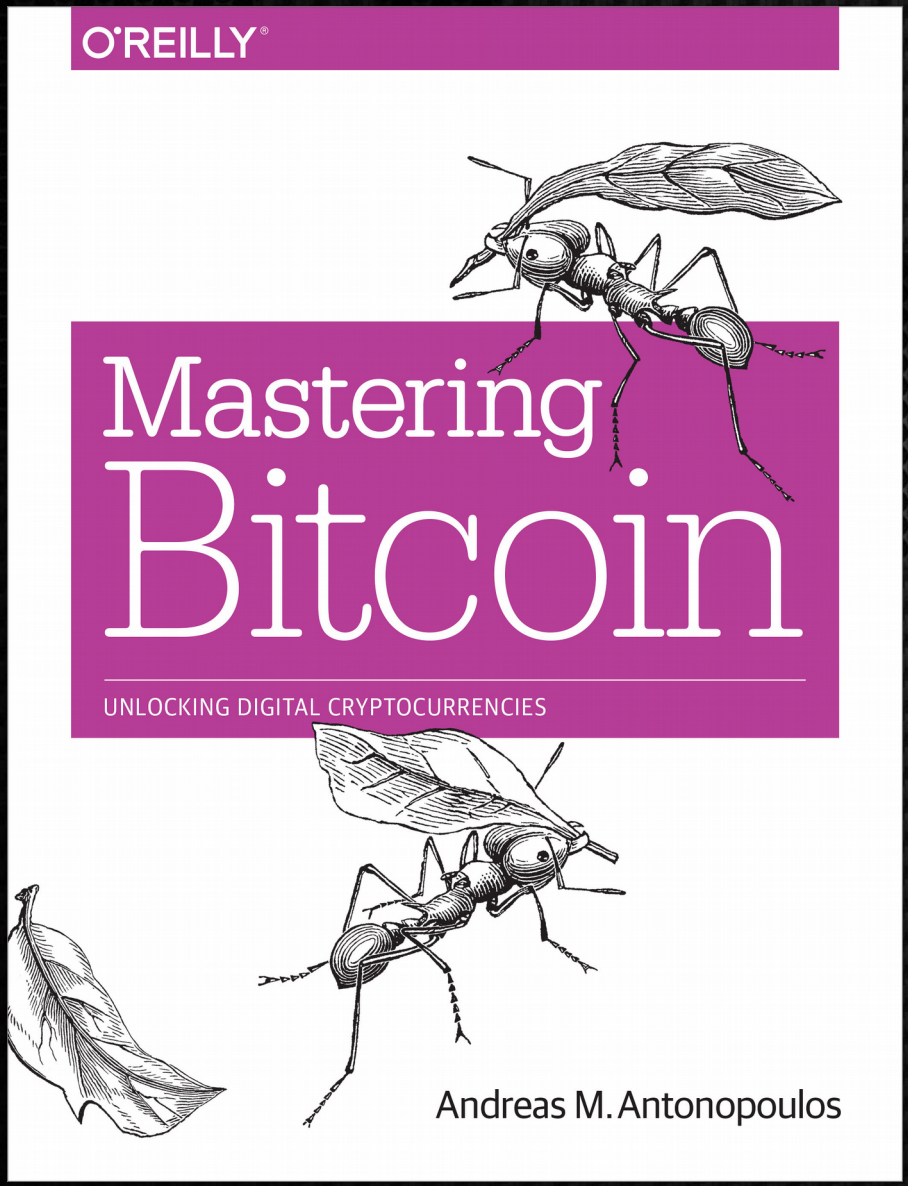
Przechowywanie kluczy prywatnych lub seedów

- Klucz prywatny/seed trzymany jest w osobnym urządzeniu (portfele sprzętowe).
- Zaszyfrowany klucz prywatny/seed trzymany jest w programie portfela (portfele desktopowe i mobilne).
- Zaszyfrowany klucz prywatny/seed trzymany jest na zewnętrznym serwerze, ale jest przysyłany do odszyfrowania i podpisu transakcji lokalnie (współczesne portfele webowe i giełdy)
- Klucz prywatny/seed trzymany jest na serwerze zewnętrznym i tam jest użyty do podpisu transakcji (stare portfele webowe i giełdy)

Część praktyczna

https://blockchain.info
https://blockexplorer.com

Literatura



- Strona domowa z linkiem do darmowej wersji w formacie markdown: www.bitcoinbook.info
- W drodze jest wydanie II, planowane na grudzień 2016
- $28 za ebook *Early Access* na stronach O'Reilly: (http://goo.gl/DYGX92)

Dziękuję za uwagę

17cK9cS2weR2v8waqSUUaxno64SATD9Gt2