Portfele i transakcje Bitcoin

W tej części zagłębisz się w podstawowe RPC bitcoina i dowiesz się o portfelach i transakcjach. Dowiesz się, jak wykorzystać dziedzictwo i portfele bitcoin SegWit. Wydobędziesz klucze publiczne i prywatne portfela. Większość tej Częsci będzie dotyczyć transakcji, od wysyłania środków w prosty sposób z wykorzystaniem testowego łańcucha bloków bitcoin po bardziej złożone transakcje. Dodatkowo dowiesz się, jak wysyłać monety za pośrednictwem podstawowego interfejsu GUI portfela bitcoin, a także dowiesz się, jak przeglądać transakcje w Block Explorer i rozumieć potwierdzenia. Przyjrzysz się surowym transakcjom i dowiesz się, jak utworzyć surową transakcję z jednym wyjściem, a także jak tworzyć transakcje z wieloma użytkownikami, którzy je podpisują. Dodatkowo zastąpisz transakcję i ustawisz czas blokady. Poznasz również różnicę między opcjami płatności a opłatami. Na koniec omówię sposób przekazywania danych w surowej transakcji. Pod koniec tej części będziesz miał znacznie lepsze zrozumienie transakcji, portfeli, opłat, opcji płatności i podstawowego RPC bitcoina.

Zasoby Bitcoin Core RPC

Nauczyłeś się, jak wchodzić w interakcje z rdzeniem bitcoin, wykorzystując demona bitcoin i funkcję rdzenia bitcoin jako serwer HTTP JSON-RPC, a teraz możesz wykonywać połączenia i odbierać odpowiedzi JSON. W tej sekcji opanujesz te umiejętności, aby zrozumieć portfele i transakcje. Pierwszym krokiem jest zainicjowanie i uruchomienie demona bitcoin.

> bitcoind – printtoconsole

Następnie w innym oknie Terminala możesz wyświetlić dostępne RPC polecenia, uruchamiając polecenie pomocy.

> bitcoin-cli help

Możesz również poprosić o pomoc dotyczącą dowolnego uruchamianego polecenia, dodając pomoc przed poleceniem. Na przykład dodaj pomoc przed poleceniem getnewaddress w ten sposób:

> bitcoin-cli help getnewaddress

Portfel Bitcoin

W Części 2 pytałeś o dostępne fundusze portfela za pomocą polecenia getbalance i utworzyłeś nowy portfel bitcoin za pomocą polecenia getnewaddress. W części 3 stworzyłeś swój własny portfel blockchain dla swojego blockchaina; zrobiłeś to, tworząc plik wallet.js wykorzystujący bibliotekę Node.js kryptografii krzywych eliptycznych i generując kombinację kluczy prywatno-publicznych, którą następnie można było ujawnić za pomocą CLI. W tej sekcji poszerzę tę wiedzę, patrząc na rdzeń bitcoina oraz sposób generowania portfeli i transakcji. Bitcoin pozwala użytkownikom wysyłać i odbierać monety. Użytkownik może wygenerować portfel, który zawiera klucz publiczny, a nadawca wyśle monety na adres klucza publicznego portfela odbiorcy. Wysyłanie monet przebiega w ten sam sposób, ale w odwrotnej kolejności. Odbiorca dostarcza nadawcy adres klucza publicznego portfela, na który oczekuje zapłaty, a nadawca wysyła monety na ten adres klucza publicznego. Adres portfela to klucz publiczny, który został wygenerowany przez algorytm mieszający klucza publicznego/prywatnego. Odbiorca może wygenerować nowy klucz publiczny za każdym razem, gdy użytkownik oczekuje płatności. Użytkownicy, którzy nie muszą być anonimowi, mogą używać tylko jednego klucza publicznego do wielu transakcji; jednak oryginalna wizja bitcoina zachęca użytkowników do podawania innego klucza publicznego dla każdej transakcji, a także do ustawiania wielu kluczy prywatnych, które odpowiadają wielu kluczom publicznym. Klucze prywatne są przechowywane w portfelu, a każdy klucz publiczny reprezentuje adres portfela.

Utwórz starszy adres portfela i odzyskaj klucze prywatne

Najpopularniejszy adres bitcoin i typ wygenerowany w rozdziale 2 to adres Pay to PubKey Hash (P2PKH). P2PKH jest kluczem publicznym, a adres klucza publicznego jest haszowany przez algorytm. Bitcoin obsługuje również protokół P2SH-SEGWIT, który omówię w dalszej części tej części.

Uwaga: Segregated Witness (SegWit) był dodatkiem do kodu rdzenia bitcoin za pomocą miękkiego widelca, który zwiększał limit rozmiaru bloku bitcoina poprzez usunięcie danych podpisu, które odblokowują transakcję. Po usunięciu kodu odblokowującego dodatkowa przestrzeń jest wykorzystywana do uwzględnienia większej liczby transakcji w łańcuchu.

Aby wygenerować adres z obsługą P2SH-SEGWIT i P2PKH, wystarczy uruchomić następujące czynności:

> bitcoin-cli getnewaddress

2N96AMUEX4VMNTApPAbUaA6wzP4V9QrbveK

Aby wygenerować adres P2PKH, użyjesz flagi starszej wersji.

> bitcoin-cli getnewaddress "" legacy

13oWKiVQ7C5Ewwjv6KRpP3Xm5YstzqFixT

Jak widać, polecenia zwracają klucze publiczne. Klucze prywatne portfela można wyświetlić, zrzucając klucze do pliku, tak jak poprzednio, lub po prostu za pomocą polecenia dumpprivkey.

> bitcoin-cli dumpprivkey "13oWKiVQ7C5Ewwjv6KRpP3Xm5YstzqFixT"

L5gDpFvfEkUSFeMSQb92kueD1BuX4JeZLAhQkXoEtjcZMog3uXB4

Klucze prywatne nie powinny być nikomu udostępniane, ponieważ odblokowują środki związane z adresem publicznym. Powiedziawszy to, dzielę się z wami tym przykładem jako uczącym się przykładem.

Uwaga: Chroń swoje klucze prywatne. Jeśli zgubisz klucze prywatne, stracisz swoje monety/środki.

Jak wiesz, możesz zrzucić klucze prywatne do pliku tekstowego.

> bitcoin-cli dumpwallet ~/mywallet.txt

{

"filename": "/Users/Eli/mywallet.txt"

}

Następnie możesz uzyskać lokalizację portfela i wyświetlić swoje klucze.

> vim /Users/[location]/mywallet.txt

Zapisany plik danych zawiera nie tylko klucze publiczne i prywatne ale także transakcje związane z Twoim portfelem. Inną użyteczną funkcją RPC, jak możesz sobie przypomnieć, jest możliwość zapytania demona bitcoin o środki z określonego portfela.

> bitcoin-cli getbalance 1Mr2G632PfQuq4uJXRBNWLoRKH71Qwor51

Aby uzyskać dostępne środki w portfelu, po prostu uruchamiasz polecenie getbalance, które zwraca saldo 0, ponieważ nie wpłaciłeś jeszcze żadnych środków.

> bitcoin-cli getbalance

0.00000000

Zapłać świadkowi hash klucza publicznego (P2WPKH): SegWit Soft Fork

Bitcoin (BTC) i bitcoin cash (BCH) zostały mocno rozwidlone głównie przez ponad niezgodność rozmiaru bloku, co oznacza, ile danych można zawrzeć w każdym bloku. W 2017 r. rdzeń kodu bitcoin został mocno wbity w bitcoin cash i umożliwił zwiększenie limitu rozmiaru bloku. W 2019 r. bitcoin cash ponownie się rozwidlił z powodu sporu o kilka nowych funkcji dla każdego widelca. Ograniczenie rozmiaru bloku w bitcoinie oznacza, że czasami transakcje muszą czekać na uwzględnienie w bloku; jednak ze względu na ograniczenie do 1 MB mogą nie zostać uwzględnione w następnym bloku, co powoduje spowolnienie czasu transakcji, gdy w sieci jest zbyt wiele transakcji, co powoduje wzrost opłat za eksploatację. Aby to naprawić, twórcy bitcoin open source stworzyli soft fork i dołączyli Segregated Witness (SegWit). SegWit zwiększył limit rozmiaru bloku bitcoina, usuwając dane podpisu, które odblokowują transakcję. Po usunięciu kodu odblokowującego dodatkowe miejsce można wykorzystać do uwzględnienia większej liczby transakcji w łańcuchu. Ta metoda zwiększa rozmiar bloku do 4 MB.

Uwaga: SegWit to proces, w którym limit rozmiaru bloku w łańcuchu bloków jest zwiększany poprzez usunięcie danych podpisu z transakcji bitcoin. Ten proces zwalnia miejsce i pozwala dodać więcej transakcji. SegWit używa adresu Bech32 zdefiniowanego w BIP173. Ma 90 znaków i składa się z części czytelnej dla człowieka, separatora i danych.

Kod weryfikacyjny odblokowania to dane świadka. Można powiedzieć, że nowy kodeks „segregował świadka”. Stąd wzięła się nazwa. W używanym przez nas buildzie v17.0 znajduje się opcja Witness Public Key Hash w portfelu i transakcji, która zastępuje parametry scriptSig i sprawdza ważność transakcji. Stary, starszy kod nadal działa, ponieważ jest to soft fork. Widziałeś to w poleceniu getaddressinfo, które zawiera zarówno scriptPubKey do obsługi starszych adresów, jak i iswitness. Możesz uruchomić polecenie getaddressinfo i zobaczyć te parametry.

> bitcoin-cli getaddressinfo $address1

Przed bitcoin core v0.16 trzeba było użyć polecenia addwitnessaddress, aby zmienić starszy adres w P2WPKH. Od wersji bitcoin core v0.16.0 adres obsługuje zarówno P2SH, jak i P2WPKH. Zatem portfel to P2SH-P2WPKH. Jeśli używasz wersji 0.18, możesz zauważyć, że adresy getaddressinfo mają oba parametry dla starszego scriptSig i dla SegWit.

Algorytm podpisu cyfrowego krzywej eliptycznej

Rdzeń Bitcoin umożliwia tworzenie podpisu za pomocą algorytmu podpisu cyfrowego krzywej eliptycznej (ECDSA). Można to osiągnąć za pomocą polecenia signmessage. Dodanie podpisu pozwala udowodnić, że posiadasz klucze prywatne portfela, a tym samym dodaje kolejną warstwę bezpieczeństwa dla nadawcy, aby upewnić się, że przesyła środki na właściwy adres.

> bitcoin-cli signmessage "13oWKiVQ7C5Ewwjv6KRpP3Xm5YstzqFixT"

"John Doe"

To polecenie generuje skrót:

HzicuTXMl1COVh7Xw9ky9A/cl7ZjMSWNH10Y/invAgHWa74gS8Eovio3FJkofpH

0nunIA7pJoGwWLRa0UdD7dc8=

Nadawca może zweryfikować portfel przed wysłaniem środków.

> bitcoin-cli verifymessage "13oWKiVQ7C5Ewwjv6KRpP3Xm5YstzqFixT"

"HzicuTXMl1COVh7Xw9ky9A/cl7ZjMSWNH10Y/invAgHWa74gS8EOvio3FJk

ofpH0nunIA7pJoGwWLRa0UdD7dc8=" "John Doe"

Polecenie weryfikacyjne wygeneruje odpowiedź prawda lub fałsz. W takim przypadku odpowie w ten sposób:

TRUE

Dzięki temu użytkownicy mogą potwierdzić, że faktycznie posiadają portfel. Jest to przydatne na przykład na poziomie kodu, ponieważ adres P2PKH będzie wykorzystywał klucz prywatny do generowania podpisu. Adres P2PKH to skrót klucza publicznego odpowiadający kluczowi prywatnemu, który złożył podpis.

Uwaga: ECDSA to algorytm kryptograficzny wykorzystywany przez bitcoin w celu zapewnienia własności środków. Służy do generowania kluczy publicznych/prywatnych, a także może zawierać podpis w algorytmie.

Podpis ECDSA można porównać z maksymalnie czterema możliwymi kluczami publicznymi ECDSA. Te klucze publiczne zostaną zrekonstruowane na podstawie skrótu podpisu; każdy klucz jest haszowany i porównywany z adresem portfela P2PKH podanym dla dopasowania. Wynik jest prawdziwy lub fałszywy. Jak widziałeś wcześniej, przykład otrzymał prawdę po uruchomieniu polecenia Verifymessage.

Uwaga : kod QR to obrazowa reprezentacja ciągu. Czytniki QR są używane do takich rzeczy, jak odczytywanie adresów URL lub kodowanie adresu klucza publicznego portfela.

Możesz wygenerować kod QR za pomocą Chart Google API: https://chart.googleapis.com.

Na przykład, aby wygenerować kod QR dla adresu: 13oWKiVQ7C5Ewwjv6KRpP3Xm5YstzqFixT w kwocie 0,00016 BTC wygenerowałbyś następujący adres URL:

https://chart.googleapis.com/chart?chs=250x250&cht=qr&chl=
bitcoin:13oWKiVQ7C5Ewwjv6KRpP3Xm5YstzqFixT?&amount=0.00016.

Transakcje

W tej sekcji omówię transakcje. Dowiesz się, jak wysyłać monety za pomocą demona bitcoina w sieci testowej, korzystając zarówno z wiersza poleceń, jak i interfejsu graficznego portfela bitcoin core. Dowiesz się, jak używać eksploratora bitcoinów do przeglądania swoich transakcji. Następnie omówię bardziej zaawansowane tworzenie transakcji, pokazując, jak utworzyć surową transakcję z jednym wyjściem, a także bardziej złożone transakcje z wykorzystaniem Multisignature (multisig), który wymaga więcej niż jednego klucza do autoryzacji transakcji. Dodatkowo omówię, jak zmienić inne opcje, takie jak zamiana transakcji na zmianę opłaty, a także ustawienie locktime. Poznasz różnicę między P2PKH a P2SH-SEGWIT. Na koniec dowiesz się, jak dołączać inne dane niż tylko monety za pomocą bitcoina, używając parametrów OP_RETURN. Zacznijmy.

Proste polecenie

Pierwsza transakcja w bloku nazywana jest transakcją coinbase; na tę transakcję składają się opłaty transakcyjne uiszczane przez transakcje zawarte w bloku. Aby wysłać transakcję, musisz uiścić opłatę

transakcyjną górnikom. Jeśli opłata jest niska lub opłata nie została zapłacona, transakcja może utknąć na długi czas lub nawet na zawsze w sieci P2P, dopóki opłata nie zostanie zmieniona. Aby ustawić opłatę transakcyjną, możesz dodać parametr do bitcoina. conf z domyślną opłatą. Najpierw musisz znaleźć lokalizację pliku. Aby to zrobić, zaraz po uruchomieniu demona możesz wyśledzić lokalizację pliku.

> bitcoind – printtoconsole

Po kilku sekundach zatrzymaj tę usługę, naciskając klawisze Control+C. Polecenie pokazuje lokalizację pliku bitcoin.conf. Zwraca lokalizację pliku konfiguracyjnego. Następnie możesz otworzyć plik i zmodyfikować go, dodając domyślną opłatę. W tym przypadku był zagnieżdżony w folderze Application Support.

/Users/[my user]/Library/Application Support/Bitcoin/bitcoin.conf

Po otwarciu pliku widać, że domyślna opłata za transakcję jest ustawiona na 0,00000020 (mintxfee=0,0000020).

Uwaga: w bitcoindzie obowiązują inne opłaty i ustawienia. Możesz modyfikować wysyłane transakcje (paytxfee), maksymalne łączne opłaty (maxtxfee), opłaty awaryjne i tak dalej. Odwiedź tę stronę bitcoin, aby uzyskać wszystkie dostępne opcje: https://en.bitcoin.it/wiki/Running_Bitcoin.

Monitorowanie i aktualizowanie opłaty za transakcję bitcoin może zapewnić, że wysyłane środki zostaną zmienione przez siły rynkowe. Istnieją strony internetowe, aplikacje i formularze, które mogą próbować przewidzieć opłatę, którą należy uiścić. Istnieje wiele witryn, które pomagają obliczać przewidywania opłat transakcyjnych, takich jak ten interfejs API, który można wywołać z kodu:

https://bitcoinfees.earn.com/api/v1/fees/recommended

API zwróciło się w momencie pisania opłaty w wysokości 20 satoshi.

{"fastestFee":20,"halfHourFee":20,"hourFee":18}

Innym przykładem jest https://bitcoinfees.net/. Ta strona pokazuje większość transakcji odbywa się na poziomie pięciu do sześciu satoshi w czasie krótszym niż sześć godzin lub 49 do 50 satoshi w czasie krótszym niż 20 minut.

Uwaga: satoshi to setna część milionowego BTC i nosi imię Satoshi Nakamoto. To najmniejszy ułamek bitcoina, który można wysłać: 0,0000001 BTC. Szybsza opłata wynosiłaby 50 satoshi w momencie pisania.

Teraz, gdy znasz już opłatę, możesz zmodyfikować plik konfiguracyjny z minimalną opłatą na wyższą, np. 50 satoshi.

> vim '/[location]/bitcoin/bitcoin.conf'

mintxfee=0.00000050

txconfirmtarget=3

Wartość mintxfee określa minimalną opłatę transakcyjną w wysokości 50 satoshi, czyli 0,00000050 ฿. To ustawi 20 satoshi/bajt danych w twojej transakcji. Oznacza to, że opłata zmienna musi wyliczyć dobrą kwotę, aby transakcja trafiła do kolejnych trzech bloków. Jak pamiętasz, hashowanie każdego bloku zajmuje około 10 minut, więc uwzględnienie transakcji będzie miało na celu 30 minut. Po zmodyfikowaniu pliku konfiguracyjnego pamiętaj, aby ponownie uruchomić bitcoind.

> bitcoind – printtoconsole

Sieć testowa

W tej sekcji dowiesz się więcej o transakcjach, a aby lepiej zrozumieć transakcje, będziesz musiał wysyłać i odbierać bitcoiny. Aby uzyskać bitcoiny w sieci mainnet (rzeczywisty łańcuch produkcyjny), musisz albo kopać monety, albo nimi handlować. Jednak nie chcesz obsługiwać rzeczywistych monet, gdy się uczysz, ponieważ będziesz musiał płacić opłaty, a także ryzykować utratę monet, jeśli popełnisz błędy. Również cena bitcoina może spaść. Na szczęście bitcoin oferuje alternatywny blockchain, który służy do testowania; to się nazywa testnet. Ten alternatywny łańcuch bloków umożliwia eksperymentowanie bez używania prawdziwych bitcoinów lub nadużywania łańcucha bitcoin. Możesz uruchomić podstawową instancję bitcoin z flagą -testnet. W testnecie odbywa się to za pomocą kranów, pozorowanych monet. Łączysz się z blockchainem testnetu zamiast głównego blockchaina, zatrzymując demona rdzenia bitcoin i uruchamiając go ponownie z flagą testnet.

> bitcoind -testnet

Należy pamiętać, że podobnie jak w przypadku łańcucha mainnet bitcoina, części synchronizujące i indeksujące mogą zająć godziny, w zależności od połączenia internetowego. Uruchom polecenie i zrób sobie długą przerwę na kawę, jeśli chcesz rozpocząć pracę z blokami. Testnet BTC oferuje darmowe bitcoiny kranowe, których możesz użyć do testowania. Testnet prosi o zwrot tych monet po zakończeniu testowania, ponieważ ta usługa jest bezpłatna, a zwrot tych monet przyniesie korzyści następnemu programiście, który ich potrzebuje. Pierwszym krokiem jest wysłanie monet do portfela. Najpierw wygeneruj nowy adres portfela P2PKH za pomocą następującego polecenia:

> bitcoin-cli getnewaddress „” legacy

mnMs77edsGV8VKwtB3d7fsnvrNuZ8ECKfh

Jak widać, wynik, który otrzymujesz, to klucz publiczny, którego możesz użyć do otrzymania środków. Następnie wklej ten adres na https://coinfaucet.eu, wybierz „Bitcoin testnet”, sprawdź, czy nie jesteś robotem i kliknij „Zdobądź bitcoiny!” przycisk. Gdy monety zostaną wysłane do Twojego portfela, otrzymasz potwierdzenie z numerem tx.

Uwaga: należy pamiętać, że te witryny testnetowe kranu często przechodzą w tryb offline i może być konieczne znalezienie nowej witryny testowej kranu.

Przeglądanie transakcji w Block Explorer

W kranie testnetowym możesz monitorować bitcoiny, które zostały wysłane, tak jak można to zrobić w łańcuchu blokowym głównego bitcoina produkcyjnego. Odbywa się to w testnetowym Eksploratorze Blockchain. Jak pamiętasz, „tx ID” oznacza identyfikator transakcji. W rzeczywistości każda transakcja, która kiedykolwiek ma miejsce w łańcuchu bloków, jest publicznie dostępna dla każdego w Eksploratorze Blockchain; która zawiera wszystkie dane transakcji z wyjątkiem kluczy prywatnych użytkowników. Chociaż dane transakcji są publicznie dostępne, informacje identyfikujące właściciela nie są informacjami publicznymi i nie są potrzebne do realizacji transakcji. To, co łączy użytkownika z wysyłanymi monetami, to klucz prywatny powiązany z kluczem publicznym. Podobnie możesz wykonać to samo sprawdzenie informacji za pomocą wiersza poleceń RPC. Wiesz już, jak sprawdzić saldo swojego portfela, jak pokazano tutaj:

> bitcoin-cli getbalance

0,000000

Po otrzymaniu monet nie będzie można ich wydać, dopóki transakcja nie zostanie potwierdzona przez potwierdzenia wydobytych bloków. Dlatego jeśli od razu sprawdzisz saldo, nadal będzie ono wskazywać 0. Będziesz mógł zobaczyć monety jako niepotwierdzone za pomocą polecenia getunconfirmedbalance zaraz po umieszczeniu transakcji w następnym bloku. Aby to sprawdzić, uruchom polecenie getunconfirmedbalance.

> bitcoin-cli getunconfirmedbalance

0.10413028

Po uzyskaniu wystarczającej liczby potwierdzeń polecenie getbalance pokaże swoje nowe saldo, a getunconfirmedbalance pokaże 0. Podobnie możesz być bardziej szczegółowy i poprosić, aby minimalne potwierdzenia wynosiły 2.

> bitcoin-cli getbalance "*" 2

Uwaga: Transakcja pozostaje „niepotwierdzona” do momentu utworzenia następnego nowego bloku. Po utworzeniu nowego bloku nowa transakcja jest weryfikowana i uwzględniana w tym bloku. Teraz transakcja będzie miała jedno potwierdzenie. Mija około dziesięciu minut, tworzony jest nowy blok i transakcja zostaje ponownie potwierdzona. Każde potwierdzenie zwiększa bezpieczeństwo transakcji, a szanse na odwrócenie transakcji maleją. Normą na giełdach jest to, że wymagane jest od czterech do sześciu potwierdzeń, aby umożliwić korzystanie z monet; rozsądnie jest poczekać nawet na sześćdziesiąt potwierdzeń dla dużych ilości monet, co zajmuje około dziesięciu godzin.

Innym przydatnym poleceniem jest polecenie listtransactions; zawiera pełną listę danych transakcyjnych związanych z Twoim portfelem.

> bitcoin-cli listtransactions

[

{

"address": "mnMs77edsGV8VKwtB3d7fsnvrNuZ8ECKfh",

"category": "receive",

"amount": 0.10413028,

"label": "",

"vout": 0,

"confirmations": 420,

"blockhash": "0000000000125d2714882704562c8442a6700c58a41ca

d0b4108305474be3bb1",

"blockindex": 4,

"blocktime": 1541783585,

"txid": "645a34a5cbdd66b126e6f81560dc79957c6e1a175a68f8ad23

ca7fd38046df85",

"walletconflicts": [

],

"time": 1541783585,

"timereceived": 1541890511,

"bip125-replaceable": "no"

}

]

Wysyłanie monet testnetowych za pośrednictwem interfejsu graficznego Bitcoin Core Wallet

Zainicjowałeś podstawową instancję bitcoin z flagą testnet; istnieje jednak jeszcze łatwiejszy sposób wysyłania i odbierania monet. Rdzeń Bitcoin zawiera portfel GUI, którego możesz użyć. Będziesz korzystać z oprogramowania GUI, które wychodzi z pudełka z rdzeniem bitcoin. Aby rozpocząć, zakończ demona bitcoind w Terminalu, naciskając Control + C, a następnie uruchom bitcoin-qt w terminalu wiersza poleceń z flagą testnet, aby połączyć się z testnet, a nie mainnet.

> bitcoin-qt –testnet

To polecenie otwiera nowe okno, a następnie synchronizuje się z blockchainem testnet. Tak jak poprzednio, jeśli nie ukończyłeś synchronizacji testnetowej, może to potrwać kilka godzin, w zależności od połączenia internetowego, jak pokazano na rysunku 4-5. Jednak w interfejsie graficznym portfela zobaczysz szacowany czas, jak długo potrwa synchronizacja. Tak jak poprzednio, musisz poczekać na zakończenie synchronizacji; tylko wtedy możesz odzyskać adres klucza publicznego swojego portfela i wydać swoje monety. W menu Przegląd zobaczysz salda, w tym środki potwierdzone (dostępne) i niepotwierdzone (oczekujące). Możesz również uzyskać listę transakcji, klikając przycisk Transakcje u góry. Aby utworzyć adres klucza publicznego nowego portfela, kliknij Odbierz u góry, a następnie kliknij przycisk Poproś o płatność. Spowoduje to wygenerowanie adresu dla Twojego portfela. Jak widać, GUI stworzył dla Twojej wygody kod QR. Możesz go zeskanować, gdy wysyłasz monety, jeśli ta funkcja jest obsługiwana. Teraz wyślijmy więcej monet do portfela za pomocą kranu testnetowego na https://live.blockcypher.com/btc-testnet/. Jak widać, możesz wtedy otrzymywać monety tak samo, jak przez wiersz poleceń. Następnie wyślesz trochę monet. Będziesz wysyłać 0,01 BTC z powrotem do kranu testnetu, aby inni programiści mogli go użyć. Aby to zrobić, kliknij przycisk Wyślij u góry GUI i wklej adres portfela kranu testnet, który otrzymałeś, gdy wysyłałeś monety do swojego portfela. Zauważ, że obok opłaty transakcyjnej w interfejsie graficznym portfela bitcoin znajduje się przycisk Wybierz. Pozwala to wybrać opłatę, a także liczbę potwierdzeń. Zawiera również sposób na włączenie opłaty „zamień na”. Ta funkcja umożliwia zmianę opłaty w przypadku, gdy opłata jest zbyt niska, a transakcja nie zostaje uwzględniona w bloku. Kran testnetowy wysyła monety na podany przez Ciebie adres portfela. Kiedy wysyłasz i odbierasz monety, otrzymujesz wyskakujące powiadomienie z GUI i zaktualizowane saldo na ekranie przeglądu. Kliknij przycisk Transakcje, aby zobaczyć informacje o transakcji. Możesz również kliknąć każdą transakcję, aby zobaczyć rzeczywiste dane transakcji. Jest to podobne do tego, co widziałeś za pomocą polecenia listtransactions

Surowa transakcja

Do tej pory otrzymałeś jedną transakcję do swojego portfela za pośrednictwem wiersza poleceń, a także monety za pomocą GUI rdzenia bitcoin. Mogłeś również przeglądać potwierdzenia, saldo opłat i transakcje. Jeśli wyślesz środki z powrotem do kranu testnetowego i otrzymasz monety, sprawy są proste. Nazywa się to transakcją jednego wejścia, jednego wyjścia, ponieważ masz jednego nadawcę i jednego odbiorcę i wydałeś tę samą kwotę, którą otrzymałeś (minus opłaty). W prawdziwym życiu

transakcje mogą stać się bardziej złożone, ponieważ istnieje wiele przypadków użycia, w których istnieje jedno wejście i wiele wyjść lub wiele wejść i wiele wyjść. Rdzeń Bitcoin zapewnia zestaw poleceń umożliwiających dostęp do surowej transakcji (RawTransaction), dzięki czemu możesz mieć bardziej szczegółową kontrolę nad transakcją. Zaczniesz od prostej transakcji z jednym wejściem i jednym wyjściem za pośrednictwem wiersza poleceń RPC.

Uwaga : Tworzenie i zrozumienie RawTransaction jest przydatne do tworzenia oprogramowania, ponieważ masz pełną szczegółową kontrolę nad transakcją. Jednak popełnianie błędów może skutkować katastrofą wyniku i utratę monet, więc zachowaj ostrożność i dokładnie sprawdź wszystko przed wysłaniem jakichkolwiek środków.

Po otrzymaniu transakcji transakcja pozostaje w stanie zwanym niewydanymi danymi wyjściowymi transakcji (UTXO) w Twoim portfelu. Aby wysłać jednorazową transakcję jednorazową, Twoja kwota musi być równa środkom, które chcesz wysłać. Następnie możesz wygenerować nowe UTXO dla odbiorcy, do którego wysyłasz monety. Odbiorca może używać tych UTXO do wysyłania transakcji do nowego odbiorcy lub odbiorców, a proces ten może trwać bez końca.

Uwaga: UTXO to indywidualna transakcja na monety przychodzące w Twoim portfelu. Gdy otrzymujesz wiele transakcji na jeden lub wiele adresów portfeli, każda z nich pozostaje jako UTXO, więc będziesz mieć wiele UTXO. Aby utworzyć nową transakcję wychodzącą, zbierasz jeden lub więcej UTXO w zależności od tego, ile chcesz wysłać.

A co, jeśli Twoje UTXO zawiera większą kwotę, niż chciałbyś wydać? Następnie musisz odesłać pozostałe monety z powrotem do portfela. Aby uzyskać listę niewydanych monet, możesz użyć polecenia listunspend. Zamknij portfel bitcoin core GUI za pomocą Control + C i uruchom demona ponownie z flagą testnet.

> bitcoind –testnet

Po uruchomieniu polecenia getbalance otrzymujesz saldo portfela, które obejmuje dwie transakcje otrzymane z https://live.blockcypher.com/btc-testnet/ pomniejszone o transakcję wysłaną z powrotem do kranu testnet.

> bitcoin-cli getbalance

0.18505841

Chciałbym zaznaczyć, że w każdej chwili możesz użyć flagi -named zamiast argumentów kolejności. Nazwany argument jest przydatny, aby upewnić się, że nie popełniasz błędów podczas pracy z mainnet. Na przykład polecenie getbalance z nazwanym argumentem wyglądałoby następująco:

> bitcoin-cli -named getbalance minconf=2

0.18505841

Następnie spójrzmy na polecenie listunspend. Jak sama nazwa wskazuje, zwraca JSON z transakcjami za monety, których nie wydałeś, innymi słowy, swoje UTXO. Polecenie listunspent zwraca również JSON ze zmienną o nazwie vout, która reprezentuje numer indeksu wyjścia w transakcji.

Uwaga: Wartość Vout reprezentuje numer indeksu wyjścia transakcji. Będziesz używać txid i vout, aby wybrać istniejące dane wyjściowe jako dane wejściowe nowej transakcji.

> bitcoin-cli listunspent

[

{

"txid": "50e91c9b73a90bd883f4a9a8a51be729770df20fae0445a

9090b80a8621f4538",

"vout": 0,

"address": "2N67MKgL5rYcbuySDFUdypU5DvKjmwZoYEb",

"label": "",

"redeemScript": "0014c27b4e6bd8eb821ee80a239e0edd59070f

57233d",

"scriptPubKey": "a9148d1c6e108c60cfdfa61565ac328be66245

91404b87",

"amount": 0.09092813,

"confirmations": 17,

"spendable": true,

"solvable": true,

"safe": true

},

{

"txid": "be05d068d1245f1c60ea4229c00eb5e96f2a5c5527f1deb7c6

de5e1e20a4b4db",

"vout": 1,

"address": "2MveVhMe6PTzuhsJHx5zXAjDBwQvzdyqGjM",

"redeemScript": "00142e29123ba343c577ab9517ede9b74f047d2c2ea3",

"scriptPubKey": "a914254f0e95fb26c0f29975f866e69543519bf5

65e787",

"amount": 0.09413028,

"confirmations": 16,

"spendable": true,

"solvable": true,

"safe": true

}

]

Te UTXO pokazują właściwość o nazwie txid, która jest zawarta w blokach bitcoin. Właściwość txid pozwala śledzić transakcje, jak widzieliście za pomocą Eksploratora Blockchain. Zauważ, że indeks zaczyna się od 0, a ponieważ masz dwie transakcje, teraz wynosi 0, a potem 1. Jeśli masz więcej transakcji, ten indeks będzie kontynuowany. Te UTXO pokazują właściwość o nazwie txid, która jest zawarta w blokach bitcoin. Właściwość txid pozwala śledzić transakcje, jak widzieliście za pomocą Eksploratora Blockchain. Zauważ, że indeks zaczyna się od 0, a ponieważ masz dwie transakcje, teraz wynosi 0, a potem 1. Jeśli masz więcej transakcji, ten indeks będzie kontynuowany. Możesz uzyskać wszystkie dane dotyczące transakcji za pomocą polecenia getrawtransaction. Tutaj wybrałem pierwszą właściwość tx z otrzymanego UTXO, a następnie dodałem flagę 1, aby zdekodować dane transakcji zakodowane szesnastkowo; spójrz na polecenie i całe wyjście, pokazane tutaj:

> bitcoin-cli getrawtransaction

50e91c9b73a90bd883f4a9a8a51be729770df20fae0445a9090b80a862

1f4538 1

{

"txid": "50e91c9b73a90bd883f4a9a8a51be729770df20fae0445a9090b

80a8621f4538",

"hash": "e420b350f5b95e29f51b722a5bd44ea2e9d27a7239d2

e17da02f28e04c757b14",

"version": 2,

"size": 248,

"vsize": 166,

"weight": 662,

"locktime": 1443113,

"vin": [

{

"txid": "

2645c128d68194640a7207eeae6ea42e8e528bcba2369

eec0ba572566228b507",

"vout": 0,

"scriptSig": {

"asm": "00143bfa0326c076fa6cab0d23aea170bac38ac9a164",

"hex": "1600143bfa0326c076fa6cab0d23aea170bac38ac9a164"

},

"txinwitness": [

"3045022100fb7f0fc2cf99c8174eb3d14169e1c206157d434d

8290b2efbefa5a37d0773923022065f0b671c0596816c062b9bdc7

b30931edfd99a846a0f1633d301bfb7c03db3c01",

"02d208ff6da0583b99392d30e33c5a12da61b9d9de4c35bb0

d20c33ba3bfc49302"

],

"sequence": 4294967294

}

],

"vout": [

{

"value": 0.09092813,

"n": 0,

"scriptPubKey": {

"asm": "OP_HASH160 8d1c6e108c60cfdfa61565ac328be66

24591404b OP_EQUAL",

"hex": "a9148d1c6e108c60cfdfa61565ac328be6624591404b87",

"reqSigs": 1,

"type": "scripthash",

"addresses": [

"2N67MKgL5rYcbuySDFUdypU5DvKjmwZoYEb"

]

}

},

{

"value": 1453.63689543,
"n": 1,
"scriptPubKey": {
"asm": "OP_HASH160 f4eb3fe1578076853a774d36f193684f86f
71d5f OP_EQUAL",
"hex": "a914f4eb3fe1578076853a774d36f193684f86f71d5f87",
"reqSigs": 1,
"type": "scripthash",
"addresses": [
"2NFaEgWoTNL5akkTuGtYQhzTvWhUaCbxBtL"
]
}
}
],
"hex": "0200000000010107b528625672a50bec9e36a2cb8b528e2ea46
eaeee07720a649481d628c14526000000000171600143bfa0326c
076fa6cab0d23aea170bac38ac9a164feffffff02cdbe8a00000
0000017a9148d1c6e108c60cfdfa61565ac328be6624591404b8
747e059d82100000017a914f4eb3fe1578076853a774d36f1936
84f86f71d5f8702483045022100fb7f0fc2cf99c8174eb3d1416
9e1c206157d434d8290b2efbefa5a37d0773923022065f0b671c
0596816c062b9bdc7b30931edfd99a846a0f1633d301bfb7c03d
b3c012102d208ff6da0583b99392d30e33c5a12da61b9d9de4c3
5bb0d20c33ba3bfc4930229051600",
"blockhash": "00000000000000321b56aece3932b187927ac3e7d
c4532f8811aa612bcfa639a",
"confirmations": 17,
"time": 1542029870,
"blocktime": 1542029870
}

Zauważ, że masz informacje o bloku, potwierdzeniu, wejściu, wyjściu i wiele więcej.

Generowanie surowych transakcji jednym wyjściem Transakcje mogą się łatwo skomplikować, ponieważ często istnieje potrzeba więcej niż jednego wejścia lub więcej niż jednego wyjścia. Na przykład, jeśli chcesz wysłać niewydane monety z powrotem do portfela, a także wysłać monety na wiele adresów, zaczyna się to komplikować. Korzystając z RawTransaction, otrzymujesz pełny dostęp do tego, gdzie trafiają monety i jesteś w stanie realizować złożone transakcje. Zaczniesz od stworzenia prostej RawTransaction, wysyłając jedno UTXO z jednego portfela do drugiego. Wcześniej wysyłałeś monety z powrotem do kranu testnetowego za pośrednictwem interfejsu graficznego portfela bitcoin core. Zróbmy to samo, ale z poleceniem RawTransaction. Na początek potwierdźmy saldo Twojego portfela przed wysłaniem monet.

> bitcoin-cli getbalance 0.18505841

Następnie wybierzmy UTXO, którego będziesz używać do sfinansowania transakcji. Jak pamiętasz, możesz uzyskać listę UTXO za pomocą polecenia listunspent, a następnie spojrzeć na odpowiedź JSON i wybrać txid transakcji. Wybierz transakcję, która ma wystarczającą ilość środków, aby zasilić nową transakcję oraz transakcję, która została potwierdzona.

> utxo_txid="50e91c9b73a90bd883f4a9a8a51be729770df20fae0445a909 0b80a8621f4538"

Jak zapewne pamiętasz, vout to numer indeksu wyjścia w transakcji. W tym przykładzie będę wskazywać na vout i generować nową transakcję. Nowa transakcja może obejmować wiele innych voutów. W tym przykładzie ustawisz pierwszy indeks dla vout.

> utxo_vout="0"

Ostatnią, ale najważniejszą zmienną, którą musisz ustawić, jest adres odbiorcy. Tutaj będziesz używać tego samego adresu portfela, którego używałeś wcześniej do wysyłania monet.

> recipient="mv4rnyY3Su5gjcDNzbMLKBQkBicCtHUtFB"

Na koniec możesz użyć polecenia echo, aby zweryfikować i dwukrotnie sprawdzić, czy poprawnie ustawiłeś zmienne.

> echo $utxo_txid

> echo $utxo_vout

> echo $recipient

Teraz, gdy masz już ustawione zmienne, możesz wygenerować obiekt RawTransaction za pomocą polecenia createrawtransaction. Robisz to, włączając wszystkie ustawione zmienne i deklarując kwotę, którą chcesz wydać. Używasz 0.xxx, ale musisz użyć UTXO pomniejszonego o opłatę, którą chciałbyś zapłacić, aby wysłać wszystkie monety, które masz w UTXO.

> rawtxhex=$(bitcoin-cli createrawtransaction "'[{ "txid":

"'$utxo_txid'", "vout": '$utxo_vout' }]"'

"'{ "'$recipient'": 0.xxx }"')

Next, you can extract the rawtxhex value.

> echo $rawtxhex

020000000138451f62a8800b09a94504ae0ff2

0d7729e71ba5a8a9f483d80ba9739b1ce9500000000000ffff

ffff0140420f00000000001976a9149f9a7abd600c0caa03983

a77c8c3df8e062cb2fa88ac00000000

Wartość rawtxhex zawiera informacje o nowej transakcji w postaci danych zakodowanych szesnastkowo. Następujące polecenie decoderawtransaction zwróci część danych wyjściowych JSON ze zdekodowanymi danymi dla Twojej transakcji:

> bitcoin-cli decoderawtransaction $rawtxhex

{

"txid": "91d4e108f8957251d2997e1f8dcdd0eec97192e8accf85a9e81

f772f586118af",

"hash": "91d4e108f8957251d2997e1f8dcdd0eec97192e8accf85a9e81

f772f586118af",

"version": 2,

"size": 85,

"vsize": 85,

"weight": 340,

"locktime": 0,

"vin": [

{

"txid": "50e91c9b73a90bd883f4a9a8a51be729770df20fae0445a9

090b80a8621f4538",

"vout": 0,

"scriptSig": {

"asm": "",

"hex": ""

},

"sequence": 4294967295

}

],

"vout": [

{

"value": 0.01000000,

"n": 0,

"scriptPubKey": {

"asm": "OP_DUP OP_HASH160 9f9a7abd600c0caa03983a77c

8c3df8e062cb2fa OP_EQUALVERIFY OP_CHECKSIG",

"hex": "76a9149f9a7abd600c0caa03983a77c8c3df8e062cb2fa

88ac",

"reqSigs": 1,

"type": "pubkeyhash",

"addresses": [

"mv4rnyY3Su5gjcDNzbMLKBQkBicCtHUtFB"

]

}

}

]

}

Jak widziałeś, aby utworzyć transakcję, generujesz podpis na podstawie skrótu publicznego portfela i skrótu klucza prywatnego. Skrypt wyjściowy transakcji pobiera klucz publiczny i podpis i sprawdza, czy masz zgodność z hashem klucza publicznego. Jeśli wyniki są prawdziwe, możesz wydać monety; w przeciwnym razie nie możesz.

Uwaga: klucz publiczny widoczny w transakcji to rodzaj transakcji o nazwie Pay to Pubkey (P2PK). Ukryty klucz publiczny, którego używasz, to rodzaj transakcji o nazwie Pay to PubKey Hash (P2PKH).

Transakcję podpiszesz przez P2PKH, aby dopasować ją do typu Twojego portfela. Istnieją dwa sposoby podpisania transakcji; możesz użyć signrawtransactionwithkey lub signrawtransactionwithportfel. Te dwie podpisane metody są dostępne w 0.18.0 RPC, w tym dane wejściowe dla transakcji surowych w serializowanym formacie zakodowanym szesnastkowo. Format polecenia signrawtransactionwithwallet wygląda następująco:

signrawtransactionwithwallet "hexstring" ([{"txid":"id","vout":

n,"scriptPubKey":"hex","redeemScript":"hex"},...] sighashtype)

Zauważ, że polecenie signrawtransactionwithwallet pozwala ci dołączyć drugi argument o nazwie „prevtxs”. „prevtxs” jest sformatowany jako tablica zawierająca poprzednie dane wyjściowe transakcji. Jeśli zdecydujesz się wykorzystać i wstawić wartość dla „prevtxs”, transakcja będzie zależeć od poprzedniej transakcji, która może jeszcze nie znajdować się w łańcuchu bloków. Jeśli nie potrzebujesz tej funkcji, po prostu ustaw „prevtxs” na null. Format polecenia signrawtransactionwithkey wygląda następująco:

signrawtransactionwithkey "hexstring" ["privatekey1",...]

Zauważ, że drugi argument to zakodowana algorytmem base58 tablica kluczy prywatnych, które będą jedynymi kluczami używanymi do podpisywania transakcji. Trzecim opcjonalnym argumentem jest tablica poprzednich danych wyjściowych transakcji, od których ta transakcja zależy, ale może jeszcze nie znajdować się w łańcuchu bloków. W naszym przypadku nie podasz drugiego argumentu, ponieważ Twoja transakcja nie musi być uzależniona od innych warunków.

> bitcoin-cli signrawtransactionwithwallet $rawtxhex

{

"hex": "020000000010138451f62a8800b09a94504ae0ff20d7729e71ba

5a8a9f483d80ba9739b1ce9500000000017160014c27b4e6bd8eb

821ee80a239e0edd59070f57233dffffffff0140420f0000000000

1976a9149f9a7abd600c0caa03983a77c8c3df8e062cb2fa88

ac0247304402205cc4b04859e34aa6b1e924745f33a7643fbe45

fcd6e900fdaa29281feae3f8f6022059d4083a3cf81c3bb8226

7931660afb8ffc4bae87ede8dfa11efcb6af6a14ac90121028

926735fcd5bf6580e6f669c240da8975dddf23a6d4015e

4e0bc1ca3f1d2b7f100000000",

"complete": true

}

Poprzednie polecenie zwróciło podpisane, zakodowane szesnastkowo dane w odpowiedzi JSON. Użyj tych danych, aby ustawić szesnastkę dla zmiennej ze znakiem signtx.

> signedtx="020000000010138451f62a8800b09a94504ae0ff20d7729e71

ba5a8a9f483d80ba9739b1ce9500000000017160014c27b4e6bd8eb821ee8

0a239e0edd59070f57233dffffffff0140420f00000000001976a9149f9a7

abd600c0caa03983a77c8c3df8e062cb2fa88ac0247304402205cc4b04859

e34aa6b1e924745f33a7643fbe45fcd6e900fdaa29281feae3f8f6022059d

4083a3cf81c3bb82267931660afb8ffc4bae87ede8dfa11efcb6af6a14ac9

0121028926735fcd5bf6580e6f669c240da8975dddf23a6d4015e4e0bc

1ca3f1d2b7f100000000"

Otóż to; możesz teraz wysłać transakcję za pomocą polecenia sendrawtransaction.

> bitcoin-cli sendrawtransaction $signedtx

ff75dbb08da6f4dc6463dd32d8f9b1a4781e1eeee338e93e8282

0d0fdfbd43ff

Dane wyjściowe dostarczają odpowiedź txid, którą możesz sprawdzić w Eksploratorze Blockchain, tak jak wcześniej. Możesz również sprawdzić, czy środki zostały usunięte z Twojego konta za pomocą polecenia getbalance.

> bitcoin-cli getbalance

0.09413028

Jak również polecenie słuchania.

> bitcoin-cli listunspent

[

{

"txid": "be05d068d1245f1c60ea4229c00eb5e96f2a5c5527f1de

b7c6de5e1e20a4b4db",

"vout": 1,

"address": "2MveVhMe6PTzuhsJHx5zXAjDBwQvzdyqGjM",

"redeemScript": "00142e29123ba343c577ab9517ede9b74f047d

2c2ea3",

"scriptPubKey": "a914254f0e95fb26c0f29975f866e69543519b

f565e787",

"amount": 0.09413028,

"confirmations": 86,

"spendable": true,

"solvable": true,

"safe": true

}

]

Dodatkowo możesz wyświetlić transakcję za pomocą polecenia listtransactions.

> bitcoin-cli listtransactions

[

…

{

"address": "mv4rnyY3Su5gjcDNzbMLKBQkBicCtHUtFB",

"category": "send",

"amount": -0.01000000,

"label": "",

"vout": 0,

"fee": -0.08092813,

"confirmations": 1,

"blockhash": "0000000000000016ba1c314375d9bb17b6a857e091fd

4924bda5c9d7d9a2fd15",

"blockindex": 1,

"blocktime": 1542070705,

"txid": "ff75dbb08da6f4dc6463dd32d8f9b1a4781e1eeee338e93e82

820d0fdfbd43ff",

"walletconflicts": [

],

"time": 1542070656,

"timereceived": 1542070656,

"bip125-replaceable": "no",

"abandoned": false

}

]

Transakcje wymagające multipodpisu

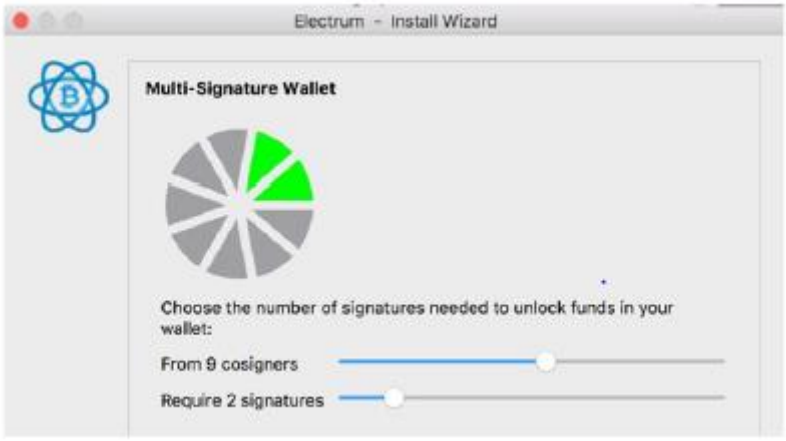
Do tej pory robiłeś standardowe „transakcje z jednym podpisem”, bo do podpisania transakcji i wykonania przelewu potrzebny był tylko jeden sygnatariusz z jednym podpisem. Jednak sieć bitcoin obsługuje bardziej skomplikowaną transakcję. Te transakcje można ustawić tak, aby wymagały podpisu wielu sygnatariuszy. Na przykład instytucje, partnerzy, małżonkowie lub zaprogramowane skrypty mogą chcieć, aby wszystkie strony podpisały się zamiast tylko jednej. W takich przypadkach przed wysłaniem środków potrzebne byłyby wszystkie klucze prywatne użytkowników. Aby przeprowadzić transakcję z wieloma sygnatariuszami, utworzysz dwa oddzielne portfele do testowania. Możesz uruchomić bitcoin core na dwóch oddzielnych maszynach i użyć wywołań RPC do wygenerowania nowego adresu publicznego dla każdego portfela lub możesz pobrać portfel Electrum ze strony https://electrum.org/#download i uruchomić go w trybie testnet, aby wygenerować Twój drugi portfel. Jako pierwszy przykład uruchomisz Electrum, ponieważ możesz użyć wbudowanego portfela z wieloma podpisami, aby zrozumieć ten proces. Po zakończeniu pobierania Electrum uruchom Electrum jako sieć testową za pomocą wiersza poleceń.

> open -n /Applications/Electrum.app --args –testnet

Ustawianie Electrum za pomocą portfela Multisignature

Po uruchomieniu Electrum wybierz "Portfel z wieloma podpisami" dla opcji tworzenia portfela, a następnie kliknij Dalej. Patrz rysunek

Na następnym ekranie możesz wybrać, ilu współsygnatariuszy jest wymaganych i ile podpisów jest potrzebnych. Transakcje te są często określane jako transakcje M-of-N, na przykład scenariusz 2 z 3. 2 z 3 oznaczałoby, że potrzebujesz co najmniej dwóch kluczy prywatnych (podpisów) od trzech współsygnatariuszy, aby autoryzować transakcję. Możesz przesuwać suwaki, aby lepiej zrozumieć tę funkcję, jak pokazano na rysunku



Tutaj wybierz portfel z wieloma podpisami 2 z 2, co oznacza dwóch współsygnatariuszy i dwa podpisy. Następnie kliknij przycisk Dalej. Na następnym ekranie kliknij "Utwórz nowe ziarno" i kliknij przycisk Dalej. Na następnym ekranie możesz wybrać rodzaj nasion. Standard oznacza P2PKH lub SegWit, co oznacza P2SH-SEGWIT, więc wybierz Standard i kliknij Dalej. W następnym kroku otrzymasz ziarno, które reprezentuje Twój klucz prywatny. Przechowuj swoje nasiona i uważaj, aby nie udostępniać ich nikomu. Następnie otrzymujesz to, co Electrum nazywa głównym kluczem publicznym, i zostajesz poproszony o udostępnienie go swoim współpracownikom, jak pokazano na rysunku

Uwaga: publiczny klucz główny Electrum jest częścią Electrum

Hierarchiczny portfel deterministyczny (HD), który generuje dla Ciebie adres na podstawie głównego materiału siewnego, którego można użyć do utworzenia kopii zapasowej wszystkich funduszy. Seed składa się ze słów używanych do pobierania kluczy prywatnych portfela; utrata nasion oznaczałaby utratę kluczy prywatnych

Kliknij Dalej, aby wprowadzić klucz publiczny lub prywatny podpisującego. Patrz rysunek

Na następnym ekranie kreatora będziesz używać głównego klucza prywatnego portfela swojego rdzenia bitcoin, aby umożliwić Electrum podpisanie drugiego portfela w Twoim imieniu. Możesz pobrać klucz prywatny z pliku kopii zapasowej klucza prywatnego. Pokazuje się pod rozszerzonym prywatnym kluczem głównym.

> vim /Users/[location]/mywallet.txt

extended private masterkey: [key]

Kreator Electrum ustawia dla Ciebie współsygnatariuszy, a następny krok kreatora instalacji poprosi Cię o ustawienie hasła, jeśli chcesz, dla dodatkowego bezpieczeństwa. Otóż to. Teraz, gdy kreator zakończył konfigurację Twojego konta, możesz wysyłać i odbierać środki zi do swojego portfela cosigner. Kliknij Odbierz u góry, aby uzyskać adres portfela, jak pokazano na rysunku

Będziesz ponownie korzystać z Coinfaucet.eu, aby zasilić swój nowy portfel:

https://coinfaucet.eu/en/btc-testnet/.

Następnie możesz odesłać te monety z powrotem do portfela Coinfaucet.eu

adres po potwierdzeniu monet; oto adres portfela Coinfaucet.eu:

2N7RzS3j2eKHVj1E5yV7iGuwfgUtobrCnrc

Ponieważ podałeś oba klucze prywatne osoby podpisującej, ta transakcja zostanie wykonana za pomocą polecenia send. Jeśli jednak ustawisz dwa konta i podasz tylko jeden klucz publiczny, drugi cosigner będzie musiał zatwierdzić tę transakcję na swoim koncie, zanim polecenie send faktycznie wyśle monety. Podobnie możesz wykonać tę transakcję za pomocą wiersza poleceń RPC.

Aby rozpocząć, kliknij Plik ➤ Usuń u góry Electrum, aby utworzyć standardowy portfel zamiast portfela cosigner. Po usunięciu tego portfela możesz zacząć od nowa i utworzyć nowy portfel standardowy (P2PKH), którego będziesz używać jako drugi cosigner. Aby pobrać adres swojego portfela, kliknij link Wyświetl u góry, a następnie kliknij Adresy. Następnie kliknij prawym przyciskiem myszy adres, dla którego chcesz zobaczyć jego klucz publiczny. Spowoduje to wyświetlenie klucza publicznego adresu. Patrz rysunek .

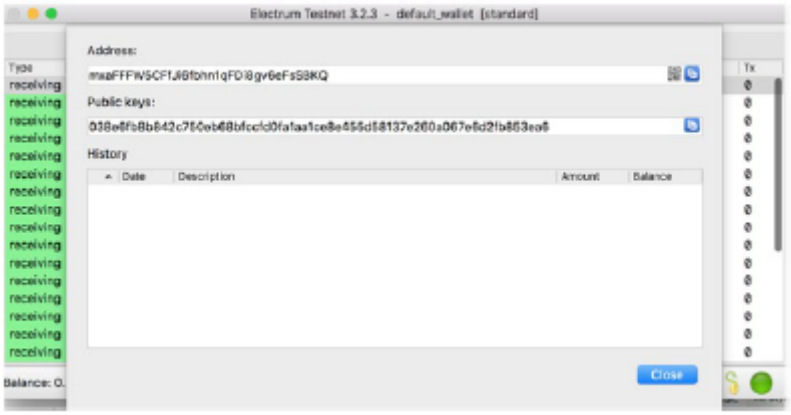
— Oto przykładowy adres portfela:

mxaFFFW5CFfJi6fbhn1qFDi8gv6eFsSBKQ

— Oto klucz publiczny w przykładzie:

038e6fb8b842c750eb68bfccfd0fa1aa1c

e8e455d58137e260a067e6d2fb853ea6



Następnie utworzysz nowy adres dla swojego cosignera za pomocą wiersza poleceń RPC.

> bitcoin-cli getnewaddress

2Msggcttx7wDDbcib6yD8ng2oKRdq8Bz4wV

Następnie możesz ustawić adresy dwóch współsygnatariuszy.

> address1=2Msggcttx7wDDbcib6yD8ng2oKRdq8Bz4wV

> address2=mxaFFFW5CFfJi6fbhn1qFDi8gv6eFsSBKQ

Upewnij się, że adres jest poprawny za pomocą polecenia validateaddress.

> bitcoin-cli validateaddress $address2

Aby utworzyć portfel cosigner, potrzebujesz kluczy publicznych obu osób podpisujących. Masz już klucz publiczny portfela Electrum; teraz potrzebujesz klucza publicznego RPC bitcoin core. Aby to uzyskać, użyj polecenia getaddressinfo, aby przyjrzeć się odpowiedzi RPC JSON i zmiennej pubkey.

> bitcoin-cli getaddressinfo $address1

{

"address": "2Msggcttx7wDDbcib6yD8ng2oKRdq8Bz4wV",

"scriptPubKey": "a91404d0a132b5796d4462f39865d56af4ff7255d1b

287",

"ismine": true,

"iswatchonly": false,

"isscript": true,

"iswitness": false,

"script": "witness_v0_keyhash",

"hex": "001440bbb1a949badb3a12a941a44bc994f7127c595c",

"pubkey": "034ffed96ffc416b90daa97df5c09b618d7fbf99076ed8100

900cfa0890e763ac0",

"embedded": {

"isscript": false,

"iswitness": true,

"witness_version": 0,

"witness_program": "40bbb1a949badb3a12a941a44bc994f7127c595c",

"pubkey": "034ffed96ffc416b90daa97df5c09b618d7fbf99076ed81

00900cfa0890e763ac0",

"address": "tb1qgzamr22fhtdn5y4fgxjyhjv57uf8ck2u4glnj9",

"scriptPubKey": "001440bbb1a949badb3a12a941a44bc994f7127c5

95c"

},

"label": "",

"timestamp": 1541782726,

"hdkeypath": "m/0'/0'/9'",

"hdseedid": "572deaa922cbf31076701942878c3e5fc2e23b60",

"hdmasterkeyid": "572deaa922cbf31076701942878c3e5fc2e23b60",

"labels": [

{

"name": "",

"purpose": "receive"

}

]

}

Teraz jesteś gotowy, aby utworzyć adres z wieloma podpisami swoich współsygnatariuszy za pomocą polecenia createmultisig, ponieważ masz klucze publiczne obu osób podpisujących.

> bitcoin-cli -named createmultisig nrequired=2 keys="'["034ffe

d96ffc416b90daa97df5c09b618d7fbf99076ed8100900cfa0890e763ac0",

"038e6fb8b842c750eb68bfccfd0fa1aa1ce8e455d58137e260a067e6d2

fb853ea6"]"'

{

"address": "2MtBkhgVLJ6VA1nFbjam36iUY1dCiWFf4ix",

"redeemScript": "5221034ffed96ffc416b90daa97df5c09b618d7fbf99

076ed8100900cfa0890e763ac021038e6fb8b842c

750eb68bfccfd0fa1aa1ce8e455d58137e260a0

67e6d2fb853ea652ae"

}

Następnie musisz wybrać txid UTXO i vout, aby podpisać transakcję, tak jak w poprzednich surowych transakcjach.

> bitcoin-cli listunspent

[

{

"txid": "ea3fb46ab103d15120e02ed6b60e3d83b265fed26794e3ed

739496b62445410b",

"vout": 0,

...

]

Następnie ustawiasz właściwość utxo_txid.

> utxo_txid=ea3fb46ab103d15120e02ed6b60e3d83b265fed26794e3ed73

9496b62445410b

> utxo_vout=0

> recipient="mv4rnyY3Su5gjcDNzbMLKBQkBicCtHUtFB"

> rawtxhex=$(bitcoin-cli -named createrawtransaction

inputs="'[{ "txid": "'$utxo_txid'", "vout": '$utxo_vout' }]"'

outputs="'{ "'$recipient'": 0.001}"')

Now decode and set the hexstring property.

> bitcoin-cli -named decoderawtransaction hexstring=$rawtxhex

> bitcoin-cli signrawtransactionwithwallet $rawtxhex

{

"hex": "020000000001010b414524b6969473ede39467d2fe65b2833d0eb

6d62ee02051d103b16ab43fea00000000017160014040c578cf60bf

00980bfde1920f54459eaab3a09ffffffff01a086010000000000

1976a9149f9a7abd600c0caa03983a77c8c3df8e062cb2fa88ac0

24730440220603883ace41bdf5cf85c87e80f7362b45e35949114

f46ac5e5b89f5e13d8d95002205c5eb45ca7de8b2da88c41c4311

711beb14e8e0d679e40d1fbc2cb8e81e053fb01210205e848e0f2

2dfe0c428d02c356d0c9a8d064a789a6bbcaa43a245d701948aba

200000000",

"complete": true

}

Lastly, sign your transaction via the signedtx command.

> signedtx="020000000001010b414524b6969473ede39467d2fe65b283

3d0eb6d62ee02051d103b16ab43fea00000000017160014040c578cf

60bf00980bfde1920f54459eaab3a09ffffffff01a0860100000000

001976a9149f9a7abd600c0caa03983a77c8c3df8e062cb2fa88ac

024730440220603883ace41bdf5cf85c87e80f7362b45e35949114

f46ac5e5b89f5e13d8d95002205c5eb45ca7de8b2da88c41c43117

11beb14e8e0d679e40d1fbc2cb8e81e053fb01210205e848e0f22dfe0

c428d02c356d0c9a8d064a789a6bbcaa43a245d701948aba200000000"

Jesteś gotowy do wysłania transakcji przy użyciu wartości sendrawtransaction.

> bitcoin-cli sendrawtransaction $signedtx

Wymienne transakcje i czas blokady

Tworząc RawTransaction za pomocą polecenia createrawtransaction możesz dodać dwie dodatkowe zmienne, których możesz użyć: locktime i replaceable.

createrawtransaction [{"txid":"id","vout":n},...] [{"address":a

mount},{"data":"hex"},...] (locktime) (wymienne)

Jak sama nazwa wskazuje, wymienny umożliwia zastąpienie surowej transakcji nową transakcją z wyższymi opłatami. Dzieje się tak, gdy ustalona opłata jest zbyt niska, co powoduje, że transakcja nie dochodzi do skutku. Na przykład, jeśli opłata, którą próbujesz zapłacić, jest zbyt wysoka, możesz otrzymać następujący komunikat o błędzie:

absurdalnie wysoka opłata, 11563419 > 10000000 (kod 256)

Rdzeń Bitcoin obsługuje argument locktime w surowej transakcji; ten argument umożliwia wysyłanie transakcji w pewnym momencie w przyszłości i dopóki nie zostaną wysłane, nadawca może anulować transakcję. Istnieją dwie opcje. Wysokość bloku jest używana dla małych liczb, a znaczniki czasu UNIX są używane dla dużych liczb. Te argumenty oznaczają, że transakcja nie zostanie wstawiona do bloku, dopóki warunki nie zostaną spełnione.

Uwaga: Wysokość bloku to liczba bloków w łańcuchu między dowolnym konkretnym blokiem a pierwszym blokiem łańcucha w łańcuchu.

Kolorowe monety Bitcoin

Transakcje Bitcoin posiadają właściwość o nazwie OP_RETURN. Ta właściwość może służyć do przechowywania do 80 bajtów danych, które można wykorzystać do przekazywania danych. To może wydawać się niewiele, ale wystarczy, aby potwierdzić własność lub przekazać małe fragmenty danych w celu uwierzytelnienia. Wykorzystanie właściwości OP_RETURN odbywa się poprzez ustawienie słowa kodowego danych we właściwości vout transakcji. Aby przekazać dane, które chcemy uwzględnić w Twojej transakcji, nadal musisz przesłać środki, aby transakcja została uwzględniona w blockchain, ale możesz ustawić odbiorcę jako swój własny portfel na wypadek, gdybyś nie chciał komuś płacić. W ten sposób możesz przechowywać dane w Blockchain trwałości Bitcoin i musisz tylko uiścić opłatę transakcyjną, ponieważ nie płacisz nikomu.

Uwaga: OP_RETURN to skrypt kodu, który definiuje transakcję jako ważną lub nieważną; można go użyć do wstawienia danych do transakcji, co spowoduje przechowywanie tych danych w łańcuchu bloków bitcoin. Należy pamiętać, że istnieją różne opinie na temat tego, czy można korzystać z tej właściwości. Niektórzy uważają, że przechowywanie danych niebędących walutą w łańcuchu bloków to zły pomysł; ponieważ istnieją mniej kosztowne i wydajniejsze sposoby przechowywania danych, tak naprawdę zależy to od użytkowania.

Wysyłanie Transakcji z OP_RETURN

Zanim ustawisz transakcję, będziesz chciał wprowadzić mały

lekki program narzędziowy o nazwie jq, który usprawnia tworzenie obiektu RawTranaction. Jest to procesor JSON wiersza polecenia, którego można użyć do przetworzenia kodu JSON RPC w terminalu. Zainstaluj go za pomocą Brew.

> brew install jq

Narzędzie jq umożliwia odzyskanie fragmentów zwróconego JSON, dzięki czemu będziesz mógł przesyłać strumieniowo transakcję szybciej i z mniejszą liczbą błędów. Następnie możesz ustawić niektóre dane do wysłania za pomocą parametru OP_RETURN. Ten przykład utworzy MD5 dla pliku. W prawdziwym życiu może to być wersja umowy między stronami lub dowolny fragment kodu, którego potrzebujesz.

Uwaga: Algorytm Message-Digest 5 (MD5) to funkcja, która generuje 128-bitową wartość skrótu. Często tworzy się plik, który zawiera pliki z sumami kontrolnymi i zapewnia integralność danych, ponieważ każda zmiana pliku skutkowałaby nowym wynikiem MD5.

Możesz wybrać jeden z podstawowych plików bitcoin, takich jak config.log, aby wygenerować skrót MD5 i ustawić zmienną op_return_data.

> md5 config.log

MD5 (config.log) = 634ef85e038cea45bd20900fc97e09dc

> op_return_data="634ef85e038cea45bd20900fc97e09dc"

Jak widzieliśmy wcześniej w tym rozdziale, możesz użyć polecenia listunspent, aby wybrać swoje UTXO, które chcesz wydać.

> bitcoin-cli listunspent

Teraz korzystając z narzędzia jq, możesz przesyłać strumieniowo proces, dzięki czemu nie musisz kopiować i wklejać i możesz uniknąć błędów.

> utxo_txid=$(bitcoin-cli listunspent | jq -r '.[0] | .txid')

> utxo_vout=$(bitcoin-cli listunspent | jq -r '.[0] | .vout')

> recipient=$(bitcoin-cli getrawchangeaddress)

Zwróć uwagę na kilka rzeczy. Tutaj ustawiasz pierwszy element JSON [0], ale możesz ustawić dowolny element, na przykład [1] lub [2]. Zauważ też, że musisz uruchomić polecenie listunspend, aby dowiedzieć się, jaką „kwotę” ma UTXO. W tym przykładzie kwota wynosi 0,1166341, a ponieważ chcesz zapłacić 0,00000200 opłat (200 satoshi), wyślesz łącznie 0,1166321. Jeśli nie ustawisz prawidłowo opłaty, możesz wydać zbyt dużo na opłaty lub otrzymać komunikat o błędzie, taki jak:

- nie spełniono minimalnej opłaty sztafetowej, 29 < 161 (kod 66)

- absurdalnie wysoka opłata, 24432219 > 10000000 (kod 256)

Możesz użyć polecenia echo, aby upewnić się, że zmienna jest poprawnie ustawiona. Następnie możesz kontynuować i ustawić dane swojej transakcji.

> rawtxhex=$(bitcoin-cli -named createrawtransaction

inputs="'[{ "txid": "'$utxo_txid'", "vout": '$utxo_vout' }]"'

outputs="'{ "data": "'$op_return_data'", "'$recipient'":

0.1166321}"')

Następnie musisz podpisać i wysłać transakcję.

> signtx=$(bitcoin-cli signrawtransactionwithportfel

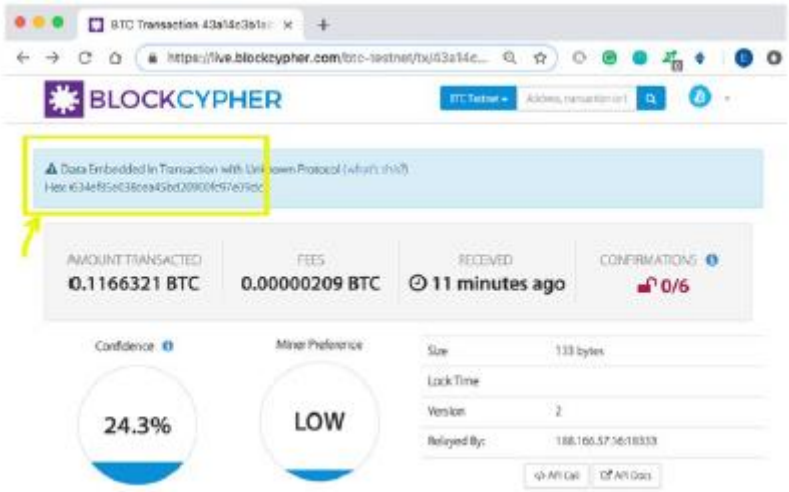
$rawtxhex | jq -r '.hex')

> bitcoin-cli sendrawtransaction $signedtx

43a14c3b1ac446e4774c5338e5ae4e23839ab65a38c45da8b790f44

49b090ae5

Teraz możesz śledzić obiekt RawTransaction w księdze testnet Blockchain Explorer, jak pokazano na rysunku



Jak widać na rysunku 4-18, otrzymujesz komunikat „Dane osadzone w transakcji z nieznanym protokołem”. Jeśli miałbyś zaprojektować oprogramowanie, które regularnie korzysta z tej metody, chciałbyś dołączyć słowo kluczowe, aby zidentyfikować swoje dane.

Kolorowe monety Bitcoina

Nazwa kolorowych monet utknęła ze starszych implementacji protokołu EPOBC w rdzeniu bitcoin, w których zasób jest powiązany z satoshi (stąd „kolorowanie”). Teraz możesz osiągnąć kolorowanie za pomocą parametru OP_RETURN. OP_RETURN pokolorował twoje monety i zapewnił nową możliwość dla łańcucha bloków bitcoin, ponieważ można było osadzić dane, które dostarczyły dowodu własności. Możesz także ustawić inne warunki, aby miały miejsce w określonym czasie lub przekazać dane związane z transakcją, którą wstawiłeś do łańcucha bloków. OP_RETURN jest potężnym narzędziem, a w dalszej części zobaczysz, jak OP_RETURN jest wykorzystywany w projektach klasy produkcyjnej do rozwiązywania wszelkiego rodzaju problemów.

Streszczenie

W tej części zagłębiłeś się w podstawowe RPC bitcoinów. Wygenerowałeś portfele bitcoinowe i SegWit, i byłeś w stanie odzyskać klucze prywatne portfela i lepiej zrozumieć algorytm podpisu cyfrowego krzywej eliptycznej (ECDSA) oraz sposób tworzenia kluczy publicznych i prywatnych. Większość czasu poświęciłeś na analizowanie transakcji; wysłałeś monety za pomocą demona bitcoina w sieci testowej, a także wykorzystałeś GUI portfela podstawowego bitcoina do wysyłania monet. Po wysłaniu monet nauczyłeś się, jak przeglądać swoje transakcje w Block Explorerze bitcoina. Kontynuowałeś, zaglądając

do RawTransaction i dowiedziałeś się, jak generować transakcje z jednym wyjściem, a także bardziej złożone transakcje z wieloma sygnatariuszami za pośrednictwem Electrum, a także wiersza poleceń. Dodatkowo poznałeś inne opcje, takie jak zamiana transakcji na zmianę opłaty, a także ustawienie zmiennej locktime. Poznałeś różnicę między P2PKH a P2SH-SEGWIT. Na koniec omówiłem, jak przekazywać dane za pomocą parametrów OP_RETURN, które mogą być używane do monet w kolorze bitcoin lub po prostu do przekazywania dodatkowych danych za pomocą łańcucha bloków bitcoina do czegoś więcej niż wydawanie monet.