Bitcoin transakcije

RGKK 2024./2025.
Prva laboratorijska vježba

Uvod

U okviru prve laboratorijske vježbe koristit ćete Bitcoinov skriptni jezik Script kako biste programirali Bitcoin transakcije koje će se izvršavati na testnet instanci lanca blokova. Vježba se sastoji od tri zadatka koje možete riješiti koristeći predložak početnog koda napisanog u programskom jeziku Python, uz pomoć biblioteke python-bitcoinlib.

Upute

Umjesto pokretanja standardnog Bitcoin čvora koji bi preuzimao i verificirao cjelokupnu povijest transakcija, poslužit ćemo se samo s nekoliko *Python* skripti kojima ćemo stvarati i emitirati transakcije na BlockCypher testnet lanac blokova. U tu svrhu koristit ćemo biblioteku python-bitcoinlib, koja pruža jednostavno "low-level" sučelje prema podacima i protokolu Bitcoina.

Početni kod koristi aplikacijsko programsko sučelje BlockCypher API kako bi emitirao vaše transakcije na BlockCypher Testnet lanac blokova. Nakon završetka i pokretanja koda za svaku vježbu, novonastala transakcija prikazat će se u formatu JSON. Primjer takvog prikaza kao i značenje svih polja možete pronaći na https://www.blockcypher.com/dev/bitcoin/#tx. U sklopu ove vježbe, nama će od posebnog značaja biti polja `hash`, `input` i `output`.

Da bismo uopće mogli uspješno izvršavati transakcije, potrebni su nam novčići. Iste ćemo zatražiti s BlockCypher Testnet fauceta pomoću tokena koji ćemo dobiti nakon otvaranja korisničkog računa. Transakcijama na BlockCypher Testnetu pristupit ćemo koristeći *online* preglednik blokova dostupan na adresi: https://live.blockcypher.com/bcy.

Početni kôd

Dohvatite početnu okolinu `rgkk-2024-25-lab1` s repozitorija predmeta i pozicionirajte se unutar vršnog direktorija. Prvo kreirajte i aktivirajte Python virtualnu okolinu:

```
python3 -m venv .venv
source .venv/bin/activate
```

Nakon toga, dohvatite i instalirajte potrebne biblioteke:

```
pip install -r requirements.txt
```

U početnom kôdu postoji nekoliko datoteka i direktorija:

- `lib/config.py`

  U ovu datoteku **trebate upisati** svoj privatni ključ dobiven pomoću BlockCypher API.

- `lib/keygen.py`

  Ovu skriptu koristit ćete za generiranje novih privatnih ključeva i pripadnih adresa. Datoteku nije potrebno modificirati.

- `lib/split_test_coins.py`

  Ovu skriptu koristit ćete da podijelite jedan nepotrošeni izlaz (*unspent transaction output* [UTXO]) na više njih. Svaki zadatak zahtjevat će od vas da upotpunite transakciju koja će trošiti jedan novonastali UTXO. U normalnim okolnostima dovoljno je podijeliti početni UTXO na tri nova. Međutim, zbog mogućih grešaka prilikom pisanja skripti preporučujemo se da početni UTXO podijelite na **barem šest** novih. U datoteci je potrebno navesti parametre transakcije: adresu transakcije koja sadrži UTXO koji želite podijeliti, indeks UTXO-a i iznos koji želite potrošiti (imajući na umu *fee* koji ide rudarima).

- `lib/utils.py`

  Sadrži pomoćne metode. Datoteku nije potrebno modificirati.

- `Q1.py`, `Q2a.py`, `Q2b.py`, `Q3a.py`, `Q3b.py`

  Ove datoteke **trebate nadopuniti** u svrhu rješavanja zadataka. Zadaci zahtjevaju da napišete skripte *scriptSig* (*unlocking*) i *scriptPubKey* (*locking*) kao i da navedete parametre transakcije. Zadatk *multisig* dodatno zahtjeva generiranje novih privatnih ključeva pomoću skripte `lib/keygen.py`

- `doc/transactions.py`

  U ovu datoteku **trebate upisati** adrese transakcija nastalih rješavanjem zadataka.

Početni kod testiran je na operacijskom sustavu Ubuntu 22.10. Ako budete imali problema s pokretanjem (moguće zbog nepodržane verzije *openssl*), molimo vas da, ako je moguće, pokušate vježbu pokrenuti na drugom računalu (na primjer, na računalu u praktikumu) prije nego što nas kontaktirate.

Priprema

1. Posjetite https://accounts.blockcypher.com, otvorite korisnički račun i generirajte token. Zatim upotrijebite dobiveni token (zalijepite ga nakon `token=`) kako biste zatražili novi par ključeva i pripadajuću adresu (nakon što zalijepite naredbu u terminal ponovno pretipkajte navodnike):

   ```
   curl -X POST 'https://api.blockcypher.com/v1/bcy/test/addrs?token=$TOKEN'
   ```

   Privatni ključ koji dobijete zalijepite na odgovarajuće mjesto unutar skripte `lib/config.py`.

2. Da bismo mogli izvršavati transakcije potrebni su nam novčići. Zalijepite token i adresu na odgovarajuća mjesta u sljedećoj naredbi da zatražite novčiće (nakon što zalijepite naredbu u terminal ponovno pretipkajte navodnike i vitičaste zagrade).

```
curl -d '{"address": "${ADRESA}", "amount": 100000}' \
https://api.blockcypher.com/v1/bcy/test/faucet?token=$TOKEN
```

Gornji poziv će generirati transakciju koja će prebaciti 100000 BlockCypher satoshija (0.001 BCY) na vašu adresu i vratiti identifikator transakcije (*txid*).

3. Transakciju možete pogledati u BlockCypher *online* pregledniku blokova dostupnom na adresi https://live.blockcypher.com/bcy tako da unesete odgovarajući *txid* (ili adresu). Navedena transakcija bi trebala imati dva UTXO-a jedan od kojih je vaš (onaj koji sadrži vašu adresu).

4. Kao što smo prije napomenuli, preporuča se da podijelite dobiveni UTXO na barem šest novih; tri su vam potrebana za rješavanje zadataka, ali zbog mogućih grešaka (primjerice ako nepovratno zaključate novčiće) preporuča se da ovaj broj bude veći. Modificirajte skriptu `lib/split_test_coins.py` tako da navedete

   - `txid_to_spend`: *hash* od prethodno dobivene transakcije *faucet* koja sadrži UTXO kojeg želite podijeliti.
   - `utxo_index`: indeks UTXO-a u navedenoj trasakciji (indeksi počinju od nula).
   - `n`: broj novih UTXO-a koji će nastati.
   - `amount_to_send`: cijeli iznos koji je zaključan, umanjen za transakcijske troškove (*fee*). Imajte na umu da rudari na mreži **ne prihvaćaju** transakcije koje imaju premali *fee*; dovoljno je da *fee* bude reda veličine 0.00001 BCY. BlockCypher periodično **uklanja** transakcije koje nemaju nijednu potvrdu (*confirmation* unutar preglednika), stoga je bitno da ne zaboravite uključiti *fee*!

   Prosječno vrijeme rudarenja bloka na BlockCypher Testnetu je jedna minuta. Transakcija na BlockCypher Testnetu mora imati **barem šest potvrda** da bismo mogli potrošiti pripadni UTXO.

Zadatci

Svaki zadatak zahtjeva da napišete odgovarajuće Bitcoin skripte *scriptSig* i *scriptPubKey* te da navedete parametre transakcije. Informacije o naredbama (*opcodes*) koje programski jezik Bitcoin Script podržava možete naći na stranici https://en.bitcoin.it/wiki/Script. Biblioteka python-bitcoinlib dozvoljava da imena naredbi navodite doslovno, onako kako se nazivaju na prethodnoj povezici. Primjerice, sljedeća funkcija vraća skriptu *scriptPubKey* koja ne može biti potrošena i služi za skladištenje proizvoljne informacije.

```
def save_message_scriptPubKey(message):
  return [ OP_RETURN,
           message,
           365 ]
```

Primjeri nekih naredbi koje ćete koristiti su `OP_DUB`, `OP_EQUALVERIFY`, i `OP_CHECKMULTISIG`. Također, za razliku od jezika *Script*, python-bitcoinlib olakšava unos brojeva koji ne stanu u jedan bajt. Primjerice, da bismo umetnuli dekadski broj 365 na vrh stoga koristeći python-bitcoinlib dovoljno je broj navesti unutar skripte kao na prethodnom primjeru, dok je kod jezika *Script* potrebno prvo navesti broj bajtova, a zatim *little-endian* reprezentaciju broja u heksadecimalnom zapisu: `OP_PUSHDATA1 02 6d01`.

Ako skripta *scriptSig* (koja „otključava" novčiće) nije valjana ($\mathsf{Eval}(scriptSig \parallel scriptPubKey) = false$), python-bitcoinlib će baciti iznimku prije nego se pripadna transakcija objavi. U suprotnom, dobit ćemo informacije o objavljenoj transakciji u formatu JSON. U sklopu svakog zadatka provjerite je li novonastala transakcija završila na lancu blokova BlockCypher Testnet te, ako jest, spremite identifikator transakcije (`hash`) u datoteku `doc/transactions.py`.

1. (1 bod) U sklopu prvog zadatka nadopunite datoteku `Q1.py` na mjestima označenima s `TODO` tako da otključate novčiće koje posjedujete (na nekom od UTXO-a) i pošaljete ih natrag Bitcoin Testnet faucetu koristeći transakciju *PayToPublicKeyHash*. Funkcije unutar kojih trebate napisati Bitcoin skripte trebaju vratiti listu koja se sastoji samo od naredbi jezika *Script* i (nekih od) parametara proslijeđenih funkciji.

2. (3 boda) U drugom zadatku napisat ćete skriptu koja rješava linearnu jednadžbu.

   (a) (2 boda) Nadopunite datoteku `Q2a.py` na mjestima označenima s `TODO` tako da generirate transakciju koja se može otključati s parom (x, y) koji predstavlja rješenje sljedeće linearne jednadžbe:

   $$x + y = \text{prve 4 znamenke vašeg JMBAG-a}$$
   $$x - y = \text{zadnje 4 znamenke vašeg JMBAG-a}$$

   Prilagodite zadnju znamenku vašeg JMBAG-a tako da postoji cijelobrojno rješenje gore navedenog sustava (oba broja moraju biti iste parnosti). Koristite isključivo naredbe jezika *Script* za sve potrebne operacije (nije dozvoljeno koristiti operatore jezika Python).

   (b) (1 bod) Nadopunite datoteku `Q2b.py` na mjestima označenima s `TODO` tako da potrošite UTXO od prethodno objavljene transakcije (2a). Skripta treba samo „gurnuti" dva prirodna broja na stog.

3. (3 boda) Za kraj, napisat ćete transakciju *multi-sig* koja će uključivati četiri strane.

   (a) (2 boda) Nadopunite datoteku `Q3a.py` na mjestima označenima s `TODO` tako da generirate transakciju *multi-sig* koja se može potrošiti uz pomoć banke i bilo kojeg od tri preostala klijenta. Dakle, niti jedino banka niti klijenti bez sudjelovanja banke ne smiju moći potrošiti transakciju. Pretpostavite da vi igrate ulogu banke tako da privatni ključ banke odgovara vašem privatnom ključu; ključeve od klijenata generirajte koristeći `lib/keygen.py` i unesite ih na odgovarajuća mjesta unutar `Q3a.py`.

   (b) (1 bod) Nadopunite datoteku `Q3b.py` na mjestima označenima s `TODO` tako da potrošite prethodno generirani UTXO (3a). Možete koristit bilo koju valjanu kombinaciju ključeva i signatura, no uvjerite se da sve kombinacije koje bi trebale raditi uistinu i rade.

Predaja

Laboratorijsku vježbu možete rješavati sami ili u grupi od najviše dva studenta. Ako se odlučite raditi laboratorijsku vježbu u grupi, morate ispuniti obrazac: https://forms.gle/VBmp45gfEj22Kze27. **Obrazac morate ispuniti najkasnije do 5.11.2023 u 23:59h.**

Laboratorijsku vježbu predajete putem sustava Moodle tako da učitate datoteku koja mora sadržavati samo nadopunjen početni izvorni kod laboratorijske vježbe koji ste preuzeli (pobrinite se da ne uključite direktorije `.venv` i `__pycache__`). Naziv predane datoteke mora biti u obliku *rgkk-2024-25-lab1-prezime-ime-jmbag.zip*. **Rok za predaju laboratorijske vježbe je 11.11.2024 do 23:59h.**

Laboratorijska vježba ukupno nosi 10 bodova i ocijenit će se na temelju predanog izvornog koda (7 bodova) i **usmene obrane koja obuhvaća pitanja vezana za gradivo laboratorijske vježbe** (3 boda). Termini obrane trebali bi biti vidljivi su u vašim kalendarima; promjena termina dozvoljena je samo u **iznimnim okolnostima**. **Prag za prolaz** laboratorijske vježbe je riješen prvi zadatak.

Ukoliko radite u grupi, naziv datoteke neka uključi ime, prezime i JMBAG jednog od studenata iz grupe. Oba člana grupe trebaju **zajedno** pristupiti obrani u bilo kojem od dodijeljenih termina. Vrijedi `broj_bodova_grupe([A, B]):= min{broj_bodova_A, broj_bodova_B}`.

Pitanja za laboratorijsku vježbu moguća su putem elektroničke pošte rgkk@fer.hr ili konzultacija uz prethodni dogovor putem elektroničke pošte.

Korisne povezice

- Programski jezik *Script*: https://en.bitcoin.it/wiki/Script.
- Bibilioteka python-bitcoinlib: https://github.com/petertodd/python-bitcoinlib
- BlockCypher Testnet Explorer: https://live.blockcypher.com/bcy/
- Format transakcije koji vraća BlockCypher: https://www.blockcypher.com/dev/bitcoin/#tx
- Anatomija Bitcoin transakcije:
  https://privatekeys.org/2018/04/17/anatomy-of-a-bitcoin-transaction/