Eötvös Loránd Tudományegyetem
Természettudományi Kar

DIGITÁLIS FIZETŐESZKÖZÖK

Szakdolgozat

Kasnyik Adrienn Szilvia

Matematika BSc, Matematikai elemző szakirány

Témavezető: Villányi Viktória

Operációkutatási tanszék

2018

Budapest

Tartalomjegyzék

Köszönetnyilvánítás

Ezúton szeretnék ilyen formában is köszönetet mondani témavezetőmnek, Villányi Viktória tanárnőnek, aki tanácsaival, magyarázataival és szakértelmével jelentősen hozzájárult szakdolgozatom elkészüléséhez. Hálásan köszönöm a témát, mint ötletet, a közös rugalmas munkát, idejét, türelmét és a konzultációkat, ahol bármilyen kérdéssel bizalommal fordulhattam hozzá. Továbbá szeretném megköszönni a családomtól kapott feltétlen bizalmat és biztatást az egyetemi tanulmányaim során, barátaim és szaktársaim éveken át tartó segítségét és támogatását.

1. fejezet

Bevezetés

Egy témaválasztás sohasem könnyű egy szakdolgozat esetén, hiszen a matematika mindenhol előbukkan. A három év során szerettem volna valami újat is hozzátenni az eddig megszerezett tudásomhoz és egy olyan területtel foglalkozni, amelyben a matematika különböző oldalai fellelhetőek. A bitcoinoknál talán egyedül a tőzsde árfolyamingadozása vet csak fel elgondolkodtatóbb kérdéseket az emberekben. A digitális fizetőeszköz 2008-as debütálásával az emberek nagy része nem tudott mit kezdeni, hiszen a társadalmi réteg számára elérhetetlen és ismeretlen fogalom volt.

A dolgozat során foglakozni fogok a bitcoinnal, mint fogalommal és annak jellemzőivel, ezután taglalom a bitcoin tranzakciókat. Részletezem a bitcoin hálózatok és tranzakciók felépítését, a blokkok, blokkláncok szerkezetét, a hálózat működési elvét. Mindemellett többet foglalkozok egy tranzakció támadási valószínűségével, valamint azzal, hogy milyen támadási fajtákkal találkozhatunk. Egy tranzakciót, hogy elküldjünk egy másik félnek, alá kell írnunk saját digitális aláírásunkkal, ezzel egyfajta védelmet adva a hálózatnak. Az Ellipttic Curve Digital Signature Algorithm (ECDSA)-t használja a bitcoin rendszer. A negyedik fejezetben az ECDSA-t, mint fogalmat, az ezen alapuló elliptikus görbéket, az aláírási algoritmust és annak ellenőrzését fejtem ki bővebben.

2. fejezet

A bitcoinok

A félreértések elkerülése végett először bontsuk két komponensre a bitcoint. Egyrészt van egy bitcoin-the-token, egy kódrészlet, amely a digitális koncepció tulajdonjogát képviseli, másfelől van egy bitcoin-the-protocol, egy elosztott hálózat, amely a bitcoin-the-token egyenlegét könyveli. Mindkettőt „bitcoinnak" nevezik. A rendszer lehetővé teszi a felhasználók közötti közvetlen fizetéseket, azaz anélkül történik egy tranzakció, hogy egy központi hatóságon, például egy bankon vagy fizetési átjárón keresztül haladna a pénz. Elektronikusan hozták létre (számítógépeken világszerte szabad szoftvereket használva) és tartották fent napjainkig. Ez volt az első példa arra, amit ma digitális pénznemnek nevezünk, egy olyan növekvő eszközosztály, amely a hagyományos devizák egyes jellemzőit osztja meg, kriptográfiai ellenőrzés mellett.

Születése

A Satoshi Nakamoto nevet viselő pszeudonim szoftverfejlesztő 2008-ban javasolta a bitcoint, mint matematikai bizonyításon alapuló elektronikus fizetési rendszert egy októberben megjelent cikkében. Az ötlet az volt, hogy minden központi hatóságtól független csereeszközt állítsanak elő, amely elektronikusan biztonságos, ellenőrizhető és megváltoztatható módon átvihető. Azt azonban, hogy valójában ki Satoshi Nakamoto napjainkig nem tudni. Elektronikus formában fizethetünk vele a termékekért, ha a tranzakcióban szereplő felek készek arra, hogy ezt le is bonyolítsák. Ebben az értelemben olyan, mint a hagyományos dollár, euró, vagy jen, amelyeket szintén digitálisan forgalmaznak.

2.1. Jellemzői

Decentralizált

A bitcoin legfontosabb jellemzője, hogy decentralizált. A Föld lassan minden területén elterjedt dedikált számítógépeknek a nyílt hálózatával működik, amelyet egy önkéntes „kódoló csapat" tart fenn. Ez az, ami annyira vonzza a magánszemélyeket, csoportokat, cégeket arra, hogy a bitcoin kereskedést válasszák, a már-már kényelmetlen bankok és kormányzati intézmények pénzük feletti ellenőrzése helyett. A bitcoin megoldja az elektronikus pénznemek kettős kiadási problémáját [1] a kriptográfia és a gazdasági ösztönzők kombinációjának köszönhetően. Az elektronikus pénzpiaci pénznemekben ezt a funkciót a bankok teljesítik, amik a hagyományos rendszer irányítását biztosítják. Ezzel szemben a bitcoinnal a tranzakciók integritását az említett elosztott és nyílt hálózat tartja fenn, amelyet senki sem tulajdonít.

Korlátozott ellátású

A Fiat devizák (dollár, euró, jen stb.) korlátlan ellátással rendelkeznek míg a bitcoin nagyságát egy szorosan szabályozó algoritmus vezérli. Minden órában néhány új bitcoin „szivárog ki" . Ez a kiszivárgott bitcoin szám folyamatosan csökken egészen addig amíg el nem éri a 21 milliót. Ez a bitcoint vonzóbbá teszi eszközként, mert elméletben ha a kereslet növekszik és a kínálat megegyezik, az érték növekedni fog.

Pszeudonim

Miközben a hagyományos elektronikus fizetések küldőit általában azonosítják (ellenőrzési célokra, valamint a pénzmosás és egyéb jogszabályok betartása végett), a bitcoin felhasználói elméletileg fél ellenőrzésben működnek. Mivel nincs központi „validátor" , a felhasználóknak nem kell azonosítaniuk magukat, amikor bitcoint küldenek egy másik felhasználónak. Amikor tranzakciós kérelem érkezik a protokoll minden korábbi tranzakciót ellenőriz annak megerősítése érdekében, hogy a feladó rendelkezik a szükséges bitcoinnal, valamint a küldési jogosultsággal. A rendszernek tehát nem kell ismernie a személy

[1] amelyben a digitális eszközök könnyen másolhatók és újra felhasználhatóak

azonosságát. A gyakorlatban minden felhasználó azonosítható a tárcája címével, a tranzakciók pedig némi erőfeszítéssel nyomon követhetők, vagyis a hálózat átlátható (egy adott ügylet előrehaladása mindenki számára látható).

Változtathatatlan

A bitcoin hálózatokon végrehajtott bármely tranzakciót nem lehet megváltoztatni vagy visszafordítani az elektronikus pénzügyi tranzakciókkal ellentétben. Ha tranzakciót rögzítünk a hálózaton, és több mint egy óra eltelt, akkor már lehetetlen módosítanunk. Ez némi nyugtalanságot azért hagy maga mögött.

Mértékegysége

A bitcoin rendszer mértékegysége a bitcoin, jelei: BTC, XBT vagy b kicsi bitcoin betű. Legkisebb egységét satoshi-nak hívják, ami aránya a bitcoinhoz 1:00000001 azaz 1 a 100 millióhoz. Megkülönböztetünk továbbá milibitcoint (mBTC) és microbitcoint (BTC) is. Utóbbi lehetővé teszi a hagyományos elektronikus pénzzel járó mikrotranzakciókat.

2.2. Bitcoin pénztárcák

Mielőtt birtokba veszi bárki is a bitcoint valahol el kell tárolnia őket. Ezt a helyet nevezik walletnak, azaz pénztárcának, amely több magánkulcsot is tartalmaz. Egy befektetőnek több tárcája lehet, így véges sok magánkulcsa. Tehát a wallet nem megtévesztően nem a bitcoinokat tárolja, hanem az általa használt nyílt titkosítást tartalmazó titkos és nyílt kulcsokat. Egyfajta digitális személyi igazolványként funkcionál. A pénztárcák akár számítógépen és/vagy mobileszközön, fizikai tárolóeszközön vagy akár egy papírlapon is élhetnek, így megkülönböztetünk elektronikus, online, mobile, hardver és papír pénztárcákat is. A legbiztonságosabb a hardveres pénztárca, amelyet offline állapotban biztonságos helyen tárolhatunk, míg a legkevésbé biztonságos lehetőség egy online tárca, mivel a kulcsokat egy harmadik fél tartja.

3. fejezet

Tranzakciók és hálózat

3.1. Bitcoin tranzakciók

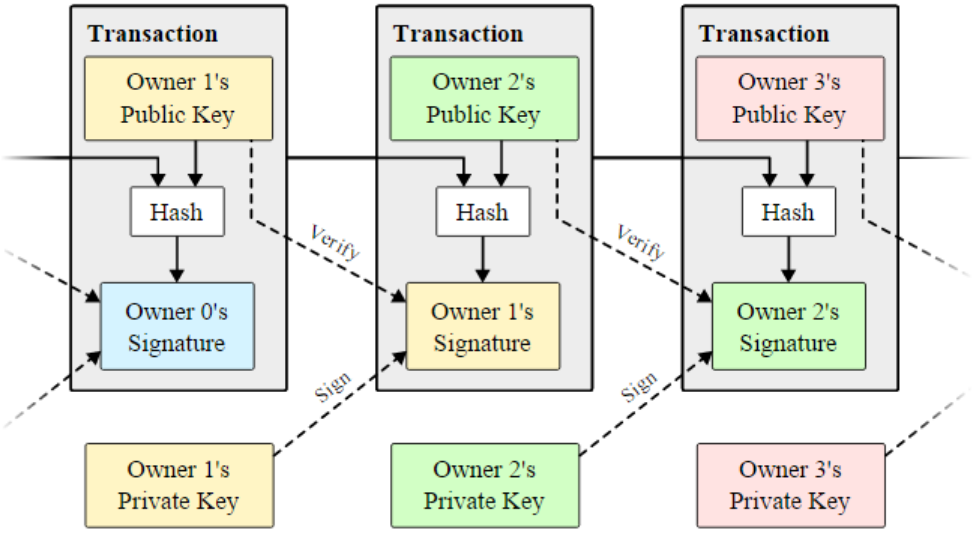
Tudjuk már, hogy a bitcoin pénztárca raktározza a bitcoin címet, amely nyilvántartást vezet az összes tranzakcióról, és így az egyensúlyról. Ez a cím egy 34 karakterből álló hosszú számsor, ami mint nyilvános kulcs néven vált közismertté. Minden cím /nyilvános kulcs 64 (hexában) betűvel és számmal rendelkező „privát kulcs"-nak, azaz titkos kulcsnak felel meg. A két kulcs kapcsolódik egymáshoz, de arra már nincs mód, hogy kitalálják a privát kulcsot a nyilvános kulcsból. Ez azért fontos, mert azt a tranzakciót, amelyet a bitcoin címről adnak ki egy titkos magánkulccsal kell aláírni. Ehhez mind a magánkulcsnak, mind pedig a tranzakció részleteinek a számítógépen vagy okostelefonban lévő bitcoin szoftverbe való helyezése szükséges. Ezzel az információval a program digitális aláírást tesz közzé. A tranzakció megerősíthető az aláírás és a nyilvános kulcs csatlakoztatásával a bitcoin programba. Ez a bitcoin egyik zseniális része: ha az aláírás a nyilvános kulcsnak megfelelő magánkulccsal történt, a program érvényesíteni fogja a tranzakciót anélkül, hogy tudná mi a magánkulcs.

Először a tranzakciót jóvá kell hagyni (megerősítés szükséges), ezt követően kerül majd egy „blokkba" több másik tranzakcióval együtt a tranzakciónk. Ahhoz, hogy megértsük a folyamatot definiáljuk a hash értéket. A hash értéket egy hash függvény [1] állítja elő,

[1] A hash függvények magyarul hasító függvények olyan számítástechnikában használt algoritmusok, amelyekkel bármilyen hosszúságú adatot adott hosszúságra képezhetünk le. Az így kapott véges adatot hash vagy hasító értéknek szokás nevezni.

amely egy komplex matematikai egyenlet, csökkenti a szöveg vagy adatok mennyiségét 64 karakteres karakterláncra. Ezek az algoritmusok az elektronikus aláírás kulcsfontosságú elemei. Nem véletlen tehát, hogy ha minden alkalommal, amikor a hash függvénybe helyezi a tulajdonos az adott adatkészletet, ugyanazt a 64 karakteres karakterláncot kapja. Ezt az egész bekezdést egy hash-re lehet csökkenteni, és ha nem változtatok, nem távolítok el vagy adok hozzá valamit a szöveghez, ugyanazt a hash-et újra és újra elő tudnám állítani. Ez egy nagyon hatékony módja annak, hogy megmondjuk megváltozott-e valami, és hogy a blockchain hogyan tudja megerősíteni, hogy a tranzakciót nem manipulálták.

Térjünk rá most a blokkokra: minden blokk az adatok részeként tartalmazza az előző blokk hash értékét. Ez az, ami egy lánc részévé teszi. Ha az előző blokk csak egy kis részébe is beavatkozunk, az aktuális blokk hash értékének változnia kell (ne felejtsük, hogy a hash függvény bemenetében lévő apró változás megváltoztatja a kimenetet). Tehát ha módosítani akarunk valamit az előző blokkban, akkor is módosítanunk kell a hash-t az aktuális blokkban, mert az, ami jelenleg szerepel már nem helyes. Ezt azonban nehéz megtenni, különösen akkor, ha félúton elérjük az említett blokkot, és azon már valószínű egy új blokk helyezkedik el. Ez teszi a bitcoint gyakorlatilag szabotázsbiztosnak.



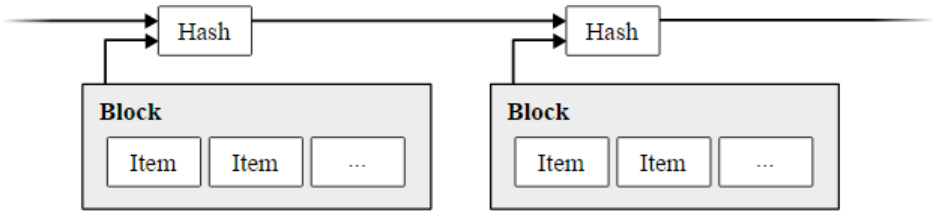
3.1. ábra. A hálózat sematikus felépítése

Úgy juttat át a küldő egy egységét egy másik félnek, hogy ellátja a saját digitális aláírásával az előző tranzakcióról (hash érték), és a következő tulajdonos publikus kulcsával.

Az új tulajdonos azonosítja magát a privát kulcsával, és ezzel igazolja, hogy övé a pénz.

Timestamp and proof of work

Első lépésben létrehozunk egy hash blokkrendszert. Ez annyiban különbözik az eddigiektől, hogy a blokkokat ellátjuk timestamppel azaz időbélyeggel. Azt, hogy az adott timestamp időpontjáig adott blokkban tárolt elemek megtörténtek-e garantálja maga az időbélyeg. Mindegyik timestamp tartalmazza az előző timestamp időpontját is, létrehozva ezzel egy láncot, amelynek tagjait az őket megelőző tagok megléte igazol. Az időbélyegző rendszernek peer to peer [2] alapon kell működnie.
A technikai megoldáshoz, a biztonság juttatásához a proof of work modell alkalmazható, amely egy olyan informatikai validáló rendszer, amelyben ha valamilyen műveletet szeretnénk végrehajtani, annak feltétele valamilyen igényes feladat végrehajtása, számítása. Ha ez bizonyos aszimmetriát követ, akkor azt mondhatjuk, hogy hatékonyan működik a modell.



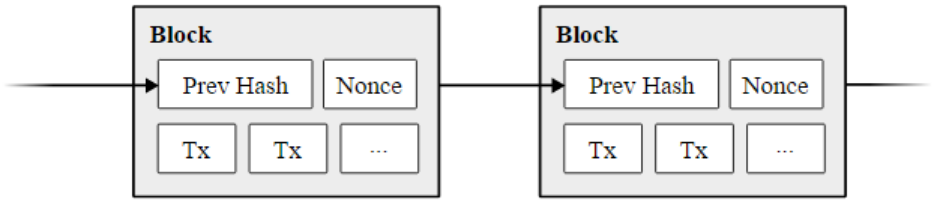
3.2. ábra. Blokk felépítése

Ez a feladat igencsak nehéz, de azt, hogy sikerült-e megoldani már nagyon könnyen ellenőrizhetjük, csak azt kell figyelembe venni, hogy engedélyezi-e a rendszer automatikusan az adott tevékenységet vagy sem.

[2] A peer to peer(P2P) hálózatok olyan együttműködésen (egyenrangú felek között) alapuló rendszerek, amelyek végpontjai közvetlenül kommunikálnak egymással anélkül, hogy valamilyen kitüntetett csomópont szerver közrejátszana.

3.1.1. A séma működése

Vesznek egy hasító függvényt, amellyel hasító értékeket generálnak egészen addig, amíg az értékük egy bizonyos kritériumnak meg nem felel. Bármelyik bemeneti bitjét ha megváltoztatjuk tudjuk, hogy véletlen módon a hasító érték is meg fog változni. A bitcoin rendszerben SHA-256-os hash függvényt alkalmaznak. Minden blokkhoz társítanak egy nonce-t [3], a hálózatban szereplők célja pedig, hogy a blokk megfelelő hash értékét megtalálják. A hasító érték elején található zéró elemek számával lehet szabályozni a probléma



3.3. ábra. Blokklánc felépítése

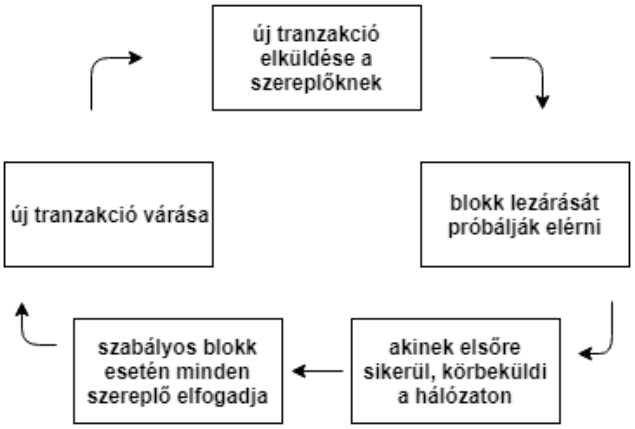
megoldásához szükséges számítástechnikai bonyolultságot. Ez az a kritérium, amit a hasító értéknek teljesítenie kell. Például, ha ilyen 256 bit hosszú hasító függvényt használunk, akkor 2^{256} féle hasító értéket kaphatunk, de ha 10 nullával kezdődőt keresünk, akkor csak 2^{256-10} a számunkra elfogadható. Arra a valószínűség, hogy ezt eltalálják nagyon kicsi $(1/2^{10})$. Védekezni úgy tudunk ez ellen, hogy a blokkokat láncba helyezzük, mert bármely múltbéli blokk megváltoztatásához szükséges az összes olyan blokk problémáját megoldani, ami a jelen és a megváltoztatni kívánt múltbéli blokk között van.

3.1.2. Hálózat működési elve

Ahhoz, hogy a rendszer pénzügyi tranzakciók között tudjon közvetítő lenni, először minden hálózatban szereplőnek meg kell kapnia az új tranzakciókat. Majd ezeket megpróbálják blokkokba rendezni és megoldani a proof of work problémát. Akinek ez először sikerül, blokkját körbeküldi a többi szereplőnek, mint a blockchain legújabb elemét. Ha a blokkban szereplő tranzakcióknak csak egy múltja van, akkor elfogadják a szereplők

[3]kriptográfiában egy adott tetszőleges szám, amit csak egyszer használnak fel

ezt az új blokkot. Elfogadásuk kimutatásaként erre a blokkra építve kezdik el az újabb tranzakciók blokkba helyezését, a folyamat pedig körforgásszerűen következik elölről.



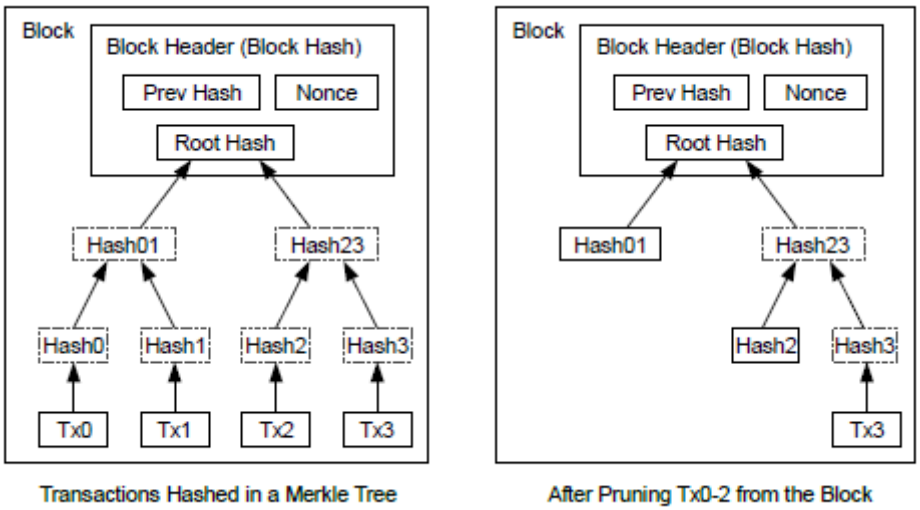
3.4. ábra. Tranzakció elfogadásának folyamata

A csomópontok mindig a leghosszabb láncot tartják megfelelőnek, ennek kiterjesztésén folyamatosan dolgoznak, magát a rendszert ők fogják vezérelni. Amennyiben ez elegendő szereplőnél megtalálható, akkor a rendszer még a nagy dinamikai változásokat is képes eltűrni. Ha két csomópont egyszerre közvetít különböző verziókat a következő blokknak, akkor az elsőként megkapottban fog dolgozni, míg a másikat elmenti. A proof of work megtalálásával az ág meghosszabbodik, így át kell kapcsolnunk a másik már elmentett ágra. Az új tranzakciós közvetítéseknek nem feltétlenül kell minden csomópontot elérniük, nem kell minden szereplőnek reagálnia, részt vennie az adatátvitelben, blokkosításban, emiatt a hálózat stabilitása jelentősen nagy. Ha egy szereplő késve lép a hálózatba, akkor amint belép tudni fogja, hogy hány blokkal van lemaradva.

Egyszerűen ha a legutóbbi tranzakció elég blokk alatt van „eltemetve” , akkor az eddigi tranzakciókat figyelmen kívül hagyhatjuk a lemezterület megmentése érdekében. Ennek megkönnyítése céljából a blokk hash lebontása nélküli tranzakcióit egy Merkle-faként felírhatjuk, ahol csak a gyökér fogja tartalmazni a blokk hash értékét. A régebbi blokkokat ezután össze lehet tömöríteni azzal, hogy a fa ágait megkötjük, a belső hash-ek pedig nem szükséges, hogy tárolva legyenek. Egy tranzakció nélküli blokk fejléce körülbelül 80 bájt lehet. Tegyük fel, hogy 10 percenként generálunk blokkot. Ebben az esetben 80 byte *

6 * 24 * 365, ami 4,2 MB / évnek felel meg. A számítógépes rendszereket 2008-tól kezdődően jellemzően 2 GB rammal értékesítettek. Moore törvénye előrejelzést adott, hogy a jelenlegi növekedés 1,2 GB / év, a tárolás viszont akkor sem jelenthet majd problémát, ha a fejléceket a memóriában kell tárolni.

A bitcoin mennyiségének növelése hasonló folyamat ahhoz, hogy az aranybányászok forgalomba tudják hozni a bányászott aranyat. Ha egy tranzakció kimeneti értéke kisebb, mint a bemeneti értéke, a különbség egy tranzakciós díj, amely hozzáadódik a tranzakciót tartalmazó blokkhoz. Miután előre meghatározott számú „érme" bekerült, a támogató áttérhet a tranzakciós díjakra. Ha egy mohó támadó képes összegyűjteni több CPU



3.5. ábra. Tranzakciók Merkle fában

teljesítményét, mint az összes csomópontot, akkor választania kell aközött, hogy megvédje az embereket a kifizetések visszaszorításával, vagy felhasználásával új érmék létrehozására összpontosít. Aggodalomra adhat okot a hálózat és a tranzakciók számának növekedésével a teljes adatbázis méretállománya. Tudjuk, hogy ezt alulbecsülték, de a teljes adatbázis méretének növekedése sokkal kisebb ütemben fog megtörténni (egységnyi adattárolást a költségéhez viszonyítva), vagyis ilyen jellegű technikai problémák nem fogják gátolni a munkánkat és a rendszer életben tartását, legalább is a közeljövőben nem.

Mekkora valószínűséggel bányászhatunk 1 perc alatt 1 Gh/s-mal egy bitcoint?

A bitcoin hálózat képes autamitikusan beállítani a bányászatnak a nehezségét, ami így átlagosan 10 percet vegyen igénybe. Tegyük fel, hogy H hash következik be 10 percenként, azaz minden hash 1/H eséllyel bányász blokkot. Ha minden percben N hash következik be, akkor a percenként talált blokkok száma binomiális elosztást fog követni. Valószínűség: 1/H. A *Blockchain.info* hálózati 6.294.617 Gh/s-es hash rátáját nézve, egy blokk bányászatának a valószínűsége:

$$\begin{aligned} P(B=1) &= \binom{N}{1}\binom{N}{N-1}\frac{1}{H}\Big(\frac{H-1}{H}\Big)^{N-1} \\ &= \frac{N^2}{H}\Big(\frac{H-1}{H}\Big)^{N-1} \\ &\approx 1,58 \cdot 10^{-8} \end{aligned}$$

Több blokk találása elhanyagolható az esetünkben, ugyanis az egy vagy több blokk megtalálásának valószínűsége gyakorlatilag megegyezik. A jobb átláthatóságért azonban kiszámíthatjuk ezt egy mínusz zéró blokk megtalálásának a valószínűségével.

$$\begin{aligned} P(B>0) &= 1 - P(B=0) \\ &= 1 - \Big(\frac{H-1}{H}\Big)^{N} \\ &\approx 1,58 \cdot 10^{-8} \end{aligned}$$

Minden bányász azért bányászik, hogy profitot termeljen. Ekkor két dolgot kell szem előtt tartania a bányásznak: : figyelnie kell a várható értéket illetve a bitcoinok számát, amelyet az összes lehetséges univerzumon keresztül átlagolnak.

$$\begin{aligned} \frac{1\ Gh/s}{10 \cdot 6.294,617\ h/s} &\approx 0.00000016 \quad \frac{blocks}{minute} \\ &\approx 0.0000004 \quad \frac{BTC}{minute} \\ &\approx 0.000024 \quad \frac{BTC}{hour} \\ &\approx 0.029 \quad \frac{USD}{hour} \end{aligned}$$

3.1.3. A hálózat Stressz tesztelése

Jogosan tehetem fel azt a kérdést, hogy ha a bitcoin piac nem elég hatékony, akkor egy kialakult piaci „sokk" -ra mégis mennyi idő áll rendelkezésemre időben reagálni. A tranzakciós hálózat esetében ez azzal egyenlő, hogy milyen gyorsan terjed egy impulzus magán a hálózaton. Az ilyen esetekben gyakran a biológiához és a biológiában szabadalmaztatott modellekhez nyúlnak vissza. A Susceptible-infected-recovered, vagyis SIR vírusterjedési modell alapján reprezentálom. Három csoportot különítünk el:

- fertőzhető (S)
- fertőzött (I)
- már nem fertőzhető (R)

Ezek segítségével írjuk fel a következő nem lineáris differenciálegyenletet, amelynek megoldásával a különböző csoportok számát tudom becsülni az idő függvényében.

$$\begin{aligned}\frac{dS}{dt} &= -\frac{\beta SI}{N} \\ \frac{dI}{dt} &= \frac{\beta SI}{N} - \gamma I \\ \frac{dR}{dt} &= \gamma I\end{aligned}$$

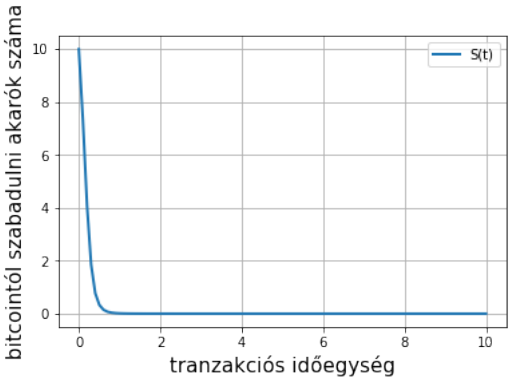
$$S(t) + I(t) + R(t) = N$$

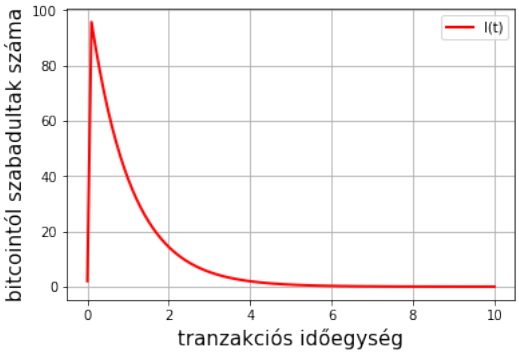
A fertőzés mértékét a β, míg a gyógyulás mértékét a γ paraméterekkel tudom szabályozni. Ezek pozitív számok. A tranzakciós hálózat esetén csak annyi módosítást kell eszközölni, hogy a fertőzés csak a már összekötött csúcsok között történhet, nem lehet az iránya akármilyen. A „sokk" szituáció modellje a következő: valamilyen ponton keresztül valamilyen negatív információ kerül a rendszerbe, emiatt a rendszer szereplői mihamarabb megóvva saját magukat meg akarnak szabadulni bitcoinjaiktól, ezért tranzakciókat kezdenek el végrehajtani a hálózaton.
Ez alapján az S, I és R halmazok új értelmezésbe kerülnek:

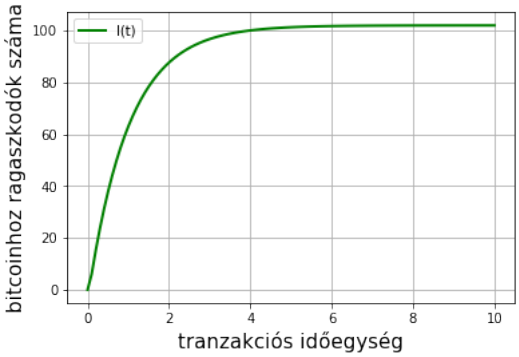
- aki még nem szabadult meg bitcoinjaitól (S)

- aki már tovább küldte, „utalta” azt valakinek, de a címzett is tovább szeretné küldeni (I)
- akik nem akarnak megszabadulni bitcoinaiktól, őket képezzük az (R) halmazba

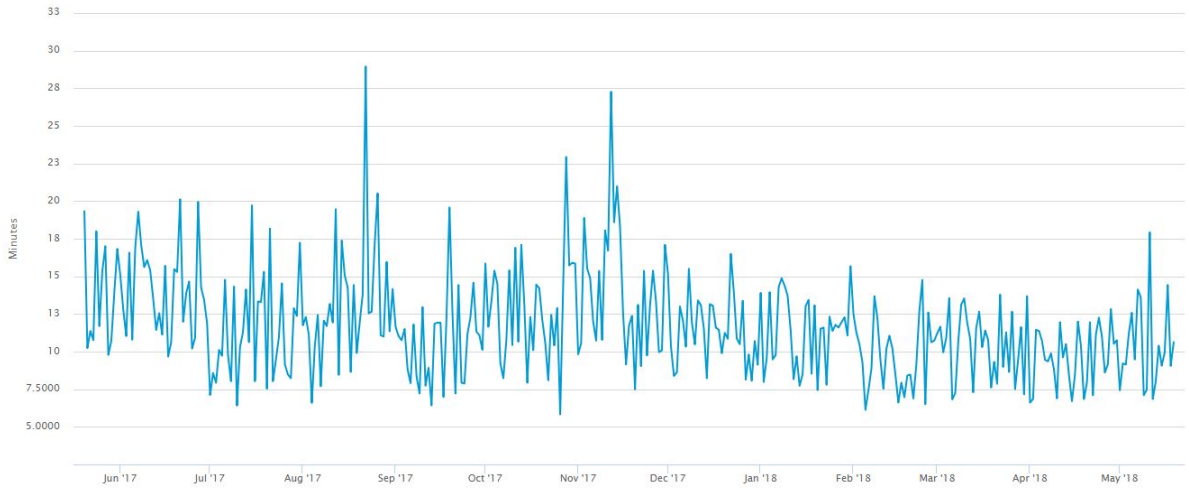
Így a sokk nagyságát β méri, γ pedig, hogy hány tranzakción keresztül jut a bitcoin nyugvó állapotba (tovább nem küldő címzetthez, pl tőzsde számlája). Ezek beállítására mérési lehetőségek vannak megadva a biológián belül ismert fertőző kórokozókra vonatkozóan. Tranzakciós hálózat esetében azonban elméleti megfontolásból nem ilyen egyszerű a helyzet. Olyan széles skálán kell vizsgálni a paramétereket, amilyen széles skálán csak lehet. Tekintsük karakterisztikus időnek a következőt: a rendszerben szereplők 99%-a a megszabadulni akaró bitcoinjaitól meg is szabadult. Adott paraméter mellett (β=γ=1) a szimulációnak S(t), I(t) és R(t) függvénye a következő ábrákon megfigyelhető.
A bal oldali ábrán látható a bitcointól szabadulni akaró szereplők száma az időegység függvényében, mellette akik meg is szabadultak bitcoinjaiktól, szintén a szám az időegység függvényében, alattuk pedig a bitcoinokhoz ragaszkodók száma hasonlóképpen.







A szimulációkból arra következtethetünk, hogy a két paramétert 0.5 és 5 között érdemes megválasztani. Ha ennél kisebb az érték, akkor a legtöbb esetben az impulzus végig sem fut a hálózaton, nagyobb érték esetében pedig olyan gyorsan bekonvergál az 1 időegységbe a folyamat, hogy egészen biztosan nem valós szituációt modellez. Egyes bitcoin tőzsdéken a ki- és beutalás hitelességét a tranzakció megerősítési számával mérik. A megerősítési számot az adott tranzakció blokkjára épülő újabb blokkok számából számolja a rendszer. Egy átlagos bitcoin tőzsde 2-3 megerősítést vár meg mielőtt elismeri, hogy valóban hiteles a ki- és beutalás. Nézzük a megerősítés érkezésének átlagos napi idejét. Ehhez a sokk karakterisztikus idejét nézik először különböző β és γ paraméterek, mint bemeneti értékpárok növekvő sorrendjének függvényében. A megerősítési időt a következő ábra [4] szemlélteti 2017. június 17. és 2018. május 18. között.



3.6. ábra. Megerősítési idő

Ennek a görbének az átlaga 11.57 $\approx$ 12 perc, tehát két megerősítéshez körülbelül 24 percnyi, 3 megerősítésre 36 percnyi időre van szükség. Nagyon kiugró értékeket is észrevehetünk az ábrán, akár 29 perces egyetlen megerősítési időt is. Ennek oka, hogy a beérkezett tranzakciók közül bizonyos tranzakciókat előtérbe helyeznek egy-egy ügylet kárára (nagyobb összegű tranzakciók hamarabb feldolgozásra kerülnek). A hálózaton a karakterisztikus idő egysége ebbe a nagyságrendbe esik, emiatt azt a következtetést vonhatjuk le, hogy egy esemény bekövetkezése után több, mint fél órás nagyságrendű

[4]https://blockchain.info/hu/charts/median-confirmation-time

(átlagot nézve $\approx$ 35 perc) az az idő, ami alatt a hálózat az eseményt le is reagálja. Így nem meglepően nem tudunk ok okozati kapcsolatot felfedezni az árfolyam és a hálózat szerkezete között.

3.1.4. Visszaélések, támadási valószínűség

Nagyon sokan nem ismerik a bitcoinokat, vagy nem szereztek róluk kellő információt, mégis kereskednek vele. Ez az a réteg, akiket első sorban a visszaélések érintenek. A bitcoin visszaéléseket három csoportra bonthatjuk:

- privát kulcs megszerzésére irányuló támadások
- árfolyam, kereskedés manipulálása
- informatikai, digitális visszaélések

Az említett típusok közös tulajdonsága, hogy sokszor emberi hibán alapulnak és nem a hálózatot támadják meg, hanem kihasználják a peremeken található kiskapukat. Egy visszaélést nehéz semmissé tenni. Az ok, hogy a bitcoin hálózatnak erős a védelme, a lopott de valós bitcoinok továbbutalását ugyanúgy védi a rendszer mint a tiszta pénzeket. Ez a védelem azonban mekkora, és mekkora valószínűséggel következik be támadás?

Vizsgáljuk a következő példát: a támadó a becsületes bányásznál hosszabb alternatív láncot akar létrehozni, úgy is felfogható mint egy virtuális pénzhamisítás. A Stressz teszt esetén végig kell gondolni, hogy milyen következményekkel járhat ez. A rendszer struktúrája a támadás után nem változna meg, nem tudnak a támadók új pénzt létrehozni, és a felhasználók sem hamis utalásokat elfogadni. Csak a korábbi tranzakciókat tudja módosítani, hogy „visszavegye" a már elutalt pénzét. A *Binomial Random Walk* (binomiális véletlen bolyongás) segítségével reprezentálható a modell. Sikertelen támadás esetén a bányász blokkllánca nő eggyel, itt +1 a lépés, és ha sikeres a támadás akkor a támadó blokkllánca nő eggyel, a lépés -1, vagyis csökken a különbség a bányász és a támadó között. Annak a valószínűsége, hogy a támadó utolérje a bányászt, vagyis lefaragja hátrányát a *Gambler's Ruin* problémához hasonló elven alapul. E szerint egy szerencsejátékosnak végtelen számú hitel áll a rendelkezésére és egy fix adósságot akar visszanyerni.

Vezessük be a következő jelöléseket:

- p: annak a valószínűsége, hogy a becsületes bányász növeli egy blokkal a blokkláncát
- q: annak a valószínűsége, hogy a támadó növeli egy blokkal a blokkláncát
- Q_z: a sikeres támadás, ahol z jelentse a következőt: a támadónak sikerül megelőznie a z-vel hosszabb blokkláncot

$$Q_z = \begin{cases} 1 & \text{ha } p \leq q \\ \left(\frac{q}{p}\right)^z & \text{ha } p > q \end{cases}$$

A Gambler's Ruin probléma

Ezt a híres problémát először 1651-ben Blaise Pascal és Pierre de Fermat tanulmányozta. A modell egy szerencsejátékos kaszinóban való játékát vizsgálja. Kezdésként i dollárral rendelkezik, majd q valószínűsséggel nyer 1$-t vagy $p = 1 - q$ valószínűséggel veszít 1$-t. A szerencsejátékos célja N összeg megnyerése, azonban ha csődbe megy, akkor kiesik a játékból, hiszen a kaszinónak nem éri meg kockázatot vállalnia a játékosáért, hiszen nincs pénze vesztés esetén fizetni. Legyen q_i annak a valószínűsége, hogy i $-ról indulva a játékos megnyeri a játékot, és jelentse $q_0 = 0$ azt, hogy csődbe fog menni. A q_i értékét kétféleképpen tudjuk meghatározni:

- a játékos 2 egymás utáni tételt nyer: q valószínűséggel nyer az első körben (ekkor $i + 1$-ja lesz) majd a nyert összeggel újra nyer már q_{i+1} valószínűséggel
- a játékos először elveszíti a tételt p valószínűséggel, majd a fennmaradó i-1 $-ral q_{i-1} valószínűséggel nyer

Ez azt jelenti, hogy ez a probléma visszatérési összefüggésként jelenhet meg, ahol a jövő csak a jelenlegi állapottól függ:

$$q_i = (q)q_{i+1} + (p)q_{i-1}$$

mivel $p + q = 1$ azt is tudjuk, hogy:

$$q_i = (p)q_i + (q)q_i$$

Ezt behelyettesítve az egyenlet bal oldalába:

$$\begin{aligned}
(p)q_i + (q)q_i &= (q)q_{i+1} + (p)q_{i-1} \\
(q)q_{i+1} - (q)q_i &= (p)q_i - (p)q_{i-1} \\
(q)(q_{i+1} - q_i) &= (p)(q_i - q_{i-1}) \\
q_{i+1} - q_i &= \left(\frac{p}{q}\right)(q_i - q_{i-1})
\end{aligned}$$

Nézzük meg pár értékre behelyettesítve, majd általánosítsuk:

$$\begin{aligned}
q_2 - q_1 &= \left(\frac{p}{q}\right)(q_1 - q_0) \\
q_3 - q_2 &= \left(\frac{p}{q}\right)(q_2 - q_1) - \left(\frac{p}{q}\right)^2 q_1 \\
&\vdots \\
q_{i+1} - q_i &= \left(\frac{p}{q}\right)^i q_1 \quad (*)
\end{aligned}$$

$q_{i+1} - q_i$-t írjuk fel a következő alakban:

$$\begin{aligned}
q_{i+1} - q_1 &= q_{i+1} + (-q_i + q_i) + (-q_{i-1} + q_{i-1}) + \ldots + (-q_2 + q_2) - q_1 \\
&= (q_{i+1} - q_i) + (q_i - q_{i-1}) + \ldots + (q_3 - q_2) + (q_2 - q_1)
\end{aligned}$$

szummázzuk:

$$= \sum_{k=1}^{n} (q_{k+1} - q_k)$$

(*)-ot alkalmazzuk a szummára:

$$= \sum_{k=1}^{i} \left(\frac{p}{q}\right)^k q_i$$

q_i-t a szumma elé vihetjük:

$$= q_1 \sum_{k=1}^{i} \left(\frac{p}{q}\right)^k$$

A szummát 0-ról indítva a kapott összefüggésünk felírható a következő alakban is:

$$q_{i+1} = q_i \sum_{k=0}^{i} \left(\frac{p}{q}\right)^k$$

Ez egy geometriai sorozat összege, ami a következőképpen írható fel:

$$\sum_{n=0}^{i} a^i = \frac{1-a^{i+1}}{1-a} \quad \text{ahol az a szám, } i \geq 1 \text{ egész}$$

Ezt alapul véve erre az alakra alkítható a szumma:

$$q_{i+1} = \begin{cases} q_1 \frac{1-\left(\frac{p}{q}\right)^{i+1}}{1-\left(\frac{p}{q}\right)} & \text{ha } p \neq q \\ q_1(i+1) & \text{ha } p = q = 0.5 \end{cases} \qquad (**)$$

Tudjuk, hogy ha a szerencsejátékos eléri célját, akkor a $qN = 1$ összefüggést felhasználva $(**)$ megfelelő eseteit átírhatjuk 1-re:

$$q_N = \begin{cases} q_1 \frac{1-\left(\frac{p}{q}\right)^{N}}{1-\left(\frac{p}{q}\right)} & \text{ha } p \neq q \\ q_1 N & \text{ha } p = q = 0.5 \end{cases}$$

Megoldjuk q_1-re minden esetben:

$$q_1 = \begin{cases} \frac{1-\left(\frac{p}{q}\right)}{1-\left(\frac{p}{q}\right)^{N}} & \text{ha } p \neq q \\ \frac{1}{N} & \text{ha } p = q = 0.5 \end{cases} \qquad (***)$$

$(***)$-ot ennek megfelelően helyettesítsük be $(**)$-ba:

$$q_{i+1} = \begin{cases} \frac{1-\left(\frac{p}{q}\right)^{i+1}}{1-\left(\frac{p}{q}\right)^{N}} & \text{ha } p \neq q \\ \frac{i+1}{N} & \text{ha } p = q = 0.5 \end{cases}$$

Most már minden i-re kitudjuk számolni q_i-t:

$$q_i = \begin{cases} \frac{1-\left(\frac{p}{q}\right)^{i}}{1-\left(\frac{p}{q}\right)^{N}} & \text{ha } p \neq q \\ \frac{i}{N} & \text{ha } p = q = 0.5 \end{cases}$$

Tegyük fel, hogy a szerencsejátékos $i = y$ \$-ral kezd, a játéknak pedig hasonlóan két kimenetele lehet: veszít és 0 \$-ral zárja a játékot, vagy pedig győz és nyer $N = y + z$ \$-t. A felvetéseink egyeznek, azaz $q_0 = 0$ és $q_N = 1$. Ekkor:

$$q_l = \begin{cases} \frac{1-\left(\frac{p}{q}\right)^y}{1-\left(\frac{p}{q}\right)^{y+z}} & \text{ha } p \neq q \\ \frac{y}{y+z} & \text{ha } p = q = 0.5 \end{cases}$$

Tekintsük azt az esetet, amikor a szerencsejátékos hajlandó pénzt veszíteni, ezért korlátlan erőforrással rendelkezik, más szóval $y \longrightarrow \infty$.
Abban az esetben ha $p < q$, $\left(\frac{p}{q}\right)^y \longrightarrow 0$, akkor $y \longrightarrow \infty$

$$\lim_{y\to\infty} \frac{1-\left(\frac{p}{q}\right)^y}{1-\left(\frac{p}{q}\right)^{y+z}} = 1$$

$p > q$ esetén pedig:

$$\frac{1-\left(\frac{p}{q}\right)^y}{1-\left(\frac{p}{q}\right)^{y+z}} = \frac{\left(\frac{p}{q}\right)^y\left(\left(\frac{p}{q}\right)^{-y}-1\right)}{\left(\frac{p}{q}\right)^y\left(\left(\frac{p}{q}\right)^{-y}-\left(\frac{p}{q}\right)^z\right)}$$
$$= \frac{\left(\frac{p}{q}\right)^{-y}-1}{\left(\frac{p}{q}\right)^{-y}-\left(\frac{p}{q}\right)^z}$$

ha $p > q$, $\left(\frac{p}{q}\right)^{-y} = \left(\frac{q}{p}\right)^y \longrightarrow 0$ amikor y$\to \infty$, akkor:

$$\lim_{y\to\infty} \frac{\left(\frac{p}{q}\right)^{-y}-1}{\left(\frac{p}{q}\right)^{-y}-\left(\frac{p}{q}\right)^z} = \frac{-1}{-\left(\frac{p}{q}\right)^z} = \left(\frac{q}{p}\right)^z \quad \text{ha } p > q$$

Mivel a támadó korlátlan erőforással rendelkezik a meglévő jelölésünk helyett célszerűbb Q_z-vel jelölnünk a valószínűséget (q_∞ hagyna kivetni valót maga mögött).

$$Q_z = \begin{cases} 1 & \text{ha } p \leq q \\ \left(\frac{q}{p}\right)^z & \text{ha } p > q \end{cases}$$

Azonban ne csak az legyen a cél, hogy a támadó utolérje a becsületes bányászt, hanem előzze is meg! Így Q_z helyett Q_{z+1} értékére vagyunk kíváncsiak. Egy pénzszerzési támadást úgy képzelhetünk el, hogy a támadó a címzettnek bizonyos mennyiségű bitcoin

küldésével egyazon időben már arra törekszik, hogy a tranzakciót semmissé tegye egy alternatív lánc létrehozásával. Ez a sikeres csalás. A címzett mindaddig vár, amíg a tranzakciót nem adják hozzá a blokkhoz, ami után z blokk lesz összekapcsolva. Azt nem tudja, hogy a támadó milyen előrehaladást ért el, viszont ezt a potenciális előrehaladást és a sikeres csalást Poisson eloszlással modellezhetjük:

$$\lambda = z\left(\frac{q}{p}\right)$$

A várt érték megállapításához a Satoshi matematikai modellt használjuk, ami Poisson kísérlet néven vált ismertté. Ahhoz, hogy ilyen modellt használjunk, tegyük fel, hogy:

1. a sikerek száma minden egyes időintervallumban független minden más intervallumtól
2. annak a valószínűsége, hogy egyetlen siker rövid időn belül bekövetkezik arányos az időintervallum időtartamával
3. az ilyen rövid idő alatti egynél több siker bekövetkezésének valószínűsége elhanyagolható
4. továbbá feltételezzük, hogy a kísérlet során a sikerességi valószínűség nem változik, bár a valóságban a bányászok növelhetik vagy csökkenthetik erőforrásaikat

A Poisson kísérletek eredményeinek kihasználásához első feladatunk egy λ érték meghatározása. Ez a sikerek átlagos száma, amit minden intervallum alatt várunk, egy arány: a sikerek / intervallum aránya. Számunkra a sikerek azok azon blokkok száma, amelyeket a támadóktól várunk felfedezni, egy intervallum pedig az az idő, ami alatt a becsületes bányászok z blokkot találnak. Más szóval, a sikerek / intervallum aránya megegyezik a blokkok / intervallum arányával. A bitcoin hálózat úgy van kialakítva, hogy minden $T = 10$ percben 1 blokkot fedeznek fel az aktuális bányászati teljesítmény 100 % -val. A becsületes bányászok minden T percben p blokkot fedezhetnek fel. Ahhoz, hogy z blokk álljon rendelkezésre intervallumra van szükség:

$$z \text{ block} \cdot \frac{T}{p} \frac{\text{minutes}}{\text{block}} = \frac{zT}{p} \text{ minutes}$$

A támadók minden T percben q blokkot fedeznek fel. Ezért az adott intervallum alatt a támadó blokkokat állít elő:

$$\lambda = \left(\frac{2T}{p}\,\frac{\text{minutes}}{\text{interval}}\right) \cdot \frac{q}{T}\,\frac{\text{block}}{\text{minute}} = \frac{zq}{p} = z\frac{q}{p}\,\frac{\text{block}}{\text{interval}}$$

A $\lambda = \frac{zp}{q}$ csak az átlag, a kérdés a tényleges sikereknek a száma. Tegyük fel, hogy X jelöli egy adott kísérletben levő sikerek számát. $X = k$ valószínűsége (azaz ennyi siker jött létre az intervallum alatt, $k \geq 0$):

$$P(X = k; \lambda) = \frac{\lambda^k e^{-\lambda}}{k!}$$

Nézzünk egy példát!

$q = \frac{1}{8}, \quad z = 7$ adottak

Kérdés:
Mi a valószínűsége annak, hogy a támadó $k = 5$ blokkot hoz létre, miközben vár, majd megelőzi a blokkhálózatot?

Először vegyük q-t. Tudjuk, hogy $p = 1 - q$ és $\lambda = z\frac{q}{p} = \frac{z}{7}$. Annak a valószínűsége, hogy a támadó $k = 5$ blokkot termel, mikozben a becsületes bányász 7 blokkot:

$$P(X = 5; \lambda = z/7) = \frac{1^2 e^{-1}}{5!} = \frac{1}{5!e^5} \approx 5,61 \cdot 10^{-5}$$

Majd számoljuk ki a $\left(\frac{q}{p}\right)^{z-k}$, ami $\left(\frac{1}{7}\right)^{7-5}$, azaz $\frac{1}{49}$. A kérdésre a válasz a két kapott eredmény szorzata: $\frac{1}{5!e^5} \cdot \frac{1}{49} \approx 1,145 \cdot 10^{-4}$ %.

A választ egy általánosabb kérdésre keressük: mi annak a valószínűsége, hogy egy adott pontban vagy a jövőben a támadó több blokkot termel, mint a becsületes bányász? Legyen X egy számot reprezentáló véletlen változó, a becsületes bányászok z blokk felfeldezésének ideje alatti támadók által felfedezett blokkok száma. $P(X, \lambda)$-t már ismertettük. Tudjuk, hogy a fennmaradó $z - k$ részből az elkapás valószínűsége $q \cdot (z - k)$. Ezért a felzárkózás teljes valószínûségének megállapításához összegezzük az X összes lehetõségeit:

$$= P(X = 0; \lambda)Q_z + P(X = 1; \lambda)Q_{z-1} + \dots$$
$$= \sum_{k=0}^{\infty} P(X = k; \lambda)Q_{z-k}$$
$$= \sum_{k=0}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!}(Q_{z-k})$$

Valójában amikor $k > z$, akkor a felzárkózás valószínűsége 1. Annak a valószínűsége, hogy a támadó még képes felzárkózni: a Poisson eloszlás szorzata az attól a ponttól való felzárkózás valószínűségével:

$$\sum_{k=0}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} \cdot \begin{cases} \left(\frac{q}{p}\right)^{z-k} & \text{ha } k \leq z \\ 1 & \text{ha } k > z \end{cases}$$

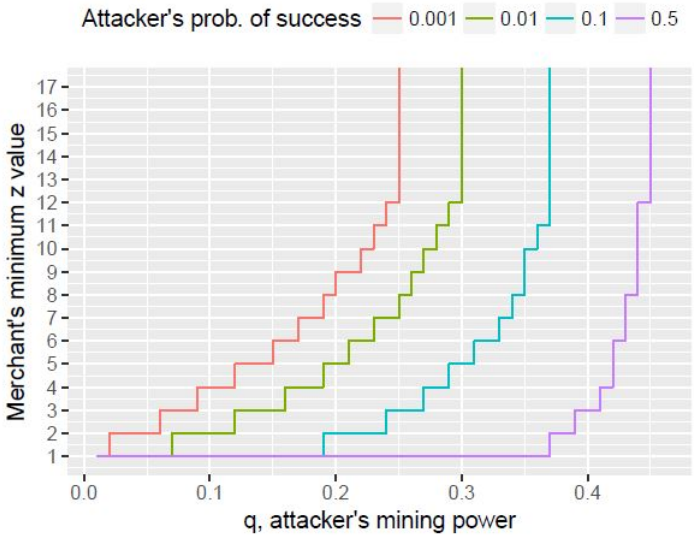
Végül mivel valami bekövetkezésének a valószínűsége ekvivalens azzal, hogy 1 mínusz nem következik be, így a Satoshi féle képletet kapjuk, miután rendeztük az egyenletünket.

$$1 - \sum_{k=0}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} \cdot \begin{cases} 1 - \left(\frac{q}{p}\right)^{z-k} & \text{ha } k \leq z \\ 1 - 1 & \text{ha } k > z \end{cases}$$
$$= 1 - \sum_{k=0}^{z} \frac{\lambda^k e^{-\lambda}}{k!} \cdot \left(1 - \left(\frac{q}{p}\right)^{z-k}\right) + \sum_{k=z+1}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} \cdot (0)$$
$$= 1 - \sum_{k=0}^{z} \frac{\lambda^k e^{-\lambda}}{k!}\left(1 - \left(\frac{q}{p}\right)^{z-k}\right)$$

Újra meg kell oldanunk Nakamoto hibáját. Érdekünk továbra is az, hogy a támadó megelőzze a becsületes bányászt.

$$1 - \sum_{k=0}^{z+1} \frac{\lambda^k e^{-\lambda}}{k!}\left(1 - \left(\frac{q}{p}\right)^{z+1-k}\right) \qquad (1)$$

Tudni szeretnénk a minimális z értéket egy olyan feltétel mellett, hogy a sikeresség valószínűsége alacsony, például csak 1 %. Ahelyett, hogy megadná ezt az értéket, Satoshi biztosítja nekünk apró bites kódokkal, amelyek az összes értéket felsorolják és kiválasztják



3.7. ábra. z minimális értékei (1) alapján

a legkisebbet. A 3.7-es ábra [5] ugyanazt szemlélteti, mint az (1)-sel jelölt képlet. Tekintettel a támadó bányászati teljesítményére és a kívánt sikerességi valószínűségre az ábra a z minimális értéket szemlélteti (1) alapján.

Korlátozások

Előfordulhat, hogy egy korábban ismeretlen bányász (vagy egy létező) új erőforrásokat szentelhet a double spend attack, azaz kettős támadásoknak. A Fischer, Lynch, Paterson (FLP) lehetetlen eredmény azt boncolgatja, hogy Satoshi algoritmusa sohasem juthat konszenzusra, mert bármelyik időpontban a lánc utolsó blokkja csak becslése annak, hogy milyen konszenzus lesz a jövőben. A másik korlátozás, hogy a támadó számítási teljesítménye is meg van szabva. Satoshi talán megpróbálta számszerűsíteni a legrosszabb esetben minden q értékét, de a valóságban ez a legrosszabb eset egyszerűen minden $q > 0.5$ Olyan mintha végtelen pénz állna a rendelkezésre, hogy legyen gáz az autódban, de ezt

[5] A.Pinar Ozisik and Brian Neil Levine:An Explanation of Nakamoto's Analysis of Double-spend Attacks (2017)

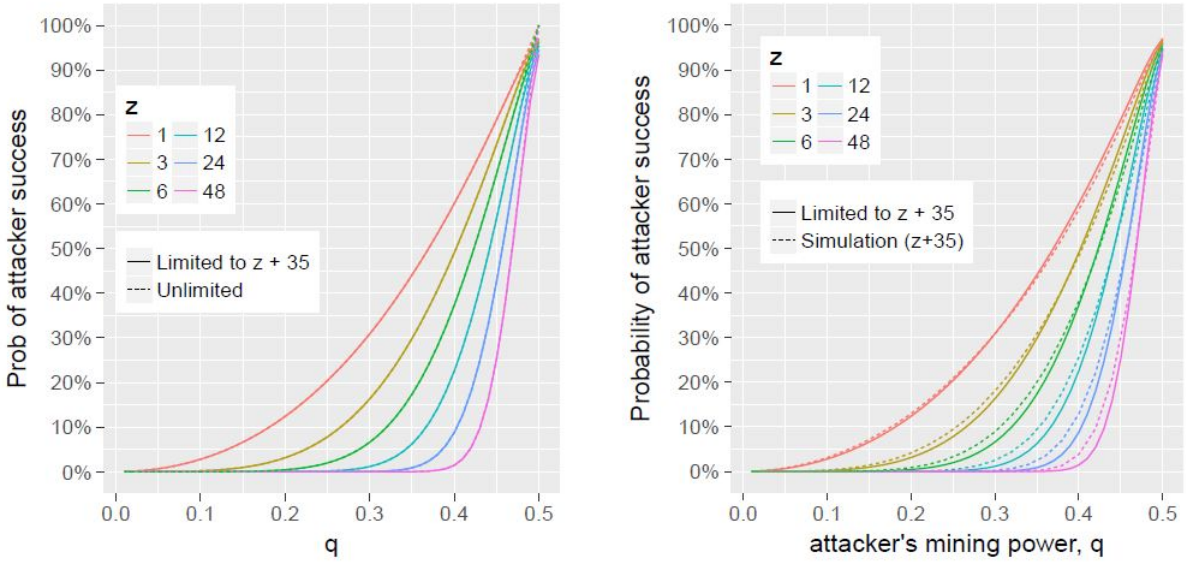
már másik, akár sokkal gyorsabb autóra, amely a birtokodban van nem használhatsz fel.

Érvényesség

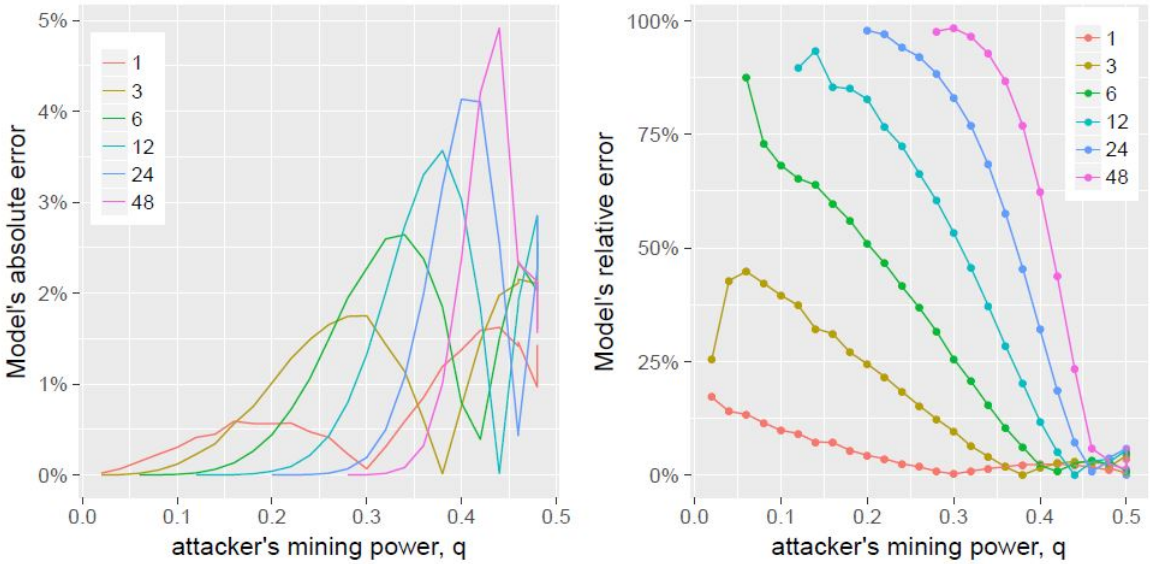
Az érvényességet bizonyítani egy elemzési módszerrel lehet, úgy, hogy szimuláljuk a támadást, valójában a bányászat és a Gambler's Ruin versenyét a támadó és a becsületes bányász között. Sok kísérlet rövid idő alatt lefut *A. Pinar Ozisik* és *Brian Neil Levine* Massachusetts-i egyetem oktatói által is használt Monte Carlo szimulációval. Nem tudják szimulálni a támadó esetét a végtelen erőforrásokkal, de szerencsére az $y = z+1+34$ költségvetése elég közel van a korlátlan esethez. Ez például a 2. ábrán látható. A korlátozott költségvetés modellezéséhez és ábrázolásához a következő egyenletet kell használni:

$$1 - \sum_{k=0}^{z+1} \frac{\lambda^k e^{-\lambda}}{k!} \left(1 - \frac{1 - \left(\frac{p}{q}\right)^{z+35-k}}{1 - \left(\frac{p}{q}\right)^{z+35-k+z+1-k}} \right) \qquad (2)$$

A bal oldali ábrán láthatjuk, hogy a korlátozott költségvetés majdnem ugyanazt az eredményt adja amikor $q < 0.45$, míg mellette a (2) egyenletnek Monte Carlo szimulációval való összehasonlítása látható.



A következő ábrákon a hibákat láthatjuk. A bal oldali mutatja az abszolút hibát: 4% alatti összes z értékre ($z \leq 24$). Mellette a modell viszonylagos hibája látható : a kisebb q értékek esetén 100%. A (2)-es egyenlet hibája lényegében egy szimulációval szemben. A relatív hibák értékei kissé ingadozóak a q alacsony értékeinél, mert a támadó győzelme rendkívül ritka.



A modell abszolút és relatív hibája

A korábban általam számolt példából is sejthettük már, de további szimulációval a simaságot tovább vizsgálva azt a következtetést vonhatjuk le, hogy nem könnyen tud csalni a támadó egy becsületes bányásszal szemben egy sokszereplős hálózaton belül.

4. fejezet

Elliptikus görbék és ECDSA

A digitális fizetőeszköz világában nincs központi szerv, amely felügyelné a bitcoin tranzakciókat, így a visszaélések, támadások elkerülése végett, a titkosítás és megbízhatóság érdekében kidolgozták az Elliptic Curve Digital Signature Algorithm (ECDSA) rendszert. Ez egy Digitális Aláírási Algoritmust (DSA) tartalmaz, amely az elliptikus görbék titkosítását tartalmazza. A bitcoin rendszer az ESDSA-t használja 256 bites domain paraméterek *secp256k1* görbéjének elfogadásával. A megfelelő tartományparaméterek értékeit az alábbi táblázat mutatja, ahol $y^2 = x^3 + 7$ *Weierstrass* kifejezéssel hexadecimális formában vannak feltüntetve.

P az F_q véges mezőt meghatározó paraméter, a,b az egyenletet meghatározó konstansok,

Parameter	**Value**
p	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFC2F
a	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
b	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000007
G	12 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798
n	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF BAAEDCE6 AF48A03B BRD25E8C D0364141
h	01

4.1. táblázat. *secp256k1* paraméterei

G a görbe alappontja, n G-nek a sorrendje (n prím), h pedig kofaktor, $h = \frac{F_q}{n}$ $(h \in Z)$.

4.0.1. Pont operátoros paraméterek és algoritmusok

Geometriailag leírható bináris műveletekkel az elliptikus görbék véges mező fölötti pontjainak keresése. A következőket, hogy tisztán lássuk definiáljuk az elliptikus görbéket és a pont operátoros paraméterekhez és algoritmusokhoz szükséges műveleteket.

1. Definíció. [1] *Legyen K egy olyan test, amelynek a karakterisztikája nem kettő, nem három és legyen* $y^2 = x^3 + ax + b$ *ahol a, b* $\in K$ *olyan harmadfokú polinom, amelynek nincsenek többszörös gyökei.*
Egy K test feletti elliptikus görbe olyan P(x, y) pontok halmaza, ahol az x, y $\in K$ *koordináták kiegyenlítik az*

$$y^2 = x^3 + ax + b$$

egyenletet, valamint hozzá tartozik a görbéhez egy úgynevezett O-val jelölt „végtelen távoli pont".

2. Definíció. *Az elliptikus görbe diszkriminánsán a következő kifejezést értjük:*

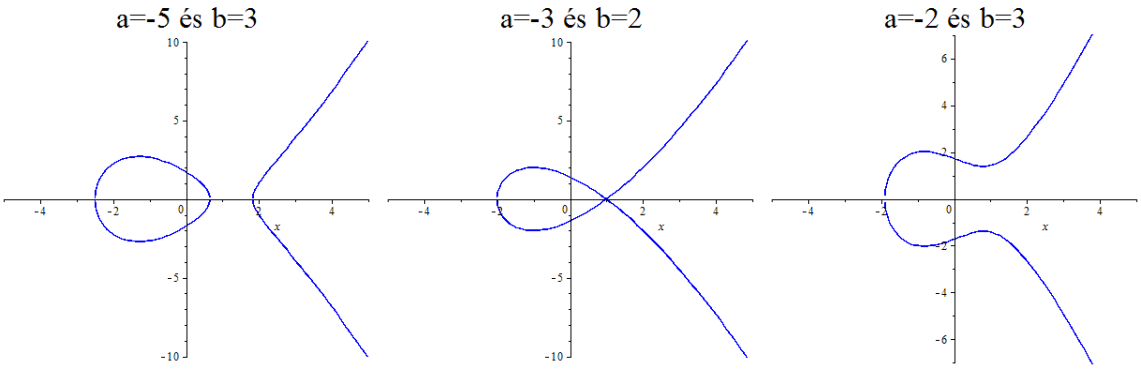
$$D = -16(4a^3 + 27b^2)$$

A diszkrimináns nem nulla, ha az egyenlet jobb oldalának három különböző gyöke van. Abban az esetben, ha a K test a valós számtest, a diszkriminánsnak bizonyos geometriai interpretációját is megadhatjuk.

- Ha $D \neq 0$ akkor az elliptikus görbe nem szinguláris (a görbe génusza 1).
- Ha $D = 0$ és $a = 0$, akkor a görbének szinguláris pontban egy érintője van, a görbe egy csúcsban végződik.
- Ha $D = 0$ és $a \neq 0$ akkor a görbe szinguláris pontját csomópontnak nevezzük, amelybe két különböző érintő húzható. Ebben az esetben a görbe elmetszi magát.

[1] Virasztó Tamás: *Titkosítás és adatrejtés*, NetAcademia Kft (2004)

A fenti három ábrán elliptikus görbét ábrázoltam a diszkrimináns különböző értékei esetén. Kriptográfiában nem foglalkozunk csak azokkal a görbékkel, amelyeknek diszkrimi-



4.1. ábra. Elliptikus görbék különböző a és b paraméterekre

nánsa nem nulla.

4.0.2. Műveletek a pontokkal geometriai, algebrai megközelítéssel

Ellentett

3. Definíció. *Egy P(x, y) pont ellentett párja R, ha a pont x tengelyre tükrözött képe rajta lesz a görbén.* $R = -P$ *azaz* $R(x, -y)$*. Algebrai leírással:*

$$x_R = x_P \quad \text{és} \quad y_R = -y_P$$

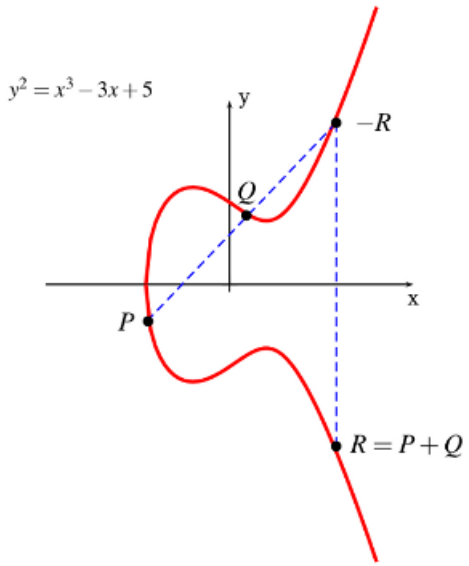
Összeadás

Vegyünk a görbén két különböző pontot P-t és Q-t. Összegüket jelölje R, azaz $P + Q = R$. Ennek folyamata:

1. Kössük össze a P és Q pontot egy egyenessel.

2. A kapott egyenes a görbét metszeni fogja, a metszéspontot jelöljük -R-rel.

3. Ezt a pontot az x tengelyre tükrözve, a kapott kép az előjelváltás szabálya szerint szintén rajta lesz a görbén. Ez lesz a keresett R pont, a P és Q pontok összege.



4.2. ábra. Két különböző pont összege

Algebrai leírással:

$$s = \frac{y_P - y_Q}{x_P - x_Q}$$

$$x_R = s^2 - x_P - x_Q$$

$$y_R = s(x_P - x_R) - y_P$$

Speciális: Pont és ellentett összege

Mennyi P+(-P)?

P és -P egymás tükörképei az x tengelyre nézve, a kettőt összekötő egyenes párhuzamos az y tengellyel, és nem lesz egy újabb pont, ami metszené a görbét. Hasonlóan az R és -R pontok összekötésével kapott egyenes párhuzamos a függőleges tengellyel, és akármelyik irányba hosszabbítjuk meg, nem fogja metszeni a görbénket.

Emiatt az eddigi $P + (-P)$ módszerrel nem számolható ki az összeg. Használjuk fel az

O pontot hozzá, amely a végtelenben van. Definíció szerint a párhuzamos egyenesek a végtelenben metszik egymást, azaz $P + (-P) = O$. Ebből következik, hogy elliptikus görbéken $P + O = P$.

Pont megkétszerezése

Egy pont megkétszerezése hasonló két pont összeadásához, ugyanis ha az előbb említett P és Q pontokat nézve Q-t elég közel visszük P-hez, akkor $P = Q$ lesz. A P és Q ponton keresztül húzott egyenes a P pontba húzott érintővé válik. Innen a korábbi eljárás alapján az érintő metszi valahol a görbét és a metszéspont tükörképe lesz a P pont kétszerese. Kivétel ha a pont pontosan az x tengelyen helyezkedik el, az érintő függőleges, vagyis sehol nem metszi a görbét. Ekkor definíció szerint a pont kétszerese 2E=O. Ezek a pontok, amelyek második koordinátája 3E, 4E, 5E-t kiszámolva megfigyelhető, hogy másképp viselkedik:

$$3E = 2E + E = O + E = E$$

$$4E = 3E + E = E + E = O$$

$$5E = 4E + E = O + E = O$$

Agebrai felírással: ha y_P nem 0 vagyis a megkétszerezendő pont nem az x tengelyen helyezkedik el, akkor:

$$s = \frac{3((x_P)^2 + a)}{2y_P}$$

$$x_R = s^2 - 2x_P$$

$$y_R = s(x_P - x_R) - y_R$$

ahol s a P pontba húzott érintő meredeksége.

4.0.3. Moduláris aritmetika közbelépésével

A moduláris aritmetika a kongruencia aritmetikája, amelynek szabadalmaztatása Friedrich Gauss nevéhez köthető. A moduláris aritmetikai számok egy kör körül keringenek,

több értékre ugyanazt a számot felvéve. Legjobb példa erre az analóg óra. Amikor 15 óra van, valójában a mutató 3-at mutat, vagyis kétszer veszi fel az óra a hármas értéket. Ezt matematikai nyelvre úgy fogalmazhatjuk át, hogy a 3 is és a 15 is egy $p > 1$ számmal osztva ugyanazt a maradékot adja (esetünkben $p = 12$). A műveleteket, beleértve az összeadást, kivonást, szorzást moduláris aritmetikában is elvégezhetjük, először az egész műveletekkel, majd csökkentjük az eredményt modulo p. Görbénk esetén:

$$y^2 \equiv x^3 + ax + b \pmod{p}, \text{ahol p prím}$$

Ha az egyenlet ugyanazt a maradékot adja p-vel osztva, akkor a P (x,y) pont a görbe pontjai közé tartozik.

- $0 \leq x \leq p-1 \quad \text{és} \quad 0 \leq y \leq p-1$
- valamint $4a^3 + 27b^2 \pmod{p} \neq 0$

Itt már véges testek feletti elliptikus görbékről van szó, ahol F_q a véges test és $2q+1$ pontja lehet a görbének (2q db pont és az O). Minden x ponthoz 2 y érték tartozhat. A véges test feletti elliptikus görbe pontjainak számára Hasse tétele jó becslést ad.

1. Tétel (Hasse). *Legyen N az F_q-pontok száma az F_q véges test feletti E elliptikus görbén. Ekkor:*

$$\mid N - (q+1) \mid \leq 2\sqrt{q}$$

Moduláris aritmetikával azért jobb számolni, mert gyors, pontos és egész számokkal dolgozik, nem úgy mint a valós számok esetén. Emellett a moduláris görbének kevesebb pontja van és a számok értelmezése behatárolható, mert a műveletek operandusai és eredménye 0 és $p-1$ közé esik. Összességében a moduláris aritmetika megnöveli a megoldások számát. Az a, b és p konkrét paraméterekre nézzünk egy görbét! Legyen:

$$a = 1 \qquad b = 0 \qquad p = 13$$

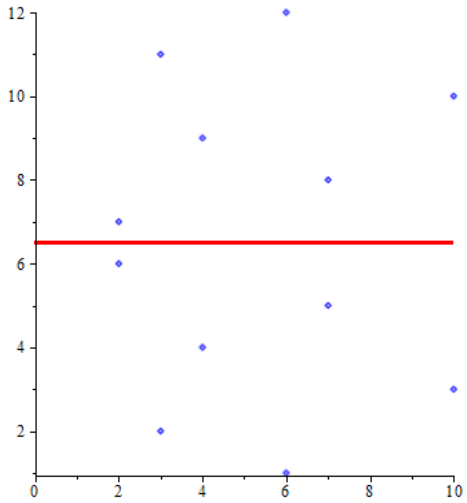
Ekkor a modulus a következőképpen néz ki:

$$y^2 \equiv x^3 + 1x + 0 \pmod{13}$$

Ennek a görbének 13 pontja van, egy az origó (0,0), a további 12 pont pedig

$$(2,6),(2,7),(3,2),(3,11),(4,4),(4,9)$$

$$(6,1),(6,12),(7,5),(7,8),(10,3),(10,10)$$



A pontok az origó kivételével szimmetrikusan helyezkednek el, az $y = 13/2$ azaz az $y = 6.5$ egyenesre. Nézzük meg miben térnek el a pont operátoros műveletek moduláris aritmetikában

Ellentett moduláris aritmetikában

P(x,y) ellentettje $(x, -y \pmod p)$, vagyis $(x, p-y \pmod p)$. Fentebb láthatjuk, hogy az egymással szemben levő y koordináták összege mindig 13. pl: (2,6) és (2,7) esetén 6+7=13, hasonlóan 2+11=13, 4+9=13...
Tehát $R = -P$ pontot a következő formulával adhatjuk meg:

$$x_R = x_P \quad \text{és} \quad y_R = -y_P \pmod p$$

Összeadás moduláris aritmetikában

$P + Q$ esetén az algebrai alakot szimplán modulárisra írjuk át:

$$\begin{aligned} s &= \frac{y_P - y_Q}{x_P - x_Q} \pmod p \\ &= (y_P - y_Q)(x_P - x_Q)^{-1} \pmod p \\ x_R &= s^2 - x_P - x_Q \pmod p \\ y_R &= s(x_P - x_Q) - y_P \pmod p \end{aligned}$$

Pont kétszerese a moduláris aritmetikában

$P + (-P) = O$ továbbra is, 2P esetén pedig szintén ha y_P nem zérus, azaz nem az x tengelyen van ($R = 2P$) akkor:

$$\begin{aligned} s &= \frac{3x_P^2 + a}{2y_P} \pmod p \\ &= (3x_P^2 + a)(2y_P)^{-1} \pmod p \\ x_R &= s^2 - 2x_P \pmod p \\ y_R &= s(x_P - x_R) - y_P \pmod p \end{aligned}$$

Legyen:

$$P(x_1, y_1) \qquad Q(x_2, y_2) \qquad R(x_3, y_3)$$

Ekkor az összeadás:

$$\begin{cases} x_3 = \lambda^2 - x_1 - x_2 \pmod p \\ y_3 = \lambda(x_1 - x_3) - y_1 \pmod p \end{cases}$$

$$\text{ahol } \lambda = \frac{y_2 - y_1}{x_2 - x_1} \pmod p$$

Pont kétszerese pedig:

$$\begin{cases} x_3 = \lambda^2 - 2x_1 \pmod p \\ y_3 = \lambda(x_1 - x_3) - y_1 \pmod p \end{cases}$$

$$\text{ahol } \lambda = \frac{3x_1^2 + a}{2y_1} \pmod p$$

Az összeadás és a pont duplázása mellett a szorzás is, mint művelet alkalmazható a görbén, hiszen ez nem más mint sorozatos összeadás (P, 2P, 2P+P, 3P+P...). Az így képzett $Q = dP$ pontot a pont skalár szorzatának nevezzük, ahol d diszkrét logaritmusa Q-nak p bázis felett.

Algoritmus

Double and Add Algorithm

Require d, P

Ensure $Q = dP$

1. if $d_1 = 1$ then
2. $Q := P$
3. end if
4. for $i = 1$ to n

   $Q := 2Q$

   if $d_i = 1$ then

   $Q := Q + P$

   end if
5. end for
6. Return Q

Nézzük meg egy példán keresztül is! Legyen az elliptikus görbe egyenlete:

$$y^2 \equiv x^3 + 9x + 17 \pmod{23}$$

Mennyi a $Q(4,5)$ pontnak az $P(16,5)$ alapú diszkrét logaritmusa?
Tudjuk, hogy $Q = dP$. Nincs más dolgunk csak a P pontot addig saját magával összeadni, amíg meg nem kapjuk a Q pontot.

Első lépésben nézzük meg $P + P$-t:

$a = 9$

$b = 17$

$p = 23$

$x_P = 16$

$y_P = 5$

---

tudjuk, hogy:

$$\lambda = \frac{3x_P^2 + a}{2y_P} \pmod{p}$$

$$\lambda = \frac{3 \cdot 16^2 + 9}{2 \cdot 5} = \frac{777}{10} = 777 \cdot 10^{-1}$$

$$10^{-1} \equiv j \pmod{23} \ azaz \ 10 \cdot j \equiv 1 \pmod{23}$$

$$j \equiv 16 \pmod{23}$$

$$\lambda = 777 \cdot 16 \equiv 12 \pmod{23}$$

$$x_R = \lambda^2 - 2x_P \pmod{p}$$

$$x_R = 12^2 - 2 \cdot 16 \pmod{23}$$

$$x_R = 144 - 32 = 112 \equiv 20 \pmod{23}$$

$$y_R = \lambda(x_P - x_R) - y_P \pmod{p}$$

$$y_R = 12(16 - 12) - 5 \pmod{23}$$

$$y_R = 43 \equiv 20 \pmod{23}$$

$$2P = (20, 20)$$

Hasonlóan már $2P$-t ismerve $3P$ számolása $2P+P$-vel és így tovább a következőket kapjuk:

$$1P = (16, 5) \qquad 2P = (20, 20) \qquad 3P = (14, 14)$$
$$4P = (19, 20) \qquad 5P = (13, 10) \qquad 6P = (7, 3)$$
$$7P = (8, 7) \qquad 8P = (12, 17) \qquad 9P = (4, 5)$$

Mivel: $Q = dP$ és $(4, 5) = 9P \longrightarrow$ d=9

Nem könnyű azt az egész d számot megtalálni, amelyre $Q = dP$ teljesül. Ezt nevezzük ECDLP-nek (*Elliptic Curve Discrete Logarithm Problem*), amely egy speciális esete az Diszkrét Logaritmus Problémának. Az elliptikus görbe titkosításának biztonsága az ECDLP-re épül. Ha az ECDLP számítógépes akkor az elliptikus görbe titkosítása kriptográfiailag erősnek tekinthető.

Az ECDLP-re épülő rendszerek vagy aláíró (ECDSA) vagy kulcscserélő (ECDH) rendszerek, mert gyors titkosításra a módszer nem használható. A következőkben az aláíró rendszert vizsgáljuk.

4.0.4. ECDSA algoritmus

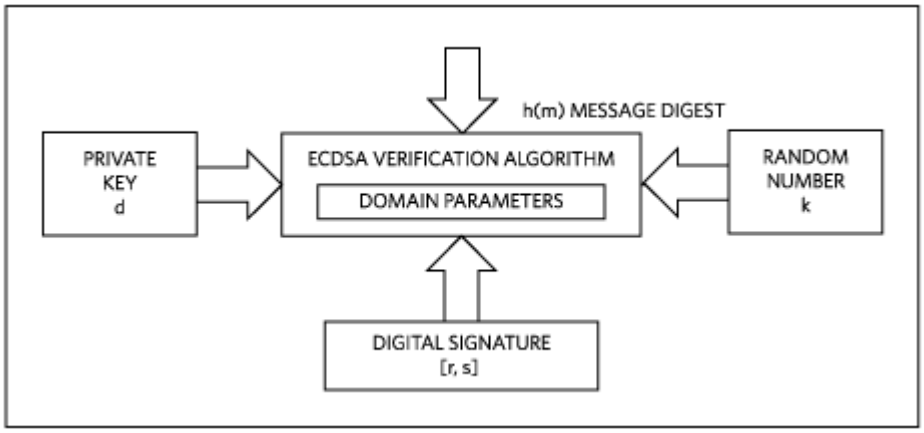
Kulcspár generálása

1. vegyünk egy $P(x_P, y_P)$ pontot a görbén és egy $d \in [1, n-1]$ egészet
2. számoljuk ki a $Q(x_q, y_q) = dP$-t és győződjünk meg róla, hogy szintén a pont a görbén van
3. publikus kulcs (E, P, n, Q) a privát kulcs pedig d

Az aláírás algoritmusa

Vegyük egy m (message) üzenetet, amit alá szeretnénk írni, és egy d privát kulcsot!

1. vegyünk egy véletlen egész $k \in [1, n-1]$ számot
2. számoljuk ki a $(x_1, y_1) = kP$ pontot és az $r = x_1 \pmod{n}$ (ha $r = 0$ kezdjük elölről és válasszunk új k számot
3. számoljuk ki k multiplikatív inverzét [2] n-re, azaz $s = k^{-1}SHA1((e + dr) \pmod{n})$ (ha $s = 0$ akkor 1. lépés.) Itt vehetjük észre, hogy r azért nem lehet nulla, mert akkor az aláírás nem tartalmazná a titkos kulcsot.
4. aláírás (r, s) pár



4.3. ábra. Aláírási algoritmus

Az ellenőrzés algoritmusa

Tegyük fel, hogy Béla rendelkezk Anna nyilvános (E, p, n, Q) kulcsával és az (r, s) aláírási párral (az m üzeneten).

[2] a,b egészek egymás multiplikatív inverzei, ha $ab \equiv 1 \pmod{m}$. A multiplikativitási tétel szerit pedig egy egész a számnak akkor és csak akkor létezik multiplikatív inverze modulo m, ha $gcd(a, m) = 1$, azaz az a és m számok relatív prímek.

A következőképpen ellenőrizheti az aláírást:

1. megnézi, hogy valóban $(r, s) \in [1, n-1]$
2. kiszámolja az üzenet pecsétjét, $e = SHA1(m)$
3. kiszámolja s multiplikatív inverzét n-re: $w = s^{-1} \pmod{n}$ (ezért nem lehetett s=0 korábban, mert akkor nem lenne inverz)
4. kiszámolja $u_1 = ew \pmod{n}$ és $u_2 = rw \pmod{n}$ eredményeket
5. végül kiszámolja a $(x_1, y_1) = u_1P + u_2Q$ elliptikus pontot.
6. Öszehasonlítja r és v értékeket. Az aláírást csak akkor fogadja el, ha $r = v$

Miért helyes az ellenőrzés?

Azt kell belátnunk, hogy az ellenőrzési algoritmus 5. pontjában kiszámolt Q pont nem más, mint a kP pont, azaz $Q = dP$. Tudjuk, hogy:

$$\begin{aligned} s &= k^{-1}(e + dr) \pmod{n} \quad / \cdot s^{-1}k \\ k &\equiv s^{-1}(e + dr) \pmod{n} \\ &\equiv s^{-1}e + s^{-1}dr \pmod{n} \end{aligned}$$

tudjuk, hogy $s^{-1} = w$, ezért

$$\equiv we + wdr \pmod{n}$$

mivel $we = u_1$ és $wr = u_2$

$$\equiv u_1 + u_2 d \pmod{n}$$

az algoritmus 5. pontját vegyük alapul,vagyis

$$(x_1, y_1) = u_1P + u_2Q$$

a kulcs generálásnál megadtuk, hogy $Q = dP$

$$\begin{aligned} &= u_1P + u_2dP \quad \text{/P-t kiemelve} \\ &= (u_1 + u_2d)P \\ &= kP \end{aligned}$$

Példa

Legyen az elliptikus görbénk egyenlete:

$$y^2 = x^3 + 4x + 7 \pmod{11}$$

és legyenek adottak a következő paraméterek:

$a = 4$

$b = 7$

$n = 11$

$q = 11$

$d = 2$

$P = (5, 2)$

ebből $Q = dP$ miatt $Q = (10, 4) \pmod{11}$

Legyen az üzenet hashelt értéke 22 és legyen $k = 3$. Ekkor a titkos ($d = 2$) kulcsunk mellett a publikus kulcs a következőképp áll elő:

$$(\underbrace{p, a, b}_{E}, P, q, Q) = (11, 4, 7, (5, 2), 11, (10, 4))$$

$$SHA(e) = 22 \qquad\qquad k = 3$$

tudjuk, hogy $Q = kP = (15, 6) \equiv (4, 4) \pmod{11}$

$$s \equiv (22 + (2 \cdot 4)) \cdot 4 \pmod{11}$$

ahol a 2. 4-es szorzó 3-nak a multiplikatív inverze 11-re nézve

$$s \equiv 10 \pmod{11}$$

$$w = 10 \pmod{11}$$

$$u_1 = we = 10 \cdot 22 \equiv 0 \pmod{11}$$

$$u_2 = wr = 10 \cdot 4 \equiv 7 \pmod{11}$$

$(x_1, y_1) = u_1 p + u_2 Q = 0 \cdot (5, 2) + 7 \cdot (10, 4) = (70, 28) \equiv (4, 6) = \pmod{11} \longrightarrow \mathrm{x_1} = 4$

És mivel a $r \equiv x_1$ azaz $4 \equiv 4$ a számolásunk helyes volt.

Miért az ECDSA?

Az aláírási sémának a biztonsága a kulcsának méretétől függ, vagyis a biztonság növelhető a k kulcs hosszának növelésével. Különböző sémákat különböző algoritmusokkal törhetünk fel, jelenleg azonban nem létezik olyan algoritmus, amely polinomiális időben törné az ECDSA-t. Az RSA kulcs törése például nagy számú tényezőt igényel, míg az Elliptic Curce Cryptography-n (ECC) alapuló ECDSA kulcs törése megköveteli az Elliptic Curve Discrete Logarithm Problem (ECDLP) megoldását. Ez azt jelenti, hogy az ECDSA-val ugyanolyan szintű biztonságot nyerhetünk, mint az RSA-val, de kisebb kulcsokat használva. A kisebb kulcsok azért célravezetőbbek, mert kisebb tanúsítványokat és adatokat tartalmaznak a TLS kapcsolat létrehozásához. Ez gyorsabb kapcsolatot és gyorsabb betöltési időt jelent a weboldalakon.

Egy 256 bites „elliptic curve key" annyi védelmet nyújt, mint egy 3,248 bites aszimmetrikus kulcs. A webhely-tanúsítványok tipikus RSA kulcsai 2048 bitesek, és ha összehasonlítjuk a kiszolgálón a 256 bites ECDSA kulcsok TLS kézfogásának azon részét, amely kriptográfiailag sokkal gyengébb a 2048 bites RSA kulcsokkal szemben, a következőket kapjuk:

	sign/s
256 bit ecdsa (nistp256)	9516.8
rsa 2048 bits	1001.8
(openssl 1.0.2 beta on $x86_64 with enable - ec\_nistp\_64\_gcc\_128$)	

4.2. táblázat. Aláírások száma ECDSA és RSA esetén

Ez a táblázat mutatja az ECDSA és RSA aláírások számát másodpercenként. A szervereinken az ECDSA tanúsítvány használatával a magánkulcsművelet költsége 9,5-szörös értékkel csökkenti a CPU ciklusokat.

5. fejezet

Összegzés

A dolgozat során betekintettem egy kicsit mélyebben a bitcoin világába, megértettem a felépítését, és azt mennyi minden kell ahhoz, hogy egy bitcoint szabadon átküldhessen egy felhasználó egy másik félnek. A szerkezeti, működési ismeretek és a támadási valószínűség matematikai háttere ugyancsak hozzájárul ahhoz, hogy jártasabb embernek mondjam magam a témát illetően. Nemcsak a valószínűségszámítási és statisztikai ismereteim, hanem a kriptovalután belüli algebra, számelmélet is geometria tudásom is hasznosult. Ahogy a bevezetésben is megfogalmaztam, szerettem volna egy olyan témát választani, aminek később is hasznát vehetem. Egyrészt okosabban tudok majd én is a digitális pénzzel kereskedni, másfelől minden 2. embertől azt hallhatjuk, hogy miért nem vásárolt bitcoint évekkel ezelőtt, amikor alacsony volt az ára, mostanra kamatozódott volna befektetése. Azonban, mint ahogy a tőzsde pillanatról pillanatra való ingadozását, úgy a bitcoin népszerűségét és árát sem lehetett előre feltérképezni. A jövőben azonban különböző modelleket elmélyülten megismerve egy kisebb előrejelzést szeretnék készíteni a digitális fizetőeszköz jövőbeli árára.

Irodalomjegyzék

[1] Noelle Acheson: *What is bitcoin?* (2018 január)

[2] Noelle Acheson: *Why use bitcoin?* (2018 január)

[3] *How do bitcoin transaction work?*, tudományos cikk (2018 január) //www.coindesk.com/information/how-do-bitcoin-transactions-work/

[4] Satoshi Nakamoto: *Bitcoin: Peer to peer Electronic Cash System* (2008)

[5] Virasztó Tamás: *Titkosítás és adatrejtés*, NetAcademia Kft (2004)

[6] Liptai Kálmán: *Kriptográfia* (2011)

[7] Di Wang: *Secure Implementation of ECDSA Signatures is Bitcoin* (2014)

[8] *Bitcoin charts and grafikons* https://blockchain.info/hu/charts

[9] Andreas M. Antonopoulos: *Mastering Bitcoin* (2014)

[10] Ninck Sulivan: *ECDSA: The digital signature algorithm of a better internet* (2014)

[11] Calder Coalson: *What is the probability of mining a bitcoin over one minute time period using a 1 Gh/s rig?* (2013)

[12] A.Pinar Ozisik and Brian Neil Levine: *An Explanation of Nakamoto's Analysis of Double-spend Attacks* (2017)

[13] The Fundamentals of an ECDSA Authentication System (2014) https://www.maximintegrated.com/en/app-notes/index.mvp/id/5767

[14] Bakó Tamás: *Bitcoin hálózatok elemzése* (2015)

Függelék

Stressz teszthez:

```
import scipy.integrate
import numpy
import matplotlib.pyplot as plt

def SIR_model(N, t, beta, gamma):
S,I,R=N
dS_dt = -beta * S * I
dI_dt = beta * S * I - gamma * I
dR_dt = gamma * I
return([dS_dt, dI_dt, dR_dt])

S0=100
I0=2
R0=0
beta=1
gamma=1
t=numpy.linspace(0, 10, 100)
solution= scipy.integrate.odeint(SIR_model, [S0, I0, R0], t, args = (beta, gamma) )
solution=numpy.array(solution)
plt.figure(figsize=[6, 4])
plt.plot(t, solution[:, 0], linewidth=2, label="S(t)")

plt.grid()
plt.legend() plt.xlabel("tranzakciós időegység", fontsize=15)
plt.ylabel("bitcointól szabadulni akarók száma", fontsize=15)
plt.show()
plt.figure(figsize=[6, 4])
plt.plot(t, solution[:, 1], color="red", linewidth=2, label="I(t)")
```

```
plt.grid()
plt.legend()
plt.xlabel("tranzakciós időegység", fontsize=15)
plt.ylabel("bitcointól szabadultak száma", fontsize=15)
plt.show()
plt.figure(figsize=[6, 4])
plt.plot(t, solution[:, 2], color="green", linewidth=2, label="I(t)")
plt.grid()
plt.legend()
plt.xlabel("tranzakciós időegység", fontsize=15)
plt.ylabel("bitcoinhoz ragaszkodók száma", fontsize=15)
plt.show()
```

görbék:

```
plot([sqrt(x^3 - 3*x + 2), -sqrt(x^3 - 3*x + 2)], x = -4 .. 4, y = -10 .. 10, color = blue);
plot([sqrt(x^3 - 5*x + 2), -sqrt(x^3 - 5*x + 2)], x = -4 .. 4, y = -10 .. 10, color = blue);
plot([sqrt(x^3 - 2*x + 3), -sqrt(x^3 - 2*x + 3)], x = -4 .. 4, y = -10 .. 10, color = blue);
line := plot([[0, 6.5], [10, 6.5]], color = red, thickness = 3);
line;
pontok := proc (a, b, c, d, e, f, g, h, i, j, k, l, m) ⟶ plot([a, b, c, d, e, f, g, h, i, j, k, l, m], style =
point, symbol = diamond, scaling = constrained, thickness = 4, color = blue);
pontok([2, 6], [2, 7], [3, 2], [3, 11], [4, 4], [4, 9], [6, 1], [6, 12], [7, 5], [7, 8], [10, 3], [10, 10], [0, 0]);
egybe := pontok([2, 6], [2, 7], [3, 2], [3, 11], [4, 4], [4, 9], [6, 1], [6, 12], [7, 5], [7, 8], [10, 3], [10, 10], [0, 0]);
display(egybe, line);
```