1 Bevezető

Ahhoz hogy megértsük valójában hogyan működnek a blokkláncok először néhány matematikai alapfogalmat kell tisztáznunk. Nyilván mindenki számára jól ismert tény hogy az internet egy megbízhatatlan médium, de ennek ellenére szeretnénk egymással kommunikálni az interneten keresztül. Mivel a blokkláncok is az interneten valósulnak meg, hasonló problémákat kell megoldani a tervezésük során, mint amikkel találkozik az emberiség mióta létezik internet és azt arra szeretné használni hogy a világ két különböző pontján lévő embereket összekösse, azoknak biztonságos kommunikációs csatornát biztosítson. Ez a felállás viszonylag régóta ismert. A legnagyobb lendületet a két világháború idején kapta ez a tudományág (kriptográfia), amikor is kulcsfontosságú volt hogy a csapatok úgy tudjanak egymással kommunikálni hogy az ellenség ne tudja őket lehallgatni. Vagyis a lehallgatást nem tudták teljesen elkerülni, de a csatornán átmenő üzenetek titkosításával lehetséges volt mégis titkos módon kommunikálni egymással. Az egyik legősibb megoldás erre a feladatra a one-time-pad (OTP), amiről biztosan tanultunk más tárgyak keretében, ezért ezt most itt nem részletezzük. További probléma az üzenetek integritása, tehát az hogy hogyan lehet garantálni azt hogy amikor egy üzenet A-ból B-be utazik, akkor ne lehessen módosítani a bitjeit, vagy ha módosítják, akkor azt valamilyen módon tudjuk legalább detektálni. Erre megoldást jelenthet a szimmetrikus kulcsú kriptográfiában a MAC (message authentication code), vagy az aszimmetrikus kulcsú kriptográfiában a digitális aláírás, amikor is az egyik fél az üzenet mellé egy olyan adatmezőt mellékel, amit csak ő a saját titkának ismeretében képes generálni, a fogadó viszont a küldő publikus kulcsa segítségével le tudja ellenőrizni az adatmező (digitális aláírás) hitelességét, és az aláírás csak az adott üzenetre érvényes. Ha az üzenet bitjeiben változás történt, akkor az aláírás már nem lesz hiteles, így pedig tudunk integritást garantálni. A blokkláncok világában hasonló a felállás. Ahogyan majd később látni fogjuk a blokklánc nem más mint egy elosztott adatbázis, amit a világban szétszórt csomópontok (szerverek) adminisztrálnak és valamilyen konszenzus mechanizmus segítségével eldöntik hogy milyen szabályok szerint kerülnek rekordok hozzáadásra az adatbázishoz. Ahhoz hogy ebből kriptovaluta legyen, még kellenek felhasználók, akiknek digitális pénzeket osztunk ki (arról hogy hogyan majd később) és ők ezeket digitális aláírások generálása mellett át tudják adni más felhasználóknak. A kriptovaluta tulajdonjoga és a kriptográfiai titok ismerete között tehát egyenlőséget lehet tenni. Ha elmondtam valakinek a titkos kulcsomat egy kriptovaluta hálózatban (bitcoin, ethereum, stb.) az ugyan olyan mintha aláírtam volna digitálisan egy üzenetet ami azt mondja hogy akkor most az én kriptovalutáim vándoroljanak át egy B felhasználóhoz. Ebből a rövid bevezetőből érezhetjük tehát hogy a digitális aláírások és azok matematikai hátterének megértése elengedhetetlen a blokkláncok megértéséhez.

2 Csoportok, testek, Galois-testek

A digitális aláírásokat szigorú szabályok (matematikai egyenletek) szerint számoljuk. Ezek a matematikai egyenletek bizonyos algebrai struktúrákban működnek. Ezekről lesz most szó. A prímtényezős felbontás talán az egyik legősibb probléma amivel a számelmélészek foglalkoznak. Emellett kriptográfiai szempontból fontos még a diszkrét logaritmus probléma is. Ezek megértéséhez be kell vezetnünk a csoport, a félcsoport, az egységelemes félcsoport, az Ábel-csoport, a test és a gyűrű fogalmát.

2.1 Félcsoport definíciója

Legyen S egy nem üres halmaz és $\circ$ egy rajta értelmezett binár művelet. Az $(S, \circ)$ algebrai struktúra félcsoport, ha a $\circ$ művelet asszociatív, azaz:

$$\forall a, b, c \in S: \quad (a \circ b) \circ c = a \circ (b \circ c)$$

2.2 Egységelemes félcsoport (monoid) definíciója

Az $(S, \circ)$ félcsoport egységelemes félcsoport (monoid), ha létezik egységelem $e \in S$, amelyre:

$$\forall a \in S: \quad e \circ a = a \circ e = a$$

2.3 Csoport definíciója

Egy $(G, \circ)$ algebrai struktúra csoport, ha teljesülnek az alábbi axiómák:

1. Zártság: $\forall a, b \in G: \quad a \circ b \in G$
2. Asszociativitás: $\forall a, b, c \in G: \quad (a \circ b) \circ c = a \circ (b \circ c)$
3. Egységelem: $\exists e \in G,\ \forall a \in G: \quad e \circ a = a \circ e = a$
4. Inverz elem: $\forall a \in G,\ \exists a^{-1} \in G: \quad a \circ a^{-1} = a^{-1} \circ a = e$

2.4 Ábel-csoport definíciója

Egy $(G, \circ)$ csoport Ábel-csoport (kommutatív csoport), ha a művelet kommutatív:

$$\forall a, b \in G: \quad a \circ b = b \circ a$$

2.5 Gyűrű definíciója

Egy $(R, +, \cdot)$ algebrai struktúra gyűrű, ha:

1. $(R, +)$ Ábel-csoport (additív csoport),
2. $(R, \cdot)$ félcsoport (a szorzás asszociatív),
3. A szorzás disztributív az összeadásra nézve:

$$\forall a, b, c \in R: \quad a \cdot (b + c) = a \cdot b + a \cdot c \quad \text{és} \quad (a + b) \cdot c = a \cdot c + b \cdot c$$

2.6 Test definíciója

Egy $(F, +, \cdot)$ algebrai struktúra test, ha:

1. $(F, +)$ Ábel-csoport (nullelemmel: 0),
2. $(F \setminus \{0\}, \cdot)$ Ábel-csoport (egységelemmel: 1),
3. A szorzás disztributív az összeadásra nézve.

Tehát a test egy olyan kommutatív gyűrű, amelyben minden nemnulla elemnek létezik multiplikatív inverze.

2.7 Galois-test definíciója

Egy Galois-test (véges test) egy olyan test, amelynek véges számú eleme van. Jelölése: $GF(q)$ vagy $\mathbb{F}_q$, ahol $q = p^n$ valamely p prímre és $n \geq 1$ egészre. Fontos tulajdonságok:

- Véges test csak p^n elemszámmal létezhet (ahol p prím).
- Adott $q = p^n$ elemszámra izomorfizmus erejéig egyetlen véges test létezik.
- $GF(p)$ esetén a test elemei: $\{0, 1, 2, \ldots, p-1\}$ és a műveletek modulo p értendők.
- $GF(p^n)$ $(n > 1)$ esetén az elemek egy p feletti n-edfokú irreducibilis polinom gyökeiként konstruálhatók.
- A $GF(q)^*$ multiplikatív csoport ciklikus, tehát létezik primitív elem (generátor) g, amelyre $GF(q)^* = \{g^0, g^1, \ldots, g^{q-2}\}$.

A leggyakrabban használt kriptográfiai algoritmusok véges testek és azok feletti algebrai struktúrákban működnek (pl. $\mathbb{Z}_p$ feletti elliptikus görbe csoportok, vagy $\mathbb{Z}_n$ az RSA esetén). Ezekben minden elemet egyértelműen ki lehet fejezni a primitív elemmel és ezt a tulajdonságot nagyon sokszor ki is fogjuk használni. Sokszor kell továbbá multiplikatív inverzt számolni, ezt pedig a kibővített euklidészi algoritmussal lehet.

3 RSA algoritmus

Gondolom mindenki számára jól ismert titkosítási és digitális aláírás algoritmus. Ezt 1977-ben Ron Rivest, Adi Shamir és Leonard Adleman publikálta [1], ma a 2048 bites RSA kulcsokat érezzük biztonságosnak. Meglehetősen lassú, hiszen hatalmas egészeken kell műveleteket végezni, így nem túl kényelmes. Ahogy a számítógépes architektúrák egyre hatékonyabbak valószínűleg tovább kell növelni a kulcsok bithosszát. Ezen probléma miatt ugyan sok helyen lehet még vele találkozni, de a modernebb rendszerekben már inkább elliptikus görbe alapú kriptográfiát használnak (ECC, elliptic curve cryptography). Az RSA a blokkláncokban nem annyira elterjedt (én még sehol nem találkoztam vele).

3.1 RSA algoritmus leírása, titkosítás és aláírás generálás

Kulcsgenerálás:

1. Válasszunk két nagy, különböző prímszámot: p és q.
2. Számítsuk ki $n = p \cdot q$ és az Euler-féle φ függvényt: $\varphi(n) = (p-1)(q-1)$, amely megadja az n-nél kisebb, n-hez relatív prím pozitív egészek számát.
3. Válasszunk egy e egészet, amelyre $1 < e < \varphi(n)$ és $\gcd(e, \varphi(n)) = 1$. A gyakorlatban e tipikusan egy kicsi, fix érték, leggyakrabban $e = 65537 = 2^{16} + 1$, mivel ennek a bináris alakja csak két 1-es bitet tartalmaz, így a nyilvános kulcsú műveletek (titkosítás, aláírás-ellenőrzés) gyorsak. Túl kicsi e (pl. $e = 3$) használata azonban veszélyes lehet: megfelelő padding hiányában Coppersmith kis-exponens elleni támadása [2] rácsredukció (lattice basis reduction) segítségével visszafejtheti az üzenetet, ha annak egy része ismert, vagy ha ugyanazt az üzenetet többször, gyenge random paddinggel titkosítják.

4. Számítsuk ki $d \equiv e^{-1} \pmod{\varphi(n)}$ (a kibővített euklidészi algoritmussal).

A publikus kulcs: (n, e), a privát kulcs: d (az n modulus nyilvános, ezért nem része a titkos kulcsnak; a p, q prímeket és $\varphi(n)$-t pedig a kulcsgenerálás után meg kell semmisíteni, mert ezekből d kiszámítható).

Titkosítás: Adott $m < n$ üzenet esetén a titkosított szöveg:

$$c = m^e \bmod n$$

Visszafejtés:

$$m = c^d \bmod n$$

Digitális aláírás generálás: A gyakorlatban nem közvetlenül az m üzenetet írjuk alá, hanem annak egy H kriptográfiai hash függvénnyel (pl. SHA-256) képzett hash-értékét. Az aláírás:

$$s = H(m)^d \bmod n$$

A hash használata több szempontból is fontos: egyrészt tetszőleges hosszú üzenetet egy n-nél kisebb, fix méretű értékre képez, másrészt kivédi a „nyers" (textbook) RSA aláírás hamisíthatóságát. Utóbbi az RSA multiplikatív tulajdonságából adódik: ha $s_1 = m_1^d$ és $s_2 = m_2^d$ érvényes aláírások, akkor $s_1 s_2 \bmod n$ érvényes aláírás lenne az $m_1 m_2 \bmod n$ üzenetre, amivel a támadó új, sosem aláírt üzenethez tud aláírást gyártani. A gyakorlatban egy szabványos padding sémát is alkalmaznak (pl. RSA-PSS aláíráshoz, RSA-OAEP titkosításhoz).

Aláírás ellenőrzés: Az ellenőrző fél a publikus kulccsal számol:

$$h' = s^e \bmod n$$

Ha $h' = H(m)$, az aláírás hiteles. A biztonság azon alapul, hogy az RSA-probléma – azaz $c = m^e \bmod n$ ismeretében az e-edik gyök (m) kiszámítása n ismeretlen prímfelbontása mellett – számítástechnikailag nehéz feladat [1]. Fontos árnyalat, hogy az RSA-probléma nehézsége *nem bizonyítottan ekvivalens* a faktorizálással: n prímtényezős felbontása (p és q megtalálása) elegendő az RSA feltöréséhez (mert ekkor $\varphi(n)$ és így d is kiszámítható), de nem ismert, hogy szükséges is lenne – elképzelhető, hogy létezik olyan módszer, amely faktorizálás nélkül oldja meg az e-edik gyök problémát.

4 Diffie-Hellman kulcscsere

1976-ban Whitfield Diffie és Martin E. Hellman [3] elegáns megoldást adtak arra a felállásra amikor van egy megbízhatatlan csatorna és a két oldalán lévő fél szeretnénk úgy egymással kommunikálni hogy egy harmadik fél még ha le is hallgatja a csatornát, akkor sem képes elolvasni az üzeneteket. 1976-ig ez csak előzetesen megosztott titkos kulccsal volt elképzelhető (pl. one-time-pad vagy más szimmetrikus titkosító), amelynek a használata feltételezte hogy a felek találkoztak egymással és kicseréltek valamilyen jellegű titkos információt amivel aztán tudtak kommunikálni. Klasszikus esetben (ahogyan a világháború idején kommunikáltak a németek és a szovjetek) ez valóban így is történt. A felek először fizikailag találkoztak és kicseréltek egy hosszú tekercset, amit miután felhasználtak azonnal meg is semmisítettek, így tudták garantálni az üzenetek visszafejthetetlenségét. A Diffie-Hellman kulcscsere algoritmus az egész problémát megoldja digitális üzenetek formájában, ráadásul úgy hogy a feleknek nem kell találkozni fizikailag. Ez nagyon nagy

szerencse, mert a mai világban teljesen elképzelhetetlen lenne hogy valakivel csak úgy tudok kommunikálni az interneten hogy előtte találkoztunk. A kulcscsere algoritmus feltételezik hogy a két fél rendelkezik egy publikus kulccsal, ami a generátor elem egy titkos hatványra emelt változata. Ebből a titkot nem lehet kiszámolni a DLP probléma miatt. A közös titkot a két fél úgy tudja kiszámolni hogy a másik publikus adatát a saját kulcsára emeli, így pedig a logaritmus azonosságai miatt pontosan ugyan azt az értéket fogják kapni. Ezt az értéket egy olyan harmadik fél aki nem ismeri egyik fél titkát sem nyilván nem fogja tudni kiszámolni. Innentől kezdve pedig a felek használhatnak bármilyen szimmetrikus kulcsú titkosítót (DES [4], AES [5], stb.), hiszen amilyen titokkal titkosították az üzeneteket azzal ki is lehet titkosítani és így megbízhatatlan csatornán keresztül tudnak egymásnak üzeneteket küldeni anélkül hogy előzetesen találkoztak volna. A Diffie-Hellman kulcscsere nagyszerűségét demonstrálja az is hogy az algoritmus a mai napig használatban van. A TLS [6] például az egyik legelterjedtebb protokoll az interneten és ez is egy Diffie-Hellman kulccsere lépéssel kezdődik.

4.1 A protokoll formális leírása

Legyen G egy q prímrendű ciklikus csoport g generátorral (a klasszikus esetben $G = \mathbb{Z}_p^*$ egy nagy p prímre, illetve annak egy q rendű részcsoportja; a modern gyakorlatban G egy elliptikus görbe pontjainak csoportja). A csoport paraméterei – G, g és q – nyilvánosak. A két fél, Alice és Bob, a következő lépéseket végzi:

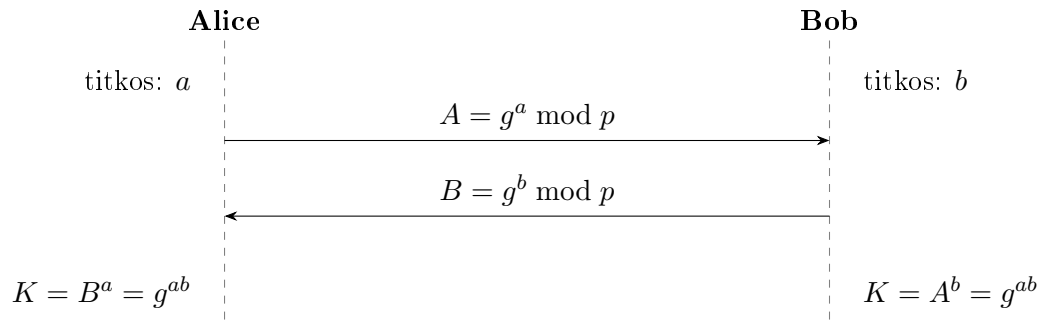
1. Alice választ egy titkos $a \in \{1, \ldots, q-1\}$ értéket, és elküldi Bobnak a publikus értékét:
$$A = g^a \bmod p.$$
2. Bob választ egy titkos $b \in \{1, \ldots, q-1\}$ értéket, és elküldi Alice-nek:
$$B = g^b \bmod p.$$
3. Alice kiszámítja a közös titkot Bob publikus értékéből:
$$K = B^a = (g^b)^a = g^{ab} \bmod p.$$
4. Bob ugyanezt teszi Alice publikus értékével:
$$K = A^b = (g^a)^b = g^{ab} \bmod p.$$

A hatványozás kommutativitása miatt mindketten ugyanazt a $K = g^{ab}$ közös titkot kapják, jóllehet a titkos a, illetve b kitevőt sosem osztották meg egymással.

A csatornát lehallgató támadó csak a nyilvános g, $A = g^a$ és $B = g^b$ értékeket látja. A közös titok kiszámításához a Diffie–Hellman-problémát (CDH) kellene megoldania: g^a és g^b ismeretében g^{ab} meghatározása. A legjobb ismert módszer ehhez a diszkrét logaritmus problémán (DLP) keresztül vezet, azaz $A = g^a$-ból az a kitevő kiszámítása:

$$a = \log_g A \quad \text{(a } G \text{ csoportban)},$$

ami kellően nagy q esetén számítástechnikailag kezelhetetlen. (Megjegyzendő, hogy – a faktorizálás és az RSA-probléma viszonyához hasonlóan – a CDH nem bizonyítottan ekvivalens a DLP-vel: a DLP megoldása elegendő a CDH-hoz, de nem ismert, hogy szükséges is lenne.)



1. ábra: A Diffie–Hellman kulcscsere futása. A csatornán csak A és B halad át; a közös $K = g^{ab}$ titkot mindkét fél a saját titkos kitevőjével számolja ki.

5 Titkosítások feltörése

A legtöbb klasszikus titkosítási algoritmus biztoságát az adja hogy egy meglehetősen nagy véges test fölött működik. Nyilván ha a test összes elemét le tudjuk tárolni a memóriánkban és csinálunk egy asszociatív tömböt amibe az összes privát kulcs-publikus kulcs értéket gyorsan kereshető formában tároljuk, akkor mindenhatók vagyunk és ellenünk nem lehet biztonságos kriptorendszert tervezni. Hasonló a helyzet ha az értékeket nem tároljuk, de nagyon gyorsan ki tudjuk számolni. Igen ám, de ha csak egy 256 bites ECC kulcsot veszünk, akkor a tér mérete 2^{256} nagyságú, ami exponenciálisan sok memóriát vagy számítást (vagy a kettő szorzatát, erről majd később) igényli, ez pedig bonyolultságelméleti szempontból nagyon nehéznek tekintett. Az ilyen jellegű támadásokat hívják brute-force (kimerítő) támadásoknak, amelynek során valaki kipróbálja az összes lehetséges értéket. Nyilvánvaló tehát hogy a kulcs méretét olyan nagyra kell választani hogy a jelenlegi (és lehetőleg a jövőbeli) hardwareken ez lehetetlen legyen. Tovább bonyolítja a helyzetet, hogy a probléma nem univerzálisan redukálódik a kimerítő keresésre, hiszen minden titkosítási algoritmust valamint digitális aláírást külön kell vizsgálni. Gyakran a DLP probléma és a hamisított aláírás generálása például nem ekvivalens. Egy további példa lehet mondjuk az is hogy RSA aláírást hamisítani nem biztos hogy annyira nehéz mint megtalálni egy nagy szám prím osztóit, vagy eldönteni egy számról hogy prím szám-e, vagy hogy a prím osztói nagyok-e. Sok problémáról kiderült hogy nem ugyanabba a bonyolultsági osztályba esnek mint a prímtényezős felbontás vagy a DLP és sokról joggal hihetjük azt hogy ezek egyszerűbb problémák mint az előbb említett kettő. Ez azt jelenti tehát hogy elképzelhető, hogy holnap felkelünk és kiderül, hogy valaki előállt egy algoritmussal ami ügyesen tud ECDSA [7] aláírást hamisítani anélkül, hogy magát az alapul szolgáló elliptikus görbe diszkrét logaritmus problémát (ECDLP) megoldaná – az aláírás-hamisítás és az azt megalapozó matematikai probléma ugyanis nem feltétlenül ekvivalens. Jelenleg nagyon kevés eredmény ismert a digitális aláírások és a titkosítási algoritmusok bonyolultságelméleti hovatartozásáról. További probléma lehet az is, hogy nem minden eredmény publikus. Lehet hogy a mai publikált eredmények alapján azt hisszük, hogy a 2048 vagy a 4096 bites RSA kulcsok biztonságosak, mert a leghatékonyabb publikus algoritmusok ezeket a kulcsméreteket képtelenek kezelni, de mi a helyzet a nem publikált algoritmusokkal? Erre ugye nem tudunk válaszolni, a nemzeti kutatóintézetek (amerikai, kínai, orosz, stb.) nem feltétlenül publikálnak minden eredményt, sőt vannak olyan titkos katonai kapcsolódású intézetek ahol szigorúan tilos publikálni ezeket az eredményeket. Ezen okok miatt jó lenne tehát olyan kriptorendszereket használni, amelyek biztonsága nem függ az alkalmazott kulcsok bithosszától. A kriptográfiának van egyébként ilyen