

Kriptovaluták elméleti hátttere

Ladóczki Bence

docens

Távközlési és Mesterséges Intelligencia Tanszék (TMIT)

Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar (VIK)

ladoczki.bence@vik.bme.hu

2026. október 15.

1 Monero

A Monero (XMR) egy 2014-ben indult kriptovaluta [1], amelynek elsődleges célja a tranzakciók teljes anonimitásának biztosítása. Míg a Bitcoin tranzakciói pseudoanonim jellegűek (a címek nyilvánosak, és láncellenőrzéssel gyakran összekapcsolhatók valós személyekkel), a Monero három kriptográfiai mechanizmust alkalmaz az anonimitás garantálására.

1.1 Ring signatures (gyűrű-aláírások)

A Monero a CryptoNote protokollra [2] épül, amely gyűrű-aláírásokat (ring signatures) [3] használ a küldő elrejtésére. Egy tranzakció aláírásakor a valódi küldő kulcsa összekeveredik több másik („csali”) publikus kulccsal, és a külső megfigyelő nem tudja megállapítani, melyik a tényleges aláíró. Formálisan: egy n tagú gyűrű-aláírás esetén az ellenőrző csak azt tudja megállapítani, hogy a gyűrű n tagja közül valamelyik írta alá – de nem tudja, melyik.

A Rivest–Shamir–Tauman konstrukció [3] eredeti motivációja nem a kriptovaluta volt, hanem az anonim kiszivárogtatás („whistleblowing”): hogyan lehet bizonyítani, hogy egy titkos információ egy meghatározott csoport (pl. a Fehér Ház kabinetje) valamelyik tagjától származik, anélkül hogy felfednénk, melyiktől. A gyűrű-aláírás lehetővé teszi, hogy az aláíró bármely ad-hoc csoportot megnevezzen (nincs szükség előzetes koordinációra, csoportmenedzserre vagy visszavonási mechanizmusra), és hiteles aláírást produkáljon, amely a csoport bármelyik tagjától származhat.

A konstrukció a következőképpen működik. Legyen r a gyűrű mérete, és minden tag P_i rendelkezzen egy RSA kulcspárral (e_i, d_i, n_i) . A konstrukció alapépítőeleme a trapdoor permutáció: olyan egyirányú függvény, amelyet bárki ki tud számítani előre irányban, de invertálni csak a titkos kulcs ismeretében lehet. Erre azért van szükség, mert a gyűrű-egyenlet megoldásához az aláírónak invertálnia kell a saját g_s függvényét – és pontosan ez az a lépés, amelyet a gyűrű többi tagja nem tud végrehajtani. A trapdoor permutáció:

$$f_i(x) = x^{e_i} \bmod n_i \tag{1}$$

Ezt kiterjesztik az egész $\{0, 1\}^b$ tartományra (b bites értékek): ha $m = q_i n_i + r_i$ (ahol $0 \leq r_i < n_i$), akkor

$$g_i(m) = \begin{cases} q_i n_i + f_i(r_i) & \text{ha } (q_i + 1)n_i \leq 2^b \\ m & \text{egyébként} \end{cases} \quad (2)$$

A gyűrű-aláírás egy kombinációs függvényt használ, amely láncolja a tagok értékeit egy szimmetrikus titkosítón E_k keresztül:

$$C_{k,v}(y_1, y_2, \dots, y_r) = E_k(y_r \oplus E_k(y_{r-1} \oplus \dots \oplus E_k(y_1 \oplus v) \dots)) \quad (3)$$

ahol $k = h(m)$ (az üzenet hash-e) és v egy véletlen „ragasztó” érték. Az aláírás érvényes, ha a gyűrű-egyenlet teljesül:

$$C_{k,v}(y_1, y_2, \dots, y_r) = v \quad (4)$$

Az aláírás generálása (s -edik tag az aláíró): az aláíró véletlenszerűen választ x_i értékeket minden $i \neq s$ -re, kiszámítja $y_i = g_i(x_i)$ -t (ezt bárki megteheti, mert e_i publikus), majd a gyűrű-egyenletből ($C_{k,v} = v$) meghatározza, milyen y_s értékre van szüksége. Ez azért lehetséges, mert E_k egy invertálható szimmetrikus titkosító: az aláíró a láncot y_1 -től y_{s-1} -ig előre irányban kiszámítja (egymás után alkalmazva E_k -t), majd y_r -tól y_{s+1} -ig hátra irányban (E_k^{-1} -gyel). A két köztes értékből y_s egyértelműen adódik egy XOR és egy E_k^{-1} alkalmazásával. Végül a saját trapdoor-jával invertál:

$$x_s = g_s^{-1}(y_s) = y_s^{d_s} \bmod n_s \quad (5)$$

Ezt kizárólag az s -edik tag tudja kiszámolni, mert ehhez a titkos kulcs d_s ismerete szükséges (az RSA probléma nehézsége garantálja, hogy d_s nélkül g_s nem invertálható). A g_s^{-1} tehát a trapdoor: ez teszi lehetővé, hogy pontosan egy tag – aki ismeri a saját titkos kulcsát – be tudja zárni a gyűrűt. A többi tag titkos kulcsát nem kell ismerni, csak a publikus kulcsait. A végső aláírás:

$$\sigma = (P_1, P_2, \dots, P_r; v; x_1, x_2, \dots, x_r) \quad (6)$$

Az ellenőrző kiszámítja $y_i = g_i(x_i)$ minden i -re, majd ellenőrzi, hogy $C_{k,v}(y_1, \dots, y_r) = v$. Ha igen, az aláírás érvényes – de nem tudja, melyik tag használta a trapdoor-t.

Az anonimitás kulcsa a következő: az ellenőrző szempontjából minden x_i egyenrangú. Nem tudja megkülönböztetni, melyik x_i -t választotta véletlenszerűen az aláíró (és számolta „előre” g_i -vel), és melyiket határozta meg a gyűrű-egyenletből (és számolta „hátra” g_s^{-1} -gyel). Mivel a g_i trapdoor permutáció, a kimenet $y_i = g_i(x_i)$ mindkét esetben egyenletesen oszlik el – tehát a véletlenszerűen választott és a trapdoor-ral invertált értékek statisztikailag megkülönböztethetetlenek. Ez az ún. feltétel nélküli aláíró-ambiguitás (unconditional signer-ambiguity): még korlátlan számítási kapacitással sem lehet meghatározni, ki az aláíró.

1.1.1 Gyakorlati implementáció a Monero-ban

A Monero nem az RSA-alapú Rivest–Shamir–Tauman konstrukciót használja, hanem egy elliptikus görbékre adaptált változatot. A kulcsok Ed25519 görbén vannak: minden felhasználónak van egy $(x, P = xG)$ kulcspárja. A Monero jelenleg a CLSAG (Concise Linkable Spontaneous Anonymous Group) [4] sémát alkalmazza (2021 óta, a korábbi MLSAG séma hatékonyabb utódjaként), amely három fontos tulajdonsággal rendelkezik:

(1) gyűrű-aláírás (az aláíró rejtett), (2) linkable (összekapcsolható a dupla-költés detekciójához), (3) spontaneous (nincs szükség előzetes csoportregisztrációra).

A CLSAG algoritmus a következőképpen működik. Legyen n a gyűrű mérete, $P_i = x_i G$ a tagok publikus kulcsai (ahol G az Ed25519 generátor pont), és s az aláíró indexe. Az aláíró ismeri x_s -t (a saját titkos kulcsát).

Aláírás generálás:

1. Kiszámítja a key image-et: $\tilde{K} = x_s \cdot H_p(P_s)$, ahol H_p egy hash-to-point függvény – olyan leképezés, amely tetszőleges bemenetet (pl. a publikus kulcs bájtjait) egy érvényes elliptikus görbe ponttá alakít úgy, hogy senki nem ismeri az eredmény diszkrét logaritmusát G -hez képest. Erre azért van szükség, mert ha egyszerűen G -t használnánk, a key image $x_s G = P_s$ lenne, ami leleplezné az aláírót. A Monero implementációja: a bemenetet SHA-512-vel hasheli, az eredményt x -koordinátaként értelmezi, és megkeresi a hozzá tartozó görbepontot.
2. Választ egy véletlen α -t és kiszámítja: $c_{s+1} = H(\mathcal{M}, \alpha G, \alpha H_p(P_s))$, ahol $\mathcal{M}$ az üzenet.
3. Minden $i \neq s$ -re választ véletlen r_i -t és kiszámítja a kihívás-láncot:

$$c_{i+1} = H(\mathcal{M}, r_i G + c_i P_i, r_i H_p(P_i) + c_i \tilde{K})$$

Ez a lánc körbejárja a gyűrűt: $c_{s+1} \rightarrow c_{s+2} \rightarrow \dots \rightarrow c_s$.

4. Végül kiszámítja $r_s = \alpha - c_s x_s$, ami „bezárja” a gyűrűt.

Az aláírás: $\sigma = (c_1, r_1, r_2, \dots, r_n, \tilde{K})$.

Ellenőrzés: az ellenőrző újraszámítja a teljes kihívás-láncot c_1 -ből kiindulva az összes r_i és P_i felhasználásával. Ha a lánc visszatér c_1 -hez (azaz konzisztens kört alkot), az aláírás érvényes. Az ellenőrző nem tudja, melyik r_i -t számolta az aláíró $\alpha - c_s x_s$ -ként és melyiket választotta véletlenszerűen – a kihívás-lánc szimmetriája garantálja az anonimitást.

A CLSAG előnye az MLSAG-hoz képest, hogy egyetlen skalár aggregált kulcsot használ több helyett, ami $\approx 25\%$ -kal csökkenti az aláírás méretét és gyorsítja az ellenőrzést.

A dupla-költés megakadályozása a key image mechanizmuson alapul: az aláíró a tranzakcióba helyez egy $I = x \cdot H_p(P)$ értéket (ahol H_p egy pont-hash függvény). Ez az érték determinisztikusan függ a titkos kulcstól, tehát ugyanaz az output nem költhető el kétszer – ha ugyanaz az I kétszer jelenik meg a blokkláncon, az dupla-költést jelez.

A kliens szoftver a következőt teszi tranzakció létrehozásakor: kiválaszt a blokkláncból $n - 1$ darab „csali” (decoy) output-ot egy statisztikai eloszlás szerint (újabb output-ok nagyobb valószínűséggel kerülnek be, mert a régebbiek gyanúsabbak lennének), a valódi küldő output-ját ezek közé keveri, majd a CLSAG algoritmust futtatja a saját titkos kulcsával. A gyűrű mérete jelenleg 16 (azaz 1 valódi + 15 csali).

1.2 A három privacy-réteg kapcsolata

A Monero három különböző kriptográfiai mechanizmust kombinál, amelyek együtt biztosítják a teljes tranzakciós anonimitást. Mindhárom egy-egy konkrét információt rejt el:

- Ring signatures: elrejtik a küldőt (ki fizetett?).
- Stealth addresses: elrejtik a fogadót (kinek fizetett?).

- RingCT: elrejt az összeget (mennyit fizetett?).

Egy blokklánc-elemző tehát egyetlen tranzakcióból sem tudja megállapítani sem a küldőt, sem a fogadót, sem az összeget – szemben a Bitcoinnal, ahol mindhárom nyilvános.

1.3 Stealth addresses (rejtett címek)

Minden tranzakció egyedi, egyszer használatos címet generál a fogadónak. A küldő a fogadó publikus kulcsából és egy véletlen értékből Diffie–Hellman-szerű protokollal állít elő egy egyedi címet, amelyre utal. Csak a fogadó tudja (a saját privát kulcsával) felismerni és elkölteni az oda utalt összeget. Ezáltal a fogadó valódi címe soha nem jelenik meg a blokkláncon.

1.4 RingCT (Ring Confidential Transactions)

2017 óta a Monero a RingCT [5] protokollt alkalmazza, amely Pedersen-commitmentekkel (commitments) rejt el a tranzakciók összegét. A hálózat ellenőrizni tudja, hogy a bemenetek összege megegyezik a kimenetek összegével (nincs pénzteremtés), anélkül hogy bárki megtudná a konkrét összegeket. A Pedersen-commitment: $C = aG + bH$, ahol a az összeg, b a blinding faktor, G és H generátor pontok.

1.5 Range proofs: negatív összegek megakadályozása

A Pedersen-commitment önmagában nem elegendő: a csoportművelet additív, tehát a hálózat ellenőrizni tudja, hogy $\sum C_{\text{bemenet}} - \sum C_{\text{kimenet}} = 0$ (azaz a blinding faktorok is kiegyenlítődnek), de ez nem akadályozza meg a negatív összegek használatát. Például ha valaki -5 XMR-t és $+15$ XMR-t ad kimenetként egy 10 XMR bemenethez, az egyenleg stimmel ($10 = 15 + (-5)$), de a -5 -ös kimenet a modulo aritmetika miatt egy hatalmas pozitív számként jelenne meg – így pénzt gyártottak volna a semmiből.

A megoldás a range proof (tartomány-bizonyíték): a küldőnek minden kimenethez bizonyítania kell, hogy az összeg egy érvényes tartományba esik, azaz $0 \leq a < 2^{64}$, anélkül hogy felfedné a pontos értékét. A Monero jelenleg Bulletproofs+ [6] mechanizmust alkalmaz erre:

- Az összeg bináris felbontását használja: $a = \sum_{j=0}^{63} a_j \cdot 2^j$, ahol minden $a_j \in \{0, 1\}$.
- Minden bithez egy Pedersen-commitmentt készít, és egy zero-knowledge bizonyítással igazolja, hogy az commitment valóban 0-t vagy 1-et tartalmaz.
- A Bulletproofs az egész bizonyítást logaritmikus méretűre tömöríti: 64 bit helyett $\approx 2 \log_2(64) = 12$ csoportelem elegendő.

Így a hálózat meg tud győződni arról, hogy minden kimenet nem-negatív és legfeljebb $2^{64} - 1$, anélkül hogy megtudná a konkrét összegeket. Ez zárja ki a negatív összegekkel való pénzgyártást.

1.5.1 A Bulletproofs protokoll részletei

A Bulletproofs [6] egy interaktív bizonyítási protokoll (amelyet Fiat–Shamir heurisztikával nem-interaktívvá alakítanak). Két szereplője van: a bizonyító (prover, P) – aki a tranzakciót létrehozza és ismeri a titkos v összeget –, és az ellenőrző (verifier, V) – a

hálózat bármely csomópontja, aki ellenőrzi a tranzakciót. A cél: P meggyőzi V-t, hogy a $V = h^\gamma g^v$ commitmentben lévő v a $[0, 2^n - 1]$ tartományba esik, anélkül hogy v -t felfedné.

A protokoll lépései:

1. lépés ($P \rightarrow V$): Commitment-ek küldése.

P felírja v bináris kifejtését: $\mathbf{a}_L \in \{0, 1\}^n$, ahol $\langle \mathbf{a}_L, \mathbf{2}^n \rangle = v$, és $\mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^n$. A bit-feltétel ($a_{L,j} \in \{0, 1\}$) ekvivalens azzal, hogy $\mathbf{a}_L \circ \mathbf{a}_R = \mathbf{0}^n$ (mert $0 \cdot (-1) = 0$ és $1 \cdot 0 = 0$). P választ véletlen α, ρ blinding faktorokat és $\mathbf{s}_L, \mathbf{s}_R$ véletlen vektorokat, majd elküldi V-nek:

$$A = h^\alpha \cdot \mathbf{g}^{\mathbf{a}_L} \cdot \mathbf{h}^{\mathbf{a}_R}, \quad S = h^\rho \cdot \mathbf{g}^{\mathbf{s}_L} \cdot \mathbf{h}^{\mathbf{s}_R} \quad (7)$$

A a bit-vektor commitment-je, S a blinding vektorok commitment-je. V nem tudja kiolvasni belőlük $\mathbf{a}_L$ -t vagy $\mathbf{s}_L$ -t.

2. lépés ($V \rightarrow P$): Challenge küldése.

V véletlenszerűen választ $y, z \in \mathbb{Z}_p$ challenge értékeket és elküldi P-nek. Ezek arra szolgálnak, hogy a három külön feltételt (összeg, bit-jelleg, $\mathbf{a}_R$ definíció) egyetlen belső szorzatra redukálják:

$$\langle \mathbf{a}_L - z \cdot \mathbf{1}^n, \mathbf{y}^n \circ (\mathbf{a}_R + z \cdot \mathbf{1}^n) + z^2 \cdot \mathbf{2}^n \rangle = z^2 \cdot v + \delta(y, z) \quad (8)$$

ahol $\delta(y, z) = (z - z^2) \langle \mathbf{1}^n, \mathbf{y}^n \rangle - z^3 \langle \mathbf{1}^n, \mathbf{2}^n \rangle$ egy V által önállóan kiszámítható korrekciós tag.

3. lépés ($P \rightarrow V$): Polinom commitment-ek küldése.

P definiálja a vektorpolinomokat:

$$l(X) = (\mathbf{a}_L - z \cdot \mathbf{1}^n) + \mathbf{s}_L \cdot X, \quad r(X) = \mathbf{y}^n \circ (\mathbf{a}_R + z \cdot \mathbf{1}^n + \mathbf{s}_R \cdot X) + z^2 \cdot \mathbf{2}^n \quad (9)$$

A belső szorzatuk: $t(X) = \langle l(X), r(X) \rangle = t_0 + t_1 X + t_2 X^2$. P elküldi a t_1 és t_2 együtthatók commitment-jeit:

$$T_1 = g^{t_1} h^{\tau_1}, \quad T_2 = g^{t_2} h^{\tau_2} \quad (10)$$

(ahol τ_1, τ_2 véletlen blinding faktorok). V ezekből nem tudja kiolvasni t_1 -et vagy t_2 -t.

4. lépés ($V \rightarrow P$): Második challenge.

V véletlenszerűen választ $x \in \mathbb{Z}_p$ -t és elküldi P-nek. Ez lesz a kiértékelési pont a polinomokhoz.

5. lépés ($P \rightarrow V$): Kiértékelte értékek küldése.

P kiszámítja és elküldi: $\hat{t} = t(x)$, $\tau_x = \tau_2 x^2 + \tau_1 x + z^2 \gamma$ (aggregált blinding faktor), $\mu = \alpha + \rho x$, valamint $\mathbf{l} = l(x)$ és $\mathbf{r} = r(x)$ vektorokat.

6. lépés (V ellenőrzi): Fő verifikáció.

V ellenőrzi, hogy a kapott $\hat{t}$ konzisztens-e a korábban kapott commitment-ekkel:

$$g^{\hat{t}} h^{\tau_x} \stackrel{?}{=} V^{z^2} \cdot g^{\delta(y,z)} \cdot T_1^x \cdot T_2^{x^2} \quad (11)$$

A bal oldal a P által állított $\hat{t}$ commitment-je. A jobb oldal az, amit $\hat{t}$ -nek lennie kellene ($z^2 v + \delta + t_1 x + t_2 x^2$) az ismert commitment-ekből összerakva. Ha egyeznek, V meggyőződött arról, hogy $t_0 = z^2 v + \delta(y, z)$, ami pontosan a tartomány-feltétel.

7. lépés: Belső szorzat argumentum (logaritmikus tömörítés).

Az $\mathbf{l}$ és $\mathbf{r}$ vektorok közvetlen elküldése $O(n)$ méretű lenne. Ehelyett P és V egy rekurzív protokollt futtatnak: minden lépésben P elküldi az L_j, R_j commitment-eket, V egy x_j challenge-et ad, és a vektorméretet felezi. $\log_2 n$ lépés után V egyetlen multi-exponenciálással ellenőrzi:

$$\mathbf{g}^{a \cdot \mathbf{s}} \cdot \mathbf{h}^{b \cdot \mathbf{s}^{-1}} \cdot u^{a \cdot b} \stackrel{?}{=} P \cdot \prod_{j=1}^{\log_2 n} L_j^{x_j^2} \cdot R_j^{x_j^{-2}} \quad (12)$$

Az aggregált bizonyítás mérete m darab n -bites tartomány-bizonyításra:

$$\underbrace{2(\lceil \log_2 n \rceil + \lceil \log_2 m \rceil) + 4}_{\text{csoportelemek}} + \underbrace{5}_{\mathbb{Z}_p\text{-elemek}} \quad (13)$$

Tehát m bizonyítás összemérete csak $\log(m)$ -mel nő – ez teszi a Bulletproofs-t gyakorlatilag használhatóvá a Monero-ban, ahol minden tranzakció tartalmaz range proof-ot.

1.6 RandomX bányászat

A Monero tudatosan CPU-barát PoW algoritmust alkalmaz (jelenleg RandomX [7]), amely ellenáll az ASIC-optimalizálásnak. A RandomX célja, hogy a bányászatot a lehető legigazságosabbá tegye: a hétköznapi számítógépek versenyképesek maradjanak a speciális hardverekkel szemben, megakadályozva a Bitcoinnál tapasztalt centralizációt.

A RandomX működési elve: minden hash-számításhoz egy véletlenszerű programot generál, amelyet végre kell hajtani. Ez a program a következő CPU-képességeket igényli:

- Egész aritmetika (összeadás, szorzás, osztás, bitműveletek).
- Lebegőpontos aritmetika (IEEE 754 double precision).
- Elágazás-előrejelzés (branch prediction) – a program tartalmaz feltételes ugrásokat.
- L3 cache hozzáférés – a program egy 2 GB-os „dataset”-ből olvas véletlenszerű címekről.
- Szuperskalár pipeline kihasználás – az utasítások függőségi láncja úgy van tervezve, hogy a modern CPU-k párhuzamos végrehajtási egységeit kihasználja.

A RandomX két módban működik:

- Fast mode: 2 GB RAM szükséges a dataset tárolásához. Ez a bányászati mód, amelyet a teljes csomópontok használnak.
- Light mode: 256 MB RAM elegendő, de $\approx 6\times$ lassabb. Ezt az ellenőrző csomópontok használják.

Az ASIC-ellenállóság azon alapul, hogy egy ASIC, amely véletlenszerű programokat hajt végre egész és lebegőpontos aritmetikával, elágazásokkal, és nagy memória-hozzáféréssel, lényegében maga is egy általános célú CPU lenne – tehát nem nyújt jelentős előnyt egy hagyományos processzorhoz képest. A GPU-k sem hatékonyak, mert a RandomX programok erősen szekvenciálisak és sok elágazást tartalmaznak, ami ellentétes a GPU-k SIMD architektúrájával.

A dataset minden 2048 blokkönként ($\approx 2,8$ naponta) újragenerálódik, és a generálás maga is számításigényes (AES, Blake2b hash-ek láncolása), ami megakadályozza, hogy a dataset-et menet közben számolják ki ahelyett, hogy tárolnák.

1.7 Tőzsdei kereskedés és szabályozási problémák

A Monero az anonimitása miatt sajátos helyzetben van a kriptovaluta-tőzsdék világában. A főbb tőzsdék közül jelenleg a következők listázzák az XMR-t: Binance (korlátozásokkal, az EU-ban 2024-ben delistázták), KuCoin, Kraken (egyes régiókban elérhető), Gate.io, Huobi (HTX), MEXC, Bitfinex, Poloniex, TradeOgre és Cake Wallet (decentralizált váltó). A Coinbase soha nem listázta a Monero-t szabályozási aggályok miatt.

A szabályozási problémák a Monero anonimitás-garanciáiból fakadnak:

- KYC/AML megfelelés: a legtöbb joghatóságban a tőzsdéknek ismerniük kell ügyfeleik személyazonosságát (Know Your Customer) és figyelniük kell a pénzmosás-gyanús tranzakciókat (Anti-Money Laundering). A Monero tranzakciók láncelemzése technológiailag nem lehetséges, ami ellentétben áll ezekkel a követelményekkel.
- Travel Rule: a FATF (Financial Action Task Force) előírja, hogy a virtuális eszközök szolgáltatói (VASP-ok) osszák meg a küldő és fogadó adatait egymással. A Monero stealth address-ei miatt ez nem teljesíthető.
- EU MiCA rendelet: a 2024-ben hatályba lépett szabályozás szigorítja az anonim kriptovaluták kezelését, ami több európai tőzsde XMR-delistázásához vezetett.
- Darknet piacok: a Monero a Silk Road utódpiacok (pl. AlphaBay, Dream Market utódok) preferált fizetőeszközevé vált a Bitcoin helyett, ami fokozott hatósági figyelmet von magára.
- IRS jutalom: az amerikai adóhivatal (IRS) 2020-ban 625 000 USD jutalmat tűzött ki a Monero-tranzakciók nyomon követésére képes eszközök fejlesztéséért [8].

A Monero közösség álláspontja szerint az anyagi tranzakciók magánszférája alapvető emberi jog, és az anonimitás technológiája önmagában semleges. A szabályozók ezzel szemben a technológiai lehetetlenséget (a tranzakciók nyomonkövethetetlenségét) rendszerszintű kockázatnak tekintik. Ez a feszültség a kriptovaluták szabályozásának egyik legélesebb nyitott kérdése.

A cryptojacking a PoW bányászat externáliáinak szélsőséges példája: a bányászati költséget (energia, hardver-amortizáció) tudtukon kívül a felhasználókra hárítja, miközben a haszon a támadóé. Ez is alátámasztja azt az érvet, hogy a PoW rendszerek gazdasági ösztönzőstruktúrája visszaélésre csábít.

2 Szabályozás: az EU MiCA rendelet

A Markets in Crypto-Assets (MiCA) rendelet [9] az Európai Unió első átfogó kriptovaluta-szabályozási keretrendszere, amelyet 2023 júniusában fogadtak el és 2024 decemberében lépett teljes hatályba. A MiCA célja, hogy egységes jogi keretrendszert teremtsen a kriptoeszközök kibocsátására, kereskedésére és szolgáltatására az EU egész területén, miközben védi a fogyasztókat és biztosítja a pénzügyi stabilitást.

2.1 A MiCA fő elemei

A rendelet három kategóriába sorolja a kriptoeszközöket: (1) asset-referenced tokens (ART) – amelyek több eszköz vagy valuta kosarához kötöttek (pl. stablecoin-ok); (2) e-money tokens (EMT) – amelyek egyetlen fiat valutához kötöttek; (3) egyéb kriptoeszközök (utility tokenek, kriptovaluták). Minden kategóriára különböző kibocsátási, tartalékezelési és tőkekövetelmények vonatkoznak.

A szolgáltatók (Crypto-Asset Service Providers, CASP-ok) számára a MiCA egységes engedélyezési rendszert vezet be: egyetlen EU-tagállamban szerzett engedéllyel az egész belső piacon működhetnek („passporting”). A CASP-oknak meg kell felelniük a prudenciális követelményeknek, fogyasztóvédelmi szabályoknak, és az AML/KYC előírásoknak.

2.2 Hatás a privacy coinokra

A MiCA 76. cikke [9] előírja, hogy a CASP-oknak azonosítaniuk kell mind a küldőt, mind a fogadót minden tranzakcióban. Az Európai Bankhatóság (EBA) iránymutatásai [10] tovább szigorítják ezt: a tőzsdék nem kezelhetnek olyan kriptoeszközöket, amelyek technológiailag lehetetlenné teszik a tranzakciók nyomkövetését. Ez a gyakorlatban a Monero, Zcash és hasonló privacy coinok delistázásához vezetett az európai tőzsdéken (Binance EU, Kraken EU).

2.3 A Transfer of Funds Regulation (TFR)

A MiCA-val párhuzamosan lépett hatályba a módosított Transfer of Funds Regulation [11], amely a FATF Travel Rule-t implementálja az EU-ban: a kriptoeszköz-átutalásoknál a küldő és fogadó személyazonosságát minden esetben rögzíteni és továbbítani kell a szolgáltatók között, összeghatártól függetlenül. Ez a legszélesebb körű ilyen szabályozás globálisan, és a privacy coinok mellett az unhosted (self-custodial) walletek használatát is jelentősen megnehezíti.

Hivatkozások

- [1] N. van Saberhagen, *CryptoNote v 2.0*, 2013. <https://www.getmonero.org/resources/research-lab/>
- [2] N. van Saberhagen, *CryptoNote v 2.0*, 2013. <https://cryptonote.org/whitepaper.pdf>
- [3] R. L. Rivest, A. Shamir, Y. Tauman, *How to Leak a Secret: Theory and Applications of Ring Signatures*, Lecture Notes in Computer Science, vol. 3895, Springer, pp. 164–186, 2006.
- [4] B. Goodall, A. Noether, *Concise Linkable Ring Signatures and Forgery Against Adversarial Keys*, Monero Research Lab, 2019. <https://eprint.iacr.org/2019/654>
- [5] S. Noether, *Ring Confidential Transactions*, Ledger, vol. 1, pp. 1–18, 2016.
- [6] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, G. Maxwell, *Bulletproofs: Short Proofs for Confidential Transactions and More*, IEEE Symposium on Security and Privacy, pp. 315–334, 2018.
- [7] T. Howard et al., *RandomX: ASIC-resistant Proof-of-Work Algorithm*, 2019. <https://github.com/tevador/RandomX>
- [8] Internal Revenue Service, *Pilot IRS Will Pay Up to \$625,000 for Cryptocurrency Tracing Tools*, IRS Criminal Investigation, Request for Information RFI-CI-20-0006, 2020.
- [9] European Parliament and Council, *Regulation (EU) 2023/1114 on Markets in Crypto-Assets (MiCA)*, Official Journal of the European Union, L 150/52, 2023.
- [10] European Banking Authority, *Guidelines on the Transfer of Funds and Certain Crypto-Assets Regulations*, EBA/GL/2024/09, 2024.
- [11] European Parliament and Council, *Regulation (EU) 2023/1113 on Information Accompanying Transfers of Funds and Certain Crypto-Assets (Recast)*, Official Journal of the European Union, L 150/1, 2023.