# Kriptovaluták elméleti háttere

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. november 3.

## 1 Monero: a küldő és fogadó elrejtése

A Monero (XMR) egy 2014-ben indult kriptovaluta [1], amelynek elsődleges célja a tranzakciók teljes anonimitásának biztosítása. Míg a Bitcoin tranzakciói pszeudoanonim jellegűek (a címek nyilvánosak, és láncelemzéssel 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 felfedenénk, 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 $0 \le r_i < n_i$), akkor

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

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

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

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

$$C_{k,v}(y_1, y_2, \ldots, y_r) = v \tag{4}$$

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:

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

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 kulcsaikat. A végső aláírás:

$$\sigma = (P_1, P_2, \ldots, P_r;\ v;\ x_1, x_2, \ldots, x_r) \tag{6}$$

Az ellenőrző kiszámítja $y_i = g_i(x_i)$ minden $i$-re, majd ellenőrzi, hogy $C_{k,v}(y_1, \ldots, y_r) = v$. 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 $\alpha$-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: $c_{s+1} \to c_{s+2} \to \cdots \to c_s$.

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

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

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: elrejti 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.

# 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