Euler–Fermat-tétel és RSA

Diszkrét modellek alkalmazásai

1. Euler-féle φ-függvény

A Kongruenciák c. jegyzetben láttuk, hogy egy $a \in \mathbb{Z}$ akkor és csak akkor invertálható modulo m, ha a és m relatív prímek, azaz $\text{lnko}(a, m) = 1$. Például $m = 10$-re 1, 3, 7 és 9 invertálható (mert minden más maradék osztható 2-vel vagy 5-tel).

Az invertálható számok egy fontos tulajdonsága, hogy a szorzatuk is invertálható:

1.1. Tétel. *Ha a és b invertálható modulo m, akkor ab is invertálható modulo m.*

Példa. $m = 10$-re $3 \cdot 9 \equiv 7\ (10)$, $3 \cdot 3 \equiv 9\ (10)$, $3 \cdot 7 \equiv 1\ (10)$ stb.

Az előző jegyzetben (Diszkrét logaritmus) ennek már láttuk egy speciális esetét, ahol a modulus prímszám volt, akkor ugyanis minden nemnulla maradék invertálható, és láttuk, hogy két nemnulla maradék szorzata sosem lesz nulla. Ez azonban csak prímszám modulusokra igaz (pl. $4 \cdot 5 \equiv 0\ (10)$), és most már azt is látjuk, hogy az általánosítás úgy lehetséges, ha a "nemnulla" jelzőt lecseréljük "invertálható"-ra.

Ugyancsak megismertük a *teljes maradékrendszer* fogalmát, pl. $\{0, 1, 2, \ldots, m-1\}$ (lásd: Kongruenciák). Most megismerünk egy hasonló fogalmat, de csak invertálható számokra:

1.2. Definíció. Ha T egy teljes maradékrendszer (modulo m), akkor $\{a \in T \mid \text{lnko}(a, m) = 1\}$ egy **redukált maradékrendszer** (modulo m).

Példa. A következők mindegyike redukált maradékrendszer modulo 10:
$\{1, 3, 7, 9\}$, $\{11, 13, 17, 19\}$, $\{1, 53, 77, -21\}$ stb.

A teljes maradékrendszerekhez hasonlóan a redukált maradékrendszerekből is minden m-re végtelen sok van (de általában csak a legegyszerűbbel foglalkozunk, pl. $\{1, 3, 7, 9\}$). Továbbá bármely rögzített m-re ugyanannyi elemük van, pl. $m = 10$-re mindegyiknek 4 eleme van.

De hány elemük van általában, azaz hány invertálható maradék van modulo m? Ennek megválaszolásához vezessük be a következő függvényt:

1.3. Definíció. Egy $n \in \mathbb{N}^+$-ra legyen $\boldsymbol{\varphi(n)}$ az n-hez relatív prím egészek száma 0 és $n-1$ között, azaz $\varphi(n) := |\{k \in \mathbb{Z} \mid 0 \leq k \leq n-1 \ \wedge\ \text{lnko}(n, k) = 1\}|$. Elnevezés: **Euler-féle $\boldsymbol{\varphi}$-függvény** (vagy röviden: *Euler-függvény*). (φ kiejtése: "fí".)

Példa. $\varphi(10) = |\{1, 3, 7, 9\}| = 4$ és $\varphi(7) = |\{1, 2, 3, 4, 5, 6\}| = 6$.

Sage-ben a $\varphi(n)$ függvény elnevezése `euler_phi(n)`. Állítsuk elő az első néhány értékét:

```
sage: matrix([[n, euler_phi(n)] for n in range(1, 26)]).transpose()
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25]
[ 1  1  2  2  4  2  6  4  6  4 10  4 12  6  8  8 16  6 18  8 12 10 22  8 20]
```

Vagy ábrázolhatjuk is:

```
sage: plot(euler_phi, 1, 100)
```

A $\varphi(n)$ függvény legfontosabb tulajdonságai:

1. $\varphi(1) = 1$ (hiszen lnko$(1, 0) = 1$).
2. Ha $n \geq 2$, akkor $\varphi(n) \leq n-1$ (hiszen lnko$(n, 0) = n \neq 1$).
3. Ha p prím, akkor (és csak akkor) $\varphi(p) = p-1$, mert 1-től $p-1$-ig minden szám relatív prím p-hez. (Vagyis $\varphi(n)$ a legmagasabb értékeit a prímszámokra veszi fel.)
4. Ha p prím, akkor $\varphi(p^k) = p^k - p^{k-1} = (p-1)p^{k-1}$, pl. $\varphi(16) = \varphi(2^4) = 2^4 - 2^3 = 8$. (Bizonyítás: az összes p^k számból ki kell venni p többszöröseit, amelyek száma $p^k/p = p^{k-1}$.)
5. Ha a és b relatív prímek, akkor $\varphi(ab) = \varphi(a) \cdot \varphi(b)$. Például $\varphi(10) = \varphi(2) \cdot \varphi(5) = 1 \cdot 4 = 4$. (A bizonyítás nehéz, ezért mellőzzük.) Fontos: ez a tulajdonság csak relatív prímekre igaz! Például $\varphi(2) \cdot \varphi(4) = 1 \cdot 2 = 2$, de $\varphi(2 \cdot 4) = \varphi(8) = 2^3 - 2^2 = 4 \neq 2$.

Általában, ha ismerjük n prímtényezős felbontását, akkor az utolsó két szabály ad egy egyszerű számítási módszert $\varphi(n)$-re. Például ha $n = 24 = 2^3 \cdot 3$, akkor, mivel 2^3 és 3 relatív prímek, ezért:

$$\varphi(24) = \varphi(2^3) \cdot \varphi(3) = (2^3 - 2^2) \cdot (3 - 1) = 4 \cdot 2 = 8.$$

Mivel különböző prímszámok hatványai mindig relatív prímek, ezért a prímfelbontással felbonthatjuk $\varphi(n)$-et prímhatványok $\varphi()$-jeinek szorzatára, azokra pedig már van képletünk. Még egy példa:

$$\varphi(4200) = \varphi(2^3 \cdot 3 \cdot 5^2 \cdot 7) = \varphi(2^3) \cdot \varphi(3) \cdot \varphi(5^2) \cdot \varphi(7) = (2^3 - 2^2) \cdot 2 \cdot (5^2 - 5) \cdot 6 = 960.$$

Általában is felírhatjuk a képletet:

1.4. Tétel (φ kiszámítása)**.** *Ha n kanonikus alakja $n = p_1^{k_1} p_2^{k_2} \cdots p_m^{k_m}$, akkor:*

$$\varphi(n) = \left(p_1^{k_1} - p_1^{k_1-1}\right)\left(p_2^{k_2} - p_2^{k_2-1}\right) \cdots \left(p_m^{k_m} - p_m^{k_m-1}\right).$$

Speciális esetként, ha n két különböző prímszám szorzata, $n = pq$, akkor $\varphi(n) = (p-1)(q-1)$.

Megjegyzés: a fentiek alapján $\varphi(n)$-et könnyen ki tudjuk számítani, ha ismerjük n prímtényezős felbontását. Ha viszont nem ismerjük – és, mint látni fogjuk, nagy n-ekre a prímfaktorizáció nagyon nehéz feladat –, akkor $\varphi(n)$-et sem fogjuk tudni kiszámítani. Ez később még kulcsfontosságú lesz.

2. Moduláris hatványozás II.

Az előző jegyzetben (Diszkrét logaritmus) foglalkoztunk moduláris hatványozással, de elsősorban csak prímszám modulusok esetén. Most általánosítjuk a kérdést tetszőleges modulusra.

Például tekintsük modulo 10 az összes maradék első néhány hatványát:

```
sage: matrix([[power_mod(a, k, 10) for k in range(30)] for a in range(10)])
[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
[1 2 4 8 6 2 4 8 6 2 4 8 6 2 4 8 6 2 4 8 6 2 4 8 6 2 4 8 6 2]
[1 3 9 7 1 3 9 7 1 3 9 7 1 3 9 7 1 3 9 7 1 3 9 7 1 3 9 7 1 3]
[1 4 6 4 6 4 6 4 6 4 6 4 6 4 6 4 6 4 6 4 6 4 6 4 6 4 6 4 6 4]
[1 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5]
[1 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6]
[1 7 9 3 1 7 9 3 1 7 9 3 1 7 9 3 1 7 9 3 1 7 9 3 1 7 9 3 1 7]
[1 8 4 2 6 8 4 2 6 8 4 2 6 8 4 2 6 8 4 2 6 8 4 2 6 8 4 2 6 8]
[1 9 1 9 1 9 1 9 1 9 1 9 1 9 1 9 1 9 1 9 1 9 1 9 1 9 1 9 1 9]
```

Összehasonlítva az előző jegyzettel, ahol a modulus prímszám volt, a következőket vehetjük észre:

- Itt az 1 nem minden sorban ismétlődik. Bizonyos sorokban periodikusan visszatér, máshol viszont az $a^0 = 1$ az egyetlen 1-es hatvány.
- Ha nem is az 1-től, de előbb-utóbb minden sorban periodikusan ismétlődnek a hatványok.
- Mely alapokra ismétlődik az 1? Ezek az 1, a 3, a 7 és a 9, azaz éppen az invertálható alapok.
- Ezen számok *rendje* (azaz az első pozitív kitevő, amelyre a hatvány 1) a táblázatból leolvasható: 1, 2 vagy 4 (pl. $\mathrm{ord}_{10}(7) = 4$). Vegyük észre, hogy ezek éppen a 4 osztói.
- Akárcsak prímszám modulusra, itt is igaz, hogy $\mathrm{ord}_{10}(1) = 1$ és $\mathrm{ord}_{10}(10-1) = 2$.

Ahhoz, hogy ezen megállapítások többségét általánosíthassuk, szükségünk van egy általános tételre a moduláris hatványozásról; olyanra, mint a kis Fermat-tétel, amely prímszám modulusokra azt mondta, hogy $a^{p-1} \equiv 1 \pmod{p}$, ha $a \not\equiv 0 \pmod{p}$. Ezt általánosítjuk tetszőleges modulusra:

2.1. Tétel (Euler–Fermat-tétel)**.** *Ha* $m \in \mathbb{N}^+$*,* $a \in \mathbb{Z}$ *és* $\mathrm{lnko}(a, m) = 1$*, akkor* $a^{\varphi(m)} \equiv 1 \pmod{m}$.

Példa. $m = 10$-re $\varphi(10) = 4$, ezért $3^4 \equiv 1 \pmod{10}$, $7^4 \equiv 1 \pmod{10}$ stb.

Általában tehát nem a $p-1$-edik hatvány lesz 1, hanem a $\varphi(m)$-edik (prímszámokra a kettő egybeesik: $\varphi(p) = p-1$).

Nagyon fontos, hogy ez a tétel csak akkor működik, ha a és m relatív prímek. Ha nem azok, akkor nemcsak a $\varphi(m)$-edik hatvány nem lesz 1, de semelyik pozitív kitevőjű hatvány sem. Például $m = 10$-re leolvasható, hogy a páros számok minden hatványa páros, és az 5 minden hatványa 5. Általában könnyen belátható, hogy minden $a^n \bmod m$ hatvány osztható $\mathrm{lnko}(a, m)$-mel (ha $n \geq 1$). Ha tehát $\mathrm{lnko}(a, m) \neq 1$, azaz nem relatív prímek, akkor az 1 biztosan nem szerepelhet a hatványok között. Ha viszont relatív prímek, akkor az Euler–Fermat-tétel garantálja, hogy szerepel az 1.

A tételből az is következik, hogy az invertálható számok rendje ($\mathrm{ord}_m(a)$) mindig osztója $\varphi(m)$-nek. Például $m = 10$ esetén a rend 1, 2 vagy 4 lehet (lásd fent), és valóban, ezek mind osztói $\varphi(10) = 4$-nek. (Ezzel általánosítottuk az előző jegyzetben prímszám modulusra kimondott Lagrange-tételt.)

Az Euler–Fermat-tétellel bizonyos esetekben leegyszerűsíthetjük a moduláris hatványokat. Például tekintsük 137^{75} mod 10-et. Először is, természetesen az alapnak vehetjük a maradékát 10 szerint: $137 \equiv 7 \pmod{10}$, ezért $137^{75} \equiv 7^{75} \ (10)$. A kitevőre viszont ugyanez nem működik: $7^{75} \not\equiv 7^5 \ (10)$. Ehelyett használhatjuk az Euler–Fermat-tételt, amely alapján $7^{\varphi(10)} \equiv 1 \ (10)$, vagyis 75 maradékát nem 10 szerint, hanem $\varphi(10) = 4$ szerint kell venni: $7^{75} \equiv 7^{18 \cdot 4 + 3} \equiv \left(7^4\right)^{18} 7^3 \equiv 1^{18} \cdot 7^3 \equiv 7^3 \ (10)$. Így 137^{75} helyett már csak 7^3-t kell kiszámítani, amely sokkal egyszerűbb (és kiszámítható például gyorshatványozással, lásd az előző jegyzetet).

Ez az egyszerűsítés azonban nem mindig működik: egyrészt az Euler–Fermat-tételhez az kell, hogy az alap relatív prím legyen a modulushoz (amely itt teljesült: $\mathrm{lnko}(7, 10) = 1$), másrészt pedig $\varphi(m)$ kiszámításához szükség van az m prímtényezős felbontására, amelyet nagy m-ekre nehéz kiszámítani.

Összegezve tehát: ha az m modulusra ismerjük $\varphi(m)$-et, és $\mathrm{lnko}(a, m) = 1$, akkor az $a^n \bmod m$ hatványt úgy számíthatjuk ki, hogy az alapot vesszük modulo m, a kitevőt pedig modulo $\varphi(m)$: $a^n \equiv (a \bmod m)^{n \bmod \varphi(m)} \pmod{m}$, és ebből az alakból hajtjuk végre a gyorshatványozást.

De mi a helyzet, ha az alap nem relatív prím a modulushoz? Például nézzük 138^{75} mod 10-et. Az első lépés marad: $138^{75} \equiv 8^{75} \ (10)$. Az Euler–Fermat-tételt viszont nem használhatjuk, mert $\mathrm{lnko}(8, 10) \neq 1$. Az ötlet az, hogy bontsuk fel a modulust két részre: $10 = 5 \cdot 2$, és ennek megfelelően a feladatot is: 8^{75} mod 10 helyett számítsuk ki 8^{75} mod 5-öt és 8^{75} mod 2-t. Ez azért jó, mert az első felére $\mathrm{lnko}(8, 5) = 1$, tehát működik az Euler–Fermat-tétel: $8^{75} \equiv 3^{75} \equiv 3^{75 \bmod 4} \equiv 3^3 \equiv 2 \ (5)$, a második fele pedig triviális: $8^{75} \equiv 0^{75} \equiv 0 \ (2)$. A két részeredményből ($138^{75} \equiv 2 \ (5)$ és $138^{75} \equiv 0 \ (2)$) a kínai maradéktétellel tudjuk kiszámítani a végeredményt (lásd: Kongruenciák).

3. Az RSA-eljárás

Az RSA-eljárás egy gyakran használt titkosítási (kriptográfiai) módszer. A betűk az alkotók neveinek kezdőbetűit jelentik (Rivest, Shamir és Adleman). Az RSA-t számos helyen használják az internetes kommunikációtól (pl. HTTPS, SSH) a bankok biztonsági rendszereiig.

3.1. Aszimmetrikus titkosítás

Az RSA egy *aszimmetrikus titkosítás* [*public-key encryption*], ami azt jelenti, hogy két különböző kulcsot használ: az egyiket titkosításra, a másikat pedig a titkosított üzenet megfejtésére. Ezzel szemben egy *szimmetrikus titkosítás* (mint például a gyakran használt AES, amelyről egy későbbi jegyzetben lesz szó) ugyanazt a kulcsot használja mind a titkosításhoz, mind a megfejtéshez. A szimmetrikus titkosítások előnye az egyszerűségük (általában sokkal hatékonyabbak, mint az aszimmetrikus eljárások), viszont hátrányuk, hogy mindkét félnek ismerni kell ugyanazt a (titkos) kulcsot. Ehhez vagy találkozniuk kell előre titokban, hogy megbeszéljék a közös kulcsot, vagy alkalmazniuk kell egy olyan eljárást, mint a Diffie–Hellman-kulcscsere (lásd az előző jegyzetet: Diszkrét logaritmus).

Az aszimmetrikus titkosításoknál viszont, mint amilyen az RSA, két kulcs van:

- **Nyilvános kulcs** [*public key*]: nyilvánosságra hozható, és az üzenetek titkosítására használják.
- **Titkos kulcs** [*private key*]: titokban kell tartani, és ezzel lehet megfejteni a megfelelő nyilvános kulccsal titkosított üzeneteket.

A két félnek tehát nem kell előre megbeszélniük egy közös kulcsot, hanem az egyik fél közzéteszi a nyilvános kulcsát, a másik fél pedig azt használva titkosítja az üzenetet, amelyet az első fél a titkos kulcsával meg tud fejteni. Vagyis aszimmetrikus titkosításnál bárki tud titkosított üzenetet küldeni egy címzettnek, de azt csak a címzett tudja megfejteni.

Ahhoz, hogy egy aszimmetrikus titkosítás működjön, egyrészt az kell, hogy a két kulcs függjön egymástól, hiszen amit az egyikkel titkosítunk, azt a másikkal meg kell tudnunk fejteni; másrészt viszont az is kell, hogy a nyilvános kulcsból ne lehessen kiszámítani a titkos kulcsot, hiszen akkor az egésznek nem lenne semmi értelme. Ez a két feltétel látszólag ellentmond egymásnak. Ráadásul ha nem lehet az egyikből kiszámítani a másikat, akkor hogyan tudjuk őket egyáltalán előállítani?

3.2. Az RSA működése

Az RSA-eljárás azon alapul, hogy a prímfaktorizáció, azaz a prímtényezős felbontás kiszámítása nagy számokra nagyon nehéz feladat. A tudomány mai állása szerint nincs olyan hatékony algoritmus, amellyel faktorizálni tudnánk egy kellően nagy számot. Azaz ha van két nagy prímszámunk, p és q, akkor azokat könnyen össze tudjuk szorozni: $n := p \cdot q$, de az n-ből senki nem fogja tudni kiszámítani p-t és q-t. (Legalábbis ha elég nagyok, és bizonyos tulajdonságoknak megfelelnek, amelyeket itt nem részletezünk.) Megjegyzés: ez hasonlít az előző jegyzetben megismert diszkrét logaritmushoz, ahol szintén volt egy művelet, a moduláris hatványozás, amelyet könnyű elvégezni, viszont a fordítottját, a diszkrét logaritmust nehéz. Az ilyen műveletet *egyirányú függvény*nek nevezzük.

Az RSA alapvető művelete – akárcsak a Diffie–Hellman-kulcscserének – a moduláris hatványozás, de a modulus ezúttal nem prímszám, hanem két prímszám szorzata: $n = p \cdot q$. Mind a titkosítás, mind a megfejtés egy hatványozással történik az n modulus szerint, csak különböző kitevőkre: a

titkosítás során az m üzenetre [*plaintext*] kiszámítjuk a $c \equiv m^e \pmod{n}$ értéket, ahol c a titkosított üzenet [*cyphertext*] és e a *nyilvános kitevő*, a megfejtés során pedig m-et visszaállítjuk az $m \equiv c^d \pmod{n}$ hatványozással, ahol d a *titkos kitevő*. A kérdés persze az, hogy mi legyen e és d, hogy ez így működjön.

Ahhoz, hogy ugyanazt az m-et kapjuk vissza, mint amelyet titkosítottunk, a következő kell:

$$m \equiv c^d \equiv (m^e)^d \equiv m^{ed} \pmod{n}.$$

Az Euler–Fermat-tételből (lásd fent) tudjuk, hogy a kitevőnek vehetjük a maradékát modulo $\varphi(n)$, azaz $m^{ed} \equiv m^{ed \bmod \varphi(n)} \pmod{n}$. Tehát ha e és d olyan, hogy $ed \equiv 1 \pmod{\varphi(n)}$ – azaz e és d egymás inverze modulo $\varphi(n)$ –, akkor éppen $m^{ed} \equiv m \pmod{n}$ lesz, ahogy szeretnénk. (Megjegyzés: ha $\text{lnko}(m, n) \neq 1$, akkor az Euler–Fermat-tételt nem használhatjuk, de más módszerrel akkor is bebizonyítható, hogy $m^{ed} \equiv m \pmod{n}$.)

A nyilvános kitevőből, e-ből csak akkor tudjuk kiszámítani a titkos kitevőt, d-t, ha ismerjük $\varphi(n)$-et. De ahhoz, hogy kiszámítsuk $\varphi(n)$-et, szükségünk van az n prímtényezős felbontására, azaz p-re és q-ra. Tehát az RSA-eljárás biztonságát valóban az adja, hogy a prímfaktorizáció nehéz.

Az RSA első lépése a **kulcsgenerálás**:

1. Generáljunk két különböző, véletlen, titkos prímszámot: p-t és q-t.
2. Számítsuk ki a szorzatukat: $n := pq$.
3. Számítsuk ki $\varphi(n)$-et: $\varphi(n) = \varphi(pq) = \varphi(p)\varphi(q) = (p-1)(q-1)$.
4. Válasszunk egy nyilvános kitevőt, e-t úgy, hogy $2 < e < \varphi(n)$ és $\text{lnko}(e, \varphi(n)) = 1$.
5. Oldjuk meg az $ed \equiv 1 \ (\varphi(n))$ kongruenciát d-re, azaz számítsuk ki e inverzét modulo $\varphi(n)$.
6. Nyilvános kulcs: az (n, e) pár. Ezt közzétehetjük.
7. Titkos kulcs: az (n, d) pár. Ezt tartsuk titokban.

(Megjegyzés: nemcsak d, hanem p, q és $\varphi(n)$ is titkos értékek, de azokra a továbbiakban nem lesz szükség, így törölhetjük azokat.)

Ezután az RSA-val így lehet **titkosítani**:

1. Írjuk fel a titkosítandó üzenetet egy m egész számként 0 és $n-1$ között. (Ha az üzenet túl hosszú, akkor bontsuk fel kisebb részekre, és külön-külön titkosítsuk azokat.)
2. Titkosítás: számítsuk ki a $c \equiv m^e \pmod{n}$ hatványt az (n, e) nyilvános kulcsból.
3. Megfejtés: számítsuk ki az $m \equiv c^d \pmod{n}$ hatványt az (n, d) titkos kulcsból.

Akárcsak a Diffie–Hellman-kulcscserénél, itt is nagyon fontos, hogy a két prímszámot, p-t és q-t biztonságos véletlenszám-generátorral állítsuk elő (lásd az előző jegyzetet), hogy egy külső megfigyelő ne tudja őket megjósolni. Szintén nagyon fontos, hogy olyan nagyok legyenek (és megfelelő tulajdonságúak), hogy n-et ne lehessen felbontani p-re és q-ra. Manapság jellemző a 2048-bites és a 4096-bites RSA, azaz n hossza kettes számrendszerben 2048, ill. 4096 bit (tehát p és q hossza egyenként kb. 1024, ill. 2048 bit).

És hogyan állítjuk elő e-t, a nyilvános kitevőt? Mivel nyilvános, ezért nem kell véletlennek lennie, viszont érdemes olyan értéket választani, amellyel hatékonyan lehet számolni. Gyakran használják az $e = 2^{16} + 1 = 65537$ értéket, amely elég egyszerű ahhoz, hogy könnyű legyen vele számolni (emlékezzünk arra, hogy a gyorshatványozás annál gyorsabb, minél kevesebb 1-es bit van a kitevő bináris alakában), de elég összetett ahhoz, hogy ne veszélyeztesse az RSA biztonságát.

3.3. Digitális aláírás

Mint láttuk, az RSA-val úgy lehet titkosítani, hogy az üzenetet a nyilvános kulccsal titkosítjuk, és a titkos kulccsal fejtjük meg. Ez azt jelenti, hogy bárki tud titkos üzenetet küldeni egy címzettnek, de csak a címzett, azaz a titkos kulcs birtokosa tudja megfejteni azt.

De mi történik, ha ezt megfordítjuk? Mi történik, ha a titkos kulccsal titkosítjuk, és a nyilvános kulccsal fejtjük meg az üzenetet? Ekkor csak a titkos kulcs birtokosa tud titkosítani, de bárki meg tudja fejteni az üzenetet. Van ennek bármi értelme?

Ezt úgy hívják, hogy *digitális aláírás* (vagy *hitelesítés*), amely a titkosításon kívül egy másik fontos alkalmazása az RSA-nak. Ilyenkor az m üzenetet a titkos kulccsal "titkosítjuk" (kódoljuk), azaz előállítjuk a $c \equiv m^d \pmod{n}$ értéket, és ezt továbbítjuk m-mel együtt. Ezután az üzenet címzettje a nyilvános kulcs segítségével vissza tudja állítani c-ből m-et: $m \equiv c^e \pmod{n}$, és ha ez sikerült, akkor biztos lehet benne, hogy az üzenet valóban a titkos kulcs birtokosától származik, aki (jó esetben) csak a feladó lehet.

3.4. Gyorsítás kínai maradéktétellel

Mint láttuk, a nyilvános kitevőt, e-t megválaszthatjuk úgy, hogy könnyű legyen vele számolni. A titkos kitevőre, d-re viszont már nincs közvetlen befolyásunk, ezért az lehet nagyon nagy is – ami a biztonság szempontjából persze egyáltalán nem baj. Ez viszont azt jelenti, hogy d-ik hatványra emelni (azaz megfejteni az üzenetet) sokkal lassabb, mint e-ikre (azaz titkosítani).

Lehet azonban ezt gyorsítani, ha nem közvetlenül számítjuk ki az $m \equiv c^d \pmod{n}$ hatványt, hanem két részletben, azaz az $n = pq$ modulus helyett külön-külön p-re és q-ra: $m_p \equiv c^d \pmod{p}$ és $m_q \equiv c^d \pmod{q}$. A két eredményt a kínai maradéktétellel tudjuk egyesíteni (lásd a Kongruenciák c. jegyzetet). Ez a felbontás azért jó, mert így nemcsak kisebb számokkal kell számolni (a kisebb modulusok miatt), de a kitevő hosszát is csökkenthetjük: a kis Fermat-tétel miatt elég a $d_p := d \bmod (p-1)$, ill. a $d_q := d \bmod (q-1)$ kitevőket használni. Így bár egy helyett két moduláris hatványozást végzünk, de azok egyenként több mint négyszer olyan gyorsak: a kitevők is fele olyan hosszúak, tehát fele annyi szorzás kell, és a szorzandó számok is fele akkorák.

A gyorsított $m \equiv c^d \pmod{n}$ hatványozás tehát így történik:

1. Számítsuk ki előre a következő értékeket:
   - $d_p := d \bmod (p-1)$,
   - $d_q := d \bmod (q-1)$,
   - q inverzét (q^{-1}) modulo p.
2. $m_p := c^{d_p} \bmod p$.
3. $m_q := c^{d_q} \bmod q$.
4. A kínai maradéktétellel számítsuk ki m_p-ből és m_q-ból m-et:
   - $m_d := q^{-1}(m_p - m_q) \bmod p$.
   - $m := m_q + m_d q$.

A fenti gyorsítás miatt gyakran nem is az (n, d) párt szokták titkos kulcsként tárolni, hanem a (p, q, d_p, d_q, q^{-1}) ötöst.

4. Prímtesztek

Az RSA-val kapcsolatban egy fontos részletről még nem volt szó: hogyan generáljuk a két véletlen prímszámot, p-t és q-t? Biztonságos véletlenszám-generátorokról már volt szó (lásd az előző jegyzetet: Diszkrét logaritmus), de honnan tudjuk, hogy az eredmény prímszám-e? Hiszen az RSA éppen azon alapul, hogy egy nagy számot nem tudunk faktorizálni – de ha nem tudjuk az osztóit megtalálni, akkor hogyan ellenőrizzük, hogy prímszám-e?

Szerencsére eldönteni egy számról, hogy prímszám-e, sokkal könnyebb feladat, mint prímtényezőkre bontani. Az ilyen módszereket *prímteszt*eknek nevezzük. Kétféle prímteszttel foglalkozunk:

1. A *determinisztikus prímteszt*ek biztosan meg tudják mondani, hogy egy szám prímszám-e.
2. A *valószínűségi prímteszt*ek nem 100%-ban adnak helyes választ, néha hibáznak. A tipikus tesztek azonban csak az egyik irányban tévednek: prímszámokra biztosan helyes választ adnak, de összetett számokat néha tévesen prímszámnak jeleznek.

De miért használnánk olyan teszteket, amelyek nem mindig mondanak igazat? Azért, mert a valószínűségi prímtesztek általában sokkal gyorsabbak, mint a determinisztikusak. Például a próbaosztásos módszer, amely 2-től $\sqrt{n}$-ig végignézi az összes számra, hogy osztja-e n-et (lásd: Oszthatóság, prímszámok), egy determinisztikus prímteszt, de nagy számokra nagyon lassú (hiszen lényegében olyan, mint a prímfaktorizáció). Az alábbiakban leírt módszerek viszont sokkal hatékonyabbak, és elég ritkán tévednek ahhoz, hogy elfogadhatóak legyenek.

A Sage `is_prime()` függvénye alapértelmezésként determinisztikus prímtesztet használ, de átállítható valószínűségi prímtesztre, hogy gyorsabb legyen, megengedve a tévedés kockázatát:

```
sage: n = 2^2048 + 981
sage: is_prime(n)    # determinisztikus: LASSÚ!
True
sage: proof.arithmetic(False)
sage: is_prime(n)    # valószínűségi: gyors, de megbízhatatlan
True
```

4.1. Fermat-prímteszt

A kis Fermat-tétel csak prímszámokról szól: ha p prímszám és $a \not\equiv 0 \pmod{p}$, akkor $a^{p-1} \equiv 1 \pmod{p}$. De vajon lehet-e igaz ez az állítás, ha p nem prímszám?

Írassuk ki az $a^{n-1} \bmod n$ értékeket különböző n-ekre és különböző a-kra (ahol $1 \leq a \leq n-1$):

```
sage: for n in range(9, 16):
....:     print(n, [power_mod(a, n-1, n) for a in range(1, n)])
9 [1, 4, 0, 7, 7, 0, 4, 1]
10 [1, 2, 3, 4, 5, 6, 7, 8, 9]
11 [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
12 [1, 8, 3, 4, 5, 0, 7, 8, 9, 4, 11]
13 [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
14 [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
15 [1, 4, 9, 1, 10, 6, 4, 4, 6, 10, 1, 9, 4, 1]
```

Vegyük észre a következőket:

1. Ha n prímszám, akkor minden érték 1, ahogy azt a kis Fermat-tétel garantálja.
2. Ha $a = 1$ (első értékek), akkor annak minden hatványa 1.
3. Ha $a = n-1$ (utolsó értékek), akkor is gyakran kapunk 1-et, hiszen $n-1 \equiv -1 \pmod{n}$, és -1 minden második hatványa 1.
4. Más esetben azonban, azaz ha n összetett szám és $2 \leq a \leq n-2$, szinte soha nem kapunk 1-et.
5. De néha igen: például $n = 15$-re és $a = 4$-re $4^{14} \equiv 1 \pmod{15}$.

Ezekből az észrevételekből készíthetünk egy valószínűségi prímtesztet: számítsuk ki a^{n-1} mod n-et valamilyen a-ra, és ha nem 1-et kapunk, akkor n biztosan nem prímszám, ha pedig 1-et, akkor *valószínűleg* prímszám. Ezt úgy hívják, hogy **Fermat-prímteszt**, és a következőképpen működik:

Fermat-prímteszt(n):

> **Bemenet:** $n \in \mathbb{N}, n \geq 4$, a tesztelendő szám
> Válasszunk egy véletlen $a \in \{2, 3, \ldots, n-2\}$ alapot.
> $x := a^{n-1} \bmod n$
> **ha** $x \neq 1 \rightarrow$ *hamis* (= "nem prímszám")
> **ha** $x = 1 \rightarrow$ *igaz* (= "valószínűleg prímszám")

Ez az algoritmus nagyon hatékony, hiszen csak egy moduláris hatványozást kell elvégezni, amely gyorshatványozással megoldható. (Összehasonlítva a próbaosztásos módszerrel, ahhoz kb. $\sqrt{n}$ osztás kell, míg a Fermat-prímteszthez csak kb. $2\log_2 n$ szorzás (lásd a gyorshatványozást). Például ha $n \approx 1000000$, akkor az előbbi kb. 1000, míg az utóbbi csak kb. 40.)

A tévedés valószínűségét csökkenthetjük, ha többször futtatjuk le a tesztet (különböző véletlen a-kra), és csak akkor fogadjuk el a számot prímszámnak, ha minden teszt igazat ad. Például ha az $n = 15$-öt csak $a = 4$-gyel tesztelnénk, akkor n átmenne a teszten, hogy "prímszám". De ha megismételjük a tesztet pl. $a = 2$-vel, akkor n már "lebukik", hogy összetett szám.

4.1. Definíció. Egy adott $a \in \mathbb{N}^+$ számra az n összetett számot **Fermat-álprím**nek nevezzük, ha $a^{n-1} \equiv 1 \pmod{n}$. (Néha egyszerűen csak jelző nélkül **álprím**nek [*pseudoprime*] hívják.)

Példa. $n = 15$ egy Fermat-álprím $a = 4$-re, mert $4^{14} \equiv 1 \pmod{15}$.

Vagyis a Fermat-álprímek azok, amelyek "becsapják" a Fermat-prímtesztet.

Megjegyzés: ez a fogalom függ az alaptól, a-tól, azaz ugyanaz az n lehet az egyik a-ra álprím, de a másik a-ra nem. Ennek hangsúlyozására használhatjuk a következő fogalmat:

4.2. Definíció. Egy n összetett számra az $a \in \mathbb{N}^+$ alapot **Fermat-tanú**nak nevezzük, ha n nem Fermat-álprím az a alapra.

Példa. $a = 2$ Fermat-tanú $n = 15$-re, mert $2^{14} \equiv 4 \not\equiv 1 \ (15)$, $a = 4$ viszont nem (lásd fent).

Szerencsére azonban a Fermat-álprímek viszonlag ritkák. Például $a = 2$-re 1-től 10000-ig csak 22 Fermat-álprímet találunk (a legkisebb álprím a 341) – miközben ugyanezen intervallumban 1229 prímszám van.

Vannak viszont olyan álprímek, amelyek nagyon makacsul ellenállnak a Fermat-prímtesztnek:

4.3. Definíció. Egy n összetett szám **Carmichael-szám**, ha minden $a \in \mathbb{N}$ alapra, amely relatív prím az n-hez, $a^{n-1} \equiv 1 \pmod{n}$.

Vagyis a Carmichael-számok azok, amelyek a lehető legtöbb alapra Fermat-álprímek. (Ennél több alapra már nem is lehetnének, hiszen ha a és n nem relatív prímek, akkor a hatvány biztosan nem lehet 1, ahogy azt a jegyzet első felében megállapítottuk.) A legkisebb Carmichael-szám az 561. Ezek szerencsére még ritkábbak, mint a közönséges Fermat-álprímek (például 1-től 10000-ig csak 7 Carmichael-szám van).

A jó hír az, hogy ha egy álprím nem Carmichael-szám, akkor viszonylag könnyű "lebuktatni":

4.4. Tétel. *Ha n összetett szám, de nem Carmichael-szám, akkor az 1 és $n-1$ közötti a-k több mint felére $a^{n-1} \not\equiv 1 \pmod{n}$.*

Ezek szerint tehát a Fermat-prímteszttel kapcsolatban három eset lehetséges:

1. Ha n prímszám, akkor a teszt biztosan helyes eredményt ad.
2. Ha n Carmichael-szám (ami ritka), akkor a teszt nagyon megbízhatatlan.
3. Egyébként viszont a teszt több mint 50% valószínűséggel helyes eredményt ad. Ez így első ránézésre kevésnek tűnik, de ez csak egy pesszimista alsó korlát, és ha megismételjük a tesztet különböző a-kra, akkor ez a korlát is tovább csökkenthető: k futtatás után a hiba valószínűsége $< 1/2^k$, pl. $k = 10$ esetén kevesebb, mint 0,001 (= 0,1%).

4.2. Miller–Rabin-prímteszt

Valós számokra tudjuk, hogy az $x^2 = 1$ egyenletnek csak két megoldása van: $x = 1$ és $x = -1$. A következő tétel azt mondja, hogy ez bizonyos esetekben moduláris aritmetikára is igaz:

4.5. Tétel. *Ha p prímszám és $x^2 \equiv 1 \pmod{p}$, akkor $x \equiv 1 \pmod{p}$ vagy $x \equiv -1 \pmod{p}$.*

Példa. Ha a modulus $p = 5$, akkor a $0, 1, 2, 3, 4$ számok moduláris négyzete rendre $0, 1, 4, 4, 1$, azaz az $x^2 \equiv 1\ (5)$ kongruenciának valóban csak az $x \equiv 1\ (5)$ és az $x \equiv 4 \equiv -1\ (5)$ megoldásai vannak. Összetett számokra viszont nem feltétlenül igaz a tétel, például $3^2 \equiv 1\ (8)$.

(Bizonyítás: a feltétel szerint $p \mid x^2 - 1 = (x-1)(x+1)$, de prímszám egy szorzatot csak úgy oszthat, ha valamelyik tényezőjét is osztja, azaz $p \mid (x-1)$ vagy $p \mid (x+1)$.)

Induljunk ki ismét a kis Fermat-tételből, azaz hogy egy p prímszámra $a^{p-1} \equiv 1 \pmod{p}$. Például ha $p = 29$, akkor $a^{28} \equiv 1\ (29)$. Ha alkalmazzuk erre a fenti tételt $x = a^{14}$-re (azaz $x^2 = a^{28}$), vagyis felezzük a kitevőt, akkor azt kapjuk, hogy $a^{14} \equiv 1\ (29)$ vagy $a^{14} \equiv -1\ (29)$. Ha $a^{14} \equiv 1\ (29)$, akkor még egyszer felezhetjük a kitevőt, és a tételből azt kapjuk, hogy $a^7 \equiv 1\ (29)$ vagy $a^7 \equiv -1\ (29)$. Itt viszont a kitevő páratlan, ezért nem tudjuk ezt tovább folytatni.

A fenti műveletsor minden lépéséhez felhasználtuk, hogy p prímszám. Ha nem az, akkor jó eséllyel valamelyik lépés elromlik (de nem biztos). Ezzel tehát kaptunk egy újabb valószínűségi prímtesztet. Például ha $n = 15$ és $a = 4$, akkor $a^{n-1} \equiv 1\ (n)$, azaz $4^{14} \equiv 1\ (15)$ még teljesül, viszont ha felezzük a kitevőt, akkor $4^7 \equiv 4\ (15)$, de $4 \neq 1$ és $4 \neq -1$, ami megsérti a fenti "kitevőfelezős" tételt, vagyis 15 nem lehet prímszám.

A **Miller–Rabin-prímteszt** ezen az elven működik, csak a műveleteket fordított sorrendben hajtja végre: nem az a^{n-1}-t számítja ki először, és felezgeti a kitevőt, hanem először kiszámítja az utolsó, már nem felezhető kitevőre a hatványt (pl. a fenti $p = 29$-re a^7-t), és onnan duplázgatja vissza a kitevőt $n-1$-ig (pl. $a^7 \to a^{14} \to a^{28}$). Ennek az az értelme, hogy a kisebb hatványt könnyebb kiszámítani, és nem is kell mindig visszamenni egészen $n-1$-ig, ha már hamarabb is eldőlt az eredmény (például mert megsértette a fenti "kitevőfelezős" tételt).

Miller–Rabin-prímteszt(n):

Bemenet: $n \in \mathbb{N}, n \geq 4$, a tesztelendő szám
ha n *páros* $\rightarrow$ *hamis* (= "nem prímszám")
Válasszunk egy véletlen $a \in \{2, 3, \ldots, n-2\}$ alapot.
Felezgessük $d := n-1$-et addig, amíg páratlan nem lesz, és legyen k a felezések száma.
$x := a^d \bmod n$
ha $x = 1 \rightarrow$ *igaz* (= "valószínűleg prímszám")
ciklus k*-szor*
  ha $x = n-1 \rightarrow$ *igaz* (= "valószínűleg prímszám")
  $x := x^2 \bmod n$
  ha $x = 1 \rightarrow$ *hamis* (= "nem prímszám")
$\rightarrow$ *hamis* (= "nem prímszám")

Az algoritmus addig duplázza a kitevőt, amíg a következők valamelyike nem teljesül:

1. A hatvány $x = -1$ lesz (pontosabban $n-1$, mert $n-1 \equiv -1\ (n)$): ekkor n átmegy a teszten ("valószínűleg prím"), mert -1 négyzete 1, és onnantól kezdve mindig 1-et kapunk.
2. A hatvány $x = 1$ lesz: ekkor n nem lehet prímszám, mert ha az lenne, akkor az előző lépésben 1-et vagy -1-et kellett volna kapnunk, és akkor amiatt már megálltunk volna.
3. A kitevő eléri az $n-1$-et k duplázás után (ahol $n-1 = 2^k d$): ekkor n nem lehet prímszám, mert ha az lenne, akkor a hatvány 1 lenne (kis Fermat-tétel), és akkor amiatt már megálltunk volna.

4.6. Definíció. Egy adott $a \in \mathbb{N}^+$ számra az n összetett számot **erős álprím**nek nevezzük, ha a Miller–Rabin-prímteszt elfogadja n-et prímszámnak az adott a-ra.

Mivel a Miller–Rabin-prímteszt a Fermat-prímteszt kiterjesztése, ezért az erős álprímek egyúttal Fermat-álprímek is, de ez fordítva nem igaz (például $n = 15$ Fermat-álprím $a = 4$-re, de nem erős álprím, ahogy azt a fenti példából láthattuk).

A Miller–Rabin-prímteszt több szempontból is jobb, mint a Fermat-prímteszt:

1. Itt nincsenek olyan számok, mint a Fermat-prímtesztnél a Carmichael-számok, azaz amelyekre a Miller–Rabin-prímteszt majdnem mindig rossz eredményt adna.
2. A Miller–Rabin-teszt az esetek 75%-ában ad helyes eredményt (szemben a Fermat-prímteszt 50%-ával). Vagyis k-szori ismétléssel a hiba valószínűsége $< 1/4^k$ lesz (szemben az $1/2^k$-nal), tehát ugyanazt a pontosságot feleannyi lépéssel érhetjük el ($1/4^k = 1/2^{2k}$).
3. Ráadásul a teszt egy lépése is tipikusan gyorsabb, mint a Fermat-prímteszté, mert nem mindig számítja ki a teljes a^{n-1} hatványt, hanem gyakran hamarabb is megállhat.

5. Szerző

Ezt a jegyzetet Uray M. János írta, részben felhasználva az alábbi forrásokat:

[1] https://compalg.elte.gitlab-pages.hu/dimoa-web/tematika/dimoa
[2] https://en.wikipedia.org/wiki/Euler%27s_totient_function
[3] https://en.wikipedia.org/wiki/RSA_cryptosystem
[4] Rivest, Shamir, Adleman (1977): A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. https://publications.csail.mit.edu/lcs/pubs/pdf/MIT-LCS-TM-082.pdf
[5] https://en.wikipedia.org/wiki/Fermat_primality_test

Hibákat, észrevételeket itt lehet jelezni: uray.janos@inf.elte.hu.