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, amely csak az invertálható számokat tartalmazza:

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.) De fontos: ez a tulajdonság csak relatív prímekre igaz! Például $\varphi(2) \cdot \varphi(4) = 1 \cdot 2 = 2$, de $\varphi(2 \cdot 4) = \varphi(8) = 2^3 - 2^2 = 4 \neq 2$.

A fentiek alapján bármely n-re ki tudjuk számítani $\varphi(n)$-et, ha ismerjük n prímfelbontását.

Példa. Ha $n = 24 = 2^3 \cdot 3$, akkor, mivel 2^3 és 3 relatív prímek, ezért:

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

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

Példa.

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

Általánosan 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, $n = p \cdot q$, 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.

Az Euler–Fermat-tételből következik, hogy a hatványok periodikusan ismétlődnek $\varphi(m)$ szerint (ha $\mathrm{lnko}(a, m) = 1$):

$$a^{n+\varphi(m)} \equiv a^n a^{\varphi(m)} \equiv a^n 1 \equiv a^n \pmod{m}.$$

Ennél rövidebb periódussal is ismétlődhetnek, sőt, az is belátható, hogy a rend ($\mathrm{ord}_m(a)$) 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.)

Megjegyzés: bár az Euler–Fermat-tétel csak invertálható alapokról szól, így teljes ismétlődés is csak ezeknél lehetséges (hiszen a többinél már az 1 sem ismétlődik), viszont a többi alap hatványai is *előbb-utóbb* ismétlődnek. Belátható, hogy ezekre is igaz, hogy a végső periódus hossza legfeljebb $\varphi(m)$, egészen pontosan ennek osztója.

Az Euler–Fermet-tétel egyik fontos alkalmazása a moduláris hatványok kitevőinek egyszerűsítése:
Példa. Számítsuk ki a $137^{75} \bmod 10$ hatványt. Először is, természetesen az alapnak vehetjük a maradékát 10 szerint:

$$137^{75} \equiv (137 \bmod 10)^{75} \equiv 7^{75} \pmod{10}.$$

A kitevőre viszont ugyanez nem működik ($7^{75} \not\equiv 7^5 \pmod{10}$). Ehelyett használhatjuk az Euler–Fermat-tételt, amely alapján a kivetőt nem modulo 10, hanem modulo $\varphi(10) = 4$ kell venni:

$$7^{75} \equiv 7^{75 \bmod 4} \equiv 7^3 \pmod{10}.$$

Ez azért működik, mert az Euler–Fermat-tétel szerint $7^{\varphi(10)} \equiv 1 \pmod{10}$, azaz:

$$7^{75} \equiv 7^{18 \cdot 4 + 3} \equiv \left(7^4\right)^{18} 7^3 \equiv 1^{18} \cdot 7^3 \equiv 7^3 \pmod{10}.$$

A $137^{75} \bmod 10$-et tehát leegyszerűsítettük $7^3 \bmod 10$-re, amelyet már sokkal könnyebb kiszámítani:

$$7^3 \equiv 7^2 \cdot 7 \equiv 49 \cdot 7 \equiv 9 \cdot 7 \equiv 63 \equiv 3 \pmod{10}.$$

A fent bemutatott egyszerűsítésnek van egy elméleti és egy gyakorlati feltétele: egyrészt az Euler–Fermat-tételhez az kell, hogy az alap relatív prím legyen a modulushoz (amely itt teljesült: $\text{lnko}(7, 10) = 1$); másrészt pedig ki kell tudni számítani $\varphi(m)$-et az m modulusra. Ez utóbbihoz szükség van az m prímtényezős felbontására, amelyet csak akkor tudunk kiszámítani, ha m nem túl nagy, vagy speciális alakú.

Összegezve tehát: ha ismerjük $\varphi(m)$-et, és $\text{lnko}(a, m) = 1$, 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 innen már könnyebb kiszámítani a hatványt (például gyorshatványozással, lásd az előző jegyzetet).

De mi a helyzet akkor, ha az alap nem relatív prím a modulushoz? Például nézzük $138^{75} \bmod 10$-et (ahol $\text{lnko}(138, 10) = 2$). Az első lépés továbbra is működik:

$$138^{75} \equiv (138 \bmod 10)^{75} \equiv 8^{75} \pmod{10}.$$

Az Euler–Fermat-tételt viszont közvetlenül nem használhatjuk, mert $\text{lnko}(8, 10) \neq 1$. Az ötlet az, hogy bontsuk fel a modulust két részre: $10 = 5 \cdot 2$, és ennek megfelelően a feladatot is: $8^{75} \bmod 10$ helyett számítsuk ki $8^{75} \bmod 5$-öt és $8^{75} \bmod 2$-t. Ez azért jó, mert az első felére teljesül, hogy $\text{lnko}(8, 5) = 1$, ezért arra már alkalmazhatjuk az Euler–Fermat-tételt:

$$8^{75} \equiv 3^{75} \equiv 3^{75 \bmod 4} \equiv 3^3 \equiv 2 \pmod{5},$$

a második fele pedig triviális:

$$8^{75} \equiv 0^{75} \equiv 0 \pmod{2}.$$

A két részeredményből a kínai maradéktétellel tudjuk kiszámítani a végeredményt (lásd: Kongruenciák):

$$\left.\begin{aligned} x &\equiv 2 \pmod{5} \\ x &\equiv 0 \pmod{2} \end{aligned}\right\} \quad \longrightarrow \quad x \equiv 2 \cdot (-2) \cdot 2 + 0 \cdot 1 \cdot 5 \equiv -8 \equiv 2 \pmod{10}.$$

3. Az RSA-eljárás

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

Vigyázat: ez a jegyzet az RSA alapvető koncepcióinak ismertetésére szorítkozik, így jelentős egyszerűsítéseket tartalmaz (ezt a változatot szokták "tankönyvi RSA"-nak is nevezni). Ahhoz, hogy az RSA valóban biztonságos legyen, ennél több információra van szükség, ezért az itt leírt algoritmust ne használjuk valódi adatok titkosítására. (Általánosabban, bármilyen kriptográfiai algoritmus implementálását – játékpéldákat leszámítva – bízzuk a témában jártas szakemberekre, mert a legkisebb hibák is súlyos biztonsági réseket okozhatnak.)

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 nem kell tehát előre megbeszélniük egy közös kulcsot, hanem az egyik fél közzéteszi a nyilvános kulcsát, a másik fél pedig ezt használva titkosítja az üzenetet, amelyet így 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 visszaállítani p-t és q-t. (Legalábbis ha elég nagyok, és bizonyos matematikai feltételeknek megfelelnek, amelyeket itt nem részletezünk.)

Megjegyzés: ez hasonlít az előző jegyzetben megismert diszkrét logaritmusra, 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 üzenetből [*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 az $m \equiv c^d \pmod{n}$ hatványozással fogjuk visszaállítani, ahol d a *titkos kitevő*.

A kérdés persze az, hogy mi legyen e és d, hogy ez így mindig működjön.

Ahhoz, hogy ugyanazt az m-et kapjuk vissza, mint amit 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 azt 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: $n := p \cdot q$.
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.
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.

(Megjegyzés: az üzenet egész számként való felírása nem triviális feladat. Egyrészt, ha túl hosszú, akkor fel kell bontani kisebb részekre. Másrészt a legegyszerűbb, bitenkénti felírás – itt nem részletezett okokból – nem biztonságos, hanem célszerű véletlen biteket hozzáadni az üzenethez ("padding"). Ez azonban már túlfeszíti a jegyzet kereteit.)

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 az $s \equiv m^d \pmod{n}$ értéket – az *aláírást* [*signature*] –, és ezt továbbítjuk m-mel együtt. Ezután az üzenet címzettje a nyilvános kulcs segítségével vissza tudja állítani s-ből m-et: $m \equiv s^e \pmod{n}$, és ha ez megegyezik a kapott m-mel, akkor biztos lehet benne, hogy az üzenet valóban a titkos kulcs birtokosától származik.

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 $n = p \cdot q$ modulus helyett külön-külön p-re és q-ra:

$$m \equiv c^d \pmod{n} \quad \longrightarrow \quad \begin{cases} m_p \equiv c^d \pmod{p}, \\ m_q \equiv c^d \pmod{q}. \end{cases}$$

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. 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

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