

Diszkrét matematika

11. előadás

MÁRTON Gyöngyvér
mgyongyi@ms.sapientia.ro

Sapientia Egyetem,
Matematika-Informatika Tanszék
Marosvásárhely, Románia

2024, őszi félév



Miről volt szó az elmúlt előadáson?

- moduláris hatványozás
- maradékosztályok, maradékrendszerek
- a kis Fermat tétel
- az Euler függvény, az Euler tétel
- az Euler függvényhez kapcsolódó összefüggések
- a Miller-Rabin prímteszt
- hatványok és generátor elemek

Miről lesz szó?

- A diszkrét logaritmus probléma
- A Diffie-Hellman kulcscsere
- Elliptikus görbék
- Elliptikus görbe Diffie-Hellman kulcscsere
- lineáris kongruenciák,
- az RSA rendszer,

A diszkrét logaritmus (DL) probléma

- a diszkrét logaritmus problémának (DL probléma) több verziója is megadható,
- a $\mathbb{Z}_p^*$ -ben, illetve az elliptikus görbéken megadott értelmezések mindegyikét széles körben alkalmazzák

1. értelmezés (A diszkrét logaritmus probléma $\mathbb{Z}_p^*$ -ben)

Az egész számok $\mathbb{Z}_p^* = \{1, 2, \dots, p-1\}$ halmaza esetében, ahol p prímszám, g generátor elem, és $g, A \in \mathbb{Z}_p^*$ a diszkrét logaritmusa probléma azt jelenti, hogy megkeressük azt az a pozitív egész számot, melyre fennáll:

$$g^a \equiv A \pmod{p}.$$

- az a számot A g alapú **diszkrét logaritmusának** hívjuk,
- **Diszkrét logaritmus feltételezés:** nagy számok esetében a DL problémára **nem ismert hatékony** algoritmus, jelenleg **minimum 2048 bites** prímszámot használnak,
- számos kriptorendszer biztonsága alapszik a DL problémán.

A Diffie-Hellman kulcscsere

- 1976-ban publikálták a szerzők, biztonsága a DL problémán alapszik,
- mérőöldkönek számít, amely alapjaiban meghatározta/meghatározza a számítógépes adatcsere biztonságát, a **publikus kulcsú kriptográfia** alapjait,
- két távoli egység (számítógép, mobil eszköz, stb.) **kulcscsere** mechanizmusára adott elsőként megoldást: hogyan tud megosztani két távoli egység egy nyilvános csatornán egy egész számot (bájt szekvenciát), amelyet rajtuk kívül más egység nem tud meghatározni,
- a **TLS 1.3** protokollban az elliptikus görbéken értelmezett Diffie Hellman típusú kulcscserét használják
- feltételezve, hogy a kommunikációban résztvevő két egység **A** és **B**, akkor a protokoll első lépéseként **A** és **B** egy hiteles szervertől lekér egy p prímszámot, és p -nek egy g generátor elemét
- a p és g értékek **publikus, domain paraméterek**,
- az **A** és **B** között megosztott egész számot a szakirodalom **mester kulcsnak**, **szimmetrikus kulcsnak** (master key) nevez, amely szigorúan titkos információ.

A Diffie-Hellman kulcscsere

- **A** a p, g ismeretében meghatározza az a, A értékeket, ahol:
 - $a \in \{2, \dots, p-2\}$ véletlenszerű, szigorún titkos, a **privát kulcs**
 - $A = g^a \pmod{p}$ a **publikus kulcs**
 - az A -t elküldi **B**-nek,
- **B** a p, g ismeretében meghatározza a b, B értékeket, ahol:
 - $b \in \{2, \dots, p-2\}$ véletlenszerű, szigorún titkos, a **privát kulcs**
 - $B = g^b \pmod{p}$ a **publikus kulcs**
 - B -t elküldi **A**-nak,

- **A** a közös K kulcsot a következőképpen határozza meg:

$$K = B^a \pmod{p}.$$

- **B** a közös K kulcsot a következőképpen határozza meg:

$$K = A^b \pmod{p}.$$

- helyesség:

$$K = A^b = B^a = g^{ab} \pmod{p}.$$

- a gyakorlatban fontos, hogy egy megbízható hatóság garantálja, hogy az A és B értékek, azaz a **publikus kulcsok hitelesek**,
- ezek hitelesítése azonban nem része az alap protokollnak.

A Diffie-Hellman kulcscsere, példa

- Legyen $p = 47, g = 13$, ahol $p = 2 \cdot 23 + 1$. Fennáll, hogy $13^{23} \equiv 46 \not\equiv 1 \pmod{47}$, azaz 13 generátor elem.
 - **A számításai:**
 - választ egy random a számot, legyen ez 12 és meghatározza $A = 13^{12} \equiv 9 \pmod{47}$,
 - elküldi **B**-nek az $A = 9$ -es számot,
 - **B számításai:**
 - választ egy random b számot, legyen ez 34 és meghatározza $B = 13^{34} \equiv 21 \pmod{47}$,
 - elküldi **A**-nak a $B = 21$ -es számot,
 - **A számítása:** $K = 21^{12} = 16 \pmod{47}$,
 - **B számítása:** $K = 9^{34} = 16 \pmod{47}$.
- a közös kulcs: $K = 16$.

A következő linken egy Diffie-Hellman kulcserén alapuló *kliens-szerver* Python program található.

Elliptikus görbék

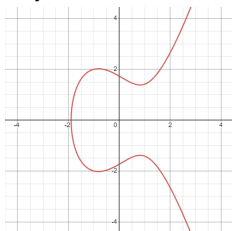
- a DL feltételezés az egész számok $(\text{mod } p)$ szerinti multiplikatív csoportjában *már nem számít elég nehéznek*,
- 2019: egy 795 bites prímszám esetében megoldották a DL problémát,
- az első ECC kriptó standardok 2000-ben jelentek meg,
- a kis kulcsméret, a jó biztonság miatt nagy a népszerűségük,
- leggyakrabban 256 bites kulcsokkal dolgoznak,
- használat: TLS, SSH, bitcoin, e-ID kártya, stb.
- Python crypto csomag: `PyCryptodome`

Elliptikus görbék

- három típusú görbét szoktak használni,
 - különböző **biztonságot** nyújtanak,
 - más **hatékonyságot** jelentenek,
 - másképp van értelmezve az **összeadás**

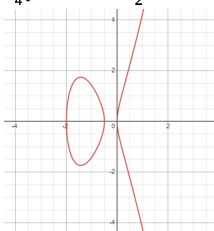
Weierstrass

$$y^2 = x^3 - 2x + 3$$



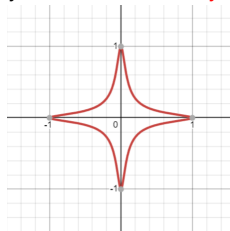
Montgomery

$$\frac{1}{4}y^2 = x^3 + \frac{5}{2}x^2 + x$$



Edwards

$$y^2 + x^2 = 1 - 300x^2y^2$$



Elliptikus görbék

- kriptóban a számításokat valamely $\mathbb{F}_p$ véges test felett végzik, és az alkalmazott alapművelet két pont összege lesz:

$$y^2 \equiv x^3 + ax + b \pmod{p},$$

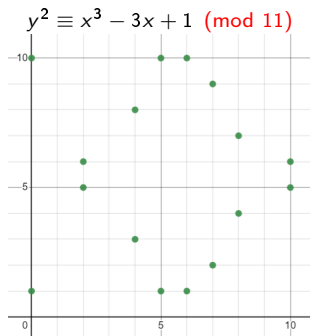
ahol fenn kell álljon:

- $a, b \in \mathbb{F}_p$, ahol $p > 3$ prímszám,
- $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$, amely biztosítja, hogy a görbe egyenletének nem lesz kétszeres gyöke,
- $(x, y) \in \mathbb{F}_p$ a görbe egy pontja lesz, ha kielégíti a görbe egyenletét
- az elliptikus görbét $E/\mathbb{F}_p$ -vel jelölik,
- ahhoz hogy az összeadást értelmezni lehessen a görbén található pontokhoz hozzá vesznek egy speciális pontot, a végtelen pontot, amelyet $\mathcal{O}$ -val jelölnék,
- a görbe pontjai és az $\mathcal{O}$ pont által meghatározott halmazt $E(\mathbb{F}_p)$ -vel jelölik:

$$E(\mathbb{F}_p) = \{(x, y) \mid y^2 \equiv x^3 + ax + b \pmod{p}\} \cup \{\mathcal{O}\}$$

Elliptikus görbék

- egy **véges test** felett értelmezett görbe grafikus alakját pontok adják,
- a következő görbe esetén a görbe pontjainak száma: $16 + 1$.



$$(0, 1), (0, 11 - 1)$$

$$(2, 5), (2, 11 - 5)$$

$$(4, 3), (4, 11 - 3)$$

$$(5, 1), (5, 11 - 1)$$

$$(6, 1), (6, 11 - 1)$$

$$(7, 9), (7, 11 - 9)$$

$$(8, 4), (8, 11 - 4)$$

$$(10, 5), (10, 11 - 5)$$

$$1^2 \equiv 0^3 - 3 \cdot 0 + 1$$

$$10^2 \equiv 0^3 - 3 \cdot 0 + 1$$

$$5^2 \equiv 2^3 - 3 \cdot 2 + 1$$

$$6^2 \equiv 2^3 - 3 \cdot 2 + 1$$

$$3^2 \equiv 4^3 - 3 \cdot 4 + 1$$

$$8^2 \equiv 4^3 - 3 \cdot 4 + 1$$

$$1^2 \equiv 5^3 - 3 \cdot 5 + 1$$

$$10^2 \equiv 5^3 - 3 \cdot 5 + 1$$

$$1^2 \equiv 6^3 - 3 \cdot 6 + 1$$

$$10^2 \equiv 6^3 - 3 \cdot 6 + 1$$

$$9^2 \equiv 7^3 - 3 \cdot 7 + 1$$

$$2^2 \equiv 7^3 - 3 \cdot 7 + 1$$

$$4^2 \equiv 8^3 - 3 \cdot 8 + 1$$

$$7^2 \equiv 8^3 - 3 \cdot 8 + 1$$

$$5^2 \equiv 10^3 - 3 \cdot 10 + 1$$

$$6^2 \equiv 10^3 - 3 \cdot 10 + 1$$

Elliptikus görbék

- legyen P a görbe egy pontja, ekkor két pont összeadására nézve az $E(\mathbb{F}_p)$ halmaz **véges, ciklikus Abel csoport** lesz:
 - megadható a **semleges elem**, az $\mathcal{O}$, amelyre fennáll: $P + \mathcal{O} = \mathcal{O} + P = P$,
 - megadható a $-P = (x_1, -y_1)$ **additív inverz elem**, minden $\mathcal{O} \neq P = (x_1, y_1) \in E(\mathbb{F}_{p^e})$ pont esetében,
 - a művelet **zárt, asszociatív**, és **kommutatív** lesz,
- a $P + P$ műveletet $2P$ -vel, a $P + P + P$ műveletet $3P$ -vel jelöljük,
- az αP azt jelenti, hogy α -szor adtuk össze a P -t, ahol α tetszőleges pozitív egész szám,
- az αP **hatékonyan meghatározható** $2\log_2 \alpha$ csoport művelettel,
- a P pont rendje q , ha fennáll: $qP = \mathcal{O}$,
- jelöljük N -el az $E(\mathbb{F}_{p^e})$ halmaz, azaz a görbe pontjainak elemszámát,
- egy N elemszámú görbe estében N -nek minden egyes d osztója meghatároz egy N/d elemszámú alcsoportot,

Elliptikus görbék

- az ECC rendszerek biztonsága az EC diszkrét logaritmus (ECDL) feltételezésen alapszik,
- **ECDL probléma:** legyen P egy pont az $E(\mathbb{F}_p)$ halmazban, amelynek rendje egyenlő a q egész számmal, ekkor az ECDL probléma azt jelenti, hogy ismerve az $P, \alpha P$ értékeket határozzuk meg az $\alpha \in \mathbb{Z}_q$ pozitív egész számot,
- **ECDL feltételezés:** megfelelően választott p és q értékek mellett nincs hatékony algoritmus, amely megoldaná az ECDL problémát,
- az ECDL problémát megoldó, leghatékonyabb algoritmus futási ideje, $O(\sqrt{q})$,
- gyakorlatban a p és q nagyságrendje 256 bit kell legyen,
- ha N minden prímosztója kisebb, mint 2^{80} , akkor az ECDL probléma könnyen megoldható, ezért a gyakorlatban olyan görbékkel dolgoznak, amelyek esetében N egyenlő $q, 4q$, vagy $8q$ -val, ahol q prímszám,
- hogy az ECDL probléma ne legyen hatékonyan megoldható, **előre kiválasztott görbével és alapponttal** dolgoznak.

Elliptikus görbék

Weierstrass görbe:

- az $E/\mathbb{F}_p$ elliptikus görbe egyenlete, ahol $p > 3$ prímszám a következő:

$$y^2 \equiv x^3 + ax + b \pmod{p}$$

és fennáll: $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$

- a $P = (x_1, y_1)$ és $Q = (x_2, y_2)$ pontok **összege** a következőképpen van értelmezve:

- ha $x_1 = x_2$ és $y_1 = -y_2$, akkor $P + Q = \mathcal{O}$
- másképp $P + Q = (x_3, y_3)$, ahol

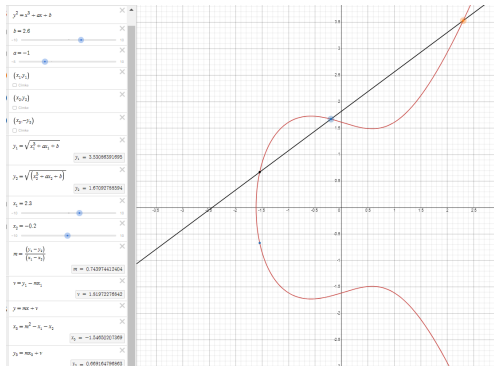
$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2 \\ y_3 &= \lambda(x_1 - x_3) - y_1 \\ \lambda &= \begin{cases} (y_1 - y_2) \cdot (x_1 - x_2)^{-1}, & \text{ha } P \neq Q \\ (3x_1^2 + a) \cdot (2y_1)^{-1}, & \text{ha } P = Q, y_1 \neq 0 \end{cases} \end{aligned}$$

- ha az egyik pont $\mathcal{O}$, akkor fennáll: $P + \mathcal{O} = \mathcal{O} + P = P$

Elliptikus görbék

két pont összegét geometriailag a **húrmódszer** alapján adhatjuk meg:

- legyen P és Q két pont a görbén, ahol $P \neq Q$,
- **húzzunk egy egyenest** a P és Q pontokon keresztül, bebizonyítható, hogy az egyenes metszeni fogja a görbét egy harmadik pontban, amelynek az x tengelyre való szimmetrikusa lesz a P és Q összege:



<https://www.desmos.com/calculator/ialhd71we3>

ECDH - Elliptic Curve Diffie–Hellman key exchange

- feltételezve, hogy a két egység **A** és **B**, akkor a protokoll a következő:
 1. **A** és **B** megegyeznek az E elliptikus görbében, a p prímszám, a görbe egy P pontjának, és a P pont n rendjének értékében
 2. az **A** egység meghatározza:
 - $a \xleftarrow{R} \{2, \dots, n-1\}$ véletlenszerű, szigorúan titkos, **privát kulcs**,
 - $A = aP = (x_A, y_A)$ **publikus kulcs**,
 - a **A**-t elküldi **B**-nek,
 3. a **B** egység meghatározza:
 - $b \xleftarrow{R} \{2, \dots, n-1\}$ véletlenszerű, szigorúan titkos, **privát kulcs**,
 - $B = bP = (x_B, y_B)$ **publikus kulcs**,
 - a **B**-t elküldi **A**-nak,
 4. az **A** egység a közös $K = x_K$ kulcsot a következőképpen határozza meg:
$$aB = (x_K, y_K)$$
 5. a **B** egység a közös $K = x_K$ kulcsot a következőképpen határozza meg:
$$bA = (x_K, \hat{y}_K).$$

Helyesség:

$$aB = bA = abP.$$

ECDH - Elliptic Curve Diffie–Hellman key exchange

- görbétől, illetve implementációtól függően elégséges ha
 - a 2. lépésben **A** csak az x_A -t küldi el **B**-nek,
 - a 3. lépésben **B** csak az x_B -t küldi el **A**-nak,
 - a E görbe egyenlete alapján meg lehet határozni a következő értékeket:

$$z_A = y_A^2, \quad z_B = y_B^2$$

- ha a p értékére igaz, hogy $p + 1$ néggyel való osztási maradéka nulla, azaz $p + 1 \equiv 0 \pmod{4}$, négyzetes maradékot lehet számolni:

$$y_A = z_A^{(p+1)/4} \pmod{p}, \quad y_B = z_B^{(p+1)/4} \pmod{p}$$

- az y_K és $\hat{y}_K$ értékek lehet, hogy nem egyeznek meg, de ez nem jelent problémát, mert az x_K értékek mindig egyformák lesznek,
- az x_K mellett elküldenek egy paritás bitet, mert az y értékek közül az egyik mindig páros, a másik páratlan,
- a Montgomery, Edwards görbék esetében **projektív koordinátákkal** dolgoznak, ekkor sem küldik el az y értékeket.

ECDH - Elliptic Curve Diffie–Hellman key exchange

- Legyen $E : y^2 \equiv x^3 - 3x + 1 \pmod{11}$, $p = 11$, $P = (4, 8)$,
- a görbe pontjai:

$2 \cdot P = (7, 9)$	$3 \cdot P = (5, 10)$	$4 \cdot P = (6, 10)$	$5 \cdot P = (2, 5)$
$6 \cdot P = (10, 5)$	$7 \cdot P = (0, 1)$	$8 \cdot P = (8, 7)$	$9 \cdot P = (8, 4)$
$10 \cdot P = (0, 10)$	$11 \cdot P = (10, 6)$	$12 \cdot P = (2, 6)$	$13 \cdot P = (6, 1)$
$14 \cdot P = (5, 1)$	$15 \cdot P = (7, 2)$	$16 \cdot P = (4, 3)$	$17 \cdot P = \mathcal{O}$

- az **A** egység:
 - választja $a = 15$ -t $\rightarrow A = a \cdot P = (7, 2)$,
 - elküldi **B**-nek az $x_A = 7$ értéket.
- a **B** egység:
 - választja $b = 6$ -t $\rightarrow B = b \cdot P = (10, 5)$,
 - elküldi **A**-nak a $x_B = 10$ értéket.
- A**:
 - $z_B = (10^3 - 3 \cdot 10 + 1) \pmod{11} = 3$, $y_B = 3^{(11+1)/4} = 5$
 - $aB = 15 \cdot (10, 5) = (2, 5)$,
- B**:
 - $z_A = (7^3 - 3 \cdot 7 + 1) \pmod{11} = 4$, $y_A = 4^{(11+1)/4} = 9$
 - $bA = 6 \cdot (7, 9) = (2, 6)$,
- a közös kulcs: $K = 2$.

A P256 görbe

- más néven **secp256r1**
- az egyik legnépszerűbb görbe
- az elnevezés eredete:
 - **sec**: Standards for Efficient Cryptography,
 - **p256**: a p prímszám 256 bites,
 - **r**: random görbe
- **Weierstrass** típusú görbe: $y^2 \equiv x^3 + ax + b \pmod{p}$, ahol p prímszám:

$$p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$$

$q = 115792089210356248762697446949407573529996955224135760342422259061068512044369$
 $a = -3 = 115792089210356248762697446949407573530086143415290314195533631308867097853948$
 $b = 41058363725152142129326129780047268409114441015993725554835256314039467401291$
 $x_1 = 48439561293906451759052585252797914202762949526041747995844080717082404635286$
 $y_1 = 36134250956749795798585127919587881956611106672985015071877198253568414405109$

A secp256k1 görbe

- **Weierstrass** típusú görbe: $y^2 \equiv x^3 + ax + b \pmod{p}$, ahol p prímszám:

$$p = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$$

- Koblitz görbének hívják: hatékonyan implementálható az αP művelet,
- a generátor $P(x_1, y_1)$ pont rendje az n prímszám,
- a görbe pontjainak a száma megegyezik n -nel,
- a **bitcoin** ezt a görbét használja, nem bízott meg a NIST secp256r1 görbájében,

$q = 115792089237316195423570985008687907852837564279074904382605163141518161494337$

$a = 0$

$b = 7$

$x_1 = 55066263022277343669578718895168534326250603453777594175500187360389116729240$

$y_1 = 32670510020758816978083085130507043184471273380659243275938904335757337482424$

A Curve25519 görbe

- Dan Bernstein tervezte,
- népszerűsége **biztonsága és gyorsasága** miatt növekvőben van,
- **Montgomery** típusú görbe: $y^2 \equiv x^3 + ax^2 + x \pmod{p}$, ahol

$$p = 2^{255} - 19$$

az a legnagyobb prímszám, amely kisebb mint 2^{255} ,

- a következő Edwards görbévé alakítható át:

$$x^2 + y^2 = 1 + (121665/121666)x^2y^2$$

- az a érték kiválasztása: a lehető legkisebb érték kell legyen, amelyre az ECDL probléma még nehéz,
- a generátor $P(x_1, y_1)$ pont rendje az n prímszám,
- a görbe pontjainak száma: $8n$.

$$a = 486662$$

$$q = 2^{252} + 2774231777372353535851937790883648493$$

$$x_1 = 9$$

$$y_1 = 14781619447589544791020593568409986887264606134616475288964881837755586237401$$

Lineáris kongruenciák

Az $a \cdot x \equiv b \pmod{m}$ típusú egyenletet lineáris kongruenciának hívjuk, ahol x ismeretlen.

- ha x_0 megoldása a fenti kongruenciának és $x_1 \equiv x_0 \pmod{m}$, akkor x_1 is megoldás

1. tétel

Legyenek a, b, m egész számok, ahol $m > 0$ és $(a, m) = d$. Ha $d \nmid b$, akkor az $a \cdot x \equiv b \pmod{m}$ kongruenciának **nincs megoldása**. Ha $d \mid b$, akkor az $a \cdot x \equiv b \pmod{m}$ kongruenciának **d inkongruens megoldása van** $\pmod{m}$ szerint.

- Az $a \cdot x \equiv b \pmod{m}$ kongruencia ekvivalens az $a \cdot x - m \cdot y = b$ egyenlettel.
- **kiterjesztett euklideszi** algoritlussal meghatározzuk a $\hat{x}, \hat{y}$ -t, úgy hogy teljesüljön: $a \cdot \hat{x} + m \cdot \hat{y} = d$.
- A kongruencia első megoldása $x_0 = \hat{x} \cdot (b/d)$ lesz.
- A kongruencia többi megoldásait az $n = 1, 2, \dots, d - 1$ lehetséges értékek alapján, a következőképpen számoljuk ki: $x_i = x_0 + n \cdot (m/d)$.

Példák

Határozzuk meg a $9 \cdot x \equiv 12 \pmod{15}$ kongruencia megoldásait.

- $(9, 15) = 3$ és $3 \mid 12 \Rightarrow$ a kongruenciának 3 inkongruens megoldása van $\pmod{15}$ szerint.
- Kiterjesztett euklideszi algoritmussal kapjuk: $9 \cdot 2 + 15 \cdot (-1) = 3$, azaz $\hat{x} = 2$, $\hat{y} = -1$.
- A kongruencia első megoldása: $x_0 = 2 \cdot (12/3) = 8$.
- A többi megoldás:
 - $x_1 = x_0 + 1 \cdot (15/3) = 8 + 5 = 13 \pmod{15}$,
 - $x_2 = x_0 + 2 \cdot (15/3) = 8 + 10 = 18 \equiv 3 \pmod{15}$.
- fennáll:
 - $9 \cdot 8 = 72 \equiv 12 \pmod{15}$,
 - $9 \cdot 13 = 117 \equiv 12 \pmod{15}$,
 - $9 \cdot 3 = 27 \equiv 12 \pmod{15}$,

Multiplikatív inverz

2. értelmezés

Az $a \cdot x \equiv 1 \pmod{m}$ kongruenciának az x megoldását, ahol $(a, m) = 1$ az a szám moduláris multiplikatív inverzének, vagy **multiplikatív inverzének**, vagy csak egyszerűen inverzének hívjuk.

Megjegyzés

- a fenti egyenlet egy sajátos esete az $a \cdot x \equiv b \pmod{m}$ egyenletnek: $b = 1$.

Példák:

- Mennyi lesz 4 inverze $(\text{mod } 7)$ szerint?, másképp fogalmazva: Mivel kell megszorozni 4-et, hogy 1-et kapjunk?
 - Válasz: 2-vel, mert $4 \cdot 2 = 1 \pmod{7}$.
- Mennyi lesz 7 inverze $(\text{mod } 31)$ szerint?, másképp fogalmazva: Mivel kell megszorozni 7-et, hogy 1-et kapjunk?
 - Válasz: 9-cel, mert $7 \cdot 9 = 1 \pmod{31}$.

Multiplikatív inverz

Egy szám multiplikatív inverzét többféleképpen is meghatározhatjuk:

- brute-force módszerrel: az összes lehetséges értéket sorra próbáljuk
- a **kiterjesztett euklideszi** algoritmussal: a gyakorlatban ezt használják
- az Euler tétel segítségével, ha $(a, m) = 1$ az $a \cdot x \equiv 1 \pmod{m}$ kongruenciának a megoldása a következőképpen adható meg:
 - $x = a^{\phi(m)-1} \pmod{m}$
 - ha m prímszám, akkor $x = a^{m-2} \pmod{m}$

Határozzuk meg a $13 \cdot x \equiv 4 \pmod{36}$ kongruencia egy megoldását, az **Euler-tétel segítségével**:

- $(13, 36) = 1 \Rightarrow$ a kongruenciának egy megoldása van,
- meghatározzuk:
 $\phi(36) = \phi(2^2 \cdot 3^2) = \phi(2^2) \cdot \phi(3^2) = (2^2 - 2^1) \cdot (3^2 - 3^1) = 2 \cdot 6 = 12$
- 13 inverze $\pmod{36}$ szerint: $13^{\phi(36)-1} = 13^{11} = 25 \pmod{36}$
- a kongruencia megoldása: $25 \cdot 4 = 28 \pmod{36}$.

Multiplikatív inverz

1. feladat

A *kiterjesztett euklideszi* algoritmussal határozzuk meg a inverzét $(\text{mod } m)$ szerint.

```
def inverz(a, m):  
    x0, x1 = 1, 0  
    b = m  
    while True:  
        q = a // b  
        r = a - b * q  
        if r == 0:  
            if b != 1: return -1  
            else: return x1 % m  
        x = x0 - q * x1  
        x0, x1 = x1, x  
        a, b = b, r
```

```
>>> inverz(13, 36)
```

25

ellenőrzés:

```
>>> (13 * 25) % 36
```

1

Multiplikatív inverz

Megjegyzések:

- az inverz függvényben egy kicsit másképp jár el, mint a kiterjesztett euklideszi algoritmus:
 - az y változóval végzett műveleteket elhagytuk, mert azok most fölöslegesek,
 - módosult a return-hoz tartozó műveletsor: csak egy értéket ad vissza a függvény, a meghatározott inverzet
- a kiterjesztett euklideszi algoritmus az x és y értékekben **negatív számot** is meghatározhat,
- azt szeretnénk hogy az inverz mindig pozitív érték legyen, ezért a return-ben a $x1 \% m$ műveletet alkalmaztuk,
- a Python $\%$ operátora mindig a **legkisebb pozitív maradékot** határozza meg:

```
>>> -5 % 7
2
```
- az inverz meghatározása Python-ban, **-1-es** kitevőt kell használnunk:

```
>>> pow(13, -1, 36)
25
```

Az RSA rendszer

- 1977-ban publikálták a szerzők: Rivest, Shamir és Adleman, biztonsága többek között a **faktorizációs problémán** alapszik,

3. értelmezés

*Az egész számok $\{1, 2, \dots, n\}$ véges halmaza esetében, ahol n összetett szám, pontosabban két prímszám szorzata a **faktorizációs probléma** azt jelenti, hogy megkeressük azt a két p és q prímszámot, amelyekre fennáll: $n = p \cdot q$.*

- nagy számok esetében a faktorizációs problémára **nem ismert hatékony** algoritmus, jelenleg **minimum 512 bites** prímszámokat használnak,
- két távoli egység nyilvános csatornán történő kis méretű, titkosított információcseréjére (**kulcscseréjére**), illetve az egységektől származó adatok **hitelesítésére** alkalmas,
- a TLS 1.3 protokollban digitális aláírás előállítására használják, 2018 előtt kulcscserére is használták, hasonlóan a Diffie-Hellman kulcscseréhez
- **digitális aláírás**: adott eszköztől származó adatok hitelesítésére alkalmas kriptográfiai eljárás

Az RSA rendszer

- a gyakorlatban az RSA-OAEP, vagy RSA-PSS rendszereket használják, amelyek az RSA módosított, biztonságos és standardizált változatai: **PKCS#1 v2.1**
- RSA-OAEP: kulcscserét, másképp fogalmazva aszimmetrikus titkosítást biztosít
- RSA-PSS: digitális aláírás (**signature**) használatát, másképp fogalmazva digitális tanusítványok (**certificate**) kibocsátását biztosítja,
- mindkettőt Bellare és Rogaway dolgozták ki az 1990-es évek közepén,
- az előadás keretén belül **nem lesz szó az RSA-OAEP, RSA-PSS** rendszerről, az 1977-es változat kerül bemutatásra, amelyet RSA-textbook, vagy baby-RSA néven említ a szakirodalom,
- a kulcsgeneráló algoritmus egy-egy értékpárt, pontosabban egy-egy kulcspárt határoz meg, ahol az egyik értékpárt **publikus kulcsnak**, a másik értékpárt **privát kulcsnak** hívjuk
- **aszimmetrikus, publikus kulcsú kriptográfia**: a kriptográfiának az a területe, amely a kulcscserék és digitális aláírások biztonságával foglalkozik

Publikus kulcsú titkosítás

- publikus kulcsú **titkosítás**:
 - feltételezve, hogy a kommunikációban résztvevő két egység **A** és **B**, akkor a protokoll első lépéseként **A**, vagy egy harmadik megbízható egység végrehajtja a **kulcsgeneráló algoritmust**, majd a publikus kulcsot egy nyilvános csatornán megosztja a **B** egységgel,
 - ha egy harmadik egység végzi a kulcsgenerálást, akkor a privát kulcsot egy titkos csatornán megosztja **A**-val,
 - **B** véletlenszerűen generál egy $1 < K < n$ egész számot, és a **titkosító algoritmust**, illetve a publikus kulccsal meghatároz egy cK értéket, a cK -t elküldi egy nyilvános csatornán **A**-nak,
 - **A** a **visszafejtő algoritmust**, illetve a privát kulccsal meghatározza a **K** értéket,
 - a megosztott titkos információ tehát a **K** lesz,
- az RSA esetében a rendszer helyessége az **Euler-tétellel** bizonyítható.