Kriptográfia és Információbiztonság
7. előadás

MÁRTON Gyöngyvér

Sapientia Egyetem, Matematika-Informatika Tanszék
Marosvásárhely, Románia
mgyongyi@ms.sapientia.ro

2024

Miről volt szó az elmúlt előadáson?

- publikus kulcsú rendszerek, alapfogalmak
- matematika modellek,
- publikus kulcsú titkosítók: RSA, RSA-OAEP
- digitális aláírások: matematikai modell
- digitális aláírások: RSA, RSA-RSS

Miről lesz szó?

- a diszkrét logaritmus (DL) probléma, a DL feltételezés
- hatványok és generátor elemek
- a Diffie-Hellman kulcscsere
- DL problémán alapuló digitális aláírások:
  - az ElGamal digitális aláírás
  - a DSA (Digital Signature Algorithm) vagy DSS (Digital Signature Standard)
- elliptikus görbéken alapuló kriptográfia

A diszkrét logaritmus feltételezés

Számos kriptorendszer biztonsága alapszik a diszkrét logaritmus feltételezésen, azaz a diszkrét logaritmus probléma (DL probléma) nehézségén:

1. értelmezés

Az egész számok $\mathbb{Z}_p^*$ multiplikatív csoportja esetében, ahol p prímszám a DL probléma a következő: az A, g-alapú diszkrét logaritmusa $\pmod p$ szerint azt jelenti, hogy megkeressük azt az a pozitív egész számot, melyre fennáll:

$$g^a \equiv A \pmod p,$$

ahol g generátor elem (primitív gyök), és $g, A \in \mathbb{Z}_p^*$.

- a g szám generátor elem $\pmod p$ szerint, ha g hatványai 1-től, $\phi(p)$-ig, azaz g, g^2, g^3, $\ldots$, $g^{\phi(p)}$, különböző maradékot adnak $\pmod p$ szerint
- a diszkrét logaritmus feltételezés azt jelenti, hogy megfelelő paraméterválasztás esetén (pl. a p prímszám legalább 1024 bites) nincs hatékony, polinomiális futási idejű algoritmus a DL problémára

Hatványok és generátor elemek

Határozzuk meg $x^n \pmod{11}$, értékeit $x = 2, 3, \ldots 10$-re illetve $n = 1, 2, \ldots 10$-re:

$2^1 = 2$	$3^1 = 3$	$4^1 = 4$	$5^1 = 5$	$6^1 = 6$	$7^1 = 7$	$8^1 = 8$	$9^1 = 9$	$10^1 = 10$
$2^2 = 4$	$3^2 = 9$	$4^2 = 5$	$5^2 = 3$	$6^2 = 3$	$7^2 = 5$	$8^2 = 9$	$9^2 = 4$	$10^2 = 1$
$2^3 = 8$	$3^3 = 5$	$4^3 = 9$	$5^3 = 4$	$6^3 = 7$	$7^3 = 2$	$8^3 = 6$	$9^3 = 3$	$10^3 = 10$
$2^4 = 5$	$3^4 = 4$	$4^4 = 3$	$5^4 = 9$	$6^4 = 9$	$7^4 = 3$	$8^4 = 4$	$9^4 = 5$	$10^4 = 1$
$2^5 = 10$	$3^5 = 1$	$4^5 = 1$	$5^5 = 1$	$6^5 = 10$	$7^5 = 10$	$8^5 = 10$	$9^5 = 1$	$10^5 = 10$
$2^6 = 9$	$3^6 = 3$	$4^6 = 4$	$5^6 = 5$	$6^6 = 5$	$7^6 = 4$	$8^6 = 3$	$9^6 = 9$	$10^6 = 1$
$2^7 = 7$	$3^7 = 9$	$4^7 = 5$	$5^7 = 3$	$6^7 = 8$	$7^7 = 6$	$8^7 = 2$	$9^7 = 4$	$10^7 = 10$
$2^8 = 3$	$3^8 = 5$	$4^8 = 9$	$5^8 = 4$	$6^8 = 4$	$7^8 = 9$	$8^8 = 5$	$9^8 = 3$	$10^8 = 1$
$2^9 = 6$	$3^9 = 4$	$4^9 = 3$	$5^9 = 9$	$6^9 = 2$	$7^9 = 8$	$8^9 = 7$	$9^9 = 5$	$10^9 = 10$
$2^{10} = 1$	$3^{10} = 1$	$4^{10} = 1$	$5^{10} = 1$	$6^{10} = 1$	$7^{10} = 1$	$8^{10} = 1$	$9^{10} = 1$	$10^{10} = 1$

$\pmod{11}$ szerint

- 2, 6, 7, 8 generátor elemek
- 1, 3, 4, 5, 9, 10 nem generátor elemek
- $p = 11$ prímszám esetében a generátor elemek száma: $\phi(p-1) = \phi(10) = 4$

Hatványok és generátor elemek

- kis Fermat tétel, kimondja, hogy ha p prím és $(x, p) = 1$, akkor $x^{p-1} = 1 \pmod{p}$,
- a táblázat alapján kijelenthetjük, hogy lesznek olyan x számok, ahol x-nek kisebb hatványértékei is kongruensek lesznek 1-el.
- x rend-jén, $\pmod{p}$ szerint, azt a legkisebb k kitevőt értjük, amelyre fennáll: $x^k \equiv 1 \pmod{p}$. Pl. $\pmod{11}$ szerint a 2-es szám rendje 10, a 3 szám rendje 5, míg a 10-es szám rendje 2,
- ha p egy prímszám, akkor mindig létezik $g \in \{1, 2, \ldots, p-1\}$, amelynek hatványértékei $\pmod{p}$ szerint előállítják az $\{1, 2, \ldots, p-1\}$ halmaz elemeit egy tetszőleges sorrendbe, ezeket a g elemeket primitív gyököknek, vagy generátor elemeknek hívjuk,
- a generátor elemek rendje $p-1$, számuk $\phi(p-1)$.

Hatványok és generátor elemek

Megjegyzések:

- fontos feladat, hogy egy adott prímszám esetében meghatározunk egy generátor elemet
- biztonságos prímeknek (safe prime) hívjuk azokat a p prímszámokat, amelyek felírhatóak a $2 \cdot q + 1$ alakba, ahol q is prímszám
- egy biztonságos prím meghatározása azonban jóval időigényesebb, mint egy "közönséges" prím kiválasztása
- ha a $p = 2 \cdot q + 1$ biztonságos prím, akkor egy generátor elem meghatározása nem számít nehéz feladatnak, a következőképpen történik:
  - ha a g egész számra fennáll, hogy $g^q \not\equiv 1 \pmod{p}$, és $g \not\equiv \pm 1 \pmod{p}$, akkor g generátor elem $\pmod{p}$ szerint
- ha $p = 2 \cdot q + 1$ biztonságos prím, akkor a generátor elemek száma $\phi(p-1) = \phi(2) \cdot \phi(q) = q - 1$

Hatványok és generátor elemek

Adott generátor elem meghatározása, példa:

- legyen $p = 47 = 2 \cdot 23 + 1$, $p - 1 = 46$,
  - $g = 13$ generátor elem?
  - Igen, mert $13^{23} = 46 \pmod{47}$.
  - $g = 3$ generátor elem?
  - Nem, mert $3^{23} = 1 \pmod{47}$.

Nem triviális algoritmusok a DL problémára:

- a Shank (baby-step giant-step) algoritmus,
- a Pohlig-Hellman algoritmus,
- az indexkalkulus algoritmus.

Diffie-Hellman kulcscsere

- 1976-ban publikálták a szerzők, hatalmas áttörést jelentett az számítógép biztonságban
- két távoli egység (számítógép, mobileszköz, stb.) kulcscsere mechanizmusára ad megoldást
- az eredeti kulcscsre protokoll az egész számok multiplikatív csoportjában működik, de megadható tetszőleges ciklikus csoport esetében is, például elliptikus görbék esetében
- a rendszer biztonsága a választott publikus paraméterek nagyságrendjétől függ, illetve a diszkrét logaritmus feltételezésen alapszik
- a publikus paraméterek: egy k bites p prímszám és egy g generátor eleme, ahol k legalább 1024 bit kell legyen
- áttérés figyelhető az elliptikus görbéken alapuló kriptográfiára

Diffie-Hellman kulcscsere

Feltételezve, hogy a kommunikációban résztvevő két eszköz **A** és **B**, akkor a protokoll a következő:

1. **A** és **B** megegyeznek a p, k-bites prímszám, és a g generátor elem értékében, ezek publikus adatok
2. az **A** eszköz a k, p, g ismeretében meghatározza az a, A értékeket, ahol:
   - $a \in \{2, \ldots, p-2\}$ véletlen szám, ez lesz az **A** privát kulcsa, amelyet titokban kell tartania,
   - $A = g^a \pmod{p}$, ez lesz az **A** publikus kulcsa, amelyet elküld **B**-nek.
3. a **B** eszköz a k, p, g ismeretében meghatározza a b, B értékeket, ahol:
   - $b \in \{2, \ldots, p-2\}$ véletlen szám, ez lesz a **B** privát kulcsa, amelyet titokban kell tartania,
   - $B = g^b \pmod{p}$, ez lesz az **B** publikus kulcsa, amelyet elküld **A**-nak.
4. az **A** eszköz a közös K kulcsot a kövekezőképpen határozza meg:
$$K = B^a \pmod{p}.$$
5. a **B** eszköz a közös K kulcsot a kövekezőképpen határozza meg:
$$K = A^b \pmod{p}.$$

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

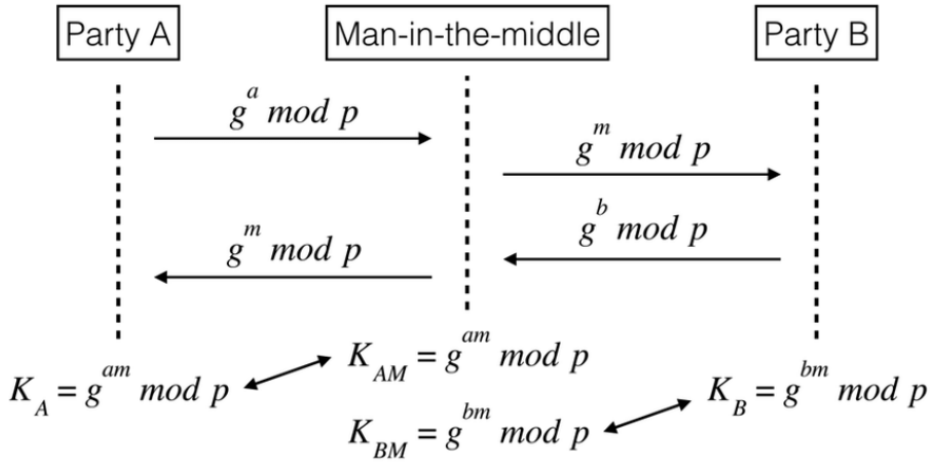
Diffie-Hellman kulcscsere, példa

- Legyen $p = 47, g = 13$ publikus paraméterek,
  - az **A** eszköz:
    - választja $a = 12$-t $\rightarrow A = g^a \pmod p = 9 \pmod{47}$,
    - elküldi Bobnak az A = 9 értéket.
  - a **B** eszköz:
    - választja $b = 34$-t $\rightarrow B = g^b \pmod p = 21 \pmod{47}$,
    - elküldi Alicenak a B = 21 értéket.
  - **A**: $K = B^a \pmod p = 21^{12} = 16 \pmod{47}$,
  - **B**: $K = A^b \pmod p = 9^{34} = 16 \pmod{47}$.
- a közös kulcs: $K = 16$.

A Diffie-Hellman kulcscsere biztonsága

- ha a kulcscsere protokollban a résztvevő felek nem tudnak meggyőződni egymás kiléte felöl, azaz a publikucs kulcsok nincsenek hitelesítve, akkor a rendszer nem lesz biztonságos, kivitelezhető a man in the meedle támadás:
  - egy **M** támadó kiadja magát **A**-nak, mint **B**, illetve kiadja magát **B**-nek, mint **A**,
  - **M** választ két random számot: m_1, m_2,
  - **A** irányába:
    - elküldi **A**-nak $M_2 = g^{m_2}$-t,
    - a közös kulcs **A**-val: $K_A = A^{m_2} = M_2^a = g^{a \cdot m_2}$,
  - **B** irányába:
    - elküldi **B**-nek $M_1 = g^{m_1}$-t,
    - a közös kulcs **B**-vel: $K_B = B^{m_1} = M_1^b = g^{m_1 \cdot b}$,
- minden kulcscsere protokoll támadható Man-in-the-Middle módszerrel, ha a protokoll résztvevői nem hitelesítik egymás publikus kulcsait, azaz nem győződnek meg egymás kiléte felől

Man-in-the-Middle támadás



- a támadó, a támadást azzal indítja, hogy meghatározza a mindkét fél irányába szükséges privát és publikus értékeket (kulcsokat)
- a publikus kulcsok hitelesítése:
  - digitális aláírással történik,
  - a protokolltól függ, hogy ezt hogyan valósítják meg a kommunikáló felek,
  - a TLS/SSL-ben egy központi hatóság (central authority) igénybevételével oldják meg, akinek tanúsítványok (certificate) kibocsájtására van joga

DL problémán alapuló digitális aláírások

- az egész számok $\mathbb{Z}_p^*$ multiplikatív csoportja esetében értelmezett diszkrét logaritmus feltételezésen alapszik:
  - az ElGamal digitális aláírás
  - a DSA digitális aláírás
- a digitális aláírások biztosítják az üzenetek integritását, hitelességét, és eredetét, de nem titkosítják az üzenetet
- az üzenethitelesítő kódok (MAC) biztosítják az átküldött üzenet sértetlenségét, de nem biztosítanak eredetszolgáltatást, azaz a küldő bármikor letagadhatja, hogy üzenetet küldött a vevőnek
- a digitális aláírások az üzenet eredetére vonatkozó bizonyítékot biztosítanak, amelyet csak a küldő állíthat elő; az üzenet integritását, eredetét, hitelességét a rendszer bármelyik résztvevője ellenőrizheti

Az ElGamal digitális aláírás

- biztonsága a diszkrét logaritmus feltételezésen alapszik
- 1985-ben publikálta Taher ElGamal
- a gyakorlatban ritkán használják, több variánsa is ismert: a DSA standardként van elfogadva

3 algoritmussal értelmezhető:

- Gen, a kulcs-generáló algoritmus: $(p, g, a) \xleftarrow{R} Gen(1^k)$, ahol
  - p-prímszám, g-primitív gyöke,
  - $A = g^a \pmod p$,
  - $p_k = (p, g, A), s_k = (a)$,
- $Aut_{(a)}(m) \to ((B, c), m)$ a hitelesítő algoritmus:
  - $b \xleftarrow{R} \{1, \ldots, p-1\}$, $B = g^b \pmod p$,
  - $c \leftarrow ((h - a \cdot B) \cdot b_1) \pmod{p-1}$, ahol
  - $b_1 \cdot b = 1 \pmod{p-1}$, és $h \leftarrow hash(m)$,
- $Ver_{(p,g,A)}((B, c), m)$ az ellenőrző algoritmus:
  - $h_1 \leftarrow g^h \pmod p$, ahol $h \leftarrow hash(m)$,
  - $h_2 \leftarrow (B^c \cdot A^B) \pmod p$,
  - ha $h_1 = h_2$, akkor az aláírás hiteles, ellenkező esetben nem.

A DSA (Digital Signatures Algorithm)

- biztonsága a diszkrét logaritmus feltételezésen alapszik
- a Schnorr, illetve ElGamal alírási sémák egy változata
- a NIST 1991-ben standardként javasolta, a jelenlegi 2023-ban megjelenő standard leírása: *link*
- megjelenése óta számos kritika érte: nem volt publikus a NIST kiírása, az elején korlátozták a p értékét 512-re

3 algoritmussal értelmezhető:

- *Gen*, a kulcs-generáló algoritmus: $(q, p, g, a, A) \xleftarrow{R} Gen(1^k)$, ahol
  - q-prímszám: $2^{159} < q < 2^{160}$,
  - p-prímszám: $2^{511+64t} < p < 2^{512+64t}$, $t = 1, \ldots, 8$,
  - q osztja $(p-1)$-t,
  - legyen g generátor elem: $g = x^{(p-1)/q} \pmod p$, ahol $x \xleftarrow{R} \{2, \ldots, p-1\}$
  - $a \xleftarrow{R} \{2, \ldots, p-1\}$, és g rendje q,
  - $A = g^a \pmod p$,
  - $p_k = (q, p, g, A), s_k = (a)$.

A DSA (Digital Signatures Algorithm)

- $Aut_{(a)}(m) \to ((B, c), m)$ a hitelesítő algoritmus:
  - $b \xleftarrow{R} \{1, \ldots, q-1\}$, $B = g^b \pmod p \pmod q$,
  - meghatározza b_1-t, úgy hogy $b_1 \cdot b = 1 \pmod q$, és $h \leftarrow hash(m)$,
  - $c \leftarrow ((h + a \cdot B) \cdot b_1) \pmod q$.
- $Ver_{(p,g,A)}((B, c), m)$ az ellenőrző algoritmus:
  - meghatározza c_1-t úgy hogy $c_1 \cdot c = 1 \pmod q$, és $h \leftarrow hash(m)$,
  - meghatározza $\hat{B}$-t:
    $$\hat{B} \leftarrow (g^{c_1 \cdot h \pmod q} \cdot A^{(c_1 \cdot B) \pmod q}) \pmod p \pmod q,$$
  - ha $\hat{B} = B$, akkor az aláírás hiteles, ellenkező esetben nem.

A digitális aláírások biztonsága

Támadási célok

- teljes feltörés (total break): a támadónak sikerül meghatároznia a titkos kulcsot,
- szelektív hamisítás (selective forgery): a támadó nem elhanyagolható valószínűséggel meg tudja határozni egy adott üzenet hiteles aláírását,
- egzisztenciális hamisítás (existential forgery): a támadó nem elhanyagolható valószínűséggel meg tudja határozni egy tetszőleges üzenet hiteles aláírását.

Megjegyzések:

- a támadó erőforrásairól azt feltételezzük, hogy polinomiálisan korlátozottak
- gyakorlatban a digitális aláírást nem az üzenetre, hanem annak egy hash értékére határozom meg, ahol a hash érték meghatározásához standard függvényeket kell használni: SHA1, SHA2, SHA3, ez utóbbi 2014 óta standard
- ha az alkalmazott hash függvény nem erősen ütközésmentes, akkor is fennáll az egzisztenciális hamisítás lehetősége

A digitális aláírások biztonsága

- p véletlenszerűen választott prím kell legyen,
- minden egyes aláíráshoz, más és más b értéket kell generálni,
- ha nem az üzenet hash értékére határozom meg a digitális aláírást, akkor lehetséges az egzisztenciális hamisítás, pl az ElGamal esetében:
  - hash nélkül fennáll: $(B^c \cdot A^B) = g^m \pmod{p}$,
  - a támadó tud szerkeszteni 3 olyan $\hat{B}, \hat{c}, \hat{m}$ értéket, melyekre teljesül a fenti egyenlőség:
    - $\hat{B} = g^u \cdot A^v \pmod{p}$, ahol $u, v \xleftarrow{R} \{1, \ldots p-1\}$ és $gcd(v, p-1) = 1$
    - $\hat{c} = -\hat{B} \cdot v^{-1} \pmod{p-1}$,
    - $\hat{m} = \hat{c} \cdot u \pmod{p-1}$.

Elliptikus görbéken alapuló kriptográfia

- Diophantosz Aritmetika könyvsorozatában (13 kötet, ebből 6 maradt fenn) a következő probléma jelenik meg: határozzuk meg azokat az (x, y) racionális számpárokat, amelyek kielégítik az $y^2 = x^3 - x + 9$ egyenletet,
- egy elliptikus görbe általános alakja:

$$Ax^3 + Bx^2y + Cxy^2 + Dy^3 + Ex^2 + Fxy + Gy^2 + Hx + Iy + J = 0$$

- elliptikus görbékkel Koblitz és Miller az 1980-as évek végén kezdett el foglalkozni, eredményeiket az RSA-val ellentétben nem védték le,
- a kriptográfiában egyszerűbb alakkal dolgoznak, például a Weierstrass alak a következő:

$$y^2 = x^3 + ax + b,$$

- az első kripto standardok 2000-ben jelentek meg,
- a kis kulcsméret, a jó biztonság miatt nagy a népszerűségük, leggyakrabban 256 bites kulcsokkal dolgoznak.

Kriptográfiai kulcsok bitmérete

Table 10.3 Comparable Key Sizes in Terms of Computational Effort for Cryptanalysis

Symmetric Scheme (key size in bits)	ECC-Based Scheme (size of n in bits)	RSA/DSA (modulus size in bits)
56	112	512
80	160	1024
112	224	2048
128	256	3072
192	384	7680
256	512	15360

Source: Certicom

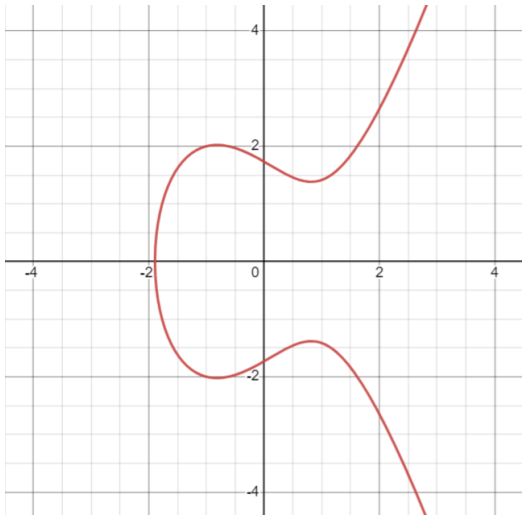
- a DL feltételezés az egész számok (mod p) szerinti multiplikatív csoportjában *már nem számít elég nehéznek*,
- 2019: egy 795 bites prímszám esetében megoldották a DL problémát, azóta legalább 2048 bites prímek használatát írják elő a DL feltételezésen alapuló protokollokban.

Elliptikus görbéken alapuló kriptográfia

- három típusú görbét szoktak használni,
- a valós számok felett értelmezett három típusú görbe grafikus alakja a következő:

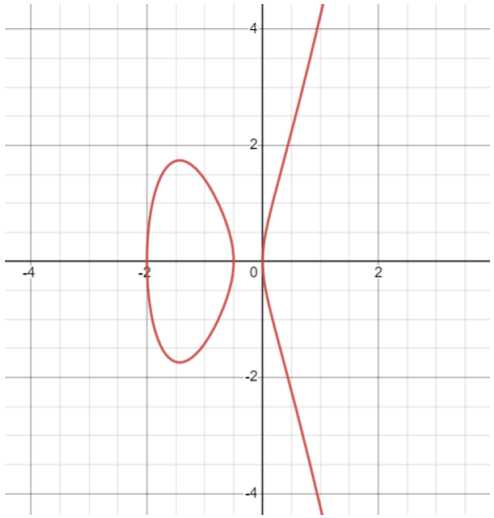
Weierstrass

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



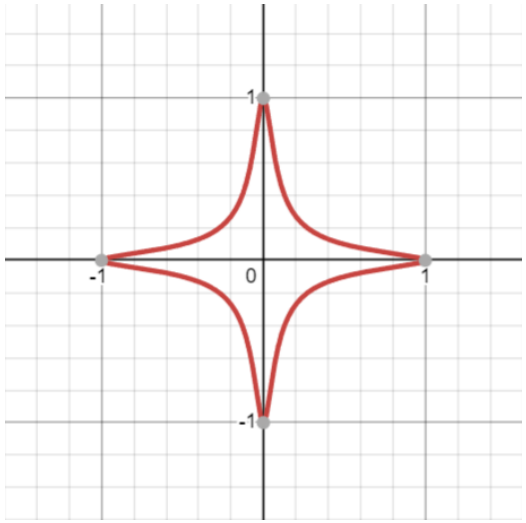
Montgomery

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



Edwards

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

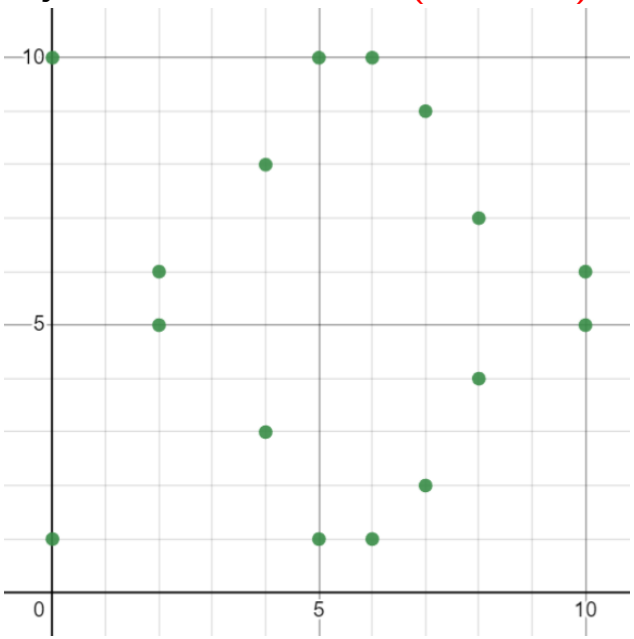


- különböző biztonságot nyújtanak,
- más hatékonyságot jelentenek,
- másképp van értelmezve az összeadás

Elliptikus görbéken alapuló kriptográfia

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

$y^2 \equiv x^3 - 3x + 1 \pmod{11}$



$(0,1), (0,11-1)$	$1^2 \equiv 0^3 - 3 \cdot 0 + 1$
	$10^2 \equiv 0^3 - 3 \cdot 0 + 1$
$(2,5), (2,11-5)$	$5^2 \equiv 2^3 - 3 \cdot 2 + 1$
	$6^2 \equiv 2^3 - 3 \cdot 2 + 1$
$(4,3), (4,11-3)$	$3^2 \equiv 4^3 - 3 \cdot 4 + 1$
	$8^2 \equiv 4^3 - 3 \cdot 4 + 1$
$(5,1), (5,11-1)$	$1^2 \equiv 5^3 - 3 \cdot 5 + 1$
	$10^2 \equiv 5^3 - 3 \cdot 5 + 1$
$(6,1), (6,11-1)$	$1^2 \equiv 6^3 - 3 \cdot 6 + 1$
	$10^2 \equiv 6^3 - 3 \cdot 6 + 1$
$(7,9), (7,11-9)$	$9^2 \equiv 7^3 - 3 \cdot 7 + 1$
	$2^2 \equiv 7^3 - 3 \cdot 7 + 1$
$(8,4), (8,11-4)$	$4^2 \equiv 8^3 - 3 \cdot 8 + 1$
	$7^2 \equiv 8^3 - 3 \cdot 8 + 1$
$(10,5), (10,11-5)$	$5^2 \equiv 10^3 - 3 \cdot 10 + 1$
	$6^2 \equiv 10^3 - 3 \cdot 10 + 1$

Elliptikus görbéken alapuló kriptográfia

- két pont összege: egy elliptikus görbe két pontja alapján könnyedén egy új pontot lehet meghatározni, amely szintén a görbén található,
- a számításokat valamely $\mathbb{F}_p$ véges test felett végzik:
  - $a, b \in \mathbb{F}_p$, ahol $p > 3$ prímszám,
  - $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$, biztosítja, hogy a görbe egyenletének nem lesz kétszeres gyöke,
  - $(x, y) \in \mathbb{F}_p$ a görbe egy pontja lesz, ha kielégíti a következő egyenletet:

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

- ahhoz hogy az összeadást értelmezni lehessen a görbén található pontokhoz hozzá vesznek egy speciális pontot, a végtelen pontot, amelyet $\mathcal{O}$-val jelölnek,
- az elliptikus görbét $E/\mathbb{F}_p$-vel jelölik,
- a görbe pontjai és az $\mathcal{O}$ pont által meghatározott halmazt $E(\mathbb{F}_p)$-vel jelölik:

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

- a számítások az $\mathbb{F}_{p^e}$, $e > 1$ bővített test felett is elvégezhetőek.

Elliptikus görbéken alapuló kriptográfia

- a görbe két pontján értelmezett összeadásra nézve az $E(\mathbb{F}_p)$ halmaz véges, ciklikus Abel csoport lesz:
  - megadható a semleges elem, az $\mathcal{O}$, amelyre fennáll: $P + \mathcal{O} = \mathcal{O} + P = P$, ahol P a görbe egy pontja,
  - 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, vagy $2 \cdot P$-vel, a $P + P + P$ műveletet $3P$-vel vagy $3 \cdot P$-vel jelöljük,
- az αP azt jelenti, hogy α-szor adtuk össze a P-t, ahol α tetszőleges pozitív egész szám,
- az αP hatékonyan meghatározható $2log_2\alpha$ csoport művelettel,
- ha az $E(\mathbb{F}_{p^e})$ halmaz elemszámát, azaz a görbe pontjainak számát N-el jelöljük, akkor megmutatható, hogy $N = p^e + 1 - t$, ahol t egész szám és $|t| \leq 2\sqrt{p^e}$,
- Schoof algoritmusával, (*wikiLink*) nagy p prímszám esetében is pontosan meg lehet határozni N-t.

Elliptikus görbéken alapuló kriptográfia

- ha a P pont rendje q, akkor $qP = \mathcal{O}$,
- G generátor elem, ha többszörösei mind különböznek és $E(\mathbb{F}_p)$ egyik pontját jelölik, vagy másképp fogalmazva a G pont rendje N lesz,
- az $E(\mathbb{F}_p)$ halmaznak mindig lesz egy generátor eleme, de nem minden pontja generátor elem,
- bármely N elemszámú, ciklikus, véges csoport esetén, ha N prímszám akkor a csoport bármely eleme, amely nem egyenlő a semleges elemmel generátor elem lesz,
- egy N elemszámú csoport estében N-nek minden egyes d osztója meghatároz egy N/d elemszámú alcsoportot,
- például ha N összetett szám és csak 2, 4, 8-al osztható, akkor az alcsoportok elemszáma $2, 4, 8, N/8, N/4, N/2$ vagy N lesz.

Elliptikus görbéken alapuló kriptográfia

- ha a P pont rendje q, akkor a P pont kofaktora h ahol $h = \frac{N}{q}$,
- ha d az N egy prímosztója, akkor az $E(\mathbb{F}_p)$-nek bármely P pontjára igaz, hogy az $\frac{N}{d} \cdot P$ pont vagy a semleges elem lesz, vagy egy olyan pont amelynek rendje d.
- a kriptográfiai protokollokban egy $E(\mathbb{F}_p)$ elliptikus görbének valamely q elemű ciklikus alcsoportját alkalmazzák, ahol q prímszám és kiválasztásra kerül egy P alap pont, a következőképpen:
  - kiválasztják a görbe egy tetszőleges Q pontját, és meghatározzák a $P = h \cdot Q$ értéket,
  - ha $P \neq \mathcal{O}$, akkor P lesz az alap pont, amelynek rendje tehát egyenlő q-val,
  - ha $P = \mathcal{O}$ akkor választanak egy új Q értéket.
- ha a kofaktor 1, akkor a teljes csoportban történnek a számítások.

Elliptikus görbéken alapuló kriptográfia

- 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 a ECDL probléma azt jelenti, hogy ismerve az $P, \alpha P$ értékeket határozzuk meg az $\alpha \in \mathbb{Z}_q$ pozitív egész számot,
- ECDL feltételezés: kellően nagy p és q értékek mellett nincs hatékony (polinomiális futási idejű) 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 alap ponttal dolgoznak.

Elliptikus görbe típusok

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

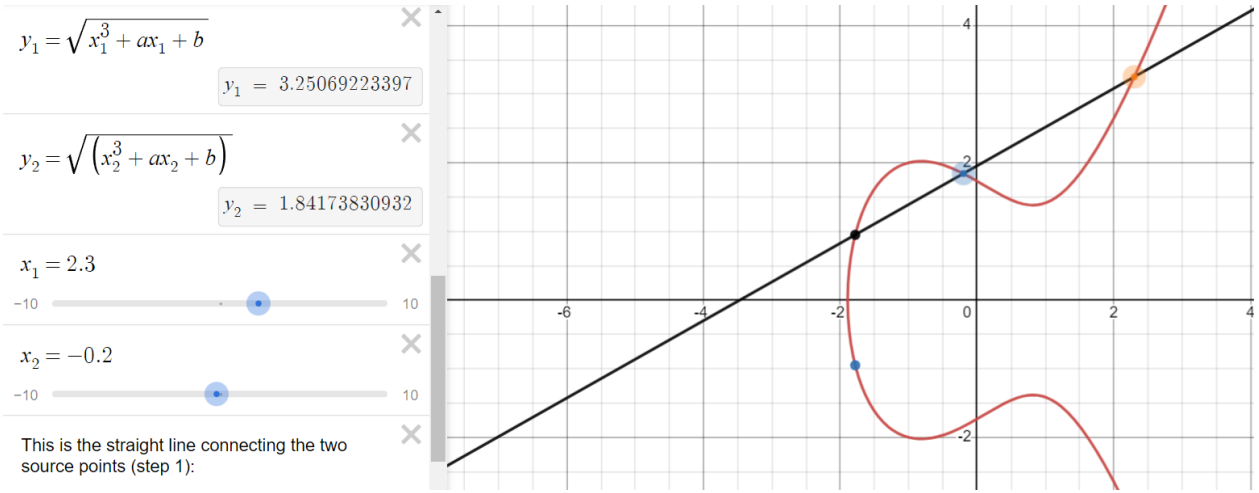
$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2 \\ y_3 &= \lambda(x_1 - x_3) - y_1 \\ \lambda &= \begin{cases} (y_2 - y_1) \cdot (x_2 - x_1)^{-1}, \text{ha } P \neq Q \\ (3x_1^2 + a) \cdot (2y_1)^{-1}, \text{ha } P = Q \end{cases} \end{aligned}$$

- ha az egyik pont $\mathcal{O}$, akkor fennáll: $P + \mathcal{O} = \mathcal{O} + P = P$
- $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$ feltétel biztosítja, hogy a görbe egyenletének nem lesz kétszeres gyöke,
- két pont összeadását geometriailag is meg szokták adni

Elliptikus görbe típusok

két pont összegének geometriai magyarázata, a húrmódszer:

- legyen P és Q két pont a görbén, ahol $P \neq Q$,
- húzzunk egy egyenest a P és Q pontokon keresztül,
- be bizonyítható, hogy az egyenes metszeni fogja a görbét egy harmadik pontban, amely a P és Q összege lesz,
- a húrmódszert többször is újrafelfedezték,
- Poincaré teremtette meg a kapcsolatot a húrmódszer, és a csoporton belüli összeadás között



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

Elliptikus görbe típusok

Montgomery görbe:

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

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

  és fennáll: $b(a^2 - 4) \not\equiv 0 \pmod{p}$
- átalakítható Weierstrass alakra, de a Weierstrass alak nem mindig hozható Montgomery alakra: $x = bu - a/3, y = bv$
- a számítások hatékonyabbak, mint a Weierstrass görbéken,
- a $P = (x_1, y_1)$ és $Q = (x_2, y_2)$ pontok összege a következőképpen van értelmezve:
  - ha $x_2 = x_1$ és $y_2 = -y_1$, akkor $P + Q = \mathcal{O}$
  - ha az egyik pont $\mathcal{O}$, akkor fennáll: $P + \mathcal{O} = \mathcal{O} + P = P$
  - másképp $P + Q = (x_3, y_3)$, ahol

$$\begin{aligned} x_3 &= b\lambda^2 - x_1 - x_2 - a \\ y_3 &= \lambda(x_1 - x_3) - y_1 \\ \lambda &= \begin{cases} (y_2 - y_1) \cdot (x_2 - x_1)^{-1}, \text{ha } P \neq Q \\ (3x_1^2 + 2ax_1 + 1) \cdot (2by_1)^{-1}, \text{ha } P = Q \end{cases} \end{aligned}$$

Elliptikus görbe típusok

Edwards görbe:

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

$$x^2 + y^2 \equiv 1 + dx^2y^2 \pmod{p}$$

  és fennáll: $d \in \mathbb{F}_p$ és $d \neq 0, 1$
- átalakítható Weierstrass alakra, de a Weierstrass alak nem mindig hozható Edwards alakra,
- az Edwards görbén is gyorsak a számítások
- az összeadás művelete egyszerűbb, a $P = (x_1, y_1)$ és $Q = (x_2, y_2)$ pontok összege a következőképpen van értelmezve:
  - ha $x_2 = x_1$ és $y_2 = -y_1$, akkor $P + Q = \mathcal{O}$
  - másképp $P + Q = (x_3, y_3)$, ahol

$$\begin{aligned} x_3 &= (x_1y_2 + x_2y_1)(1 + dx_1x_2y_1y_2)^{-1} \\ y_3 &= (y_1y_2 - x_1x_2)(1 - dx_1x_2y_1y_2)^{-1} \end{aligned}$$

Elliptikus görbe pontjainak meghatározása

Példa:

- legyen E egy elliptikus görbe, amelynek pontjait az $\mathcal{O}$ és a következő egyenlet megoldásai adják: $y^2 \equiv x^3 - 3x + 1 \pmod{11}$,
- a görbe pontjainak a meghatározása a következőképpen történik:
  - minden $x \in \mathbb{Z}_{11}$-re meghatározzuk az $z = x^3 - 3x + 1 \pmod{11}$ értéket,
  - mivel $11 \equiv 3 \pmod 4$ ezért $(p+1)/4$ egész szám lesz,
  - meghatározható a $t = z^{(p+1)/4} \pmod p$ hatványérték,
  - megvizsgáljuk milyen esetben lesz $t \cdot t$ egyenlő z-vel, azaz mikor lesz z négyzetes maradék,
  - a görbe pontjait azok az (x, t), $(x, p - t)$ értékek adják amelyek esetében $z \equiv t \cdot t \pmod{11}$.

Elliptikus görbe pontjainak meghatározása

Az előző oldalon megadott egyenleten végzett számítások a következők, ahol $z = x^3 - 3x + 1 \pmod{11}$ és $t = z^{(p+1)/4} \pmod{p}$:

x	z	t	$p - t$	$t \cdot t$	négyzetes maradék-e?
0	1	1	10	1	igen
1	10	10	1	1	nem
2	3	5	6	3	igen
3	8	6	5	3	nem
4	9	3	8	9	igen
5	1	1	10	1	igen
6	1	1	10	1	igen
7	4	9	2	4	igen
8	5	4	7	5	igen
9	10	10	1	1	nem
10	3	5	6	3	igen

- a görbe pontjai: $\mathcal{O}$, (0, 1), (0, 10), (2, 5), (2, 6), (4, 3), (4, 8), (5, 1), (5, 10), (6, 1), (6, 10), (7, 9), (7, 2), (8, 4), (8, 7), (10, 5), (10, 6)
- a pontok száma 17.

Elliptikus görbe pontjainak meghatározása

- tehát a $y^2 \equiv x^3 - 3x + 1 \pmod{11}$ görbének 17 pontja van,
- mivel a csoport elemszáma 17, ezért bármely elem a csoportból generátor elem is lesz, azaz bármely pont esetében, ha ismételten meghatározzuk önmagával az összegét, akkor sorra megkapjuk a görbe pontjait, és a 17-ik összeadás után visszakapjuk az eredeti pontot,
- legyen a kiinduló pont $P = (4, 8)$

$$
\begin{aligned}
2 \cdot P &= (7, 9) \\
3 \cdot P &= (5, 10) \\
4 \cdot P &= (6, 10) \\
5 \cdot P &= (2, 5) \\
6 \cdot P &= (10, 5) \\
7 \cdot P &= (0, 1) \\
8 \cdot P &= (8, 7) \\
9 \cdot P &= (8, 4)
\end{aligned}
$$

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

Elliptikus görbéken alapuló kriptográfia

A `P` és `Q` pontok összege az $y^2 \equiv x^3 + ax + b \pmod{p}$ görbe esetében:

```
def sumPoint(P, Q, p, a):
    (x1, y1) = P
    (x2, y2) = Q
    if Q == (None, None):
        return P
    if P == (None, None):
        return Q
    if x1 != x2:
        lamb = ((y2 - y1) * pow(x2 + p - x1, -1, p)) % p
        x3 = (lamb * lamb - x1 - x2) % p
        y3 = (lamb * (x1 - x3) - y1) % p
        return (x3, y3)
    else:
        if y1 == y2 and y1 != 0:
            lamb = ((3 * x1 * x1 + a) * pow(2 * y1, -1, p)) % p
            x3 = (lamb * lamb - 2*x1) % p
            y3 = (lamb * (x1 - x3) - y1) % p
            return (x3, y3)
        else:
            return (None, None)
```

Elliptikus görbéken alapuló kriptográfia

Az $alpha \cdot P$ érték meghatározása, Montgomery ladder technikával:

```
def multPoint(alpha, P, p, a):
    X = sumPoint(P, P, p, a)
    binAlpha = bin(alpha)[2:]
    for bit in binAlpha[1:]:
        if bit == '0':
            X = sumPoint(X, P, p, a)
            P = sumPoint(P, P, p, a)
        else:
            P = sumPoint(P, X, p, a)
            X = sumPoint(X, X, p, a)
    return P
```

Gyakran használt görbék

- 1999-ben a NIST 5 görbét adott meg, amelyek elsősorban hatékonyság és nem biztonsági szempontok alapján kerültek meghatározásra, (*https://secg.org/sec2-v2.pdf*)
- a NIST-prímek:

$$\begin{aligned} P192 &= 2^{192} - 2^{64} - 1 \\ P224 &= 2^{224} - 2^{96} + 1 \\ P256 &= 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1 \\ P384 &= 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1 \\ P521 &= 2^{521} - 1 \end{aligned}$$

- észrevehető, hogy mindegyik prim felírható valamely 2 hatvány összege, vagy különbségeként
- a $P521$ prímet leszámítva, mindegyik kitevő 32 többszöröse, ami hatékony számításokat tesz lehetővé 32 bites architektúrán

Gyakran használt görbék

P256, vagy secp256r1: az egyik legnépszerűbb görbe

- `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:

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

- a generátor $P(x_1, y_1)$ pont rendje az n prímszám,
- a görbe pontjainak a száma megegyezik n-nel,
- a b értéke egy *seed* érték alapján került meghatározásra, ahol a *seed* érték kiválasztására nincs magyarázat adva!!!
- azt feltételezik, hogy ebben a görbében az ECDL problémát 2^{128} csoportművelet elvégzésével lehet megoldani,
- a Diffie-Helmman kulcscsere során a `TLS 1.3` összes implementációja megköveteli ennek a görbének a támogatását,

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

Gyakran használt görbék

secp256k1:

- 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
- 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` görbéjében,

$n = 115792089237316195423570985008687907852837564279074904382605163141518161494337$
$a = 0$
$b = 7$
$x_1 = 55066263022277343669578718895168534326250603453777594175500187360389116729240$
$y_1 = 32670510020758816978083085130507043184471273380659243275938904335757337482424$

Gyakran használt görbék

Curve25519:

- 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: $8n$.

$a = 486662$
$n = 2^{252} + 27742317777372353535851937790883648493$
$x_1 = 9$
$y_1 = 14781619447589544791020593568409986887264606134616475288964881837755586237401$

Elliptikus görbéken alapuló kriptográfia

Kulcsmegosztás ECC-vel(ECDH, Elliptic Curve Diffie–Hellman key exchange):

- egy nyilvános csatornán keresztül két fél meg tud állapodni csak egy általuk ismert közös titokban,
- Diffie–Hellman típusú kulcscsere, biztonsága az ECDL feltételezésen alapszik,
- a közös titok alkalmazható valamely szimmetrikus titkosító kulcsaként, vagy MAC kulcsként is,
- X25519:
  - Daniel J. Bernstein javasolta,
  - a CURVE25519 görbe alkalmazása ECDH-ban,
  - az egyik leggyorsabb protokoll, kulcsmérete 256 bit, nincs levédve,
  - a TLS 1.3 által támogatott.
- ECIES(Elliptic Curve Integrated Encryption Scheme):
  - a legszélesebb körben használt hibrid titkosítás,
  - átmeneti (ephemeral) kulcsot generálnak,
  - a kulcscserét ECDH-val valósítják meg, amelyet aztán az AES-GCM kulcsaként használnak.

Elliptikus görbéken alapuló kriptográfia

Publikus kulcsú titkosítás ECC-vel (Elliptic Curve Public Key Encryption):

- egy nyilvános csatornán keresztül, a küldő fél a fogadó publikus kulcsát használva titkosít egy általa kiválasztott random értéket, amelyet a fogadó fél a privát kulcsával vissza tud fejteni,
- biztonsága az ECDL feltételezésen alapszik,
- EEECC, ElGamal Encryption Elliptic Curve Cryptography

Elliptikus görbéken alapuló kriptográfia

Digitális aláírás (EC Digital Signature):

- egy nyilvános csatornán keresztül, az aláíró fél a privát kulcsával hitelesíteni tud egy üzenetet, amelynek hitelességét az aláíró publikus kulcsát használva bárki le tud ellenőrizni,
- biztonsága az ECDL feltételezésen alapszik,
- ECDSA: Elliptic Curve Digital Signatures Standard
  - számos szabványon keresztül lehet alkalmazni, ahol a szabványok nem mindig kompatibilisek egymással
  - az egyik leggyakrabban alkalmazott aláírási séma,
- EdDSA twisted Edward Digital Signature Standard
  - Daniel J. Bernstein tervezte 2011-ben
  - a Schnorr digitális aláírási sémán alapszik, miután ez felhasználhatóvá vált, mert 2008-ban lejárt a Schnorr aláírási séma levédettségének ideje.

ECDH - Elliptic Curve Diffie–Hellman key exchange

- feltételezzük, hogy a kommunikációban résztvevő két eszköz **A** és **B**:

  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** eszköz meghatározza:
     - $a \xleftarrow{R} \{2, \ldots, n-1\}$ privát kulcs, $Q_A = aP = (x_{Q_A}, y_{Q_A})$ publikus kulcs,
     - Q_A-t elküldi **B**-nek.
  3. a **B** eszköz meghatározza:
     - $b \xleftarrow{R} \{2, \ldots, n-1\}$ privát kulcs, $Q_B = bP = (x_{Q_B}, y_{Q_B})$ publikus kulcs,
     - Q_B-t elküldi **A**-nak.
  4. az **A** eszköz számításai:

     $$K = x_K, \text{ ahol } aQ_B = (x_K, y_K)$$

  5. a **B** eszköz számításai:

     $$K = x_K, \text{ ahol } bQ_A = (x_K, \hat{y_K}).$$

Projektív koordináták

- egy elliptikus görbe pontjai a standard (x, y) affin koordináták mellett megadhatók a projektív térben is,
- erre azért van szükség mert két pont összeadásakor a multiplikatív inverzek meghatározása költséges,
- a projektív térben való ábrázoláskor nagymértékben csökkenthető az aritmetikai műveletek költségei,
- az implementációkban a projektív koordinátákkal dolgoznak
- az (x, y) affin koordinátájú pontot az $(X : Y : Z), Z \neq 0$ projektív térbeli koordinátákkal helyettesítenek
- az (x, y) affin pont esetében az $x = X/Z^i, y = Y/Z^i$ helyettesítés alkalmazható, amely a $X/Z^i : Y/Z^i : 1$ pontot fogja jelenteni
- a standard projektív koordináták esetében: $i = 1, j = 1$,
- a Jacobian projektív koordináták esetében: $i = 2, j = 3$,
- az $(X : Y : Z)$, amikor $Z = 0$ egyetlen egy affin pontnak sem felel meg, a végtelen pont lesz.