Kriptográfia és Információbiztonság
8. előadás

MÁRTON Gyöngyvér

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

2023

Miről volt szó az elmúlt előadáson?

- véges testek aritmetikája: az AES
- publikus kulcsú rendszerek biztonsága: feltételezések (assumptions)
- kvadratikus (négyzetes) maradékok
- kvadratikus maradék feltételezés
- a Jacobi szimbólum
- a kvadratikus maradék gyökének a meghatározása
- a Rabin titkosító, az SAEP rendszer

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 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,
2. az **A** eszköz a k, p, g ismeretében meghatározza az a, A értékeket, ahol:
   - $a \in \{2, \ldots, p-2\}$ véletlen szám,
   - $A = g^a \pmod{p}$,
   - az a értékét titokban tartja, A-t pedig elküldi **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,
   - $B = g^b \pmod{p}$,
   - a b értékét titokban tartja, B-t pedig elküldi **A**-nak.
4. az **A** eszköz a közös K kulcsot a következőképpen határozza meg:
$$K = B^a \pmod{p}.$$
5. a **B** eszköz a közös K kulcsot a következőképpen határozza meg:
$$K = A^b \pmod{p}.$$

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

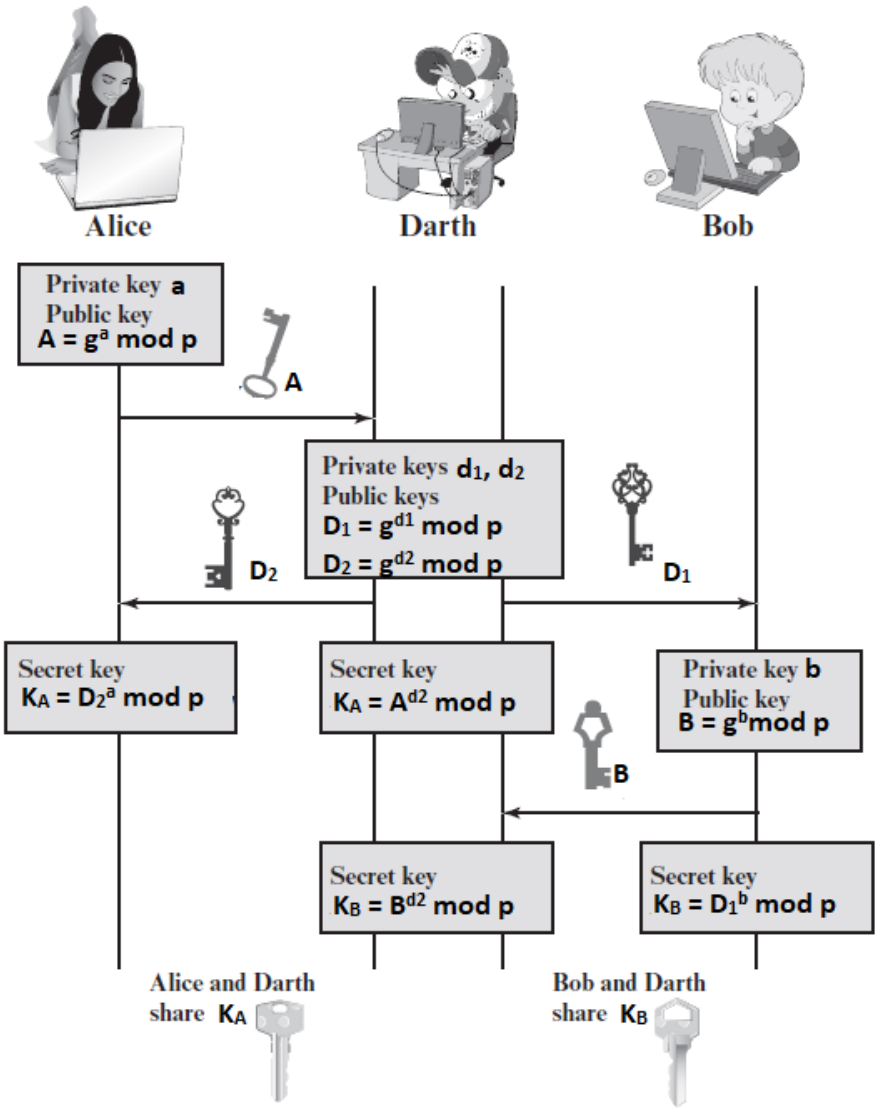
Diffie-Hellman kulcscsere, példa

- 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 nincsenek hitelesítve a résztvevő felek, akkor a rendszer nem lesz biztonságos egy aktív támadó esetében (man in the meedle):
  - egy **D** támadó kiadja magát **A**-nak, mint **B**, illetve kiadja magát **B**-nek, mint **A**,
  - **D** választ két random számot: d_1, d_2,
  - **A** irányába:
    - elküldi **A**-nak $D_2 = g^{d_2}$-t,
    - a közös kulcs **A**-val: $K_A = A^{d_2} = D_2^a = g^{a \cdot d_2}$,
  - **B** irányába:
    - elküldi **B**-nek $D_1 = g^{d_1}$-t,
    - a közös kulcs **B**-vel: $K_B = B^{d_1} = D_1^b = g^{d_1 \cdot b}$,
  - a felhasználók hitelesítése digitális aláírással történik.

Man-in-the-Middle támadás



- minden kulcscsere protokoll támadható Man-in-the-Middle módszerrel, ha a protokoll résztvevői nem hitelesítik egymást
- kivédhető ha digitális aláírást alkalmaznak, ha a publikus-kulcsokra/publikus adatokra tanúsítványokat használnak
- az ábra a Diffie-Hellman kulcsere protokoll elleni Man-in-the-Middle támadást mutatja be
- Darth a támadó, a támadást azzal indítja, hogy meghatározza a mindkét fél irányába szükséges privát és publikus értékeket (kulcsokat)

A digitális aláírás, bevezető

- üzenethitelesítő kód (MAC):
  - biztosítja az átküldött üzenet sértetlenségét
  - hitelesíthető a küldő
  - a MAC nem biztosít eredet szolgáltatást, azaz a küldő bármikor letagadhatja, hogy üzenetet küldött a vevőnek
- az aszimmetrikus/publikus kulcsú rendszerek megoldják az eredet szolgáltatást:
  - az üzenet eredetére vonatkozó bizonyítékot csak a küldő állíthatja elő
  - az üzenet eredetét, hitelességét a rendszer bármelyik résztvevője ellenőrizheti $\rightarrow$ digitális aláírás
- a digitális aláírás biztosítja az üzenetek integritását, hitelességét, és eredetét
- a digitális aláírás nem titkosítja az üzenetet

A digitális aláírás matematikai modellje

Digitális aláírások, egyéb megnevezések: publikus kulcsú tanúsítványok (digital signature, public key certificates). Három algoritmust szükséges értelmezni, ahol K a kulcsok, és $M = f(K)$ az üzenetek halmaza.

- Gen, a kulcs-generáló algoritmus, polinom idejű, lehet véletlenszerű is, meghatározza a pk - publikus kulcsot, és az sk - privát kulcsot:

$$(pk, sk) \xleftarrow{R} Gen(1^k),$$

  ahol $(pk, sk) \in K$ és k a rendszer biztonsági paramétere, legtöbb esetben a generált kulcs bit-hossza, és fennáll: $k \in \mathbb{Z}_{\geq 0}$

- $Aut(sk, \cdot)$ a hitelesítő algoritmus, polinom idejű, véletlenszerű aláírja az m üzenetet:

$$(m, c) \xleftarrow{R} Aut(sk, m),$$

- a $Ver(pk, \cdot)$ az ellenőrző algoritmus, polinom idejű, determinisztikus, ellenőrzi, hogy az m üzenet hiteles-e, kimenete `True` vagy `False` aszerint, hogy $m = \hat{m}_1$-el vagy sem:

$$\hat{m} \leftarrow Ver(pk, c).$$

A helyesség fennáll, ha minden $m \in M$ esetében:

$$Ver(pk, Aut(sk, m)) = m.$$

A digitális aláírási módok

- DA üzenet appendix módban (with message appendix): az aláírást hozzáfűzik az eredeti üzenethez és ez alapján az értékpáros alapján ellenőrzik az aláírást:

$$(c, m) \xleftarrow{R} Aut_{sk}(m)$$

- DA üzenet helyreállítási módban (with message recovery): az eredeti üzenetet nem küldik el, az aláírás alapján helyre lehet állítani az eredeti üzenetet:

$$c \xleftarrow{R} Aut_{sk}(m)$$

- a helyreállítási módban alkalmazott aláírási sémák helymegtakarítása jelentős, ezért inkább ezt a módot alkalmazzák smart kártyák esetében

A digitális aláírás biztonsága

Támadási típusok

- kulcs alapú támadás (key-only attack): csak a nyilvános kulcs áll a támadó rendelkezésére,
- ismert-szöveg alapú támadás (known message attack): a támadó rendelkezésére több szöveg és azoknak megfelelő hiteles aláírás áll,
- választott-szöveg alapú támadás (chosen message attack): a támadó rendelkezik, a saját maga által kiválasztott szövegek és az azoknak megfelelő hiteles aláírással,
- adaptívan választott-szövegek alapú támadás: a kiválasztott üzeneteket és azok hiteles aláírását a támadó adaptív módon, azaz a korábban megszerzett aláírások függvényében tudja kiválasztani

A digitális aláírás 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

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.

Az ElGamal digitális aláírás

Helyesség

$$\begin{aligned} h_2 &= B^c \cdot A^B \\ &= g^{b\cdot c} \cdot g^{a\cdot B} \\ &= g^{b\cdot(h\cdot b_{\mathbf{1}} - a\cdot B\cdot b_{\mathbf{1}})+a\cdot B} \\ &= g^{h-b\cdot b_{\mathbf{1}}\cdot a\cdot B+a\cdot B} \\ &= g^{h-a\cdot B+a\cdot B} \\ &= g^{h} \pmod{p} \\ &= h_1 \end{aligned}$$

mert fennáll:

$$\begin{aligned} c &= (h \cdot b_1 - a \cdot B \cdot b_1) \pmod{p-1} \\ b_1 \cdot b &= 1 \pmod{p-1} \end{aligned}$$

Az ElGamal digitális aláírás, biztonság

- 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:
  - 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 $gcd(v, p-1) = 1$
    - $\hat{c} = -\hat{B} \cdot v^{-1} \pmod{p-1}$,
    - $\hat{m} = \hat{c} \cdot u \pmod{p-1}$.
- ha az alkalmazott hash függvény nem erősen ütközésmentes, akkor is fennáll az egzisztenciális hamisítás lehetősége,
- átalakítható *message recovery* módba

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
- 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)

- $Aut_{(a)}(m) \rightarrow ((B, c), m)$ 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 DSA (Digital Signatures Algorithm)

Helyesség

$$\begin{aligned}\hat{B} &= g^{c_1 \cdot h \pmod q} \cdot A^{(c_1 \cdot B) \pmod q} \\ &= g^{c_1 \cdot (h + a \cdot B)} \\ &= g^{c_1 \cdot c \cdot b} \\ &= g^{b \pmod p} \pmod q,\end{aligned}$$

mert fennáll:

$$c = (h + a \cdot B) \cdot b_1 \pmod q$$

és ez alapján felírható:

$$\begin{aligned}c \cdot b &= (h + a \cdot B) \cdot b \cdot b_1 \pmod q \\ c \cdot b &= (h + a \cdot B) \pmod q\end{aligned}$$

A DSA, megjegyzések

- 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:
- ha az alkalmazott hash függvény nem erősen ütközésmentes, akkor is fennáll az egzisztenciális hamisítás lehetősége,
- előszámításokkal, az aláírás-generálás algoritmus időigénye nagyon felgyorsítható: a b inverze és a $g^b \pmod p$ számítható ki előre, ebben az esetben meg kell oldani ezek biztonságos tárolását,
- átalakítható *message recovery* módba.

Elliptikus görbéken alapuló kriptográfia

- az ECC-vel kapcsolatos kutatásokat Koblitz és Miller az 1980-as évek végén kezdték el, és az RSA-val ellentétben nem védték le,
- egy elliptikus görbe általános alakja:

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

- a kriptográfiában alkalmazott ECC rendszerek esetében egy előre megadott görbe pontjait adják össze, ahol a görbe pontjainak koordinátái egész számok, és a számításokat valamely véges test felett határozzák meg,
- a görbén található pontokhoz hozzávesznek egy speciális pontot, a végtelen $\mathcal{O}$ pontot, ekkor a görbe pontjai és a $\mathcal{O}$ pont által meghatározott halmazt $E(\mathbb{F}_p)$-vel jelöljük, ahol $\mathbb{F}_p$ véges test és p egy prímszám,
- ha az elliptikus görbe pontjai az $\mathbb{F}_{p^e}, e > 1$ felett vannak értelmezve, akkor azt mondjuk, hogy az $\mathbb{F}_p$ egy kiterjesztése felett értelmeztük a görbe pontjait,
- a kriptográfiai protokollok mindig egy adott görbével dolgoznak és a különböző görbék különböző biztonságot nyújtanak: Weierstrass, Montgomery, Edwards típusú görbék,
- gyakran használt görbék: `Secp256k1`, `curve25519`, vagy `p521`
- a bitcoin görbéje: `Secp256k1`, $y^2 = x^3 + 7$. (*Secp256k1*)

Elliptikus görbéken alapuló kriptográfia

az ECC kriptográfia alkalmas:

- kulcscserére (key exchange, key agreement): EC-t alkalmazva egy nyilvános csatornán keresztül két fél meg tud állapodni csak egy általuk ismert közös titokban, amelyet aztán a kommunikáció későbbi lépéseiben szimmetrikus vagy MAC kulcsként tudnak használni például:
  - ECDH, Elliptic Curve Diffie-Hellman kulcscsere
  - X25519: amikor a CURVE25519 görbét alkalmazzák a ECDH-ban, az egyik leggyorsabb protokoll, nincs levédve,
  - FHMQV, Fully Hashed Menezes-Qu-Vanstone kulcscsere
- publikus kulcsú titkosításra (public key encryption): EC-t alkalmazva 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 visszatud fejteni, például:
  - ECIES, Elliptic Curve Integrated Encryption Scheme: átmeneti (ephemeral) kulcs létrehozására használják
  - EEECC, ElGamal Encryption Elliptic Curve Cryptography
- digitális aláírásra (digital signature): EC-t alkalmazva 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 például:
  - ECDSA: Elliptic Curve Digital Signatures Standard
  - EdDSA twisted Edward Digital Signature Standard

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 pontokat, amelyek kielégítik az $y^2 = x^3 - x + 9$ egyenletet
- Schoof algoritmusával, (*wikiLink*) nagy p prímszám esetében is, hatékonyan meg lehet határozni a görbe pontjainak elemszámát
- a görbe két pontján értelmezett összeadásra nézve az $E(\mathbb{F}_{p^e})$ halmaz véges csoport lesz, (*wikiLink*), (*WolframLink*):
  - az egységelem: $\mathcal{O}$, fennáll: $P + \mathcal{O} = \mathcal{O} + P = P$
  - minden $\mathcal{O} \neq P = (x_1, y_1) \in E(\mathbb{F}_{p^e})$ pontnak van additív inverze: $-P = (x_1, -y_1)$
  - a művelet zárt, és asszociatív
  - a művelet kommutatív $\rightarrow$ ábel csoport
- a $P + P$ műveletet $2P$-vel, a $P + P + P$ műveletet $3P$-vel fogjuk jelölni, az αP pedig 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
- az elliptikus görbét $E/\mathbb{F}_p$-vel is jelöljük, ahol $p > 3$ és $\mathbb{F}_p$ véges test

Elliptikus görbéken alapuló kriptográfia

- az αP meghatározásához az `alphaP` (Double and Add algorithm) Python függvény analóg módon jár el, mint a gyorshatványozás, a moduláris hatványozás algoritmusok:

```
def alphaP(alpha, P)
    (x3, y3) = 𝒪
    while alpha > 0:
        if alpha % 2 == 1:
            x3, y3 = sumPoint((x3, y3), P)
        x3, y3 = sumPoint(P, P)
        alpha = alpha // 2
```

- a `sumPoint` függvény két pont összegét határozza meg, aszerint, hogy milyen görbével dolgozunk

Elliptikus görbéken alapuló kriptográfia

- a DL feltételezés az egész számok $\pmod p$ 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
- az ECC rendszerek biztonsága az ECC diszkrét logaritmus (ECC-DL) feltételezésen alapszik
- ECC-DL probléma: legyen P egy pont a $E(\mathbb{F}_p)$ halmazban, amelynek rendje q, azaz $qP = \mathcal{O}$, ekkor a DL probléma azt jelenti, hogy ismerve az $P, \alpha P$ értékeket határozzuk meg az α pozitív egész számot,
- ECC diszkrét logaritmus (ECC-DL) feltételezés: nincs hatékony (polinomiális futási idejű) algoritmus, amely megoldaná az ECC-DL problémát
- az ECC-DL problémát megoldó, leghatékonyabb algoritmus futási ideje, $\mathcal{O}(\sqrt{q})$, ahol q a görbe pontjainak elemszáma, ez azt jelenti, hogy a p nagyságrendje 256 bit kell legyen

Elliptikus görbéken alapuló kriptográfia

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

Elliptikus görbéken alapuló kriptográfia

Weierstrass görbe:

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

$$y^2 \equiv x^3 + a_3 \cdot x + a_4 \pmod{p}$$

  és fennáll: $4 \cdot a_3^3 + 27 \cdot a_4^2 \not\equiv 0 \pmod{p}$

- 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

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2 \\ y_3 &= \lambda \cdot (x_1 - x_3) - y_1 \\ \lambda &= \begin{cases} (y_2 - y_1) \cdot (x_2 - x_1)^{-1}, \text{ha } P \neq Q \\ (3 \cdot x_1^2 + a_3) \cdot (2 \cdot y_1)^{-1}, \text{ha } P = Q \end{cases} \end{aligned}$$

- ha az egyik pont $\mathcal{O}$, akkor fennáll: $P + \mathcal{O} = \mathcal{O} + P = P$
- $4 \cdot a_3^3 + 27 \cdot a_4^2 \not\equiv 0 \pmod{p}$ feltétel biztosítja, hogy a görbének ne legyenek dupla gyökei
- két pont összeadását geometriailag is megszokták adni

Elliptikus görbéken alapuló kriptográfia

Montgomery görbe:

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

  $$a_0 \cdot y^2 \equiv a_1 \cdot x^3 + a_2 \cdot x^2 + a_3 \cdot x + a_4 \pmod{p}$$

  és fennáll: $a_1 \cdot (a_2^2 - 4) \not\equiv 0 \pmod{p}$
- $x = a_1 \cdot x - a_2/3, y = a_1 \cdot y$ helyettesítéssel a Weierstrass alakot kapjuk
- nem minden $\mathbb{F}_p$ feletti Weierstrass alak hozható Montgomery alakra
- a Montgomery görbéken végzett számítások hatékonyabbak, mint a Weierstrass görbéken végzett számítások

Elliptikus görbéken alapuló kriptográfia

Edwards görbe:

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

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

  és fennáll: $d \in \mathbb{F}_p$ és $d \neq 0, 1$
- átalakítható Weierstrass 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

$$\begin{aligned} x_3 &= (x_1 \cdot y_2 + x_2 \cdot y_1) \cdot (1 + d \cdot x_1 \cdot x_2 \cdot y_1 \cdot y_2)^{-1} \\ y_3 &= (y_1 \cdot y_2 - x_1 \cdot x_2) \cdot (1 - d \cdot x_1 \cdot x_2 \cdot y_1 \cdot y_2)^{-1} \end{aligned}$$

Elliptikus görbéken alapuló kriptográfia

Példa: legyen E egy elliptikus görbe, amelynek pontjait az $\mathcal{O}$ és a következő egyenlet megoldásai adják: $y^2 \equiv x^3 - 3 \cdot x + 1 \pmod{11}$. Mivel fennáll, hogy $p = 11 \equiv 3 \pmod 4$ a görbe pontjainak a meghatározása a következőképpen történik:

- minden $x \in \mathbb{Z}_{11}$-re meghatározzuk az $z = x^3 - 3 \cdot x + 1 \pmod{11}$ értéket
- a kapott z értékeket rendre a $k = (p+1)/4$ hatványra emeljük: $t = z^{(p+1)/4} \pmod p$
- 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 eleget tesznek az előző pontban megadott feltételnek

Elliptikus görbéken alapuló kriptográfia

Az előző oldalon megadott egyenleten végzett számítások eredményei, ahol $z = x^3 - 3 \cdot x + 1 \pmod{11}$ 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 görbéken alapuló kriptográfia

- a görbének bármely pontja, amely nem egyenlő $\mathcal{O}$-vel generátor elem is lesz, mert bármely p elemszámú csoport ciklikus, ahol p prímszám
- ez az jelenti, hogy bármely pont esetében, ha ismételten meghatározzuk önmagával az összegét, akkor sorra megkapjuk a görbe pontjait, és a 18-ik összeadás után visszakapjuk az eredeti pontot
- legyen a kiinduló pont $P = (4, 8)$

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

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

Elliptikus Diffie-Hellman kulcscsere

- két távoli egység (számítógép, mobileszköz, stb.) kulcscsere mechanizmusára, hitelesítésére ad megoldást.
- feltételezve, hogy a kommunikációban résztvevő két eszköz **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** eszköz meghatározza az a, Q_A értékeket, ahol:
     - $a \in \{2, \ldots, n-1\}$ véletlen szám,
     - $Q_A = a \cdot P = (x_{Q_A}, y_{Q_A})$,
     - az a értékét titokban tartja, x_{Q_A}-t pedig elküldi **B**-nek.
  3. a **B** eszköz meghatározza a b, Q_B értékeket, ahol:
     - $b \in \{2, \ldots, n-1\}$ véletlen szám,
     - $Q_B = b \cdot P = (x_{Q_B}, y_{Q_B})$,
     - a b értékét titokban tartja, x_{Q_B}-t pedig elküldi **A**-nak.

Elliptikus görbe Diffie-Hellman kulcscsere

4. az **A** eszköz a közös K kulcsot a kövekezőképpen határozza meg:
   - az E görbe egyenlete alapján meghatározza $t = y_{Q_B}^2$ értékét, ami alapján meghatározza az egyik négyzetes maradékot: $y_{Q_B} = t^{(p+1)/4} \pmod p$

   $$K = x_K, \text{ ahol } a \cdot Q_B = (x_K, y_K)$$

5. a **B** eszköz a közös K kulcsot a kövekezőképpen határozza meg:
   - az E görbe egyenlete alapján meghatározza $t = y_{Q_A}^2$ értékét, ami alapján meghatározza az egyik négyzetes maradékot: $y_{Q_A} = t^{(p+1)/4} \pmod p$

   $$K = x_K, \text{ ahol } b \cdot Q_A = (x_K, \hat{y_K}).$$

- helyesség:

  $$a \cdot Q_B = b \cdot Q_A = a \cdot b \cdot P.$$

- 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, tehát sikerül egy közös értékben megegyezni

Elliptikus görbe Diffie-Hellman kulcscsere, példa

- Legyen $E : y^2 \equiv x^3 - 3 \cdot x + 1 \pmod{11}, p = 11, P = (8, 4)$,
  - az **A** eszköz:
    - választja $a = 9$-t $\rightarrow Q_A = a \cdot P = (6, 1)$,
    - elküldi Bobnak az $x_{Q_A} = 6$ értéket.
  - a **B** eszköz:
    - választja $b = 7$-t $\rightarrow Q_B = b \cdot P = (2, 6)$,
    - elküldi Alicenak a $x_{Q_B} = 2$ értéket.
  - **A**: $t = 5, a \cdot Q_B = (10, 5)$,
  - **B**: $t = 6, b \cdot Q_A = (10, 6)$.
- a közös kulcs: $K = 10$.

Elliptikus görbéken alapuló kriptográfia

- 1999-ben a NIST 5 görbét adott meg, amelyek elsősorban hatékonyság és nem biztonsági szempontok alapján kerültek meghatározásra
- 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

Elliptikus görbéken alapuló kriptográfia

P256:

- az egyik legnépszerűbb görbe, `secp256r`-nek is hívják
- a Diffie-Helmman kulcscsere során a `TLS 1.3` összes implementációja megköveteli ennek a görbének a támogatását
- a $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$ prímszám felett van értelmezve
- Weierstrass típusú görbe: $y^2 \equiv x^3 - 3 \cdot x + b \pmod{p}$,
  $b =$ 5ac635d8aa3a93e7b3ebbd55769886bc651d06b0cc53b0f63bce3c3e27d2604b
- a b értéke egy *seed* érték alapján került meghatározásra, ahol a *seed* érték kiválasztására nincs magyarázat adva!!!
- a görbe pontjainak a száma egy q prímszám, amely közel van a 2^{256}-os értékhez
- a standard implementációkban megadják a `G` pontot is, amely a görbe egy generátor eleme
- azt feltételezik, hogy ebben a görbében az ECC-DL problémát 2^{128} csoportművelet elvégzésével lehet megoldani,

Elliptikus görbéken alapuló kriptográfia

Curve25519:

- Dan Bernstein tervezte
- népszerűsége biztonsága és gyorsasága miatt növekvőben van
- a $p = 2^{255} - 19$ prímszám felett van értelmezve, ahol p a legnagyobb prím, amely kisebb mint 2^{255}
- Montgomery típusú görbe: $y^2 \equiv x^3 + a_2 \cdot x^2 + a_4 \pmod{p}$
- a következő Edwards görbévé alakítható át:
  $x^2 + y^2 = 1 + (121665/121666) \cdot x^2 \cdot y^2$
- az a_2 érték kiválasztása: a lehető legkisebb érték kell legyen, amelyre az ECC-DL probléma még nehéz
- a görbe pontjainak a száma egy prímszám értékének a nyolcszorosával egyezik meg
- a generátor $P(x_1, y_1)$ pont rendje:
  $2^{252} + 27742317777372353535851937790883648493$
- a további paraméterek értékei:

$a_2 = 486662$
$a_4 = 1$
$x_1 = 9$
$y_1 = 14781619447589544791020593568409986887264606134616475288964881837755586237401$

Elliptikus görbéken alapuló kriptográfia

a Curve25519 görbét két pontjának a $P = (x_1, y_1)$ és $Q = (x_2, y_2)$-nak az 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

$$\begin{aligned} x_3 &= \lambda^2 - a_2 - x_1 - x_2 \\ y_3 &= \lambda \cdot (x_1 - x_3) - y_1 \\ \lambda &= \begin{cases} (y_2 - y_1) \cdot (x_2 - x_1)^{-1}, \textit{ha } P \neq Q \\ (3 \cdot x_1^2 + 2 \cdot a_2 \cdot x_1 + a_4) \cdot (2 \cdot y_1)^{-1}, \textit{ha } P = Q \end{cases} \end{aligned}$$

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

Elliptikus görbéken alapuló digitális aláírás

Globális paraméterek:

- 2009-ben a FIPS 186: ECDSA
- a globális paraméterek: $p, a, b : y^2 \equiv x^3 + a \cdot x + b \pmod{p}$
- $P = (x_p, y_p)$ a görbe egy generátor eleme
- a P pont n rendje

Kulcs-generálás:

- $a \xleftarrow{R} \{1, n-1\}$
- $Q_A = (x_{Q_A}, y_{Q_B}) = aP$
- a publikus kulcs Q_A, a privát kulcs a

Elliptikus görbéken alapuló digitális aláírás

Az $Aut((a), m)$ algoritmus meghatározza az aláírást, az (r, s) értéket:

- $e = H(m)$, ahol H az SHA-256 hash függvény
- $b \xleftarrow{R} \{1, n-1\}$, $Q_B = (x_{Q_B}, y_{Q_B}) = bP$, $r = x_{Q_B} \pmod n$
- $s = b^{-1}(e + ar) \pmod n$, ahol $b \cdot b^{-1} = 1 \pmod n$

A $Ver((Q_A), (m, r, s))$ algoritmus ellenőrzi az (r, s) aláírást:

- ellenőrzés: fenn kell álljon, hogy $r, s \in \{1, \ldots n-1\}$
- $e = H(m)$, ahol H az SHA-256 hash függvény
- $u_1 = es^{-1}, u_2 = rs^{-1}$, ahol $s \cdot s^{-1} = 1 \pmod n$
- $X = (x_1, y_1) = u_1 P + u_2 Q_A$
- ha $X = \mathcal{O}$ akkor REJECT, másképp $v = x_1 \pmod n$
- az aláírás helyes, ha $v = r$

Helyesség:

$$
\begin{aligned}
s &= b^{-1}(e + ar) \pmod n \Rightarrow b = s^{-1}(e + ar) = s^{-1}e + s^{-1}ar = u_1 + u_2 a \\
X &= (x_1, y_1) = u_1 P + u_2 Q_A = u_1 P + u_2 aP = (u_1 + u_2 a)P = bP = Q_B \\
Q_B &= (x_{Q_B}, y_{Q_B})
\end{aligned}
$$