Aszimmetrikus (nyílt) kulcsú titkosítás

Wettl Ferenc

2014-06-25

Tartalomjegyzék

Jelölések

$a \bmod N$	az a egész N-nel való osztási maradéka
$a \equiv b \pmod N$	az a és b egészek N-nel való osztási maradéka azonos
$\mathbb{Z}_N$	az N-nel való osztás maradékosztályainak additív csoportja (vagy gyűrűje)
$\mathbb{Z}_N^*$	az N-hez relatív prímek maradékosztályainak multiplikatív csoportja
$\mathrm{GF}(q)$, $\mathbb{F}_q$	q elemű véges test
$\mathbb{F}_q[x]$	$\mathbb{F}_q$ feletti polinomok gyűrűje
$\mathbb{F}_q[x]_N$	N-nél kisebb fokú $\mathbb{F}_q$ feletti polinomok
1^n	n csupa 1-esből álló bitlánc, általában a biztonsági paraméter átadására
$\lvert x\rvert$	az x karakterlánc hossza bináris ábrázolásban
$\ell(x)$	az x szám bináris alakjában a számjegyek száma
$\exp(a, b, N)$	moduláris hatványozás, értéke $a^b \bmod N$
$\mathrm{inv}(a, N)$	moduláris inverz, értéke $a^{-1} \bmod N$
log, ln	2-es és természetes alapú logaritmus
$\log_g a$	diszkrét logaritmus, $\log_g a = b$, ha $a = g^b$ (ciklikus csoportban)
$\mathcal{G}$	generátorfüggvény (pl. kulcsgenerátor, függvénygenerátor,...)
$\mathcal{G}^{\mathsf{RSA}}$	RSA paramétergenerátor
$\mathcal{G}^{\mathsf{mod}}$	két prím szorzatát generáló algoritmus ($N = pq$)
$\mathcal{G}^{\mathsf{group}}$	ciklikus csoportot generáló algoritmus
$\mathcal{G}^{\mathsf{prime}}$	prímet generáló algoritmus
E_k	a k kulccsal rejtjelező/kódoló függvény (*encoding*)
D_k	dekódoló függvény (*decoding*)
$\mathsf{Exp}^{\mathsf{xx}}_{A,B}$	kísérlet (*experiment*): A támadja B-t az xx cél/feltétel/körülmény mellett
$\mathsf{Adv}^{\mathsf{xx}}_{A,B}$	az A támadó „előnye” a B (algoritmus/kriptorendszer/...) ellen az xx cél/feltétel/körülmény mellett
$x \leftarrow X$	egy véletlen x elem választása az X halmazból
$y \leftarrow f(x)$	értékadás, melyben f értéke egy véletlen algoritmus eredménye
$y = f(x)$	értékadás determinisztikus f függvénnyel
$\Vert$	az összefűzés (konkatenáció) művelete
$\oplus$	bitenkénti XOR művelet

Bevezetés

E jegyzet célja, hogy a nyilt kulcsú titkosítás ma használt, és használatra ajánlható típusait, és az azokkal kapcsolatos biztonsági kérdéseket, mindenekelőtt a bizonyítható biztonság kérdéseit áttekintse, és a felhasználás szempontjai szerint értékelje.

A kriptográfia történetében mérföldkőnek számít a nyilvános vagy nyílt kulcsú rejtjelezés 70-es évekbeli föltalálása. Ebben az üzenet titkosítása olyan kulccsal történik, melynek megszerzése nem jelent segítséget a támadónak, olyannyira nem, hogy akár nyilvánosságra is lehet hozni. A rejtjelezett üzenet visszafejtése ugyanis másik kulccsal történik. Első alkalmazói a titkosszolgálatok voltak, a kémeknek adott kulcs ellenséges területen való használata nem jelentett biztonsági kockázatot, csak a kémközpontban őrzött titkos kulcsra kellett vigyázni.

A nyílt kulcsú rejtjelezés ötletét Kerckhoffs elvének kiterjesztéseként is felfoghatjuk. Az ő XIX. században megfogalmazott elve az volt, hogy a titkosítás módja nem lehet a titok része, nem okozhat veszélyt, ha azt az ellenség megszerzi, így akár teljesen nyilvános is lehet.

A 80-as évektől kezdődően a kriptográfiában több további forradalmi változás történt.

- Korábban a kriptográfia fő alkalmazói a katonai biztonsági szervezetek, a szerelmes párok és a magányos naplóírók voltak. Mára a kommunikáció formáinak radikális megsokszorozódása – legfőképpen az internet kialakulása – után a kriptográfia eszközeinek használata általánossá, és mindannyiunk életét nem elhanyagolható módon befolyásoló tényezővé vált.

- Korábban a kriptográfia alkalmazása a titkos üzenetküldésre szorítkozott, a kriptográfia csak minél feltörhetetlenebbnek hitt rejtjelezők kitalálásából, és megszerzett titkos üzenetek feltöréséből állt. Mára a kriptográfia az adatkezelésnek egy lényegesen szélesebb körét öleli fel. Egyrészt nem csak a titkosság megőrzése, de a küldő azonosíthatósága, az üzenet sértetlensége, hitelessége, az elküldött üzenet utólagos letagadhatatlansága is a kívánalmak közé került. Másrészt számtalan új protokoll jelent meg a kriptográfiában. A teljesség igénye nélkül felsorolunk néhányat:

  – titokmegosztás,

  - digitális pénz,
  - digitális hitelesítés, digitális időpecsét,
  - biztonságos közös sokrésztvevős számítások (pl. elektronikus választás),
  - digitális jogok kezelése, biztosítása,...

- Végül egy rendkívül fontos változás: a bizonyítható biztonság fogalmának és matematikai technikáinak megszületésével a kriptográfia elmés művészetből, tapasztalatokra épülő mesterségből egzakt tudománnyá vált a 2000-es évekre. Ma ezt tekintjük a modern kriptográfiának.

E rövid írásban igyekszünk a modern kriptográfia tudományának alapfogalmait precízen definiálni és a gyakorlati alkalmazások szempontjából is fontos eredményeit áttekinteni.

1. Elméleti alapok

1.1. Algoritmikus számelméleti alapok

A nyílt kulcsú titkosítás szinte minden ma használt eljárása az algoritmikus számelmélet „nehéz" problémáira épül. Ezért elsőként e téma alapismereteit foglaljuk össze.

Euklideszi algoritmus Két egész szám legnagyobb közös osztóját $\gcd(a, b)$ jelöli (az angol *gratest common divisor* kifejezésből). Ez az a legnagyobb pozitív egész, mely a és b mindegyikének osztója. Ha a két szám egyike 0, a másik abszolút értéke lesz a legnagyobb közös osztó. Kiszámítására létezik hatékony algoritmus, mely arra az egyszerű összefüggésre épül, hogy bármely két a, b egészre $\gcd(a, b) = \gcd(b, a \bmod b)$. Könnyen igazolható, hogy a következő rekurzív algoritmusban a rekurzív függvényhívások száma legföljebb $2 \cdot \ell(a)$, azaz az input hosszának lineáris függvénye. Mivel b zérus voltának ellenőrzése, a abszolút értékének és $a \bmod b$ értékének kiszámítása, polinom időben elvégezhető, így az euklideszi algoritmus is polinom időben befejeződik.

A kriptográfiai alkalmazásokban az euklideszi algoritmust általában csak pozitív egészekre használjuk. Az algoritmus kibővített változata nem csak a

function $\gcd(a, b)$
  input: $a, b \in \mathbb{Z}$
  output: az a és b legnagyobb közös osztója
  if $b = 0$ **then**
    return $|a|$
  else
    return $\gcd(b, a \bmod b)$
  end
end

1. algoritmus: Euklideszi algoritmus

legnagyobb közös osztót, de azt a – mindig létező – két x és y egész számot is megadja, melyekre

$$\gcd(a, b) = xa + yb.$$

A 2. algoritmusban az előzőn annyit változtatunk, hogy csak pozitív egészekre szorítkozunk, így az utolsó lépés – melyben a maradék 0 – elmarad, viszont a b zérus voltának ellenőrzése helyett azt vizsgáljuk, hogy a osztható-e b-vel.

function $\mathrm{gcdx}(a, b)$
  input: $a, b \in \mathbb{Z}$, $a \geqslant b > 0$
  output: $((\gcd(a, b), x, y)$, ahol $\gcd(a, b) = xa + yb$
  if $b \mid a$ **then**
    return $(b, 0, 1)$
  else
    $q = \lfloor \frac{a}{b} \rfloor$, $r := a \bmod b$ (ez az $a = qb + r$ maradékos osztás)
    $(d, x, y) = \mathrm{gcdx}(b, r)$
    return $(d, y, x - qy)$
  end
end

2. algoritmus: Kibővített euklideszi algoritmus

1.1. példa. Kövessük végig a 2. algoritmust $\mathrm{gcdx}(40, 18)$ kiszámításán!

Megoldás. A lépéseket egy táblázatba írjuk. Az (1)–(3) lépések a rekurzív függvényhívásokat és a mellékszámítások eredményeit, a sorban következő (4)–(6) lépések a visszaadott számhármasokat mutatják.

(1) $\mathrm{gcdx}(40,18)$, $40 = 2\cdot 18 + 4$, $q=2$, $r=4$ (6) $(d,x,y) = (2,-4,9)$
(2) $\mathrm{gcdx}(18,4)$, $18 = 4\cdot 4 + 2$, $q=4$, $r=2$ (5) $(d,x,y) = (2,1,-4)$
(3) $\mathrm{gcdx}(4,2)$, (4) $(d,x,y) = (2,0,1)$

Eszerint tehát $\mathrm{gcdx}(40,18) = (2,-4,9)$, azaz $\gcd(40,18) = 2$ és $2 = -4\cdot 40 + 9\cdot 18$. □

$\mathbb{Z}_N$ és a moduláris hatványozás A nyílt kulcsú titkosítás szinte minden típusában szerepet kap a **moduláris hatványozás**. Ezen az $a^b \bmod N$ hatványmaradék kiszámítását értjük, melyre az $\exp(a,b,N)$ függvényjelölést használjuk (az argumentumok száma miatt nem téveszthető össze a valós exponenciális függvénnyel). Kiszámításának nyilván ügyetlen módja az a^b hatvány kiszámítása után venni az N-nel való osztási maradékot, hisz az alkalmazásokban tipikusan a és b többszáz jegyű is lehet. Hasonlóképp nem elég hatékony az sem, ha az a-t b-szer megszorozzuk önmagával, de a számolás egyszerűsítésére minden szorzás után vesszük a szorzat N-nel való osztási maradékát. Ez az algoritmus exponenciális idejű, hisz $b-1$ az $\ell(b)$-nek exponenciális függvénye, és a hatvány kiszámításához ennyi szorzásra van szükség. Polinomiális idejűvé válik az algoritmus, ha a szorzást nem csak az a-val való szorzásra, hanem négyzetre emelésre is használjuk. Ennek módját mutatja a 3. algoritmus.

Könnyen igazolható, hogy az itt ismertetett algoritmussal a moduláris hatványozás az inputok hosszának polinomja időben lefut. Legyen $n = \ell(N)$. Mivel a függvényhívások száma a kitevő bináris jegyeinek számával egyenlő, egy moduláris összeadás műveletigénye $O(n)$, a szorzás pedig elvégezhető $O(n^2)$ lépésben[1], így a moduláris hatványozás műveletigénye $O(n^3)$.

1.2. példa. Számítsuk ki $13^{41} \bmod 53$ értékét!

Megoldás. Előbb hatványozva, majd a maradékot véve (számítógéppel számolva) ezt kapjuk:

$$13^{41} = 469545242509890879708897140933742203507612881 3 \equiv 10 \pmod{53}.$$

41 bináris alakja 101001, így a rekurzív algoritmust a kiértékelésektől kezdve akár „kézzel” számolva is követhetjük:

[1]Kifinomult algoritmusokkal ennél is jobb, aszimptotikusan $O(n \log n)$ korlát is elérhető.

```
function exp(a, b, N)
    input: N ∈ ℤ⁺ a modulus, a ∈ ℤ_N a hatványozás alapja, b ∈ ℤ⁺ a
           kitevő
    output: a^b mod N
    if b = 1 then
        return a
    else
        if 2 | b then
            x := exp(a, b/2, N)
            return (x² mod N)
        else
            x := exp(a, (b − 1)/2, N)
            return (a · x² mod N)
        end
    end
end
```

3. algoritmus: Moduláris hatványozás

bit	számolandó	eredmény	mod 53
1	$13 =$	$13 =$	13
10	$(13)^2 =$	$169 \equiv$	10
101	$(10)^2\,13 =$	$1300 \equiv$	28
1010	$(28)^2 =$	$784 \equiv$	42
10100	$(42)^2 =$	$1764 \equiv$	15
101001	$(15)^2\,13 =$	$2925 \equiv$	10

Minden részletszámítás 3000 alatt maradt, és a 6 bináris jegyből álló kitevő mellett 7 szorzás és 5 maradék kiszámolására volt szükség. Az eredmény: $13^{41} \bmod 53 = 10$. □

Tudjuk, hogy ha $N > 1$, akkor $\mathbb{Z}_N = \{0, 1, \ldots, N-1\}$ halmaz elemein a bináris $(a, b) \mapsto a + b \bmod N$ művelet kommutatív, asszociatív és minden elemnek van inverze a 0-ra, mint zéruselemre nézve (azaz $\mathbb{Z}_N$ e művelettel kommutatív **csoportot** alkot). Hasonlóképp tudjuk, hogy $\mathbb{Z}_N$ a bináris $(a, b) \mapsto ab \bmod N$ művelettel kommutatív, asszociatív, egységelemes algebrai struktúrát ad, de e művelet nem invertálható (azaz $\mathbb{Z}_N$ e művelettel egységelemes kommutatív **félcsoport**, $\mathbb{Z}$ pedig e két művelettel kommuta-

tív **gyűrűt**[2] alkot). Nem fog zavart okozni, ha a $\mathbb{Z}_N$ e két műveletét is az egészeknél használt műveleti jelekkel fogjuk jelölni, tehát e jelöléssel például $\mathbb{Z}_5$-ben számolva $2+3=0$, $2\cdot 3=1$.

$\mathbb{Z}_N$ egy másik reprezentációjához jutunk, ha elemeivel azt a maradékosztályt jelöljük, amelynek az adott szám is tagja. Hogy a zavart elkerüljük, e rövid bekezdés alatt $\mathbb{Z}_N$ elemeit szögletes zárójelbe tesszük, tehát $\mathbb{Z}_N := \{[0],[1],\ldots,[N-1]\}$, ahol

$$[a] = \{\ldots, a-2N, a-N, a, a+N, a+2N, \ldots\},$$

tehát $[a]$ azon egészek halmaza, melyek N-nel osztva a maradékot adnak. E halmazokat hívjuk maradékosztályoknak. A köztük lévő műveletek tehát a következőképp definiálhatók: $[a]+[b] := [a+b \bmod N]$, $[a]\cdot[b] := [ab \bmod N]$. E definíció azért vezet egy értelmes matematikai fogalomhoz, mert ha $[a]+[b]=[c]$ és $[a]\cdot[b]=[d]$, akkor bármely $[a]$-beli és $[b]$-beli egész szám összege a $[c]$ halmazba, szorzata a $[d]$ halmazba, azaz maradékosztályba fog esni. A fenti $\mathbb{Z}_5$-beli számolások tehát e jelöléssel felírva a $[2]+[3]=[0]$, $[2]\cdot[3]=[1]$ alakot öltik. Mindenezek alapján mondhatjuk azt is, hogy $\mathbb{Z}_N$ elemei az N modulusú maradékosztályok, melyek közt az összeadás és szorzás művelete „kompatibilis” az egészek közti műveletekkel.

A moduláris hatványozással valójában a $\mathbb{Z}_N$ multiplikatív félcsoportban végzünk hatványozási műveletet. A 3. algoritmus tetszőleges (multiplikatív) félcsoportban és így csoportban is használható. Ha egy S félcsoportban a művelet polinom időben elvégezhető akkor egy tetszőleges $g \in S$ elemre és egy $b \in \mathbb{Z}^+$ pozitív egészre a g^b hatvány kiszámításához szükséges idő az $\ell(b)$ és S méretének polinomjával becsülhető. Bár a fenti algoritmus gyengéje, hogy tovább számol, ha egy részhatvány 0, de a kriptográfia gyakorlatában ez nem fordul elő, mivel ott az alap és a modulus általában relatív prímek, erről szól a következő néhány bekezdés.

Végül megjegyezzük, hogy additív csoportban is használható az algoritmus, ha egy elem konstansszorosát kell kiszámolni. A $b\cdot g$ kiszámolásához a b bináris alakjából ismételt összeadásokkal kapjuk meg az eredményt. Ez az írásmód az elliptikus görbe kriptográfiában szokásos.

[2]A matematikai pontosság kedvéért mindig jelezhetnénk, hogy épp mely algebrai struktúrára gondolunk, a szokásos megoldások egyike, hogy az elemek halmaza mellé a műveleti jeleket is fölsoroljuk, ekkor $\mathbb{Z}_N$ egy halmaz, $\langle \mathbb{Z}_N, + \rangle$ az additív csoport, $\langle \mathbb{Z}_N, \cdot \rangle$ a multiplikatív félcsoport és $\langle \mathbb{Z}_N, +, \cdot \rangle$ a gyűrű. E jelölések használatára nem lesz szükségünk, ha szükséges, meg fogjuk mondani, melyik struktúrára gondolunk.

$\mathbb{Z}_N^*$ és a moduláris inverz Az a elem N modulus szerinti moduláris inverzén azt a 0 és N közé eső b számot értjük, melyre

$$ab \equiv 1 \pmod{N}.$$

Moduláris inverz csak N-hez relatív prím a számokra létezik, azaz ha $\gcd(a, N) = 1$. Ekkor a kibővített euklideszi algoritmus szerint létezik olyan x és y egész, hogy

$$xa + yN = \gcd(a, N) = 1,$$

ahonnan $xa = 1 - yN \equiv 1 \pmod{N}$, és így a moduláris inverze $b = x \bmod N$. A moduláris inverz a kibővített euklideszi algoritmus leegyszerűsítésével közvetlenül is számolható a 4. algoritmussal. Az algoritmus helyességét az bizonyítja, hogy minden lépésben fennáll az $ax_i \equiv y_i \pmod{N}$ összefüggés $(i = 1, 2)$, így az algoritmus végén $ax_2 \equiv y_2 \pmod{N}$, ami y_2 folyamatosan csökkenése miatt véges sok lépésen belül vagy az $y_2 = 1$ vagy az $y_2 = 0$ értéket eléri.

function inv(a, N)
  input: $a \in \mathbb{Z}$, $N \in \mathbb{Z}^+$, $N > 1$
  output: $b \bmod N$, ahol $ab \equiv 1 \pmod{N}$ és $0 < b < N$, vagy egy üzenet, hogy „Nincs ilyen szám”
  $x_1, y_1 := 0, N$
  $x_2, y_2 := 1, a \bmod N$
  while $y_2 > 1$ **do**
    $q := \lfloor \frac{y_1}{y_2} \rfloor$
    $x_1, y_1, x_2, y_2 := x_2, y_2, x_1 - qx_2, y_1 - qy_2,$
    if $y_2 = 0$ **then**
      return „Nincs ilyen szám”
    end
  end
  return $x_2 \bmod N$
end

4. algoritmus: Moduláris inverz kiszámolása

Mint láttuk, $\mathbb{Z}_N$ a moduláris szorzással nem alkot csoportot, mert e művelet nem invertálható. Tekintsük tehát a $\mathbb{Z}_N^* := \{a \in \mathbb{Z}_N : \gcd(a, N) = 1\}$ halmazt. Ez tehát a 0 és N közé eső, N-hez relatív prím számokból áll.

Például

$$\mathbb{Z}_5^* = \{1,2,3,4\}, \quad \mathbb{Z}_{12}^* = \{1,5,7,11\}, \quad \mathbb{Z}_{14}^* = \{1,3,5,9,11,13\}.$$

Általában, ha p prím, akkor $\mathbb{Z}_p^* = \{1,2,\ldots,p-1\}$. $\mathbb{Z}_N^*$ elemeinek száma $\varphi(N)$, ahol φ az Euler-féle „fí" függvény. Ha N törzstényezős alakja $N = p_1^{e_1} p_2^{e_2} \ldots p_m^{e_m}$, ahol a p_i számok különböző prímek, akkor

$$\varphi(N) = N \prod_{i=1}^{m} \left(1 - \frac{1}{p_i}\right).$$

Használni fogjuk ezt az összefüggést abban az esetben, ha N két prím szorzata, azaz $N = pq$, ekkor $\varphi(N) = (p-1)(q-1)$.

Annak, hogy minden N-hez relatív prím egésznek van moduláris inverze, fontos következménye, hogy $\mathbb{Z}_N^*$ a $\mathbb{Z}_N$-ben már bevezetett szorzásművelettel csoportot alkot, azaz minden $\mathbb{Z}_N^*$-beli elemnek van inverze, és $\mathbb{Z}_N^*$-ban minden $a \cdot x = b$ egyenletnek egyetlen x megoldása van, vagyis a szorzásművelet invertálható.[3]

A $\mathbb{Z}_N^*$ csoport ciklikus, ha N egy páratlan prím hatványa. Több prímosztóval rendelkező N esetén $\mathbb{Z}_N^*$ szerkezetét a kínai maradéktételről szóló következő részben tárgyaljuk.

További következmény, hogy a moduláris hatványozás $\mathbb{Z}_N^*$-ban tovább egyszerűsödik, nevezetesen ha $g \in \mathbb{Z}_N^*$, és b egy pozitív egész, akkor $g^b = g^{b \bmod \varphi(N)}$. Itt kihasználtuk az Euler-tételt, mely szerint ha g és N relatív prímek, akkor $g^{\varphi(N)} \equiv 1 \pmod{N}$.

Általánosságban, ha G egy csoport, melynek rendje (azaz elemszáma) n, akkor $g^b = g^{b \bmod n}$, hisz egy n-edrendű csoportban bármely g elemre $g^n = e$, ahol e a csoport egységelemét jelöli.

Kínai maradéktétel Az RSA kriptorendszerben is alkalmazott kínai maradéktétel legegyszerűbb alakjában azt állítja, hogy az

$$\begin{aligned} x &\equiv a \pmod{p} \\ x &\equiv b \pmod{q} \end{aligned}$$

ekvivalenciarendszer minden egész a és b esetén megoldható ha p és q relatív prímek, és e megoldás egyértelmű modulo $N = pq$, nevezetesen

$$x = \big(a(q^{-1} \bmod p)q + b(p^{-1} \bmod q)p\big) \bmod N, \tag{1}$$

[3]Az előző lábjegyzetben bevezetett jelölést használva írhatjuk, hogy $\langle \mathbb{Z}_N^*, \cdot \rangle$ csoport.

ahol $q^{-1} \bmod p$ a q moduláris inverze a p modulus szerint, míg $p^{-1} \bmod q$ a p moduláris inverze modulo q. Annak ellenőrzése, hogy ez valóban megoldás, behelyettesítéssel ellenőrizhető, hisz

$$\begin{aligned} a(q^{-1} \bmod p)q &\equiv a \pmod p \\ a(q^{-1} \bmod p)q &\equiv 0 \pmod q \\ b(p^{-1} \bmod q)p &\equiv 0 \pmod p \\ b(p^{-1} \bmod q)p &\equiv b \pmod q. \end{aligned}$$

Tekintsük a következő f leképezést:

$$f : \mathbb{Z}_n \to \mathbb{Z}_p \times \mathbb{Z}_q; x \mapsto (x \bmod p, x \bmod q). \tag{2}$$

Annak ellenőrzése, hogy az (1) egyenletbeli x az egyetlen megoldás mod N, épp azzal ekvivalens, hogy f bijekció, és adott (a, b) párhoz az (1) képlettel rendelt x, mint leképezés épp az f inverze. Ennek igazolása arra épül, hogy $|\mathbb{Z}_N| = |\mathbb{Z}_p \times \mathbb{Z}_q| = pq$, így ha volna olyan $(a, b) \in \mathbb{Z}_p \times \mathbb{Z}_q$ pár, mely semmilyen $x \in \mathbb{Z}_N$-re sem lenne egyenlő $f(x)$-szel, akkor létezne olyan (a, b) pár is, melyre a kongruenciarendszer nem lenne megoldható.

Szemléltessük ezt egy egyszerű példán, legyen $p = 3$, $q = 5$. Ekkor a $\mathbb{Z}_{15}$ és a $\mathbb{Z}_3 \times \mathbb{Z}_5$ elemei közt f a következő bijekciót létesíti:

$0 \leftrightarrow (0,0)$	$6 \leftrightarrow (0,1)$	$12 \leftrightarrow (0,2)$	$3 \leftrightarrow (0,3)$	$9 \leftrightarrow (0,4)$
$10 \leftrightarrow (1,0)$	$1 \leftrightarrow (1,1)$	$7 \leftrightarrow (1,2)$	$13 \leftrightarrow (1,3)$	$4 \leftrightarrow (1,4)$
$5 \leftrightarrow (2,0)$	$11 \leftrightarrow (2,1)$	$2 \leftrightarrow (2,2)$	$8 \leftrightarrow (2,3)$	$14 \leftrightarrow (2,4)$

A táblázatba az elemeket a $\mathbb{Z}_3 \times \mathbb{Z}_5$ elempárjai szerint rendeztük, az egy sorban lévő párok első, az egy oszlopban lévők második eleme azonos. A bekeretezett részbe a $\mathbb{Z}_{15}^*$, illetve a $\mathbb{Z}_3^* \times \mathbb{Z}_5^*$ elemei kerültek. Ez az észrevétel sejteti, hogy több igaz egyszerű bijekciónál. Az f művelettartó is.

1.3. tétel (Kínai maradéktétel)**.** *Legyen p és q két relatív prím egész, és legyen $N = pq$. Ekkor a (2) képlettel definiált f függvény izomorfizmust létesít a $\mathbb{Z}_N$ és $\mathbb{Z}_p \times \mathbb{Z}_q$ gyűrűk, valamint a $\mathbb{Z}_N^*$ és $\mathbb{Z}_p^* \times \mathbb{Z}_q^*$ csoportok közt, azaz*

$$\mathbb{Z}_N \cong \mathbb{Z}_p \times \mathbb{Z}_q, \quad \mathbb{Z}_N^* \cong \mathbb{Z}_p^* \times \mathbb{Z}_q^*.$$

Ez azt jelenti, hogy f bijekció $\mathbb{Z}_N$ és $\mathbb{Z}_p \times \mathbb{Z}_q$ közt, valamint $\mathbb{Z}_N^*$ és $\mathbb{Z}_p^* \times \mathbb{Z}_q^*$ közt is, és bármely két $x_1, x_2 \in \mathbb{Z}_N$ esetén $f(x_1 + x_2) = f(x_1) + f(x_2)$ és $f(x_1 \cdot x_2) = f(x_1) \cdot f(x_2)$.

A $\mathbb{Z}_p \times \mathbb{Z}_q$ elemei közti műveletek definíciói természetes módon, az $(a,b)+(c,d) = (a+c, b+d)$ és az $(a,b)\cdot(c,d) = (a\cdot c, b\cdot d)$ képletekkel vannak definiálva.

Megjegyezzük, hogy az (1)-beli $x = f^{-1}(a,b)$ képlet a kínai maradéktételből adódó

$$f^{-1}(a,b) = f^{-1}(a(1,0)+b(0,1)) = af^{-1}(1,0) + bf^{-1}(0,1)$$

felbontásból érthetőbbé válik, hisz

$$f((q^{-1} \bmod p)q) = (1,0), \quad f((p^{-1} \bmod q)p) = (0,1).$$

A kínai maradéktétel szemléltetésére és használatának bemutatására szolgál a következő példa:

1.4. példa. Számítsuk ki az *(a)* $13+7 \bmod 15$, *(b)* $13\cdot 7 \bmod 15$, *(c)* $13^7 \bmod 15$ értékeket, mind $\mathbb{Z}_{15}$-ben, mind $\mathbb{Z}_3 \times \mathbb{Z}_5$-ben számolva!

Megoldás. Az f-re és inverzére vonatkozó képleteket fogjuk használni, de a fenti táblázatban ellenőrizhetjük is az eredményeket. $f(13) = (13 \bmod 3, 13 \bmod 5) = (1,3)$, $f(7) = (7 \bmod 3, 7 \bmod 5) = (1,2)$, azaz $13 \leftrightarrow (1,3)$, $7 \leftrightarrow (1,2)$. Másrészt $f^{-1}(1,0) = (5^{-1} \bmod 3)\cdot 5 = 2\cdot 5 = 10$, $f^{-1}(0,1) = (3^{-1} \bmod 5)\cdot 3 = 2\cdot 3 = 6$.

(a) $13+7 = 20 \equiv 5 \pmod{15}$, másrészt $(1,3)+(1,2) = (2,0) \leftrightarrow 5$, ugyanis $f^{-1}(2,0) = 2\cdot f^{-1}(1,0) = 2\cdot 10 = 20 \equiv 5 \pmod{15}$.

(b) $13\cdot 7 = 91 \equiv 1 \pmod{15}$, másrészt $(1,3)\cdot(1,2) = (1,1) \leftrightarrow 1$, ugyanis $f^{-1}(1,1) = f^{-1}(1,0) + f^{-1}(0,1) = 10+6 = 16 \equiv 1 \pmod{15}$.

(c) $13^7 = 62748517 \equiv 7 \pmod{15}$, másrészt $(1,3)^7 = (1^7 \bmod 3, 3^7 \bmod 5) = (1,2) \leftrightarrow 7$, ugyanis $f^{-1}(1,2) = f^{-1}(1,0) + 2\cdot f^{-1}(0,1) = 10+2\cdot 6 = 22 \equiv 7 \pmod{15}$.

Az utolsó feladat azt is mutatja, hogy a moduláris hatványozás tovább egyszerűsíthető a kínai maradéktétel alkalmazásával. Ezt az RSA titkosításnál használni fogjuk. □

1.2. A modern kriptográfia alapja, a bizonyítható biztonság

A bevezetőben említett bizonyítható biztonság, mint a modern kriptográfia megteremtésének alapfogalma szükségessé teszi, hogy a biztonság lehetséges szintjeit áttekintsük. A tökéletes biztonság fogalmát először Shannon

fogalmazta meg, aki erre vonatkozó tételével megalapozta a kriptográfia tudománnyá válását.

A teljesség – és az egységes tárgyalás – kedvéért felidézzük a szimmetrikus kulcsú rejtjelezés definícióját:

1.5. definíció. Legyen adva az n biztonsági paraméter. A polinom idejű algoritmusokból álló $(\mathcal{G}, \mathrm{E}, \mathrm{D})$ hármasról azt mondjuk, hogy szimmetrikus (privát kulcsú) rejtjelező vagy titkosító rendszer, ha

1. $k \leftarrow \mathcal{G}(1^n)$, azaz $\mathcal{G}$ egy véletlen polinom idejű algoritmus, mely a biztonsági paraméter függvényében visszaad egy legalább n hosszú sztringet, ez lesz a szimmetrikus kulcs.
2. $c \leftarrow \mathrm{E}_k(m)$, azaz a véletlen polinom idejű E algoritmus a k kulcshoz és az $m \in \{0,1\}^*$ nyílt szöveghez ($|k| + |m|$-ben polinom időben) hozzárendel egy c sztringet, az ún. kriptoszöveget.
3. $m = \mathrm{D}_k(c)$, azaz a polinom idejű determinisztikus D algoritmus k-hoz és c-hez ($|k| + |c|$-ben polinom időben) hozzárendel egy m sztringet.
4. Minden n, $\mathcal{G}$ által generált k és tetszőleges m esetén fönnáll az

$$\mathrm{D}_k(\mathrm{E}_k(m)) = m \tag{3}$$

összefüggés.

Az m gyakran nem a $\{0,1\}^*$ halmaznak, hanem a fix hosszú sztringek $\{0,1\}^{M(n)}$ halmazának eleme, ahol $M(n)$ polinomja n-nek. A lehetséges m sztringek $\mathcal{M}$ halmazát nyílt szöveg térnek, a lehetséges kriptoszövegek $\mathcal{C}$ halmazát kripto szöveg térnek, míg a lehetséges kulcsok $\mathcal{K}$ halmazát kulcstérnek nevezzük. Szokásos az a megfogalmazás is, hogy $(\mathcal{G}, \mathrm{E}, \mathrm{D})$ szimmetrikus kulcsú rejtjelező a $(\mathcal{K}, \mathcal{M}, \mathcal{C})$ hármas fölött.

Tökéletes biztonság Egy rejtjelezőt tökéletesen biztonságosnak nevezünk, ha c ismeretében semmilyen információt nem tudhatunk meg m-ről. E fogalomnak több precíz, egymással ekvivalens matematikai definíciója is létezik. Mi itt most azt említjük, mely a további biztonsági definíciókhoz a legjobban illeszkedik, bár nem ez volna a legegyszerűbb vagy legkézenfekvőbb.

1.6. kísérlet ($\mathsf{Exp}^{\mathsf{indCPA}}_{\mathcal{A},\Sigma}(n)$ nyílt szöveg megkülönböztethetetlenségi kísérlet)**.** Legyen $\Sigma = (\mathcal{G}, \mathrm{E}, \mathrm{D})$ egy szimmetrikus kulcsú rejtjelező, $\mathcal{A}$ algoritmus, itt most a támadó, melyet két különböző állapotában hívunk meg, ezeket $\mathcal{A}^{(1)}$ és $\mathcal{A}^{(2)}$ jelöli.

1. $(m_0, m_1) \leftarrow \mathcal{A}^{(1)}(1^n)$, ahol $m_0, m_1 \in \mathcal{M}$, és $|m_0| = |m_1|$.
2. $k \leftarrow \mathcal{G}(1^n)$, $b \leftarrow \{0,1\}$ egy véletlen bit, $c \leftarrow \mathrm{E}_k(m_b)$.
3. $b' \leftarrow \mathcal{A}^{(2)}(c)$

$\mathsf{Exp}^{\mathsf{indCPA}}_{\mathcal{A},\Sigma}(n) = 1$, ha $b' = b$, egyébként $= 0$ (az indCPA jelölés eredete: *indistinguishability under chosen-plaintext attack*)

1.7. definíció (Tökéletes biztonság)**.** A $\Sigma = (\mathcal{G}, \mathrm{E}, \mathrm{D})$ szimmetrikus kulcsú rejtjelező tökéletesen biztonságos, ha bármely korlátlan számítási kapacitással rendelkező $\mathcal{A}$ algoritmus előnye a Σ rejtjelezővel szemben 0, azaz

$$\mathsf{Adv}^{\mathsf{indCPA}}_{\mathcal{A},\Sigma}(n) = \left| \mathbb{P}[\mathsf{Exp}^{\mathsf{indCPA}}_{\mathcal{A},\Sigma}(n) = 1] - \frac{1}{2} \right| = 0.$$

A definícióbeli Adv (az *advantage* szóból) jelölés azt a valószínűséget jelenti, amekkora valószínűséggel $\mathcal{A}$ különbséget tud tenni m_0 és m_1 közt csak c-t ismerve. A tökéletes biztonság azt jelenti, hogy $\mathcal{A}$ előnye a Σ rejtjelezővel szemben 0, még akkor is, ha $\mathcal{A}$ korlátlan számítási kapacitással rendelkezik. Más szóval $\mathcal{A}$ előnye Σ-val szemben ε, ha a c kriptoszöveg ismeretében $\frac{1}{2} + \varepsilon$ valószínűséggel eltalálja, hogy $c = \mathrm{E}_k(m_0)$ vagy $c = \mathrm{E}_k(m_1)$. Azaz ekkora valószínűséggel jut a kriptoszöveg alapján a nyílt szövegre vonatkozó információhoz.

A tökéletes biztonság elérhető, és például a „one time pad (OTP)" nevű rejtjelező meg is valósítja[4]. A tökéletes biztonság elérésének azonban súlyos ára van.

1.8. tétel (Shannon tétele)**.** *Legyen $\Sigma = (\mathcal{G}, \mathrm{E}, \mathrm{D})$ egy szimmetrikus kulcsú rejtjelező a $(\mathcal{K}, \mathcal{M}, \mathcal{C})$ hármas fölött.*

1. *Ha Σ tökéletesen biztonságos, akkor $|\mathcal{M}| \leqslant |\mathcal{K}|$, és Σ elveszíti tökéletes biztonságát, ha ($\mathcal{G}$-t megkerülve) egy k kulcsot többször használunk.*

2. *Ha $|\mathcal{M}| = |\mathcal{C}| = |\mathcal{K}|$, akkor Σ pontosan akkor tökéletesen biztonságos, ha $\mathcal{G}$ egyenletes eloszlás szerint választ $|\mathcal{K}|$-ból (azaz minden k kulcs kiválasztásának $1/|\mathcal{K}|$ a valószínűsége), és minden $c \in \mathcal{C}$ és $m \in \mathcal{M}$ elemhez pontosan egy $k \in \mathcal{K}$ kulcs létezik, melyre $\mathrm{E}_k(m) = c$.*

Azonnal következik Shannon tételéből, hogy nyílt kulcsú rejtjelező nem lehet tökéletesen biztonságos, hisz a rejtjelezést végző nyílt kulcs többször használatos, egy támadó maga tetszőleges sokszor meghívhatja.

[4]Alkalmazásának híres történelmi esete a kubai atomválság után Moszkva és Washington közt kiépített forró drót, az is az OTP-t használta.

Számítási biztonság – gyakorlati megközelítés A szimmetrikus kulcsú titkosítás esetén is nagyon ritkán vannak olyan körülmények, amelyek lehetővé teszik tökéletesen biztonságos protokoll használatát. Ezért akár a szimmetrikus, akár az aszimmetrikus kulcsú rejtjelezésben szükség van a biztonságnak egy más, nem tökéletes, de elegendő szintjét elérni, azaz ha matematikailag lehetséges is a rejtjelező feltörése, ez a gyakorlatban ne sikerülhessen. E biztonság a számítási biztonság (*computational security*), mely arra épül, hogy a támadónak ugyan lehetősége van a rejtjelező feltörésére, de a valóságban a ma és a várható közeljövőben számára rendelkezésre álló erőforrások igénybevétele mellett belátható időn belül csak elhanyagolható valószínűséggel érhessen el sikert.

Egy algoritmus elvégzéséhez szükséges időt legjobb gépi ciklusokban számolni. Szokásos mértékegység a flops (**fl**oating pont **o**peration **p**er **s**econd), vagyis az egy másodperc alatt elvégzett lebegőpontos műveletek száma. A mai szuperkomputerek 10-es nagyságrendű petaflops sebességűek, ez azt jelenti, hogy egy másodperc alatt nagyságrendileg 10^{16} műveletet végeznek el. Például ha egy algoritmus elvégzéséhez 2^{80} flop (**fl**oating pont **op**eration) művelet elvégzésére van szükség, akkor ehhez egy szuperkomputernek $2^{80}/10^{16}$ másodpercre, azaz kb. 46 hónapra van szüksége. Ennél is kifinomultabb megközelítés, ha azt próbáljuk meg egy algoritmusról megbecsülni, hogy amennyiben adott t flop elvégzésére van lehetősége, akkor mekkora a siker valószínűsége.

1.9. definíció. Azt mondjuk, hogy egy kriptográfiai rendszer (t, ε)-biztonságú, ha bármely legföljebb t flopot végrehajtó támadó algoritmus legföljebb ε valószínűséggel sikeres.

A manapság legelterjedtebb kulcshossz 2^{128}. Ha például egy algoritmus egyetlen kulcs ellenőrzését 2^{10} flop alatt elvégzi, akkor annak valószínűsége, hogy egy *brute force* támadással 100 év alatt megtalálja a kulcsot egy mai szuperkomputerrel

$$\frac{100 \cdot 365 \cdot 24 \cdot 60 \cdot 60 \cdot 10^{16}}{2^{128} \cdot 2^{10}} \approx 9 \cdot 10^{-17},$$

ami elegendő biztonságnak tűnik, még akkor is, ha 1000-szeresére gyorsítjuk az algoritmust.

E gyakorlati megközelítésben nehéz definiálni azt, hogy mit kell elég biztonságosnak, és mit nem biztonságosnak tekintenünk. Egyrészt a hardverek képességei gyorsan nőnek (Moore 40 éve megfogalmazott sejtése szerint

exponenciálisan, leggyakrabban idézett formájában azt állítja, az integrált áramkörökben lévő tranzisztorok száma másfél évente duplázódik). Másrészt teljesen más a biztonsági igénye egy szerelmes levélnek és egy kényes hadititoknak. Annyit azért általában mondhatunk, hogy ha $t \cdot \varepsilon < 1/2^{80}$, akkor a technika jelen állása szerint még biztonságos, ha $t \cdot \varepsilon > 1/2^{40}$, akkor már nem biztonságos a rendszer.

Számítási biztonság – aszimptotikus megközelítés A gyakorlati alkalmazásokhoz készült algoritmusok megvalósításakor figyelembe kell venni az előző pontban definiált biztonsági fogalmat, a bizonytalanul minősíthető tartalma miatt viszont más megközelítésre lesz szükségünk.

Az első szempont, hogy a megközelítésnek függvényalapúnak kell lennie, hisz a rendszer használhatósága változik, ha annak biztonsági paramétere (leegyszerűsítve az input sztring, pl. a kulcs hossza) változtatható.

A második szempont, hogy a támadó rendelkezésére álló erőforrás nem lehet korlátlan, sőt kimondható, a támadó algoritmusának hatékonynak kell lennie, így az is a biztonsági paraméter függvényében vizsgálandó.

Végül mérnünk kell a támadó sikerességének valószínűségét, annyit kívánunk, hogy az elhanyagolható legyen.

A függvényalapú megközelítés kényelmesen és jól kezelhetően fölépíthető a véletlen polinomidejű algoritmus fogalmára. Egy $\mathcal{A}$ algoritmus polinom idejű, ha létezik egy olyan p polinom, hogy bármely $x \in \{0,1\}^*$ inputra az $\mathcal{A}(x)$ algoritmus $p(|x|)$ lépésben véget ér, ahol $|x|$ az x hosszát jelöli. Az $\mathcal{A}$ algoritmus véletlen polinom idejű, ha polinom idejű, és az algoritmus hozzáfér egy véletlen függvényhez, mely meghívásakor $\frac{1}{2}$ valószínűséggel 0, $\frac{1}{2}$ valószínűséggel 1 választ ad (a Turing gépek nyelvén az algoritmus hozzáfér egy elegendően hosszú véletlen 0-1 sorozatot tartalmazó szalaghoz). Azt tudjuk, hogy vannak olyan problémák, melyekre van olyan véletlen polinomidejű algoritmus, mely gyorsabb bármely ismert determinisztikusnál, az azonban nincs bizonyítva, hogy volna olyan probléma, melynek megoldására van véletlen polinomidejű algoritmus, de nincs determinisztikus. Így nem biztos, hogy szükség van-e a véletlen jelenlétére, de hátrányt nem okoz. A polinomidejű algoritmusok használatának fő előnye, hogy ha egy polinom sok függvényhívásból álló $\mathcal{A}$ algoritmus csupa polinom idejű algoritmussal kiszámolható függvényt hív meg, akkor maga is polinom idejű.

1.10. definíció. A ν függvény elhanyagolható, ha minden pozitív főegyütt-

hatójú p polinomhoz létezik olyan n_p küszöbindex, hogy ha $n > n_p$, akkor

$$0 \leqslant \nu(n) < \frac{1}{p(n)}.$$

Az n^{-2}, n^{-3}, n^{-1000}, $\frac{1}{n^6-n^4+2}$ függvények nem elhanyagolhatók, míg a 2^{-n}, 3^{-n}, $2^{-\sqrt{n}}$, $n^{-\log n}$ függvények elhanyagolhatók.

Könnyen igazolható, hogy elhanyagolható függvények összege, és polinomszorosa is elhanyagolható.

Mielőtt az aszimmetrikus kulcsú titkosítás biztonságát vizsgálnánk, egy pillanatra térjünk vissza a szimmetrikus kulcsú rejtjelező és a tökéletes biztonság definíciójára (1.5. és 1.7. definíciók). A valóságos alkalmazások nagy részében le kell mondanunk a tökéletes biztonságról, mert nehézségekbe ütközik minden üzenetváltás előtt – egy az üzenet hosszánál nem rövidebb – kulcsot cserélni, cserébe viszont a valóságos támadónak is le kell mondania a korlátlan számítási kapacitás lehetőségéről, mert a valóságban ilyen nincs. Így a tökéletes biztonság definíciójának minimális megváltoztatása elvezet minket egy gyakorlatban használhatóbb új fogalomhoz, a számítási biztonság fogalmához. Ezen belül a biztonságnak több szintje is van. A következő fogalomra több ekvivalens definíció is létezik, ennek megfelelően különböző neveken is szokás említeni. A nyíltszöveg-megkülönböztethetetlenség alapvető volt a tökéletes biztonság definíciójában is, itt is erre fogjuk építeni a biztonság-fogalmunkat, melyet szokás még szemantikai biztonságnak is nevezni, mert mint látni fogjuk, azt fejezi ki, hogy egyetlen hatékony támadó sem tudhat meg nem elhanyagolható valószínűséggel semmit a kriptoszöveg alapján a nyílt szövegről.

1.11. definíció (Szemantikai biztonság)**.** A $\Sigma = (\mathcal{G}, \mathrm{E}, \mathrm{D})$ szimmetrikus kulcsú rejtjelezőt szemantikailag biztonságosnak nevezzük, ha bármely véletlen polinom idejű $\mathcal{A}$ támadó előnye a Σ rejtjelezővel szemben elhanyagolható, azaz bármely $\mathcal{A}$ algoritmushoz van olyan elhanyagolható ν függvény, hogy

$$\mathsf{Adv}^{\mathsf{indCPA}}_{\mathcal{A},\Sigma}(n) = \left| \mathbb{P}[\mathsf{Exp}^{\mathsf{indCPA}}_{\mathcal{A},\Sigma}(n) = 1] - \frac{1}{2} \right| \leqslant \nu(n).$$

A szemantikai biztonság fenti definíciójára pontosabb kifejezés az „üzenet-megkülönböztethetetlenség” vagy még pontosabban a „nyíltszöveg-megkülönböztethetetlenség lehallgató jelenlétében” kifejezések, amit az indCPA rövidítés is jelöl. De mivel ekvivalens fogalmakról van szó, egyszerűbb ezt

használni. E „lehallgató” (*eavesdropper*) kifejezés arra a valóságban előforduló helyzetre utal, amelyben a támadónak van sejtése arról, hogy mi lehet az üzenetben (pl. $m_0 =$ „igen” $\leftrightarrow m_1 =$ „nem”; $m_0 =$ „támadunk” $\leftrightarrow m_1 =$ „maradunk”,...), és lehallgatja az elküldött kódolt üzenetet, azaz megszerzi a $c = \mathrm{E}_k(m_b)$ kriptoszöveget, és abból próbál információhoz jutni a nyílt szöveg kiválasztásához.

Az aszimmetrikus rejtjelező és biztonsága Az aszimmetrikus kulcsú rejtjelezés definíciója csak annyiban különbözik a szimmetrikus kulcsúétól, hogy a kulcsgenerálás során nem egy, hanem két kulcsot kell kapnunk, egyet az E, egyet a D algoritmus részére.

1.12. definíció (Aszimmetrikus kulcsú rejtjelező)**.** Legyen adva az n biztonsági paraméter. A polinom idejű algoritmusokból álló $\Pi = (\mathcal{G}, \mathrm{E}, \mathrm{D})$ hármasról azt mondjuk, hogy aszimmetrikus (nyílt kulcsú) rejtjelező vagy titkosító rendszer, ha

1. $(pk, sk) \leftarrow \mathcal{G}(1^n)$, azaz $\mathcal{G}$ egy véletlen polinom idejű algoritmus, mely a biztonsági paraméter függvényében visszaad két (legalább n bit hosszú) sztringet, pk lesz a nyílt, sk a titkos kulcs.

2. $c \leftarrow \mathrm{E}_{pk}(m)$, azaz a véletlen polinom idejű E algoritmus a pk kulcshoz és az $m \in \mathcal{M}$ nyílt szöveghez ($|pk|+|m|$-ben polinom időben) hozzárendel egy c sztringet.

3. $m = \mathrm{D}_{sk}(c)$, azaz a determinisztikus polinom idejű D algoritmus sk-hoz és c-hez hozzárendel egy m sztringet.

4. Minden n, $\mathcal{G}$ által generált (sk, pk) és tetszőleges $m \in \mathcal{M}$ esetén fönnáll az
$$\mathbb{P}[\mathrm{D}_{sk}(\mathrm{E}_{pk}(m)) = m] = 1 - \nu(n) \tag{4}$$
összefüggés, ahol ν elhanyagolható.

Az E és D algoritmusokra föltehető – ami a gyakorlatban is megeshet –, hogy a dekódolás valamilyen ok miatt meghiúsul, ekkor a D algoritmus egy megkülönböztetett jelet küld (a kriptográfiai irodalomban a $\perp$ jelet szokás erre használni).

A (4) biztonságos megfogalmazása az elvárt $\mathrm{D}_{sk}(\mathrm{E}_{pk}(m)) = m$ egyenlőségnek, amely arra is számít, hogy akár a kulcsgenerálás, akár a rejtjelezés véletlen algoritmusában valamilyen elhanyagolható esélyű hiba történik.

A megkülönböztethetetlenségi kísérlet itt is elvégezhető, de a kulcsgeneráláson kívül ez abban is különbözik a szimmetrikus kulcsú esettől, hogy itt a támadó folyamatosan hozzáfér a nyílt kulcshoz, így maga nyílt szövegeket rejtjelezhet tetszése szerint.

1.13. kísérlet ($\mathsf{Exp}^{\mathsf{indCPA}}_{\mathcal{A},\Pi}(n)$ nyílt szöveg megkülönböztethetetlenségi kísérlet)**.** Legyen $\Pi = (\mathcal{G}, \mathrm{E}, \mathrm{D})$ egy aszimmetrikus kulcsú rejtjelező, $\mathcal{A}$ olyan algoritmus, melyet két különböző állapotában hívunk meg, ezeket $\mathcal{A}^{(1)}$ és $\mathcal{A}^{(2)}$ jelöli, és amely közben polinom sokszor meghívhatja E_{pk}-t.

- $(pk, sk) \leftarrow \mathcal{G}(1^n)$
- $(m_0, m_1) \leftarrow \mathcal{A}^{(1)}(1^n, pk, \mathrm{E}_{pk})$, ahol $m_0, m_1 \in \mathcal{M}$, és $|m_0| = |m_1|$.
- $b \leftarrow \{0,1\}$ egy véletlen bit, $c \leftarrow \mathrm{E}_{pk}(m_b)$.
- $b' \leftarrow \mathcal{A}^{(2)}(c, \mathrm{E}_{pk})$

$\mathsf{Exp}^{\mathsf{indCPA}}_{\mathcal{A},\Pi}(n) = 1$, ha $b' = b$, egyébként $= 0$.

1.14. definíció (Szemantikai biztonság, CPA-biztonság)**.** A $\Pi = (\mathcal{G}, \mathrm{E}, \mathrm{D})$ aszimmetrikus kulcsú rejtjelezőt szemantikailag biztonságosnak vagy CPA-biztonságosnak nevezzük, ha bármely véletlen polinom idejű $\mathcal{A}$ támadó előnye a Π rejtjelezővel szemben elhanyagolható, azaz bármely $\mathcal{A}$ algoritmushoz van olyan elhanyagolható ν függvény, hogy

$$\mathsf{Adv}^{\mathsf{indCPA}}_{\mathcal{A},\Pi}(n) = \left| \mathbb{P}[\mathsf{Exp}^{\mathsf{indCPA}}_{\mathcal{A},\Pi}(n) = 1] - \frac{1}{2} \right| \leqslant \nu(n).$$

Szokás az $\mathsf{Exp}^{\mathsf{indCPA}}_{\mathcal{A},\Pi}(n)$ kísérletet kicsit másként definiálni. Ekkor Exp 2-argumentumos, második argumentuma a b bit, és kimenete a b' bit, azaz

$$b' = \mathsf{Exp}^{\mathsf{indCPA}}_{\mathcal{A},\Pi}(n, b).$$

Ekkor, az $\mathcal{A}$ algoritmus Π-vel szembeni előnyére a fenti definícióbelivel lényegében ekvivalens más definíció adható:

$$\mathsf{Adv}^{\mathsf{indCPA}}_{\mathcal{A},\Pi}(n) = \left| \mathbb{P}[\mathsf{Exp}^{\mathsf{indCPA}}_{\mathcal{A},\Pi}(n, 0) = 1] - \mathbb{P}[\mathsf{Exp}^{\mathsf{indCPA}}_{\mathcal{A},\Pi}(n, 1) = 1] \right|$$

E definíció kicsit életszerűbbnek tűnik, itt $\mathcal{A}$ nem a b bitet próbálja eltalálni, hanem csak az a kérdés, van-e olyan statisztikai próba, mely különbséget tud

tenni a $b = 0$ esetére adott 1-es válaszok és a $b = 1$ esetére adott 1-es válaszok között? Mert ha igen, akkor a rejtjelezés nem biztonságos!

Még erősebb támadásra ad lehetőséget, ha a támadó valamilyen módon hozzá tud jutni maga által előállított kriptoszövegekek dekódoltatásával kapott nyílt szövegekhez, vagy legalább azokról némi információhoz juthat. A nem elég gondosan megtervezett kriptográfiai protokollokban számtalan olyan lehetőség adódhat, amikor erre a támadónak lehetősége nyílik. Például ha a támadó lehallgat egy titkosított üzenetet tartalmazó emailt, majd azt a saját nevén is elküldi a címzettnek, lehet, hogy a címzett válaszában mellékeli a dekódolt üzenetet.

1.15. kísérlet ($\mathsf{Exp}^{\mathsf{CCA}}_{\mathcal{A},\Pi}(n)$ választott kriptoszöveg alapú támadás (CCA))**.** Legyen $\Pi = (\mathcal{G}, \mathrm{E}, \mathrm{D})$ egy aszimmetrikus kulcsú rejtjelező, $\mathcal{A}$ olyan algoritmus, melyet két különböző állapotában hívunk meg, ezeket $\mathcal{A}^{(1)}$ és $\mathcal{A}^{(2)}$ jelöli, és amely közben polinom sokszor meghívhatja E_{pk} és a D_{sk} algoritmust, utóbbit anélkül, hogy magát az sk-t is megkapná.

- $(pk, sk) \leftarrow \mathcal{G}(1^n)$
- $(m_0, m_1) \leftarrow \mathcal{A}^{(1)}(1^n, pk, \mathrm{E}_{pk}, \mathrm{D}_{sk})$, ahol $m_0, m_1 \in \mathcal{M}$, és $|m_0| = |m_1|$.
- $b \leftarrow \{0, 1\}$ egy véletlen bit, $c \leftarrow \mathrm{E}_{pk}(m_b)$.
- $b' \leftarrow \mathcal{A}^{(2)}(c, \mathrm{E}_{pk}, \mathrm{D}_{sk})$, ahol tehát $\mathcal{A}\,(2)$ továbbra is hozzáfér a D_{sk} algoritmushoz, egyedül csak a $\mathrm{D}_{sk}(c)$ függvényhívásra nincs lehetősége.

$\mathsf{Exp}^{\mathsf{CCA}}_{\mathcal{A},\Pi}(n) = 1$, ha $b' = b$, egyébként $= 0$.

1.16. definíció (nyílt szöveg megkülönböztethetetlenség választott kriptoszöveg alapú támadás mellett, CCA-biztonság)**.** A $\Pi = (\mathcal{G}, \mathrm{E}, \mathrm{D})$ aszimmetrikus kulcsú rejtjelezőt CCA-biztonságosnak nevezzük (szokásos elnevezés még: CCA2-biztonság, adaptív CCA-biztonság), ha bármely véletlen polinom idejű $\mathcal{A}$ támadó előnye a Π rejtjelezővel szemben elhanyagolható, azaz bármely $\mathcal{A}$ algoritmushoz van olyan elhanyagolható ν függvény, hogy

$$\mathsf{Adv}^{\mathsf{CCA}}_{\mathcal{A},\Pi}(n) = \left| \mathbb{P}[\mathsf{Exp}^{\mathsf{CCA}}_{\mathcal{A},\Pi}(n) = 1] - \frac{1}{2} \right| \leqslant \nu(n).$$

A CCA-biztonságnak van egy olyan gyengébb támadást feltevő definíciója is, ahol a támadónak csak az m_0 és m_1 kiválasztásáig van lehetősége meghívni

a dekódolót, ekkor ezt CCA1-biztonságnak hívják, a fenti definiált fogalmat pedig CCA2-biztonságnak.

A fenti kísérletet és definíciót aszimmetrikus rejtjelezőre fogalmaztuk meg, de ezek egyszerűen módosíthatóak, hogy szimmetrikus rejtjelezőre is érvényesek legyenek, egyszerűen Π egy Σ szimmetrikus rendszerre cserélendő, és a (pk, sk) pár a közös k kulcsra, minden más marad.

Hogyan lehet aszimmetrikus kulcsú rejtjelezőt konstruálni? Az összes fontosabb ilyen konstrukció valamelyik makacs számelméleti problémára, és abból származó ún. egyirányú kiskapu-függvényre, illetve egyirányú kiskapu-permutációra épül. Ezért először ezeket vesszük sorra.

1.3. Makacs számelméleti problémák

Makacs számelméleti problémáknak nevezzük azokat a kérdéseket, melyek megválaszolására eddig nem találtunk hatékony algoritmust, eddig makacsul ellenálltak minden próbálkozásnak, de ugyanakkor az sincs bizonyítva, hogy ilyen algoritmus nem létezik. Ráadásul nem elég, hogy a probléma bizonyos esetekben legyen nehéz, hanem hogy az esetek egy pontosan és egyszerűen definiálható tartományán belül elhanyagolható számú esetet leszámítva mindig.

Faktorizáció A számelmélet legősibb problémáinak egyike az egészek faktorizációja. Egész számok összeszorzása polinomidejű algoritmus, tényezőkre bontására ezidáig hatékony algoritmust nem ismerünk (nem számítva a Shor-algoritmust, mely polinom időben fogja megoldani e problémát kvantumkomputeren – ha valamikor lesz olyan).

1.17. kísérlet ($\mathsf{Exp}^{\mathsf{Factor}}_{\mathcal{A},\mathcal{G}}(n)$ faktor kísérlet)**.** $\mathcal{G}$ egy polinomidejű algoritmus, mely a biztonsági paraméter függvényében előállít két n-bites számot és azok szorzatát. A két szám n függvényében elhanyagolható valószínűséggel nem prím. $\mathcal{A}$ egy algoritmus, mely egy számpárt ad vissza:

- $(p, q, N) \leftarrow \mathcal{G}(1^n)$
- $(p', q') \leftarrow \mathcal{A}(N)$

$\mathsf{Exp}^{\mathsf{Factor}}_{\mathcal{A},\mathcal{G}}(n) = 1$, ha $p'q' = N$, egyébként $= 0$.

1.18. definíció. Azt mondjuk, hogy a **faktorizáció nehéz**, ha bármely véletlen polinomidejű $\mathcal{A}$ algoritmushoz létezik olyan elhanyagolható $\nu(n)$ függvény, hogy

$$\mathsf{Adv}^{\mathsf{Factor}}_{\mathcal{A},\mathcal{G}}(n) = \mathbb{P}[\mathsf{Exp}^{\mathsf{Factor}}_{\mathcal{A},\mathcal{G}}(n) = 1] \leqslant \nu(n).$$

Moduláris gyökvonás, az RSA-probléma Bár a faktorizáció nehéz problémának tűnik, még sincs széles körben elterjedt kriptográfiai alkalmazása, van azonban olyan, amely szoros kapcsolatban áll vele. Ezek legismertebbike az RSA-probléma. Ez bizonyos korlátozó feltevések mellett lényegében az ismert e kitevővel végzett moduláris hatványozás inverzének, azaz a moduláris e-edik gyökvonás elvégzésének nehézségét feltételezi, ha a modulus két prím szorzata.

Könnyen igazolható, hogy az $f : \mathbb{Z}_N^* \to \mathbb{Z}_N^*; x \mapsto x^e$ függvény invertálható, ha N tetszőleges 1-nél nagyobb egész, és $\gcd(e, \varphi(N)) = 1$. Másként fogalmazva f a $\mathbb{Z}_N^*$ egy permutációja. Megmutatjuk, hogy ha $d = e^{-1} \bmod \varphi(N)$, akkor f inverze az $f^{-1} : \mathbb{Z}_N^* \to \mathbb{Z}_N^*; y \mapsto y^d$ függvény. Ennek ellenőrzéséhez csak azt kell látnunk, hogy ha d moduláris inverze e-nek, azaz $ed \equiv 1 \pmod{\varphi(N)}$, akkor van olyan $k > 0$ szám, hogy $ed = 1 + k\varphi(N)$, és így tetszőleges $x \in \mathbb{Z}_N^*$ számra

$$\begin{aligned}(x^e)^d = x^{ed} &\equiv x^{1+k\varphi(N)} \pmod{N} \\ &\equiv x(x^{\varphi(N)})^k \pmod{N} \\ &\equiv x1^k \pmod{N} \\ &= x.\end{aligned}$$

Kérdés, hogy határozható meg e ismeretében d. Ha $N = pq$, ahol p és q két különböző prím – és mostantól csak erre az esetre szorítkozunk –, akkor $\varphi(N) = (p-1)(q-1)$, az e moduláris inverzének kiszámítására pedig van polinomidejű algoritmusunk. A nehézséget az okozza, hogy $\varphi(N)$-et meghatározni épp oly nehéz, mint faktorizálni N-et. Ha ugyanis ismerjük N tényezőit, akkor polinom időben ki tudjuk számolni a $\varphi(N) = (p-1)(q-1)$ értéket, ha pedig ismerjük N és $\varphi(N)$ értékét, akkor az

$$\begin{aligned}N &= pq \\ \varphi(N) &= (p-1)(q-1)\end{aligned}$$

egyenletrendszerből meghatározható p és q, nevezetesen a $q = N/p$ helyettesítés után a második egyenletből a másodfokú

$$p^2 - (N - \varphi(N) + 1)p + N = 0$$

egyenletet kapjuk, ami polinom időben megoldható. A kérdés tehát az, van-e olyan algoritmus, mely N, e és y ismeretében kiszámítja azt az x számot, melyre $x^e \equiv y \pmod{N}$. Mindmáig nyitott kérdés, hogy ez a probléma ekvivalens-e a faktorizációs problémával, csak annyit tudunk, hogy nyilvánvalóan nem nehezebb nála. Nem ismeretes olyan algoritmus, mely az e-edik gyököt visszaadó függvény ismeretében faktorizálni tudná N-et.

A függvény inverze egyszerű d-edik hatványozás modulo N. Ez az eljárás az RSA-függvény esetén kb. négyszer olyan gyors algoritmusra cserélhető a kínai maradéktétel alkalmazásával. Legyen $d_p = d \bmod p - 1$, és $d_q = d \bmod q - 1$. Mivel $\varphi(p) = p - 1$ és $\varphi(q) = q - 1$, ezért $y^d \equiv y^{d_p} \pmod{p}$ és hasonlóképp $y^d \equiv y^{d_q} \pmod{q}$, és a kínai maradéktétel szerint az

$$\begin{aligned} x &\equiv y^{d_p} \pmod{p} \\ x &\equiv y^{d_q} \pmod{q} \end{aligned}$$

egyértelműen megoldható, nevezetesen az (1) képletet használva

$$x = \left(y^{d_p}(q^{-1} \bmod p)q + y^{d_q}(p^{-1} \bmod q)p\right) \bmod N, \tag{5}$$

ahol a d_p, d_q, $q^{-1} \bmod p$ és $p^{-1} \bmod q$ mind előre számolhatók. Szemléltetésül lássunk egy példát.

1.19. példa. Legyen $p = 11$, $q = 17$, így $N = 187$ és $\varphi(N) = 160$. Legyen $e = 3$ és $x = 16$. Számítsuk ki az $y = f_{N,e}(x)$ és az $f^{-1}_{N,d}(y)$ értékeket, az utóbbit a kínai maradéktétellel!

Megoldás. $y = f_{187,3}(16) = 16^3 \bmod 187 = 4096 \bmod 187 = 169$ még könnyen számolható. Az inverz függvényhez meg kell határoznunk d értékét: $d = 3^{-1} \bmod 160 = 107$. Az előre kiszámolható paraméterek:

$$\begin{aligned} d_p &= 107 \bmod 10 = 7, \\ d_q &= 107 \bmod 16 = 11, \\ q^{-1} \bmod p &= 17^{-1} \bmod 11 = 2, \\ p^{-1} \bmod q &= 11^{-1} \bmod 17 = 14 \end{aligned}$$

Ha csak egyszerű moduláris hatványozással próbáljuk az inverzet kiszámolni, akkor

$$f_{N,d}(y) = f_{187,107}(169) = 169^{107} \bmod 187 = 16.$$

Mivel 107 bináris alakja 1101011, ezért a moduláris hatványozás a 3. algoritmusa szerint 12 darab 187-es modulusú szorzás elvégzése után megkapjuk az eredményt.

Ha a kínai maradéktétellel próbálkozunk, akkor a következő számításokat kell elvégezni, ahol a moduláris hatványozások modulusai csak 11 és 17:

$$\begin{aligned} x_p &= y^{d_p} \bmod p = 169^7 \bmod 11 = (169 \bmod 11)^7 \bmod 11 = 4^7 \bmod 11 = 5 \\ x_q &= y^{d_q} \bmod p = 169^{11} \bmod 17 = (169 \bmod 17)^{11} \bmod 17 = 16^{11} \bmod 17 = 16 \\ x &= \left(y^{d_p}(q^{-1} \bmod p)q + y^{d_q}(p^{-1} \bmod q)p\right) \bmod N \\ &= (5 \cdot 2 \cdot 17 + 16 \cdot 14 \cdot 11) \bmod 187 = 16. \end{aligned}$$

Nagyságrendileg negyed annyi műveletet igényel ez utóbbi módszer alkalmazása. □

1.20. kísérlet ($\mathsf{Exp}^{\mathsf{RSA}}_{\mathcal{A},\mathcal{G}}(n)$ RSA kísérlet)**.** $\mathcal{G}$ egy véletlen polinomidejű algoritmus, mely a biztonsági paraméter függvényében előállít két n-bites prímszámot, azok N szorzatát, egy $\varphi(N)$-hez relatív prím e számot és azt a d számot, melyre $ed \equiv 1 \pmod{\varphi(N)}$. Az $\mathcal{A}$ algoritmus egy $\mathbb{Z}_N^*$-beli számot ad vissza.

- $(N, e, d) \leftarrow \mathcal{G}(1^n)$
- $y \leftarrow \mathbb{Z}_N^*$
- $x \leftarrow \mathcal{A}(N, e, y)$

$\mathsf{Exp}^{\mathsf{RSA}}_{\mathcal{A},\mathcal{G}}(n) = 1$, ha $x^e \equiv y \pmod{N}$, egyébként $= 0$.

1.21. definíció. Azt mondjuk, hogy az **RSA probléma nehéz**, ha bármely véletlen polinomidejű $\mathcal{A}$ algoritmushoz létezik olyan elhanyagolható ν függvény, hogy

$$\mathsf{Adv}^{\mathsf{RSA}}_{\mathcal{A},\mathcal{G}}(n) = \mathbb{P}[\mathsf{Exp}^{\mathsf{RSA}}_{\mathcal{A},\mathcal{G}}(n) = 1] \leqslant \nu(n).$$

Négyzetgyökvonás, a Rabin-probléma Érdekes a helyzet a gyökvonással. Ha $N = pq$ két páratlan prím szorzata, akkor $\varphi(N)$ páros, így a négyzetre emelés, azaz az $f : \mathbb{Z}_N^* \to \mathbb{Z}_N^*; x \mapsto x^2$ függvény nem invertálható, sőt, minden négyzetszámnak 4 négyzetgyöke van.

Euler tételéből azonnal következik, hogy ha p páratlan prím, akkor $\mathbb{Z}_p^*$-ban egy y elem pontosan akkor négyzetelem (kvadratikus maradék mod p),

ha $y^{(p-1)/2} \equiv 1 \pmod{p}$, ugyanis ha van olyan x, hogy $x^2 \equiv y \pmod{p}$, akkor $y^{(p-1)/2} \equiv x^{p-1} \equiv 1 \pmod{p}$. Ebből adódik, hogy $\mathbb{Z}_p^*$ elemeinek fele négyzetelem, fele nem négyzetelem, továbbá ha $p \equiv 3 \pmod{4}$, akkor y négyzetgyökei $\mathbb{Z}_p^*$-ban a $\pm y^{(p+1)/4} \mod p$ számok, ugyanis

$$\begin{aligned}(\pm y^{(p+1)/4})^2 &\equiv y^{(p+1)/2} \pmod{p} \\ &\equiv y^{(p-1)/2} y \pmod{p} \\ &\equiv y \pmod{p}\end{aligned}$$

Legyen tehát p és $q \equiv 3 \pmod{4}$, és $N = pq$. Ekkor az

$$x^2 \equiv y \pmod{N}$$

kongruencia megoldása a kínai maradéktétel szerint ekvivalens az

$$\begin{aligned}x^2 &\equiv y \pmod{p} \\ x^2 &\equiv y \pmod{q}\end{aligned}$$

kongruenciarendszer megoldásával. Mindkettőnek két-két megoldása van, így a következő, valójában négy különböző kongruenciarendszer négy megoldást ad:

$$\begin{aligned}x &\equiv \pm y^{(p+1)/4} \pmod{p} \\ x &\equiv \pm y^{(p+1)/4} \pmod{q}.\end{aligned}$$

Azt látjuk, hogy az N modulusú négyzetre emelés inverze könnyen számolható, ha ismerjük N prímtényezős felbontását. Meglepő módon a négyzetgyökvonás nehézségéről tudjuk, hogy ekvivalens a faktorizációs probléma nehézségével (ellentétben az e-edik gyökvonással), az erre épülő Rabin kriptorendszer mégsem terjedt el, mert a 4 gyök közül az egyetlen helyes kiválasztásához az üzenetbe valamilyen redundáns információt kell tenni.

Diszkrét logaritmus Legyen G egy tetszőleges q-elemű ciklikus csoport, és g egy generátoreleme, azaz G elemeinek halmaza $\{g^0 = 1, g, g^2, \ldots, g^{q-1}\}$.

1.22. kísérlet ($\mathsf{Exp}^{\mathsf{Dlog}}_{\mathcal{A},\mathcal{G}}(n)$, diszkrét logaritmus kísérlet)**.** $\mathcal{G}^{\mathsf{group}}$ egy véletlen polinomidejű algoritmus, mely a biztonsági paraméter függvényében előállít egy q-adrendű ciklikus csoportot, és annak egy g generátorelemét, mely csoportban a művelet hatékonyan számolható. Az $\mathcal{A}$ algoritmus egy G-beli elemet ad vissza.

- $(G, q, g) \leftarrow \mathcal{G}^{\mathsf{group}}(1^n)$, ahol $\ell(q) = n$,
- $h \leftarrow G$ egy egyenletes eloszlás szerint választott véletlen elem
- $x \leftarrow \mathcal{A}(G, q, g, h)$

$\mathsf{Exp}^{\mathsf{Dlog}}_{\mathcal{A},\mathcal{G}^{\mathsf{group}}}(n) = 1$, ha $g^x = h$, egyébként $= 0$.

1.23. definíció. Azt mondjuk, hogy az **diszkrét logaritmus probléma $\mathcal{G}^{\mathsf{group}}$-ra nézve nehéz**, ha bármely véletlen polinomidejű $\mathcal{A}$ algoritmushoz létezik olyan elhanyagolható ν függvény, hogy

$$\mathsf{Adv}^{\mathsf{Dlog}}_{\mathcal{A},\mathcal{G}^{\mathsf{group}}}(n) = \mathbb{P}[\mathsf{Exp}^{\mathsf{Dlog}}_{\mathcal{A},\mathcal{G}^{\mathsf{group}}}(n) = 1] \leqslant \nu(n).$$

A diszkrét logaritmus feltevés azt jelenti, hogy van olyan $\mathcal{G}^{\mathsf{group}}$ algoritmus, melyre nézve a diszkrét logaritmus probléma nehéz. A modern kriptográfia több $\mathcal{G}^{\mathsf{group}}$ algoritmusról is feltételezi ezt.

Diffie–Hellman-problémák A diszkrét logaritmus problémához szorosan kapcsolódik a Diffie–Hellman kulcscserénél szereplő probléma néhány változata. A számítási Diffie–Hellman probléma lényege, hogy ha ismerjük a g^a és g^b elemeket egy g generátorú ciklikus G csoportban, akkor ki tudjuk-e számítani a $g^{a \cdot b}$ elemet? Ha a diszkrét logaritmust könnyen ki tudjuk számolni, akkor igen! Azt azonban nem tudjuk, hogy ha a diszkrét logaritmus probléma nehéz, akkor a számítási Diffie–Hellman probléma is nehéz-e.

1.24. kísérlet ($\mathsf{Exp}^{\mathsf{CDH}}_{\mathcal{A},\mathcal{G}^{\mathsf{group}}}(n)$, számítási (computational) Diffie–Hellman kísérlet)**.** $\mathcal{G}^{\mathsf{group}}$ egy véletlen polinomidejű algoritmus, mely a biztonsági paraméter függvényében előállít egy q-adrendű ciklikus csoportot, és annak egy g generátorelemét. Az $\mathcal{A}$ algoritmus egy G-beli elemet ad vissza.

- $(G, q, g) \leftarrow \mathcal{G}^{\mathsf{group}}(1^n)$, ahol $\ell(q) = n$,
- $a, b \leftarrow \mathbb{Z}_q$
- $x \leftarrow \mathcal{A}(G, q, g, g^a, g^b)$

$\mathsf{Exp}^{\mathsf{CDH}}_{\mathcal{A},\mathcal{G}^{\mathsf{group}}}(n) = 1$, ha $x = g^{a \cdot b}$, egyébként $\mathsf{Exp}^{\mathsf{CDH}}_{\mathcal{A},\mathcal{G}^{\mathsf{group}}}(n) = 0$.

1.25. definíció. Azt mondjuk, hogy a **számítási Diffie–Hellman probléma $\mathcal{G}^{\mathsf{group}}$-re nézve nehéz**, ha bármely véletlen polinomidejű $\mathcal{A}$ algoritmushoz létezik olyan elhanyagolható ν függvény, hogy

$$\mathsf{Adv}^{\mathsf{CDH}}_{\mathcal{A},\mathcal{G}^{\mathsf{group}}}(n) = \mathbb{P}[\mathsf{Exp}^{\mathsf{CDH}}_{\mathcal{A},\mathcal{G}^{\mathsf{group}}}(n) = 1] \leqslant \nu(n).$$

Még érdekesebb és erősebb a döntési (decisional) Diffie–Hellman probléma (DDH)! Az előbb definiált kísérletben megkonstruált (G, q, g, g^a, g^b) ismeretében itt nem az a kérdés, hogy ki tudjuk-e számítani $g^{a\cdot b}$-t, hanem hogy egyáltalán meg tudjuk-e különböztetni egy véletlenül választott elemtől! Legyen tehát $\mathcal{G}^{\mathsf{group}}$ ugyanaz, mint előbb, az $\mathcal{A}$ algoritmus pedig adjon vissza egy b bitet, azaz legyen

$$b \leftarrow \mathcal{A}(G, q, g, g^a, g^b, r)$$

Ha e bit statisztikai jellemzői nem elhanyagolható eséllyel különböznek egy véletlen r elemre és az $r = g^{a\cdot b}$ elemre, akkor e probléma nem nehéz.

1.26. definíció. Azt mondjuk, hogy a **döntési Diffie–Hellman probléma $\mathcal{G}^{\mathsf{group}}$-re nézve nehéz**, ha bármely véletlen polinomidejű $\mathcal{A}$ algoritmushoz létezik olyan elhanyagolható ν függvény, hogy

$$\mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{A},\mathcal{G}^{\mathsf{group}}}(n) = \left|\mathbb{P}[\mathcal{A}(G, q, g, g^a, g^b, r) = 1] - \mathbb{P}[\mathcal{A}(G, q, g, g^a, g^b, g^{a\cdot b}) = 1]\right| \leqslant \nu(n),$$

ahol $r \leftarrow G$.

Az világos, hogy ha a diszkrét log probléma $\mathcal{G}^{\mathsf{group}}$-re nézve könnyű, akkor a CDH probléma is. Ha pedig a CDH könnyű $\mathcal{G}^{\mathsf{group}}$-re nézve, akkor a DDH probléma is. Az állítás megfordítása nem igaz, van olyan G csoport, melyben a diszkrét logaritmus és a CDH nehéznek tűnik, de a DDH könnyű.

Bár vannak olyan $\mathcal{G}^{\mathsf{group}}$ algoritmusok, melyekre nézve a DDH problémát nehéznek hisszük, mégis nem alaptalan az a félelem, hogy esetleg később valaki talál olyan matematikai algoritmust, mely képes $g^{a\cdot b}$ bizonyos tulajdonságai alapján nem elhanyagolható valószínűséggel jól tippelni a g^a-val és g^b-vel való kapcsolatára. Ennek esélyét csökkenti a hash Diffie–Hellman feltevés. Itt egy hash-függvénnyel próbáljuk a $g^{a\cdot b}$ esetlegesen nem teljesen véletlenszerű viselkedését egy véletlenszerű hash függvénnyel „eltakarni”.

1.27. definíció. Legyen $H : G^2 \to G$ egy hash-függvény. Azt mondjuk, hogy a **hash Diffie–Hellman probléma $\mathcal{G}$-re nézve nehéz**, ha bármely véletlen polinomidejű $\mathcal{A}$ algoritmushoz létezik olyan elhanyagolható ν függvény, hogy

$$\mathsf{Adv}^{\mathsf{HDH}}_{\mathcal{A},\mathcal{G}}(n) = \left|\mathbb{P}[\mathcal{A}(G, q, g, g^a, g^b, r) = 1] - \mathbb{P}[\mathcal{A}(G, q, g, g^a, g^b, H(g^b, g^{a\cdot b})) = 1]\right| \leqslant \nu(n),$$

ahol $a, b \leftarrow \mathbb{Z}_q$, $r \leftarrow G$, azaz ezek egyenletes eloszlás szerint véletlenül választott elemek.

Hogy miért épp a $H(g^b, g^{a \cdot b})$ értéket választjuk, miért nem csak például a $H(g^{a \cdot b})$-t, az csak a Diffie–Hellman feltevésekre épülő ElGamal-rejtjelezők konstrukciójából lesz világos, ugyanis ez teszi majd lehetővé a rendszer adott feltevések mellett való biztonságának bizonyíthatóságát.

A később ismertetendő ElGamal rendszerek választott kriptoszöveg alapú támadása elleni biztonságának bizonyíthatóságához még erősebb feltevés szükséges. Ebben a támadó hozzáfér egy F függvényhez is, melyet többször is meghívhat, és amely megmondja, hogy két $u, v \in G$ elem közt fennáll-e az $u^a = v$ összefüggés a rögzített, de a támadó számára ismeretlen a mellett.

1.28. kísérlet ($\mathsf{Exp}^{\mathsf{IDH}}_{\mathcal{A},\mathcal{G}}(n)$, interaktív Diffie–Hellman kísérlet)**.** $\mathcal{G}$ egy véletlen polinomidejű algoritmus, mely a biztonsági paraméter függvényében előállít egy q-adrendű ciklikus csoportot, és annak egy g generátorelemét. Az $\mathcal{A}$ algoritmus egy G-beli elemet ad vissza, miután k-szor meghívja az F függvényt az általa választott paraméterekkel.

- $(G, q, g) \leftarrow \mathcal{G}(1^n)$, ahol $\ell(q) = n$,
- $a, b \leftarrow \mathbb{Z}_q$
- $F(u, v) = 1$, ha $u^a = v$, egyébként $= 0$.
- $x \leftarrow \mathcal{A}(G, q, g, g^a, g^b, F(u_1, v_1), \ldots, F(u_k, v_k))$

$\mathsf{Exp}^{\mathsf{IDH}}_{\mathcal{A},\mathcal{G}}(n) = 1$, ha $x = g^{a \cdot b}$, egyébként $\mathsf{Exp}^{\mathsf{IDH}}_{\mathcal{A},\mathcal{G}}(n) = 0$.

1.29. definíció. Azt mondjuk, hogy az **interaktív Diffie–Hellman probléma $\mathcal{G}$-re nézve nehéz**, ha bármely véletlen polinomidejű $\mathcal{A}$ algoritmushoz létezik olyan elhanyagolható ν függvény, hogy

$$\mathsf{Adv}^{\mathsf{IDH}}_{\mathcal{A},\mathcal{G}}(n) = \mathbb{P}[\mathsf{Exp}^{\mathsf{IDH}}_{\mathcal{A},\mathcal{G}}(n) = 1] \leqslant \nu(n).$$

Az előzőekben felsoroltuk a legismertebb és legfontosabb makacs számelméleti problémákat. Csak azokkal foglalkoztunk, amelyek legalább említés szintjén, vagy az összefüggések megvilágítása érdekében szükségeseknek tűnnek ahhoz, hogy a legelterjedtebb nyílt kulcsú rejtjelezőket megismerjük. Nem foglalkoztunk azokkal a makacs problémákkal, amelyek csak elméleti vagy történeti szempontból érdekesek, például nem ismertettük a hátizsákfeladatra épülő nyílt kulcsú titkosító rendszert, amelyet végül sikerült föltörni az LLL-algoritmussal.

1.4. Egyirányú kiskapufüggvények

Az egyirányú függvények a legalapvetőbb kriptográfiai primitívek közé tartoznak. Lényegük, hogy a függvény kiértékelése könnyű, invertálása vagy ha nem invertálható, akkor az őskép bármelyik elemének kiszámítása viszont nehéz. Azaz adott x-re könnyű kiszámolni $f(x)$-et, de egy $y = f(x)$ érték ismeretében nehéz olyan z-t találni, melyre $f(z) = y$. Használatuk számtalan helyen és számtalan módon előfordul kriptográfiai protokollokban, a szimmetrikus titkosítás egésze lényegében ezekre épül. Elég visszautalnunk arra, hogy a Feistel-tipusú szimmetrikus rejtjelezők (pl. a DES) arra az ötletre építenek, mely egy nem invertálható egyirányú függvényből invertálható egyirányú függvényt képez. A bijektív egyirányú függvényeket szokás egyirányú permutációknak is nevezni. Az angol elnevezések: *one-way function* és *one-way permutation*.

A nyílt kulcsú rejtjelezőkben az egyirányú függvény és permutáció egy speciális változata, az egyirányú kiskapufüggvény, illetve az egyirányú kiskapupermutáció játssza a kulcsszerepet. Ezek is egyirányúak, viszont van egy olyan információ f-ről, melynek birtokában már könnyű f-et invertálni, illetve egy elem valamely ősképét megkeresni. Természetesen ehhez a plusz információhoz nem lehet könnyen hozzájutni csak f ismeretében, de könnyű ilyen f függvényt létrehozni a kiskapu-információval együtt. A formális definíció a következő.

1.30. definíció (Egyirányú függvény)**.** Az $f : \{0,1\}^* \to \{0,1\}^*$ függvény **egyirányú**, ha

1. könnyen számolható, azaz létezik egy olyan polinom idejű $\mathcal{F}$ algoritmus, hogy bármely x-re $\mathcal{F}(x) = f(x)$;

2. nehezen invertálható, azaz tetszőleges x-re, az $y = f(x)$, $n = \ell(x)$ jelölések mellet, tetszőleges véletlen polinomidejű $\mathcal{A}$ algoritmushoz, mely a biztonsági paraméter és y függvényében egy $\{0,1\}^*$-beli elemet ad vissza, létezik olyan elhanyagolható ν függvény, hogy

$$\mathbb{P}[f(\mathcal{A}(1^n, y)) = f(x)] \leqslant \nu(n).$$

Az egyirányú függvény fogalma úgy is definiálható, hogy nem egy függvényt, hanem függvények egy egész osztályát definiáljuk. Ekkor a függvényosztály definíciójában a függvényeket indexelő algoritmusnak is szerepet kell

kapnia. Ezt az általánosabb fogalomalkotást a következő definíciókban meg is tesszük, de már csak az egyirányú függvények egy speciális osztályán, a kiskapu-függvényeken.

1.31. definíció (Egyirányú kiskapu-függvény és kiskapu-permutáció)**.** Egyirányú **kiskapu-függvények egy családján** algoritmusoknak azt a $(\mathcal{G}, f, f^{-1})$ hármasát értjük, melyre:

1. $(k, t, \mathcal{D}_k, \mathcal{R}_k) \leftarrow \mathcal{G}(1^n)$, ahol $\mathcal{G}$ véletlen polinom idejű algoritmus, mely egy (k, t) párt, és a k-val indexelt f_k függvényhez a $\mathcal{D}_k$ értelmezési tartományt és az $\mathcal{R}_k$ értékkészletet generálja.

2. $y = f_k(x)$, ahol f determinisztikus polinom idejű algoritmus, mely a $\mathcal{G}$ által generált k indexhez és egy $x \leftarrow \mathcal{D}_k$ elemhez az $y \in \mathcal{R}_k$ elemet rendeli.

3. f_k egyirányú, azaz tetszőleges véletlen polinomidejű $\mathcal{A}$ algoritmushoz, mely a biztonsági paraméter és y függvényében egy $\mathcal{D}_k$-beli elemet ad vissza, létezik olyan elhanyagolható ν függvény, hogy

$$\mathbb{P}[f_k(\mathcal{A}(1^n, y)) = f_k(x)] \leqslant \nu(n).$$

4. Az f^{-1} algoritmus determinisztikus polinom idejű, mely a $\mathcal{G}$ által generált t indexhez és az $y = f_k(x)$ értékhez olyan $f_t^{-1}(y)$ értéket (esetleg több ilyen értéket) rendel, mely(ek)re

$$f_k(f_t^{-1}(y)) = y.$$

Ha f_k minden k-ra bijekció, egyúttal $\mathcal{D}_k = \mathcal{R}_k$, akkor a $(\mathcal{G}, f, f^{-1})$ hármast egyirányú kiskapu-permutációnak nevezzük. Ilyenkor $\mathcal{G}(1^n)$ kimenete $(k, t, \mathcal{D}_k)$.

Az egyirányú kiskapu-függvény, illetve kiskapu-permutáció angol nevéből származó rövidítések TDF (*oneway trapdoor function*), illetve TDP (*oneway trapdoor permutation*).

Hamarosan látni fogunk példát egyirányú függvényre, egyirányú kiskapu-függvényre és egyirányú kiskapu-permutációra is. Ilyenek konstrukcióihoz legkönnyebben – és az alkalmazásokban is ezek a legfontosabbak – a makacs számelméleti problémák segítségével juthatunk.

Megjegyezzük, **egyirányú függvény létezése** nincs bizonyítva! Annyit tudunk, hogy ha sikerülne létezésüket bizonyítani, akkor abból következne, hogy $FP \neq FNP$, abból pedig, hogy $P \neq NP$. Meglepő módon azt nem tudjuk, hogy $P \neq NP$-ből következik-e egyirányú függvény létezése. Azokra a problémákra, amelyekre a legfontosabb egyirányú kiskapufüggvények épülnek, az sincs bizonyítva, hogy NP-teljesek lennének, azt sejtik, hogy ezek $P \neq NP$ esetén egy NP-köztes (NP-intermediate) osztályba tartoznak. Érdekes módon az NP-teljes problémákat (pl. a hátizsákfeladatot) nem sikerült igazán fölhasználni kriptográfiai célokra, mert bár a legrosszabb esetekben nehéz őket megoldani, egy átlagos feladatra lehet, hogy könnyű.

Egyirányú függvény a faktorizációs problémából Az első gondolatunk az, hogy az $f : (p, q) \to pq$ **egyirányú függvény**, és valóban, ha a faktorizáció nehéz, akkor e függvény egyirányú. Szépséghibája, hogy eleve csak prímpárokra van értelmezve. Némi ügyeskedéssel definiálható olyan függvény, mely pozitív egész x számokra van értelmezve, és amelyben a p és q prímek kiválasztása x-től determinisztikusan függ, de mivel nincs gyakorlati kriptográfiai alkalmazása, ismertetését mellőzzük.

Az RSA-függvény Ha az RSA-probléma nehéz, akkor az 1.20. RSA-kísérletben definiált $\mathcal{G}$ generátorral előállított paramétereket használva az

$$f_{N,e}(x) = x^e \bmod N$$

függvény **egyirányú kiskapu-permutáció** a $\mathbb{Z}_N^*$ halmazon, melynek inverze az

$$f_{N,d}^{-1}(y) = y^d \bmod N$$

függvény. Nyilvánvaló, hogy itt a d a kiskapu-információ. Ezt az $f_{N,e}$ függvényt szokás „tankönyvi RSA"-nak is nevezni, miután több tankönyv e függvényt, mint az RSA rejtjelezőt mutatja be. Fontos megjegyezni, hogy ez „csak" egy egyirányú kiskapu-permutáció, rejtjelezésre önmagában nem használható, amint azt később részletesen indokolni fogjuk.

Moduláris négyzetre emelés, a Rabin-függvény Legyen p és $q \equiv 3 \pmod 4$, és $N = pq$. Amennyiben a faktorizációs probléma nehéz, akkor az

$$f_N(x) = x^2 \bmod N$$

függvény **egyirányú kiskapufüggvény**. Nem invertálható, a kiskapu-információ az N felbontása, azaz p és q. Egy y szám mind a négy ősképe kiszámolható az

$$x \equiv \pm y^{(p+1)/4} \pmod{p}$$
$$x \equiv \pm y^{(p+1)/4} \pmod{q}.$$

kongruenciarendszer megoldásával. Erre a függvényre épül a Rabin-féle kriptorendszer.

Diszkrét logaritmus Ha a diszkrét logaritmus probléma az 1.22. kísérletben definiált $\mathcal{G}$ generátorra nézve nehéz, akkor a q-adrendű, g által generált G ciklikus csoportbeli exponenciális függvény, azaz az

$$f : \mathbb{Z}_q \to G; x \mapsto g^x$$

leképezés **egyirányú permutáció**, ahol az inverz függvény $f^{-1}(y) = \mathrm{Dlog}_g(y)$. E függvény tehát nem egyirányú kiskapu-permutáció, de a Diffie–Hellman-féle ötlettel ilyen is konstruálható belőle.

Diffie–Hellman függvény Tekintsük az 1.24. kísérletben definiált $\mathcal{G}$ által generált q-adrendű, g által generált ciklikus G csoportot, legyen $a \in \mathbb{Z}_q$ olyan, hogy a $h = g^a$ is generátora legyen G-nek (azaz a és $q-1$ legyenek relatív prímek). Ekkor, ha a számítási Diffie–Hellman probléma $\mathcal{G}$-re nézve nehéz, akkor az

$$f_{G,q,g,h} : G \to G; h^b \mapsto g^b$$

függvény **egyirányú kiskapu-permutáció**. A kiskapu-információ az a kitevő, a függvény inverze az $f^{-1}_{G,q,g,a}(x) = x^a$, mely tehát g^b-hez a $(g^b)^a = (g^a)^b = h^b$ elemet rendeli.[5]

Kriptográfiai hash függvények Az egyirányú függvények egy különlegesen fontos osztályát alkotják a kriptográfiai hash függvények (magyarítása hasító függvény).

Egy H függvény **hash függvény**, ha $H : \{0,1\}^* \to \{0,1\}^m$. Fogalmazhatunk úgy, hogy a hash függvény egy tetszőleges véges hosszú sztringet adott

[5]Megjegyezzük, itt az f függvényt az értelmezési tartomány, azaz G elemeinek csak h^b alakban megadott formáján tudjuk polinom időben végrehajtani. Ez a „szépséghiba" azonban nem akadályozza meg, hogy jól használható rejtjelező készüljön belőle.

hosszúságúra tömörít, vagy hogy róla adott hosszúságú lenyomatot készít. Ha $H : \{0,1\}^M \to \{0,1\}^m$, ahol $M > m$, akkor fix hosszú hash függvényről beszélünk.

Hash függvényeket használnak az adatok számítógépes tárolásában, kezelésében. A hash-tábla egy olyan rögzített méretű táblázat, mely az x objektumra vonatkozó valamilyen adatot a táblázat $H(x)$-edik cellájába tesz, ha az üres, ahol H a hash függvény. Nyilván akkor jó egy ilyen H függvény, ha ritkán fordul elő ütközés, azaz ritkán esik meg, hogy $H(x) = H(y)$ két különböző x és y objektumra. A kriptográfiai (vagy kriptográfiailag biztonságos) hash függvényekre sokkal erősebb kikötéseket kell tennünk, nem elég, hogy ütközés ritkán forduljon elő, hanem nehézzé (a gyakorlatban reménytelenné) kell tenni, hogy egy támadó egy ütköző párt szándékosan találhasson. Az ilyen hash függvényeket ütközésállónak fogjuk nevezni. A kriptográfiai, tehát ütközésálló hash függvény definíciójában függvények egy családját fogjuk definiálni. A függvényeket egy kulccsal indexeljük, ez a kulcs azonban nem a titokban tartandó kulcsot jelenti, csak a függvények megkülönböztetésére szolgál. A kulcsot pedig a szükségleteknek megfelelően választjuk, nem feltétlenül egyenletes eloszlás szerint véletlenszerűen. Szokás az így definiált hash függvényt kulcsolt hash függvénynek is nevezni.

1.32. definíció. Hash függvényen (hash függvények egy családján) egy olyan véletlen polinomidejű algoritmusokból álló $H = (\mathcal{G}, \mathcal{H})$ párt értünk, ahol

1. $k \leftarrow \mathcal{G}(1^n)$, azaz a biztonsági paraméter függvényében $\mathcal{G}$ generál egy kulcsot,

2. van olyan m polinom, hogy $\mathcal{H}$ a k és $m(n)$ függvényében tetszőleges $x \in \{0,1\}^*$ sztringhez egy $\{0,1\}^{m(n)}$-beli sztringet rendel, melyet $H_k(x)$-szel jelölünk.

Nem fog zavart okozni, ha H_k-ra, mint $\{0,1\}^* \to \{0,1\}^{m(n)}$ függvényre tekintünk. H-t fix hosszúságú hash függvénynek nevezzük, ha $x \in \{0,1\}^{M(n)}$, azaz $H_k : \{0,1\}^{M(n)} \to \{0,1\}^{m(n)}$, ahol $M(n) > m(n)$.

A hash-függvény kriptográfiai alkalmazhatóságához szükséges, hogy a függvény ütközésálló (további elnevezések: ütközés-ellenálló, ütközésmentes, *collision resistant*) legyen, mely azt jelenti, hogy nehéz találni két olyan x és z sztringet, melyre $H_k(x) = H_k(z)$.

1.33. definíció. A $H = (\mathcal{G}, \mathcal{H})$ hash függvény **ütközésálló** ha tetszőleges véletlen polinomidejű $\mathcal{A}$ algoritmusra – melynek inputja a $\mathcal{G}$ által generál

k kulcs, kimenete egy (x, z) pár –, létezik olyan elhanyagolható ν függvény, hogy

$$\mathbb{P}[H_k(x) = H_k(z)] \leqslant \nu(n).$$

A definícióból egyrészt világos, hogy egy ilyen H_k függvény szükségképpen **egyirányú**, másrészt ha ismerünk egy $(x, H_k(x))$ párt, akkor **nehéz megváltoztatni** x-et úgy, hogy hash értéke, azaz $H_k(x)$ ne változzék (erre a tulajdonságra használják a „gyenge ütközésálló", „második őskép ellenálló", angolul a *second preimage resistant* kifejezéseket).

Az ütközésálló hash függvények számtalan kriptográfiai protokollban kapnak szerepet, a hitelesítési eljárásokban, így a nyílt kulcsú rejtjelezésben is nélkülözhetetlenek.

A Diffie–Hellman hash függvény A Diffie–Hellman függvényből konstruálható ütközésálló hash függvény, mely szerepet kap az ElGamal kriptorendszer bizonyos változataiban is.

1.34. konstrukció (Diffie–Hellman hash függvény)**.** Tekintsük az 1.24. kísérletben definiált $\mathcal{G}$ algoritmust. Legyen $(G, q, g) \leftarrow \mathcal{G}(1^n)$, legyen $h \leftarrow G$ egy véletlen elem, és legyen $k = (G, q, g, h)$ a $H = (\mathcal{G}', \mathcal{H})$ hash függvény definíciójában szereplő $\mathcal{G}'$ generátor algoritmus kimenete, azaz a kulcs. A $\mathcal{H}$ algoritmus eredménye legyen

$$H_k : \mathbb{Z}_q \times \mathbb{Z}_q \to G; (x, y) \mapsto g^x h^y.$$

Az így definiált hash függvényt Diffie–Hellman hash függvénynek nevezzük.

1.35. tétel. *Ha a diszkrét logaritmus probléma nehéz az 1.22. kísérletben definiált $\mathcal{G}$ algoritmusra nézve, akkor az 1.34. konstrukcióban megadott Diffie–Hellman hash függvény ütközésálló.*

1.5. Támadások az RSA-függvény ellen

Az egyirányú (kiskapu)függvények és permutációk elleni támadáson mindazoknak a körülményeknek és lehetőségeknek az áttekintését értjük, amelyek mellett megszűnik a jelölt függvény egyirányúsága. A legtöbb és legérdekesebb támadás az RSA-függvény ellen készült, ezekre koncentrálunk mi is.

Amikor e kicsi Kis e választása mellett az szól, hogy ekkor igen gyors a rejtjelezés. Ekkor azonban az $f(x) = x^e \bmod N$ függvény könnyen invertálható, ha e és x is kicsi. Például ha $e = 3$, és $x < \sqrt[3]{N}$, akkor $y = x^e = x^3 < N$, így y-ból x egy egészek fölött elvégzett egyszerű köbgyökvonással megkapható.

Coppersmith a fenti elemi példának egy mély, és sok alkalmazással rendelkező, bizonyításában az LLL-algoritmusra építő általánosítását bizonyította:

1.36. tétel (Coppersmith, 1997)**.** *Legyen N pozitív egész, $f \in \mathbb{Z}[x]$ egy d-edfokú polinom, és $M = N^{1/d-\varepsilon}$, ahol $\varepsilon \geqslant 0$. Ekkor az összes olyan x hatékonyan megtalálható, amelyre $|x| < M$ és $f(x) \bmod N = 0$. A futási idő az LLL-algoritmusnak egy $O(\min(\frac{1}{\varepsilon}, \log N))$-dimenziós rácson való futási idejétől függ.*

Mindennek ellenére van az RSA-függvényre épülő rejtjelezőnek olyan implementációja, ahol $e = 3$, de így természetesen x garantáltan nem kicsi.

Kis e közös üzenettel Az egyszerűség kedvéért legy $e = 3$, és legyen N_1, N_2, N_3 három páronként relatív prím egész. A

$$c_i = x^3 \bmod N_i, \quad i = 1, 2, 3$$

függvényértékekből – bár külön-külön nem tudnánk megkapni x értékét, e három egyenletből már igen, ugyanis a kínai maradéktétel szerint van olyan $c^* < N_1 N_2 N_3 = N^*$ egész, melyre

$$\begin{aligned} c^* &\equiv c_1 \pmod{N_1} \\ c^* &\equiv c_2 \pmod{N_2} \\ c^* &\equiv c_3 \pmod{N_3} \end{aligned}$$

De mivel $c^* = x^3 \bmod N^*$, $x < N_i$ $(i = 1, 2, 3)$, ezért $x^3 < N^*$. Tehát ismét egy egyszerű egészek fölötti köbgyökvonás segít.

Hastad támadása Az előző gyenge megoldás látszólag könnyen javítható, ha a közös x érték elé megkülönböztető jelzést, paddinget teszünk, pl. legyen

$$c_i = (i \cdot 2^m + x)^3 \bmod N_i, \quad i = 1, 2, 3$$

Meglepő módon ekkor is meghatározható x a c_i értékekből. Sőt, még akkor is, ha

$$c_i = (p_i(x))^e \bmod N_i, \quad i = 1, 2, \ldots, k$$

ahol p_i mindegyike polinom és k elegendően nagy. Az alábbi egészen általános eredmény fontos kriptográfiai következménye, hogy a paddingnek mindig véletlennek kell lennie.

1.37. tétel (Hastad). *Legyen $N_1,\ldots,N_k$ páronként relatív prím és jelölje $N_{\min}$ közülük a legkisebbiket. Legyen $g_i \in \mathbb{Z}_{N_i}[x]$ legföljebb d-edfokú polinom $(i = 1,\ldots,k)$. Tegyük fel, hogy egyetlen olyan $x < N_{\min}$ létezik, hogy*

$$g_i(x) \equiv 0 \pmod{N_i} \text{ minden } i = 1,2,\ldots,k \text{ indexre.}$$

Ha $k > d$, akkor x hatékonyan meghatározható az (N_i, g_i) párok ismeretében.

E tétel egyszerűen fogalmazva azt mondja, hogy egyváltozós polinomkongruenciák relatív prím modulusok mellett megoldhatóak, ha elegendő számú kongruencia áll rendelkezésre.

Ezt az RSA-függvény invertálásához a következőképp használhatjuk. Tekintsük az f_{N_i,e_i} RSA-függvényeket $(i = 1,2,\ldots,k)$. Kiszámoljuk az $c_i = f_{N_i,e_i}(p_i(x))$, ahol a p_i polinomok alkalmazásával kívánjuk megakadályozni x kiszámíthatóságát a c_i értékekből. Legyen $g_i = p_i^{e_i} - c_i \bmod N_i$. Az előző tétel szerint ha $k > \max_i(e_i \deg p_i)$, akkor e kongruenciarendszer megoldható, tehát az RSA-függvényekből még ilyen módosítással is felfedhető a kiértékelésre szánt közös x.

Amikor d kicsi Ahogy kis e a rejtjelező, kis d a dekódoló futási idejét csökkenti. Meglepő módon, csak az e és az $N = pq$ szám ismeretében megszerezhető a kiskapu-információ, azaz d, ha az kicsi.

1.38. tétel (Wiener, 1990). *Ha $N = pq$, $q < p < 2q$ és $d < \frac{1}{3}\sqrt[4]{N}$, akkor N és e ismeretében d polinom időben meghatározható.*

Az algoritmus arra a számelméleti tételre épít, hogy ha az $\frac{e}{N}$ törthöz létezik olyan $\frac{k}{d}$ tört, hogy

$$\left|\frac{e}{N} - \frac{k}{d}\right| < \frac{1}{d^2},$$

és $d < N$, akkor a $\frac{k}{d}$ tört csak az $\frac{e}{N}$ lánctörtalakjának egy szelete lehet. A fenti egyenlőtlenség pedig könnyen bizonyítható a tétel feltételeiből.

Az $\frac{e}{N}$ lánctörtalakja összes szeletének végigpróbálása hatékonyan elvégezhető – tapasztalatok szerint még a $\frac{1}{3}\sqrt[4]{N}$-es korlát fölött, de azért $\sqrt{N}$ alatt. (A felső korlát pontos meghatározása nyitott kérdés.)

Amikor x kicsi – gyökös támadás Az világos, hogy ha x-ről tudjuk, hogy egy adott M-elemű halmazba esik, akkor M-ben lineáris lépésben az $y = f(x)$ értékből meghatározhatjuk x-et. Tegyük fel, hogy tudjuk, $0 \leqslant x < M = 2^m$, azaz x egy legföljebb m-bites szám. A következő ötletes algoritmus $\sqrt{M}$ lépésben nagy valószínűséggel megtalálja a $c = x^e \bmod N$ rejtett szöveg ismeretében x-et! Az ötlet sok egyéb támadásnak is alapötlete.

input: a nyilvános kulcs (N, e); a rejtett szöveg c; a szöveg hossza m
output: x, ahol $x^e \bmod N = c$ és x hossza m.

$\mu \leftarrow (\frac{1}{2}, 1)$
$R = 2^{\mu m}$
for $r = 1$ **to** R **do**
  $c_r = \frac{c}{r^e} \bmod N$
end
sort $(r, c_r)|_{r=1}^{R}$ a második elemek szerint
for $s = 1$ **to** R **do**
  if $s^e \bmod N = c_r$ valamelyik r-re **then**
    return $r \cdot s \bmod N$
  end
end

5. algoritmus: Gyökös támadás az RSA-függvény inverzének kiszámítására

Ha választunk egy véletlen $x < 2^m$ üzenetet, akkor nagy valószínűséggel van két olyan $1 < r, s \leqslant 2^{\mu m}$ egész, hogy $x = r \cdot s$. Így

$$c \equiv x^e \equiv r^e \cdot s^e \pmod{N},$$

azaz $c/r^e \equiv s^e \pmod{N}$. Az algoritmus idejéből $O(m2^{\mu m})$ lépés a rendezéshez, $O(m)$ lépés a bináris kereséshez kell.

Ez rendkívül erős támadás, ha belegondolunk, egy 2^{80} bites x-ből képzett $y = f(x)$ invertálása kimerítő kereséssel a mai technikai körülmények közt belátható időn belül kivitelezhetetlen, viszont 2^{40} lépés már nem!

Közös modulus Két különböző RSA-függvénynek nem szabad közös modulust választani, mert az egyik kiskapu-információját ismerve a másik is

megszerezhető. Ugyanis ha $N = pq$, és ismerjük az f_{N,e_i} RSA-függvények e_i értékeit, és csak egyikük, kiskapu-információját, pl. d_1-et, akkor abból az összes többit is ki tudjuk számolni, hisz $e_1 d_1 \equiv 1 \pmod{\varphi(N)}$, ahonnan N felbontása, abból pedig a többi függvény d_i-értéke ($i = 2, 3 \dots$) kiszámolható.

Kérdés, egy tulajdonos használhat-e egy modulushoz különböző e_i-ket? A válasz erre is határozott nem! Az RSA-függvény egyirányú kiskapu-permutáció volta azonnal megszűnik, ha az $c_i = f_{N,e_i}(x)$ értékeket relatív prím e_1 és e_2 kitevővel is hozzáférhetővé válnak. Mivel e_1 és e_2 relatív prímek, ezért létezik olyan u és v, hogy $ue_1 + ve_2 = 1$, így viszont

$$c_1^u c_2^v = x^{ue_1} x^{ve_2} = x^{ue_1+ve_2} \equiv x^1 = x \bmod N.$$

Megváltoztathatóság Ha csak annyit tudunk, hogy $y = f(x) = x^e \bmod N$ argumentuma egy olyan x érték, mely egy számadat, akkor megváltoztathatjuk y értékét úgy, hogy a változás hatását előre ki tudjuk számolni. Például ha azt tudjuk, hogy x egy árajánlat, akkor ugyan nem tudjuk meg mennyi az x, de alá tudunk ígérni, ha ugyanis x 5%-a egész szám, akkor $y \cdot (\frac{19}{20})^e \bmod N$ dekódolás után $0.95x$-et ad.

Implementációk elleni támadások Egy-egy egyirányú függvény megvalósítása valamely valós rendszerben sok olyan támadásra ad lehetőséget, melyek az implementációval kapcsolatosak. Ezek egy része kifinomult műszaki megoldásokra épít, de az ezek elleni védelem nem szükségképpen műszaki megoldást igényel.

Itt csak két egyszerű támadást említünk. A legfontosabb, leggyakoribb, és bizonyos esetekben igen hatékony az időméréses (timing) támadás. Ez nem csak az egyirányú függvények ellen, de szinte bármilyen kriptográfiai protokoll ellen működik, ahol egy program futási ideje, korrelációba kerülhet érzékeny adatokkal. Például az RSA-függvény dekódolásakor alkalmazott moduláris hatványozásban használt d kitevő, bitről bitre kitalálható a dekódolónak küldött – megfelelően kiválasztott – üzenetek visszafejtésére fordított időből.

Az egyik általánosan elterjedt védekezés ez ellen a **vakítás**. A vakításnál a dekódoló nem a kapott y-t emeli d-edik hatványra, hanem választ egy r véletlen számot, és a $z = yr^e \bmod N$ számot hatványozza, mely a támadónak már ismeretlen. Ekkor a számítás $z^d = y^d r \bmod N$, vagyis a keresett $x = y^d \bmod N$ számot a $z^d/r \bmod N$ képlet adja.

Természetesen lehet próbálkozni azzal is, hogy minden programrész úgy legyen megírva, hogy semelyik futási ideje se kerüljön korrelációba érzékeny adatokkal. Ilyesmi elérhető pl. felesleges ciklusok beszúrásával, melyek révén elérhető, hogy a program mindig azonos ideig fusson.

Ha támadónak nem csak a futási időt, de az elektronikus eszköz áramfelhasználását, vagy bármely más fizikai – időben változó – jellemzőjét mérni tudja, további támadási lehetőségekhez jut!

Még érdekesebb és veszélyesebb a helyzet, ha a támadó be is tud avatkozni az eszköz működésébe. Az ilyen támadások egyike, mely pl. smart kártyákon viszonylag könnyen megvalósítható, RSA-függvény kiskapu-információjának megszerzése futási hiba okozásával. Az 1.4. példa *(c)* pontjában, illetve az 1.19. példában láttuk, hogyan lehet moduláris hatványozást a kínai maradéktételre építve kiszámolni. Tegyük fel, hogy egy támadó annyira nyomon tudja követni egy smart kártya működését, hogy tudja mikor számolja ki az $x_p = y^{d_p} \bmod p$, illetve az $x_q = y^{d_q} \bmod q$ értékeket. Elég e számolás közben valamilyen gyors külső beavatkozással megzavarni a program működését! Tegyük fel, hogy x_p korrekt, de $\bar{x}_q$ nem, azaz a belőlük kiszámított $\bar{x}$ sem korrekt, nevezetesen

$$\bar{x}^e \equiv y \pmod{p}, \text{ de } \bar{x}^e \not\equiv y \pmod{q}.$$

Ebből pedig következik, hogy $\gcd(N, \bar{x}^e - y) = p$. Ezzel pedig hozzájutottunk a kiskapu-információhoz.

1.6. Ciklikus csoportok az elliptikus görbéken

Az eddigi számításainkat, akár a moduláris aritmetikára, akár a diszkrét logaritmus problémára gondolunk, mindig valamely ciklikus csoportban végeztük. A ciklikus csoportok egy kriptográfiai szempontból különösen fontos osztályával ismerkedünk meg, melyek az elliptikus görbék segítségével definiálhatók. Fontosságukat az adja, hogy a diszkrét logaritmus problémára e csoportban eddig nem ismerünk szubexponenciális algoritmust, míg a moduláris aritmetika ciklikus csoportjaiban igen.

Elliptikus görbék Legyen $\mathbb{F}$ egy test, melynek karakterisztikája nem 2 és nem 3.[6] Tipikus alkalmazásokban e test az $\mathbb{F}_p$ véges test, ahol $p > 3$ prím.

Legyen $a, b \in \mathbb{F}$ olyan, hogy $4a^3 + 27b^2 \neq 0$. Az

$$y^2 = x^3 + ax + b \tag{6}$$

egyenletet kielégítő $(x, y) \in \mathbb{F} \times \mathbb{F}$ pontok, valamint egy végtelen távolinak nevezett $\mathcal{O}$ pont halmaza elliptikus görbét alkot[7]. A $4a^3 + 27b^2 \neq 0$ kikötés azt biztosítja, hogy az elliptikus görbén ne legyen ún. szinguláris pont, azaz hogy az $x^3 + ax + b = 0$ egyenletnek ne legyen többszörös gyöke.

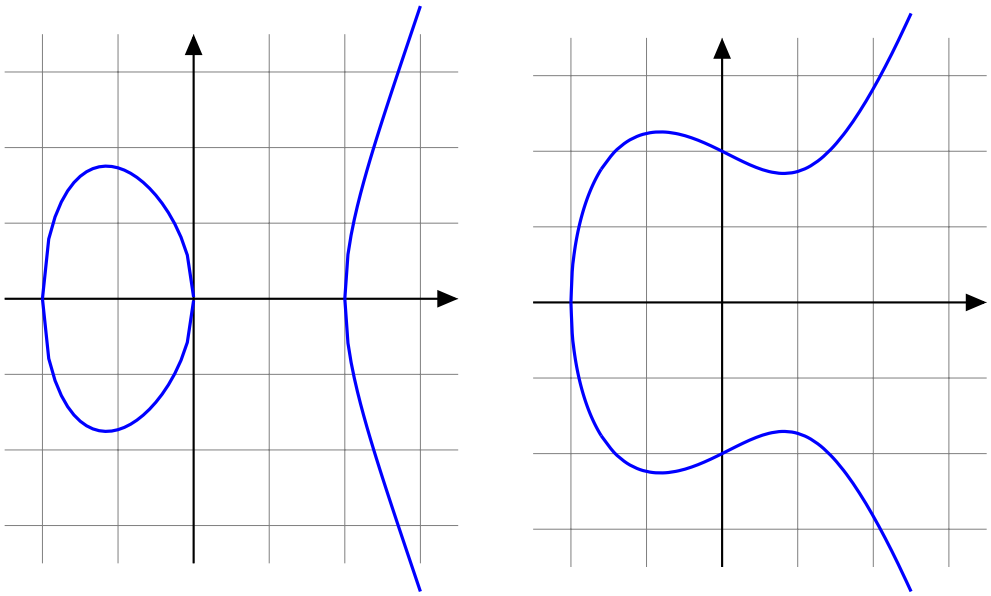
Először két valós test fölötti elliptikus görbe ábráját mutatjuk (ld. 1. ábra), majd három $\mathbb{F}_{11}$ fölötti görbéjét, melyek pontjait 11×11-es négyzetrácson ábrázolunk (ld. 2. ábra). Az utóbbi ábrán látható piros pontok a végtelen távoli $\mathcal{O}$ pontot ábrázolják, a pontok közt futó vonalak magyarázatát később adjuk meg.

Az $y^2 = x^3 + ax + b$ egyenletből világos, hogy a függvény grafikonja valós esetben szimmetrikus az x-tengelyre. Ha véges test fölötti elliptikus görbét egy négyzetrácson ábrázolunk, akkor ez a szimmetriatulajdonság megmarad, amennyiben $\mathbb{Z}_p$ elemeit a $\{\frac{p-1}{2}, \ldots, -1, 0, 1, \ldots, \frac{p-1}{2}\}$ elemekkel reprezentáljuk.

A végtelen távoli pont: affin és projektív koordináták A projektív sík legyszerűbben úgy modellezhető, hogy a 3-dimenziós tér origón átmenő egyeneseit tekintjük a projektív sík pontjainak, és az origón átmenő síkokat a projektív sík egyeneseinek. Egy origón átmenő egyenest bármely irányvektorával jellemezhetünk, tehát az $[X, Y, Z]$ vektor és a $[cX, cY, cZ]$ vektor $(c \neq 0)$ ugyanazt az egyenest – vagyis a projektív síknak ugyanazt a pontját – határozza meg. Minden $[X, Y, Z]$ vektornak megfelel egy projektív pont, kivéve a nullvektort. Ha $Z \neq 0$, akkor végigosztva vele, az $[X/Z, Y/Z, 1]$ vektort kapjuk, mely a $Z = 1$ egyenletű sík egy pontjába mutat, vagyis ezek a pontok megfeleltethetők egy affin sík pontjainak. Ezt az $[x, y, 1] \leftrightarrow (x, y)$ megfeleltetéssel fejezzük ki. Ha egy projektív pont koordinátás alakja $[X, Y, 0]$, akkor

[6] Az elliptikus görbék tárgyalása némileg különbözik e két karakterisztikára. Ugyan a 2-karakterisztikájú, tehát 2-hatvány elemű testekre épített elliptikus görbéket is használják kriptográfiai célokra, az utóbbi időben ellenük talált bizonyos támadások okán ezeket itt nem fogjuk tárgyalni.

[7] Elliptikus görbéket más alakú egyenletekből is kaphatunk, nekünk azonban ez a speciális, ún. Weierstrass-alak mindenre elég lesz.



1. ábra. Valós test fölötti $y^2 = x^3 - 4x$ és $y^2 = x^3 - 2x + 4$ egyenletű elliptikus görbék grafikonjai

a $Z = 1$ egyenletű sík egyik pontja sem feleltethető meg e pontnak: az ilyen pontokat nevezzük végtelen távoli vagy ideális pontoknak. Ha az

$$y^2 = x^3 + ax + b \tag{7}$$

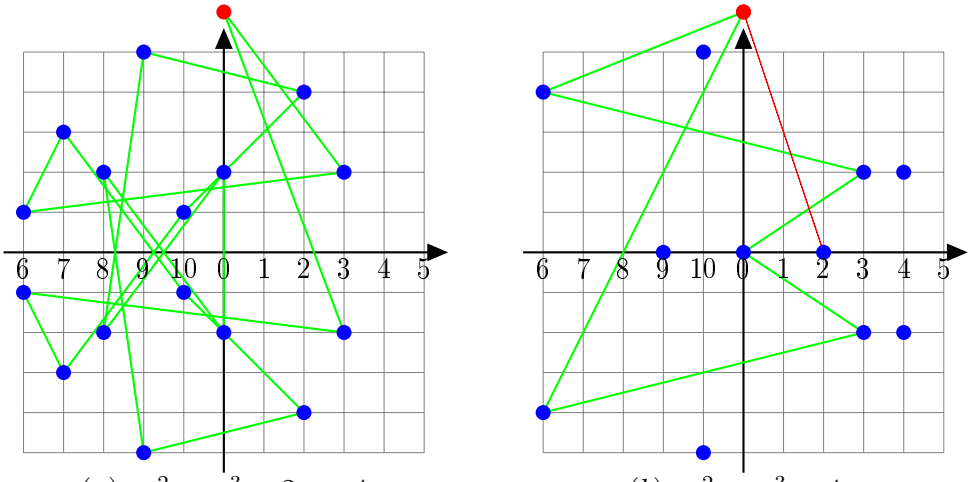
egyenlet helyett az

$$Y^2 Z = X^3 + aXZ^2 + bZ^3 \tag{8}$$

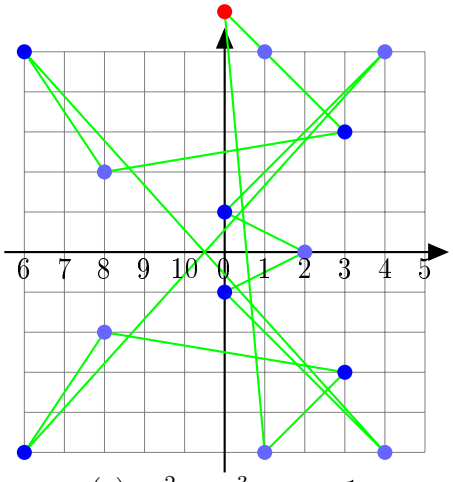
egyenletet tekintjük, akkor $Z \neq 0$ esetén, Z^3-bel való leosztás után az

$$\left(\frac{Y}{Z}\right)^2 = \left(\frac{X}{Z}\right)^3 + a\left(\frac{X}{Z}\right) + b$$

egyenlethez jutunk, ami az $\frac{X}{Z} = x$, $\frac{Y}{Z} = y$ helyettesítés után megegyezik a (7) egyenlettel. Ha viszont $Z = 0$, akkor X-nek is 0-nak kell lennie, így a $[0, Y, 0]$ vektoroknak ($Y \neq 0$) megfelelő végtelen távoli pont is kielégíti a (8) egyenletet. Ezt a végtelen távoli pontot $\mathcal{O}$-val jelöljük, és úgy tekintjük, hogy ez is rajta van a (7) egyenlettel megadott elliptikus görbén.



(a) $y^2 = x^3 + 2x + 4$ (b) $y^2 = x^3 - 4x$



(c) $y^2 = x^3 + x + 1$

2. ábra. Az $\mathbb{F}_{11} = \mathbb{Z}_{11}$ test fölötti három különböző elliptikus görbe grafikonja: (*a*) a görbének 17 pontja van (a pontművelet csoportja 17 elemű ciklikus), (*b*) a görbe 12-elemű (csoportja egy 6 elemű ciklikus és egy 2-elemű ciklikus csoport direkt szorzata), (*c*) a görbe 14-elemű (csoportja ciklikus, melynek van egy nagy prímelemű – nevezetesen 7-elemű – ciklikus részcsoportja). Mindhárom ábrán a tengelyeket -5-től 5-ig számoztuk, de mivel $\mathbb{Z}_{11}$ elemeit szívesebben jelöljük a 0-tól 10-ig terjedő számokkal, a tengelyeken is e számokat használtuk, tehát $-5 = 6, \ldots, -1 = 10$. Az ábrán látható piros pontok a végtelen távoli pontot jelölik, a pontokat összekötő színes vonalak a pontok egy ciklikus sorrendjét adják meg a következőkben bevezetendő pontműveletre nézve.

Például az $\mathbb{F}_{11}$ fölötti $y^2 = x^3 + 2x + 4$ egyenletű elliptikus görbe ponthalmaza (2. (a) ábra) affin koordinátás alakú pontokkal megadva:

$$\mathcal{O} \cup \{(3,2),(6,1),(7,3),(10,10),(2,7),(9,6),(8,2),(0,9),\\ (0,2),(8,9),(9,5),(2,4),(10,1),(7,8),(6,10),(3,9)\}.$$

Ugyanennek a görbének a pontjai projektív koordinátákkal – a pontokat azonos sorrendben felsorolva:

$$\{[0,1,0],[3,2,1],[6,1,1],[7,3,1],[10,10,1],[2,7,1],[9,6,1],[8,2,1],[0,9,1],\\ [0,2,1],[8,9,1],[9,5,1],[2,4,1],[10,1,1],[7,8,1],[6,10,1],[3,9,1]\}.$$

Pontművelet az elliptikus görbén Az elliptikus görbe egy szép tulajdonsága, hogy minden egyenes, mely két pontban metszi (a metszetszámot multiplicitással számolva), metszi azt egy harmadik pontban is. A 3. ábra bal grafikonja azt mutatja, hogy ha P és Q a görbe két különböző pontja, akkor a rajtuk átmenő egyenesen lévő harmadikat R-rel jelölve, a P és Q összegén ennek x-tengelyre való tükörképét fogjuk jelölni. Ha $P = Q$, akkor az összekötő egyenesen az érintőt kell érteni. Két olyan pont, melyek egymás tükörképei, a végtelen távoli pontban metszik a görbét, mely megegyezik az y-tengelyen lévő ideális ponttal.

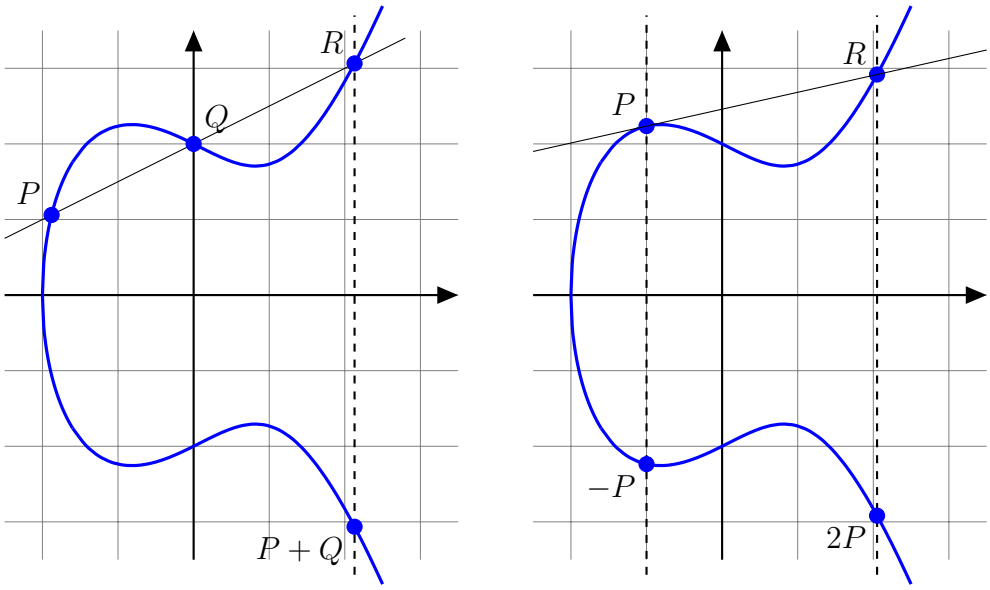
A klasszikus algebrai geometria elemi ismeretei közé tartozik, hogy e művelettel az elliptikus görbe pontjai kommutatív csoportot alkotnak, azaz e művelet kommutatív, asszociatív, invertálható, zéruseleme az $\mathcal{O}$ pont. Az asszociativitás geometriai eszközökkel szépen igazolható, a többi azonosság bizonyítása egyszerű, de itt mellőzzük.

Legyen $P = (x_1, y_1)$ és $Q = (x_2, y_2)$ az $\mathcal{E}$ elliptikus görbe két tetszőleges affin pontja. Három esetet különböztethetünk meg:

1. $x_1 \neq x_2$,
2. $x_1 = x_2$ és $y_1 = -y_2$,
3. $x_1 = x_2$ és $y_1 = y_2$.

Az 1. esetben legyen a két ponton átmenő egyenes egyenlete $y = \mu x + \nu$. Az egyenes iránytangense

$$\mu = \frac{y_2 - y_1}{x_2 - x_1},$$



3. ábra. $P+Q$, $2P=P+P$ és $-P$ megszerkesztése

és

$$\nu = y_1 - \mu x_1 = y_2 - \mu x_2.$$

Az elliptikus görbe és az egyenes metszéspontja kielégíti az

$$(\mu x + \nu)^2 = x^3 + ax + b$$

egyenletet, azaz harmadfokú egyenletalakba rendezve

$$x^3 - \mu^2 x^2 + (a - 2\mu\nu)x + b - \nu^2 = 0.$$

A gyökök és együtthatók összefüggése szerint a gyökök összege μ^2 (az x^2 együtthatójának -1-szerese), és mivel a másik két gyök x_1 és x_2, ezért a harmadik

$$x_3 = \mu^2 - x_1 - x_2.$$

Ha $P+Q$ koordinátás alakja (x_3, y_3), akkor az egyenesen az $(x_3, -y_3)$ pont van, ahonnan

$$\mu = \frac{-y_3 - y_1}{x_3 - x_1},$$

ahonnan

$$y_3 = \mu(x_1 - x_3) - y_1.$$

A 2. esetben $P + Q = \mathcal{O}$. Ez az elliptikus görbe projektív alakjából könnyen ellenőrizhető, ennek részletezését mellőzzük.

Az utolsó, 3. esetben $P = Q$, így az érintő egyenletére van szükségünk. Valós test fölött $\mu = y'$, de a deriválási szabályokat formálisan értelmezve ugyanez a módszer működik véges testek fölött is. A 7 egyenlet deriválásával kapjuk, hogy

$$2yy' = 3x^2 + a.$$

P koordinátáit behelyettesítve kapjuk, hogy

$$\mu = \frac{3x_1^2 + a}{2y_1}.$$

Innen a többi formula azonos az első esetben megkapottakkal.

Végül már csak azokat az eseteket kell áttekinteni, amikor legalább az egyik pont a végtelen távoli $\mathcal{O}$ pont. A pontok projektív alakjából azonnal adódnak a következő összefüggések:

$$\mathcal{O} + \mathcal{O} = \mathcal{O}$$
$$P + \mathcal{O} = P, \text{ azaz bármely } (x, y) \in \mathcal{E} \text{ esetén } (x, y) + \mathcal{O} = (x, y)$$
$$P - P = \mathcal{O}, \text{ azaz bármely } (x, y) \in \mathcal{E} \text{ pontra } -(x, y) = (x, -y)$$

Az utóbbi összefüggést az $\{0, 1, \ldots, p-1\}$ számokkal reprezentált $\mathbb{F}_p$ test esetén úgy kell érteni, hogy $-(x, y) = (x, p - y)$.

Azt kapjuk tehát, hogy az $\mathcal{E}$ elliptikus görbe pontjai a fönt bevezetett pontművelettel additív csoportot alkotnak, amit szokás az $\langle \mathcal{E}, + \rangle$ jelöléssel kifejezni. Ráadásul láttuk, hogy e művelet eredménye igen egyszerű képletekkel igen gyorsan számolható.

Az elliptikus görbe csoport E csoport mérete és struktúrája is meghatározható. Méretére a híres – nem triviális – Hasse-tétel ad becslést, mely szerint az $\mathbb{F}_p$ fölötti $\mathcal{E}$ elliptikus görbe pontjainak száma:

$$p + 1 - 2\sqrt{p} \leqslant |\mathcal{E}| \leqslant p + 1 + 2\sqrt{p}.$$

A pontos érték a hatékony Schoof-algoritmussal számolható.

Ha $\mathcal{E}$ egy $\mathbb{F}_p$ fölötti elliptikus görbe ($p > 3$), akkor a rajta definiált csoporthoz létezik két olyan n_1 és n_2 szám, hogy

$$\langle \mathcal{E}, + \rangle \cong \mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2},$$

ahol $n_2 \mid n_1$ és $n_2 \mid (p-1)$.

A csoport struktúrájára tehát néhány különböző esetet tudunk fölsorolni. Ha $n_2 = 1$, akkor a csoport ciklikus! Szerencsés esetben akár prímrendű, ami azért érdekes, mert a diszkrét logaritmus problémához prímrendű csoportot használunk. A 2. ábrán az első görbe 17 pontos, így csoportja izomorf $\mathbb{Z}_{17}$-tel. Elemeit az egyik ciklikus sorrend szerint fölsorolva:

$$\begin{aligned}\mathbb{Z}_{17} \cong \mathcal{E} = \{&\mathcal{O},(3,2),(6,1),(7,3),(10,10),(2,7),(9,6),(8,2),(0,9),\\ &(0,2),(8,9),(9,5),(2,4),(10,1),(7,8),(6,10),(3,9)\},\end{aligned}$$

azaz $(3,2)$ egy generátorelem.

A harmadik elliptikus görbe, melynek $y^2 = x^3 + x + 1$ az egyenlete, 14 pontú, csoportja izomorf $\mathbb{Z}_{14}$-gyel, egyik generátoreleme $(1,6)$:

$$\begin{aligned}\mathbb{Z}_{14} \cong \mathcal{E} = \{&\mathcal{O},(1,6),(3,8),(8,9),(6,6),(4,5),(0,1),(2,0),\\ &(0,10),(4,6),(6,5),(8,2),(3,3),(1,5)\},\end{aligned}$$

E csoportból kiválasztható egy 7-edrendű G részcsoport:

$$\mathbb{Z}_7 \cong G = \{\mathcal{O},(3,8),(6,6),(0,1),(0,10),(6,5),(3,3)\}, \tag{9}$$

Egy primitív elemet és azok többszöröseit egy vékony zöld vonal mentén végig követhetjük a végtelen távoli pontból, azaz a zéruselemből indulva, és végül oda érkezve. (Emlékeztetünk rá, hogy e csoport additív, azaz a művelet az összeadás, nem a szorzás, így a generátorelemnek nem a hatványait, hanem a többszöröseit kell számolnunk. A moduláris hatványozás technikája minden probléma nélkül átvihető skalárral szorzásra.) E példában a 14-edrendű csoport 7-edrendű részcsoportjának elemeit kékkel jelöltük, a többit halványabb színnel különböztettük meg.

A második grafikonon látható $y^2 = x^3 - 4x$ egyenletű elliptikus görbe két ciklikus csoport direkt szorzata, az egyik 2-elemű, a másik 6-elemű:

$$\begin{aligned}\mathcal{E} &\cong \mathbb{Z}_2 \times \mathbb{Z}_6,\\ \mathbb{Z}_2 &\cong \{\mathcal{O},(2,0)\},\\ \mathbb{Z}_6 &\cong \{\mathcal{O},(4,2),(4,9),(9,0),(10,5),(10,6)\},\end{aligned}$$

és valóban, $2 \mid 12$ és $2 \mid 6$ (a 2-elemű csoportot piros vonal jelzi).

Erős és biztonságos prímek Az összes konstrukcióban fontos szerepet játszik a prímek megfelelő kiválasztása. Bármelyik rendszer elleni jövőbeni támadás lehetősége függhet a megfelelő prímek kiválasztásától. Az ismert támadások – akár a faktorizációs, akár a diszkrét logaritmus ellen – közt vannak olyanok, amelyek sikerét csökkenti, ha a prím úgynevezett biztonságos vagy ha erős prím.

A p prím **biztonságos** (*safe prime*), ha $p = 2q + 1$ alakú, ahol q is prím. Ezek a diszkrét logaritmus problémában fontosak. Az 500 alatti biztonságos prímek: 5, 7, 11, 23, 47, 59, 83, 107, 167, 179, 227, 263, 347, 359, 383, 467, 479 (a sorozat sorszáma a sorozatok OEIS katalógusában A005385)

A p **erős prím**, ha

1. $(p-1)$-nek van egy nagy prímfaktora, tehát $p = ap_1 + 1$, ahol p_1 nagy prím, és a pozitív egész,

2. $(p_1 - 1)$-nek is van nagy prímfaktora, azaz $p_1 = bp_2 + 1$, ahol p_2 nagy prím, és b pozitív egész,

3. $p+1$-nek is van nagy prímfaktora, azaz $p = cp_3 - 1$, ahol p_3 nagy prím, és c pozitív egész.

A biztonságos prímek és az erős prímek kapcsolata világos, hisz $p = 2q+1$ esetén az erős prímekre vonatkozó első feltétel teljesül.

A faktorizáció elleni támadásokban valójában nem nyújt nagy segítséget, ha a két prím erős prím, mert bár a Pollard ρ algoritmus ellen véd, de az újabb és erősebb számtest szita, és a Lenstra elliptikus görbe faktorizációs módszere ellen nem. Mindennek ellenére akad olyan szabvány, mely az RSA-hoz erős prímek választását javasolja, általában azonban a szabványok ezt nem írják elő, úgy tűnik, szükségtelen RSA-hoz erős prímeket választani.

A diszkrét logaritmus probléma esetén más a helyzet. Pohling–Hellman-tétele szerint, ha $p-1$ minden prímosztója kicsi, akkor polinom időben kiszámolható a diszkrét logaritmus. Ez annak következménye, hogy a Pohling–Hellman- vagy Silver–Pohling–Hellman-algoritmus futási ideje az $n = \prod_i p_i^{e_i}$ számra $O\left(\sum_i e_i(\log n + \sqrt{p_i})\right)$. Így minden diszkrét logaritmus problémára épülő kriptográfiai rendszer esetén legalább $p-1$-nek kell, hogy legyen nagy prímosztója. A gyakorlatban biztonságos prímeket szoktak választani.

A SECG csoport keretében a Certicom cég által a paraméterválasztáshoz adott javaslatok szerint a diszkrét logaritmus problémára épülő elliptikus görbéhez olyan p választandó, melyre $p-1$ és $p+1$ legnagyobb q prímfaktora ki kell elégítse a $\log_p(q) > \frac{19}{20}$ feltételt.

2. A nyílt kulcsú titkosítás alapfogalmai

2.1. Előzmények, kezdetek

Évszázadokon át a szimmetrikus kulcsú titkosítás volt maga a titkosítás, ahol a két kommunikáló fél mindegyikének birtokolnia és őriznie kellett a kulcsot. E kulcs cseréje és őrzése sok nehézséget okozott és okoz ma is. Ezek megoldására az első próbálkozások az 1970-es években születtek. Az első aszimmetrikus algoritmust valószínűleg 1973-ban – az angol titkosszolgálatnál (Government Communications Headquarters, GVHQ) – James H. Ellis, Clifford Cocks és Malcolm Williamson készítette el. Ez az információ csak 1997-ben került nyilvánosságra.

1974-ben egy Merkle nevű diák egy szemináriumon olyan ötlettel állt elő, mely lehetővé tenné a kulcscserét nyilvános csatornán. A remek ötlet gyengéje az volt, hogy a kulcscserét végzők számítási igényének a négyzetével arányos idő alatt egy hallgatózó támadó meg tudta szerezni a kulcsot, és ez még nem tűnt elég biztonságosnak. Az ötletet azonban Diffie és Hellman továbbfejlesztette, algoritmusukban már a négyzetes helyett exponenciális volt a két igény közti függvénykapcsolat. E kulcscsere-protokollokban megjelent az aszimmetria! Innen már csak egyetlen lépés volt az aszimmetrikus kulcsú titkosítás titkosszolgálaton kívüli feltalálása.

Merkle's puzzle Merkle ötlete a következő volt: Aliz Bobbal kíván kommunikálni, amihez közös kulcsot kell találni. Aliz nagy számú, mondjuk 2^{30} darab kulcsot generál, és mindegyiket titkosítva elrejti egy üzenetben: „Az i-edik kulcs k_i", ahol az i sorszám 1-től 2^{30}-ig változik. A titkosítás azonban annyira gyenge, hogy bármelyik feltörhető belátható időn belül, mondjuk néhány perc alatt! Ezután a különböző kulcsokkal titkosított 2^{30} darab szöveget Aliz véletlen sorrendben elküldi Bobnak, aki véletlenül választ közülük egyet, azt feltöri, és a nyilvános csatornán visszaküldi Aliznak i értékét. A közös kulcs k_i lesz. A támadónak átlagosan a feladványok felét fel kell törnie, hogy megtudja, melyik az i-edik üzenet és ezzel megtudja a k_i kulcsot.

Diffie-Hellman kulcscsere Merkle ötletét tovább fejlesztve Diffie és Hellman a következő igen egyszerű protokollt találta ki:

2.1. konstrukció (Diffie–Hellman kulcscsere)**.** Adva van az 1^n biztonsági paraméter.

1. Aliz: egy véletlen polinomidejű $\mathcal{G}$ algoritmussal előállít egy p-edrendű G ciklikus csoportot és annak egy g generátorelemét: $(G, p, g) \leftarrow \mathcal{G}(1^n)$. Egyenletes eloszlás szerint választ egy véletlen $a \leftarrow \mathbb{Z}_q$ elemet, és kiszámítja a $h_1 = g^a$ elemet, végül üzenetet küld Bobnak

   $$\text{Aliz} \rightarrow \text{Bob}: \ (G, p, g, h_1).$$

2. Bob: fogadja Aliz üzenetét, választ egy véletlen $b \leftarrow \mathbb{Z}_q$ elemet, kiszámítja $h_2 = g^b$ és $k_B = h_1^b$ értékét, végül üzenetet küld Bobnak:

   $$\text{Bob} \rightarrow \text{Aliz}: \ h_2$$

3. Aliz: fogadja Bob üzenetét, kiszámolja $k_A = h_2^a$ értékét.

A protokoll lényege, hogy az Aliz és Bob közt zajló beszélgetést passzívan hallgató támadó ismeri (G, p, g, h_1, h_2) elemeit, azonban ha ezekre a CDH-probléma nehéz, nem tudja kiszámolni a

$$k_A = h_2^a = (g^b)^a = g^{ab} = (g^a)^b = h_1^b = k_B$$

értéket, amely Aliz és Bob közös kulcsa lesz a további kommunikációhoz. Ennél több is igaz! Végezzük el azt a kísérletet a hallgatózó támadóval, hogy miután végighallgatta Aliz és Bob kulcscsere-kommunikációját, kap egy kulcsot, mely $\frac{1}{2}$ valószínűséggel egy véletlen bitsorozat, $\frac{1}{2}$ valószínűséggel a $k_A = k_B$ kulcs. A támadó előnye a kulcscsereprotokollal szemben $\nu(n)$, ha $\frac{1}{2}$-től $\nu(n)$-nel eltérő valószínűséggel el tudja találni, hogy melyik a valódi kulcs. Biztonságosnak fogjuk nevezni a Diffie-Hellman kulcscserét, ha bármely $\mathcal{A}$ hallgatózó támadóra ez az előny elhanyagolható.

2.2. tétel. *Ha a DDH probléma nehéz, akkor a Diffie-Hellman kulcscsere protokoll biztonságos egy hallgatózó jelenlétében.*

E protokollt szellemessége ellenére nem használják, mert ugyan a passzív hallgatózó ellen véd, de nem véd az aktív támadó ellen, aki képes Aliz és Bob üzeneteit fogadni, és magát Aliz számára Bobnak, Bob számára Aliznak kiadni. E támadás neve man-in-the-middle. Tehát e támadás tömören leírva:

1. Aliz: $(G, p, g) \leftarrow \mathcal{G}(1^n)$, $a \leftarrow \mathbb{Z}_q$, $h_1 = g^a$,

   $$\text{Aliz} \rightarrow \text{Támadó}: \ (G, p, g, h_1).$$

2. Támadó: $x \leftarrow \mathbb{Z}_q$, $h_x = g^x$, $k_A = h_1^x$

Támadó $\rightarrow$ Bob: (G, p, g, h_x).

3. Bob: $b \leftarrow \mathbb{Z}_q$, $h_2 = g^b$, $k_B = h_x^b = g^{bx}$

   Bob $\rightarrow$ Támadó: h_2

4. Támadó: $y \leftarrow \mathbb{Z}_q$, $h_y = g^y$, $k_B = h_1^x = g^{bx}$, $k_A = h_1^y = g^{ay}$

   Támadó $\rightarrow$ Aliz: h_y

5. Aliz: $k_A = h_y^a = g^{ay}$.

Így a Támadónak van Alizzal és Bobbal is egy-egy közös kulcsa, a köztük lévő kommunikációt kedvére módosíthatja magát Aliz felé Bobnak, Bob felé Aliznak kiadva.

2.2. Aszimmetrikus rejtjelező egyirányú kiskapu-permutációból

El kell oszlatnunk egy közkeletű tévedést! Az egyirányú kiskapufüggvények önmagukban nem használhatók titkosításra úgy, hogy az egyirányú függvény a rejtjelező, a kiskapu-információ a titkos kulcs és a függvény inverze a dekódoló. De más nehézségekkel is szembe kell nézni nyílt kulcsú rejtjelező konstrukciójakor.

1. Egy aszimmetrikus rejtjelező E függvénye nem lehet determinisztikus, hisz az azonnal tönkreteszi a szemantikai biztonságot: ha a támadó ismeri az m_0 és m_1 sztringeket, akkor ki tudja számolni az $\mathrm{E}_{pk}(m_i)$ értékeket is, így el tudja dönteni, hogy melyikük egyezik meg a $c = \mathrm{E}_{pk}(m_b)$ kriptoszöveggel. Tehát E-nek véletlen algoritmusnak kell lennie!

2. Mivel az egyirányú kiskapufüggvények determinisztikusak, ha belőlük akarunk aszimmetrikus rejtjelező E algoritmust konstruálni, az argumentumuknak kell véletlennek lennie, tehát nem az üzenetre kell őket alkalmazni, hanem vagy egy véletlenül választott számra kell őket alkalmazni, vagy az üzenetek mellé véletlen üzenetet kell paddingbe tenni.

3. Abból, hogy az f egyirányú kiskapufüggvény nehezen invertálható, nem következik, hogy az $y = f(x)$ ismeretében az x-ről semmi információ nem szerezhető meg. Így az sem elég, hogy E nem determinisztikus! Valahogy az x-ről az f ismeretén keresztül megszerezhető információt is el kell rejteni, amihez ugyancsak a véletlent lehet segítségül hívni!

Első próbálkozásként próbáljunk meg egy egyirányú kiskapu-permutációt használva rejtjelezőt konstruálni, melyben a tökéletes biztonságot nyújtó *one time pad* (OTP) is visszaköszön.

2.3. konstrukció (Aszimmetrikus rejtjelező)**.** Legyen $(\mathcal{G}^{\mathsf{TDP}}, f, f^{-1})$ az 1.31. definíció szerinti kiskapu-permutációk egy családja. Definiáljuk a $\Pi = (\mathcal{G}, \mathrm{E}, \mathrm{D})$ aszimmetrikus kulcsú, az 1.12. definíciónak megfelelő rejtjelezőt a következőképp:

1. $\mathcal{G}$: $(k, t, \mathcal{D}_k) \leftarrow \mathcal{G}^{\mathsf{TDP}}(1^n)$, azaz választunk az egyirányú kiskapu-permutációk családjából egy (k, t) indexpárt, és megadjuk az értelmezési tartományt. $\mathcal{G}$ választ egy $H : \mathcal{D}_k \to \mathcal{D}_k$ hash függvényt, és publikálja a $pk = (k, f_k, H)$ nyílt kulcsot. A titkos kulcs az $sk = (t, f_t^{-1})$ lesz.

2. E: $r \leftarrow \mathcal{D}_k$, $\mathrm{E}_{pk}(m) = (f_k(r), H(r) \oplus m)$, tehát $\mathrm{E}_{pk} : \mathcal{D}_k \times \mathcal{D}_k \to \mathcal{D}_k \times \mathcal{D}_k$.

3. D: $\mathrm{D}_{sk}(c_1, c_2) = H(f_t^{-1}(c_1)) \oplus c_2$.

Könnyen ellenőrizhető, hogy $\mathrm{D}_{sk}\,\mathrm{E}_{pk}(m) = m$, ugyanis $r = f_t^{-1}(c_1)$ és $H(r) \oplus c_2 = m$.

Látható, hogyan épül e konstrukcióba korábban szerzett tudásunk: az egyirányú függvényt egy véletlenül választott elemre használjuk. E ponton folytathatnák a rejtjelezést úgy is, hogy a nyílt szöveget az OTP-hez hasonló módon $r \oplus m$-mel rejtjeleznénk. Ez tökéletes biztonságot nyújtana, ha nem itt és most kéne az r-et is átadni. Azt csak úgy tudjuk átadni, hogy elhanyagolható eséllyel megszerezhető, ezt nyújtja az egyirányú f_k kiskapu-permutáció. Csakhogy láttuk, az egyirányú permutációk nem biztonságosak abban az értelemben, hogy semmi információ ne lenne megszerezhető az argumentumáról, azaz jelen esetben r-ről. (Pl. láttuk, az RSA-függvény értékéből megmondható az argumentumának Jacobi-jele.) Ezért az r használata sem nyújt elég biztonságot az OTP-ben való használathoz, ezért egy „véletlen” függvénnyel el kell rejteni még azt a kevés információt r-ről, amit egy támadó megszerezhet róla. Erre szolgál a H hash-függvény. Világos, hogy az egész rendszer biztonsága az egyirányú kiskapu-permutáció biztonságától (az inverz kiszámításának nehézségétől) és a hash függvény véletlenszerűségétől függ.

Véletlen orákulum A véletlen orákulum modell kriptográfiai rendszerek, protokollok biztonságának megfogalmazásában és bizonyításában segít. Azt fejezi ki, hogy a rendszer az adott biztonsággal rendelkezik, amennyiben a véletlen orákulumot helyettesítő függvény valóban „elég véletlenszerű" módon működik. E fogalom használata azért terjedt el, mert egyrészt

- a mai kriptográfiai rendszerek nagy részének biztonsága nem bizonyítható precízen, mivel olyan számelméleti feltevésekre épül, amelyek mindmáig bizonyítatlanok, másrészt
- számtalan olyan kriptográfiai protokoll született/születik, amelyben később valaki biztonsági rést talál, azaz kiderül, hogy a rendszer könnyen támadható, de a támadás nem a bizonyítatlan számelméleti feltevések valamelyikének összeomlására épül, hanem egyszerűen a konstrukció más elemei voltak rosszul kitalálva.

Mindezek miatt nagy szükség van arra, hogy valahol a „precízen, minden kétséget kizáróan" bizonyított biztonság és a „megérzésre biztonságos" között találjunk olyan matematikailag precízen bizonyítható állításokat, amelyek pontosan megfogalmazzák a bizonyított biztonság feltételeit. E feltételek egyike a nyílt kulcsú titkosításban használt makacs számelméleti probléma nehézsége! Világos, amint egy ilyen problémára találnak hatékony algoritmust, a rá épülő kriptográfiai rendszerek összeomlanak. A másik fontos, és gyakran előforduló feltétel, az alkalmazott véletlen függvények, tipikusan a hash függvények „közelsége" a valódi véletlenhez. E feltétel kikerülésére használjuk a véletlen orákulum alkalmazását. Ez úgy képzelhető el, hogy van egy olyan képzeletbeli résztvevője a protokollnak, „aki" minden neki küldött x számra válaszul küld egy véletlen r számot, azonban minden üzenetváltást följegyez, így ha később ugyanaz az x érkezik, mint egyszer már korábban, válaszul ugyanazt az r-et küldi. Mintha mindent följegyezne, és minden válasz előtt megnézné a nagykönyvben, hogy nem kapta-e már valakitől korábban ugyanezt a számot.

A véletlen orákulum modell legfőbb gyengéje, hogy a valóságban a véletlen orákulum helyén egy hash függvény van, amely nem csak hogy determinisztikus, de többnyire a támadó előtt ismert, ezért előre tudhatja annak válaszát bármely bemenetre, sőt elemezheti annak programkódját is! Mindennek ellenére mára alapvetővé vált, hogy a kriptográfiai protokollok biztonsága legalább ilyen szinten bizonyítva legyen.

A nyílt kulcsú titkosítás fönti definíciójáról a következő tétel bizonyítható:

2.4. tétel. *Amennyiben az egyirányú kiskapu-permutációk* $(\mathcal{G}^{\mathsf{TDP}}, f, f^{-1})$ *családjában az inverz kiszámítása a kiskapu-információ ismerete nélkül nehéz, és a 2.3. konstrukcióban definiált* $\Pi = (\mathcal{G}, \mathrm{E}, \mathrm{D})$ *rejtjelezőben* H*-t véletlen orákulummal helyettesítjük, akkor* Π *szemantikailag biztonságos. Egyszerűen fogalmazva: a fenti rejtjelező szemantikailag biztonságos a véletlen orákulum modellben.*

Bizonyítás. Megmutatjuk, hogy ha a véletlen orákulum modellben létezik egy olyan $\mathcal{A}$ véletlen polinomidejű algoritmus, mely nem elhanyagolható valószínűséggel cáfolja Π szemantikai biztonságát, akkor létezik olyan hatékony $\mathcal{B}$ algoritmus is, mely nem elhanyagolható valószínűséggel invertálja f-et, azaz f nem egyirányú. Megmutatjuk tehát, hogy ha van olyan $\mathcal{A}$, melyre

$$\mathsf{Adv}^{\mathsf{indCPA}}_{\mathcal{A},\Pi}(n) = \left| \mathbb{P}[\mathsf{Exp}^{\mathsf{indCPA}}_{\mathcal{A},\Pi}(n) = 1] - \frac{1}{2} \right| = \nu(n)$$

nem elhanyagolható, akkor van olyan $\mathcal{B}$ algoritmus, melyre

$$\mathbb{P}[f(\mathcal{B}(1^n, f, y)) = f(x)] \geqslant \nu(n).$$

E bizonyításhoz magunk definiálunk egy orákulumot, melyet a H helyébe helyettesítünk. Ezt az algoritmust $\mathcal{H}^{\mathsf{RO}}$ fogja jelölni (RO a *random oraculum* kifejezésből).

Legyen tehát m_0 és m_1 a két üzenet, melyekre $|m_0| = |m_1|$. Legyen $\mathrm{E}_{pk}(m_b) = (c_1, c_2)$, ahol b egy véletlen bit. Az $\mathcal{A}$ algoritmus az $\mathsf{Exp}^{\mathsf{indCPA}}_{\mathcal{A},\Pi}(n)$ nyílt szöveg megkülönböztethetetlenségi kísérlet közben sokszor meghívja a H-t, vagyis e modellben lefuttatja a $\mathcal{H}^{\mathsf{RO}}$ algoritmust.

A $\mathcal{H}^{\mathsf{RO}}$ algoritmus egy listában följegyez minden hozzá beérkező r értéket, ennek a listának a neve RL lesz, míg a visszaadott $\mathcal{H}^{\mathsf{RO}}(r)$ értékeket egy másik, HL nevű listába jegyzi. Így valóban orákulumként fog tudni viselkedni, hisz emlékezni fog minden korábbi hívására. Ha véletlenül egy olyan r-et kap, melyre $\mathcal{H}^{\mathsf{RO}}(r) = y$, és r még nem szerepel az RL listában, akkor olyan értéket fog visszaadni, melyre a (c_1, c_2) pár véletlenszerűen vagy az m_0 vagy az m_1 valamelyikének titkosított változata lesz. Egyébként mindig véletlen választ ad, tehát valóban véletlen orákulumként viselkedik.

Végül meg kell mutatnunk, hogy a $\mathcal{B}$ algoritmus legalább ν valószínűséggel megtalálja y inverzét. Megbecsüljük annak valószínűségét, hogy $\mathcal{A}$ sikeres

```
function B(1^n, f, y)
    global RL, HL, h   (az RL és HL listák aktuális hossza h)
    function H^RO(r)
        i = 1
        InRL = FALSE
        while i ⩽ h AND NOT InRL do
            if RL[i] = r then
                InRL = TRUE
            else
                i = i + 1
            end
        end
        if InRL then
            return HL[i]
        end
        if f(r) = y then
            b ← {0,1} (egy véletlen bit)
            o = c_2 ⊕ m_b
        else
            o ← D_k  (egy véletlen elem)
        end
        h = h + 1
        RL[h] = r
        HL[h] = o
    end

    function main
        c_1 = y
        c_2 ← D_k
        h = 1
        Exp^indCPA_{A,Π}(n)
        for i = 1 to h do
            if f(RL[i] = y then
                return RL[i]
            end
        end
        return „nem sikerült invertálni”
    end
end
```

6. algoritmus: Az f inverzét kiszámító $\mathcal{B}$ algoritmus, és benne a $\mathcal{H}^{\mathsf{RO}}$ algoritmus pszeudokódja, melyet az $\mathcal{A}$ sokszor meghívhat az $\mathsf{Exp}^{\mathsf{indCPA}}_{\mathcal{A},\Pi}(n)$ kísérlet közben

lesz m_0 és m_1 megkülönböztetésében:

$$\begin{aligned}\mathbb{P}\left[\mathsf{Exp}^{\mathsf{indCPA}}_{\mathcal{A},\Pi}(n)=1\right] = \\ \mathbb{P}\left[\mathsf{Exp}^{\mathsf{indCPA}}_{\mathcal{A},\Pi}(n)=1 \mid f^{-1}(r)\in RL\right]\cdot\mathbb{P}\left[f^{-1}(r)\in RL\right] + \\ \mathbb{P}\left[\mathsf{Exp}^{\mathsf{indCPA}}_{\mathcal{A},\Pi}(n)=1 \mid f^{-1}(r)\notin RL\right]\cdot\mathbb{P}\left[f^{-1}(r)\notin RL\right]\end{aligned}$$

Világos, hogy

$$\mathbb{P}\left[\mathsf{Exp}^{\mathsf{indCPA}}_{\mathcal{A},\Pi}(n)=1 \mid f^{-1}(r)\notin RL\right]=\frac{1}{2},$$

ugyanis $f^{-1}(y)$ ismerete nélkül nem lehet választani m_1 és m_1 közül. Mivel

$$\mathbb{P}\left[\mathsf{Exp}^{\mathsf{indCPA}}_{\mathcal{A},\Pi}(n)=1 \mid f^{-1}(r)\in RL\right]\leqslant 1,$$

és feltételezhetjük, hogy a kísérlet $\frac{1}{2}$-nél nagyobb valószínűséggel sikeres, azt kapjuk, hogy

$$\begin{aligned}\frac{1}{2}+\nu(n) &\leqslant \mathbb{P}\left[\mathsf{Exp}^{\mathsf{indCPA}}_{\mathcal{A},\Pi}(n)=1\right] \\ &\leqslant \mathbb{P}\left[f^{-1}(r)\in RL\right]+\frac{1}{2}\mathbb{P}\left[f^{-1}(r)\notin RL\right] \\ &\leqslant \mathbb{P}\left[f^{-1}(r)\in RL\right]+\frac{1}{2}.\end{aligned}$$

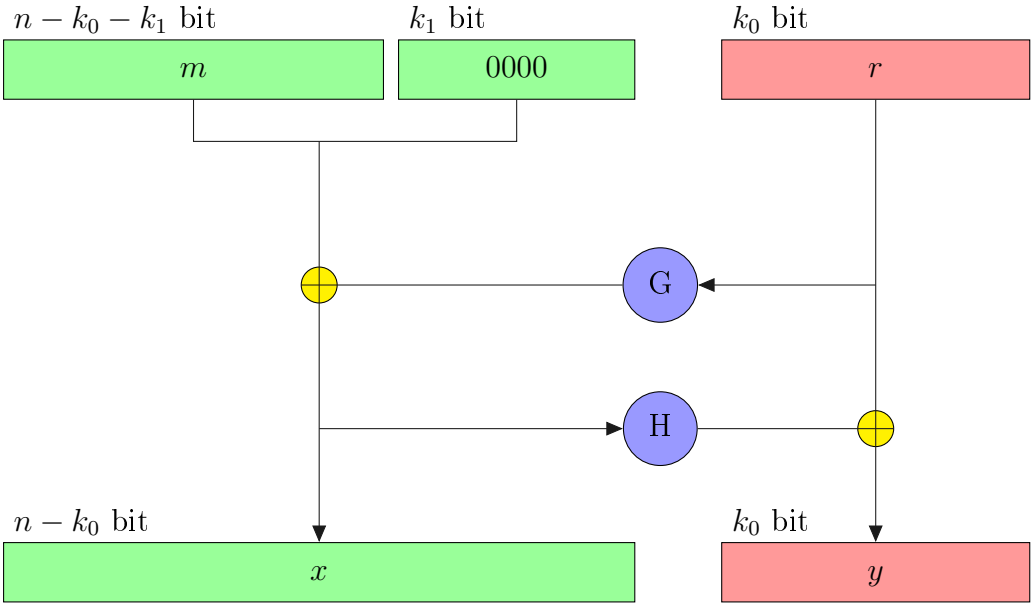
Így

$$\mathbb{P}\left[f^{-1}(r)\in RL\right]\geqslant\nu(n),$$

vagyis nem elhanyagolható valószínűséggel sikerült invertálni f-et. A szükséges lépések száma pedig polinomidejű, hisz ha $t_{\mathcal{A}}$ az $\mathcal{A}$, t_f az f futási ideje, és $\mathcal{A}$ h-szor hívja meg az orákulumot, akkor az invertálás futási ideje $t_{\mathcal{A}}+O(h^2+ht_f)$. □

2.3. OAEP – Optimal Asymmetric Encryption Padding

E módszer nevét nem magyarítjuk, az OAEP rövidítést fogjuk használni. A padding általában egy sztring szükséges hosszúságúra való föltöltését jelenti valamilyen előre megadott módon, pl. csupa 0-val. Később látni fogjuk, hogy a paddinget használhatjuk arra is, hogy az üzenetet kibővítsük véletlen bitekkel, ezzel azt legalább részben véletlenszerűvé téve. Az e fajta padding



4. ábra. Az OAEP diagramja

nem bizonyult teljesen biztonságosnak, a sztring egy része ugyan véletlen bitekből áll, de a másik felén lévő bitekre nyert információ a rendszer gyengéjét jelzi. Ezt küszöböli ki e szellemes, a Feistel-típusú rejtjelező ötletére építő megoldás. Először csak magát a paddingelést mutatjuk meg, majd azt, hogy hogyan építhető felhasználásával biztonságos nyílt kulcsú rejtjelező.

A 4. ábra az OAEP diagramja jól mutatja működését.

2.5. definíció (OAEP)**.** Az OAEP egy Feistel-típusú $\mathcal{P}$ algoritmus mely egy m nyílt szöveget egy r véletlen szöveggel és nullákkal kiegészít, majd a nyílt szöveget és a hozzáadott véletlen valamint 0 elemeket két véletlen orákulum véletlen elemekkel helyettesíti, de oly módon, hogy ezután e két orákulum meghívásával visszakaphatók is legyenek az eredeti elemek.

Legyen n bites a paddingelt szöveg hossza, ebből k_0 a véletlen r szám bithossza, és k_1 a zérusok száma. Így az m üzenet $n - k_0 - k_1$ bites.

$$G : \{0,1\}^{k_0} \to \{0,1\}^{n-k_0} \text{ és } H : \{0,1\}^{n-k_0} \to \{0,1\}^{k_0}$$

két véletlen orákulum (a gyakorlatban egyirányú függvények). A $\mathcal{P}$ algoritmus bemenete az $m \in \{0,1\}^{n-k_0-k_1}$ sztring, kimenete egy n-hosszú sztring,

az algoritmus lépései:

$$\begin{aligned} r &\leftarrow \{0,1\}^{k_0} \\ i &= m \parallel 0^{k_1} \parallel r \\ x &= (m \parallel 0^{k_1}) \oplus G(r) \\ y &= r \oplus H(x) \end{aligned}$$

E $\mathcal{P}$ algoritmus eredménye tehát az (x, y) pár. Ebből visszanyerhető m és r is az orákulumok segítségével:

$$\begin{aligned} r &= y \oplus H(x) \\ m \parallel 0^{k_1} &= x \oplus G(r) \end{aligned}$$

Ezután bármely egyirányú kiskapu-permutációval kombinálva az OAEP-t egy nyílt kulcsú titkosítóhoz jutunk.

2.6. definíció (Aszimmetrikus rejtjelező (OAEP + TDP)-ből)**.** Legyen $(\mathcal{G}^{\mathsf{TDP}}, f, f^{-1})$ az 1.31. definíció szerinti kiskapu-permutációk egy családja. Definiáljuk a $\Pi = (\mathcal{G}, \mathrm{E}, \mathrm{D})$ aszimmetrikus kulcsú 1.12. definíciónak megfelelő rejtjelezőt a következőképp:

1. $\mathcal{G}$: $(k, t) \leftarrow \mathcal{G}^{\mathsf{TDP}}(1^n)$, ahol f_k és f_t^{-1} mindketten $\{0,1\}^n \to \{0,1\}^n$ függvények. $\mathcal{G}$ választ egy k_0 és egy k_1 számot, melyekre $k_0 + k_1 < n$, valamint egy $G : \{0,1\}^{k_0} \to \{0,1\}^{n-k_0}$ és egy $H : \{0,1\}^{n-k_0} \to \{0,1\}^{k_0}$ hash függvényt, és publikálja a $pk = (k, f_k, H, G)$ nyílt kulcsot. A titkos kulcs az $sk = (t, f_t^{-1})$ lesz. A nyílt szöveg $m \in \{0,1\}^{n-k_0-k_1}$.

2. Az E algoritmus lépései:

$$\begin{aligned} r &\leftarrow \{0,1\}^{k_0} \\ i &= m \parallel 0^{k_1} \parallel r \\ x &= (m \parallel 0^{k_1}) \oplus G(r) \\ y &= r \oplus H(x) \\ c &= \mathrm{E}_{pk}(m) = f_k(x \parallel y) \end{aligned}$$

3. A D algoritmus lépései:

$$\begin{aligned} r &= y \oplus H(x) \\ \hat{m} &= f_t^{-1}(x \oplus G(r)) \end{aligned}$$

A D algoritmus e ponton ellenőrzi, hogy az $\hat{m}$ utolsó k_1 bitje valóban 0-e, ha nem, $\perp$-et ad vissza, a dekódolás sikertelen, egyébként $\hat{m} = m \parallel 0^{k_1}$ így $\mathrm{E}_{sk}(c)$ megegyezik az $f_t^{-1}(x \oplus G(r))$ első $n - k_0 - k_1$ bitjéből álló sztringgel.

Bizonyítható, hogy e rejtjelező megfelelő k_0 megválasztása mellett szemantikailag biztonságos a véletlen orákulum modellben. Az ugyanakkor nem bizonyítható, hogy a választott kriptoszöveg alapú támadás ellen is biztonságos lenne. Másrészt a gyakorlatban különösen fontos RSA esetére ez a biztonság is bizonyítható, nevezetesen az RSA-függvényre épülő ún. RSA-OAEP két független véletlen orákulum esetén bizonyos e kitevőkre CCA-biztonságos. Erre az RSA-ról szóló részben visszatérünk.

2.4. Hibrid rejtjelezők

Az aszimmetrikus rejtjelezők egyik komoly hátránya a szimmetrikusokkal szemben, hogy lényegesen lassabbak. Nagyobb szöveg titkosítására ezért praktikusabbnak tűnik a szimmetrikus rejtjelezők használata. A két rendszer előnyeit ötvözik a hibrid rendszerek, amelyekben az egyirányú függvényt vagy az aszimmetrikus rejtjelezőt csak annak a kulcsnak a titkosítására és cseréjére használják, amellyel aztán az valódi üzenetet titkosítják egy szimmetrikus rejtjelezővel. Két megoldást mutatunk, az első csak a 2.3. konstrukció egy továbbfejlesztett változata, melyben az aszimmetrikus rejtjelező konstrukciójának részévé válik a szimmetrikus kulcsú, míg a második megoldásban két létező rendszert ötvözünk, egy aszimmetrikusat és egy szimmetrikusat.

2.7. konstrukció (Hibrid aszimmetrikus rejtjelező TDP-ből)**.** Tekintsük a $(\mathcal{G}^{\mathsf{TDP}}, f, f^{-1})$ az 1.31. definíció szerinti kiskapu-permutációk egy családját. Definiáljuk a $\Pi = (\mathcal{G}, \mathrm{E}, \mathrm{D})$ aszimmetrikus kulcsú, az 1.12. definíciónak megfelelő rejtjelezőt a következőképp:

1. $\mathcal{G}$: $(k, t, \mathcal{D}_k) \leftarrow \mathcal{G}^{\mathsf{TDP}}(1^n)$, azaz választunk az egyirányú kiskapu-permutációk családjából egy (k, t) indexpárt, és megadjuk az értelmezési tartományt. $\mathcal{G}$ választ egy $H : \mathcal{D}_k \to \mathcal{D}_k$ hash függvényt, valamint egy szimmetrikus kulcsú $\Sigma = (\mathcal{G}^{\mathsf{s}}, \mathrm{E}^{\mathsf{s}}, \mathrm{D}^{\mathsf{s}})$ rejtjelezőt, és publikálja a $pk = (k, f_k, H, \Sigma)$ nyílt kulcsot. A titkos kulcs az $sk = (t, f_t^{-1})$ lesz.

2. E: $r \leftarrow \mathcal{D}_k$, $\mathrm{E}_{pk}(m) = \big(f_k(r), \mathrm{E}^{\mathsf{s}}_{H(r)}(m)\big)$, tehát a szimmetrikus kulcsú rejtjelező kulcsa a $H(r)$ érték lesz, ahol r egy véletlen elem.

3. D: $y = H(f_t^{-1}(c_1))$, $\mathrm{D}_{sk}(c_1, c_2) = \mathrm{D}^{\mathsf{s}}_y(c_2)$.

Könnyen ellenőrizhető, hogy $\mathrm{D}_{sk}\,\mathrm{E}_{pk}(m) = m$, ugyanis az üzenethez mind kódoláskor, mind dekódoláskor az $y = H(r) = H(f_t^{-1}(c_1)$ szimmetrikus kulcsot használtuk.

Ha valamilyen fejlett, kifinomult szimmetrikus és aszimmetrikus rejtjelezők állnak rendelkezésünkre, egyszerűbb, ha a kettő összeolvasztásával konstruálunk új hibrid rendszert.

2.8. konstrukció (Hibrid rejtjelező)**.** Legyen $\Pi = (\mathcal{G}^{\mathsf{a}}, \mathrm{E}^{\mathsf{a}}, \mathrm{D}^{\mathsf{a}})$ egy aszimmetrikus kulcsú, és $\Sigma = (\mathcal{G}^{\mathsf{s}}, \mathrm{E}^{\mathsf{s}}, \mathrm{D}^{\mathsf{s}})$ egy szimmetrikus kulcsú rejtjelező. Az üzenet egy tetszőleges $m \in \{0,1\}^*$ sztring. Definiáljuk a $\Pi = (\mathcal{G}, \mathrm{E}, \mathrm{D})$ aszimmetrikus kulcsú rejtjelezőt a következőképp:

1. $\mathcal{G}$: $(pk, sk) \leftarrow \mathcal{G}^{\mathsf{a}}(1^n)$,

2. Az E algoritmus lépései:

   - $r \leftarrow \{0,1\}^n$,
   - $c_1 \leftarrow \mathrm{E}^{\mathsf{a}}_{pk}(r)$, $c_2 \leftarrow \mathrm{E}^{\mathsf{s}}_r(m)$,

   így $\mathrm{E}_{pk}(m) = (c_1, c_2)$.

3. A D algoritmus lépései:

   - $r = \mathrm{D}^{\mathsf{a}}_{sk}(c_1)$,
   - $m = \mathrm{D}^{\mathsf{s}}_r(c_2)$,

   azaz $\mathrm{D}_{sk}(c_1, c_2) = m$.

Egyszerű számítás mutatja, hogy elegendően hosszú üzenet esetén a hibrid rendszer egyrészt nyújtani tudja az aszimmetrikus rendszer funkcionalitását és annak előnyeit, másrészt a szimmetrikus rendszer hatékonyságát.

Bizonyítható az is, hogy ha Π és Σ szemantikailag biztonságosak, akkor a hibrid rendszer is.

3. Nyílt kulcsú rejtjelezők

3.1. RSA

Az RSA a legelterjedtebb, legtöbbet vizsgált nyílt kulcsú titkosító rendszer. Ron Rivest, Adi Shamir és Leonard Adleman publikálta 1977-ben, a rendszer az ő neveik kezdőbetűit tartalmazza.

RSA rejtjelező A korábbiakban részletesen tárgyaltuk az RSA-függvényt, és az egyirányúsága elleni támadásokat. Ugyancsak a korábbi fejezetek alapján könnyen készíthetünk az RSA-függvényből nyílt kulcsú rejtjelezőt. Most azonban először egy olyan ötletet mutatunk, amely történetileg is az első valódi aszimmetrikus kriptorendszer alapját képezte.

3.1. konstrukció (RSA véletlen paddinggel)**.** Legyen $\mu(n) < 2n - 1$, ahol az üzenet $m \in \{0,1\}^{\mu(n)}$.

1. $\mathcal{G}$: $(N, e, d) \leftarrow \mathcal{G}^{\mathsf{RSA}}(1^n)$, $pk = (N, e)$, $sk = (N, d)$.
2. E: $r \leftarrow \{0,1\}^{\ell(N)-\mu(n)-1}$, $\mathrm{E}_{pk}(M) = (r \parallel m)^e \bmod N$, ahol r választása olyan, hogy $(r \parallel m) \in \mathbb{Z}_N^*$,
3. D: $\mathrm{D}_{sk}(c) = (c^d \bmod N$ utolsó $\mu(n)$ bitje).

Bár azt sejtenénk, hogy ha $\mu(n) \approx n$, pl. $\mu(n) = cn$ valamilyen $c < 2$ konstansra, akkor a 3.1. konstrukció szemantikailag biztonságos, ezt eddig nem sikerült bizonyítani. Csak annyit tudunk, hogy ha $\mu(n) = O(\log n)$, akkor a konstrukció szemantikailag biztonságos. Egy hasonló, kissé kifinomultabb konstrukció került a PKCS #1 v1.5 standardba[8]. A biztonsága mindmáig annak sincs bizonyítva, de miután egy ötletes kriptoszöveg alapú támadással sikerrel járt ellene Daniel Bleichenbacher, a PKCS #1 v2 standardba már a bizonyítottan biztonságos OAEP padding került, amit korábban már bemutattunk.

CCA-biztonságú RSA A 2.3. konstrukció RSA-függvényre alkalmazott speciális esetét kiemeljük, mint olyat, amely már CCA-biztonságú, legalábbis a véletlen orákulum modellben.

[8]PKCS = Public-Key Cryptography Standards, melyeket az RSA Security Inc., ma RSA Security LLC bocsát ki és gondoz. A #1 standard maga az RSA.

3.2. konstrukció (RSA egy hash-függvénnyel, Ferguson–Schneier-RSA)**.** Legyen H egy hash-függvény, pl. $H = \mathrm{SHA256} \circ \mathrm{SHA256}$[9], és $\Sigma = (\mathcal{G}^{\mathsf{s}}, \mathrm{E}^{\mathsf{s}}, \mathrm{D}^{\mathsf{s}})$ egy szimmetrikus kulcsú rejtjelező. Definiáljuk a $\Pi = (\mathcal{G}, \mathrm{E}, \mathrm{D})$ aszimmetrikus kulcsú rejtjelezőt a következőképp:

1. $\mathcal{G}$: $(N, e, d) \leftarrow \mathcal{G}^{\mathsf{RSA}}(1^n)$, $pk = (N, e)$, $sk = (N, d)$, $l = \lceil \log(n+1) \rceil$, $r \leftarrow [0, 2^l - 1]$, $k = H(r)$
2. E: $\mathrm{E}_{pk}(m) = (r^e \bmod N, \mathrm{E}^{\mathsf{s}}_k)$,
3. D: $r = c_1^d \bmod N$, $k = H(r)$, $\mathrm{D}_{sk}(c_1, c_2) = \mathrm{D}_k(c_2)$.

Ferguson és Schneier az $e = 3$ értéket javasolja digitális aláíráshoz és az $e = 3$ értéket rejtjelezéshez. Ezért természetesen a $\gcd(3, (p-1)(q-1)) = 1$ és a $\gcd(5, (p-1)(q-1)) = 1$ feltételeknek is fönn kell állniuk.

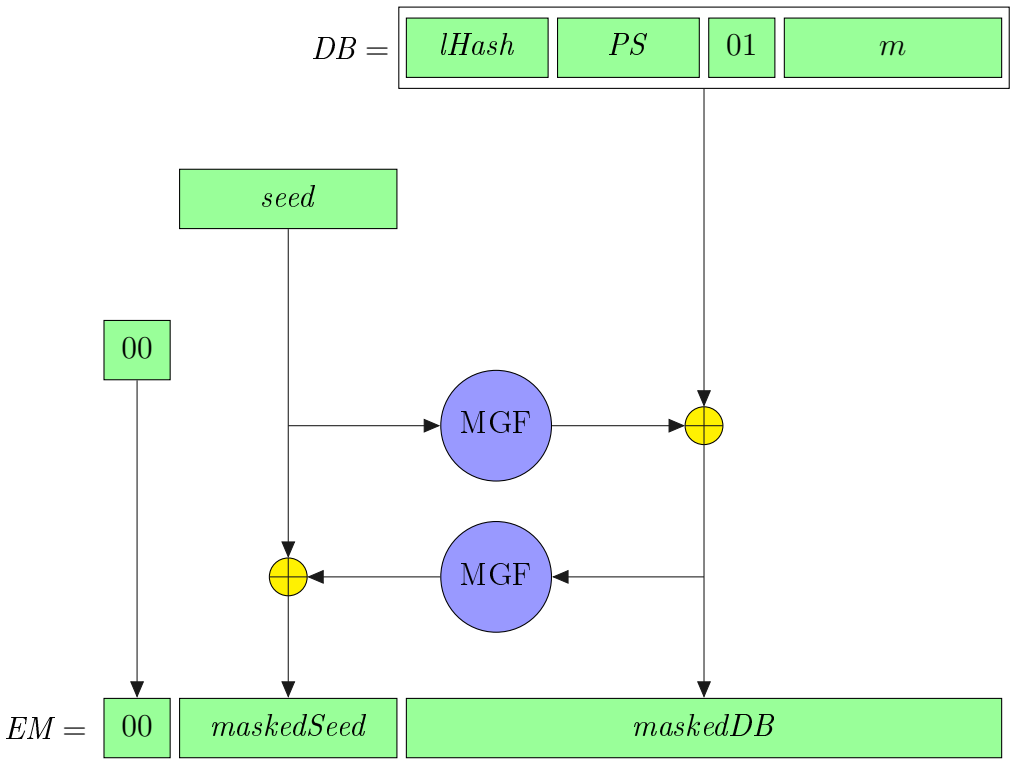
A PKCS #1 v2 standardba került RSA részletes ismertetésére itt nincs lehetőség, de a lényege bemutatható. Az OAEP leírásában megadott két független hash függvény helyett a protokollban MGF (ún. *mask generation function*) szerepel, ami egy fajta hash függvény, ahol az output mérete is rugalmasan változtatható. Konstrukciójukhoz gyakran szabványos hash függvényeket használnak. Counter-módú konstrukciójának biztonsága bizonytalan.

A PKCS standardok nyelvében az oktett (*octet*) egyszerűen 8-bites bájtot jelent, az oktett sztring bájtok egymásutánját jelenti. Általában hexadecimális formában vannak megadva. Mi egyszerűen bájtot írunk helyette. A lényeg, a standardban minden adat valahány bájtos, tehet bitben mért hossza 8-nak egész számú többszöröse. A PKCS #1 v2.2 egy egyszerűsített változatát megadjuk az alábbiakban. Az egyszerűsítés abból áll, hogy a konverziós részek és a hibaüzenetek részletezését, továbbá néhány apróbb technikai részt kihagyunk vagy kissé leegyszerűsítünk. Az RSA-ban használt OAEP-változatot az 5. ábrán szemléltetjük.

3.3. definíció (RSA-OAEP, a PKCS #1 2.2 egyszerűsített változata)**.** Az algoritmusban használt jelölések:

EM	az OAEP kimenete
H	hash függvény

[9] A SHA256 egy 256-bites értéket adó szabványos hash függvény, mely a ma ismert legerősebbek egyike.



5. ábra. Az RSA-ban használt OAEP diagramja

$HLen$	H kimenetének hossza bájtban
L	opcionális címke
MGF	maszkot generáló függvény (változó hosszú hash)

Az üzenet egy tetszőleges $m \in \{0,1\}^{8MLen}$ sztring ($MLen$ bájt hosszú). Definiáljuk a $\Pi = (\mathcal{G}, \mathrm{E}, \mathrm{D})$ aszimmetrikus kulcsú rejtjelezőt a következőképp:

1. $\mathcal{G}$: $(N, e, d) \leftarrow \mathcal{G}^{\mathsf{RSA}}(1^n)$, $pk = (N, e)$, $sk = (N, d)$.

2. Az E algoritmus bemenete a pk nyílt kulcs, az m üzenet és egy opcionális L címke. Az algoritmus (a standardban RSAES-OAEP a neve) lépései:

$$\begin{aligned}
lHash &= \mathrm{H}(L) \quad \text{ha } L \text{ üres, értéke előre meg van adva} \\
PS &= 00\ldots0 \quad \text{összesen } \ell(N) - MLen - 2HLen - 2 \text{ bájt} \\
DB &= lHash \parallel PS \parallel 0x00 \parallel m \\
seed &\leftarrow \{0,1\}^{8HLen} \\
maskedDB &= DB \oplus \mathrm{MGF}(seed, \ell(N) - HLen - 1) \\
maskedSeed &= seed \oplus \mathrm{MGF}(maskedDB, HLen) \\
EM &= 0x00 \parallel maskedSeed \parallel maskedDB
\end{aligned}$$

   $\mathrm{E}_{pk}(m[, L]) = (EM)^e \bmod N$.

3. A D algoritmus bemenete a c kriptoszöveg, a titkos sk kulcs és az opcionális L címke. Az algoritmus lépései:

$$\begin{aligned}
EM &= c^d \bmod N \\
&= Y \parallel maskedSeed \parallel maskedDB \\
seed &= maskedSeed \oplus \mathrm{MGF}(maskedDB, HLen) \\
DB &= maskedDB \oplus \mathrm{MGF}(seed, \ell(N) - HLen - 1) \\
&= lHash' \parallel PS \parallel 0x00 \parallel m
\end{aligned}$$

   $\mathrm{D}_{sk}(c, [, L]) = m$. Természetesen, ha hiányzik az $0x01$ bájt, ha $lHash' \neq lHash$, ha $Y \neq 00$, akkor a dekódolás sikertelen.

A PKCS #1 2.2 CCA-biztonságát Fujisaki, Okamoto, Pointcheval és Stern bizonyították. Egyrészt az OAEP-re épülő nyílt kulcsú rejtjelezők CCA-biztonsága nem áll fenn, Shoup bizonyította, hogy ha az egyirányú

kiskapu-permutáció invertálása ugyan nehéz, de részlegesen nem, akkor az OEAP CCA-biztonsága nem bizonyítható. Másrészt viszont az OAEP CCA-biztonságos, ha a részleges invertálás is nehéz. Ezt sikerült az RSA esetén bizonyítani. Némi óvatosságra int, hogy a CCA-biztonság bizonyításában, ha egy $\mathcal{A}$ támadó t időben ε eséllyel sikeres az RSA-OAEP ellen, akkor az RSA inverz sikerének esélye $\varepsilon^2/2^{18}$, és az algoritmus t^2 ideig fut. Tehát e bizonyítás alapján elvileg nem elképzelhetetlen, hogy könnyebb támadni az RSA-OAEP ellen, mint az egyirányú RSA-függvényt invertálni. (Ez csak elvi lehetőség, semmi jel nem mutat arra, hogy ilyen támadás létezne.)

3.2. ElGamal és ECC

A Certicom nevű cég fontos szerepet játszik a diszkrét logaritmusra épülő kriptográfiai – legfőképpen az elliptikus görbékre épülő – rendszerek elméletének kidolgozásában, szabványosításában, szabadalmazásában és terjesztésében. Legalább 30 szabvány fűződik a nevükhöz. A Certicom 1998-ban alapított egy *Standards for Efficient Cryptography Group (SECG)* nevű csoportot, melynek tagjai közt olyan cégek találhatóak, mint Entrust, Fujitsu, NIST, Pitney Bowes, Unisys, Visa.

A diszkrét logaritmus probléma, pontosabban a Diffie–Hellman-probléma nehézségére épülő rejtjelezők kiindulópontja az ElGamal rendszer.

Az ElGamal rejtjelező Taher Elgamal (nevét szokás ElGamal vagy El Gamal formában is irni, de maga az Elgamal alakot preferálja) egyiptomi származású kriptográfus. Mindennek ellenére a róla elnevezett rejtjelezőt a széles körben elterjedt formában ElGamal-nak fogjuk írni.

3.4. konstrukció (ElGamal rejtjelező)**.** Legyen $\mathcal{G}^{\mathsf{group}}$ az 1.22. kísérletben definiált, ciklikus csoportot generáló algoritmus. Aszimmetrikus rejtjelezőt definiálunk a következőképp:

1. $\mathcal{G}$: $(G, q, g) \leftarrow \mathcal{G}^{\mathsf{group}}(1^n)$, majd választunk egy véletlen $x \leftarrow \mathbb{Z}_q$ elemet, és kiszámoljuk $h = g^x$ értékét. A nyílt kulcs $pk = (G, q, g, h)$, a titkos kulcs $sk = (G, q, g, x)$.

2. E: $r \leftarrow \mathbb{Z}_q^*$, $m \in G$,
$$\mathrm{E}_{pk}(m) = (g^r, h^r \cdot m).$$

3. D: Jelölje (c_1, c_2) az $\mathrm{E}_{pk}(m)$ kimenetét, ekkor

$$\mathrm{D}_{sk}(c_1, c_2) = \frac{c_2}{c_1^x}.$$

Könnyen ellenőrizhető, hogy $\mathrm{D}_{sk}\,\mathrm{E}_{pk}(m) = m$, ugyanis

$$\frac{c_2}{c_1^x} = \frac{h^r \cdot m}{(g^r)^x} = \frac{(g^x)^r \cdot m}{(g^r)^x} = \frac{g^{xr} \cdot m}{g^{xr}} = m.$$

Fontos eleme e rejtjelezőnek az az egyszerű megoldás, hogy a nyílt szöveg meg van szorozva egy – a DDH-probléma nehézsége esetén – véletlennek tekinthető elemmel. Ha pedig $g \in G$ egy véletlen elem, akkor a G csoport egy tetszőleges g' elemére

$$\mathbb{P}[m \cdot g = g'] = \mathbb{P}[g = m^{-1} \cdot g'] = \frac{1}{|G|},$$

vagyis itt is arról van szó, hogy ha g-t nem a rejtjelezett szöveggel együtt kéne elküldeni, akkor az $m \cdot g$ tökéletes biztonságot nyújtana. Így bizonyítható az alábbi tétel:

3.5. tétel. *Ha a DDH-probléma nehéz, akkor a 3.4. konstrukcióban megadott ElGamal rejtjelező szemantikailag biztonságos.*

Csoport generálása az ElGamal számára Kérdés, hogy generálunk olyan csoportot, amelyben könnyen találunk generátorelemet! Első jelölt a $\mathbb{Z}_p^*$, mely ciklikus, van hatékony algoritmus egy generátorelemének megtalálására és e csoportban a diszkrét logaritmus problémát nehéznek hisszük. Céljainknak mégsem fog e csoport megfelelni, mivel egyrészt rendje nem prím, másrészt – mint azt nem nehéz belátni – e csoportban a döntési Diffie–Hellman-probléma (DDH) nem nehéz! Ezért másik csoport után kell néznünk.

Kézenfekvő megoldás $\mathbb{Z}_p^*$ helyett annak egy megefelelő részcsoportját választani. Ha a kvadratikus maradékok (azaz a négyzetelemek) csoportját választjuk, akkor egy $(p-1)/2$-elemű részcsoportot kapunk, mivel négyzetelemek szorzata és inverze is négyzetelem. Ha ráadásul p biztonságos prím, azaz $p = 2q + 1$ alakú, ahol q is prím, akkor e csoport már prímrendű. Ráadásul generátorelemet is könnyű benne találni, hisz minden egységtől

különböző eleme generátorelem. Így elég egy $r \leftarrow \mathbb{Z}_p^*$ elemet választani, ha $r^2 \bmod p \neq 1$, akkor $g = r^2 \bmod p$ egy generátorelem.

A biztonsági paraméter függvényében tehát p-t és q-t úgy választjuk, hogy $\ell(p) = n + 1$, $\ell(q) = n$. Így az üzenetek is egy q-elemű halmazból vehetők, praktikusan $\bar{m} \in \{1, 2 \ldots, q\}$. Ezek az elemek viszont nincsenek G-ben, ezért négyzetre kell őket emelni. Az $m = \bar{m}^2$ megfelelő lesz az algoritmus számára. Kérdés már csak az, hogy dekódolunk. Miután a D algoritmus visszaadja m-et, abból négyzetgyököt vonunk $\mathbb{Z}_p$-ben. A két négyzetgyök összege p, így ki tudjuk választani, melyik esik az $[1, \frac{p-1}{2}]$ intervallumba. Mindenzt egy példán szemléltetjük:

3.6. példa. Legyen $p = 359$, $p = 2q+1$, ahol $q = 179$ (a példa szépségéért itt p mellett q is biztonságos prím). A $\mathbb{Z}_p = \mathbb{Z}_{359}$ csoport egy $q = 179$-edrendű elemét megkapjuk egy tetszőleges (egységtől különböző) elem négyzetre emelésével, legyen a G csoport egy generátora $g = 5^2 = 25$. Legyen a titkos kulcs az $53 \in \mathbb{Z}_{179}$, így a publikus kulcs

$$pk = (359, 179, 25, 25^{53} \bmod 359) = (359, 179, 25, 340).$$

Legyen az üzenet az $\{1, 2, \ldots, 179\}$ halmaz egy eleme, mondjuk $\bar{m} = 103$. Így $m = 103^2 \bmod 359 = 198 \in G$. Legyen a véletlen elem $r = 138 \in \mathbb{Z}_{179}$, így a rejtjelezett üzenet:

$$(25^{138} \bmod 359, 340^{138} \cdot 198 \bmod 359) = (187, 235).$$

A visszafejtéshez a titkos kulcsot használva

$$m = 235 \cdot 187^{-53} \bmod 359 = 198.$$

198 négyzetgyökei

$$\pm 198^{\frac{p+1}{4}} \bmod p = \pm 198^{90} \bmod 359 = \pm 256 \bmod 359 = \{256, 103\}$$

Mivel e két szám közül $103 < 179$, ezért 103 az üzenet.

Elliptikus görbére épülő rejtjelező Az elliptikus görbékre épülő rejtjelezők konstrukciójának használata mögött az a tény rejlik, hogy a diszkrét logaritmus problémára a $\mathbb{Z}_p^*$-gal ellentétben nem ismeretes szubexponenciális algoritmus. Így az ElGamal-rendszerek valamilyen változatát elliptikus görbéken érdemes megvalósítani. Ez annyi előnyt ad az elliptikus görbékre

épülő kriptográfiai rendszereknek, hogy ma ezeket tekinthetjük a leghatékonyabbnak.

Az ECIES (Elliptic Curve Integrated Encryption Scheme) meglehetősen bonyolult rendszer. Kiváló kriptográfusok alkotása, Abdalla, Bellare és Rogaway, majd Shoup munkája után nyerte el ma használt alakját. Itt most csak egy leegyszerűsített változatát ismertetjük.

A görbe pontjainak tárolására az (x, y) affin alak használható, azonban tudjuk, hogy a görbe szimmetrikus az x-tengelyre, ezért x ismeretében csak két y érték jöhet szóba. Ha (x, y) az egyik pontja a görbének, akkor $(x, p-y)$ a másik, és mivel p páratlan, ezért y és $p-y$ mindig különböző paritású (ha nem 0). Ezért b_y-nal jelölve y paritását, a pontokat az $(x, y) \in \mathbb{Z}_p \times \mathbb{Z}_p$ helyett a kb. felényi $(x, b_y) \in \mathbb{Z}_p \times \mathbb{Z}_2$ alakban tárolhatjuk. Egy pont ilyen módon való tömörítése nagyon egyszerű, de a tömör alakból való visszaszámolás is egyszerűen és hatékonyan számolható.

3.7. konstrukció (ECIES – leegyszerűsített változat)**.** Legyen $\mathcal{G}$ egy $\mathbb{Z}_p$ fölötti $\mathcal{E}$ elliptikus görbét generáló algoritmus, mely az elliptikus görbe csoportban talál egy q-adrendű ciklikus G részcsoportot, melyet egy P pont által reprezentált elem generál. Aszimmetrikus rejtjelezőt definiálunk a következőképp:

1. $\mathcal{G}$: $(\mathcal{E}, p, G, q, P) \leftarrow \mathcal{G}^{\mathsf{group}}(1^n)$, ahol $\ell(q) \geqslant n$. $x \leftarrow \mathbb{Z}_q^*$ a titkos kulcs, azaz $sk = (x)$. $Q = xP$ az $\mathcal{E}$ egy másik pontja, a publikus kulcs

$$pk = (\mathcal{E}, p, G, q, P, Q).$$

2. E: $r \leftarrow \mathbb{Z}_q^*$, $m \in \mathbb{Z}_p^*$,

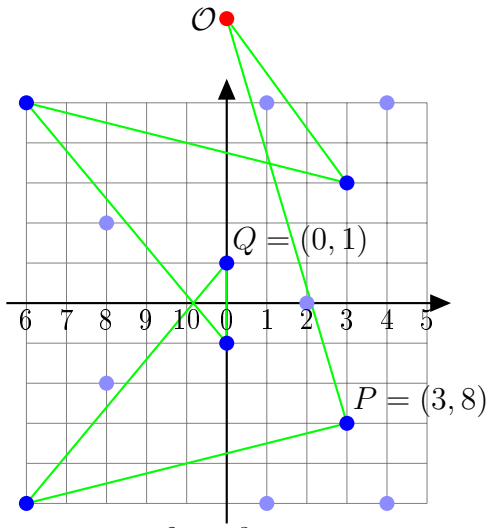
$$\mathrm{E}_{pk}(m) = (rP, m \cdot x_r \bmod p),$$

   ahol $rQ = (x_r, y_r)$, és $x_r \neq 0$.

3. D: Jelölje (c_1, c_2) az $\mathrm{E}_{pk}(m)$ kimenetét, ahol $c_1 \in \mathcal{E}$, $c_2 \in \mathbb{Z}_p^*$. Ekkor az $(x_r, y_r) = x \cdot c_1$ jelöléssel

$$\mathrm{D}_{sk}(c_1, c_2) = \frac{c_2}{x_r} \bmod p.$$

Biztonsági szerepe nincs, hogy számolás közben a pontot tömörített vagy tömörítetlen formájában kezeljük.



(c) $y^2 = x^3 + x + 1$

6. ábra. A 14-pontú elliptikus görbe egy 7-pontú része, melyen a pontművelet egy $\mathbb{Z}_7$-tel izomorf csoportot ad. A görbe G csoporthoz nem tartozó pontjait halvány színnel jelöltük.

3.8. példa. A 2. ábra harmadik görbéje 14-pontú, csoportjának 7-elemű részcsoportját a (9) egyenletben megadtuk, itt felidézzük:

$$\mathbb{Z}_7 \cong G = \{\mathcal{O}, (3,8), (6,6), (0,1), (0,10), (6,5), (3,3)\}.$$

A paraméterek tehát: $p = 11$, $q = 7$, $P = (3, 8)$. Legyen a titkos kulcs $sk = 3$, így $Q = 3P = (0, 1)$. Legyen az üzenet $m = 4 \in \mathbb{Z}_{11}^*$. A titkosításhoz válasszunk egy véletlen elemet: $r = 2 \in \mathbb{Z}_7^*$. Ezzel

$$rP = 2 \cdot (3, 8) = (6, 6), \quad rQ = 2 \cdot (0, 1) = (3, 3),$$

tehát $x_r = 3$, így

$$(rP, m \cdot x_r \bmod 11) = ((6, 6), 1).$$

A dekódoláshoz vegyük a $(c_1, c_2) = ((6, 6), 1)$ párt, és a titkos kulcsot, ami 3, így $(x_r, y_r) = 3 \cdot (6, 6) = (3, 3)$, tehát

$$\frac{c_2}{x_r} \bmod p = \frac{1}{3} \bmod 11 = 4$$

tehát az üzenet 4. (Ha számolás közben a pontok tömör formáját használjuk, akkor $P = (3, 0)$, $Q = (0, 1)$, $2P = (6, 0)$, $2Q = (3, 1)$.)

3.3. Összehasonlítások

Megismerve az RSA rejtjelezőt, az ElGamal rejtjelezőt, valamint annak elliptikus görbéken megvalósított változatát, egy rövid összefoglalást adunk ezek működésének hatékonyságáról. A biztonsági szint az n biztonsági paraméter értékét jelenti. A szimmetrikus kulcsú rejtjelezők lényegében ezzel azonos, vagy esetleg néhány bittel kisebb biztonságot el tudnak érni, azaz a feltöréshez n-ben exponenciális idejű algoritmus kell. Az elliptikus görbékre épülő kriptorendszerek a leghatékonyabbak, az első generációs RSA és ElGmal rendszerek csak jóval nagyobb kulcshossz esetén nyújtanak azonos biztonságot. Ugyanakkor még ma is óvatosságból sokan döntenek az RSA mellett, mivel a mögött már több évtizedes tapasztalatok gyűltek össze, az elliptikus görbe kriptográfia pedig a mögötte lévő összetettebb matematikai módszerek és fogalmak miatt nagyobb bizonytalanságban tartja a felhasználók egy részét.

Biztonsági szint	szimmetrikus	ECC	RSA/DH	Meddig véd?
80	80	160	1024	2010
112	112	224	2048	2030
128	128	256	3072	2040
192	192	384	7680	2080
256	256	512	15360	2120

1. táblázat. Kulcsméretek összehasonlítása (forrás: Certicom Research, Elliptic Curve Cryptography, SEC 1, 2009-05-21, v2.0)

E fejezetben a gyakorlatban ma használt két egyértelműen legelterjedtebb rendszerre koncentráltunk. Megjegyzésekben azonban jeleztük, hogy mely más megoldások lehetségesek, és hogy azok miért nem terjedtek el. Igyekeztünk a modern kriptográfia szemléletmódjával közelíteni a nyílt kulcsú kriptográfia e két legfontosabb példájához. Ez azt jelenti, hogy világos fogalmunk van arról, hogy e rendszerek milyen biztonsági szintet nyújtanak, és hogy azok pontosan milyen matematikai feltételek esetén állnak fenn. Megközelítésünk az volt, hogy a makacs számelméleti problémák felsorolása után a belőlük képzett egyirányú függvények és kiskapu-permutációk segítségével általános konstrukciókat adtunk nyílt kulcsú rejtjelezőkre. Így azt is meg tudtuk mutatni, hogy speciálisan miért épp az RSA-függvényre épülő ma-

radt máig is az egyik legfontosabb rejtjelező (pl. a mögötte lévő tapasztalat, és a bizonyított CCA-biztonsága miatt), másrészt láthattuk azt is, hogy e pillanatban az ismert matematikai támadási lehetőségek adta keretek közt az elliptikus görbékre épülő rendszerek nyújtanak egy adott biztonsági szintet a leghatékonyabban.

Tárgymutató