Normaforma egyenletek alkalmazása a kriptográfiában

doktori (PhD) értekezés

Ködmön József

Debreceni Egyetem
Természettudományi Kar

Debrecen, 2004.

Ezen értekezést a Debreceni Egyetem Matematika- és számítástudományok doktori iskola Diofantikus és konstruktív számelmélet programja keretében készítettem 1997-2004 között és ezúton benyújtom a Debreceni Egyetem doktori (Ph.D.) fokozatának elnyerése céljából.

Debrecen, 2004. február 10.

.....
Ködmön József
jelölt

Tanúsítom, hogy Ködmön József doktorjelölt 1997–2004 között a fent megnevezett doktori program keretében irányításommal végezte munkáját. Az értekezésben foglaltak a jelölt önálló munkáján alapulnak, az eredményekhez önálló alkotó tevékenységével meghatározóan hozzájárult. Az értekezés elfogadását javaslom.

Debrecen, 2004. február 10.

.....
Dr. Pethő Attila
témavezető

Köszönetnyilvánítás

Ezúton is szeretnék köszönetet mondani mindazoknak, akik közvetlenül vagy közvetve hozzájárultak a disszertációm elkészítéséhez.

Témavezetőmnek, Dr. Pethő Attilának, aki megszerettette velem a számelméletet és a kriptográfiát, aki elindított a tudományos pályán, és akitől a szakmán túl is számtalan fontos dolgot tanultam.

Dr. Győry Kálmánnak, aki szigorú, de mindig szeretteljes kritikájával sokat segített, akitől nem csak szakmai dolgokat tanultam.

Dr. Bérczes Attilának, aki fiatalos lendületével és remek stílusával tette emlékezetessé közös munkánkat.

Feleségemnek, aki mindig mindenben segített és mellettem állt, elviselte a tudományos munkával járó megpróbáltatásokat.

Végül szüleimnek, akik felneveltek és mindig mindenben segítettek.

Tartalomjegyzék

Bevezetés

Az informatika fejlettségi szintjére ma már nem a technikai problémák a jellemzőek, hanem sokkal inkább szervezési, jogi, sőt morális kérdések merülnek fel. A hangsúly áttevődött arra a kultúrára, amelyet a számítógépek kiterjedt használatával teremtettünk meg.

Ez a világ azonban minden eddiginél sokkal törékenyebb. Korábban az ember hozzánőtt a természet ritmusához, tevékenységét csak a természetből fakadó veszélyek fenyegették. Ma a természet adta alapokra ráépítettük a technika, az informatika óriási méretű rendszereit. Kétségtelenül magasabban állunk, messzebbre látunk, de nagyon kell figyelnünk a felépített világ stabilitására. Elég egy apró hiba, emberi mulasztás és könnyen összeomolhatnak létfontosságú informatikai rendszereink. A szobányi irattárakat óriási kapacitású, pici méretű adattárolók váltották fel. A különféle típusú és eltérő operációs rendszerű számítógépek integrálhatók egy hálózati rendszerré, amelyek egyetlen alkalmazásként működnek. A képfeldolgozás, a vezetői döntéseket támogató számítógépes rendszerek ma már nem számítanak újdonságnak. A fejlődés nem áll meg, senki sem tudja megmondani, milyen informatikai infrastruktúrát használunk tíz év múlva.

A számítógépek segítségével tárolt és feldolgozott adatoktól való függőség nagyon megnövelte az informatikai biztonság szerepét, jelentőségét. A számítógépek használata komoly veszélyeket hordoz, de ezekre viszonylag ritkán hívják fel a figyelmet.

Az adatok elvesztése vagy illetéktelenekhez kerülése igen jelentős anyagi és erkölcsi károkat okozhat. A hagyományos, papír hordozójú rendszerek biztonságos használata áttekinthető, több évszázados tapasztalat szerint, közismerten egyszerű szabályrendszeren alapul. Az iratkezelés általános szabályai évtizedek alatt sem avultak el. Azonban az informatikai rendszerek biztonságos üzemeltetése igen nehéz feladat, hiszen a rendkívül gyors fejlődés és a permanens változás miatt szinte követhetetlen a fenyegető tényezők felmérése, a megfelelő biztonsági intézkedések bevezetése. Korunk informatikai for-

radalma nem hozott egyértelmű fejlődést ezen a területen, inkább csak új problémákat vetett fel, melyek jelentősen nehezítik a biztonsági szakemberek dolgát.

A megfelelő szintű informatikai biztonság megteremtéséhez központi segítség is nehezen nyújtható. A törvényhozás és az általános irányelvek kiadása sem képes követni a folyamatosan változó követelményeket.

A rejtjelezést, a titkosított üzenetváltás szükségletét az állam kialakulása teremtette meg. Olyan tények és adatok keletkeztek, amelyek nyilvánosságra kerülése veszélyeztette az állam érdekeit. Az államtitok megjelenésével egyidőben azok a módszerek és szabályok is megjelentek, amelyek lehetővé tették a bizalmas adatok többé-kevésbé hatásos védelmét. Napjainkban - a katonai és diplomáciai területeken kívül - a magán szektorban is megnövekedett a bizalmas kommunikáció iránti igény. A titkos kommunikáció klasszikus problémáival a kriptográfia foglalkozik, amely külön tudományterületté vált. A nyilvános kutatások a 70-es évek közepétől egyre növekvő intenzitással folynak. A kriptográfia részletesebb tanulmányozásához a [60], [47] és [38] könyveket ajánljuk.

A kriptográfia legfontosabb építőelemei az egyirányú függvények. Ezek segítségével lehet megalkotni a kriptográfiában használatos további fontos elemeket. A titkos kommunikációt megvalósító rendszerekhez többnyire csapóajtós egyirányú függvényeket alkalmaznak. A véletlenszám generátorok, a hash függvények, a kriptográfiai protokollok és technikák megvalósításához szintén nélkülözhetetlenek.

Az adatvédelem gyakorlati kivitelezésében is nagy szükség van az egyirányú, valamint az egyirányú hash függvények alkalmazására. Az adatintegritás, a hitelesítés, a partner azonosítás és a digitális aláírás problémáinak megoldása aligha képzelhető el egyirányú függvények használata nélkül.

Az utóbbi években az egészségügyi informatika területén is egyre nagyobb jelentőséggel bír az adatvédelem, hiszen a gyógyítási folyamatok során óriási mennyiségben keletkeznek olyan érzékeny adatok, amelyek megfelelő védelme csak fejlett kriptográfiai eszközökkel lehetséges.

Értekezésünkben egy *új egyirányú hash függvény* elkészítését fogjuk leírni. Ehhez a diofantikus egyenletek elméletét kívánjuk felhasználni. Célunk eléréséhez azonban nem a megoldások megtalálásához vezető algoritmusokkal foglalkozunk, hanem - éppen ellenkezőleg - az olyan egyenleteket igyekszünk alkalmazni, amelyek megoldása nem látszik kivitelezhetőnek. Egyirányú füg-

gvényünk invertálási nehézségét az algebrai számelméletből jól ismert általános normaforma egyenletek megoldásának nehézségére alapozzuk.

A konstrukció megalkotásában tisztán matematikai eszközöket fogunk használni. Igyekszünk függvényünkről minél több jó tulajdonságot matematikai módszerekkel bizonyítani. Azokban az esetekben, ahol ez nem sikerült, a tulajdonságok vizsgálatára alkalmazásokat készítettünk, amelyek tesztelésével támasztottuk alá eredményeinket.

Új egyirányú függvényünk segítségével *egyirányú hash függvényt* készítettünk, amelyet ajánlunk a gyakorlati felhasználás számára.

Sikerült egy megfelelő többváltozós lineáris forma normájának felhasználásával elkészíteni az $\mathcal{N}$ függvényt, amely egyirányú függvényünk első változata. Erről bizonyítottuk (lásd még [8]), hogy az $\mathcal{N}(x_1, \ldots, x_m)$ függvényérték egész aritmetikát alkalmazó algoritmussal kiszámítható, és az eljárás bonyolultsága polinomiális.

A függvényérték kiszámítására három algoritmust készítettünk, amelyeket MAPLE rendszerben implementáltunk is. A programok futásidejét összehasonlítottuk, a legjobb mátrix reprezentációt alkalmaz és a számításokat moduláris aritmetikával végzi.

Az $\mathcal{N}_P$ egyirányú függvényt a normaforma egy általánosítása alapján kaptuk, amely egyszerűbb szerkezetű, de megtartotta az előző változat jó tulajdonságait.

Az $\mathcal{N}_{P,s}$ függvény már egyirányú hash függvény, amelyet az előző konstrukcióból RSA típusú modulus alkalmazásával kaptunk. Itt a függvényérték kiszámítását egy véges algebra felett végeztük. Bizonyítottuk (lásd még [9]), hogy a függvényérték kiszámításának bonyolultsága polinomiális.

Értekezésünk fő eredménye, hogy bizonyítottuk (lásd még [9]), hogy az $\mathcal{N}_{P,s}$ függvény ütközésmentes, ha a modulus elegendően nagy. Ez pedig azt jelenti, hogy alkalmas egyirányú hash függvénynek.

Teszteléssel sikerült azt is valószínűsíteni, hogy az $\mathcal{N}_{P,s}$ függvény rendelkezik erős lavina hatással, így függvényünk alkalmas minden olyan feladatra, amit az ilyen típusú függvények el szoktak látni.

Bizonyítottuk (lásd még [9]), hogy az $\mathcal{N}_P$ és $\mathcal{N}_{P,s}$ függvények - megfelelő feltételek teljesülése esetén - egyirányú függvények. Továbbá ezekből készíthető - a gyakorlati használat szempontjából fontos - egyirányú függvény család.

Megvizsgáltuk, hogy a gyakorlati alkalmazások szempontjából milyen polinomot célszerű használni a fenti függvények implementációjához.

Végül készítettünk egy példa alkalmazást, amely az $\mathcal{N}_{P,s}$ függvény felhasználásával kiszámítja egy szöveg hash értékét.

A kriptográfia alapjai, fogalmak és definíciók

Tágabb értelemben véve, a titkos, védett kommunikáció tudománya a *kriptológia*, amelynek két - egymással állandóan rivalizáló - ága a *kriptográfia* és a *kriptoanalízis*. A kriptográfia olyan módszerekkel foglalkozik, amelyek biztosítják az üzenetek vagy tárolt információk titkosságát, védettségét, illetve hitelességét. Eszközei matematikai módszereket alkalmazó *algoritmusok*, amelyek használatának pontos leírását a *kriptográfiai protokollok* tartalmazzák. A kriptoanalízis pedig a titok - többnyire illetéktelen - megfejtésére, feltörésére irányuló eljárásokkal foglalkozik.

A titkos kommunikáció folyamatában a *küldő (sender)* és a *fogadó (receiver)* titkos üzenetváltása valósul meg. A küldő rendelkezésére áll a *nyílt szöveg (plaintext)*, amelyből *titkosítás (encryption)* segítségével állítja elő a *kriptoszöveget (ciphertext)*. Ezt a kommunikációs csatorna használatával juttatja el a fogadónak, aki azt *visszafejti (decryption)*, és így megkapja az eredeti nyílt szöveget. Mivel a titkosítás általában matematikai műveleteket használ, a nyílt szöveget számok sorozataként kell interpretálni. Ezt a folyamatot *kódolásnak (encoding)* nevezzük. Az üzenet fogadójának pedig a visszafejtés után el kell végezni a *visszaállítást (decoding)*, amely a számsorozatokból készíti el az eredeti nyílt szöveget.

A kriptoszöveg előállításához a titkosító algoritmuson kívül általában egy úgynevezett *kulcs (key)* is kell, amelynek ismerete mind a titkosításnál, mind pedig a visszafejtésnél szükséges. A megfelelő titkosító és visszafejtő kulcs nélkül általában az algoritmus ismeretében sem lehetséges a titkosítás és visszafejtés műveleteinek elvégzése.

Az összes lehetséges P nyílt szövegek halmazát *nyílt szöveg térnek (plaintext space)* nevezzük és P_T-vel jelöljük. Hasonlóan az összes lehetséges C kriptoszövegek halmaza a *kriptoszöveg tér (ciphertext space)*, amit C_T-vel jelölünk. Az összes lehetséges K kulcsok halmaza pedig a *kulcs tér (key space)*, amit K_T-vel jelölünk. Így az $E : P_T \to C_T$ leképezés jelenti a titkosítást és a $D : C_T \to P_T$ leképezés pedig a visszafejtést.

A kulcs szerepének kiemelésére szokásos az $E_K(M) = C$ jelölés, amely az M üzenet K kulcs segítségével történt titkosítását jelenti. Az eredményül kapott C kriptoszöveg visszafejtését pedig a $D_K(C) = M$ jelöli, ahol a $D_K(E_K(M)) = M$ tulajdonságnak teljesülnie kell.

A titkosító és visszafejtő kulcsnak nem kell föltétlenül megegyeznie.

Az olyan titkosító algoritmust, amelyben a titkosításnál és visszafejtésnél is ugyanazt a K kulcsot kell használni, *szimmetrikus algoritmusnak (symmetric algorithm)* nevezzük. A *nyilvános kulcsú algoritmus (public-key algorithm)* - vagy más szóval *aszimetrikus algoritmus (asymmetric algorithm)* - pedig kulcspárt használ. Az egyik kulcs a *nyilvános kulcs (public-key)*, amellyel a titkosítást végezzük. A másik kulcs pedig a *titkos kulcs (private key)*, amelyet a visszafejtésnél használunk. A nyilvános kulcs bárki számára hozzáférhető - például egy publikus adatbázisból kikereshető -, míg a privát kulcsot csak a tulajdonosa ismeri. Ebben a rendszerben a titkosítás és visszafejtés a következő összefüggésekkel írható le:

$$E_{K_1}(M) = C$$
$$D_{K_2}(C) = M,$$

ahol K_1 a nyilvános, K_2 pedig a titkos kulcsot jelöli. A kulcsokra érvényesnek kell lenni a

$$D_{K_2}(E_{K_1}(M)) = M$$

összefüggésnek.

Lehetséges az is, hogy a küldő és a fogadó fél ugyanaz. Ilyenkor a két fél között nincs kommunikáció, a titkosítás célja az, hogy a küldő és fogadó által tárolt adatokat megvédjük az illetéktelen hozzáféréstől.

A titkosító algoritmusoknak két alaptípusa van. A *blokk titkosítás (block cihper)* a nyílt szöveget meghatározott hosszúságú darabokra vágja, és ezekkel hajtja végre az algoritmus lépéseit. A *folyam titkosítás (stream cipher)* pedig bitenként, folyamatosan alkalmazza az algoritmust a nyílt szövegre.

A *kriptográfiai protokoll* olyan véges számú lépések sorozata, amely két vagy több résztvevő között valósít meg egy kriptográfiai algoritmust.

A küldő és fogadó közötti kommunikáció titkosságát kétféle *betolakodó* támadhatja. A *passzív betolakodó* elsősorban a kommunikációs csatorna lehallgatásával igyekszik információt szerezni. A *passzív támadás (passive attack)* lényege tehát az észrevétlen megfigyelés, amely nem változtatja meg a kommunikáció tartalmát, betartja a protokoll szabályait.

Az *aktív támadás (active attack)* viszont nem tartja be a protokoll szabályait, megváltoztatja a kommunikáció tartalmát. Az aktív támadó először általában passzív módszerekkel dolgozva megszerzi a lehető legtöbb információt, és ezt felhasználva kezd az aktív támadáshoz.

Ha a támadó jogosult felhasználó, azaz a protokoll egyik szereplője, akkor *csalónak (cheater)* szokás nevezni. Az előbbiekhez hasonlóan megkülönböztetünk aktív és passzív csalót. A *passzív csaló (passive cheater)* betartja

a protokoll szabályait, de több információt szerez, mint amennyit a protokoll neki szánt. Az *aktív csaló (active cheater)* nem tartja be a protokoll szabályait, és ilyen módon szerez a neki szántnál lényegesen több információt. Már a meghatározások alapján is látszik, hogy a csalók tevékenysége igen veszélyes lehet egy szervezet informatikai biztonságára, mivel ebben az esetben a szervezethez tartozó, belső felhasználók illegális tevékenységéről van szó.

Míg a *kriptográfusok* a kriptográfiai algoritmusok és protokollok segítségével igyekeznek megvédeni a küldő és fogadó fél közötti kommunikáció titkosságát, addig az *analizálók* ezek feltörésére törekszenek.

A hivatásos kriptoanalitikusok egy részének az a feladata, hogy megkeresse az adott kriptorendszer gyenge pontjait. A megtalált biztonsági lyukak alapján javaslatot tesz a fejlesztők számára a hiba kijavítására, továbbá nyilvános publikációkban felhívja a felhasználók figyelmét az egyes rendszerek hibáira, gyenge pontjaira.

A professzionális kriptoanalitikusok egy másik csoportja nemzetbiztonsági feladatokat lát el, de a feladatuk itt is ugyanaz.

A kártékony analizálókat - akik saját érdekükben igyekeznek a kriptorendszereket feltörni - *hacker*eknek nevezzük. A kriptoanalitikusok és hackerek lényegében ugyanazokat a technikákat alkalmazzák.

A szabványos rendszerek alkalmazása miatt feltételezhető, hogy a kriptoanalitikus ismeri a titkosító és a visszafejtő eljárást is, sőt ez a modern kriptográfia egyik alaptézise. Mindenki ismerheti az alkalmazott algoritmusokat, de nem ismeri a kulcsokat. A feladata tehát ezek esetleges paramétereinek és főként a kulcsoknak a megfejtéséből áll.

A kriptoanalízis alapvető módszerei a következők:

- *Kriptoszöveg alapú támadás:* Az analizáló rendelkezésére csak kriptoszövegek állnak, ezek alapján kell a nyílt szöveget és esetleg a kulcsot meghatározni.

  Ismert: $C_1, C_2, \ldots, C_i$.

  Meghatározandó: vagy $P_1, P_2, \ldots, P_i$ és K, vagy egy olyan algoritmus, amely C_{i+1} alapján meghatározza P_{i+1}-et.

- *Ismert nyílt szöveg alapú támadás:* Az analizálónak ismert nyílt szöveg - kriptoszöveg párok alapján kell meghatározni a kulcsot, hogy az egész kommunikációt vissza tudja fejteni.

  Ismert: $P_1, C_1 = E_K(P_1), P_2, C_2 = E_K(P_2), \ldots, P_i, C_i = E_K(P_i)$.

  Meghatározandó: vagy a K kulcs, vagy egy olyan algoritmus, amely C_{i+1} alapján meghatározza P_{i+1}-et.

- *Választott nyílt szöveg alapú támadás:* Az analizálónak nemcsak nyílt szöveg - kriptoszöveg párok állnak rendelkezésére, hanem a nyílt szövegeket az analizáló választhatja.

  Ismert: $P_1, C_1 = E_K(P_1), P_2, C_2 = E_K(P_2), \ldots, P_i, C_i = E_K(P_i)$, ahol $P_1, P_2, \ldots, P_i$ az analizáló által választott nyílt szöveg.

  Meghatározandó: vagy a K kulcs, vagy egy olyan algoritmus, amely C_{i+1} alapján meghatározza P_{i+1}-et.

- *Módosított választott nyílt szöveg alapú támadás:* Az előző speciális esete, amelyben az analizáló módosíthatja a kiválasztott nyílt szöveget aszerint, hogy mi volt az eredménye az előző titkosításnak, illetve visszafejtésnek.

- *Választott kriptoszöveg alapú támadás:* Az analizáló megválaszthatja azokat a kriptoszövegeket, amelyeknek visszafejtését szeretné megismerni. Az így kapott párok alapján kell a kulcsot meghatározni.

  Ismert: $C_1, P_1 = D_K(C_1), C_2, P_2 = D_K(C_2), \ldots, C_i, P_i = D_K(C_i)$, ahol $C_1, C_2, \ldots, C_i$ az analizáló által választott kriptoszöveg.

  Meghatározandó: a K kulcs.

- *Választott kulcs alapú támadás:* Ritkán előforduló analizáló módszer, mivel a kulcs megválasztása általában nem lehetséges. A kulcsgenerálás gyengeségeire épül, az egyes kulcsok közötti összefüggéseket igyekszik kihasználni.

- *Kulcs megszerzésén alapuló támadás:* A támadó nemcsak a kriptoanalízis módszereit használja, hanem zsaroláshoz, megvesztegetéshez és egyéb hagyományos illegális módszerekhez folyamodik a kulcs megszerzése érdekében. Gyakran ez a leghatékonyabb módja egy védelmi rendszer feltörésének.

Látható, hogy az analízis nem mindig jár tökéletes eredménnyel, az analizáló csak részben tudja a protokollt, illetve az algoritmust hatástalanítani. Az analizálás eredményessége az alábbi csoportokba sorolható:

- *Teljes feltörés:* az analizáló megtalálja a helyes K kulcsot, és bármilyen üzenetet vissza tud vele fejteni. A $D_K(C) = P$ összefüggés tetszőleges C esetén teljesül. Ekkor a küldő és fogadó közötti teljes kommunikáció ismertté válik az analizáló számára.

- *Globális megfejtés:* az analizáló a K kulcs ismerete nélkül talál egy A alternatív algoritmust, amelyre az $A(C) = P$ minden C kriptoszöveg esetén fennáll. Ekkor az analizáló számára szintén ismertté válik a teljes kommunikáció.

- *Lokális megfejtés:* az analizálónak sikerül visszafejteni a megszerzett kriptoszövegek egy részét, így a kommunikáció felfedése csak részben történik meg.

- *Információnyerés:* az analizálónak csak különféle információkat sikerül szerezni a kulcsról vagy a nyílt szövegről. A kommunikáció felfedése nem történik meg, csak például a nyílt szöveg formátuma, a kulcs egy részlete válik ismertté az analizáló számára.

Egy algoritmust *feltétel nélküli biztonságúnak* nevezünk, ha az analizáló bármennyi kriptoszöveg alapján sem tudja az algoritmust feltörni. A gyakorlati használat számára ilyen algoritmusok sajnos nem léteznek. Ezt a kritériumot egyedül a *one-time pad* nevű algoritmus teljesíti, de ez a gyakorlatban nem használható.

Minden más kriptográfiai rendszer elvileg a kriptoszöveg alapú támadás módszerével feltörhető. Ez tulajdonképpen az összes lehetséges kulcs végigpróbálását jelenti. Ezt a módszert *primitív próbálgatásnak (brute-force attack)* nevezik. Ha a kriptográfiai rendszer jól tervezett, ennek a módszernek az időigénye több ezer év is lehet.

Meg kell elégednünk a *kiszámíthatóan biztonságos (computationally secure)* algoritmusok használatával, amelyek biztonsága valamilyen nehéz probléma megoldhatatlanságán alapul. A két legismertebb ilyen probléma a *faktorizáció* és a *diszkrét logaritmus*.

A faktorizáció problémája lényegesen bonyolultabb, mint a prím tulajdonság megállapítása, amelyről ma már tudjuk, hogy a P bonyolultsági osztályba tartozik (lásd [1]). Vannak olyan implementációk, amelyek több ezer decimális jegyű számokról el tudják dönteni, hogy prímek-e. Ilyen például F. Morain programja (lásd [50]).

Ezzel szemben a legjobb ismert faktorizációs algoritmusok csak szubexponenciális bonyolultságúak és a legnagyobb összetett szám, amelyet általános célú algoritmussal faktorizáltak 174 decimális jegyű (RSA-174 probléma, lásd *www.rsasecurity.com*[1] honlapot). Az igazi lélektani áttörést azonban a korábbi, RSA-158 probléma megoldása jelentette (lásd [3]).

[1] pontosabban: http://www.rsasecurity.com/rsalabs/challenges/factoring/numbers.html

A biztonságosnak tartott RSA algoritmus feltörhetősége egy legalább 200 decimális jegyű ösztetett szám faktorizálásán múlik.

A hatékony faktorizáló algoritmusok konstruálásánál is elsősorban az algebrai számelmélet eredményei hasznosíthatók. A legismertebb modern faktorizációs eljárások a következők:

- lánctört módszer,
- Pollard féle Monte Carlo algoritmus,
- elliptikus görbe módszer,
- kvadratikus szita algoritmus,
- algebrai számtest szita algoritmus.

Egy kriptográfiai rendszer biztonsága nyugodtan alapozható a faktorizáció problémájának nehézségére.

Hasonlóan nehéz probléma az úgynevezett diszkrét logaritmus kiszámítása, amely az

$$x \equiv a^n \pmod{p}$$

moduláris hatvány kiszámításának az inverze, azaz x, a és p ismeretében az n meghatározását jelenti.

A moduláris hatványozás intelligens hatványozással viszonylag egyszerűen kivitelezhető. A diszkrét logaritmus kiszámítása sokkal nehezebb feladat.

Ennek a problémának a kriptográfiai alkalmazását először Diffie és Hellman javasolta egy nyilvános kulcsú rendszer kidolgozásában.

A diszkrét logaritmus kiszámítására számos módszert dolgoztak ki, de bizonyos nagyságrendet meghaladva ezek mindegyike általában - a nagyon hosszú futási idő miatt - csődöt mond. Lényegében mindegyik módszer alapja az, hogy készíteni kell egy nagy méretű logaritmustáblát, majd ennek segítségével módszeres próbálgatással lehet eredményt elérni. Az algoritmusok részletesebb tanulmányozása azt mutatja, hogy a fejlettebbek meglepően eredményesek, ezért nem árt az óvatosság.

A probléma megoldására használt legismertebb módszerek a következők:

- Shanks-algoritmus(Baby-Step Giant-Step Method),
- Pohling-Hellman-Silver-algoritmus,
- indexkalkulus-algoritmus,

- Coppersmith-algoritmus.

Az *elliptikus görbék* alkalmazása is egyre jobban terjed, főként a nyilvános kulcsú kriptográfiai rendszerek konstruálásánál. Egy F_q véges test feletti $E(F_q)$ elliptikus görbe bizonyos tulajdonságú pontjainak meghatározása esetén szintén a diszkrét logaritmus kiszámításának problémájába ütközünk. Ha S és T az elliptikus görbe pontjai, akkor igen nehéz meghatározni azt az m egész számot, amelyre igaz lesz a $T = mS$ egyenlőség.

A véges test feletti elliptikus görbét alkalmazó kriptorendszerek előnyös tulajdonsága, hogy viszonylag kis méretű kulcsok alkalmazásával is elérhető a megfelelő informatikai biztonság.

Látható tehát, hogy a számelmélet tartalmaz olyan problémákat, amelyek megoldása a tudomány jelenlegi állása szerint nagyon nehéz, és így a kriptográfiai alkalmazásokban jó eredménnyel használhatók.

Az algoritmusok működési sebességének méréséhez a bonyolultságelmélet eszközeit szokás alkalmazni. Ez az elmélet az algoritmusok futási idejének elemzésével is behatóan foglalkozik. A kriptográfia szempontjából ez igen fontos terület, mivel lényeges, hogy a titkosító és visszafejtő algoritmusok minél egyszerűbb és gyorsabb módon működjenek. A kriptorendszerek feltörését célzó algoritmusok vizsgálata hasonlóan fontos, hiszen egy rendszer csak akkor tekinthető biztonságosnak, ha nem létezik belátható időn belül eredményt szolgáltató feltörő algoritmus.

Az *algoritmus* fogalmának számos hosszadalmas és matematikailag korrekt leírása létezik, de számunkra most a következő erősen leegyszerűsített megfogalmazás is megfelel: Az algoritmus egy olyan módszer, amely egy bizonyos megadott input esetén véges sok lépés után az elvárt outputot szolgáltatja.

Az algoritmusokkal kapcsolatban fontos annak tisztázása, hogy milyen feltételek esetén fejeződnek be véges sok lépés után, korrekt eredményt szolgáltatnak-e, végrehajthatók-e egyértelműen. Az algoritmus gyakorlati megvalósítása szempontjából pedig az alábbiak vizsgálata szükséges:

- *Időbonyolultság (time complexity):* az algoritmus végrehajtásához szükséges idő becslése átlagos és legrosszabb esetre.

- *Tárolási bonyolultság (space complexity):* az algoritmus működéséhez szükséges számítógépes tárolókapacitás vizsgálatát jelenti, beleértve az input és output méretét is.

Ez utóbbit bitekben mérjük. Például az n pozitív egész szám mérete $m = \lfloor \log_2 n \rfloor + 1$ bit. Az algoritmusok futásidejét pedig bit műveletekkel mérhetjük, ami a $\{0,1\}^*$ halmazon végzett aritmetikai és logikai műveletek számát jelenti.

Egy algoritmus *kiszámítási bonyolultságát (computational complexity)* mérhetjük az időbonyolultság (T) és a tárolási bonyolultág (S) értékeivel. Ha egy algoritmus inputjának méretét m-el jelöljük, akkor a T és S előállítható m valamilyen függvényeként. Például, ha egy algoritmus futási idejét a $3m^2 + 2m + 8$ függvény jellemzi, akkor annak kiszámítási bonyolultsága lényegében az m^2-el arányos, amit a $O(m^2)$ fejez ki. Így a bonyolultság meghatározása a használt számítógépes rendszerektől függetlenül lehetséges.

Például, ha $T = O(m)$ jellemzi az időbonyolultságot, akkor megduplázva az input méretét, a futási idő is lényegében megduplázódik. Ha $T = O(2^m)$, akkor pedig már egy bittel növelve az input méretét, megduplázódik a futási idő.

Az algoritmusokat osztályokba sorolhatjuk az időbonyolultság vagy tárolási bonyolultság szempontjából az alábbiak szerint:

- *Konstans:* független az m-től, $O(1)$ jellemzi.
- *Lineáris:* az algoritmus bonyolultsága lineárisan függ m-től, tehát $O(m)$ jellemzi.
- *Polinomiális:* az algoritmus bonyolultsága n-ed fokú polinommal jellemezhető, amit kifejez a $O(m^n)$, ahol n konstans. Kvadratikusnak nevezzük a bonyolultságot, ha $O(m^2)$ jellemzi.
- *Szuperpolinomiális:* a bonyolultság $O(c^{f(m)})$-el jellemezhető, ahol c konstans $f(m)$ pedig legfeljebb lineáris polinom.
- *Exponenciális:* a bonyolultságot $O(t^{f(m)})$ írja le, ahol $t > 1$ konstans és $f(m)$ tetszőleges fokszámú polinom.

A kriptográfiai rendszerek tervezői arra törekednek, hogy a titkosító és visszafejtő algoritmus bonyolultsága legfeljebb polinomiális legyen, ugyanakkor pedig a legjobb lehetséges feltörő algoritmus bonyolultsága haladja meg a polinomiálist. Az ilyen tulajdonságú algoritmus valószínűleg megfelelő informatikai biztonságot nyújt.

Az ismert kriptográfiai algoritmusok esetén a tárolási bonyolultság általában nem jelent problémát, így ezeknél az időbonyolultság a döntő.

Az alábbi táblázat az algoritmusok különféle osztályait hasonlítja össze a

futási idő szempontjából ($m = 10^6$ és a számolási kapacitás 10^6 művelet/sec.):

Osztály	Bonyolultság	Műveletek száma	Idő
Konstans	$O(1)$	1	1 μsec.
Lineáris	$O(m)$	10^6	1 sec.
Kvadratikus	$O(m^2)$	10^{12}	11, 6 nap
Köbös	$O(m^3)$	10^{18}	32000 év
Exponenciális	$O(2^m)$	10^{301030}	∞

A táblázatban szereplő ∞ jel itt nem a végtelen hosszú időt jelenti, "csak" az univerzum korának 10^{301006}-szorosát.

A különféle problémák osztályokba sorolhatók, aszerint, hogy milyen módszerrel oldhatók meg. Ennek tárgyalásához szükséges a *Turing-gép* fogalmával megismerkedni, ami tulajdonképpen az algoritmus fogalom egy változata.

Determinisztikus Turing-gépnek nevezzük a (Q, Σ, M, Φ) absztrakt objektumot, ahol Q a gép állapotainak halmaza, Σ a gép által használt szalagábécé jeleinek halmaza, M a gép író-olvasó fejének mozgásait (jobbra lép, balra lép, mozdulatlan) tartalmazó halmaz és Φ pedig a gép működését meghatározó leképezés, amelynek definíciója az alábbi:

$$\Phi : (Q \times \Sigma) \to (Q \times \Sigma \times M).$$

Ez tulajdonképpen azt jelenti, hogy a Turing-gép valamely állapotának és az olvasott jelnek a hatására a leképezés által meghatározott új állapotba kerül, és ír vagy olvas az író-olvasó fej szalag feletti elmozdításával. Tehát előre meghatározott lépéseket hajt végre, azaz ez egy determinisztikus algoritmus.

A *nem determinisztikus Turing-gép* hasonlóan működik, azzal a lényeges eltéréssel, hogy nem előre meghatározott lépéseket hajt végre, hanem "megsejti" egy probléma megoldását, és ellenőrzi a megoldás helyességét. Ezt a Φ leképezés segítségével az alábbi módon lehet kifejezni:

$$\Phi : (Q \times \Sigma) \to H \subseteq (Q \times \Sigma \times M).$$

Ez azt jelenti, hogy minden egyes lépésben valamilyen valószínűségű alternatívákból választhat, tehát ez egy valószínűségi algoritmus.

Azokat a problémákat, amelyek polinomiális időbonyolultságú algoritmussal megoldhatók *kezelhetőnek (tractable)* nevezzük, amelyeket pedig nem tudunk így megoldani, *kezelhetetlen (intractable)* problémának nevezzük. Ez utóbbira használatos a *nehéz (hard)* megjelölés is.

A problémák tehát bonyolultsági osztályokba sorolhatók. Ezek az alábbiak:

- *P osztály:* Az ide tartozó problémák kezelhetők, polinomiális időbonyolultságú algoritmussal megoldhatók. Ez azt is jelenti, hogy determinisztikus Turing-gép segítségével megtalálható a problémák megoldása, ezért ezeket determinisztikus időbonyolultságú problémáknak is nevezik.

- *NP osztály:* Ez problémák olyan halmaza, amelyek nem determinisztikus Turing-géppel oldhatók meg, tehát a megoldáshoz szükséges valamilyen intuíció. A megoldás helyességének ellenőrzése legfeljebb polinomiális időbonyolultságú algoritmussal lehetséges.

- *NP-teljes osztály:* Az ide tartozó problémák az NP osztályban vannak és *NP-nehezek*. Ez azt jelenti, hogy található olyan leképezés, amely polinomiális időbonyolultsággal minden NP osztálybeli problémát redukál NP-nehéz problémára.

  Analóg módon kezelhető az osztályozás szempontjából a tárolási bonyolultság kérdése is. Itt a Turing-gép eredeti szalagméretét kell összevetni a gép által a számítás során használt maximális szalagmérettel. Mivel a gép egy algoritmus végrehajtása során írhat is a szalagra, a hossznövekedés jelentős is lehet.

- *P-SPACE osztály:* Ennél az osztálynál a Turing-gép szalaghossz növekedése valamilyen polinommal kifejezhető. Ez az osztály az NP osztály problémáinál nehezebb problémákat is tartalmaz.

  Lehetne egy NP-SPACE osztályt is bevezetni, ahol a szalaghossz növekedés már nem fejezhető ki polinomiális összefüggéssel, de ebben az esetben is az előző osztály problémáit kapnánk.

- *P-SPACE-teljes osztály:* Hasonlóan definiálható, mint az NP-teljes osztály.

- *EXPTIME osztály:* Ide az exponenciális időbonyolultságú problémák tartoznak.

Az egyes osztályok közötti tartalmazási relációk nem mindenütt tisztázottak, vannak megoldatlan problémák. A $P = NP$ kérdés az elméleti számítástudomány legnagyobb problémája.

A korábban említett faktorizáció és diszkrét logaritmus problémák az NP osztályhoz tartoznak. Azt viszont egyikről sem tudjuk, hogy a P osztályba tartoznak-e. Ezt csak a prímtesztről sikerült bizonyítani. Az pedig nyitott kérdés, hogy a faktorizáció és a diszkrét logaritmus problémák P-hez

tartoznak-e. Gyakorlati szempontból az a helyzet, hogy léteznek olyan szoftverek, amelyek 3000 jegyű számok prímtulajdonságát egyértelműen és gyorsan el tudják dönteni. Ugyanakkor egy olyan számot, amelyik két 100 jegyű prímszám szorzata, a mai technológiával nem lehet prímtényezőkre bontani.

A szakirodalom több mint 1500 NP-teljes problémát ismer (lásd [25]). Ilyenek például az alábbiak:

- *Utazó ügynök probléma (lásd [43]):* Az ügynöknek n települést kell felkeresnie gépkocsival úgy, hogy csak egy tank üzemanyagot használhat fel, azaz csak meghatározott távolságot tehet meg. Lehetséges-e az n települést összekötő útvonal bejárása úgy, hogy minden települést csak egyszer érint?

- *Házasságkötési probléma (lásd [29]):* Adott n nő, n férfi és n lelkész valamint egy lista, amely az elfogadható házasságkötéseket tartalmazza. A lista minden sora egy olyan férfit, nőt és lelkészt tartalmaz, akik együtt szerepelhetnek egy házasságkötésnél. Lehetséges-e n olyan házasságkötést létrehozni, amelyben minden férfi, nő és lelkész szerepel, valamint az előre meghatározott lista elvárásainak is megfelel?

- *A három változós logikai kifejezések problémája (3SAT):* Adott n logikai kifejezés, amelyek mindegyike három változót tartalmaz. Lehetséges-e a változóknak olyan állandó logikai értéket adni, amelyekkel az összes kifejezés értéke igaz lesz? További részletek találhatók C. H. Papadimitriou [51] könyvében.

A modern - bonyolultságelméleti alapon kidolgozott - kriptográfia felhagy azzal a feltevéssel, hogy a betolakodónak tetszőlegesen nagy számítási kapacitás áll rendelkezésére. E helyett azt feltételezi, hogy a betolakodónak csak valamilyen kezelhető módon korlátozott számítási kapacitása van. Ez pontosabban azt jelenti, hogy a betolakodó egy valószínűségi algoritmust (nem determinisztikus Turing-gépet) használ, amelynek a futási ideje polinomiális.

A titkosítás, a visszafejtés és a betolakodó algoritmusainak futásidejét a k biztonsági paraméter függvényeként tekintjük. A k paraméter értéke megegyezik a kriptorendszer inputjának bináris hosszával, és rögzített a rendszer teljes működése alatt. A polinomiális futásidejű algoritmus tehát azt jelenti, hogy a futásidő a k paraméter valamilyen polinomja.

A modern kriptográfia egy kriptorendszer feltörésére a *megvalósíthatatlan (infeasibility)* és nem a *lehetetlen (impossibility)* jelzőt használja.

Megjegyzendő, hogy az itt definiált új betolakodó fogalom esetén biztonságosnak tekintett kriptorendszer nem föltétlenül biztonságos a korábban

használt korlátlan számítási kapacitással rendelkező betolakodó fogalom szerint. Erre az esetre csak a one-time pad (Vernam titkosítás, lásd [35], [47]) tekinthető biztonságosnak.

Az egyik alapvető fontosságú kriptográfiai eszköz az egyirányú függvény. Az ilyen függvény értékét "könnyű" kiszámítani, de invertálni "nehéz". A modern kriptográfiában a "könnyű" azt jelenti, hogy PPT, azaz polinomiális futásidejű valószínűségi algoritmussal kiszámítható. A "nehéz" pedig azt jelenti, hogy a kiszámítás valószínűsége PPT algoritmussal "kicsi".

A kriptográfia különféle területein jelentős szerepet töltenek be az egyirányú függvények. Többek között jelszó ellenőrzésére, üzenetkivonat és digitális aláírás készítésénél használják, de szerepük van a kriptográfiailag megbízható véletlen számok előállításában is. A kriptográfiai protokollok fontos alkotóeleme. A közismert egyirányú függvények leírása, valamint ezek különféle kriptográfiai alkalmazása például a [60] és [47] könyvekben található meg.

Igen fontos megjegyezni, hogy a széles körű használat ellenére jelenleg nem ismert ***matematikai bizonyítás*** *arra, hogy ezek az egyirányú függvények valóban léteznek. Elterjedt használatuk létjogosultságát a gyakorlati tapasztalatok igazolják. Helyesebb ezeket a függvényeket egyirányú függvény jelölteknek nevezni. A továbbiakban azonban az irodalomban szokásos egyirányú függvény elnevezést használjuk*

A bonyolultságelmélet jelenlegi állása szerint tehát még nem sikerült bizonyítani egyirányú függvény létezését. Ha feltesszük, hogy $P \neq NP$, akkor létezhet egyirányú függvény. A betolakodó korlátozott ideig használhat PPT algoritmust, így ez tulajdonképpen egy BPP algoritmus. A $BPP \subseteq NP$ kérdés viszont nyitott, mivel ez éppen a $P \neq NP$ probléma megoldatlanságára vezet (lásd [51] és [27]).

A $P \neq NP$ feltételezés csak azt biztosítja, hogy a kriptorendszert a *legrosszabb esetben* nehéz feltörni. Ez nem zárja ki annak lehetőségét, hogy a feltörés *majdnem minden esetben* sikeres. Lehetséges olyan "kriptorendszert" készíteni, amelynek feltörési problémája NP-teljes, mégis van hozzá egy olyan algoritmus, amellyel a feltörés az esetek 99 %-ában sikeres. Ilyen például egy hátizsák problémán alapuló Merkle - Hellman [49] kriptorendszer, amely feltörhető (lásd [63] és [41]) az LLL algoritmus segítségével. Így tehát a biztonságos kriptorendszer létezésének szükséges feltétele, hogy létezzen NP-beli probléma, amely majdnem mindig nehezen vagy *legalább átlagosan nehezen törhető fel*. Ezt azonban a $P \neq NP$ feltétel nem garantálja.

A kriptográfiában potenciálisan hasznos, *átlagosan nehéz* problémák jel-

lemzésére Leonid Levin [44] javasolt szép megoldást.

Átlagosan NP-teljes probléma rácsokban rövid vektorok keresése, ezen alapul az Ajtai-Dwork kriptorendszer (lásd [2]).

Egészségügyi informatika és kriptográfia

A katonai, diplomáciai és ipari adatok védelmén túl igen fontos az egészségügyben keletkező nagy mennyiségű különleges személyes adat megfelelő kezelése és védelme.

Az egészségügyi adatvédelem az utóbbi években erősödő problémaként jelentkezett. Ennek egyik oka az információ- és kommunikációtechnológia fejlődése, amely lehetővé tette az elektronikus adatkezelés tömegessé válását. Az információk globalizációja egyrészt nagyban segíti az egészségügyi ellátás hatékonyságát, de az információk elérhetősége ugyanakkor az illetéktelen hozzáférés lehetőségét is megsokszorozta, így felértékelődtek az adatvédelem, adatbiztonság kérdései.

A problémák felerősödésének másik oka a személyiségi jogok általános felértékelődése, ami a politikai rendszerváltáshoz kapcsolódik. Az értékrendváltás igen markánsan nyilvánult meg az egészségügy területén.

A két tényező nagyjából egyidőben jelent meg, így elkerülhetetlenné vált az adatkezelés hathatós szabályozása, amely egy bonyolult előíráshalmaz létrehozását jelentette. Ezek egy része jogszabály, más részük szerződéseken alapul, végül a maradék rész az intézmény által kidolgozott helyi szabályozás, adatvédelmi szabályzat (lásd [40]).

A szabályozás azonban nyilvánvalóan nem oldja meg automatikusan az összes adatkezelési, adatvédelmi problémát. A szabályozás végrehajtásához megfelelő informatikai eszközök szükségesek. Ezek természetesen a kriptográfia eszközei, a protokollok, az algoritmusok és egyéb technikák, amelyeknél igen jelentős szerep jut az egyirányú, és az egyirányú hash függvényeknek.

Talán nem könnyelműség kijelenteni, hogy az egészségügy intézményeiben lehetséges a biztonságos papírmentes adminisztráció és kommunikáció megvalósítása. Ez természetesen nem azt jelenti, hogy minden egészségügyi adat kizárólag számítógépes adattárolókon és biztosítási mágneskártyákon található meg. A beteg valószínűleg igényli a hagyományosan írásba foglalt kórházi zárójelentést, de például a betegforgalmi napló adatait elegendő kizárólag számítógépen tárolni.

Az egészségügyi eljárások papírmentessé tételéhez a hagyományos, papír

hordozójú iratok digitális változatainak hitelesítő eljárásait, a megfelelő kriptográfiai protokollokat kell kidolgozni (lásd [39]). A titkosítás, a partner azonosítás, a digitális aláírás, az időpecsét és a nyilvános kulcsú adatkommunikáció alkalmazása igen fejlett, csaknem teljesen papírmentes adminisztráció bevezetését teszi lehetővé (lásd [7] és [20]). Jelentős fejlődést lehet elérni az Internetes technológiák bevezetésével is (lásd [6]).

Vannak azonban olyan adatkezelési, adatvédelmi problémák, amelyek hatékony megoldása csak komplex módon kezelhető, mivel ezek mögött súlyos szakmai, etikai és jogi konfliktusok húzódnak meg. Ilyen problémát okoz például a személyiségi jogok és a közegészségügyi érdekek, illetve a reális felvilágosítás joga és az állapotromlás kockázata között feszülő ellentmondás. További problémát okoz az egészségügyi személyazonosító és a tájékoztatás kérdése. Különösen érdekes probléma az adatkezelés pontossága, a nagy mennyiségű adat megfelelő archiválása, valamint az adatkezelés célhoz kötött torzítása, például a finanszírozási pozíció javítása miatt.

Ezek a csaknem megoldatlan problémák mutatják, hogy az informatikai, kriptográfiai eszközök nem mindenhatóak, nem alkalmasak minden probléma megoldására.

Az egészségügyi informatika és az egészségügy kriptográfiai megoldásainak tanulmányozásához ajánjuk a [21] és [33] könyveket.

Egyirányú függvények

Definíciók és tulajdonságok

Most az egyes fogalmak pontos definícióját adjuk meg. Az irodalomban azonban egyes fogalmakra többféle megjelölést is használnak, az ezek közötti azonosságot minden esetben jelezni fogjuk. Elsősorban a modern - bonyolultságelméleti alapon kidolgozott - kriptográfia fogalmait fogjuk használni, amely a determinisztikus algoritmusok helyett valószínűségi algoritmusokat alkalmaz. Ezt a módszert először Goldwasser és Micali [28] javasolta. A fogalmak meghatározásához a [47] és [27] könyveket használtuk.

Az *egyirányú függvény* erős és gyenge tipusát szokás megkülönböztetni. Mindkét definíció két módon, uniform és nem uniform invertálási algoritmus felhasználásával is megfogalmazható. Így összesen négy definíció létezik az egyirányú függvény meghatározására. Ezek leírásához szükségünk lesz az *elhanyagolható függvény* fogalmára:

1. Definíció. *A $\nu : \mathcal{N} \to \mathbb{R}$ függvény **elhanyagolható**, ha minden $c \geq 0$ konstanshoz létezik k_c egész úgy, hogy minden $k \geq k_c$ esetén $\nu(k) < k^{-c}$.*

A ν elhanyagolható függvény segítségével már megadhatók az egyirányú függvény különféle definíciói. Elsősorban az első definició szerinti függvényt szokás egyirányú függvénynek tekinteni, a további definíciók kevésbé használtak.

2. Definíció. *Az $f : \{0,1\}^* \to \{0,1\}^*$ függvény **(erős) egyirányú** függvény ha:*
(1) Létezik PPT (polinomiális valószínűségi) algoritmus, amelynek x inputja esetén az output $f(x)$ lesz.
(2) Bármely $\mathcal{A}$ PPT algoritmushoz létezik egy $\nu_{\mathcal{A}}$ elhanyagolható függvény úgy,

hogy bármely elegendően nagy k esetén:

$$\mathbf{P}\left[f(z)=y: \quad x \overset{R}{\in} \{0,1\}^k; \quad y=f(x); \quad z=\mathcal{A}(k,y)\right] \leq \nu_{\mathcal{A}}(k),$$

ahol a valószínűséget úgy vesszük, hogy az x-ek befutják lehetséges értékeiket és az $\mathcal{A}$ algoritmus is az érmefeldobásait.

Itt az $x \overset{R}{\in} \{0,1\}^k$ azt jelenti, hogy az x szót véletlenszerűen választjuk az összes lehetséges k hosszúságú bináris szavak közül.

A definícióban szereplő k konstans az úgynevezett biztonsági paraméter, amely jellemez egy adott kriptorendszert. A k paraméter általában megegyezik az x input bináris hosszával.

A definíció szerint a következő esemény valószínűsége elhanyagolhatóan kicsi: véletlenszerűen választunk egy k hosszúságú x szót, meghatározzuk $y = f(x)$-et, majd az $\mathcal{A}$ algoritmus k és y inputra megadja a z outputot, amelyre teljesül az $f(z) = y$ egyenlőség.

Most megadjuk az előbbi definíció gyenge változatát.

3. Definíció. *Az $f : \{0,1\}^* \to \{0,1\}^*$függvény **gyenge egyirányú** függvény ha:*
(1) Létezik PPT algoritmus, amelynek x inputja esetén az output $f(x)$ lesz.
(2) Bármely $\mathcal{A}$ PPT algoritmushoz létezik egy Q polinom úgy, hogy bármely elegendően nagy k esetén:

$$\mathbf{P}\left[f(z)\neq y: \quad x \overset{R}{\in} \{0,1\}^k; \quad y=f(x); \quad z=\mathcal{A}(k,y)\right] \geq \frac{1}{Q(k)},$$

ahol a valószínűséget úgy vesszük, hogy az x-ek befutják lehetséges értékeiket és az $\mathcal{A}$ algoritmus is az érmefeldobásait.

Bizonyítható, hogy gyenge egyirányú függvény akkor és csak akkor létezik, ha erős egyirányú függvény létezik. A bizonyítás vázlata megtalálható a [27] könyvben.

Az előbbi két definícióban az $\mathcal{A}$ invertáló algoritmus polinomiális valószínűségi (PPT) algoritmus volt. Ha megengedünk nem uniform invertálási algoritmust, akkor ezek erősebb változatát adhatjuk meg.

4. Definíció. *Az $f : \{0,1\}^* \to \{0,1\}^*$függvény **nem uniform egyirányú** függvény ha:*

(1) Létezik PPT algoritmus, amelynek x inputja esetén az output $f(x)$ lesz.
(2) Bármely $\mathcal{A} = \{M_k\}_{k\in\mathbb{N}}$ polinomiális algoritmus családhoz (még nem uniformhoz is) létezik egy $\nu_{\mathcal{A}}$ elhanyagolható függvény úgy, hogy bármely elegendően nagy k esetén:

$$\mathbf{P}\left[f(z) = y : \quad x \overset{R}{\in} \{0,1\}^k; \quad y = f(x); \quad z = M_k(y)\right] \leq \nu_{\mathcal{A}}(k),$$

ahol a valószínűséget úgy vesszük, hogy az x-ek befutják lehetséges értékeiket és az M_k algoritmus család is az érmefeldobásait.

Hasonlóan fogalmazható meg a gyenge egyirányú függvény esetében is a nem uniform eset.

5. Definíció. *Az $f : \{0,1\}^* \to \{0,1\}^*$ függvény* ***nem uniform, gyenge egyirányú*** *függvény ha:*
(1) Létezik PPT algoritmus, amelynek x inputja esetén az output $f(x)$ lesz.
(2) Bármely $\mathcal{A} = \{M_k\}_{k\in\mathbb{N}}$ polinomiális algoritmus családhoz (még nem uniformhoz is) létezik egy Q polinom úgy, hogy bármely elegendően nagy k esetén:

$$\mathbf{P}\left[f(z) \neq y : \quad x \overset{R}{\in} \{0,1\}^k; \quad y = f(x); \quad z = M_k(y)\right] \geq \frac{1}{Q(k)},$$

ahol a valószínűséget úgy vesszük, hogy az x-ek befutják lehetséges értékeiket és az M_k algoritmus család is az érmefeldobásait.

Lehetséges, de nem túlságosan valószínű, hogy (erős) egyirányú függvény létezik, de nem uniform nem létezik.

Az azonban bizonyítható, hogy ha az f függvény nem uniform egyirányú függvény, akkor f (erős) egyirányú függvény is. A bizonyítás vázlata megtalálható a [27] könyvben.

A definíciók alapján látható, hogy a biztonsági garancia valószínűségi. A betolakodó nem alkalmatlan a függvény invertálására, hanem csak kicsi annak valószínűsége, hogy ez sikerül neki.

A betolakodó nem az x-et akarja megtalálni - hiszen az csaknem megvalósíthatatlan - hanem az y valamelyik inverzét. Ha az f függvény kölcsönösen egyértelmű, akkor y egyetlen inverze az x.

A definíció alapján triviális, hogy az f függvény outputjának mérete korlátozott a k paraméter egy polinomjával, így $f(x)$ polinom időben kiszámítható.

Mivel a definíció az f^{-1} függvény értékeinek kiszámítási nehézségén alapul, csak megfelelően nagy k értékekre ad elfogadható biztonságot. Ha k nem elég nagy, az invertálás a lehetséges x értékek módszeres kipróbálásával sikerülhet, ami exponenciális időben mindig elvégezhető. Ez a gyakorlatban azt jelenti, hogy ma már az 56 bites DES kulcs nem tekinthető biztonságosnak (lásd [10] és [46]).

Ezek a definíciók az egyirányú függvények gyakorlati használatához nem illeszkednek jól, de az elméleti vizsgálatokhoz megfelelőek. A gyakorlati alkalmazásokban nem egy egyirányú függvényt, hanem egy *egyirányú függvény családot* szokás használni.

Most megadjuk ennek a definícióját.

6. Definíció. *Legyen I egy indexhalmaz. Minden $i \in I$ esetén D_i és R_i véges számhalmazok. Az*

$$F = \{f_i : D_i \mapsto R_i\}_{i \in I}$$

halmazt ***egyirányú függvény családnak*** *nevezzük, ha teljesíti az alábbi feltételeket:*
(1) Létezik $\mathcal{S}_1$ PPT algoritmus, ami a k inputra előállít egy legfeljebb k bináris hosszúságú $i \in I$ indexet.
(2) Létezik $\mathcal{S}_2$ PPT algoritmus, amely az $i \in I$ inputra $x \in D_i$ outputot szolgáltat.
(3) Létezik $\mathcal{A}_1$ PPT algoritmus, amely az $i \in I$ és $x \in D_i$ inputra $f_i(x)$ outputot szolgáltat, azaz $\mathcal{A}_1(i, x) = f_i(x)$.
(4) Bármely $\mathcal{A}$ PPT algoritmushoz létezik egy $\nu_{\mathcal{A}}$ elhanyagolható függvény úgy, hogy bármely elegendően nagy k esetén:

$$\mathbf{P}\left[f_i(z) = y : \quad i \overset{R}{\in} I, \quad x \overset{R}{\in} D_i; \quad y = f_i(x); \quad z = \mathcal{A}(i, y)\right] \leq \nu_{\mathcal{A}}(k).$$

Ahol a valószínűséget úgy értelmezzük, hogy az i és x értékek befutják lehetséges értékeiket és az $\mathcal{A}$ algoritmus is az érmefeldobásait.

Az $\mathcal{S}_1$ algoritmus választja ki az indexet, az $\mathcal{S}_2$ az indexnek megfelelő értelmezési tartományból az x értéket, az $\mathcal{A}_1$ meghatározza a függvényértéket az $\mathcal{A}$ pedig megkísérli invertáni az f_i függvényt.

Bizonyítható, hogy egyirányú függvény család akkor és csak akkor létezik, ha létezik egyirányú függvény. A bizonyítás megtalálható a [27] könyvben.

Az egyirányú függvények között igen nagy jelentőségűek az *egyirányú csapóajtós* függvények, amelyek bizonyos titok ismeretében könnyen invertálhatók. Ilyen például az RSA.

Szintén nagy szerepe van a kriptográfiában a *hash függvényeknek*. Most ezek definícióját adjuk meg.

7. Definíció. *A h függvényt* ***hash*** *függvénynek nevezzük, ha legalább az alábbi két feltételt teljesíti:*
(1) ***összenyomás****: a h függvény egy tetszőleges bináris hosszúságú x inputot egy rögzitett méretű $h(x)$ outputba képezi le.*
(2) ***könnyű kiszámítani****: adott h és x input esetén $h(x)$ kiszámítása legfeljebb polinomiális algoritmussal lehetséges.*

Egy hash függvény rendelkezhet még további fontos tulajdonságokkal.

8. Definíció. ***Inverzkép rezisztencia****: előre megadott y outputhoz keresztülvihetetlen egy x input megadása úgy, hogy $h(x) = y$ teljesüljön.*

9. Definíció. ***Második inverzkép rezisztencia****: előre megadott x_1 inputhoz keresztülvihetetlen egy $x_2(\neq x_1)$ második input megadása úgy, hogy $h(x_1) = h(x_2)$ teljesüljön.*

10. Definíció. *Egy h hash függvény* ***ütközésmentes (collision resistant)****, ha tetszőleges x_1 és x_2 értékekre és elegendően nagy k-ra, ha $x_1 \neq x_2$, akkor*

$$\mathbf{P}\left[h(x_1) = h(x_2)\right] \leq \nu(k),$$

ahol a valószínűséget úgy értelmezzük, hogy az $x_1, x_2 \in \{0,1\}^k$ értékek egymástól függetlenül befutják lehetséges értékeiket.

1. Megjegyzés. *Természetesen az utóbbi három tulajdonság nem csak hash függvényre, hanem tetszőleges függvényre - például egyirányú függvényre is - teljesen azonos módon értelmezhető.*

Nyilvánvalóan a kölcsönösen egyértelmű függvények ütközésmentesek is. Most két további fontos definíció következik:

11. Definíció. *A h hash függvény* ***egyirányú hash*** *függvény, ha inverzkép rezisztens és második inverzkép rezisztens.*

12. Definíció. *A h hash függvény* ***ütközésmentes hash*** *függvény, ha második inverzkép rezisztens és ütközésmentes.*

2. Megjegyzés. *Az utóbbi definícióban nincs szükség a második inverzkép rezisztencia tulajdonságra, mivel az az ütközésmentességből következik. Az ütközésmentesség viszont általában nem garantálja az inverzkép rezisztenciát.*

Az előbbi fogalmakra léteznek alternatív megjelölések is. A második inverzkép rezisztencia helyett használatos a gyenge ütközésmentesség, illetve az ütközésmentesség helyett az erős ütközésmentesség is.

Az egyirányú hash függvényt nevezik még gyenge egyirányú hash függvénynek is, valamint az ütközésmentes hash függvényt nevezik (erős) egyirányú hash függvénynek is (lásd [60] és [27]). Az egyirányú függvény helyett általában használható az inverzkép rezisztens függvény kifejezés, azonban az irodalomban gyakran inkább a második inverzkép rezisztens szókapcsolat jelenik meg helyette. Ez viszont kétértelműségre vezet, mivel a második inverzkép rezisztencia nem garantálja az inverzkép rezisztenciát, és ez megfordítva is így van (lásd [47]).

A fenti függvényeknél a gyakorlati alkalmazások és a biztonság szempontjából fontos még az input és output kapcsolatát leíró tulajdonság, amit *lavina hatásnak* nevezünk.

A *lavina hatás* fogalmát Feistel, Notz és Smith [23] említik először a DES algoritmus alapjául szolgáló módszer (Feistel titkosítás) ismertetésében. Később Kam és Davida [36] ezt a tulajdonságot *teljességnek* nevezte és a következőképpen definiálta: egy függvény *teljes*, ha minden egyes output bit függ az összes input bit értékétől.

Webster és Tavares [69] vezetett be egy *erősebb lavina hatás* fogalmat, amely igen szigorú:

13. Definíció. *(Strict avalanche criterion) Az f függvény* ***erős lavina hatással*** *rendelkezik, ha egy input bit megváltoztatása az összes output bit változását okozza $\frac{1}{2}$ valószínűséggel.*

Ez lényegében azt jelenti, hogy egy input bit változása esetén az output bitek átlagosan fele megváltozik. A bitek megváltozását a kódoláselméletből ismert *Hamming-távolság* segítségével lehet mérni. Ennek definíciója a következő:

14. Definíció. *(Hamming-távolság) Legyen az $x, y \in \mathbb{Z}$ számok bináris reprezentációja $x = (\xi_1, ...\xi_n)$ illetve $y = (\eta_1, ...\eta_n)$, ahol $\xi_i, \eta_i \in \{0, 1\}$. Ekkor a két szám Hamming-távolsága:*

$$\varrho(x, y) = \sum_{i=1}^{n} \xi_i \oplus \eta_i,$$

ahol $\oplus$ jelenti a bitenként vett kizáró vagy műveletet.

Ismert egyirányú függvények

A legfontosabb egyirányú függvény jelöltek az alábbiak:

1. *A faktorizáláson alapuló*

   Az $f : (x, y) \mapsto xy$; $x, y \in \mathbb{Z}$ függvény sejtés szerint egyirányú.

2. *A diszkrét logaritmuson alapuló*

   Az $f : (p, g, x) \mapsto (p, g, (g^x \bmod p))$ függvény sejthetően egyirányú, ahol p prím, és $\mathbb{Z}_p^* = (\{x < p \mid (x, p) = 1\}, \mathrm{mod}p)$ csoport ciklikus. Így $\mathbb{Z}_p^* = \{g^i \bmod p \mid 1 \leq i \leq p-1\}$ a $g \in \mathbb{Z}_p^*$ primitív gyökre.

3. *A részhalmazösszeg problémán alapuló*

   Az $f : (\underline{a}, \underline{s}) \mapsto (\underline{a}, \sum_{i=1}^{n} s_i a_i)$ függvény sejthetően egyirányú, ahol $a_i \in \{0,1\}^n$, $\underline{a} = (a_1, ..., a_n)$ és $s_i \in \{0, 1\}$, $\underline{s} = (s_1, ..., s_n)$.

4. *A DES algoritmuson alapuló*

   Az $f_K : M \mapsto DES_K(M)$ függvény sejtés szerint egyirányú. Itt a közismert DES algoritmust 64 bites rögzített M üzenettel és 56 bites K kulcs alkalmazásával használjuk, így az f_K függvény outputja szintén 64 bit hosszú lesz.

5. *Az RSA algoritmuson alapuló*

   Az $f : x \mapsto (x^e \bmod n)$ sejtés szerint egyirányú csapóajtós függvény. Itt p, q prímek, $n = pq$ és e relatív prím $\varphi(n)$-hez. A csapóajtó információ a d érték, amelyre $de \equiv 1 \pmod{\varphi(n)}$.

6. *A diszkrét elliptikus logaritmuson alapuló*

   $f : (x, B) \to (xB = P, B)$, ahol $x \in \mathbb{Z}$, B és P az elliptikus görbe pontjai. Az xB művelet pedig a görbén szokásos $+$ művelet ismétlésével határozható meg. Az így megadott f függvény sejtés szerint egyirányú.

Ez a felsorolás természetesen nem teljes, csak a legismertebb, hosszú idő óta sikeresen használt konstrukciókat tartalmazza.

Fontos még megemlíteni, hogy a DES (Data Encryption Standard) algoritmust a 2000. év óta az AES (Advanced Encryption Standard) váltotta fel, amelynek eredeti neve Rijndael. A NIST (National Institute of Standards

and Technology, USA) pályázatán győztes algoritmust Joan Daemen és Vincent Rijmen fejlesztették (lásd [19]). Ez az algoritmus egy olyan szimmetrikus blokk titkosító, amely a $GF(2^8)$ véges testben végzett műveleteken alapul, a jelenleg ismert feltörési módszereknek ellenáll.

Az itt felsorolt egyirányú függvény jelöltekről jelenleg csak további szükséges feltétel teljesítése esetén bizonyítható, hogy kielégítik az egyirányú függvény fentebb megfogalmazott definícióját.

Például a 2. pontban említett diszkrét logaritmuson alapuló egyirányú függvény jelölt esetében szükséges az erős (SDLA) vagy gyenge (WDLA) diszkrét logaritmus feltétel az egyirányú függvény tulajdonságainak bizonyításához. Ezek a feltételek az alábbiakat jelentik.

SDLA: Minden Q polinomhoz és $\mathcal{A}$ PPT algoritmushoz létezik k_0 egész és minden $k > k_0$ esetén fennáll, hogy

$$P[\mathcal{A}(p, g, y) = x, \text{ ahol } y \equiv g^x \bmod p \text{ és } 1 \leq x \leq p-1] < \frac{1}{Q(k)}.$$

WDLA: Van olyan Q polinom, hogy minden $\mathcal{A}$ PPT algoritmushoz létezik k_0 egész és minden $k > k_0$ esetén fennáll, hogy

$$P[\mathcal{A}(p, g, y) = x, \text{ ahol } y \equiv g^x \bmod p \text{ és } 1 \leq x \leq p-1] < 1 - \frac{1}{Q(k)}.$$

Mindkét feltételben a valószínűséget úgy értelmezzük, hogy az $x \in \mathbb{Z}_p^*$, a g generátor és a p prím ($\log p \leq k$) befutják lehetséges értékeiket és az $\mathcal{A}$ algoritmus is az érmefeldobásait. WDLA esetén az bizonyítható, hogy a függvény gyenge egyirányú függvény, lásd [27].

Típusok és alkalmazásaik

Az egyirányú függvények a kriptográfia legfontosabb építőkövei. Szinte minden alkalmazásban szerepük van. A kulcsot és csapóajtót is tartalmazó egyirányú függvények valósítják meg a szimmetrikus és a nyilvános kulcsú titkosító algoritmusokat.

Igen nagy a jelentőségük a hash függvényeknek is. A kriptográfiában elsősorban az egyirányú és ütközésmentes hash függvényeket alkalmazzák. A kulcsot is tartalmazó hash függvények legfontosabb alkalmazása az üzenet hitelesítés (MAC). A kulcs nélküli hash függvényeket a kriptográfiai protokollok alkalmazzák, ahol legtöbbször manipuláció detektálásra (MDC) használják

őket. Ezek legfontosabb alkalmazása a felhsználó azonosítási protokoll jelszó, illetve jelmondat ellenőrző algoritmusaiban van.

Nagyon sok kriptográfiai protokoll döntő fontosságú alkotóeleme álvéletlen számok generálása. Ebben is jelentős szerepük van a különféle egyirányú függvényeknek.

Napjaink egyik legfontosabb kriptográfiai tevékenysége a digitális aláírás. Természetesen ebben is több ponton használnak valamilyen egyirányú függvényt.

A gyakorlatban használt legismertebb hash függvények az MD5, RIPE-MD és SHA algoritmusok, amelyek leírása szabványokban is szerepel. Ezek működése nagyrészt blokk titkosító algoritmusokon alapul.

A hash függvényekről további ismeretek találhatók még például [47] 9. fejezetében és [60] 18. fejezetében.

A fentiekből látható, hogy az egyirányú függvény mennyire meghatározó jelentőségű a kriptográfia legváltozatosabb területein. Így még különösebbnek tekinthető, hogy ennek a meghatározó elemnek a létét még nem sikerült matematikai alapossággal bizonyítani. Ez a tény méginkább ráirányítja a figyelmet a kriptográfiai alkalmazások óvatos használatára, hiszen ezek megfelelő működésének garanciája alapvetően a gyakorlati teszteken alapul.

Azt is lényeges azonban megjegyezni, hogy a tudomány - főként a matematika és az informatika - már igen sokat tett a kriptográfiai alkalmazások biztonságának megerősítéséért. A gyakorlati tapasztalatok és a tudomány új eredményei azt mutatják, hogy a kriptográfiai alkalmazásokat használó nagyon elterjedt eszközök - például a bankkártyák - nyugodtan használhatók, de az óvatosságot is mindig szem előtt kell tartani.

Feltörési lehetőségek

Ha adott egy M üzenet, a h függvény, valamint a $h(M)$ hash érték, akkor a h hash függvény feltörése azt jelenti, hogy a betolakodó keres egy M' üzenetet, amire $h(M) = h(M')$. Látható, hogy ez csak akkor lehetséges, ha a h függvénynek vannak ütközései.

A kriptográfiai irodalomban nagyon sok támadási lehetőséget írnak le a különféle hash függvények ellen. Mi itt csak a legismertebb támadási módot mutatjuk be, amely független a hash függvényt megvalósító algoritmustól.

Az egyik legjelentősebb ilyen támadási lehetőség a *születésnap támadás*, amelyet G. Yuval [71] dolgozott ki. Az algoritmus megtalálható a [47] könyvben is. A támadás ötletét az alábbi probléma kriptográfiai alkalmazása adta.

Legyen m és n nem negatív egész számok úgy, hogy $n \leq m$. Ekkor a második tipusú Stirling-számokat az alábbi módon jelöljük:

$$\left\{ {m \atop n} \right\} = \frac{1}{n!} \sum_{k=0}^{n} (-1)^{n-k} \binom{n}{k} k^m,$$

ahol $\left\{ {0 \atop 0} \right\} = 1$.

A $\left\{ {m \atop n} \right\}$ szimbólum értékei azt számlálják, hogy hány féleképpen lehet egy m elemű halmazt n nem üres részhalmazra szétvágni.

Használni fogjuk még az

$$m^{(n)} = m(m-1)(m-2)\cdots(m-n+1)$$

jelölést.

Tekinsük a következő problémát (*klasszikus birtoklási probléma*). Egy urnában m darab golyó van, amelyeket 1-től m-ig megszámozunk. Az urnából kiveszünk egyessével n darab golyót úgy, hogy a sorszámukat feljegyezzük és visszatesszük őket. Annak a valószínűsége, hogy pontosan t darab különböző golyót húzunk

$$P_1(m,n,t) = \left\{ {m \atop n} \right\} \frac{m^{(t)}}{m^n},$$

ahol $1 \leq t \leq n$.

A *születésnap probléma* ennek egy speciális esete: Egy urnában m darab golyó van, amelyeket 1-től m-ig megszámozunk. Az urnából kiveszünk egyessével n darab golyót úgy, hogy a sorszámukat feljegyezzük és visszatesszük őket. Annak a valószínűsége, hogy egy golyót legalább kétszer kihúzunk

$$P_2(m,n) = 1 - P_1(m,n,n) = 1 - \left\{ {m \atop n} \right\} \frac{m^{(n)}}{m^n},$$

ahol $1 \leq n \leq m$.

A *születésnap paradoxon* pedig a fentiek alapján úgy szól, hogy 183 embernek kell együtt lennie ahhoz, hogy valamelyiknek szignifikáns valószínűséggel megegyezzen a születésnapja egy megadott dátummal (hónap és nap), de elegendő mindössze 23 ember jelenléte ahhoz, hogy közülük kettőnek $P_2(365,23) = 0.507$ valószínűséggel ugyanazon napra essen a születésnapja. Ráadásul még a $P_2(365,n)$ érték n növekedésével igen gyorsan növekszik, például $P_2(365,30) = 0.706$. Ezek a meglepő adatok jól kihasználhatók egy hatékony támadás megtervezésére.

Egy m bináris hosszúságú üzenetet leképező hash függvény feltöréséhez a mindig rendelkezésre álló kimerítő próbálgatás (az összes lehetséges érték

végigpróbálása) $O(2^m)$ lépésben végezhető el. Ugyanez az érték a párokat kereső születésnap támadás esetén $O(2^{m/2})$, ami már viszonylag biztatónak tűnik.

Ez a támadási módszer nem használható, ha a h hash függvény kölcsönösen egyértelmű. Egyébként pedig megad egy x_1, x_2 üzenetpárt, amire $h(x_1) = h(x_2)$, ahol az x_1 legális, az x_2 pedig illegális üzenet. Ez tulajdonképpen azt jelenti, hogy a h függvény ütközéseit keressük meg, ami a fentiek alapján szignifikáns valószínűséggel $O(\sqrt{m})$ lépés után már jelentkezhet.

Ha az üzenetek hossza 64 bit és a számítógépünk 1 millió próbát tud elvégezni másodpercenként, akkor a primitív kimerítő próbálgatással 2^{64} esetet kellene végignézni, amihez hozzávetőleg 580 ezer évre lenne szükség. Ezzel szemben a születésnap támadás alkalmazásával mindössze 72 percet igényelne a feltörés, mivel a lehetőségek száma most csak 2^{32}. Ez mutatja a támadás erősségét, de egyben azt is, hogy például a gyakorlatban alkalmazott MD5 hash függvény feltörése ezzel a módszerrel keresztülvihetetlen, mert egy ütközés megtalálásához $O(2^{64})$ művelet szükséges(lásd [47]).

További specifikus támadási lehetőségek ismertek még, amelyek főként a különféle blokk titkosító algoritmusokat és különálló összenyomó függvényt alkalmazó hash függvényeket veszik célba. Ezek leírása megtalálható a [47] könyv 9.7. fejezetében.

Normaforma egyenlet alkalmazásán alapuló egyirányú függvény

Előzmények

Egyirányú függvények készítésével sokan és sokféleképpen próbálkoznak. Lényegében azonban három fő irányvonal különböztethető meg. Az egyik főként matematikai módszereket alkalmazva, egzakt definiciókban és tételekben adja meg a megfelelő fogalmak és az elért eredmények leírását. A másik inkább az informatika oldaláról közelíti a problémát, fogalmai és eredményei kevésbé egzaktak. Az eredményeket főként kisérletekkel és tesztekkel bizonyítja, elsősorban a gyakorlati alkalmazásokra koncentrál. Léte törvényszerű, mivel matetematikai bizonyítás nem ismert az egyirányú függvény létezésére, de mégis széles körben használják. A harmadik irányvonal pedig az előbbi kettőt ötvözi úgy, hogy bizonyos esetekben az egyik, más esetekben pedig a másik szokásos módszereit alkalmazza.

Az első kutatási irány szép példája az RSA algoritmus, amely Euler klaszszikus tételét alkalmazza lenyűgözően egyszerű matematikai apparátussal. A szintén klasszikus DES algoritmus S-dobozai és bit manipulációi inkább az informatikai megközelítést példázzák. A Rijndael algoritmus pedig a harmadik kutatási irányvonal eredménye.

Az egyirányú függvények kutatásának igen kiterjedt irodalma van, így csak néhány fontosabbat emelünk ki.

Egyirányú függvények konstrukciójáról szólnak például a [66], [48], [16] és [34] cikkek.

J. Black, S. Halevi, H. Krawczyk, T. Krovetz és P. Rogaway [11] alkotta meg a UMAC algoritmust, amely jelenleg valószínűleg a leggyorsabban

működő üzenet hitelesitési algoritmus.

Szintén üzenet hitelesítésre használható D.L. Whiting és M.J. Sabin [70] MPH (Montgomery Prime Hashing) algiritmusa, amely a Montgomery-maradék képzést alkalmazza titkos modulussal.

O. Goldreich, L. Levin és N. Nisan az RSA, illetve a diszkrét logaritmus probléma invertálási nehézségén alapuló kölcsönösen egyértelmű egyirányú függvény konstrukcióját mutatja be [26] cikkében.

Kvadratikus polinomokból álló egyenletrendszereket használ J. Patarin [52] egy secret public key kriptorendszer konstruálásához. A rendszer biztonsága azon a sejtésen alapszik, hogy ilyen egyenletrendszerek megoldhatósága nem dönthető el algoritmussal.

Algebrai számelméleti eredményeket használ a J. Buchmann és S. Paulus által a [14] cikkben leírt egyirányú függvény, amelynek invertálási nehézsége egy rács legrövidebb vektorának megkeresésén alapul. Az invertálási probléma egy számtest ideálaritmetikai számításainak nehézségéhez vezet.

A következő részben az algebrai számelméletből ismert speciális szétesö forma egyenlet, a *normaforma egyenlet* segítségével készítünk egyirányú függvényt. Egy lineáris forma normája egy többváltozós polinom, amelyből egy függvényt képezünk. Meg fogjuk mutatni, hogy az így kapott függvény helyettesítési értékének kiszámítása könnyű, az invertálása pedig nehéz feladat. A szétesö forma egyenleteknek és széles körű alkalmazásaiknak igen jelentős irodalma van. Néhány ezek közül: [12], [30], [31], [64] [22], [5], [59], [24] és [32].

Az $\mathcal{N}$, $\mathcal{N}_P$ és $\mathcal{N}_{P,s}$ függvények konstrukciója

Egy kriptográfiai szempontból megfelelő egyirányú függvény konstrukciójához az ötletet az általános normaforma egyenletek megoldásának nehézsége adta. A normaforma egyenlet bal oldala egy többváltozós polinom. Ennek helyettesítési értékét kiszámítani általában nehéz feladat.

Tekintsük a $P(\underline{X})$ k változós polinomot, amelynek fokszáma n. Az ilyen polinomot a

$$P(\underline{X}) = \sum_{j=1}^{m} a_j \underline{X}^{\underline{e}_j},$$

formában írhatjuk fel, ahol $\underline{e} = (e_1, ..., e_k) \in \mathbb{Z}^k$, $0 \leq e_1 + ... + e_k \leq n$ és $\underline{X}^{\underline{e}} := X_1^{e_1} X_2^{e_2} \cdots X_k^{e_k}$.

Jelölje $P(\underline{x})$ a $P(\underline{X})$ polinom $\underline{x} = (x_1, ..., x_k)$ helyen vett helyettesítési értékét. A P tagjainak száma legfeljebb $\binom{n+1+k-1}{k} = \binom{n+k}{k} \approx \frac{2^{n+k}-2}{n+k-1} < 2^{n+k}$. Naiv számítás segítségével egy tag helyettesítési értékének kiszámításához $O(n^2 \log^2 \mathbb{X})$ bináris művelet szükséges, ahol $\mathbb{X} = \max\{|x_i|\}$ és a O-ban szereplő konstans csak $\max\{|a_j|\}$ értéktől függ. Így a P polinom helyettesítési értéke kiszámításának bonyolultsága $O(2^{n+k} n^2 \log^2 \mathbb{X})$, amely $n+k$-ban exponenciális.

Természetesen létezik sokkal hatékonyabb szorzási algoritmus is. A legjobb bonyolultsága $O(n \log n \log\log n)$, amelyet Schönhage és Strassen (lásd [62]) készített. Azonban ezt csak akkor érdemes használni, ha a szorzótényezők legalább ezer jegyűek. Kisebb számok esetén jól használható a Karatsuba-Ofman algoritmus (lásd [37]).

Ha az $\underline{x} = (x_1, ..., x_k)$ vektor komponensei racionális egészek, akkor legeredményesebben az intelligens hatványozási algoritmust (lásd [18] 1.2.1 Algoritmus) alkalmazhatjuk, amelynek bonyolultsága $O(\log n \log^2 \mathbb{X})$. Ebben az esetben a helyettesítési érték kiszámításának bonyolultsága $O(2^{n+k+1} \log n \log^2 \mathbb{X})$.

Látható tehát, hogy a $P(\underline{x})$ érték kiszámítása általános esetben valóban nehéz feladat.

Vannak azonban speciális többváltozós polinomok, amelyek helyettesítési értékének meghatározása sokkal gyorsabb.

Tekintsük a

$$P(\underline{X}) = \begin{vmatrix} L_{1,1}(\underline{X}) & \cdots & L_{1,n}(\underline{X}) \\ \cdots & & \cdots \\ L_{n,1}(\underline{X}) & \cdots & L_{n,n}(\underline{X}) \end{vmatrix}$$

polinomot, ahol $L_{i,j}(\underline{X}) = a_{ij1}X_1 + ... + a_{ijk}X_k$, $a_{ijk} \in \mathbb{Z}$; $1 \leq i, j \leq n$, $1 \leq k \leq n$ lineáris formák. A determináns kifejtése után itt is egy k változós, n-ed fokú homogén polinomot kapunk. A helyettesítési érték kiszámítása azonban polinomiális, mivel az $L_{i,j}(\underline{X})$ lineáris formák helyettesítési értékének meghatározása után egy determinánst kell kiszámítani, ami polinomiális bonyolultságú.

Ha egy algebrai egész együtthatós lineáris forma normáját akarjuk kiszámítani, akkor ezt a fentihez hasonló determináns meghatározásával is megtehetjük. Az ilyen módon megadott polinom tulajdonképpen egy normaforma. A normaformák fontos szerepet játszanak a diofantikus egyenletek elméletében (lásd [12] és [57]).

A kriptográfiailag megfelelő egyirányú függvény elkészítéséhez három konstrukciót fogunk bemutatni. Ezek az $\mathcal{N}$, $\mathcal{N}_P$ és $\mathcal{N}_{P,s}$ leképezések, melyek-

nek sorrendisége mutatni fogja a konstrukció fejlődését. A gyakorlatban való használatra az $\mathcal{N}_{P,s}$ függvényt ajánljuk, amely az $\mathcal{N}_P$ függvény moduláris aritmetikát használó változata.

Legyen θ egy n-ed fokú algebrai egész, és jelölje $T(X) \in \mathbb{Z}[X]$ a $\mathbb{Q}$ fölötti minimál polinomját. Legyen $K := \mathbb{Q}(\theta)$, és jelölje $\sigma_i : K \to \mathbb{C}$ a K számtest különböző beágyazásait a koplex számtestbe. Továbbá, valamely $\alpha \in K$ esetén jelölje $\alpha^{(i)} := \sigma_i(\alpha)$ $(i = 1, \ldots, n)$ az α konjugáltjait.

Legyenek $\alpha_1, \alpha_2, ..., \alpha_u \in K$ $\mathbb{Q}$-linearisan független algebrai egészek. Tekintsük az

$$L(\underline{X}) = \alpha_1 X_1 + ... + \alpha_u X_u \tag{1}$$

lineáris formát, ahol $u \leq n$ és legyen

$$L^{(i)}(\underline{X}) = \alpha_1^{(i)} X_1 + ... + \alpha_u^{(i)} X_u. \tag{2}$$

A

$$Norm_{K/\mathbb{Q}}(L(\underline{X})) = \prod_{i=1}^{n} L^{(i)}(\underline{X})$$

polinomot normaformának nevezzük. Könnyen látható, hogy $Norm_{K/\mathbb{Q}}(L(\underline{X}))$ egy n-ed fokú, egész együtthatós homogén polinom.

Tekintsük most azt a speciális esetet, amikor $\alpha_1 = 1, \alpha_2 = \theta, ..., \alpha_u = \theta^{u-1}$. Feltételezzük még, hogy $1, \theta, ..., \theta^{n-1}$ egy hatvány egész bázisa $\mathbb{Z}_K$-nak, ahol $\mathbb{Z}_K$ jelöli a K számtest egészeinek gyűrűjét. Így minden normaforma racionális együtthatós lineáris transzformációval konvertálható az alábbi speciális alakba:

$$\mathcal{N}(\underline{X}) = Norm_{\mathbb{K}/\mathbb{Q}}(X_1 + \theta X_2 + \cdots + \theta^{u-1} X_u). \tag{3}$$

Így az általánosság megsértése nélkül használhatjuk a (3) alakú normaformát.

A fentiek alapján elkészítjük leképezésünk első változatát.

1. Konstrukció. *Definiáljuk az $\mathcal{N} : \mathbb{Z}^u \to \mathbb{Z}$ leképezést az alábbi módon:*

$$\mathcal{N} : (x_1, \ldots, x_u) \mapsto Norm_{\mathbb{K}/\mathbb{Q}}(x_1 + \theta x_2 + \cdots + \theta^{u-1} x_u). \tag{4}$$

Most nézzük meg leképezésünk második változatát, amely egyszerűbb felépítésű.

Legyen $P(X) \in \mathbb{Z}[X]$ rögzített n-ed fokú ($n \geq 3$) főpolinom, amelynek nincsenek többszörös gyökei. Jelölje $\alpha_1, \ldots, \alpha_n$ a P polinom gyökeit és legyen

$$L^{(i)}(\underline{X}) := \sum_{j=1}^{m} \alpha_i^{j-1} X_j, \quad \text{ahol} \;\; i = 1, \ldots, n \text{ és } m \leq n.$$

Legyen definiálva a P polinomhoz tartozó normaforma az alábbi módon:

$$\mathcal{N}_P(\underline{X}) := \prod_{i=1}^{n} L^{(i)}(\underline{X}).$$

Valójában $\mathcal{N}_P(\underline{X})$ a normaforma egy általánosítása, egy speciális szóteső forma.

2. Konstrukció. *Definiáljuk az $\mathcal{N}_P : \mathbb{Z}^m \to \mathbb{Z}$ leképezést a következő módon:*

$$\mathcal{N}_P : (x_1, \ldots, x_m) \mapsto \mathcal{N}_P(x_1, \ldots, x_m). \tag{5}$$

Mivel $\mathcal{N}_P(\underline{X})$ egy egész együtthatós homogén polinom, ezért az $\mathcal{N}_P(\underline{x})$ függvényérték minden $\underline{x} \in \mathbb{Z}^m$ esetén racionális egész lesz.

Most nézzük meg leképezésünk harmadik megvalósítását, amely az előző moduláris aritmetikát alkalmazó változata.

Kriptográfiai szempontból jobbnak tűnik véges testet alkalmazni a leképezés megvalósításához. Legyen valamely $s \in \mathbb{Z}$ esetén $\mathbb{Z}_s := \mathbb{Z}/s\mathbb{Z}$.

3. Konstrukció. *Legyen s egy racionális egész és definiáljuk az $\mathcal{N}_{P,s} : \mathbb{Z}_s^m \to \mathbb{Z}_s$ leképezést az alábbi módon:*

$$\mathcal{N}_{P,s} : (x_1, ..., x_m) \mapsto \mathcal{N}_P(x_1, \ldots, x_m) \mod s. \tag{6}$$

3. Megjegyzés. *A 3. konstrukcióban a biztonsági paraméter az s konstans, amelynek nagysága döntően befolyásolja a biztonságot.*

Meg kell még vizsgálnunk, hogy milyen tipusú szám legyen az s konstans. Tegyük fel, hogy s egy prím. Ekkor azonban $\mathcal{N}_{P,s}$ biztosan nem alkalmas egyirányú függvénynek, mivel - alkalmazva az alábbi feltörési módszert - tetszőleges $b \in \mathbb{Z}_s$ racionális egészhez elfogadható időn belül meghatározható az $(x_1, ..., x_m) \in \mathbb{Z}_s^m$ input, amire $\mathcal{N}_{P,s}(x_1, ..., x_m) = b$.

A hatékony feltörési módszer lényege a következő. Kiszámítjuk az $a = \mathcal{N}_{P,s}(1, 0, ..., 0) \mod s, a \neq 0$ értéket, majd választunk egy $c \in \mathbb{Z}_s$ számot úgy, hogy

$$c^n \mathcal{N}_{P,s}(1, 0, ..., 0) \equiv \mathcal{N}_{P,s}(c, 0, ..., 0) \equiv c^n a \equiv b \pmod{s},$$

azaz

$$c^n \equiv \frac{b}{a} \pmod{s}. \tag{7}$$

Tehát, ha s prímszám, akkor a (7) egyenlet elfogadható időn belül megoldható a Berlekamp faktorizáló algoritmus (lásd [18] 3.4.11 Algoritmus) alkalmazásával.

Válasszuk tehát az s kostanst két prím szorzatának, azaz $s := pq$, ahol p és q megfelelően nagy prímszámok. Ebben az esetben a fenti feltörési módszer nem működik, mivel a (7) egyenlet megoldása belátható időn belül nem lehetséges.

A függvényértékek kiszámítása könnyű

Három különböző algoritmust mutatunk be a függvényértékek kiszámítására. Az $\mathcal{N}(\underline{x})$ érték kiszámítására fogunk koncentrálni, mivel az ott alkalmazott algoritmusok és a hozzájuk tartozó bonyolultsági értékek könnyen átvihetők a másik két függvényre.

A függvényérték könnyű kiszámíthatósága azt jelenti, hogy van olyan polinomiális bonyolultságú algoritmus, amellyel a függvényérték egyértelműen meghatározható. A bonyolultságot minden esetben az input értékek $\mathbb{X}$ maximuma és az n fokszám függvényében adjuk meg.

A függvényérték kiszámítása a definíció alapján

Az első esetben nem tudunk egész aritmetikával számolni, ezért az $\alpha_j^{(i)}$ algebrai számok $\widetilde{\alpha}_j^{(i)}$ lebegőpontos valós közelítéseit fogjuk használni. Tudjuk, hogy minden $\underline{x} \in \mathbb{Z}^u$ esetében $\mathcal{N}(\underline{x})$ racionális egész, ezért elegendő az $\alpha_j^{(i)}$ értékeket az $\widetilde{\alpha}_j^{(i)}$ értékekkel olyan pontossággal közelíteni, hogy $\left|\mathcal{N}(\underline{x}) - \widetilde{\mathcal{N}}(\underline{x})\right| < \frac{1}{2}$ legyen, ahol $\widetilde{\mathcal{N}}(\underline{x}) := \prod_{i=1}^{n} \left(\sum_{j=1}^{u} \widetilde{\alpha}_j^{(i)} x_j\right)$. Készítsünk egy becslést a számítás hibájának felső korlátjára.

1. Lemma. *A fenti jelöléseket használva, legyen*

$$\begin{aligned}
\overline{|\alpha_j|} &= \max\left\{\left|\alpha_j^{(i)}\right| : 1 \le i \le n\right\}, \\
A &= \sum_{j=1}^{u} \overline{|\alpha_j|} \ge 2, \\
\Delta &= \max\left\{\left|\alpha_j^{(i)} - \widetilde{\alpha_j}^{(i)}\right| : 1 \le i \le n,\ 1 \le j \le u\right\} \ \textit{és} \\
\mathbb{X} &= \max\left\{|x_1|, \ldots, |x_u|\right\}.
\end{aligned}$$

Ekkor a számítás hibájának felső korlátja:

$$\Delta_{\widetilde{\mathcal{N}}(\underline{x})} = \left|\mathcal{N}(\underline{x}) - \widetilde{\mathcal{N}}(\underline{x})\right| \leq nu\Delta(A\mathbb{X})^n$$

feltéve, hogy $\Delta \leq \frac{A}{nu\mathbb{X}}$.

Bizonyítás. Legyen $\beta^{(i)} = \sum_{j=1}^{u} \alpha_j^{(i)} x_j$, $\widetilde{\beta}^{(i)} = \sum_{j=1}^{u} \widetilde{\alpha}_j^{(i)} x_j$ és $\widetilde{\beta}^{(0)} = 1$. Ekkor a becslés az alábbi formájú lesz:

$$\begin{aligned}
\Delta_{\widetilde{\mathcal{N}}(\underline{x})} &= \left|\mathcal{N}(\underline{x}) - \widetilde{\mathcal{N}}(\underline{x})\right| = \left|\prod_{i=1}^{n} \beta^{(i)} - \prod_{i=1}^{n} \widetilde{\beta}^{(i)}\right| \\
&= \left|\sum_{j=0}^{n-1} \left(\prod_{i=0}^{j} \widetilde{\beta}^{(i)} \prod_{i=j+1}^{n} \beta^{(i)} - \prod_{i=1}^{j+1} \widetilde{\beta}^{(i)} \prod_{i=j+2}^{n} \beta^{(i)}\right)\right| \\
&= \left|\sum_{j=0}^{n-1} \left(\beta^{(j+1)} - \widetilde{\beta}^{(j+1)}\right) \prod_{i=0}^{j} \widetilde{\beta}^{(i)} \prod_{i=j+2}^{n} \beta^{(i)}\right|.
\end{aligned}$$

Most felhasználjuk, hogy

$$\begin{aligned}
\left|\beta^{(i)}\right| &\leq \mathbb{X} \sum_{j=1}^{u} \left|\alpha_j\right| = A\mathbb{X}, \\
\left|\widetilde{\beta}^{(i)}\right| &\leq \mathbb{X} \sum_{j=1}^{u} \left(\left|\alpha_j\right| + \Delta\right) = \mathbb{X}\left(A + u\Delta\right), \\
\left|\beta^{(i)} - \widetilde{\beta}^{(i)}\right| &= \left|\sum_{j=1}^{u} \left(\alpha_j^{(i)} - \widetilde{\alpha_j}^{(i)}\right) x_j\right| \leq \mathbb{X}u\Delta
\end{aligned}$$

és

$$\Delta \leq \frac{A}{nu\mathbb{X}}.$$

Ekkor azt kapjuk, hogy

$$\begin{aligned}
\Delta_{\widetilde{\mathcal{N}}(\underline{x})} &\leq u\Delta\mathbb{X}^n \sum_{j=0}^{n-1} (A+u\Delta)^j A^{n-j-1} = u\Delta\mathbb{X}^n A^{n-1} \sum_{j=0}^{n-1} \left(1+\frac{u\Delta}{A}\right)^j \\
&= u\Delta\mathbb{X}^n A^{n-1} \frac{\left(1+\frac{u\Delta}{A}\right)^n - 1}{\frac{u\Delta}{A}} = (A\mathbb{X})^n \left(\left(1+\frac{u\Delta}{A}\right)^n - 1\right) \\
&= (A\mathbb{X})^n \sum_{i=1}^{n} \binom{n}{j} \left(\frac{u\Delta}{A}\right)^j \leq (A\mathbb{X})^n \sum_{i=1}^{n} \frac{(nu\Delta)^j}{A^j j!} \\
&\leq (A\mathbb{X})^n \frac{nu\Delta}{2A} \sum_{i=1}^{infty} \frac{(nu\Delta)^j}{A^j j!} \leq (A\mathbb{X})^n \frac{nu\Delta}{2A} \exp\left(\frac{nu\Delta}{A}\right) \\
&< \frac{enu\Delta}{2A} (A\mathbb{X})^n \leq nu\Delta (A\mathbb{X})^n .
\end{aligned}$$

Ezzel a lemma bizonyított. □

A függvény definíciójából következik, hogy $\mathcal{N}(\underline{x}) \in \mathbb{Z}$ ha $\underline{x} \in \mathbb{Z}^u$, ezért a számítási hiba felső korlátjának teljesíteni kell a $\Delta_{\widetilde{\mathcal{N}}(\underline{x})} < \frac{1}{2}$ egyenlőtlenséget. Ebből következik, hogy a számítás pontosságát a

$$\Delta < \frac{1}{2nu\,(A\mathbb{X})^n}$$

egyenlőtlenség szerint kell választanunk.

Miután meghatároztuk az $\widetilde{\alpha}_i^{(j)}$ értékek szükséges pontosságát, amivel az $\alpha_i^{(j)}$ értékeket kell közelíteniük, foglalkozhatunk az $\mathcal{N}(\underline{x})$ függvényérték kiszámításának bonyolultságával. A következő tételt bizonyítjuk:

1. Tétel. *Az $\mathcal{N}(\underline{x})$ függvényérték meghatározásának bonyolultsága a (3) definíció alapján $O\left(n^6 + n^4 \log^2 \mathbb{X}\right)$, ahol a O konstansa csak A értékétől függ.*

Bizonyítás. Legyen $m := n \log \mathbb{X} + \log(2nuA^n)$ és válasszuk az $\alpha_i^{(j)}$ számok minden $\widetilde{\alpha}_i^{(j)}$ közelítését úgy, hogy teljesüljenek az alábbiak:

- az $\widetilde{\alpha}_i^{(j)}$ számok tört részének valós és képzetes része is legfeljebb $m+1$ bináris jegyet tartalmaz,
- $|\Re(\alpha_i^{(j)}) - \Re(\widetilde{\alpha}_i^{(j)})| < 2^{-m-1}$ és

- $|\Im(\alpha_i^{(j)}) - \Im(\widetilde{\alpha}_i^{(j)})| < 2^{-m-1}$.

Ekkor azt kapjuk, hogy $\Delta < 2^{-m}$. Továbbá könnyen belátható, hogy

$$m \leq n \log \mathbb{X} + C_1 n u.$$

Jelölje $l\,(z)$ a z koplex szám bináris hosszát, amely a valós és képzetes rész bináris hosszának maximumát jelenti. Mivel $u \leq n$ azt kapjuk, hogy

$$l\left(\widetilde{\alpha}_i^{(j)}\right) \leq \log \overline{|\alpha_i|} + n \log \mathbb{X} + C_1 n^2 = n \log \mathbb{X} + C_1 n^2 + C_2.$$

Most meghatározzuk az $L_j(\underline{x}) = \sum_{i=1}^{u} \widetilde{\alpha}_i^{(j)} x_i$ ételek kiszámításának bonyolultságát.

Mindenek előtt megbecsüljük azoknak a bináris műveleteknek a számát, amelyek az $\alpha_i^{(j)}$ számok megfelelő közelítésének meghatározásához szükségesek. A [61] 19.2. tétele szerit a szükséges műveletek száma:

$$O(n^3 \log n + n^4 \log \mathbb{X} + C_1 n^5 + C_2 n^3) = O(n^5 + n^4 \log \mathbb{X}). \tag{8}$$

Most rátérünk a számítás hátra lévő részének bonyolultsági becslésére. Egy szorzás elvégzéséhez legfeljebb

$$C_3(n \log^2 \mathbb{X} + C_1 n^2 \log \mathbb{X} + C_2 \log \mathbb{X})$$

bináris művelet szükséges, ezért u $(\leq n)$ darab ilyen szorzás legfeljebb

$$C_3 n^2 \log^2 \mathbb{X} + C_4 n^3 \log \mathbb{X} + C_5 n \log \mathbb{X}$$

bináris műveletet igényel.

Az $L_j(\underline{x})$ valós szám maximális bináris hossza

$$\begin{aligned} l := \max_j \{l(L_j)\} := \max_j \{l(L_j(\underline{x}))\} &\leq \max_{i,j} \left\{ l\left(\widetilde{\alpha}_i^{(j)}\right) \right\} + \log \mathbb{X} + n \\ &\leq (n+1) \log \mathbb{X} + (C_1 + 1) n^2 + C_2. \end{aligned}$$

Most már meg tudjuk határozni a $\prod_{j=1}^{n} L_j(\underline{x})$ kiszámításának bonyolultságát. A szükséges bináris műveletek száma nyilvánvalóan az

$$l(L_1)l(L_2), [l(L_1) + l(L_2)]l(L_3), ..., [l(L_1) + ... + l(L_{n-1})]l(L_n)$$

sorozat elemeinek összege szorozva egy konstanssal. Ez becsülhető az $l^2, 2l^2, ..., (n-1)l^2$ sorozat elemeinek összegével, amelyet még egy konstanssal kell szorozni. Így a teljes szorzat bonyolultsága:

$$\begin{aligned} C_6 \frac{n(n-1)}{2} l^2 =& C_6 \frac{n(n-1)}{2} \left((n+1) \log \mathbb{X} + C_1 n^2 + C_2 \right)^2 \\ \approx& C_7 n^6 + C_8 n^4 \log^2 \mathbb{X}. \end{aligned} \tag{9}$$

Továbbá n példány L_j kiszámításához

$$C_3 n^3 \log^2 \mathbb{X} + C_4 n^4 \log \mathbb{X} + C_2 n^2 \log \mathbb{X} \tag{10}$$

bináris művelet szükséges.

Most már (8), (9) és (10) alapján látható, hogy a teljes számítás bonyolultsága:

$$O\left(n^6 + n^4 \log^2 \mathbb{X}\right).$$

Ezzel a tétel bizonyított. □

A függvényérték meghatározása mátrix reprezentációval

Ha $\Omega = \{\omega_1, ..., \omega_n\}$ a K algebrai számtest egy $\mathbb{Q}$-bázisa és ha $\alpha \in K$, akkor az α-val való szorzás a K számtestnek, mint $\mathbb{Q}$ feletti vektortérnek egy endomorfizmusa. Az α algebrai számot reprezentálni tudjuk az Ω bázisban az endomorfizmus M_α mátrixával. A mátrix elemei általában egészek. Ez a reprezentáció egyértelmű és az $\alpha \mapsto M_\alpha$ leképezés egy homomorfizmus, amely K-t képezi le a $\mathbb{Q}$ feletti $n \times n$ elemű mátrixok algebrájába. Ha $\Omega := \{1, \theta, \theta^2, \ldots, \theta^{n-1}\}$ a K hatvány egész bázisa és $\alpha = a_1\omega_1 + ... + a_n\omega_n$, ahol $a_i \in \mathbb{Z}$ $(1 \leq i \leq n)$, akkor M_α elemei racionális egészek. Ismert, hogy az α szám normája a hozzá tartozó M_α mátrix determinánsa. Az algebrai számok mátrix reprezentációjáról további részletek találhatók a [18] és [12] könyvekben.

2. Tétel. *Legyen P egy n-ed fokú főpolinom, amelynek nincs többszörös gyöke. Jelölje α a P egy gyökét. Legyen $\underline{x} \in \mathbb{Z}^m$ és legyen $\mathcal{N}_P(\underline{x})$ definiálva* (5) *szerint. Akkor az $\mathcal{N}_P(\underline{x})$ függvényérték egész aritmetikával meghatározható.*

A kostruktív bizonyításban az $L^{(i)}(\underline{X})$ lineáris formák mátrix reprezentációját fogjuk alkalmazni.

Bizonyítás. A bizonyításban konkrét algoritmust fogunk adni az $\mathcal{N}_p(\underline{x})$ függvényérték kiszámítására.

Jelölje az n-ed fokú $P(X)$ polinom gyökeit $\alpha_1, \ldots, \alpha_n$ és legyen

$$L^{(i)}(\underline{X}) := \sum_{j=1}^{m} \alpha_i^{j-1} X_j, \quad \text{ahol} \;\; i = 1, \ldots, n \text{ és } m \leq n.$$

Legyen definiálva a P polinomhoz tartozó normaforma a korábban leírt módon:

$$\mathcal{N}_P(\underline{X}) := \prod_{i=1}^{n} L^{(i)}(\underline{X}).$$

Határozzuk meg az $L^{(i)}(\underline{X})$ lineáris formák mátrix reprezentációját az $\Omega_i := \{1, \alpha_i, \alpha_i^2, \ldots, \alpha_i^{n-1}\}$ bázisokban. Mivel az α_i^k hatványok standard reprezentációja az Ω_i bázisokban meghatározható, azt kapjuk, hogy

$$\alpha_i^k = \sum_{j=0}^{n-1} r_{k,j} \alpha_i^j, \quad (i = 1 \ldots n, \text{ és } k = 0 \ldots 2n-2),$$

ahol az $r_{k,j} \in \mathbb{Z}$ értékei a Newton rekurziós formulával határozhatók meg a $P(X)$ polinom együtthatóiból. Ezt pontosan a 2. Lemmában írjuk le.

Ez azt jelenti, hogy

$$L^{(i)}(\underline{X}) \begin{pmatrix} 1 \\ \alpha_i \\ \vdots \\ \alpha_i^{n-1} \end{pmatrix} = \Lambda(\underline{X}) \cdot \begin{pmatrix} 1 \\ \alpha_i \\ \vdots \\ \alpha_i^{n-1} \end{pmatrix}, \tag{11}$$

ahol $1 \leq i \leq n$, valamint

$$\Lambda(\underline{X}) = \begin{pmatrix} F_{0,0}(\underline{X}) & \ldots & F_{0,n-1}(\underline{X}) \\ \ldots & \ldots & \ldots \\ F_{n-1,0}(\underline{X}) & \ldots & F_{n-1,n-1}(\underline{X}) \end{pmatrix},$$

ahol $F_{k,j}(\underline{X}) := \sum_{j=0}^{m-1} r_{k,j} X_j$; $(0 \leq k \leq n-1)$ kifejezések m változós lineáris polinomok.

A (11) alapján azt kapjuk, hogy

$$A \cdot \begin{pmatrix} L^{(1)}(\underline{X}) \\ L^{(2)}(\underline{X}) \\ \vdots \\ L^{(n)}(\underline{X}) \end{pmatrix} = \Lambda(\underline{X}) \cdot A,$$

ahol

$$A = \begin{pmatrix} 1 & \dots & 1 \\ \dots & \dots & \dots \\ \alpha_1^{n-1} & \dots & \alpha_n^{n-1} \end{pmatrix}.$$

Ebből pedig azt kapjuk, hogy

$$L^{(1)}(\underline{X}) \cdot L^{(2)}(\underline{X}) \cdots L^{(n)}(\underline{X}) \det(A) = \det(\Lambda(\underline{X})) \cdot \det(A).$$

Ez pedig azt mutatja, hogy

$$\mathcal{N}_P(\underline{x}) = L^{(1)}(\underline{x}) \cdot L^{(2)}(\underline{x}) \cdots L^{(n)}(\underline{x}) = \det(\Lambda(\underline{x})).$$

Így tehát a kívánt függvényérték egész aritmetikával kiszámítható és ezzel a tétel bizonyított. □

4. Megjegyzés. *A 2. Tétel igaz a* (3)*-ban definiált* $\mathcal{N}(\underline{x})$ *függvényérték kiszámításának esetére is. Legyen* $T(x) = P(x)$ *és legyen például* $\theta = \alpha_1$*, ekkor a tételben leírt kiszámítási módszer lényegében ugyanúgy használható.*

Az előző tétel bizonyításában felhasználtuk Newton rekurziós formuláját(lásd [18]), amelyet a következő lemmában fogalmazunk meg pontosan.

2. Lemma. *(Newton rekurziós formula) Legyen* $T(X) = \sum_{s=0}^{n} t_s X^s \in \mathbb{Z}[X]$ *egy n-ed fokú főpolinom* $\mathbb{Q}$ *fölött. Jelölje* θ *a* $T(X)$ *egy gyökét. Akkor* $\theta^{n+i} = \sum_{j=0}^{n-1} r_{n+i,j}\theta^j$*,* $i \geq 0$*, ahol* $r_{n,j} = -t_j$*;* $j = 0, \dots, n-1$ *és*

$$r_{n+i+1,j} = \begin{cases} r_{n+i,j-1} - t_j r_{n+i,n-1} & \text{ha } j \geq 1 \\ \\ -t_0 r_{n+i,n-1} & \text{ha } j = 0. \end{cases}$$

Bizonyítás. Legyen $\theta^i = \sum_{j=0}^{n-1} r_{i,j}\theta^j$, $i \geq 0$. Nyilvánvaló, hogy ha $0 \leq i \leq n-1$, akkor $r_{i,j} = 1$, ha $i = j$ és $r_{i,j} = 0$ egyébként. Továbbá $r_{n,j} = -t_j$, $0 \leq j \leq n-1$ és ha $i \geq 0$, akkor

$$\begin{aligned} \theta^{n+i+1} &= \left(\sum_{j=0}^{n-1} r_{n+i,j}\theta^j\right)\theta = \sum_{j=0}^{n-2} r_{n+i,j}\theta^{j+1} + r_{n+i,n-1}\theta^n \\ &= \sum_{j=1}^{n-1} r_{n+i,j-1}\theta^j + \sum_{j=0}^{n-1} -t_j r_{n+i,n-1}\theta^j \\ &= -t_0 r_{n+i,n-1} + \sum_{j=1}^{n-1}(r_{n+i,j-1} - t_j r_{n+i,n-1})\theta^j. \end{aligned}$$

Tehát ha $i \geq 0$, akkor

$$r_{n+i+1,j} = \begin{cases} r_{n+i,j-1} - t_j r_{n+i,n-1} & \text{ha } j \geq 1 \\ -t_0 r_{n+i,n-1} & \text{ha } j = 0. \end{cases}$$

Ezzel a lemma bizonyított. □

Most bemutatjuk azon algoritmus lépéseit, amely csupán egészaritmetikát használva meghatározza az $\mathcal{N}_P(\underline{x})$ függvényértéket.

1. Algoritmus (`Az` $\mathcal{N}_P(\underline{x})$ `függvényérték meghatározása mátrix reprezentációval`). Adott az $x_1, ..., x_m \in \mathbb{Z}$ input és a $P(X)$ polinom $t_0, ..., t_n \in \mathbb{Z}$ együtthatói. Az algoritmus meghatározza az $\mathcal{N}_P(x_1, ..., x_m)$ függvényértéket.

1. [`Az` $r_{i,j}$ `értékek meghatározása Newton rekurziós formulával`] A 2. Lemmát használjuk az $r_{i,j}$ számok meghatározásához. Ez a számítás előre elvégezhető, mivel ehhez csak a $P(X)$ polinom együtthatói szükségesek.

2. [`Az` $F_{k,j}(\underline{x})$ `értékek meghatározása`] A $\Lambda(\underline{X})$ mátrix elemei az $F_{k,j}(\underline{X})$ m változós lineáris polinomok, melyek együtthatói az $r_{k,j}$ számok. Az n^2 darab $F_{k,j}(\underline{X})$ helyettesítési értékét számítjuk ki az $\underline{x} = (x_1, ..., x_m) \in \mathbb{Z}^m$ helyen.

3. [`A` $\det(\Lambda)$ `determináns kiszámítása`] A $\Lambda(\underline{x})$ determináns értékét határozzuk meg, amelynek elemei egészek, így a determináns is egész lesz.

4. [`Befejezés`] Az $\mathcal{N}_P(\underline{x}) = \det(\Lambda(\underline{x}))$ érték az algoritmus végeredménye.

Most vizsgáljuk meg a fenti algoritmus bonyolultságát. Bizonyítjuk a következő tételt.

3. Tétel. *Ha a fenti egész aritmetikát alkalmazó algoritmust (1. Algoritmust) használjuk az $\mathcal{N}_P(\underline{x})$ függvényérték kiszámítására, akkor annak bonyolultsága $O(n^7 + n^6 \log \mathbb{X} + n^5 \log^2 \mathbb{X})$, ahol a O konstansa csak a $P(X)$ polinom együtthatóitól függ.*

Bizonyítás A fent leírt lépések bonyolultságát fogjuk becsülni, ezek összege fogja megadni a teljes bonyolultságot.

Az $r_{i,j}$ számok meghatározásához csak a $P(X) = \sum_{i=0}^{n} t_i X^i$ polinom t_i együtthatói szükségesek. Legyen $\mathbb{X} = \max\{|x_i|\}$ és $t = \max\{|t_i|\}$. Az $\mathbb{X}$ és t bináris hossza ekkor $\log \mathbb{X}$ és $\log t$. Az $r_{i,j}$ számokat tartalmazó $n \times (n-1)$ méretű mátrix elemeinek meghatározásához összesen $n^2 - 2n$ szorzás szükséges, így az első lépés bonyolultsága

$$O(n^2). \tag{12}$$

Az $F_{kj}(\underline{x})$ lineáris polinomok helyettesítési értékének meghatározásához legfeljebb m szorzás szükséges. Mivel az $r_{i,j}$ számok hossza $C_1 n \log t = C_2 n$ és $m \leq n$, az n^2 darab érték meghatározásának bonyolultsága

$$O(n^4 \log \mathbb{X}), \tag{13}$$

amely egyben a második lépés bonyolultsága is. A kapott számok bináris hossza legfeljebb $C_3(C_2 n + \log \mathbb{X}) = C_4 n + C_3 \log \mathbb{X}$.

A $\det(\Lambda(\underline{x}))$ determináns kiszámításához a $\Lambda(\underline{x})$ mátrixot Gauss-eliminációval háromszög alakúra hozzuk, majd vesszük a főátló elemeinek szorzatát. Így azonban $\mathbb{Q}$-ban kell műveleteket végezni. A probléma úgy oldható meg, hogy minden eliminált sort beszorzunk az elemek közös nevezőjével. Az eljárás végén a kapott determinánst elosztjuk az előbbi szorzókkal. Természetesen a végeredmény racionális egész lesz. Jelölje l a $\Lambda(\underline{x})$ mátrix elemei hosszának maximumát. Így a j-edik eliminációs lépésben kapott számok hossza jl lesz. Mivel nagyon valószínű, hogy a kapott nevezők relatív prímek lesznek, a közös nevező meghatározásának bonyolultsága $O(jl(jl-1+1)) = O(j^2 l^2)$. Az eliminációs lépésekhez összesen $O(n^3)$ művelet szükséges, így a mátrix háromszögre való transzformálásához legfeljebb $O(n^5 l^2)$ bináris műveletre van szükség. A főátlóban lévő elemek maximális hossza nl, így ezek összeszorzásának bonyolultsága $O\left(\frac{n^3-n^2}{2} l^2\right)$.

Mivel $l = C_4 n + C_3 \log \mathbb{X}$, így a determináns egész aritmetikával történő kiszámításának bonyolultsága legfeljebb

$$O\left(n^5 + \frac{n^3 - n^2}{2}\right)(C_4 n + C_3 \log \mathbb{X})^2 = O(n^7 + n^6 \log \mathbb{X} + n^5 \log^2 \mathbb{X}). \tag{14}$$

A (12), (13) és (14) bonyolultsági értékek összege $O(n^7 + n^6 \log \mathbb{X} + n^5 \log^2 \mathbb{X})$ és ezzel a tétel bizonyított. □

5. Megjegyzés. *A fenti algoritmus 1. lépése előre elvégezhető, a gyakorlati bonyolultság meghatározásához csak a 2. és 3. lépést kell figyelembe venni, ezen lépések bonyolultsága viszont megegyezik a tételben bizonyítottal.*

6. Megjegyzés. *Tekintettel a 4. Megjegyzésre, a 3. Tételben megadott bonyolultság igaz a* (3)*-ban definiált* $\mathcal{N}(\underline{x})$ *függvényérték kiszámításának bonyolultságára is.*

A függvényérték meghatározása moduláris aritmetikával

Ebben a fejezetben az előzőben leírt algoritmus moduláris verzióját mutatjuk be.

Mielőtt az algoritmus leírásához kezdenénk röviden ismertetjük a moduláris aritmetika alkalmazásának lényegét.

Ebben a részben a *mod* m a szimmetrikus maradék függvényt jelöli, azaz $-\left[\frac{m-1}{2}\right] \leq x \bmod m \leq \left[\frac{m}{2}\right]$, minden $x \in \mathbb{Z}$ esetén, ahol [ ] az egészrészt jelenti.

Legyenek az $m_1, ..., m_v > 0$ számok páronként relatív prímek és $M := m_1 m_2 \cdot \cdots m_v$. Ekkor minden $x \in \mathbb{Z}$ számhoz, amire $-\left[\frac{M-1}{2}\right] \leq x \leq \left[\frac{M}{2}\right]$ teljesül egy v elemű vektort tudunk rendelni a következő módon:

$$\psi(x) := x^{(M)} := (x \bmod m_1, ..., x \bmod m_v).$$

A ψ: $x \leftrightarrow x^{(M)}$ leképezés egy gyűrű homomorfizmus $\mathbb{Z}$-ből $\mathbb{Z}/(m_1) \times ... \times \mathbb{Z}/(m_v) \cong \mathbb{Z}/(m_1 \cdots m_v)$ gyűrűbe. Az $x^{(M)}$ vektort szokás az x egész szám moduláris reprezentációjának nevezni.

Jelölje most $\circ$ az összeadás, kivonás vagy szorzás műveletek egyikét. Ha fennáll

$$-\left[\frac{M-1}{2}\right] \leq x, y, x \circ y \leq \left[\frac{M}{2}\right]$$

feltétel, akkor az $x \circ y$ művelet eredményét az alábbi lépésekben tudjuk moduláris aritmetikával meghatározni:

1. Az $x^{(M)}$ és $y^{(M)}$ meghatározása maradékos osztással.
2. Az $x^{(M)} \circ y^{(M)}$ művelet elvégzése a maradékosztály gyűrűben.
3. Az $x \circ y = \psi^{-1}(x^{(M)} \circ y^{(M)})$ művelet eredményének meghatározása kinai maradék algoritmussal.

Az itt leírt eljárás lényeges előnye, hogy a 2. lépésben viszonylag kicsi egész számokkal lehet a műveleteket elvégezni. Hátránya viszont, hogy előre ismerni kell a $x \circ y$ művelet lehetséges eredményének felső korlátját. További hátrány, hogy ilyen módon nem lehet elvégezni a maradékos osztást, valamint általában nem lehetséges a számok összehasonlítása.

A [53] könyv 4.3 fejezetében található hatékony kinai maradék algoritmus, amely rekurziót használ és csökkenti a bonyolultságot bizonyos paraméterek előre történő kiszámításával.

Most pedig bemutatjuk a (3)-ban definiált $\mathcal{N}(\underline{x})$ függvényértéket kiszámító algoritmus moduláris aritmetikát alkalmazó változatát.

Legyen ismét $\mathbb{X} := \max\{|x_i|\}$, $T(X) := \prod_{i=0}^{n} t_i X^i$ és $t := \max\{|t_i|\}$. Ekkor $|\mathcal{N}(\underline{x})| \leq B := n^{2n} t^{n^2} \mathbb{X}^n$. Választanunk kell $m_i := p_i^{k_i}$ prímhatvány modulusokat, ahol $k_i \in \mathbb{Z}_{\geq 0}$, $1 \leq i \leq v$ és $p_1, ..., p_v$ különböző prímek úgy, hogy $M := p_1^{k_1} p_2^{k_2} \cdots p_v^{k_v} > 2B$. A prímek ilyen választásával tudjuk garantálni a

$$-\left[\frac{M-1}{2}\right] \leq \mathcal{N}(\underline{x}) \leq \left[\frac{M}{2}\right]$$

feltétel teljesülését.

Ezeket a prímhatvány modulusokat azonban úgy kell megadnunk, hogy valamilyen C konstansra a $p_i \leq C$, $(i = 1, \ldots, v)$ és $M := p_1^{k_1} p_2^{k_2} \ldots p_v^{k_v} > 2B$ feltételek teljesüljenek. Az $M > 2B$ feltétel teljesüléséhez elegendő feltételezni, hogy $\prod_{p \leq C} p > 2B$, és így mivel $\prod_{p \leq C} p > \sqrt{C}^{\pi(C)-\pi(\sqrt{C})}$ [2] azt nyerjük, hogy $\sqrt{C}^{\pi(C)-\pi(\sqrt{C})} > 2B$. Így a prímek felső korlátjára azt kapjuk, hogy $C = C'\sqrt{\log(2B)}$.

A következő algoritmusban írjuk le, hogyan határozható meg moduláris aritmetikát használó módszerrel az $\mathcal{N}(\underline{x})$ függvényérték.

2. Algoritmus (`Az` $\mathcal{N}(\underline{x})$ `függvényérték meghatározása moduláris aritmetikával`). Adott az $x_1, ..., x_u \in \mathbb{Z}$ input és a $T(X)$ polinom $t_0, ..., t_n \in \mathbb{Z}$ együtthatói. Az algoritmus meghatározza az $\mathcal{N}(x_1, ..., x_u)$ függvényértéket.

1. [`Az` $r_{i,j}$ `számok meghatározása Newton rekurziós formulával`] Ebben a lépésban még nem használjuk a moduláris aritmetikát, mivel egyrészt az $r_{i,j}$ számok maximális mérete $cn \log t$, másrészt pedig ez a számítás előre elvégezhető, mivel csak a $T(X)$ polinom t_i együtthatói szükségesek hozzá. Egyébként a számítás menete ugyan az, mint az 1. Algoritmus 1. lépésében.

2. [`Az` $F_{kj}(\underline{x})$ `értékek meghatározása`] Itt már alkalmazzuk a moduláris aritmetikát. Ebben a lépésben az $F_{kj}(\underline{x})$ mod $p_1^{k_1}, ...,$ mod $p_v^{k_v}$ értékeket határozzuk meg az $\underline{x} = (x_1, ..., x_u) \in \mathbb{Z}^u$ helyen. Így tulajdonképpen a $\Lambda^{(1)}(\underline{x}), ..., \Lambda^{(v)}(\underline{x})$ mátrixok elemeit számítjuk ki, amik racionális egészek. Itt a $\Lambda^{(i)}(\underline{x}) = \Lambda(\underline{x})$ mod $p_i^{k_i}$ jelölést használtuk.

[2] $\pi(x)$ jelöli a prímszámok számát x-ig

3. [`A` $\det(\Lambda^{(M)}(\underline{x})) = \mathcal{N}^{(M)}(\underline{x})$ `értékek meghatározása`] Kiszámítjuk a $\Lambda^{(1)}(\underline{x}), ..., \Lambda^{(v)}(\underline{x})$ mátrixok determinánsait. Ezek az $\mathcal{N}(\underline{x})$ függvény érték $\left(\mathcal{N}^{(1)}(\underline{x}), \ldots, \mathcal{N}^{(v)}(\underline{x})\right)$ moduláris reprezentációjának elemei.

4. [`Az` $\mathcal{N}(\underline{x})$ `függvényérték meghatározása`] Kiszámítjuk az $\mathcal{N}(\underline{x})$ függvény értéket az $\left(\mathcal{N}^{(1)}(\underline{x}), \ldots, \mathcal{N}^{(v)}(\underline{x})\right)$ moduláris reprezántációból kinai maradék algoritmussal.

5. [`Befejezés`] A kínai maradék algoritmus eredménye lesz az $\mathcal{N}(x_1, ..., x_u)$ függvényérték.

Most vizsgáljuk meg a fenti algoritmus bonyolultságát. Bizonyítjuk az alábbi tételt.

4. Tétel. *Az $\mathcal{N}(\underline{x})$ függvényérték moduláris aritmetikával történő meghatározásának bonyolultsága $O(n^7+n^6 \log \mathbb{X}+n^2 \log^{3/2} \mathbb{X})$, ahol a O konstansai csak a $T(X)$ polinom együtthatóitól függnek.*

Bizonyítás. A fenti négy lépés bonyolultságát fogjuk becsülni, az algoritmus teljes bonyolutságát ezek összege adja. Legyen $t := \max\{|t_i|\}$ és $P := \max\left\{p_i^{k_i}\right\}$.

Mivel az $r_{i,j}$ számok a korábban leírtakkal azonos módon határozhatók meg itt is, az első lépés bonyolultsága

$$O(n^2). \tag{15}$$

Az $F_{kj}(\underline{x})$ lineáris formák helyettesítési értékének kiszámításához először az $x_i \bmod p_1^{k_1}, ..., x_i \bmod p_v^{k_v}$, $i = 1, \ldots, u$ és az $r_{i,j} \bmod p_1^{k_1}, ..., r_{i,j} \bmod p_v^{k_v}$ maradékokat kell kiszámítani, ahol $i = n, \ldots, n-2$ és $j = 0, \ldots, n-1$. Ehhez $O(vn \log P \log \mathbb{X} + vn^3 \log P)$ bináris műveletre van szükség. Innentől minden előforduló szám bináris hossza legfeljebb $\log P$. Mivel $u \leq n$ az n^2 darab érték kiszámításának bonyolultsága $O(vn^3 \log^2 P)$. Így a második lépés teljes bonyolultsága

$$O(vn \log P \log \mathbb{X} + vn^3 \log^2 P). \tag{16}$$

A harmadik lépésben a $\Lambda^{(M)}(\underline{x}) \in \mathbb{Z}^{n \times n}$ mátrixok determinánsát kell meghatározni. Minden ilyen determináns kiszámításának bonyolultsága $O(n^5 \log^2 P)$, így a harmadik lépés bonyolultsága

$$O(vn^5 \log^2 P). \tag{17}$$

A negyedik lépésben az $\mathcal{N}(\underline{x})$ függvény értéket kell meghatározni az $\left(\mathcal{N}^{(1)}(\underline{x}),\ldots,\mathcal{N}^{(v)}(\underline{x})\right)$ mátrix reprezentációból kínai maradék algoritmussal. Most alkalmazni fogjuk a [53] könyv 4.1. algoritmusát. Ennek inputjai lesznek az $\left(\mathcal{N}^{(1)}(\underline{x}),\ldots,\mathcal{N}^{(v)}(\underline{x})\right)$, $p_1^{k_1},\ldots,p_v^{k_v}$, $q_2,...,q_v$ és $s_2,...,s_v$ értékek, ahol

$$q_i=\prod_{j=1}^{i-1}p_j^{k_j},\quad s_i\equiv\prod_{j=1}^{i-1}s_j^{(i)}\bmod p_i^{k_i}\quad(i=2,...,v),$$

ahol

$$s_j^{(i)}p_j^{k_j}+s_i^{(j)}p_i^{k_i}=1.$$

Az $s_j^{(i)}$ és $s_i^{(j)}$ számok meghatározhatók $O(\log^2 P)$ bináris művelettel a kiterjesztett euklideszi algoritmus segítségével (lásd [53] 3.8. Tétel). Az s_i számok meghatározásához $O((v-1)^2\log^2 P)$ bináris művelet szükséges, a q_i számok kiszámításához pedig $O(\frac{1}{2}(v^2-3v+2)\log^2 P)$. Az eddig végzett számítások előre elvégezhetők voltak. A kinai maradék algoritmus hátra lévő része pedig $O(v\log^2 P)$ bináris műveletet igényel. Így a negyedik lépés bonyolultsága

$$O(v^2\log^2 P).\tag{18}$$

A (15), (16), (17) és (18) eredményeket összegezve az algoritmus teljes bonyolultsága

$$O(v^2n^5\log^2 P+vn\log P\log\mathbb{X}).\tag{19}$$

Mivel $v\le\frac{cC}{\log C}$ és $\log P\le\log C$ azt kapjuk, hogy $v\log P\le cC$. Így most a (19) eredményt

$$O(C^2n^5+Cn\log\mathbb{X})\tag{20}$$

formába írhatjuk át. Továbbá, mivel $C=C'\sqrt{\log(2B)}$ és $B=n^{2n}t^{n^2}\mathbb{X}^n$ láthatjuk, hogy

$$C^2\le c_1n^2+c_2n\log\mathbb{X}$$

és

$$C\le c_3n+c_4n^{1/2}\log^{1/2}\mathbb{X}<c_5n\log^{1/2}\mathbb{X}.$$

Ez azt mutatja, hogy az algoritmus teljes bonyolultsága legfeljebb

$$O(n^7+n^6\log\mathbb{X}+n^2\log^{3/2}\mathbb{X}).$$

Ezzel a tétel bizonyított. □

7. Megjegyzés. *Az 1. lépés számításai és a 4. lépésben az s_i és q_i számok meghatározása előre elvégezhetők, de ez lényegében nem változtat a tételben bizonyított bonyolultsági értéken.*

Tekintettel a 4. Megjegyzésre, igaz a következő tétel, amely a 4. Tétel $\mathcal{N}_P(\underline{x})$ kiszámítására vonatkozó változata.

5. Tétel. *Az $\mathcal{N}_P(\underline{x})$ függvényérték moduláris aritmetikával történő meghatározásának bonyolultsága $O(n^7+n^6\log\mathbb{X}+n^2\log^{3/2}\mathbb{X})$, ahol a O konstansa csak a $P(X)$ polinom együtthatóitól függ.*

Most vizsgáljuk meg az $\mathcal{N}_{P,s}(\underline{x})$ kiszámításának bonyolultságát.

Az $\mathcal{N}_{P,s}(\underline{x})$ függvényértéket az $\underline{x}=(x_1,...,x_m)\in\mathbb{Z}^m$ helyen a 2. Tételben leírt mátrix reprezentációt használó módszerrel fogjuk kiszámítani, azzal az eltéréssel, hogy minden műveletet $\bmod s$ végzünk.

3. Algoritmus (`Az` $\mathcal{N}_{P,s}(\underline{x})$ `függvényérték meghatározása`). Adott az $x_1,...,x_m\in\mathbb{Z}$ input és a $P(X)$ polinom $t_0,...,t_n\in\mathbb{Z}$ együtthatói. Az algoritmus meghatározza az $\mathcal{N}_{P,s}(x_1,...,x_m)$ függvényértéket.

1. [`Az` $r_{i,j}$ `értékek meghatározása` $\bmod s$] A 2. Lemmát használjuk az $r_{i,j}$ számok $\bmod s$ meghatározásához. Ez a számítás előre elvégezhető, mivel ehhez csak a $P(X)$ polinom együtthatói szükségesek.

2. [`Az` $F_{k,j}(\underline{x})$ `értékek meghatározása` $\bmod s$] A $\Lambda(\underline{X})$ mátrix elemei az $F_{k,j}(\underline{X})$ m változós lineáris formák, melyek együtthatói az $r_{i+k,j}$ számok. Az n^2 darab $F_{k,j}(\underline{X})$ helyettesítési értékét számítjuk ki $\bmod s$ az $\underline{x}=(x_1,...,x_m)\in\mathbb{Z}^m$ helyen.

3. [`A` $\det(\Lambda)$ `determináns kiszámítása` $\bmod s$] A $\Lambda(\underline{x})$ determináns értékét határozzuk meg $\bmod s$.

4. [`Befejezés`] Az $\mathcal{N}_{P,s}(\underline{x})=(\det(\Lambda(\underline{x}))\bmod s)$ érték az algoritmus végeredménye.

6. Tétel. *Az $\mathcal{N}_{P,s}(\underline{x})$ függvényérték kiszámításának bonyolultsága a fenti algoritmussal $O(n^5\log^2 s)$, ahol a O konstansa abszolút.*

Bizonyítás. A bizonyítást ugyan úgy végezzük, mint a 3. Tételét, azzal az eltéréssel, hogy minden műveletet $\bmod s$ végzünk.

Így az első lépés bonyolultsága

$$O(n^2). \tag{21}$$

Az $x_1,\ldots,x_m$ értékek maradékainak kiszámításához m darab osztás szükséges, a $t_{k,j}$ számok maradékainak meghatározása pedig n^2-n darab

osztást igényel. Így egy $F_{k,j}(\underline{X})$ lineáris polinom helyettesítési értékének meghatározása $\bmod s$ legfeljebb $m + n^2 - n + Cm \log^2 s$ darab szorzással és osztással kiszámítható. Így mivel $m \leq n$, az n^2 darab helyettesítési érték kiszámításának bonyolultsága

$$O(n^4 + n^3 \log^2 s). \tag{22}$$

A kapott helyettesítési értékek bináris hossza a $\bmod s$ számolás miatt legfeljebb $l = \log s$.

A determináns egész aritmetikával történő meghatározásához Gauss-eliminációt alkalmazunk $\bmod s$. A moduló inverzek meghatározásához $O(n^2 l^2)$ bináris műveletre van szükség. Az eliminációs lépések $O(n^3)$ bináris művelettel elvégezhetők. Így a háromszög mátrix kialakításához $O(n^5 l^2)$ bináris művelet szükséges. A főátló elemeinek összeszorzása $O(nl^2)$ bináris művelettel végezhető el.

Mivel $l = \log s$, a determináns meghatározása legfeljebb

$$O(n^5 \log^2 s) \tag{23}$$

bináris műveletet igényel. Összegezve (21), (22) és (23) értékeit, azt kapjuk, hogy a fenti algoritmus bonyolultsága $O(n^5 \log^2 s)$, amivel a tétel bizonyítást nyert. □

8. Megjegyzés. *Az 1. lépés számításai előre elvégezhetők, de ez lényegében nem változtat a tételben bizonyított bonyolultsági értéken.*

Az algoritmusok összehasonlítása

Itt most össze fogjuk hasonlítani az algoritmusok működési sebességét. Ebben a részben is elsősorban az $\mathcal{N}$ függvényre fogunk koncentrálni, amelynél az $\mathcal{N}(\underline{x})$ függvényérték kiszámítására három algoritmust mutattunk be.

Annak az algoritmusnak a bonyolultsága, amely a függvényértéket közvetlenül a definícióból számítja ki $O(n^6 + n^4 \log^2 \mathbb{X})$. Viszont lényeges hátránya, hogy $\frac{1}{2nu(A\mathbb{X})^n}$ pontossággal a valós számok körében kell a számításokat elvégezni.

Az $L(\underline{X})$ lineáris forma mátrix reprezentációját használó algoritmus bonyolultsága $O(n^7 + n^6 \log \mathbb{X} + n^5 \log^2 \mathbb{X})$. Azonban fontos előnye, hogy egész aritmetikát használ.

A moduláris aritmetikát alkalmazó algoritmus bonyolultsága $O(n^7 + n^6 \log \mathbb{X} + n^2 \log^{3/2} \mathbb{X})$, ami a legalacsonyabb érték.

Most vizsgáljuk meg, hogy milyen gyorsan működnek ezek az algoritmusok a gyakorlatban. Elkészítettük a fenti három algoritmus implementációját *MAPLE V Release 4* (lásd [17]) rendszerben. Ezek az *ALG1*, *ALG2* és *ALG3* eljárások. Ezek működési sebességét fogjuk összehasonlítani egymással és az *ALG4* eljárással, amely a *MAPLE* rendszer *Norm()* eljárását alkalmazza. Az algoritmusok forráskódját a dolgozat végén lehet megtalálni.

Mind a négy algoritmus véletlenszerűen generált polinomokkal és adatokkal működik. Az összehasonlító tesztelés mindegyik algoritmust 50-szer futtatja ugyanazokkal a paraméterekkel, majd pedig a futási idők átlagértékét veszi másodpercben. A következő táblázat mutatja a teszt jellemző értékeit:

n =	5	5	5	5	5	5
u =	4	4	4	4	4	4
Et =	10	10	10	10	10	10
Ex =	50	70	90	110	130	150
ALG1	0.2427	0.4892	0.5942	0.6128	1.233	1.244
ALG2	0.0130	0.0115	0.0126	0.0215	0.0181	0.0206
ALG3	0.0697	0.1032	0.1727	0.3099	0.2329	0.2897
ALG4	0.0221	0.0220	0.0262	0.0395	0.0506	0.0542

A táblázatban az n jelöli a minimálpolinom fokszámát, az u a lineáris forma változóinak számát, az Et a minimál polinom együtthatóinak nagyságrendjét és az Ex pedig a helyettesítési értékek nagyságrendjét a decimális jegyek számában megadva.

A táblázat azt mutatja, hogy a leggyorsabb az *ALG2* algoritmus, amely egész aritmetikával számol és alkalmazza a lineáris forma mátrix reprezentációját. Az *ALG1* algoritmus az első tesztben 400 jegy pontos valós számokkal számol, a hatodik tesztben pedig már 900 jegy pontosságra van szükség. Ez indokolja, hogy miért a legrosszabb a futási ideje.

Az input értékek nagyságrendjének növekedésével legjobban az *ALG1* algoritmus sebessége csökken, és legkevésbé az *ALG2* algoritmusé. Az igazi

lassító tényező a fokszám. Ezt mutatja az alábbi táblázat:

n =	4	5	6	7	8	9
u =	4	4	5	6	7	8
Et =	10	10	10	10	10	10
Ex =	100	100	100 100	100	100	100
ALG1	0.3996	0.6247	1.533	4.035	6.751	10.13
ALG2	0.0085	0.0160	0.0527	0.1161	0.2153	0.3811
ALG3	0.0757	0.1570	0.5232	1.137	2.245	4.316
ALG4	0.0245	0.0306	0.0581	0.0847	0.1383	0.2104

Itt a hatodik tesztben az *ALG1* algoritmus már 1500 jegy pontossággal számol, így a futásideje már igen nagyra növekszik. A harmadik tesztig az *ALG2* futási ideje a legjobb. A további tesztekben az *ALG4* lett a leggyorsabb. Ezzel kapcsolatban megjegyezzük, hogy az *ALG4* a *MAPLE* saját *Norm()* eljárását használja, amelyet nem a saját programozási nyelvében, hanem *C* nyelven implementáltak. Ez valamelyest megmagyarázza a futási sebesség különbségeit.

A moduláris aritmetikát használó *ALG3* algoritmus sem elég gyors, mert a *MAPLE* saját nyelvében implementált *chrem()* kínai maradék algoritmust alkalmazza, amelyben nincs lehetőség a 7. Megjegyzésben említett gyorsító előre számolást kihasználni.

A gyakorlati tesztek csak részben igazolták a bonyolultsági értékek által mutatott tendenciákat. Ennek fő oka az, hogy a *MAPLE* rendszerben készült implementációk egyes részei a rendszer saját nyelvében implementáltak, míg más részek a gyorsabb működést garantáló *C* programozási nyelvben. A különbségek okai között megemlítendő még az is, hogy a bonyolultsági értékek konstansai is elfedhetnek lényeges tényezőket, amelyek a futási időket nagy mértékben befolyásolhatják.

A tesztelésben nem szerepeltek az $\mathcal{N}_P$ és $\mathcal{N}_{P,s}$ függvények algoritmusai. Az első esetében ez felesleges lett volna, mivel lényegében ugyanazokat az eljárásokat kaptuk volna, mint az $\mathcal{N}$ függvény esetében, így a teszt eredményei is azonosak lettek volna. Az $\mathcal{N}_{P,s}$ függvény pedig nem ugyanazt a számértéket adja meg, mint az $\mathcal{N}$ függvény, így az összehasonlítás nem lett volna értelmes.

Az ebben a részben leírt eredmények azt mutatják, hogy az $\mathcal{N}(\underline{x})$, $\mathcal{N}_P(\underline{x})$ és $\mathcal{N}_{P,s}(\underline{x})$ függvényértékek kiszámítása - a gyakorlatban is - valóban könnyű.

Az invertálás nehéz

Ebben a részben megvizsgáljuk, hogy az $\mathcal{N}$, $\mathcal{N}_P$ és $\mathcal{N}_{P,s}$ leképezések inverzének kiszámítása mennyire nehéz feladat. Ehhez elegendő az $\mathcal{N}_P(\underline{X}) = b$, illetve $\mathcal{N}_P(\underline{X}) = b \bmod s$ normaforma egyenletek megoldásának lehetőségeit áttekinteni, ahol $0 \neq b \in \mathbb{Z}$ és a megoldásokat az $\underline{x} \in \mathbb{Z}^m$ vektorok között keressük.

Az ilyen általános normaforma egyenletek megoldására jelenleg nem ismert a gyakorlatban is használható algoritmus, sőt még az is nyitott kérdés, hogy a probléma algoritmussal megoldható-e. Mint ismeretes, Jurij Matijaszevics 1970-ben bizonyította, hogy nem létezik olyan algoritmus, amely az általános diofantikus egyenletek megoldhatóságát eldöntené. Ezzel kapcsolatban további részletek találhatók a [45] és [55] könyvekben.

A normaforma egyenletekkel szoros kapcsolatban álló lineáris rekurzív sorozatokról szól Cerlienco, Mignotte és Piras [15] következő sejtése:

1. Sejtés. *Létezik k pozitív egész szám és $\xi^{(1)}, \ldots, \xi^{(k)}$ egészekből álló lineáris rekurzív sorozat a következő tulajdonsággal: Algoritmussal nem dönthető el, hogy vannak-e olyan $n_1, \ldots, n_k \in \mathbb{N}^k$ értékek, amelyekre $\xi_{n_1}^{(1)} + \cdots + \xi_{n_k}^{(k)} = 0$.*

Ezt felhasználva, talán belátható lenne, ha igaz ez a sejtés, akkor a normaforma egyenletek megoldhatósága algoritmikusan nem dönthető el. Pontosabban, kimondható a következő sejtés:

2. Sejtés. *Ha az 1. Sejtés igaz, akkor nem létezik olyan algoritmus, amelyik nem elfajuló normaforma egyenletekről eldönti, hogy megoldhatóak-e.*

Most vizsgáljuk meg a Thue-egyenlet esetét, amely tekinthető egy speciális normaforma egyenletnek. Legyen $F(X, Y) \in \mathbb{Z}[X, Y]$ irreducibilis polinom és $\deg(F) \geq 3$. Ekkor minden nem nulla $b \in \mathbb{Z}$-re az

$$F(X, Y) = b \tag{24}$$

diofantikus egyenletet Thue-egyenletnek nevezzük, ahol a megoldások $(x, y) \in \mathbb{Z}^2$ számpárok. Látható, hogy a Thue-egyenlet lényegében az $\mathcal{N}_P(\underline{X}) = b$ normaforma egyenlet két ismeretlenes specializálása ($m = 2, n \geq 3$).

A. Thue 1909-ben a [67] cikkben bizonyította, hogy (24)-nek csak véges sok megoldása van. 1968-ban A. Baker [4] effektív felső korlátot határozott meg a megoldások abszolút értékének maximumára, ami annyit jelent, hogy a (24) Thue egyenlet megoldásai algoritmussal meghatározhatóak. Huszonegy évvel

később, 1989-ben Tzanakis és de Weger [68] gyakorlatban használható algoritmust adott a (24) egyenlet megoldásainak megkeresésére az LLL algoritmus (lásd [18] 2.6.3. Algoritmus) felhasználásával.

Arra a kérdésre, hogy milyen nehéz a gyakorlatban megkeresni a (24) egyenlet összes megoldását csak bonyolultsági elemzéssel lehet megadni a választ.

N.P. Smart [65] elvégezte ezt az elemzést, amelynek a lényege a következő.

Legyen $n = \deg(F)$ rögzített és vizsgáljuk a megoldási módszerek bonyolultságát $\log b$ és $\log L(F)$ függvényében, ahol $L(F) = \sum_{i=1}^{n} |a_i|$ és $F(x,1) = x^n + a_1 x^{n-1} + ... + a_n$.

A megoldásokat három fő csoportra oszthatjuk: kicsi, közepes és nagy megoldások.

Legyen $K = \mathbb{Q}(\theta)$, ahol $F(\theta, 1) = 0$ és jelölje $\theta^{(1)}, ..., \theta^{(n)}$ a θ primitív elem konjugáltjait. Legyen továbbá

$$\Delta(\theta) = \min \left| \theta^{(i)} - \theta^{(j)} \right| \quad 1 \leq i < j \leq n$$

és

$$Y_0 = \left\{ \begin{array}{ll} \left(\frac{2^n |b|}{\Delta(\theta)^{n-1}} \right)^{\frac{1}{n-2}}, & \text{ha } \Delta(\theta) < 1 \\ \\ \frac{(2^n |b|)^{\frac{1}{n-2}}}{\Delta(\theta)}, & \text{ha } \Delta(\theta) \geq 1 \end{array} \right\}$$

valamint

$$Y_1 = O(R_K^2 + (1 + R_K)(\log |b| + \log L(F))),$$

ahol R_K a K számtest regulátora.

A (24) egyenlet $(x, y) \in \mathbb{Z}^2$ megoldásai kicsinek számítanak, ha $|y| \leq Y_0$. Ekkor a megoldások megadásának szokásos módszere az $y \in [-Y_0, Y_0]$ intervallum egészeinek kipróbálása úgy, hogy a megfelelő x értéket az $F(x, y) - b$ polinom faktorizálásával határozzuk meg, majd pedig teszteljük az (x, y) párt. Ennek a direkt keresésnek a bonyolultsága rögzített $F(x, y)$ és rögzített b esetén $\log |b|$-ben illetve $\log L(F)$-ben exponenciális. Abban a speciális esetben, ha $\Delta(\theta) \geq (\log L(F))^{-c}$ valamely c konstansra, azaz $\Delta(\theta)$ nem túl kicsi, akkor rögzített b esetén a bonyolultság $\log L(F)$-ben polinomiális.

Ha $Y_0 < |y| \leq e^{Y_1}$, akkor az $(x, y) \in \mathbb{Z}^2$ megoldásokat közepesnek szokás nevezni. Ekkor alkalmazható az úgynevezett lánctört redukciós módszer. Ez azon alapul, hogy az x/y értékek tartanak a θ egy $\theta^{(i)}$ valós konjugáltjának lánctört kifejtéséhez, ha y elég nagy. Ilyen lánctört redukciós algoritmus

például a [53] könyv 5.5. algoritmusa. Ennek a módszernek a bonyolultsága rögzített $F(x,y)$ esetén $\log|b|$-ben polinomiális, illetve rögzített b esetén $\log L(F)$-ben exponenciális.

Ha $\log|y| > Y_1$, akkor a (24) egyenlet megoldásai nagynak számítanak. Erre az esetre a Baker által alkalmazott logaritmusos lineáris forma módszert kombinálják a Tzanakis és de Weger által alkalmazott rács redukciós módszerrel, amely az LLL algoritmust használja. A gyakorlat azt mutatja, hogy a redukciós lépést legalább egyszer mindig lehet alkalmazni. Ennek bonyolultsága polinomiális.

A módszer gyakorlati alkalmazása számos problémát rejt magában. Többek között meg kell határozni a K számtest alapegységeit és regulátorát. Ezt J. Buchmann szubexponenciális bonyolultságú algoritmusának általánosításával lehet meghatározni, amelyet [13] cikkében írt le. Feltételezni kell az általánosított Riemann-hipotézis helyességét is.

Mindezt összegezve a (24) egyenlet összes megoldásának megkeresése általában $\log|b|$-ben és $\log L(F)$-ben is exponenciális bonyolultságú.

A fentiek alapján látható, hogy az $\mathcal{N}_P(\underline{x}) = b$, illetve $\mathcal{N}_P(\underline{x}) = b \bmod s$ egyenlet megoldásainak meghatározása elég nehéz probléma. Így az $\mathcal{N}$, $\mathcal{N}_P$ és $\mathcal{N}_{P,s}$ függvények lehetnek egyirányú függvény jelöltek, különösen akkor, ha a változók száma legalább 3.

Az $\mathcal{N}_P$ és $\mathcal{N}_{P,s}$ függvények tulajdonságai

Érdemes megvizsgálni az $\mathcal{N}_{P,s}$ függvény ütközéseinek bekövetkezési valószínűségét, amit a

$$P_{coll} = P[\mathcal{N}_{P,s}(\underline{x}) = \mathcal{N}_{P,s}(\underline{y}) : \underline{x}, \underline{y} \in {\mathbb{Z}_s}^m]$$

kifejezés értéke mutat meg.

Fő eredményünk a következő két tétel.

7. Tétel. *Legyen $P(X) \in \mathbb{Z}[X]$ egy legalább harmadfokú főpolinom, amelynek nincsenek többszörös gyökei. Legyenek p és q olyan prímszámok, hogy $q > p > q/2$ és $s := pq$. Tegyük fel, hogy lnko[3]$(m, \varphi(s)) = 1$.[4] Jelölje $N(P,b,s)$ az $N_P(x_1,\ldots,x_m) \equiv b \pmod{s}$ kongruencia megoldásainak számát.*

[3] $lnko(n,m)$ jelöli az n és m egészek legnagyobb közös osztóját.

[4] $\varphi(x)$, szokás szerint, az Euler féle φ-függvényt jelöli.

- *Ha* $lnko(b,s)=1$ *teljesül, akkor*

$$|N(P,b,s)-s^{m-1}|<c_1(P)s^{m-1-\frac{1}{4}}$$

- *egyébként pedig*

$$N(P,b,s)<c_2(P)s^{m-1}.$$

8. Tétel. *Legyen* $P(X)\in\mathbb{Z}[X]$ *egy legalább harmadfokú főpolinom, amelynek nincsenek többszörös gyökei. Legyenek* p *és* q *prímszámok olyanok, hogy* $q>p>q/2$ *és* $s:=pq$. *Tegyük fel, hogy* $lnko(m,\varphi(s))=1$. *Akkor az* $\mathcal{N}_{P,s}$ *függvény* P_{coll} *ütközési valószínűsége teljesíti a*

$$P_{coll}<\frac{C}{s},$$

egyenlőtlenséget, ahol a C *konstans csak a* P *polinomtól függ.*

A tételek bizonyításához két lemmára lesz szükségünk. Az első a klasszikus Schöneman - Eisenstein tétel megfelelője polinom gyűrűkre és egy következménye Capelli tételének (lásd [54] 662. oldal). A teljesség kedvéért azonban mi is adunk rá bizonyítást.

3. Lemma. *Legyen* $P(X)\in\mathbb{C}[X]$ *egy polinom, amelynek van legalább egy egyszeres gyöke. Akkor a* $Z^n-P(X)\in\mathbb{C}[X,Z]$ *polinom abszolút irreducibilis minden* $n\geq 1$ *esetén.*

Bizonyítás.
Ha $n=1$, akkor az állítás nyilvánvaló. A továbbiakban legyen $n\geq 2$. Tegyük fel indirekt módon, hogy a $Z^n-P(X)$ polinom reducibilis. Ekkor

$$\begin{aligned}Z^n-P(X)=&\left(Z^k+P_{k-1}(X)Z^{k-1}+\cdots+P_0(X)\right)\\&\left(Z^{n-k}+Q_{n-k-1}(X)Z^{n-k-1}+\cdots+Q_0(X)\right),\end{aligned}\tag{25}$$

ahol k egy egész úgy, hogy $0<k<n$ teljesül.

Jelölje α a $P(X)$ polinom egy egyszeres gyökét. A (25) szerint azt kapjuk, hogy $-P(X)=P_0(X)Q_0(X)$ és így α vagy P_0 vagy Q_0 gyöke, de nem lehet gyöke mindkettőnek. Az általánosság megsértése nélkül tegyük fel, hogy $P_0(\alpha)=0$ és $Q_0(\alpha)\neq 0$. Legyen $P_i(X)$ és $Q_j(X)$ a konstans 0 polinom minden

$i > k$ és $j > n-k$ esetén, és $P_k(X) := Q_{n-k}(X) := 1$, a konstans 1 polinom. Ha kiszámítjuk (25) jobb oldalát, akkor azt kapjuk, hogy

$$Z^n - P(X) = Z^n + \sum_{i=0}^{n-1} \left(\sum_{j=0}^{i} P_j(X) Q_{i-j}(X) \right) Z^i \tag{26}$$

és így

$$\sum_{j=0}^{i} P_j(X) Q_{i-j}(X) \equiv 0 \qquad \text{minden } i = 1, \ldots, n-1 \text{ esetén.} \tag{27}$$

Most $i = 1$ esetében (27) azt mutatja, hogy $P_0(X)Q_1(X) + P_1(X)Q_0(X) \equiv 0$ és így $P_1(\alpha) = 0$. Felhasználva a (27) egyenletet sorban bizonyítható az $i = 2, \ldots, n-1$ esetekben is, hogy $P_i(\alpha) = 0$. Az $X = \alpha$ helyettesítést elvégezve (25)-ben, azt kapjuk, hogy

$$Z^n = Z^n + Q_{n-k-1}(\alpha) Z^{n-1} + \cdots + Q_0(\alpha) Z^k,$$

ahol ez az egyenlet egy $\mathbb{C}[Z]$-beli polinomegyenlet. Azonban ez egy ellentmondás, mivel $0 < k < n$ és $Q_0(\alpha) \neq 0$. Ezzel a lemma bizonyított. □

4. Lemma. *Legyen* $f(\underline{X}) := f(X_1, \ldots, X_m) := \prod_{i=1}^{n} (\alpha_{i1} X_1 + \cdots + \alpha_{im} X_m) \in \mathbb{C}[X_1, \ldots, X_m]$ *egy forma, amelyre teljesül, hogy* $\alpha_{ij} \in \mathbb{C}$ *minden* $1 \leq i \leq n$, $1 \leq j \leq m$ *esetén, és* $\prod_{i=1}^{n} \prod_{j=1}^{m} \alpha_{ij} \neq 0$. *Továbbá tegyük fel, hogy létezik* $1 \leq j_1 < j_2 \leq n$ *úgy, hogy*

$$\frac{\alpha_{ij_1}}{\alpha_{ij_2}} \notin \left\{ \frac{\alpha_{kj_1}}{\alpha_{kj_2}} : 1 \leq k \leq n, k \neq i \right\}. \tag{28}$$

Akkor az $f(\underline{X}) - a$ *polinom irreducibilis* $\mathbb{C}[\underline{X}]$*-ben minden* $0 \neq a \in \mathbb{C}$ *esetén.*

Bizonyítás. Tegyük fel indirekt módon, hogy $f(\underline{X}) - a$ reducibilis. Legyen $f_0(X_{j_1}, X_{j_2}) = \prod_{i=1}^{n} (\alpha_{ij_1} X_{j_1} + \alpha_{ij_2} X_{j_2}) \in \mathbb{C}[X_{j_1}, X_{j_2}]$.

Könnyen belátható, hogy az $f(\underline{X}) - a$ reducibilitásából következik $f_0(X_{j_1}, X_{j_2}) - a$ reducibilitása. Azonban ha létezik az $f(\underline{X}) - a$ nem triviális $f(\underline{X}) - a = g(\underline{X})h(\underline{X})$ felbontása, akkor ezt felhasználva az $f(\underline{X}) - a$ teljes fokszáma n és $0 < \deg_{X_i}(f(\underline{X}) - a) < n$ minden $i = 1, \ldots, m$ esetén. Ebből

pedig az következik, hogy $0 < \deg_{X_i} g(\underline{X}) < n$ és $0 < \deg_{X_i} h(\underline{X}) < n$ minden $i = 1, \ldots, m$ esetén. Így az $f(\underline{X}) - a$ minden nem triviális faktorizációja indukálja az $f_0(X_{j_1}, X_{j_2}) - a$ nem triviális faktorizációját. Továbbá

$$f_0(X_{j_1}, X_{j_2}) - a = a_0 X_{j_1}^n + a_1 X_{j_1}^{n-1} X_{j_2} + \cdots + a_n X_{j_2}^n - a = \\ = X_{j_2}^n \left(a_0 \left(\frac{X_{j_1}}{X_{j_2}} \right)^n + a_1 \left(\frac{X_{j_1}}{X_{j_2}} \right)^{n-1} + \cdots + a_n \right) - a.$$

Most legyen $P(Y) := a_0 Y^n + a_1 Y^{n-1} + \cdots + a_n \in \mathbb{C}[Y]$ és $Z := a^{1/n} X_{j_2}^{-1}$. Mivel $f_0(X_{j_1}, X_{j_2}) - a$ reducibilis $\mathbb{C}[X_{j_1}, X_{j_2}]$-ben, így a $P(Y) - Z^m \in \mathbb{C}[Y, Z]$ polinom szintén reducibilis $\mathbb{C}[Y, Z]$-ben. Azonban (28) alapján az következik, hogy $P(Y)$ rendelkezik egy egyszeres gyökkel, és így a 3. Lemma szerint $P(Y) - Z^n$ irreducibilis, ami ellentmondás. Ezzel a lemma bizonyított. □

Most már rétérhetünk a tételek bizonyítására.

A 7. Tétel bizonyítása. Először tegyük fel, hogy $lnko(b, s) = 1$. Tekintsük az $\mathcal{N}_P(X_1, ..., X_m) \equiv b \pmod{s}$ egyenletet és legyen $f := \mathcal{N}_P(X_1, ..., X_m) - b$. A 4. Lemma szerint az f polinom abszolút irreducibilis. Választhatjuk azt, hogy $i = 1$, $j_1 = 2$ és $j_2 = 1$, így most $\alpha_1 \notin \{\alpha_2, \ldots, \alpha_n\}$, és mivel a P polinomnak nincsenek többszörös gyökei, a 4. Lemma feltételei teljesülnek, és így az f polinom valóban abszolút irreducibilis.

Most pedig alkalmazhatjuk S. Lang és A. Weil [42] tételét, amely szerint ha a p prím modulus elegendően nagy, akkor az $f \equiv 0 \pmod{p}$ egyenlet $N(f, p)$ megoldásainak száma teljesíti az

$$\left| N(f, p) - p^{m-1} \right| < C(f) p^{m-1-\frac{1}{2}} \tag{29}$$

egyenlőtlenséget, ahol a $C(f)$ konstans értéke csak az f abszolút irreducibilis polinomtól függ. Továbbá megjegyezzük, hogy a mi esetünkben a $C(f)$ konstans értéke nem függ a b értékétől sem. Mivel $lnko(m, \varphi(s)) = 1$ minden $0 \neq b \in \mathbb{Z}_s$ esetén, ezért létezik egy $0 \neq a \in \mathbb{Z}_s$ úgy, hogy $a^m \equiv b \pmod{s}$ és így azt kapjuk, hogy

$$\mathcal{N}_P(X_1, ..., X_m) - b \equiv \mathcal{N}_P(X_1, ..., X_m) - a^m \\ \equiv a^m \left(\mathcal{N}_P \left(\frac{X_1}{a}, ..., \frac{X_m}{a} \right) - 1 \right) \pmod{s}.$$

Ezért az $f \equiv 0 \pmod{s}$ egyenlet megoldásai származtathatók az $\mathcal{N}_P(Y_1, ..., Y_m) \equiv 1 \pmod{s}$ egyenlet megoldásaiból úgy, hogy alkalmazzuk az $(X_1, ..., X_m) = a(Y_1, ..., Y_m)$ lineáris transzformációt. Tehát az

$\mathcal{N}_P(Y_1, ..., Y_m) \equiv 1 \pmod{s}$ egyenlet megoldásainak száma valóban nem függ a b értéktől.

A (29) felhasználásával azt kapjuk, hogy

$$|N(P, b, p) - p^{m-1}| < C_1 p^{m-1-\frac{1}{2}} \tag{30}$$

és

$$|N(P, b, q) - q^{m-1}| < C_1 q^{m-1-\frac{1}{2}}. \tag{31}$$

A kínai maradéktétel (lásd [18]) szerint az $f \equiv 0 \pmod{s}$ kongruencia megoldásainak száma $N(P, b, s) = N(P, b, p)N(P, b, q)$. Így azt kapjuk, hogy

$$\begin{aligned} N(P, b, s) &< (pq)^{m-1} + (pq)^{m-1}\left(\frac{C_1}{p^{1/2}} + \frac{C_1}{q^{1/2}}\right) + (pq)^{m-1}\frac{C_1^2}{(pq)^{1/2}} \\ &< (pq)^{m-1} + (pq)^{m-1}\frac{C_2}{pq^{3/2}} < s^{m-1} + \frac{C_3}{s^{m-1-1/4}}. \end{aligned}$$

Hasonlóan kapjuk az

$$N(P, b, s) > s^{m-1} - \frac{C_4}{s^{m-1-1/4}}$$

egyenlőtlenséget is, amivel az állítás első része bizonyított.

Most tegyük fel, hogy $lnko(b, s) \neq 1$. Ha $lnko(b, p) \neq 1$ akkor $\mathcal{N}_P(X_1, \ldots, X_m) \equiv 0 \pmod{p}$, és mivel $\mathcal{N}_P$ lineáris faktorokra bomlik, a kongruencia összes megoldását a $\mathbb{Z}_p^m$ legfeljebb n darab $m-1$ dimenziós altereinek egyesítése tartalmazza. Így $N(P, b, p) < np^{m-1}$. Ha $lnko(b, s) = 1$ akkor (30) szerint $N(P, b, p) < C_5 p^{m-1}$. Hasonlóan belátható ugyanez $N(P, b, q)$ esetére is. Így azt kapjuk, hogy bármely b esetén

$$N(P, b, s) < c_2(P)s^{m-1}. \tag{32}$$

Ezzel a tétel bizonyítást nyert. □

A 8. Tétel bizonyítása. A (32) alapján azt kapjuk, hogy

$$P_{coll} := \frac{N(P, b, s)}{s^m} < \frac{C}{s}.$$

Ezzel a tétel bizonyított. □

1. Következmény. *Ha az s modulus elég nagy, akkor az $\mathcal{N}_{P,s}$ függvény a 10. Definició szerint ütközésmentes.*

Most magvizsgáljuk az $\mathcal{N}_P$ és $\mathcal{N}_{P,s}$ függvényeket az input és output összefüggésének szempontjából. Ezt a lavinahatás írja le, amit azonban nem sikerült megfelelő matematikai módszerekkel kezelni, így csak teszt eredményeket tudunk közölni.

Az $\mathcal{N}_P$ és $\mathcal{N}_{P,s}$ függvények lavina hatásának teszteléséhez egy MAPLE programot készítettünk, amely az alábbi relatív Hamming-távolságokat határozza meg:

$$RH_{P,s} := \frac{\rho(\mathcal{N}_{P,s}(\underline{x}), \mathcal{N}_{P,s}(\underline{x}'))}{l(s)}$$

és

$$RH_P := \frac{\rho(\mathcal{N}_P(\underline{x}), \mathcal{N}_P(\underline{x}'))}{l(\mathcal{N}_P(\underline{x}'))},$$

ahol az $\underline{x}$ egy input az $\underline{x}'$ pedig a hozzá tartozó egy bitben megváltoztatott input, valamint $l(.)$ jelöli a bináris hossz függvényt és $\rho(.,.)$ a 14. Definicióban megadott Hamming-távolságot. A program forráskódja megtalálható a dolgozat végén, a MAPLE programok fejezetben.

A tesztet több alkalommal is ezerszer futtattuk véletlenszerűen generált 2^{1024} nagyságrendű input adatokkal. Az $\underline{x}'$ input egyetlen bitjének megváltoztatása is véletlenszerű volt.

Az $RH_{P,s}$ értékek átlaga 0.50 volt, 0.004 szórással. Az $\mathcal{N}_P$ függvény tesztje egy kicsit rosszabb eredményt hozott, az RH_P értékek átlaga 0.48 volt, 0.003 szórással. Ez a függvény csak közelítőleg teljesíti az 13. Definícióban megfogalmazott erős lavinahatás kritériumot.

3. Sejtés. *Az $\mathcal{N}_{P,s}$ függvény teljesíti az 13. Definícióban megfogalmazott erős lavinahatás kritériumot.*

Az $\mathcal{N}_{P,s}$ függvény egyirányú hash függvény

Az előző részek eredményei azt mutatják, hogy az $\mathcal{N}_P$ és $\mathcal{N}_{P,s}$ függvények invertálása általában nehéz, míg a függvényérték meghatározása könnyű. Ezt a tényt az alábbi feltétel fejezi ki.

15. Definíció. *Erős Moduláris Normaforma Feltétel (SMNA): Minden Q polinom és $\mathcal{A}$ PPT algoritmus esetén, ha az s elegendően nagy*

$$P[\mathcal{A}(s,b) = (x_1, ..., x_m) \text{ úgy, hogy } b = \mathcal{N}_{P,s}(x_1, ..., x_m)] < \frac{1}{Q(k)},$$

ahol $x_i \in \mathbb{Z}$ és $1 \leq \max\{|x_i|\} \leq s-1$, valamint a valószínűséget úgy kell venni, hogy az x_i inputok befutják lehetséges értékeiket, és az $\mathcal{A}$ algoritmus is az érmefeldobásait.

9. Megjegyzés. *Az előbbihez hasonlóan az $\mathcal{N}_P(\underline{X})$ függvényhez lehet definiálni az Erős Normaforma Feltételt (SNA), ami az $\mathcal{N}_P$ függvény invertálásának keresztülvihetetlenségét fejezi ki.*

Bizonyítható a következő tétel.

9. Tétel. *Az SMNA teljesülése esetén az $\mathcal{N}_{P,s}$ függvény egyirányú hash függvény.*

Bizonyítás. Meg kell mutatnunk, hogy az $\mathcal{N}_{P,s}$ függvény megfelel a 2. és 7. definícióknak.

Mivel a függvényérték kiszámításának bonyolultsága $O(n^5 \log^2 s)$ (lásd 6. Tétel) a 2. Definíció (1) része és a 7. Definíció (1) része is teljesül, a (2) része pedig nyilvánvaló a függvény moduláris konstrukciója miatt.

A 2. Definíció (2) részének teljesülése egyenes következménye az SMNA feltételnek. Ezzel a tétel bizonyított. □

10. Megjegyzés. *A fentihez hasonlóan bizonyítható, hogy az SNA teljesülése esetén az $\mathcal{N}_P$ függvény egyirányú függvény.*

A kriptográfiai gyakorlat általában nem egy egyirányú függvényt alkalmaz, hanem inkább *egyirányú függvény családot.* Egy ilyen családot fogunk készíteni az $\mathcal{N}_{P,s}$ függvény felhasználásával. Tekintsük az alábbi konstrukciót:

4. Konstrukció. *Legyen $P(X) \in \mathbb{Z}[X]$ egy olyan főpolinom, amelynek nincsenek többszörös gyökei. Legyenek p és q elegendően nagy prímszámok úgy, hogy $q > p > q/2$ és $s := pq$. Tegyük fel továbbá, hogy $lnko(m, \varphi(s)) = 1$. Definiáljuk a $\mathcal{P} = \{P(X) : deg(P(X)) = n \geq 4,\ rögzített\}$ polinom halmazt és az*

$$\mathbf{MNFF} = \left\{ \mathcal{N}_{P,s} : \mathbb{Z}_s^m \mapsto \mathbb{Z}_s;\ \mathcal{N}_{P,s}(\underline{X}) = (\prod_{i=1}^{n} L_i(\underline{X}), \mathrm{mod} s) \right\}_{P \in \mathcal{P}}$$

függvény családot, ahol $3 \leq m \leq n$, és az $L^{(i)}(\underline{X})$ definíciója ugyanaz, mint korábban.

11. Megjegyzés. *A fenti konstrukcióban a biztonsági paraméter az s konstans.*

Bizonyítható a következő tétel.

10. Tétel. *Ha minden $P(X) \in \mathcal{P}$ polinomhoz tartozó $\mathcal{N}_{P,s}$ függvény teljesíti az SMNA-t, akkor az MNFF függvény halmaz egy egyirányú hash függvény család.*

Bizonyítás. Meg kell mutatnunk, hogy ha az $\mathcal{N}_{P,s}$ hash függvények teljesítik az SMNA-t, akkor az MNFF függvényhalmaz teljesíti a 6. Definíció feltételeit.

Nyilvánvalóan az $\mathcal{N}_{P,s}$ függvények értelmezési tartománya és értékkészlete véges.

Az $\mathcal{S}_1$ algoritmus kiválaszt egy $P \in \mathbb{Z}[X]$ polinomot és s modulust, amely megfelel a konstrukcióban leírtaknak.

Az $\mathcal{S}_2$ algoritmus pedig kiválasztja az $\underline{x}$ inputnak megfelelő $\mathbb{Z}_s^m$ értelmezési tartományt.

Legyen az $\mathcal{A}_1$ algoritmus a 6. Tételben leírt algoritmus. Ez az $\mathcal{N}_{P,s}(\underline{x}) \in \mathbb{Z}$ függvényértéket $O(n^5 \log^2 s)$ polinomiális bonyolultsággal számítja ki.

Mivel az $\mathcal{S}_1, \mathcal{S}_2$ és $\mathcal{A}_1$ algoritmusok bonyolultsága polinomiális, a 6. Definíció (1), (2) és (3) feltételei teljesülnek. A (4) feltétel pedig egyenes következménye az SMNA-nak. Ezzel a tétel bizonyított. □

12. Megjegyzés. *A fentihez hasonló módon lehet az $\mathcal{N}_P$ függvény alkalmazásával megkonstruálni az NFF függvénycsaládot, amelyről bizonyítani lehet, hogy SNA esetén az NFF egy egyirányú függvénycsalád.*

Alkalmazások

Az $\mathcal{N}_P$ és $\mathcal{N}_{P,s}$ függvény alkalmazható minden szokásos kriptográfiai feladatra. Többek között használható adatintegritás ellenőrzésére, felhasználó azonosításra, digitális aláírásban és álvéletlen sorozatok generálásában. Ezek a függvények fontos szerepet játszhatnak kriptográfiai protokollok felépítésében.

Most egy rekurzív hash függvényt konstruálunk az $\mathcal{N}_{P,s}(x_1, ..., x_m)$ függvény alkalmazásával.

Legyen adott az M üzenet, amelyet egy tetszőleges hosszúságú bináris szónak tekintünk. Ezt szétvágjuk $x_1, ..., x_k$ részszavakra úgy, hogy minden x_i $(1 \leq i \leq k)$ egy egész számot reprezentáljon az $[1, s-1]$ intervallumban.

Tegyük fel, hogy $k \geq m$, egyébként pedig egészítsük ki a sorozatot nullákkal. Először definiáljuk a

$$h(x_1, \ldots, x_m) := N_{P,s}(x_1, \ldots, x_m)$$

kezdőértéket, majd pedig legyen rekurzív módon

$$h(x_1, \ldots, x_{m+t(m-1)}) := \\ N_{P,s}(h(x_1, \ldots, x_{m+(t-1)(m-1)}), x_{m+(t-1)(m-1)+1}, \ldots, x_{m+t(m-1)}).$$

Ha k valamely alkalmas $t \in \mathbb{Z}$ számra nem $m + t(m-1)$ alakú, akkor egészítsük ki az M üzenetet 0-val, mindaddig, míg k megfelelő alakú lesz.

Most vizsgáljuk meg azt a kérdést, hogy miként kell megválasztani a $P(X)$ polinomot, hogy a lehető legegyszerűbben lehessen számolni a függvényértéket.

Legyen $P(X) := X^n - 1$ úgy, hogy $n \geq 3$. Jelölje ζ ennek egy 1-től különböző gyökét és legyen

$$L(\underline{X}) := X_1 + \zeta X_2 + \cdots + \zeta^{n-1} X_n.$$

A

$$\zeta^j L(\underline{X}) = X_{n-j+1} + \zeta X_{n-j+2} + \cdots + \zeta^{j-1} X_n + \zeta^j X_1 + \cdots + \zeta^{n-1} X_{n-j}$$

egyenletből, amely minden $1 \leq j \leq n$ esetén fennáll, látható, hogy az $\mathcal{N}_{P,s}(\underline{X})$ egy

$$\begin{pmatrix} X_1 & X_2 & \ldots & X_n \\ X_n & X_1 & \ldots & X_{n-1} \\ \ldots & \ldots & \ldots & \ldots \\ X_2 & X_3 & \ldots & X_1 \end{pmatrix}$$

ciklikus mátrix determinánsa, amely nagyon egyszerű formájú. Ha $L(\underline{X}) = X_1 + \zeta X_2 + \cdots + \zeta^{m-1} X_m$, ahol $m < n$, akkor $X_{m+1}, \ldots, X_n$ helyett minden sorban $n - m$ darab 0 áll, de a determináns értéke nem lesz 0, csak a formája lesz még egyszerűbb.

Mivel a $P(X)$ polinomnak csak egyszeres gyökei vannak, teljesülnek a 8. Tétel feltételei, és így az $\mathcal{N}_{P,s}(\underline{X})$ függvény ütközésmentes.

Javasoljuk az $N_{P,s}(\underline{X})$ egyirányú hash függvény és a segítségével konstruált MNFF egyirányú hash függvénycsalád alkalmazását a gyakorlatban.

A gyakorlatban használható algoritmus a következő:

4. Algoritmus (`Hash érték meghatározás az` $\mathcal{N}_{P,s}(\underline{X})$ `függvény segítségével`). Adott az M üzenet, mint input és a $P(X)$ polinom $t_0, ..., t_n \in \mathbb{Z}$ együtthatói. Az algoritmus meghatározza a $h(M)$ hash értéket.

1. [`Inicializálás és konstans definíció`] Válasszuk az n értékét úgy, hogy $n \geq 3$ legyen. Válasszuk ki m-et az $n \geq m \geq 3$ feltétel szerint. Válasszuk ki az 1024 bites $s := pq$ modulust, ahol p és q titkos prímek. Ehhez először választani kell egy 2^{512} nagyságrendű q prímet, majd pedig egy p prímet úgy, hogy a $q > p > q/2$ és $lnko(m, (p-1)(q-1)) = 1$ feltételek teljesüljenek. Ez Csebisev prímszám tétele szerint biztosan lehetséges. Ezután rögzítjük az $s := pq$ modulust és a p és q prímeket pedig megsemmisítjük.

2. [`Szétvágás blokkokra`] Vágjuk szét az M üzenetet $x_1, \ldots, x_k$ részszavakra úgy, hogy minden x_i, $(i = 1, \ldots, k)$ egy egész számot reprezentáljon az $[1, s-1]$ intervallumból.

3. [`Kiegészítés`] Egészítsük ki az előző lépésben kapott blokkok sorozatát alkalmazva az alábbi rekurziót:

$$h(x_1, \ldots, x_m) := \mathcal{N}_{P,s}(x_1, \ldots, x_m)$$

és

$$h(x_1, \ldots, x_{m+t(m-1)}) := \\ N_{P,s}(h(x_1, \ldots, x_{m+(t-1)(m-1)}), x_{m+(t-1)(m-1)+1}, \ldots, x_{m+t(m-1)}).$$

Ha k valamely alkalmas $t \in \mathbb{Z}$ számra nem $m + t(m-1)$ alakú, akkor egészítsük ki az M üzenetet 0-val, mindaddig, míg k megfelelő alakú lesz.

4. [`A számítások elvégzése`] Számítsuk ki a $h(x_1, \ldots, x_k) \in \mathbb{Z}_s$ hash értéket alkalmazva az előző pontban leírt rekurziót.

5. [`Befejezés`] Az algoritmus végeredménye a $h(M) = h(x_1, \ldots, x_k)$ érték lesz.

Így kaptunk egy gyakorlatban is használható h egyirányú hash függvényt. Ez a függvény minden iterációs lépésben megtartja ütközésmentes tulajdonságát és sejthetően rendelkezik lavina hatással is (lásd 8. Tétel és 3. Sejtés). Ezek a tulajdonságok garantálják biztonságos használatát.

Most pedig bemutatunk egy példát a h függvény használatára.

Egyre több informatikai rendszer alkalmaz a felhasználó azonosításához jelszó (password) helyett jelmondatot (passphrase). A jelmondat lényegesen hosszabb, mint a jelszó, valamint általában értelmes mondat, amely tartalmaz kisbetűket, nagybetűket és írásjeleket. Az ellenőrzése hasonlóan történhet, mint a jelszó esetében. Az informatikai rendszer az alábbi táblázatot tárolja:

Név	**Hash érték**	**Salt**	**Mod**
Alice	$h(PPH_1)$	ST_1	s
Bob	$h(PPH_2)$	ST_2	s
...	...	...	...

Legyenek most az előzőekben használt jelölés szerint a paraméterek értékei a következők: $n := 4$, $m := 3$, $s := pq$, ahol az s modulust tároljuk a fenti táblázatban, de a p, q prímeket megsemmisítjük. Az s modulus mérete 1024 bit. Rögzítjük a ciklikus mátrixot, amely most a következő alakú:

$$\begin{pmatrix} X_1 & X_2 & X_3 & 0 \\ 0 & X_1 & X_2 & X_3 \\ X_3 & 0 & X_1 & X_2 \\ X_2 & X_3 & 0 & X_1 \end{pmatrix}.$$

A minden felhasználóhoz külön generált ST_i véletlen sztring 128 bájt méretű, és két 64 bájtos véletlen sztring konkatenációja az alábbiak szerint:

$$ST_i := RS_{i,1} + RS_{i,2},$$

ahol $RS_{i,j}$ véletlenszerűen generált 64 bájtos sztring.

A jogosult felhasználók jelmondatát is két részre választjuk szét az alábbiak szerint:

$$PPH_i := PPH_{i,1} + PPH_{i,2},$$

ahol $PPH_{i,j}$ a jelmondat két része, amelyek mérete legalább 10 bájt.

Így most a h függvényértéket a következő módon számítjuk ki:

$$h(PPH_i) := \mathcal{N}_{P,s}(PPH_{i,1} + RS_{i,1}, PPH_{i,2} + RS_{i,2}, ST_i),$$

ahol az első két argumentum legalább 74 bájt, a harmadik pedig 128 bájt méretű.

A rendszer például Alice jelszavát akkor fogadja el, ha

$$h(PPH_A) = h(PPH_1),$$

ahol PPH_A az Alice által megadott jelmondat és $h(PPH_1)$ pedig Alice jelmondatának tárolt hash értéke.

Az alkalmazás használatához mindössze a fenti táblázatot és ciklikus mátrixot, valamint a működéshez szükséges algoritmust kell tárolni.

Végül bemutatunk egy konkrét példát az $\mathcal{N}_{P,s}$ függvény használatára. Ebben az alábbi szöveg hash értékét fogjuk meghatározni:

"Ebben a dolgozatban egy új egyirányú függvény elkészítését fogjuk leírni. A konstrukció megalkotásában tisztán matematikai eszközöket fogunk használni. Egyirányú függvényünk invertálási nehézségét az algebrai számelméletből jól ismert általános normaforma egyenletek megoldásának nehézségére alapozzuk. Igyekszünk függvényünkről minél több jó tulajdonságot matematikai módszerekkel bizonyítani."

A szövegből eltávolítjuk az írásjeleket és a szóközöket, valamint csak ékezetmentes betűket használunk, így kapjuk az

M:=EbbenadolgozatbanegyujegyiranyufuggvenyelkeszitesetfogjukleirniAkonst rukciomegalkotasabantisztanmatematikaieszkozoketfogunkhasznalniEgyirany ufuggvenyunkinvertalasinehezsegetazalgebraiszamelmeletboljolismertaltalano snormaformaegyenletekmegoldasanaknehezsegerealapozzukIgyekszunkfuggve nyunkrolmineltobbjotulajdonsagotmatematikaimodszerekkelbizonyitani

üzenetet, ami az algoritmus inputja. A korábbi jelöléseket használva, az egyes paraméterek értékei az alábbiak:

$n := 4$, $m := 3$. A két 512 bináris jegyű prímszám:
q:=
p:=
A modulus:
s:=

627403426474159611888796609416764532436734428785540410262641531413965733845673498328716727827370341996619576661367652180056591

Az x_1, x_2 és x_3 ASCII kódolással kapott értékek az alábbiak:

x_1:=1403571638454077782928457294557729315881679388293470444795387489392773864231807512713207239184880737688128258955736412090930841644177433858717865267618790422853092409476038864873159130160975732981083113717731441746814940738424318780434541435726705679479430839062332505422

x_2:=9184514886826311531411310815794572804533622386562099578717248777356818116082125446065867311536346856609590396692064286900318111224608721751481403518900750169472682229167163552205126721885209989993957122272113691467040429877687177459273473638627141933599468245317107636 1

x_3:=3426577422893136844564128890891932215934401255636468803503586349134337654079247214772239355705945711024737508067881634943288413924828866544545977271834657532390418007874610435882798522834908303 4

És végül a hash érték:

$h(M) = \mathcal{N}_{P,s}(x_1, x_2, x_3)$=6818912767035557111325959155356468437011330025782168816535995676748610853453114883768554008914924721479723978154075785738973686039913741130275355212765423507742410415730334170029370825230781522575865508387826473955607769349867110322605605362378093875877534901069459528782486131310327805770998696136567862955 0

A példában a p és q prímek bináris hossza 512, illetve 513 bit, az s modulusé pedig 1025 bit. Az x_1, x_2 blokkok hossza 128 bájt, az x_3 pedig 92 bájt. A $h(M)$ hash érték bináris hossza pedig 1023 bit.

Ha az M üzenet első betűjét E-ről F-re változtatjuk, akkor a $h(M')$ hash érték lényegesen megváltozik:

$h(M')$=3420450883753292803580040595512330323197613905242755161899932986872240915988135543623648106919478969352971099633019833791815615584559053066650516114348847371432707728205275735223109772554764918202177786404641683924792284669499436384603740403103240063022306110965023179745996742429113532266379739537009294969 26

A két hash érték relatív Hamming-távolsága:

$$\varrho(h(M), h(M'))/l(s) = 0.49875,$$

ami jól mutatja az erős lavina hatást.

A példa azt mutatja, hogy az $\mathcal{N}_{P,s}$ függvény a gyakorlatban jól használható hash függvényként.

MAPLE programok

Ebben a fejezetben különféle MAPLE rendszerben készített implementációk forráskódját mutatjuk be.

Inicializáló eljárás az $\mathcal{N}(\underline{x})$ függvényérték kiszámításának teszteléséhez

```
INIT:=proc(n,u,Et,Ey)

local i,Rnd;
global t,T,x,y;

restart:
with(linalg):
readlib(randomize)():
# Paraméterek:
# n: a számtest T definiáló polinomjának fokszáma,
# u: az L lineáris forma tagjainak száma,
# Et: a T polinom együtthatóinak nagyságrendje,
# Ey: a helyettesítési értékek nagyságrendje (Et < =Ey),
# t: a T polinom együtthatói,
# x: a T polinom ismeretlenje
# y: a helyettesítési értékek,

t:=array(0..n-1):y:=array(0..u-1):

# A T irreducibilis polinom együtthatóinak generálása :
T:=1:
while not irreduc(T) do
Rnd:=rand(2*10^(Et-2)..2*10^Et)-10^Et:
```

```
for i from 0 to n-1 do
t[i]:=Rnd():
od:

# A T polinom meghatározása Horner-elrendezéssel:
T:=x:for i from 1 to n-1 do
T:=(T+t[n-i])*x
od:
T:=expand(T+t[0]);
od:

# A véletlen helyettesítési értékek generálása:
Rnd:=rand(2*10^(Ey-2)..2*10^Ey)-10^Ey:
for i from 0 to u-1 do
y[i]:=Rnd():
od:
end;
```

Az ALG1 eljárás

```
ALG1:=proc(n,u,T,t,y)

local i,j, A,xmax,Rroot,st,a,L:
global N1,tt,Delta:

# A normaforma helyettesítési értékének kiszámítása
# a definíció alapján

Rroot:=vector(n):

# A T polinom összes gyöke valós alakban:
Rroot:=fsolve(T,x,complex):

# A számolási pontosság meghatározása
# A: a legnagyobb konjugáltak összege,
# X: a legnagyobb helyettesítési érték:
a:=0:
for i from 1 to n do
```

```
if (abs(Rroot[i])>a) then a:=abs(Rroot[i])
fi
od:
A:=a^(u-1):
xmax:=0:
for i from 0 to u-1 do
if (abs(y[i])> xmax) then xmax:=abs(y[i])fi
od:
Delta:=ceil(evalf(log10(n*u*xmax^n*A^n))):
Digits:=Delta:

# A T polinom gyökei valós alakban (még egyszer a
# számolási pontosság változtatása miatt):
st:=time(): #idömérés indul
Rroot:=fsolve(T,x,complex):

# Az L forma normája helyettesítési értékének
# meghatározása:
N1:=1:
for i from 1 to n do
L := 0:
for j from 0 to u-1 do
L := L+y[j]*Rroot[i]^j
od:
N1:=N1*L:
od:
tt[1]:=time()-st: #idömérés vége
end;
```

Az ALG2 eljárás

```
ALG2:=proc(n,u,t,y)

local i,j,k,st,R,La,s;
global N2,tt;

# A normaforma helyettesítési értékeinek kiszámítása
# egész aritmetikával,
```

```
# mátrix reprezentáció használatával
# Az R és La mátrixok definiálása és a azámolási pontosság
# visszaállítása alapértelmezettre:
R:=array(0..2*n-2,0..n-1):
La:=array(1..n,1..n):
Digits:=10:

# Az R mátrix elemeinek meghatározása a
# Newton rekurziós formulával:

for i from 0 to n-1 do
for j from 0 to 2*n-2 do
R[j,i]:=0
od
od:
for i from 0 to n-1 do
R[i,i]:=1
od:
for i from 0 to n-1 do
R[n,i]:=-t[i]
od:
for i from 1 to n-2 do
R[n+i,0]:=-t[0]*R[n+i-1,n-1]:
for j from 1 to n-1 do
R[n+i,j]:=R[n+i-1,j-1]-t[j]*R[n+i-1,n-1]
od
od:

# A  La  mátrix elemei (az F lineáris polinomok)
# helyettesítési értékének kiszámítása:
st:=time(): # idömérés indul
for k from 0 to n-1 do
for j from 0 to n-1 do
s:=0:
for i from 0 to u-1 do
s:=s+R[i+k,j]*y[i]
od:
La[k+1,j+1]:=s:
```

```
od:
od:
# A  La  mátrix determinánsának kiszámítása:
N2:=det(La):
tt[2]:=time()-st: # idömérés vége
end;
```

Az ALG3 eljárás

```
ALG3:=proc(n,u,T,t,y,w)

local i,j,k,l,R,tmax,xmax,B,Em,m,Lm,Vm,Lam,s,st,Ld,PR,Prod:
global N3,tt,mdb;

# A norma helyettesítési értékének kiszámítása
# mátrix reprezentációval és
# moduláris aritmetikával:

# A B konstans meghatározása
tmax:=0:
for i from 0 to n-1 do
   if (abs(t[i])>tmax) then tmax:=abs(t[i]) fi
od:
xmax:=0:
for i from 0 to u-1 do
   if (abs(y[i])> xmax) then xmax:=abs(y[i])fi
od:
B:=tmax^(2*n)*xmax^n*n!:

# A PRÍMHATVÁNY modulusok meghatarozasa:
PR[1]:=2:Vm[1]:=PR[1]^w:Prod:=Vm[1]:mdb:=1:
for i from 2 while (Prod < 2*B) do
PR[i]:=nextprime(PR[i-1]):
for j from 2 while (PR[i]^j < Vm[1]) do
Vm[i]:=PR[i]^j:
od:
mdb:=mdb+1:Prod:=Prod*Vm[mdb]:
od:
```

```
# A modulusok listájának meghatározása:
Lm:=[]:
for i from 1 to mdb do
Lm:=[op(Lm),Vm[i]]
od:

# Az R és La mátrixok definiálása és a azámolási
# pontosság visszaállítása
# alapértelmezettre:
R:=array(0..2*n-2,0..n-1):
for i from 1 to mdb do
Lam[i]:=array(1..n,1..n):s[i]:=array(1..n)
od:
Digits:=10:

# Az R mátrix elemeinek meghatározása a
# Newton rekurziós formulával:
for i from 0 to n-1 do
   for j from 0 to 2*n-2 do R[j,i]:=0
od
od:
for i from 0 to n-1 do
R[i,i]:=1
od:
for i from 0 to n-1 do
R[n,i]:=-t[i]
od:
for i from 1 to n-2 do
         R[n+i,0]:=-t[0]*R[n+i-1,n-1]:
   for j from 1 to n-1 do
            R[n+i,j]:=R[n+i-1,j-1]-t[j]*R[n+i-1,n-1]
   od:
od:

# A  Lam[l]  mátrixok elemei (az Fm lineáris polinomok)
helyettesítési értékének kiszámítása mod Lm[l]:
‘mod‘:=mods: #Áttérés a legkisebb abszolútértékü
```

```
            # maradékrendszerre
st:=time(): #idömérés inditása
for k from 0 to n-1 do
   for j from 0 to n-1 do
      for l from 1 to mdb do
             s[l]:=0:
          for i from 0 to u-1 do
             s[l]:=s[l]+R[i+k,j]*y[i] mod Lm[l]
          od:
Lam[l][k+1,j+1]:=s[l]
      od:
   od:
od:

# A determinánsok listája CRA-hoz
Ld:=[]:
for i from 1 to mdb do
Ld:=[op(Ld),det(Lam[i]) mod Lm[i]]
od:

# A norma meghatározása kínai maradék algoritmussal (CRA)
N3:=chrem(Ld,Lm):
tt[3]:=time()-st:# idömérés vége

end;
```

Az ALG4 eljárás

```
ALG4:=proc(u,T,y)

local Theta1,st,i,L;
global N4, tt;

# A norma helyettesítési értékének kiszámítása
# a MAPLE Norm() függvényével:

# Az L lineáris forma definiálása:
Theta1:= RootOf(T):
```

```
L := 0:
for i from 0 to u-1 do
L := L+y[i]*Theta1^i
od:
# Az L forma normája:
st:=time(): #idömérés indul
N4:=evala(Norm(L,Theta1)):
tt[4]:=time()-st: #idömérés vége
end;
```

A tesztelő eljárás

```
LNORMA:=proc(No,n,u,Et,Ey,w)

local i,j,TT,DN1,Delt;

TT:=array(1..4):

read ‘ALG1.prg‘:
read ‘ALG2.prg‘:
read ‘ALG3.prg‘:
read ‘ALG4.prg‘:
read ‘INIT.prg‘:

for i from 1 to 4 do
 TT[i]:=0
od:
DN1:=0;
Delt:=0:
for j to No do
INIT(n,u,Et,Ey):
ALG1(n,u,T,t,y):
ALG2(n,u,t,y):
ALG3(n,u,T,t,y,w):
ALG4(u,T,y):
for i from 1 to 4 do TT[i]:=TT[i]+tt[i] od:
Delt:=Delt+Delta:
if (abs(N1-N4)>.5) then DN1:=DN1+1
```

```
fi:
od:

Digits:=6;
print(‘ALG1, ALG2, ALG3 és ALG4 átlagos futási ideje:‘);
for i from 1 to 4
do print(TT[i]/No);
od;

print(‘Az ALG1 pontatlanságainak száma:‘);
print(DN1/No);
print(‘Az ALG1-ben használt értékes jegyek számának átlaga:‘);
print(evalf(Delt/No));
print(‘Az ALG3 modulusainak száma:‘);
print(mdb);
end;
```

A lavina hatást tesztelő eljárások

```
INIT2:=proc(n,m,Er,Ey)

local i,Rnd,P,Q;
global t,S,x,y;

restart:
with(linalg):
readlib(randomize)():
# Paraméterek:
# n: a P polinom fokszáma (n=>3),
# m: az L lineáris forma tagjainak száma (m<=n),
# Er: a P polinom gyökeinek nagyságrendje,
# Ey: a helyettesítési értékek nagyságrendje (2^1024),
# t: a P polinom együtthatói,
# x: a P polinom ismeretlenje,
# y: a helyettesítési értékek,
# S: a modulus (S:=PQ)

t:=array(0..n-1):y:=array(0..m-1):

# A P reducibilis polinom generálása
#(nincsenek többszörös gyökök):
P:=1:
for i from 1 to n do
Rnd:=rand(2*10^(Er-2)..2*10^Er)-10^Er:
P:=P*(x-Rnd()):
od:

# A P főpolinom együtthatóinak meghatározása:
P:=collect(P,x):
for i from 0 to n-1 do
t[i]:=coeff(P,x,i):
od:

# A véletlen helyettesítési értékek generálása:
Rnd:=rand(2*10^(Ey-2)..2*10^Ey)-10^Ey:
for i from 0 to m-1 do
```

```
y[i]:=abs(Rnd()):
od:

# Az S:=PQ modulus meghatározása:
Q:=nextprime(2^512):
P:=nextprime(floor(Q/2)):
while gcd(m,(P-1)*(Q-1))<> 1 do
Q:=nextprime(Q+1):
P:=nextprime(floor(Q/2)):
od:
S:=P*Q:

end;
```

Az $\mathcal{N}_{P,s}(\underline{x})$ függvényértéket meghatározó eljárás

```
ALG5:=proc(n,m,t,y,S)

local i,j,k,R,La,s;
global N5;

# A normaforma helyettesítési értékeinek kiszámítása
# VÉGES TEST FELETT (mod S:=PQ)
# egész aritmetikával, mátrix reprezentáció használatával

# Az R és La mátrixok definiálása:
R:=array(0..2*n-2,0..n-1):
La:=array(1..n,1..n):

# Az R mátrix elemeinek meghatározása Newton rekurzióval:

for i from 0 to n-1 do
for j from 0 to 2*n-2 do R[j,i]:=0 od od:
for i from 0 to n-1 do R[i,i]:=1 od:
for i from 0 to n-1 do R[n,i]:=(-t[i] mod S) od:
for i from 1 to n-2 do
R[n+i,0]:=(-t[0]*R[n+i-1,n-1] mod S):
for j from 1 to n-1 do
R[n+i,j]:=(R[n+i-1,j-1]-t[j]*R[n+i-1,n-1] mod S)
```

```
od
od:

# A  La  mátrix elemei (az F lineáris polinomok)
# helyettesítési értékének kiszámítása:

for k from 0 to n-1 do
for j from 0 to n-1 do
s:=0: for i from 0 to u-1 do
s:=(s+R[i+k,j]*y[i] mod S) od:
La[k+1,j+1]:=s:
od:
od:

# A  La  mátrix determinánsának kiszámítása:
N5:=(Det(La) mod S):
end;
```

A Hamming-távolságot meghatározó eljárás

```
HAMMING:=proc(X,Y)

local BX,BY,S,S1,M,MM,h,i,P2:

global HAM:

BX:=convert(abs(X),base,2):
BY:=convert(abs(Y),base,2):
S:=nops(BX):
S1:=nops(BY):
M:=min(S,S1):MM:=max(S,S1):
h:=0:
for i from 1 to M do
    if BX[i] <> BY[i]then h:=h+1 fi
od:

HAM:=evalf(h/M);
end;
```

A lavina hatást tesztelő eljárás

```
LAVINA:=proc(n,m,Ey,DB)

# DB: a tesztek száma

local TH,j,k,l,NM1,NM2,BY,BYY,Y1,YY,R1,R2,
b,hh,RH,SL,SD,i,P:
global N5,y:
read‘INIT2.prg‘:
read‘ALG5.prg‘:
read‘hamming.prg‘:
TH:=array(1..DB):

readlib(randomize)():

hh:=0:
for j from 1 to DB do
INIT2(n,m,5,Ey):
ALG5(n,m,t,y,S):
NM1:=N5:
R1:=rand(0..u-1):k:=R1():
BY:=convert(y[k],base,2):
R2:=rand(1..nops(BY)):b:=R2():
if op(b,BY)=1 then
      BYY:=subsop(b=0,BY)
              else BYY:=subsop(b=1,BY)fi:
Y1:=convert(BYY,base,2,10):
YY:=sum(Y1[l]*10^(l-1),l=1..nops(Y1)):
y[k]:=YY:
ALG5(n,m,t,y,S):
NM2:=N4:
TH[j]:=HAMMING(NM1,NM2):
hh:=hh+TH[j]:
od:

# A relativ Hamming távolságok átlaga:
RH:=hh/DB:
print(‘Rel.Hamming:‘,RH):
```

```
# A szórás meghatározása:
with(stats):
SL:=[]:
for i from 1 to DB do SL:=[op(SL),TH[i]]od:
SD:=evalf(describe[standarddeviation](SL));
print(‘Szoras:‘,SD):
end:
```

Összefoglaló

Értekezésünkben bemutattunk egy új egyirányú függvényt, amely ütközésmentes és a tapasztalatok szerint rendelkezik lavina hatással. A függvény biztonsága az általános normaforma egyenlet megoldásának nehézségén alapul.

Legyen $P(X) \in \mathbb{Z}[X]$ egy $n \geq 3$ fokú főpolinom, amelynek nincsenek többszörös gyökei. Jelölje $\alpha_1, \ldots, \alpha_n$ a P gyökeit és legyen

$$L^{(i)}(\underline{X}) := \sum_{j=1}^{m} \alpha_i^{j-1} X_j \quad \text{ahol} \quad i = 1, \ldots, n \text{ és } m \leq n.$$

Definiáljuk a P polinomhoz tartozó normaformát az

$$\mathcal{N}_P(\underline{X}) := \prod_{i=1}^{n} L^{(i)}(\underline{X})$$

egyenlőséggel. Valójában $\mathcal{N}_P(\underline{X})$ egy speciális szétesző forma, amely egy általánosítása a normaformának.

Definiáljuk az $\mathcal{N}_P : \mathbb{Z}^m \to \mathbb{Z}$ leképezést az alábbi módon:

$$\mathcal{N}_P : (x_1, \ldots, x_m) \mapsto \mathcal{N}_P(x_1, \ldots, x_m).$$

Mivel $\mathcal{N}_P(\underline{X})$ egy egész együtthatós n-ed fokú homogén polinom, az $\mathcal{N}_P(\underline{x})$ függvényérték racionális egész lesz minden $\underline{x} \in \mathbb{Z}^m$ esetén.

Legyenek p és q olyan prímek, amikre $q > p > q/2$ és legyen $s := pq$. Definiáljuk az $\mathcal{N}_{P,s} : \mathbb{Z}_s^m \to \mathbb{Z}_s$ leképezést a következő módon:

$$\mathcal{N}_{P,s} : (x_1, ..., x_m) \mapsto \mathcal{N}_P(x_1, \ldots, x_m) \pmod{s}.$$

Az $\mathcal{N}_P(\underline{x})$ és $\mathcal{N}_{P,s}(\underline{x})$ függvényértékek kiszámítására bemutattunk három algoritmust. Elemeztük ezek bonyolultságát és összehasonlítottuk MAPLE implementációik futási idejét.

Így sikerült bizonyítanunk, hogy az $\mathcal{N}_P(\underline{x})$ és $\mathcal{N}_{P,s}(\underline{x})$ függvényértékek hatékonyan kiszámíthatók.

Jelölje $P_{coll} = P[\mathcal{N}_{P,s}(\underline{x}) = \mathcal{N}_{P,s}(\underline{y}) : \underline{x}, \underline{y} \in {\mathbb{Z}_s}^m]$ az $\mathcal{N}_{P,s}$ függvény ütközéseinek valószínűségét.

Fő eredményünk a következő két tétel.

7. Tétel. *Legyen $P(X) \in \mathbb{Z}[X]$ egy legalább harmadfokú főpolinom, amelynek nincsenek többszörös gyökei. Legyen p és q prímszámok úgy, hogy $q > p > q/2$ és $s := pq$. Tegyük fel, hogy $lnko(m, \varphi(s)) = 1$. Jelölje $N(P, b, s)$ az $N_P(x_1, \ldots, x_m) \equiv b \pmod{s}$ kongruencia megoldásainak számát.*

- *Ha $lnko(b, s) = 1$ teljesül, akkor*

$$|N(P, b, s) - s^{m-1}| < c_1(P) s^{m-1-\frac{1}{4}}$$

- *egyébként pedig*

$$N(P, b, s) < c_2(P) s^{m-1}.$$

8. Tétel. *Legyen $P(X) \in \mathbb{Z}[X]$ egy legalább harmadfokú főpolinom, amelynek nincsenek többszörös gyökei. Legyen p és q prímszámok úgy, hogy $q > p > q/2$ és $s := pq$. Tegyük fel, hogy $lnko(m, \varphi(s)) = 1$. Akkor az $\mathcal{N}_{P,s}$ függvény P_{coll} ütközési valószínűsége teljesíti a*

$$P_{coll} < \frac{C}{s},$$

egyenlőtlenséget, ahol a C konstans csak a P polinomtól függ.

Teszteltük az $\mathcal{N}_{P,s}(\underline{x})$ függvény MAPLE implementációját, és megmutattuk, hogy rendelkezik lavina hatással.

Készítettünk egy alkalmazást, amelyben ez a függvény egyirányú hash függvényként működik. Az alkalmazás tudja ellenőrizni egy számítógép felhasználóinak jelmondatait. Egy másik konkrét alkalmazás elkészíti egy üzenet hash értékét.

Így javasoltuk az $\mathcal{N}_{P,s}$ függvény - mint egyirányú hash függvény - használatát a kriptográfiai gyakorlatban.

Találtunk még további eredményeket is, amelyek azt mutatják, hogy az $\mathcal{N}_{P,s}$ függvény egy egyirányú függvény család.

Summary

In this thesis we present a new one-way function with collision resistance and avalanche effect. The security of this function based on the difficulty of solving a general norm form equation.

Let $P(X) \in \mathbb{Z}[X]$ be a monic polynomial of degree $n \geq 3$ having no multiple roots. Denote by $\alpha_1, \ldots, \alpha_n$ the roots of P and put

$$L^{(i)}(\underline{X}) := \sum_{j=1}^{m} \alpha_i^{j-1} X_j \quad \text{for} \quad i = 1, \ldots, n \text{ and } m \leq n.$$

Define the norm form corresponding to the polynomial P by

$$\mathcal{N}_P(\underline{X}) := \prod_{i=1}^{n} L^{(i)}(\underline{X}).$$

In fact, $\mathcal{N}_P(\underline{X})$ is a generalization of the concept of norm form and is a special decomposable form.

Define the mapping $\mathcal{N}_P : \mathbb{Z}^m \to \mathbb{Z}$ in the following way:

$$\mathcal{N}_P : (x_1, \ldots, x_m) \mapsto \mathcal{N}_P(x_1, \ldots, x_m).$$

Since $\mathcal{N}_P(\underline{X})$ is a homogeneous polynomial of degree n, with integer coefficients the function value $\mathcal{N}_P(\underline{x})$ will be a rational integer for every $\underline{x} \in \mathbb{Z}^m$.

Let p and q be primes such that $q > p > q/2$ and $s := pq$. Define the mapping $\mathcal{N}_{P,s} : \mathbb{Z}_s^m \to \mathbb{Z}_s$ in the following way:

$$\mathcal{N}_{P,s} : (x_1, ..., x_m) \mapsto \mathcal{N}_P(x_1, \ldots, x_m) \pmod{s}.$$

We present three algorithms for the calculation of the value $\mathcal{N}_P(\underline{x})$ and $\mathcal{N}_{P,s}(\underline{x})$. Their complexity are discussed and the running time of their imlementations in MAPLE are compared.

And so we proved that the value $\mathcal{N}_P(\underline{x})$ and $\mathcal{N}_{P,s}(\underline{x})$ can be computed efficiently.

Denote by $P_{coll} = P[\mathcal{N}_{P,s}(\underline{x}) = \mathcal{N}_{P,s}(\underline{y}) : \underline{x}, \underline{y} \in {\mathbb{Z}_s}^m]$ the probability of the collision for the function $\mathcal{N}_{P,s}$.

Our main result is the following two theorems.

Theorem 7 *Let $P(X) \in \mathbb{Z}[X]$ be a monic polynomial of degree at least 3 having no multiple roots. Let p and q be primes such that $q > p > q/2$ and $s := pq$. Suppose that $gcd(m, \varphi(s)) = 1$. Let $N(P, b, s)$ denote the number of solutions of the congruence $N_P(x_1, \ldots, x_m) \equiv b \pmod{s}$.*

- *If $gcd(b, s) = 1$ we have*

$$|N(P, b, s) - s^{m-1}| < c_1(P) s^{m-1-\frac{1}{4}}$$

- *othervise*

$$N(P, b, s) < c_2(P) s^{m-1}.$$

Theorem 8 *Let $P(X) \in \mathbb{Z}[X]$ be a monic polynomial of degree at least 3 having no multiple roots. Let p and q be primes such that $q > p > q/2$ and $s := pq$. Suppose that $gcd(m, \varphi(s)) = 1$. Then the probability of collision P_{coll} for the function $\mathcal{N}_{P,s}$ satisfies the inequality*

$$P_{coll} < \frac{C}{s},$$

where the constant C depends only on the polynomial P.

We tested a MAPLE implementation of the function $\mathcal{N}_{P,s}(\underline{x})$ and we can showed that this function has avalanche effect.

We construct an application, which uses this function as a one-way hash function. This application can check passphrase of user of a computer. An other application makes the hash value of a message.

So we propose to apply the function $\mathcal{N}_{P,s}$ in the practice of the cryptography as a one-way hash function.

Further we found also some fact, which show that $\mathcal{N}_{P,s}$ is probably a family of one-way functions.

Irodalomjegyzék

[1] M. Agrawal, N. Kayal, N. Saxena, *PRIMES is in P*, IIT Kaupur, Preprint, 2002. augusztus 6.

[2] M. Ajtai, C. Dwork, *A public-key cryptosystem with worst-case/average-case equivalence*, ECCC Report TR96-065, 1996.

[3] F. Bahr, J. Franke, T. Kleinjung, *New prime factorisation record obtained using the general number field sive*, ERCIM News No. **49** 2002.

[4] A. Baker, *Contributions to theory of diophantine equations I. and II.*, Phil. Trans. Roy. Soc. London Ser A., **263** (1968) 173-208.

[5] A. Baker, *Transcendental number theory (3rd edn.)*, Cambridge Univ. Press, Cambridge 1990.

[6] D.B. Baker, D.R. Masys, *PCASSO: a design for secure communication of personal health information via the internet*, International Journal of Medical Informatics, **54** (1999) 97-116.

[7] A.R. Bakker, *Security in medical information systems*, In: J.H. van Bemmel, A.T. McCray, eds. IMIA Yearbook of Medical Informatics, Stuttgart, New York: Schattauer Verlag, 1993, 52-60.

[8] A. Bérczes, J. Ködmön, *Methods for the calculation of values of a norm form*, Publ. Math. Debrecen, **63** (2003), 751-768.

[9] A. Bérczes, J. Ködmön, A. Pethő, *A one-way function based on norm form equations*, Periodica Math. Hungar. (megjelenés alatt).

[10] E. Biham, A. Shamir, *Differential cryptanalysis of DES-like cryptosystems*, Advances in Cryptology - CRYPTO '90 (LNCS 537), 2-21, 1991.

[11] J. Black, S. Halevi, H. Krawczyk, T. Krovetz, P. Rogaway, *UMAC: Fast and Secure Message Authetication*, In Advances in Cryptology - CRYPTO '99 (1999), vol. 1666 of Lecture Notes In Computer Science, Springer-Verlag, pp. 216-233.

[12] Z.I. Borevich and I.R. Shafarevich, *Number Theory*, Academic Press, New York, 1967, 2nd ed.

[13] J. Buchmann, *A subexponential algorithm for the determination of class groups and regulators of algebraic number fields*, Séminare de théorie des nombres, Paris (1988-1989), 28-41.

[14] J. Buchmann, S. Paulus, *A one-way function based on ideal arithmetic in number fields*, Lect. Notes Comput. Sci. **1294** (1997), 385-394.

[15] L. Cerlienco, M. Mignotte, F. Piras, *Suites récurrentes linéaires. Propriétés algébriques et arithmétiques. (Linear recurrent sequences. Algebraic and arithmetic properties)*, L'Enseign. Math. **33** (1987), 67-408.

[16] L.R. Chao, Y.C. Lin, *Associative one-way function and its significances to cryptographics*, In. J. Inf. Manage. Sci. **5** (1994), 53-59.

[17] B.W. Char, K.O. Geddes, G.H. Gonnet, B.L. Leong, M.B. Monagan and S.M. Watt, *Maple V Language reference manual*, Springer Verlag, 1990.

[18] H. Cohen, *A course in computational algebraic number theory*, Springer Verlag, 1993.

[19] J. Daemen, V. Rijmen, *The Design of Rijndael, AES-The Advanced Encryption Standard*, Springer-Verlag, 2002.

[20] A. Donaldson, *Policy for cryptography in healthcare, a view from the NHS*, International Journal of Medical Informatics, **60** (2000), 23-42.

[21] *Egészségügyi informatika*, Szerk: Kékes E., Surján Gy., Balkányi L., Kozmann Gy., Medicina Könyvkiadó, 2000.

[22] J.H. Evertse, K. Győry, *Decomposable form equations, In: New advances in transcendence theory (ed. by A. Baker)*, 175-202, Cambridge Univ Press, Cambridge 1988.

[23] H. Feistel, W.A. Notz, J.L. Smith, *Some cryptographic techniques for machine-to-machine data communications*, Proceedings of the IEEE, **63** (1975), 1545-1554.

[24] I. Gaál, *Computing power integral bases in algebraic number fields, In: Number theory: Diophantine, computational and algebraic aspects (ed. by K. Győry, A. Pethő and V.T. Sós)*, 243-254, Walter de Gruyter, Berlin 1998.

[25] M.R. Garey, D.S. Johson, *Computers and intractability: A guide to the theory of NP-completeness*, Freeman, New York, 1983.

[26] O. Goldreich, L. Levin, N. Nisan, *On constructing 1-1 one-way functions*, Electronic colloquium on computational complexity, TR-95-029, 6/25/95, 1995.

[27] S. Goldwasser, M. Bellare, *Lecture Notes on Cryptography*, MIT Press, Cambridge, Massachusetts 2001.

[28] S. Goldwasser, S. Micali, *Probabilistic encryption*, J. CSS, **28** (1984), 270-299.

[29] D. Gusfield, R.W. Irving, *The Stable Marriage Problem: Structure and Algorithms*, MIT Press, Cambridge, Massachusetts, 1989.

[30] K. Győry, *Résultats effectifs sur la représentation des entires par des formes décomposables (Queen's Papers in Pure and Appl. Math.* **56***)*, Queen's Univ., Kingston 1980.

[31] K. Győry, *On norm form, diskriminant form and index form equations, In: Topics in classical number theory(ed. by D. Halász; Colloq. Math. Soc. János Bolyai* **34***)*, 617-676, North-Holland, Amsterdam 1984.

[32] K. Győry, *On the distribution of solutions of decomposable form equations. Number theory in progress, Vol. 1 (Zakopane-Kościelisko, 1997)*, 237–265, de Gruyter, Berlin, 1999.

[33] *Handbook of medical informatics*, Eds: J.H. van Bemmel, M.A. Musen, Springer Verlag, 2002.

[34] L.A. Hemaspaandra, J. Rothe, *Creating strong, total, commutative, associative one-way funktions from any one-way function in comlexity theory*, J. Comput. Syst. Sci. **58** (1999), 648-659.

[35] D. Kahn, *The Codebreakers: The Story of Secret Writing*, Macmillan Publishing Co., 1967.

[36] J. Kam, G. Davida, *Structured design of substitution-permutation encryption networks*, IEEE Transactions on Computers, **28** (1979), 747-753.

[37] A. Karatsuba and Yu. Ofman, *Multiplication of Many-Digital Numbers by Automatic Computers*, Doklady Akad. Nauk SSSR **145** (1962), 293-294.

[38] Ködmön J., *Kriptográfia, Az informatikai biztonság alapjai, A PGP kriptorendszer használata*, ComputerBooks Könyvkiadó, 1999.

[39] Ködmön J., *Adatvédelem, adatbiztonság az egészségügyben*, Lege Artis Medicinae **4**, (1997), 278-281.

[40] Ködmön J., *Az egészségügyi adatvédelem szabályozása*, Orvosi Hetilap **140** (1999), 1113-1115.

[41] J.C. Lagarias, A.M. Odlyzko, *Solving low-density subset sum problems*, Proceedings of the 24th IEEE Symp. on the Foundations of Computer Science, 1983, 1-10.

[42] S. Lang, A. Weil, *Number of points of varieties in finite fields*, Am. J. Math. **76** (1954), 819-827.

[43] E. Lawler, J. Lenstra, A. Rinnooy Kan, D. Shmoys, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Wiley, New York, 1985.

[44] L.A. Levin, *Problems complete in 'average' instance*, Proc. 16th ACM Symp. on the Theory of Computing, 1984, 465.

[45] Y.V. Matiyasevich, *Hilbert's Tenth Problem*, MIT Press, 1993.

[46] M. Matsui, *Linear cryptanalysis method for DES cipher*, Advances in Cryptology - EUROCRYPT '93 (LNCS 765), 386-397, 1994.

[47] A.J. Menezes, P.C. van Oorschot, S. Vanstone, *Hadbook of Applied Cryptography*, CRC Press, 1997.

[48] R.C. Merkle, *A fast software one-way hash function*, J. Cryptology **3** (1990), 43-58.

[49] R.C. Merkle, M.E. Hellman, *Hiding information and signatures in trapdoor knapsacks*, IEEE Trans. on Information Theory, **24** (1978), 525-530.

[50] F. Morain, *Implementation of the Atkin-Goldwasser-Kilian primality testing algorithm*, INRIA Research Report **911**, 1988.

[51] C.H. Papadimitriou, *Computational Complexity*, Addison-Wesley Publishing Company, 1994.

[52] J. Patarin, *Secret public key schemes*, In:Public-Key Criptography and Computational Number Theory, (ed. by K. Alster, J. Urbanowicz and H. C. Williams), 221-237. Walter de Gruyter, 2001.

[53] A. Pethő, *Algebraische Algorithmen*, Vieweg Verlag, 1999.

[54] L. Rédei, *Algebra I*, Akademische Verlagsgeselschaft Geest & Portig K.-G., Leipzig, 1959.

[55] L. Rónyai, G. Ivanyos, R. Szabó, *Algoritmusok*, TYPOTEX, 1998.

[56] W.M. Schmidt, *Linearformen mit algebraischen Koeffizienten II.*, Math. Ann. **191** (1971), 1-20.

[57] W.M. Schmidt, *Diophantine Approximation*, Lecture Notes in Mathematics **785** (1980), Springer Verlag, Berlin.

[58] W.M. Schmidt, *The number of solutions of norm form equations*, Trans. Amer. Math. Soc. **317** (1990), 197-227.

[59] W.M. Schmidt, *Diophantine approximations and diophantine equations* Lecture Notes in Math. **1467** (1991), Springer, Berlin.

[60] B. Schneier, *Applied Cryptography*, John Wiley & Sons, 1996.

[61] A. Schönhage, *The fundamental theorem of algebra in terms of computational complexity (Preliminary report)*, nem publikált kézirat.

[62] A. Schönhage and V. Strassen, *Schnelle Multiplikation grosser Zahlen*, Computing **7** (1971), 281-292.

[63] A. Shamir, *A plynomial-time algorithm for breaking the basic Merkle-Hellman cryptosystem*, Proc. 23rd IEEE Symp. on the Foundations of Computer Science, 1982, 142-152.

[64] T.N. Shorey, R. Tijdeman, *Exponential diophantine equations*, Cambridge Univ. Press, Cambridge 1986.

[65] N.P. Smart, *How Difficult is it to Solve a Thue Equation?*, In ANTS-2, LNCS **1122**, 363-373, 1996.

[66] Q. Sun, *A kind of trap-door one-way function over algebraic integers*, J. Sichuan Univ., Nat. Sci. Ed. No. **2** (1986), 22-27.

[67] A. Thue, *Über Annäherungswerte algebraischer Zahlen*, J. Reine Angew. Math. **135** (1909), 284-305.

[68] N. Tzanakis, B.M.M. de Weger, *On the practical solution of the Thue equation*, J. Number Theory, **31** (1989), 99-132.

[69] A.F. Webster, S.E. Tavares, *On the design of S-boxes*, Advances in Cryptology-CRYPTO'85 (LNCS **218**), 523-534, 1986.

[70] D.L. Whiting, M.J. Sabin, *Montgomery Prime Hashing for Message Authentication*, CT-RSA 2003 (ED: M. Joye), LNCS **2612**, pp. 50-67, Springer-Verlag, 2003.

[71] G. Yuval, *How to swindle Rabin*, Cryptologia **3** (1979), 329-333.

A. Függelék

A jelölt publikációs jegyzéke

1. Ködmön József, Számítástechnika Szabolcs-Szatmár-Bereg megye iskoláiban, *AV Kommunikáció*, **5**, (1990), 170-174.

2. József Ködmön, Bildungsystem der Experten für Gesundheitswesen in Ungarn, *HvA Press*, **11**, (1993), 17-21.

3. József Ködmön, Die Funktion SOUNDEX und Algorithm der Textdistanz, *HvA Press*, **12**, (1993), 22-25.

4. Ködmön József, Adatszerkezetek (főiskolai jegyzet), *DOTE, Egészségügyi Főiskolai Kar, Nyíregyháza*, 1996.

5. Ködmön József, Adatvédelem, adatbiztonság az egészségügyben, *Lege Artis Medicinae*, **4**, (1997), 278-281.

6. Ködmön József, Az egészségügyi adatvédelem szabályozása, *Orvosi Hetilap*, **140** (1999), 1113-1115.

7. Ködmön József, Kriptográfia, Az informatikai biztonság alapjai, *ComputerBooks Könyvkiadó*, Budapest, 1999.

8. Attila Bérczes, József Ködmön, Methods for the calculation of values of a norm form, *Publ. Math. Debrecen*, **63** (2003), 751-768.

9. Attila Bérczes, József Ködmön, Attila Pethő, A one-way function based on norm form equations, *Periodica Math. Hungar.* (megjelenés alatt).

B. Függelék

A jelölt konferencia előadásainak jegyzéke

1. Az informatika oktatás aktuális gyakorlati problémái, V.Továbbképző és tudományos konferencia, Miskolc 1992.
2. Az egészségügyi informatika oktatásának gyakorlati problémái, Informatika a felsőoktatásban, Debrecen, 1993.
3. A számítástechnika helye és szerepe az egészségügyben, különös tekintettel a védőnői munkára, Regionális védőnői továbbképzés és konferencia, Nyíregyháza, 1993.
4. Feleletválasztós teszt számítógépes megvalósítása, CAI konferencia, Nyíregyháza, 1993.
5. Bildungsystem der Experten für Gesundheitswesen in Ungarn, HvA, Amsterdam, 1993.
6. Die Funktion SOUNDEX und Algorithm der Textdistanz, HvA, Amsterdam, 1993.
7. Számítógéppel segített oktatás az egészségügyben, VI.Továbbképző és tudományos konferencia, Miskolc 1994.
8. Datenschutz und Datensicherheit im Unterricht an Hochschule in Nyíregyháza, HvA, Amsterdam, 1994.
9. Elegendő-e a jelszavas védelem?, VI. Egészségügyi Informatikai Vándorgyűlés, Budapest, 1995.

10. Hálózatbiztonsági technikák az egészségügyben, Networkshop, Debrecen, 1996.

11. Az egészségügyi adatvédelem szabályozása, XXI. Neumann Kollokvium, Veszprém, 1998.

12. On the construction of a candidate one-way function, HAJDUCRYPT 2002, Central European Conference on Cryptology, Debrecen, 2002.

13. A one-way function based on norm form equations, TATRACRYPT 2003, Central European Conference on Cryptology, Bratislava, 2003.

14. Some Applications of a one-way function based on norm form equations, ICAI 2004, 6th International Conference on Applied Informatics, Eger, 2004.

Normaforma egyenletek alkalmazása a kriptográfiában

Értekezés a doktori (PhD) fokozat megszerzése érdekében a matematika tudományában.

Írta: Ködmön József okleveles matematika, ábrázoló geometria, számítástechnika szakos középiskolai tanár

Készült a Debreceni Egyetem Matematika- és számítástudományok doktori iskola
Diofantikus és konstruktív számelmélet programja keretében

Témavezető: Dr. Pethő Attila

A doktori szigorlati bizottság:

elnök: Dr. .....
tagok: Dr. .....
Dr. .....

A doktori szigorlat időpontja: 2004.....

Az értekezés bírálói:

Dr. .....
Dr. .....
Dr. .....

A bírálóbizottság:

elnök: Dr. .....
tagok: Dr. .....
Dr. .....
Dr. .....
Dr. .....

Az értekezés védésének időpontja: 2004.....