Alkalmazott Matematikai Lapok **23** (2006), 207–294.

A KRIPTOGRÁFIAI BIZTONSÁG MEGKÖZELÍTÉSI MÓDJAI – ALKALMAZÁSOK, KRIPTOGRÁFIAI STRUKTÚRÁK

PAPP PÁL, SZABÓ ISTVÁN

1. Bevezetés

Jelen Tanulmány elsősorban a kriptográfiai biztonság elérésének módjait elemzi, különös tekintettel a nyilvános kulcsú rendszerek (PKI) kriptográfiai biztonságára. Ehhez áttekintésre kerülnek:

- a biztonságot veszélyeztető tényezők, a szakirodalomból ismert támadási módszerek osztályozása;
- a kriptográfiai biztonság különböző megközelítései:
  - információelméleti megközelítések, bizonyított biztonság,
  - redukciós-bonyolultságelméleti biztonság,
  - kalkulációs biztonság,
  - a kvantumkriptográfiai eredmények hatása a kriptográfiai biztonság klasszikus eredményeire,
- a kriptográfiai rendszerek struktúrái, részosztályai:
  - kriptográfiai primitívek:
    - kulcsnélküli primitívek,
    - titkos kulcsú primitívek (folyamrejtjelzők, blokkrejtjelzők),
    - nyilvános kulcsú primitívek,
  - kriptográfiai sémák,
  - kriptográfiai protokollok,
  - kriptográfiai alkalmazások,
- kriptográfiai szabványok.

A téma rendkívül sokrétű, és szinte áttekinthetetlenül hatalmas szakirodalma van. Számtalan könyv, interneten elérhető publikáció, az új eredményeket évente áttekintő – hatalmas kutatói létszámmal megrendezett – konferencia (pl. EUROCRYPT, CRYPTO, ASISACRYPT, FSE /Fast Software Encryption/, SHARCS /Special Purpose Hardware for Attacking Cryptographic Systems/, stb. kiadványai mutatják a kriptográfia dinamikus fejlődését. Jelen áttekintés elsősorban a kriptográfiai alkalmazás rendszerszemléletét, a biztonság különböző megközelítéseit emeli ki (néhány, a konferenciákon elhangzó, nehezen hozzáférhető eredményre figyelemfelhívással). Akit a téma részletesebben érdekel, a hatalmas irodalomból külön is ajánljuk a magyar nyelven is elérhető, a klasszikus eredményeket (DES, LFSR, KnapSack, ...) áttekintő [6] könyvet, a modern biztonsági algoritmustervezés követelményeit részletező [2] könyvet, az átfogó ismeretterjesztést megvalósító [10] könyvet, illetve [5] interneten szabadon hozzáférhető könyvét: Handbook of applied cryptography, `http://www.cacr.math.uwaterloo.ca/hac/`.

Kevés olyan alkalmazotti szakterület van, ahol annyira szerteágazó matematikai részterületek eredményei kerülnek komplex alkalmazásra, mint a kriptográfia területén. Néhány matematikai terület, melynek eredményeire épít a kriptográfia: algebra (csoportelmélet, Galois-testek ...), számelmélet (prímszámelmélet, faktorizáció, kongruenciák ...), bonyolultságelmélet (NP elmélet, NPC osztályok, redukció bizonyítottan nehéz problémákra ...), valószínűségszámítás, matematikai statisztika, információelmélet (entrópia, Shannon-féle tökéletes rejtjelrendszer, véletlenszám-generálások ellenőrzése ...), számítógéptudomány (algoritmusok gyorsítása, párhuzamosítása), és így tovább.

Ugyanakkor a hatás nem egyirányú, a felvetődő gyakorlati problémák az elméleti kutatásokra is jelentős visszahatást eredményeztek, melyek alapján jelentős új matematikai eredmények születtek. Gondoljunk itt az információelméletre gyakorolt hatáson túl pl. az RSA algoritmus által indukált fejlődésre a faktorizációs módszerekben, a stream-cipherek területén bevezetett lineáris ekvivalens fogalma alapján kidolgozott új statisztikai próbákra, a kriptográfiai célú véletlenszám generálás követelményei alapján kifejlesztett statisztikai programrendszerekre,[1] továbbá az NP elmélet kriptográfiai indíttatású továbbfejlesztéseire (pl. az egyirányú függvény, keménybit, véletlentől megkülönböztethetőség NP technikát felhasználó, továbbfejlesztő bizonyításaira, melyekről részletesen olvashatnunk a [2] könyvben) vagy a kriptográfiai szakirodalomban az 1990-es évektől új kutatási irányként megjelenő (NP elméleti módszerek felhasználásával) bizonyított biztonságú primitívek gyakorlati számításigény-becslési megközelítésére, és a sor még hosszan folytatható.

A titkosírás több mint kétezer éves története során megszámlálhatatlan mennyiségű algoritmus (eljárás) került kidolgozásra és (rengeteg) feltörésre (ld. például David Kahn híres könyvét: The Coedebreakers, New York, (1996)). Ez a folyamat a legutóbbi néhány évtizedben felgyorsult (ld. pl. a hatalmas mennyiségű újkeletű

[1] (*ld. pl. NIST Special Publication 800-2: A Statistical Test Suite for Random and Pseudorandom Number Generators, May 2001;* Maurer Ueli M.: „A universal statistical test for random number generators", Journal of Cryptology, vol.5, no. 2., 1992, pp. 89–105.);

algoritmusok egy részének összefoglalását Bruce Schneier: Applied Cryptography: Protokols, Algorithms, and Source Code in C, John Wiley & Sons, New York, 2nd edition, (1996) könyvében).

Ahogy az információ-technológiai rendszerek fejlődésében is nemzetközi szinten az egységesítésre, a szabványosításra való törekvések térnyerése figyelhető meg, ugyanúgy az információ védelme, és ennek fontos részterülete, a kriptográfiai eljárások területén is – a korábbi egyedi törekvésekkel szemben – egységes követelmények, szabványos védelmi megoldások (és szabványok) jelentek meg.

A kriptográfia nem öncélú eszköz, hanem az informatikai (kommunikációs) rendszerek biztonságához tud nélkülözhetetlen eszközeivel hozzájárulni, mind a bizalmasság (Confidentiality, Secrecy) megőrzése, az integritás (Integrity) biztosítása (pl. MAC), mind a hitelesség (Authenticity) biztosítása (pl. elektronikus aláírás, hitelesítési protokollok) területén – de csak kellő erősségű, bizonyításokkal alátámasztott eljárások, rendszerek alkalmazása esetén.

2. A kriptográfia helye az információvédelemben

Az informatikai biztonság témaköre rendkívül szerteágazó terület, melyből a kriptográfia alkalmazása több kiemelt területen elősegíti az informatikai biztonság növelését.

Az informatikai biztonság két nagy ága:

- A *megbízható működés* biztosítása, mely terület többek között magában foglalja az alábbi veszélyek kivédését:
  - működési (SW, HW) hibák okozta károk;
  - a rendszer fizikai sérülései;
  - szolgáltatások megbénítása, megbénulása (pl. DOS, DDOS: Distributed Denial of Services), támadások;
  - számítási kapacitás lopása; stb.
- *Információvédelem*, melynek leggyakrabban említett részterületei:
  - bizalmasság (Confidentiality, Secrecy) megőrzése;
  - integritás (Integrity) biztosítása: a rendszerek (programok) és adatok integritásának, konzisztenciájának fenntartása;
  - hitelesség (Authenticity), beleértve az eredet (küldő), és a tartalom hitelességének /pl. elektronikus aláírással történő/ igazolását.
- Kiegészítő informatikai biztonsági *elvárás:*
  - Tranzakció utólagos *letagadhatatlanság*ának garantálása (Non-repudiation): megakadályozni a felhasználókat abban, hogy utólag letagadjanak valamilyen általuk elvégzett tevékenységet.

A fenti biztonsági elvárások teljesítéséhez a kriptográfiai eljárások több területen nyújthatnak magas színvonalú, (környezeti feltételek biztosítása mellett) bizonyítható biztonságú módszereket, például a bizalmasság megőrzése területén, de ezen kívül a hitelesség, integritás megőrzése, a rendszerekhez való illetéktelen hozzáférés kizárása (ld. jelszóképek egyirányú függvények segítségével való tárolása, egyszeri jelszóképzések, stb.) területein is.

A kriptográfia alkalmazásával kapcsolatos szabályozás nemzetközi szinten nem egységes. A legtöbb országban, ahol a kriptográfiával törvényi szinten is foglalkoznak, a kriptográfiai jogalkotás többnyire csak a titkosításra használt kriptográfiára vonatkozik (a sértetlenség, hitelesség, letagadhatatlanság céljából használt kriptográfiára nem). Magyarországon a titkosítással kapcsolatos jogszabályok hatóköre csak az ún. minősített információk (állam- és szolgálati titok) védelmére vonatkozó hatósági felügyeletet szabályozzák. A hitelesség kérdéseivel – az elektronikus aláírások szabályozásaival – külön jogszabályok foglalkoznak.

Korábban a kriptográfiai eszközök exportját a Wassenaar Egyezmény (Wassenaar Arrangement – WA) értelmében 32 ország együttesen korlátozta. Néhány évtizede a COCOM (Coordinating Committee for Multilateral Export Controls) egy nemzetközi szervezet volt, amely a tagországok stratégiai termékeinek és technikai adatainak közös szabályozását, védelmét biztosította meghatározott célországok vonatkozásában.

A Wassenaar Megállapodást 1998 decemberében felülvizsgálták, könnyítéseket vezettek be. Ennek alapján a következő termékek szabadon exportálhatók lettek: az összes szimmetrikus kriptográfiai termék 56 bit-ig, az összes aszimmetrikus kriptográfiai termék 512 bit-ig és az összes alcsoporton alapuló kriptográfiai termék (beleértve az elliptikus görbét) 112 bit-ig.

A 2000. november 30-án megtartott ülés alkalmával a Wassenaar egyezményhez csatlakozott államok *feloldották* az 56 bit-es export szabályozási korlátot a tömegpiaci kriptográfiai szoftverek és hardverek vonatkozásában.

Ma elfogadott – sőt követelmény – a kriptográfiai megoldások alkalmazása, szabványosítása – így nyilvánossága – az információvédelem több területén: a bizalmasság és a hitelesség biztosítása, a sértetlenség segítése, az utólagos letagadhatatlanság garantálása területén.

A kriptográfia (alapvetően elméleti matematikára támaszkodó) eredményei döntő hozzájárulást adhatnak az informatikai rendszerek biztonságának garantálásához. Ugyanakkor az informatikai rendszerek biztonsága (benne a kriptográfiai eljárások biztonsága is) több más szakterülethez kapcsolódik. Ilyen, a matematikai diszciplínától eltérő területek:

- *jogtudomány* (ld. jogszabályi követelmények az adatvédelemre, konkrét matematikai eljárások megjelenése jogszabályokban – pl. az elektronikus aláírási törvény, illetve végrehajtási rendeletei, ahol meghatározták a jogilag releváns aláíró algoritmusokat, ajánlásokat fogalmaztak meg paraméterválasztásra);
- *minőségbiztosítási, szervezetirányítási szakterületek:* pl. termékek és folyamatok minőségbiztosítása, informatikai rendszereket üzemeltető szervezetek mű-

ködtetése, szabályozása, emberi tényezők kezelése: a legbiztonságosabb rendszerek is veszélybe kerülhetnek emberi mulasztásokból, ezt használja ki az ún. „social engineering" féle támadás: ld. pl. Kevin Mitnick: „A legendás hacker" című könyvét.

A kriptográfiai rendszerek alapját a kriptográfiai algoritmusok, alapelemek biztosítják, melyek egzakt leírását a szakirodalomban ***kriptográfiai primitívek***nek nevezik. Erre épülnek a ***kriptográfiai sémák***, azokra a ***kriptográfiai protokollok***. Ezeket használják fel a ***kriptográfiai alkalmazások***ban, melyeket komplex informatikai rendszerekben működtetnek. A biztonsági elvárások teljesüléséhez minden szinten biztosítani kell a követelményeket teljesítő biztonságos eljárásokat, valamint ezek konzisztenciáját (későbbiekben példát látunk arra, hogy erős titkosítás, jó kriptográfiai protokoll mellett is gyenge konstrukciót valósíthat meg, ld. 10.1. fejezet).

3. A szteganográfia fogalmai, osztályozásai, felhasználási területei

A bizalmasság biztosítása a kommunikáció során két – módszereiben elkülönülő – módon (illetve ezek együttes alkalmazásával) is biztosítható /kellő erősségű eljárásokkal/:

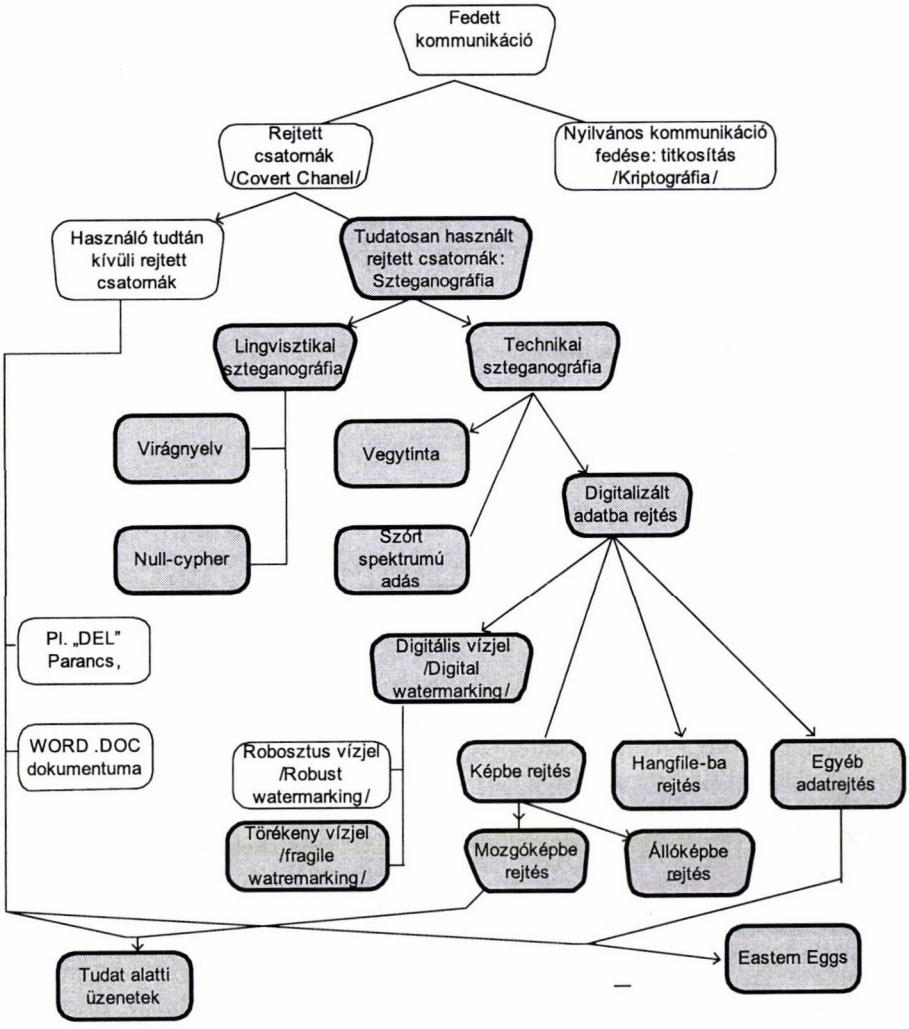
- a kommunikáció tényének /létének/ elfedésével,
- a kommunikáció tartalmának elfedésével.

Az első eljárást a szakirodalom a szteganográfiai módszerekhez, míg a másodikat kriptográfiai módszerekhez sorolja.

Jelen tanulmány tárgya elsősorban a kriptográfia, de a bizalmasság biztosításában a görögöktől napjainkig (sőt ma ismét egyre növekvő arányban) jelentős szerepe volt (van és lesz) a szteganográfiának. A szteganográfia rejtett írást jelent. A kommunikáció védelme tárgyában olyan kommunikációt értenek alatta, amikor a kommunikáció (információ közlés) ténye is rejtve marad egy illetéktelen figyelő előtt. Sok esetben ez is a biztonságot növelő fontos szempont. (Szokás még a „DATA HIDING", katonai szakirodalomban a „TRANSEC" /transmission security/ kifejezések analóg használata.)

A szteganográfia két nagy ága: a lingvisztikai szteganográfia és a technikai szteganográfia (mely utóbbiak közé tartozik például az ún. „szórt spektrumú" adás, amikor a kommunikáció – illetékteleneknek – „fehérzajként" fogható, de adó-vevő oldal által ismert eljárással az üzenet kivehető a zajból, vagy amikor digitalizált adatfolyamba – pl. képekbe – illesztik a védendő üzeneteket).

Az 1. ábra összefoglalja a szteganográfia néhány nagyobb csoportját (terjedelmi okokból a teljesség igénye nélkül):


Fedett kommunikáció
Rejtett csatornák /Covert Chanel/
Nyilvános kommunikáció fedése: titkosítás /Kriptográfia/
Használó tudtán kívüli rejtett csatornák
Tudatosan használt rejtett csatornák: Szteganográfia
Lingvisztikai szteganográfia
Technikai szteganográfia
Virágnyelv
Vegytinta
Digitalizált adatba rejtés
Null-cypher
Szórt spektrumú adás
Pl. „DEL" Parancs,
Digitális vízjel /Digital watermarking/
WORD .DOC dokumentuma
Robosztus vízjel /Robust watermarking/
Képbe rejtés
Hangfile-ba rejtés
Egyéb adatrejtés
Törékeny vízjel /fragile watremarking/
Mozgóképbe rejtés
Állóképbe rejtés
Tudat alatti üzenetek
Eastern Eggs

1. ábra

4. A kriptográfia fogalmai, osztályozásai, felhasználási területei

A **kriptológia** két nagy ága: a **kriptográfia**, mely alatt elsősorban a rejtjelrendszerek tervezését és használatát értik; míg **kriptoanalízis** alatt egyrészt a rejtjelrendszerek erősségének vizsgálatát, másrészt az illetéktelen támadó módszereinek tárházát, amely a rejtjeles üzenetek feltörésével foglalkozik, vagyis azzal, hogy hogyan lehet visszanyerni a nyílt szöveget a megfelelő kulcs ismerete nélkül. (A következőkben a szakirodalomban is összevontan használt fogalomként kriptográfiáról beszélünk.)

A kriptográfiában az alábbi fontosabb elnevezések használatosak:

- a védendő üzenet: nyílt szöveg /**Plaintext**/;
- az üzenet tartalmának az olyan kódolása, transzformálása, amely elrejti annak tartalmát a kívülállók elől: rejtjelzés /**Encryption**/;
- a transzformálás eljárása: rejtjelző algoritmus;
- míg ennek eredménye, a transzformált üzenet: rejtjeles szöveg /**Ciphertext**/;
- az a folyamat, amelynek során a nyílt szöveget visszanyerjük a rejtjelesből: megoldás /**Decryption**/;
- a rejtjelzés során általában egy kulcsot /**Key**/ használnak úgy, hogy a megoldást a címzett a kulcs ismeretében elvégezheti.

Az algoritmusok általában nyilvánosak, sőt a legújabb algoritmusok esetén (pl. AES, NESSI projektek) az algoritmusok tervezési, tesztelési szempontjai is nyilvánosak. Elsődleges cél – ez csak kellő erősségű algoritmus és a rendszerkörnyezetre vonatkozó feltételek biztosítása esetén teljesül –, hogy a kulcs ismeretének hiányában illetéktelen fél a nyílt szöveget ne tudja visszaállítani.

A kriptográfia biztonsági mechanizmusokat biztosít a biztonságos üzenetküldési, hitelesítési, a digitális aláírási és egyéb információvédelmi problémák megoldásához is.

A kulcsokat felhasználó kriptográfiai algoritmusok két nagyobb csoportba sorolhatók:

- a szimmetrikus (titkos) kulcsú és
- az aszimmetrikus (nyilvános) kulcsú algoritmusok.

A **szimmetrikus kulcsú algoritmusok** ugyanazt a kulcsot használják a rejtjelzés és a megoldás során (vagy a megoldó /dekódoló/ kulcs könnyen származtatható a rejtjel kulcsból).

Az **aszimmetrikus kulcsú** (vagy nyilvános kulcsú: PKI) algoritmusoknál kicsit félreérthető a „nyilvános" kulcsú megjelölés, mivel külön kulcsot használnak a rejtjelzéshez és a megoldáshoz, és csak a rejtjelző kulcs nyilvános, míg a megoldó kulcs számítása a rejtjelző kulcsból – jó rendszerek esetén – nem kivitelezhető.

Általában a szimmetrikus kulcsú algoritmusok sokkal gyorsabbak, mint az aszimmetrikus kulcsú algoritmusok, ezért a gyakorlatban ezeket gyakran együttesen használják úgy, hogy egy nyilvános kulcsú algoritmussal egyeztetnek (viszonylag rövid) titkos rejtjelkulcsot a hosszabb üzenet – szimmetrikus kulcsú, gyors – rejtjelzéséhez.

5. A kriptográfiai rendszerek struktúrái, részosztályai

A kriptográfiai rendszerek sokféle célból (különböző elvárások teljesítésére) számos algoritmust, algoritmusok együttesét alkalmazzák, melyek egy részének biztonsága (és tervezési elve) közös matematikai problémákra vezethető vissza.

A tervezés során biztonsági szempontból megkülönböztethetők

- intuitív módszerek (legyen minél „bonyolultabb");
- jól struktúrált, klasszikus matematikai problémákra visszavezethető algoritmusok.

Az első típusra példák a DES (ahol utólag dolgoztak ki tervezési elveket, pl. a véletlen permutációk, S-dobozok tervezésére), vagy AES blokkrejtjelzők.

A második típus inkább a nyilvános kulcsú rendszerekre jellemző, a felhasznált klasszikus matematikai problémák, melyekre próbálják visszavezetni a kriptográfiai biztonságot: a faktorizáció (IFP: Integer factorization Problem), diszkrét logaritmus probléma (DLP), hátizsák probléma (KSP), stb., melyekről részletesebben szólunk majd a kriptográfiai rendszerek biztonságával foglakozó fejezetben.

A kriptográfiai rendszerek alapját a kriptográfiai algoritmusok biztosítják, melyek egzakt leírását a szakirodalomban *kriptográfiai primitívek*nek nevezik, a kriptográfiai primitívek tipikus – klasszikus matematikai problémákkal kapcsolatos – *kriptográfiai függvények*et használhatnak fel (ld. lentebb).

A kriptográfiai primitívekre épülnek a *kriptográfiai sémák*, azokra a *kriptográfiai protokollok*. Ezeket használják fel a *kriptográfiai alkalmazások*ban, melyeket komplex informatikai rendszerekben működtetnek. A biztonsági elvárások teljesüléséhez minden szinten biztosítani kell a követelményeknek biztonságos eljárásokat, valamint ezek konzisztenciáját.

A kriptográfiai rendszer hierarchiáját mutatja az alábbi összegzés (melynek részletezése későbbi fejezet tárgya):

- a kriptográfiai primitívek (kriptográfiai algoritmusok),
  melyek tipikus – klasszikus matematikai problémákkal kapcsolatos – kriptográfiai függvényeket használhatnak fel (pl. véletlenszám generálás, véletlen permutációk alkalmazása, véges testbeli műveletek ...).
- kriptográfiai sémák,
  melyek a primitívek kritográfiai alkalmazását vezérlik:
  - meghatározzák, hogyan bontsuk blokkokra a rejtjelzendő üzenetet;
  - hogyan kell kiegészíteni a csonka blokkokat;
  - hogyan kapcsolódjanak egymáshoz a blokkok az üzenet rejtjelzése folyamán;
  - hogyan alakítsuk ki a rejtjelzéshez a kulcsokat (a kulcsok egyeztetése már a protokollok területe);
- kriptográfiai protokollok:

  melyet a partnerek közötti kapcsolat határoz meg. A protokoll a résztvevők közötti egyértelműen meghatározott lépések sorozata, amely két, vagy több

résztvevő között zajlik le a biztonsági elvárások maradéktalan teljesülésének érdekében.

A protokolloknak illeszkedniük kell a kommunikációs protokollokhoz. Alkalmazásukat meghatározzák a biztonsági elvárások: például a kulcskialakítási és szétosztási lehetőségek és követelmények (ld. kulcscsere protokollok), vagy speciális elvárások (mint a partnerhitelesítési, résztvevői kiegészítő védelmi elvárások, pl. zero-knowledge, secret sharing ...).

- kriptográfiai alkalmazások:

  A kriptográfiai rendszer fenti elemei nem öncélúak, hanem egy általános informatikai alkalmazást segítő komplex feladat megoldásához szükséges elemek.

  Komplex kriptográfiai rendszert valósít meg egy védett levelező rendszer, a mobil kommunikáció magánszférába tartozó (privacy) információinak védelmét megoldó GSM rendszer kriptográfiai alrendszere vagy az elektronikus fizetés védelmére kidolgozott SET (Secure Electronic Transaction) ...

6. A kriptográfiai biztonságot veszélyeztető tényezők, a főbb támadási módszerek osztályozása

Az alábbi osztályozás természetesen nem diszjunkt csoportokra bontja a lehetséges eseteket, ráadásul nem is törekedhet teljességre az esetszámok hihetetlen nagy száma (sok esetben ismeretlen alesetek) miatt sem, csak a legfőbb típusokat szándékoztunk kiemelni, amelyeket a védelmi módszerek tervezésénél szükséges figyelembe venni.

6.1. A támadás megcélzott eredménye szerint (cél lehet):

- a rejtjelzett üzenet visszaállítása;
- megoldókulcs megszerzése;
- hamisítás elkövetése (pl. üzenetblokkok kihagyása, felcserélése, módosítása ...), elektronikus aláírás hamisítása;
- véletlentől való megkülönböztetés felismerése /distinguish attack/, mely egy algoritmikus támadás kiindulópontja lehet;
- forgalomanalízisból értékes információk szerzése ...

6.2. A támadó aktivitása szerint

Passzív módszerek (passive attack):

Amikor a támadó „csak" a hozzáférhető titkosított szöveget analizálja (interception, eavesdropping, wiretapping /pl. switch-eknél/), megfigyeli a kommunikációs – nyílt – csatornát vagy elektronikusan tárolt rejtjeles szöveget. Beletartozik a lehetőségébe:

- a nyílt szöveg valamilyen *a*-priori feltételezése, kipróbálása (azaz összetartozó nyílt-rejtjeles pár/ok/ vizsgálata), így az összetartozó nyílt-rejtjeles párokból próbálja a támadó meghatározni a titkos kulcsot, mellyel azután más rejtjelzett szövegek védett üzenetét is visszaállíthatja;
- known-key attack: néhány kulcselőzményből következtetnek a következő (új) kulcsra.

Hasonlóan ide tartoznak a lehallgatás egyéb módszerei, pl. forgalomanalizálás, az eszközök működése közbeni elektromágneses kisugárzás vételéből származó technikai információk felhasználása (melyekhez nem kell „hozzányúlni" a kriptográfiai algoritmust végrehajtó hardverhez, távolból lehet levenni, analizálni a támadáshoz szükséges információkat).

Aktív módszerek (active attack):

Amikor a támadó megkísérelheti a rejtjelszöveget módosítani (törölni, hozzáírni, blokksorrendet változtatni), rákényszeríteni a legális alkalmazót valamilyen számára nem tervezett műveletre (valamilyen szöveget titkosíttatni/aláíratni), a kommunikációba harmadik félként belépni (ld. pl. az „*intruder in the middle attack*", vagy ennek speciális esetét a „*grandmaster chess*" támadást ...).

Tipikus példái ennek az úgynevezett visszajátszás (*replay attack*), megszemélyesítés (*impersonation attack*), összefésülés (*interleaving attack*) ...

Beletartozik a „fegyvertárba" valamilyen „side information" aktív megszerzése (pl. a titkosító kulcs megszerzése, részinformáció szerzése a titkosítás folyamatából az eszköz aktív támadásával, pl. *power analysis attack*, *timing analysis attack* ...).

Ilyen támadásokat tipikusan Smart Cardokra dolgoztak ki, amikor a kártyán külön van a processzor és ennek tápellátása, s a kettő közötti vezetéken mérhetők a fizikai jelek. Ilyen lehetőségeket bizonyítottak és prezentáltak például az ASIACRYPT 2000 konferencián Mehdi-Laurent Akkar, Régis Bevan, Paul Dischamp, and Didier Moyart „Power Analysis, What Is Now Possible" c. előadásukban (megjelent Springer-Verlag kiadónál).

A DES chipek energiafelvételből történő támadását is részletesen elemezték több kutatásban, például a „Public Key Cryptography: 5th International Workshop on Practice and Theory in Public Key Cryptosystems, Paris, France, February 2002. konferencián hangzott el „**Differential Power Analysis**" címmel Paul Kocher, Joshua Jaffe, and Benjamin Jun előadása, melyben ábrákkal szemléltették a power analysis attack működését a DES működése során. Az RSA hasonló támadásáról később lesz szó.

Másfajta aktív támadást mutatott be a **CRYPTO 2000 konferencián**, „*Differential Fault Attacks* on Elliptic Curve Cryptosystems" címmel, ahol még bon-

tásvédelemmel ellátott eszközökben is hibabit beszúrásával (bejuttatásával) olyan működési hibát lehetett előidézni, mely az elliptikus görbéken alapuló kriptorendszer támadását teszi lehetővé. (A módszer a korábban RSA-ra kidolgozott „differential fault attack" továbbfejlesztése: ld. pl. D. Boneh, R.A. DeMillo, and R.J. Lipton: On the Importance of Checking Cryptographic Protocols for Faults, Lectures Notes of Computer Science 1233, Proceedings of EUROCRYPT'97, Springer, pp. 37–51.)

6.3. Támadási technikák szerint

A támadási technikákat is sokféleképp lehet csoportosítani.

Algoritmikus:

- Ezen belül számítási „erőfölényt" kihasználó (ld. „*brute force attack*", „*exhaustive attack*" pl. a DES-re: EFF Cracking DES projekt /The Electronic Frontier Foundation projekt, ld.: `http://www.eff.org/descracker/` /.
- A rendszerek strukturális összefüggéseit kihasználó, matematikai eszköztárat igénybevevő támadások (statisztikai, algebrai, számelméleti ... módszerekkel)

A felhasználó közreműködését kiváltó támadások:

- A támadót ráveszik, valamilyen üzenet rejtjelzésére/aláírására, melyből a támadó sikeres támadást hajthat végre (pl. az I. világháború előtt az osztrák-magyar monarchia rejtjelfejtői úgy fejtették meg a bevezetett olasz katonai kódkönyvet, hogy érdeklődésre számot tartó „fal" üzenetet jelentettek meg egy konstantinápolyi újságban, melyet az olasz katonai attasé rejtjelezve hazaküldött). Ennek egyik mai alkalmazása, amikor elektronikus aláírási rendszerekben egy „közjegyzőt" rávesznek tetszőleges üzenet aláírására, mely után hamisítani lehet egy másik üzenet aláírását.
- „Social engineering attack" támadások, amikor a felhasználót a támadó ráveszi valamilyen, a rendszer biztonságát gyengítő tevékenységre (Nemrég jelent meg magyarul Kevin Mitnick – A legendás hacker c. könyve, melyben sok példa olvasható, amikor a befolyásolás és rábeszélés eszközével megtévesztik a felhasználókat, meggyőzik őket, a rendszer gyengítését okozó tevékenységek végzésére).

Fizikai eszközt igénybevevő aktív támadási módszerek:

pl. üzenetátirányítás, hamisítás, intruder in the middle attack, power analysis attack, timing analysis attack ...

6.4. A támadás inputja szerint

a) Pusztán rejtjelszövegből végrehajtott támadás /***Ciphertext only attack***/

Klasszikus passzív támadás:
itt a támadó ismerheti a nyílt szöveg statisztikai tulajdonságait (ezzel ellenőrzi a támadás sikerességét), pl.

- a nyelvszerű statisztikát;
- vagy a DES FFT támadásánál csak azt feltételezik, hogy a nyelvszöveg rejtjelzett karakterei az összes lehetséges byte érték kb. negyedét vehetik fel (kis- és nagybetűk, számok, írásjelek);
- vagy esetleg kihasználhatják, ha azonos nyílt blokkok kerültek – különböző kulcsokkal, vagy eltérő nyílt szöveg azonos kulcsokkal rejtjelzésre, pl. OTP (One-time-pad rejtjelzés) esetén, illetve egyes protokollok támadása esetén, ahol sok üzenethez azonos protokollépesek rejtjelződnek le, stb.

b) Nyílt szövegből /nyílt töredékből/ végrehajtott támadás /***plaintext and corresponding ciphertext attack***/.

Itt a támadó feltételezhet nyílt szöveget:

- Pl. a file rejtjelzése során az utolsó blokk vége 8 db. H00 érték, hasonló feltételezés működik Hellman első DES fejtő ötleténél,
- vagy JPEG képek első blokkja jellegzetes struktúrájú,
- az EXE file-ok első két byte-ja: „MZ", azaz H4D, H5A, decimálisan: 77,90, stb.

Érdekes side information attackról szól John Kelsey: Compression and Information Leakage of Plaintext c. cikke, mely elhangzott a Fast Software Encryption 9th International Workshop, FSE 2002 konferencián. Ebben a tömörített adat (az input és output méretének eltérése) tömörítési aránya ad side-információt.

(Az inputról valami információ kiszivárgása valószínűbb, mint a kulcsról.

Például, ha tudjuk, hogy egy 1 MB-os file 1 KB-osra tömörödött, tudjuk, hogy nagyon redundáns volt.

Triviális, ha van néhány valószínűsített üzenet, ezek között a támadó tud választani.)

c) Választott rejtjelszövegű – és hozzátartozó nyílt szövegű – támadás /***Selected ciphertext and corresponding plaintext attack***/. Ide tartozó gyakorlati módszer pl. a *differential cryptoanalysis.*

Ennek alesete az adaptív választott rejtjelszövegű támadás, amikor a menetközbeni eredmények szerint választják a következő rejtjeles blokkot. /*Adaptive chosen-ciphertext attack*/

Példa erre Lars R. Knudsen and John Erik Mathiassen „A *Chosen-Plaintext Linear Attack* on DES" c. cikke, mely elhangzott az EUROCRYPT 2001 konferencián.

Speciális algoritmusokra egyre újabb módszereket dolgoznak ki, példa erre John Kelsey, Tadayoshi Kohno, and Bruce Schneier *Amplified Boomerang Attacks* Against Reduced-Round MARS and Serpent, EUROCRYPT 2001-es konferencián elhangzott előadása. A bumeráng attack egy adaptív választott nyílt-rejtjelszövegpár alapú támadás.

További algoritmusra Jongsung Kim, Dukjae Moon, Wonil Lee, Seokhie Hong, Sangjin Lee, and Seokwon Jung szerzőktől az „Amplified Boomerang Attack against Reduced-Round SHACAL" c., ASIACRYPT 2002-es konferencián elhangzott ismertetőt említjük.
Figyelemre méltó speciális vizsgálati módszereket fejlesztettek ki például a Clinton adminisztráció által 1993 áprilisában javasolt Skipjack algoritmusra, melynek nyilvánosságra kerülése után ezt tudományos körökben is intenzíven analizálták. Új módszerrel való támadás hangzott el a Fast Software Encryption 9th International Workshop, FSE 2002 konferencián „Saturation Attacks on Reduced Round Skipjack" címmel Kyungdeok Hwang, Wonil Lee, Sungjae Lee, Sangjin Lee, and Jongin Lim szerzőktől. Szintén ennek vizsgálatáról hangzott el előadás „Flaws in Differential Cryptanalysis of Skipjack" címmel Louis Granboulan-től az FSE 2001 konferencián. A 2002-es Fast Software Encryption 9th International Workshop konferencián „New Results on Boomerang and *Rectangle Attacks*" címmel Eli Biham, Orr Dunkelman, and Nathan Keller szerzőktől hangzott el előadás a differenciális és lineáris kriptoanalízis kombinációs támadásáról, ahol a differenciák lineárisan közelíthetők.

d) Választott nyílt – és hozzátartozó rejtjeles – szövegű támadás /***selected plaintext and corresponding ciphertext attack***/. (Például a blind signature eljárást lehet így hatékonyan támadni.)
Ennek a módszernek alesete az adaptív választott nyíltszövegű támadás, amikor a menet közbeni eredmények szerint választják a következő nyílt blokkot. /***Adaptive chosen-plaintext attack***/

e) A ***rejtjelrendszerről nyilvánosságra került információkból*** történő támadás.
Erre jó példa az RSA modulus faktorizálása a nyilvános kulcsokból. Ehhez nem kell rejtjelszöveg, viszont pl. sikeres faktorizálás esetén az RSA alapú rejtjelzés/aláírás fejthető/hamisítható. (Hasonló nyilvános információkból történő támadás más PKI rendszereknél is elvileg lehetséges.)

6.5. A támadás célpontja szerint

- Kriptográfiai algoritmus feltörése, kulcs megszerzése.
- Kriptográfia protokoll támadása.
- Kriptográfiai rendszer támadása (kulcsmenedzsment, kulcsfelhasználás, algoritmus felhívása ...).

(Pl. Bruce Schneier, Adam Shostack „Breaking Up Is Hard To Do: Modeling Security Threats for Smart Cards, `http://www.counterpane.com/smart-card-threats.html`) című cikkben külön elemzi az egyes – rosszindulatú – résztvevők lehetséges támadásait SmartCard-okat használó rendszerekben a többi résztvevővel szemben, kiemelve az alábbi kockázatokat:

- Jogosult résztvevők támadásai más résztvevőkkel szemben:
  - terminál tulajdonos támadásai
    - a kártyabirtokos vagy adattulajdonos ellen;
    - a kártyakibocsátó ellen;
  - a kártyabirtokos támadásai
    - a terminál ellen;
    - az adattulajdonos ellen;
    - a kártya kibocsátója ellen;
    - a szoftvergyártó ellen;
  - a kártyakibocsátó támadásai a kártyabirtokos ellen;
  - a gyártó támadásai az adatok tulajdonosa ellen.
- Nem jogosult résztvevők támadásai más résztvevőkkel szemben:
  - kívülállók támadásai lopott kártyákkal.
- Kooperatív támadások.

Vizsgálják különböző kriptográfiai sémák viselkedését véletlenszerű hardver hibák kihasználása szempontjából (látens hibák, tranziens hibák, indukált /gerjesztett/ hibák), például ilyen vizsgálat szerepel D. Boneh, R. A. DeMillo, R. J. Lipton, On the Importance of Eliminating Errors in Cryptograpic Computations cikkében.

Röviden, csak a lényeget kiemelve elmondhatjuk, hogy a titkos kulcsú algoritmusokkal szemben az alábbi minimálisan elvárt követelménycsoportokat emelik ki, melyek teljesítése elengedhetetlen, ugyanakkor ezek még nem biztosítják pl. a kalkulációs biztonságot. Ilyen minimális követelmények (a fogalmak pontosítása nélkül):

Bitsoros algoritmusokkal szembeni követelmények:

- legyen hosszú periódus, nagy kulcstér;
- legyen a generált sorozat (bit, byte ...) egyenletes;
- lineáris és ugrás-komplexitása véletlenszerű legyen;
- Lempel–Ziv komplexitása véletlenszerű legyen;
- a generált sorozat differenciasorozata egyenletes legyen;
- állapottér elemei közötti korrelációk véletlenszerűek legyenek.

Blokkos algoritmusokkal szembeni követelmények, alaptámadási módok:

- Elegendően *nagy kulcstér:* a kulcstér teljes kipróbálása ellen („brute force attack").
- *Lavinahatás* teljesülése: hiba, ha a bemenet csak kis mértékben változtatja meg a kimenetet, elég valamilyen közelítő bemenet megtalálása, mind a kulcs, mind a nyílt bemenet lavinahatása szükséges).
- *Statisztikai egyenletesség,* statisztikai összefüggések kizárása:
  Statisztikai egyenetlenségek strukturális hibákra mutatnak, melyeket esetleg ki lehet használni. Az egyenetlenség csökkenti a keresett ismeretlen entrópiáját, így csökkenti a sikeres teljes kipróbálás esetszámának várható értékét.

- *Lineáris kriptoanalízissel szembeni ellenállóképesség:*
  A kódoló leképezések által meghatározott ANF (algebrai normál forma) egyenletek közelíthetőek lineáris formulával, az így kapott közelítő lineáris egyenletrendszerek Gauss-eliminációval könnyen megoldhatók. Sok ilyen lineáris egyenletet megoldva közelítik az ismeretlen kulcsokat.
- *Differenciál kriptoanalízissel szembeni ellenállóképesség:*
  Input-output közötti differencia-sorozatok egyenetlenségéből a felhasznált kulcsokra lehet következtetni (szűkíteni).
- *Találkozunk középen támadás kivédése:*
  Ismert bemenet és kimenet-pár esetén véletlenszerű kulcsokkal rejtjelezve a bemenetet és véletlenszerű kulcsokkal megoldva a kimenetet a két halmaz közös elemeit keressük; ha találunk, akkor az ismeretlen kulcsú egyszeri rejtjelzést a két ismert kulcsú rejtjelzés/megoldás kompozíciójára vezettük vissza.
- *Láncolt alkalmazásnál ne legyen rövid periódus* (pl. CBC üzemmódú blokkrejtjelző csupa nulla sorozatra).
  Stream cipher alkalmazásnál ez a tulajdonság mint kulcsismétlés közvetlenül támadható.

A titkos kulcsú algoritmusokkal szemben támasztott, fent nagyon vázlatosan ismertetett minimálisan elvárt követelménycsoportok teljesítése elengedhetetlen, ugyanakkor ezek még teljes körűen nem garantálják a biztonságot, melynek sok megközelítése van (ld. később).

7. Kriptográfiai rendszerek elemei

7.1. Kriptográfiai primitívek

A kriptográfiai primitívek meghatározása elemeik pontos meghatározását jelenti:

- jelölések és *input* pontos meghatározása (kulccsal rendelkező primitívek esetén a nyílt bemenő adatok és a kulcs /plain text- key/ formátumának, méretválasztékának megadása);
- *output* pontos meghatározása;
- *algoritmus* egzakt leírása (teszteredményekkel adott input-output párokra);
- esetleges *feltételezések* megadása.

A kriptográfiai primitívek három nagyobb osztálya:

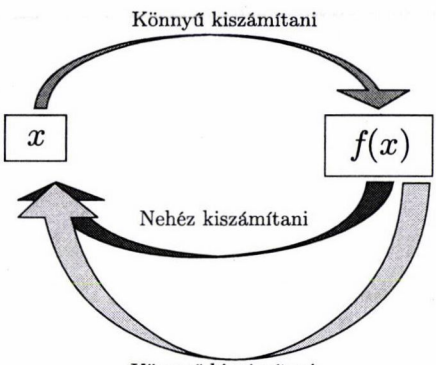
- kulcsnélküli primitívek;
- titkos kulcsú primitívek;
- nyilvános kulcsú primitívek.

7.1.1. Kulcsnélküli primitívek

7.1.1.a. Egyirányú leképezés

A leképezés nyilvános, mindenki megvalósíthatja, és könnyű kiszámítani (gyorsan: a bemenő paraméterek méretének függvényében polinomiális időben számítható), viszont az inverz leképezésre nem ismert polinomiális idejű algoritmus (ld. pontosabban a 8.2. fejezetben).

Csapda egyirányú irányú leképezés (trapdoor one way function)



A fordított irányú leképezés egy rejtett (*magán*) kiegészítő információ birtokában hatékonyan számítható. (Ld. például az RSA nyilvános kulcsú rendszerben a két prím, melyből számítható a visszafelé irány, vagy a hátizsák algoritmusoknál a csapda információt egy szupernövekedő sorozat jelenti.)
Jelentős az egyirányú leképezések felhasználása a jelszóképzések területén. Jelszóképzésre vonatkozó legfontosabb szabványok:
FIPS 112, Password Usage May 1985.
FIPS 181, Automated Password Generator October 1993.

7.1.1.b. Hash függvények

A (bináris) hash függvények: $M \to H(M)/\{0,1\}^n \to \{0,1\}^m$ leképezést valósítanak meg egy tetszőleges n ($\geq n_0$) hosszúságú bináris sorozatot egy rögzített m ($\geq m_0$) hosszúságú sorozatra képezve le. Ennek felhasználási területe lehet az üzenet integritásának (sértetlenségének) igazolása, ilyen értelemben a hibajelző kódok egy változatának is tekinthetők (pl. CRC: Cyclic Redundancy Check), mely kódok megfelelőek voltak a kommunikáció /tárolás/ véletlen hibáinak kimutatására, azonban a szándékos módosítások okozta integritás-megsértést nem tudják kivédeni. Egyik legfontosabb felhasználási területe az elektronikus aláírási rendszerekben van, ahol nem a teljes dokumentumot írják alá (aláírási primitívvel), hanem csak egy lenyomatát, amit hash függvénnyel állítanak elő. Ehhez elengedhetetlen, hogy adott lenyomathoz ne lehessen másik olyan dokumentumot találni, amely azonos lenyomatot (így azonos aláírást) képez. Ilyen tulajdonságokkal a hagyományos hibajavító kódolók nem rendelkeznek (mint alább látni fogjuk, nem is egyszerű megfelelő hash függvényt konstruálni).

Hash függvényekre nagyon sok algoritmust javasoltak, ilyenek például: az MD család (Message Digest rövidítéséből: MD2, MD4, MD5), az SHA család (SHA-0,

SHA-1, SHA-256, HSA-384, SHA-512), RIPMD-160, és egyebek (HAVAL, N-HAsh, Snefru, Tiger, Whirpool).

Szabvány leírás Hash függvényekre:

FIPS 180-2, Secure Hash Standard (SHS), August 2002. (Ebben definiálták az SHA-1, SHA-256, SHA-384 és SHA-512, valamint 2004. február 25-én kiegészítették az SHA-224-gyel.)

7.1.1.c. Kriptográfiai célú véletlenszám generálás

A kriptográfiai véletlenszám generátorok kriptográfiai alkalmazásokhoz – például kulcsok generálásához, protokollok működéséhez, esetleg véletlen feltöltésekhez /random padding/ – állítanak elő véletlen számokat.

A valódi véletlen számok valamilyen véletlen fizikai forráson alapulnak, amelynek outputja nem megjósolható. Ilyen forrás lehet például egy félvezetőből származó zaj, egy hang input legkevésbé szignifikáns bitje, vagy billentyűzet leütések közötti időtartamok. A fizikai forrásból származó zajt ezután „feljavítják", tesztelik, amely olyan outputot eredményez, amelyben a kívánt statisztikai tulajdonságok, valamint az előzményekből „jósolhatatlansági" elvárások garantálhatóak.

Legtöbb alkalmazáshoz fizikai véletlen generátor nem áll rendelkezésre, pszeudo-véletlen számokat alkalmaznak.

A hagyományos véletlen-szám generátorok, amelyek a legtöbb programozási nyelvben rendelkezésre állnak, nem alkalmasak kriptográfiai célokra (ezek valamilyen statisztikai véletlenszerűségre vannak tervezve, és nem arra, hogy ellenálljanak a kriptoanalízisnek).

Például C-nyelvben a véletlenszám generátor alapja egy kongruencia-elvű számítás:

s_0 initial vector,

$s_i \equiv as_{i-1} + b$ (modulo m),

ahol a, b és m fix konstansok, általában $m = 2^{24}$, vagy $m = 2^{32}$.

A véletlenszám generátorokra is irányt mutatnak szabványok, például **ANSI** /American National Standard/ **X9.17** (1991): Key Management; **FIPS** /Federal Information Processing Standard Publication/**186–2**, Digital Signature Standard /DSS/ (2000), Appendix 3-ban szerepel két véletlenszám generátorspecifikációja; tesztelésükre **NIST** /National Institute of Standards and Technology/ Special Publication **800–22** (2001): Statisztikai próbák a kriptográfiai alkalmazásoknál használt véletlen- és pszeudo-véletlen szám generátorokra ...

A kriptográfiai célú véletlenszám generálás követelményeiről, leggyakrabban alkalmazott eljárásairól részletesen olvashatunk Papp Pál ebben a folyóiratban megtalálható cikkében.

7.1.2. Titkos kulcsú primitívek

7.1.2.a. Egyirányú fv-ek

Az általános elnevezés szerint MAC (Message Authentication Code) eljárásaival lehet kulcshoz kötötten egy üzenet olyan tömörített leképezését előállítani, mely

a vételi hibák mellett, a szándékos módosítások kivédését is – titkos kulcstól függően – detektálja. (Ez hasonló a HASH képzéshez, csak ott nincs titkos kulcs). A legismertebb ilyen eljárás a DES láncolt üzemmódjához kapcsolódik, ahol a teljes üzenet láncolt rejtjelzése után még egy blokk rejtjelzésével, az utolsó blokk 4 byte-ja adja a MAC képet, melyet a rejtjeles blokkokhoz fűzve detektálható minden módosítás, blokkelhagyás, blokk-sorrend csere stb.

MAC-ra vonatkozó szabványok:

MAC (DAC)	*FIPS 113*,Computer Data Authentication May 1985
HMAC	*FIPS 198*,The Keyed-Hash Message Authentication Code (HMAC) March 2002.

7.1.2.b. Folyam-rejtjelzők /Stream ciphers/

A kulcsfüggő – rejtjelző – primitívek hatalmas osztályát jelentik az ún. folyamrejtjelzők, melyek karakterenkénti (bitenként, betünkénti, byte-onkénti) rejtjelzést valósítanak meg, míg a blokkrejtjelzők egyszerre több karaktert (pl. 64, vagy 128 bitet, illetve ennek megfelelő byte-ot) alakítanak át a kulcs függvényében.

A folyamrejtjelzők a karaktereket véletlen (valódi-, vagy pszeudo véletlen) kulcselemekkel módosítják bitről bitre (byte-ról byte-ra). Szokták még a biteken operáló folyamrejtjelzőt bitsoros rejtjelzőnek is nevezni.

A bitsoros rejtjelzések szerepe a blokkrejtjelzések terjedésével is jelentős, melynek sebességi, kommunikációs és biztonsági okai is vannak.

Bizonyos alkalmazásokban a blokkrejtjelzésekhez szükséges karakterek bevárása rejtjelzés előtt (kiegészülve a rejtjelzés idejével) a kommunikációban nem elfogadható, vagy a blokkos rejtjelézek esetén fellépő nagyobb hibaterjedés hátrányos (ezen okok miatt használnak a GSM rendszerben stream cipher kódolást, ld. később az A5 algoritmust). További előnye a folyam-rejtjelzőknek, hogy nagyon gyors működéssel implementálhatók célhardverben is.

Nagy számú kripto-analitikai módszert dolgoztak ki folyam-rejtjelzőkre, élesítettek, melyeket azután hatékonyan használhatnak blokk-rejtjelzők esetében is (pl. a lineáris kriptoanalízist, vagy a blokkrejtjelzők egyik tesztje az Output feedback üzemmódú stream cipher felhasználás tesztelése).

A folyam-rejtjelzők legnagyobb jelentőségét mégis az adja, hogy az egyetlen bizonyítottan megfejthetetlen rejtjelzés, az **OTP** *(one-time-pad)* is folyam-rejtjelzés. Ennél – megfelelő tulajdonságokkal generált, adó, vevő oldalra védetten szétosztott, ott védetten tárolt – valódi véletlen elemek módosítják a nyílt szöveg karaktereit.

Az eljárás a Vernan rejtjelezésen (1926) alapul. Shannon tárgyalta az információ-elméleti hátterét (1949), leírása szerint – tökéletes rejtjelezésnek nevezve – az eljárás:

- üzenet: $m_0, m_1, m_2, \ldots, m_i \in \{0, 1, \ldots, N-1\}$
- kulcs: $k_0, k_1, k_2, \ldots, k_i \in \{0, 1, \ldots, N-1\}$ (az üzenet karaktereivel azonos jelkészletű) valódi véletlen számok

- rejtjelszöveg: $c_0, c_1, c_2, \ldots$
  ahol $c_i = m_i + k_i \bmod N$

Feltétel: a küldő és fogadó ismeri a kulcsot, de a támadó nem.

Ekkor – Shannon bizonyította – a támadó a rejtjelest szövegből semmilyen következtetésre nem tud jutni a nyílt üzenetről (annak hosszán kívül).

Precíz matematikai leírás, bizonyítás olvasható [6], valamint [2] könyvekben.

A fentiek alapján megkülönböztetünk

- végtelen kulcsterű folyam-rejtjelzőt (OTP)
- és véges kulcsterű folyam-rejtjelzőket.

A véges kulcsterű folyam-rejtjelzők, mint álvéletlen generátorok legtöbbje leírható véges automata modellel, annak állapot és kimeneti függvényeivel.

A folyam-rejtjelzők legelterjedtebb osztályát a lineáris visszacsatolású shift regiszterek (jelölésben az angol betűkezdetekből: LFSR) kombinációin alapuló véletlenszám generátorok alkotják, amikor a „feedback function" (visszacsatoló függvény) $f(x_1, x_2, \ldots, x_n)$ lineáris, azaz felírható a következő formulával:

$$f(x_1, x_2, \ldots, x_n) = c_1 x_1 \oplus c_2 x_2 \oplus, \ldots, \oplus c_n x_n.$$

Általános esetben a műveletek $GF(q)$ felett értendők. Ha a c_i konstansok 0 vagy 1 értéket vehetnek fel, a művelet moduló 2 /azaz $GF(2)$/, akkor bináris lineáris visszacsatolású shift regiszterekről beszélünk.

A generált véletlen számok az output bitjei, amelyek néhány órajel (léptetés) sebességgel generálódhatnak (azaz sok GHz sebességgel az órajel függvényében, mely nagyságrendekkel nagyobb sebesség, mint a PC-ken megvalósított véletlenszám-generálás sebessége).

Lineáris shift-regiszterek előnyei:

Nagyon jól kidolgozott elméleti háttér van,

- nagy periódus: m állapot esetén a maximális periódus $2^m - 1$ ($GF(q)$ feletti LFSR esetén $q^m - 1$);
- jó tulajdonságok (Golomb postulátumok)
  egyenletesség, autokorreláció ...
- hardver-megvalósítási hatékonyság (szoftver is).

Lineáris shift-regiszterek hátrányai:

- megjósolhatóság;
  - a sorozat lineáris komplexitása kicsi (regiszter-hossznyi);
  - néhány ismert nyílt és hozzátartozó rejtjeles karakter ($2m$ bit) elegendő az állapot visszaállításához a Berlekamp–Massey algoritmussal;
- lineáris módszerekkel támadható /azaz akár a rejtjeles sorozatból is visszaállíthatóak a kezdeti induló állapotok (a kulcs)/.

Shift regiszteren alapuló konstrukciók a bonyolultság növelésére:

- filter generátorok, melyeknél az LFSR kimenete bonyolult Bool-függvénye az állapotoknak;
- kombinációs generátorok, melyeknél több lineáris visszacsatolású shift regiszter outputját – egy nem lineáris függvénnyel – kombinálják, és így a több outputból együttesen számítható a generátor kimenete.

A kombinációs generátorok legismertebb típusa az ún. *Geffe generátor*, ahol a z_i kimenet három LFSR (a_i, b_i, c_i) kimenetéből kapható: $z_i = a_i b_i + c_i(b_i + 1)$. A képlet szerint (ez az eredeti hardver konstrukcióból adódik), a három LFSR közül a középső (b_i) vezérli, hogy a kimenetet az első (a_i), vagy a harmadik (c_i) LFSR outputja legyen-e.

Egyéb LFSR-eken alapuló ismert generátorok:

- clock control /alternatív léptetés/ generátor;
- Stop-and-go generátorok;
- Gollmann cascade;
  - egymást léptető shiftregiszterek;
  - multiplexor (Jennings): az egyik LFSR mondja meg, hogy melyik állapotbitet vegyük ki a másik generátorból;
- FCSR – Feedback with carry (nemlinearitás a carry bit segítségével, például Klapper, Goresky 1995).

Léteznek nemlineáris visszacsatolású shift regiszterek is, például a De Bruijn sorozatok bizonyítottan megfelelő statisztikai tulajdonságokkal (maximális $2L$ periódussal).

Néhány gyakorlati alkalmazás:

A legáltalánosabban használt, lineáris visszacsatolású shift regisztereken alapuló alkalmazás a GSM kommunikáció rejtjelző algoritmusa, az A5, melynek két verziója van: az A5/1 és A5/2.

Ez az algoritmus egyszerre *kombinációs* (három LFSR-t használva) *és clock-control* generátor is (az A5/1 az adott három regiszterből kivett bitek alapján, míg az A5/2 algoritmus egy negyedik shift regiszterből kivett bitek alapján, ezek által vezérelve szabálytalanul lépnek az egyes regiszterek).

86 kulcsbit betöltés után 100 léptetés történik a rejtjelzés megkezdése előtt /lavinahatáshoz/, mellyel „eléggé" behatnak a nem lineáris struktúrák.

Leírását ld. pl.: `http://calliope.uwaterloo.ca~ggong/ECE710T4/lec8-ch6b.pdf` vagy `http://cryptome.org/gsm-a512.htm`.

Mind az A5/1, mind az A5/2 algoritmusára, kihasználva a protokoll hibáit is, léteznek támadási algoritmusok (ld. pl.: Alex Biryukov- Adi Shamir: Real Time Cryptanalysis of the Alleged A5/1 on a PC, 1999.)

Más, nem LFSR alapú folyam-rejtjelzéshez használt algoritmusok:

- RC4 (Rivest, RSA Security):

  Az egyszerű, bájt orientált rejtjelezést Rivest tervezte 1992-ben, mely kódolás 256 bájtos önmagát módosító permutációs táblát használ:

  $S[i]$: 256 byte elemű tömb inicializálása után:

  $i = i + 1$, $j = j + S[i] \bmod 256$

  $S[i]$, $S[j]$ felcserélése

  Kimenet $z = S\big[S[i] + S[j] \bmod 256\big]$

  Széles körben használják (pl. SSL, WEP, Windows alkalmazások).

- PKZIP (Schaffely)

  3 db 32 bites regiszter

  - 2 db lineáris shift-regiszter,
  - 1 db kongruencia generátor.

Léteznek blokkos rejtjelzésen alapuló bitgenerátorok is, melyek generálási sebessége lényegesen lassabb a fenti konstrukcióknál.

Jelentős kutatások folynak arra vonatkozóan, hogy mikor nem tudja megkülönböztetni egy feltételezett támadó a pszeudo-véletlen (kis kulcsból generált) sorozatot a valódi véletlen sorozatoktól, mert ekkor a támadó számára olyan nehézséget jelentene feltörni a kódolást, mint a valódi véletlen sorozatok esetében – azaz lehetetlen – (ld. pl. [2]).

7.1.2.c. Blokk-rejtjelző algoritmusok

1973 májusában tűzték ki egy nyilvános, széles körben alkalmazható algoritmus kidolgozását, melynek neve ***DES*** (Data Encryption Standard) lesz. Először egy LUCIFER elnevezésű jelölt algoritmust publikáltak 1975. márciusában, majd széleskörű viták után 1977. január 15-én fogadták el hivatalosan a „Feistel" struktúrán alapuló, ma DES-ként ismert algoritmust.

A *National Bureau of Standards* a FIPS No. 46 sz. publikációban szabványosította DES „Nemzeti Adatfeldolgozási Szabványt".

A blokkméret 64 bit, a nyílt üzeneteket először 64 bites blokkokra bontja (ha az utolsó blokk csonka, akkor azt kiegészíti). Ezeket ugyanazzal a kóddal képezi le ugyancsak 64-bites rejtjeles blokkokba, amelyeket egymás után írva kapjuk a rejtjeles üzeneteket. A megválasztható kódok száma 2^{56} (az 56 szabadon választható kulcsbiten kívül további 8 bit ellenőrzési célokat szolgál), tehát az effektív kulcsméret nagysága 56 bit.

A DES megfejthetőségére utaló publikációk sorát Martin Hellman egy 1977-ben tartott előadása nyitotta meg, aki a teljes kipróbálást is kivitelezhetőnek tartotta megfelelően épített hardware segítségével (akkoriban ennek költségét 20 millió dollárra becsülte).

Javításként először a Hellman ötletén alapuló fejtést gyakorlatilag kivitelezhetetlennek beállító eljárásként a DES kétszeres alkalmazását javasolták. Ehhez

az üzenetet kétszer kellett egymás után rejtjelezni két, egymástól függetlenül választott kulccsal, ami a kulcs méretét immár 112 bit hosszúságúvá tette. Ezzel kapcsolatban már 1992-ben Merkle és Hellman nyilvánvalóvá tette azonban, hogy ha a „simple DES" fejthető, akkor a „meet in the middle attack" (egy „középen találkozunk" elnevezésű módszer) lehetővé teszi a „Double DES" fejtését is. A hírek szerint több helyen építettek olyan célgépet, amellyel a Double DES-t fejteni lehet.

A DES jelenleg elterjedt változata a „Triple DES", /más jelölésekben "T-DES", „3-DES"/. Ez vagy kettő, vagy három 56 bites kulccsal dolgozik. Az üzenetet először az első kulccsal rejtjelzik normál DES módban, majd a második kulccsal a megoldó algoritmust alkalmazzák. Az így nyert közbülső szövegre alkalmazzák ismét az első, három kulcsos rendszerben a harmadik kulcsot.

A TDES struktúra algoritmusát egyes leírásokban TDEA algoritmussal jelölik:

A TDES modellje:

Rejtjelzés képlete: $C = E_{K3}(D_{K2}(E_{K1}(M)))$.

Megoldás képlete: $M = D_{K1}(E_{K2}(D_{K3}(C)))$.

A TDES-hez 3 kulcsra van szükség (K_1, K_2, K_3), melyeket az egyes standard leírásokban az alábbiak szerint specifikálnak:

1. Opció: K_1, K_2, és K_3 független kulcsok.
2. Opció: K_1 és K_2 független kulcsok, és $K_3 = K_1$.
3. Opció: $K_1 = K_2 = K_3$.

(A 3. változat előnye, hogy egyszeres DES-sel rendelkező célhardverekkel is képes kommunikálni – természetesen a biztonság rovására.)

A DES kriptoanalízisének sikerei, az amerikai hozzáállás miatti bizalmatlanság alapján az európaiak is törekedtek egy szabvány (megbízható – azaz saját) rejtjelző algoritmus elterjesztésére.

1990-ben Lai és Massey javasoltak a DES kiváltására egy **PES** (Proposed Encryption Standard) elnevezésű algoritmust, majd a differenciál kriptoanalízis nevű támadási módszer által felfedett gyengeség miatt ezt módosították, kezdeti elnevezésben a módosított algoritmus neve IPES /Improved Proposed Encryption Standard/ volt, melynek végső elnevezése – ahogy a szakirodalomban ismert, a legtöbb kriptoprotokoll rejtjelző készletében szerepel – **IDEA** /International Data Encryption Algorithm/ lett.

Bár az IDEA algoritmust a szakemberek kellő erősségűnek tartják, több protokollba is implementálták, mégsem váltotta fel a DES-t világszabványként.

A nemzetközi bizalom megrendülése a DES biztonságában inspirálta a National Institute of Standards and Technology (NIST) azon döntését, hogy ki kell fejleszteni a DES utódját, amely az Advanced Encryption Standard (AES) nevet kapta.

A NIST döntése alapján az AES algoritmust nyilvános pályázat során választották ki. Ehhez 1997 szeptemberében közzétették azon elvárásoknak a listáját, amelynek az AES algoritmusnak meg kell felelni. Ugyanakkor deklarálták,

hogy a benyújtott rejtjelzési algoritmusok nyilvánosak, szabadon felhasználhatók lesznek. A kiírás szerint blokkos (128, 196, 256 biten operáló), 128, 196 és 256 bites kulcsméret opcionálisan egyaránt választható algoritmusnak kell lenni, mely ellenáll minden ismert fejtési támadásnak. Követelmény volt továbbá, hogy hatékonyan implementálható (azaz gyors, bármely plattformon kis memóriával is megvalósítható) legyen.

A tervek szerint az elvárásoknak megfelelő rejtjelzési algoritmus hosszú távra, akár 20-25 évre is megoldja a polgári életben keletkező adatok biztonságos védelmének a kérdését.

A beérkezett pályaművek közül 15 felelt meg a formai elvárásoknak. A benyújtott algoritmusok mögött komoly multinacionális cégek sorakoznak fel (*IBM, Microsoft, RSA Laboratories, Deutsche Telekom AG, Nippon Telegraph and Telephone Corporation, Centre National pour la Recherche Scientifique, stb.*) Ezek elemzése a legszélesebb nyilvánosság bevonásával folyt. Maguk a szerzők, de más kriptográfusok is elemezték az algoritmusokat, s több, kifejezetten e témának szentelt konferenciát is tartottak.

A NIST *1999-ben Rómában* rendezett konferenciáján 15 algoritmust értékeltek. A konferencia megrendezésével a NIST-nek az volt a célja, hogy az itt elhangzott értékelések segítséget nyújtsanak annak az öt algoritmusnak a kiválasztásához, amelyek továbbra is versenyben maradnak. Ezután már csak ezeknek az algoritmusoknak a vizsgálata folyt.

Az öt, a végső versenyben maradt algoritmus ábécé sorrendben:

- **Mars** algoritmus: szerzői: az IBM[2] több mint tízfős csapata (USA),
- **RC6** algoritmus: szerzői: Ron Rivest[3] és csapata, RSA Laboratories (USA),
- **Rijndael** algoritmus: szerzői: Daemen, Rijmen, belga kriptográfusok,
- **Serpent** algoritmus: szerzői: Anderson, Biham[4], Knudsen nemzetközi csapat,
- **Twofish** algoritmus: szerzői: Bruce Schneier[5] és csapata (USA).

Az algoritmusok analizálása, elemzése rendkívül nagy erőket kötött le. A viszonylag rövid, jó kétéves vizsgálati idő ellenére állítható, hogy a DES után ez az öt algoritmus a világ legmélyebben elemzett szimmetrikus kulcsú algoritmusa. A győztes algoritmust 2000. október másodikán jelentették be. A versenyt a RIJNDAEL algoritmus nyerte. Szerkezete alapján az algoritmus nem hasonlít a legtöbb blokkos algoritmusra, nem követi a DES Feistel-struktúráját. Ez is számos iterációs lépésben valósul meg. Minden iteráció három rétegből áll, ezek szerepe különböző.

[2] közülük több kriptográfus már a DES kifejlesztésében is jelentős szerepet játszott
[3] az RSA algoritmus egyik feltatálója
[4] a differenciális kriptoanalízis kidolgozója
[5] az „Applied Cryptography" c. könyv és több jelentős hatást kiváltó módszer szerzője

- *ByteSub:* nem-lineáris keverés réteg (S-dobozok),
- *ShiftRow:* pozíció-keverő réteg,
- *MixColoumn:* oszlop-keverő réteg,
- *Round Key Addition:* kulcs-addíciós réteg.

Az algoritmus több technikát alkalmaz (pl. byte-szintű operációk a 256 elemű véges test fölött).

7.1.3. Nyilvános kulcsú primitívek

7.1.3.a. PKI kódolók

RSA kódoló:

Az RSA rendszer olyan ismert, hogy részletes leírását itt mellőzzük.

- Előnyei:

  könnyen megérthető algoritmus;
  biztonsága klasszikus (egész számok faktorizálása, IFP) problémára vezethető vissza.

- Hátrányai:

  ha $m(= p * q)$ nagy, nagyon számításigényes (lassú).

Hátizsák (knapsack) probléma, Merkle-Hellman algoritmus

Legyenek $a_1, a_2, \ldots, a_n$ természetes számok, ún. „súlyok", $\underline{x_i} \in \{0,1\}^n$ bitvektor

$$S = \Sigma x_i a_i$$

Ekkor S, $a_i - k$ ismeretében „nehéz" x_i-ket meghatározni, ugyanis a KnapSack (hátizsák) probléma NP-teljes, ha a_i-k véletlenszerűek.

Viszont könnyű a meghatározás, ha a_i-k „szupernövekedő" sorozatot alkotnak (pl. 2 hatványai).

Az ezen alapuló rejtjelzés leírása szerepel a 8.2. redukciós biztonságot tárgyaló fejezetben.

Elliptikus görbék pontjain értelmezett DLP (Elliptic Curve Discreet Logarithm Problem, ECDLP)

Néhány ECDLP-n alapuló algoritmus:

- *Elliptic Curve Diffie–Hellmann Key Agreement (ECDH),*
- *Elliptic Curve Menezes-Qu-Vanstone (ECMQV),*
- *Elliptic Curve Integrated Encryption Scheme (ECIEC),*
- *Elliptic Curve Digital Signature Algorithm (ECDSA),*
- *Elliptic Curve Nyberg-Rueppel (ECNR).*

Az EC-n alapuló rejtjelző algoritmus rövid leírása:

A $P(x,y) : y^2 = x^3 + ax + b \pmod p$ görbe pontjain értelmezünk egy algebrai struktúrát az alábbi műveletekkel:
Összeg: $P(x_1,y_1) + P(x_2,y_2) = P(x_3,y_3)$:

- $x_3 = k^2 - x_1 - x_2$
- $y_3 = k(x_1 - x_3) - y_1$, ahol
  $k = (y_2 - y_1)/(x_2 - x_1)$, ha $P(x_1,y_1) \neq P(x_2,y_2)$ egész;
  $k = (3x_1 + a)/(2y_1)$, ha $P(x_1,y_1) = P(x_2,y_2)$.

Ezen a struktúrán is értelmezhető az úgynevezett diszkrét logaritmus probléma: $e * P = Q$ probléma megoldása e-re. A felsorolt algoritmusok mutatják, hogy ebben a struktúrában is létrehozhatók a nyilvános kulcsú algoritmusok megfelelői. A rendszer előnye az RSA-hoz képest, hogy lényegesen kisebb számokkal is elérhető a biztonság, mert ebben a struktúrában nem tudunk olyan hatékonyan diszkrét logaritmust számolni.

Több más nyilvános kulcsú primitív is ismert (pl. Rabin, ElGamal, McEliece, Chor-Rivest knapsack ..., ld. pl. [5]), itt csak szemléltettünk néhányat.

7.1.3.b. Aláíró primitívek

RSA alapú aláírás algoritmusai:

Ismert az aláíró előtt	Ismert az ellenőrző előtt	
Prímek: p, q $n = p * q$ Kulcsok: d, e $d * e = 1 \bmod (p-1) * (q-1)$	n e y	Bár Európában az RSA a legelterjedtebb, legismertebb aláíró primitív, de ismertek (Amerikában inkább használatosak) a **DSA**, és az **elliptikus görbéken** alapuló primitívek is, ld. **FIPS 186-2** szabványt.
Az x üzenet aláírása: $y = \text{hash}(x)^d \bmod n \Rightarrow$	Ellenőrzés: $\Rightarrow z = y^d \bmod n$ $Z = \text{hash}(x)$?	

További, kevésbé ismert primitívek:

XTR (Efficient Compact Subgroup Trace representation)

ld. `www.ecstr.com`

- Egyesíti az RSA, ECL előnyeit (gyorsaság, kis memória igény, kis méretek).
- Erőssége a DLP-n alapul (Diszkrét Logaritmus Probléma).

NTRU

ld. `www.ntru.com`

- Rácsredukción alapul.
- Még rövidebb programkód, paraméterméretek.

7.1.4. Törekvések új primitívek, szabványok kialakítására – A NESSIE projekt

A NESSIE (New European Schemes for Signatures, Integrity and Encryption) projekt 2000 januárjától 2003 márciusáig tartott. A projektről angol nyelven a `www.cryptonessie.org` honlapon található részletes információ.

A projekt – ellentétben az AES projekttel – a bizalmasságot, adat sértetlenséget és hitelesítést szolgáltató primitívek széles készletének szabványosítását tűzte ki célul: ezek a primitívek blokkos rejtjelezéseket, bitsoros rejtjelezéseket, hash függvényeket, MAC (üzenet hitelesítő kód) algoritmusokat, digitális aláírás sémákat és nyilvános kulcsú rejtjelzési rendszereket tartalmaznak. A végső cél a kriptográfia területén az európai kutatás erős pozíciójának megtartása és az európai ipar pozíciójának erősítése volt.

A meghirdetett kategóriák nagy részében elfogadtak egy vagy több kriptográfiai primitívet. A következő táblázat azt mutatja, hogy az ajánlott primitívek között kategóriánként mindig akadt egy európai fejlesztésű is, de a többi mind tengerentúli (amerikai vagy japán) volt. A bitsoros rejtjelző algoritmusok kategóriájában nem fogadtak el ajánlásra egyetlen pályázatot sem. Ez részben annak volt köszönhető, hogy nagyon erősen határozták meg a biztonsági szintet, elvárták a véletlentől való megkülönböztethetetlenséget is, amit adott szinten egyik algoritmus sem tudott teljesíteni. A kulcs teljes kipróbálásánál kevesebb lépésben mindegyik algoritmust meg lehetett különböztetni a valódi véletlen sorozattól. Emiatt a bitsoros rejtjelző algoritmusokra egy új projektet írtak ki a 2005–2007 időszakra, amelynél csak azt a minimális követelményt fogalmazzák meg, hogy a valódi véletlentől való megkülönböztethetőséghez legalább 2^{64} lépésre legyen szükség.

A következő táblázat azt mutatja, hogy a pályázatokat mely kategóriákban hirdették meg, az egyes kategóriákban hány pályázat érkezett, a végső fázisban hányat ajánlottak közülük és melyek ezek. Az utolsó oszlopban dőlt betűvel szereplő elemek az USA-ban használatos szabványok, melyeket szintén ajánlanak európai használatra is.

Típus	Érkezett	Ajánlott
Blokkos rejtjelző algoritmus	17 db	MISTY-1[6] (Mitsubishi) Camellia (Nippon) SHACAL-2 (Gemplus,F) *AES (USA FIPS 197)*
Bitsoros rejtjelző algoritmus	6	–
Önszinkronizáló bitsoros rejtjelző algoritmus	0	–
Üzenet hitelesítő kód (MAC)	2	Two-Track-MAC (B,D) UMAC (USA) *CBC-MAC (ISO/IEC 9797-1)* *HMAC (ISO/IEC 9797-1)*
Hash függvény	1	Whirlpool (B, Brazilia) *SHA-xxx (USA FIPS 180-2)*
Nyilvános kulcsú rejtjelző algoritmus	5	ACE Encrypt (IBM Zürich) PSEC-KEM (Nippon) *RSA-KEM (ISO/IEC 18033-2)*
Digitális aláírás séma	7	SFLASH (F) ECDSA (USA, Kanada) RSA-PSS (USA)
Aszimmetrikus azonosítási séma	1	GPS (F)

IEEE P1363:

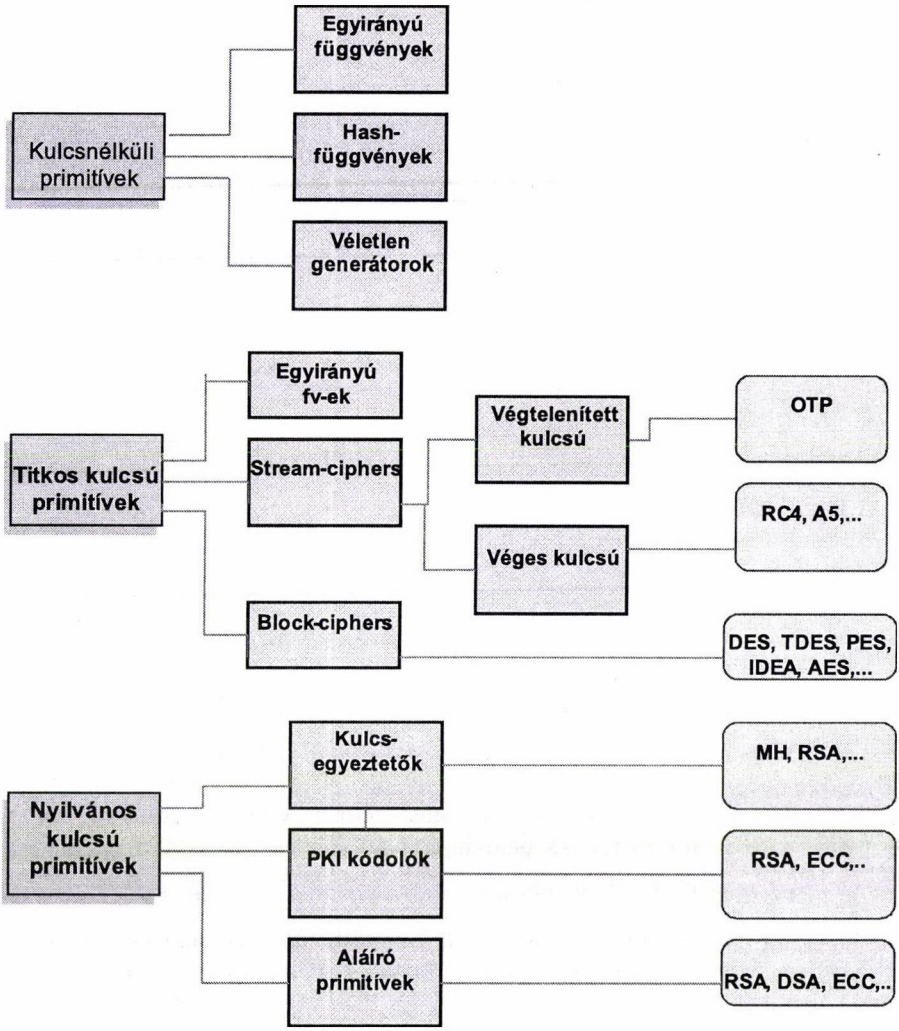
„Standard Specifications for Public-Key Cryptography" céljára indított projektről részletek olvashatók az interneten.

7.2. Kriptográfiai sémák

A kriptográfiai sémák feladata annak meghatározása, hogy a kriptográfiai primitívek hogyan rejtjelezzék a teljes rejtjeles üzenetet, azaz mit kezdjenek a rejtjelzőblokknál hosszabb üzenetekkel (darabolás), ezeket hogyan láncolják össze, mit kezdjenek az utolsó, nem teljesen kitöltött blokkal, milyen kulcsokat használjanak, hogyan generálják a kulcsokat, stb. Közbülső réteget képeznek a primitívek és a protokollok között. Ide tartoznak például az alábbiak:

- üzenet feldarabolása blokkokra;
- blokkfeltöltés (pl. utolsó blokk esetén), feltöltés jelölése; az elektronikus aláírás területén szabványként használandók az „emsa-pkcs1-v1_5", „emsa-pss" blokkfeltöltő eljárások;
- blokk-rejtjelzés üzemmódjai;
- véletlen választás, prímteszt;
- KAS: Key Agreement Sceme;

[6] A MISTY blokkos algoritmuson alapuló bitsoros rejtjelző algoritmus lett a 3. generációs mobil telefonok rejtjelző algoritmusa.



2. ábra. A primitívek összefoglalása

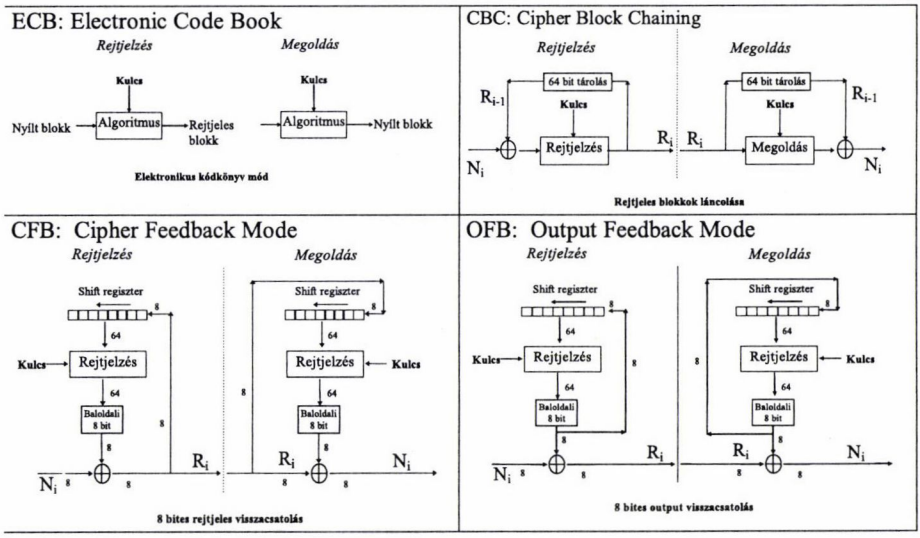
– SSA: Signature Scheme with Appendix; stb.

7.2.a. Blokkrejtjelző üzemmódok:

Közismertek a blokkos rejtjelzések különböző „üzemmódjai", melyek akár DES, TDES, AES, IDEA esetén használhatóak.

A blokkos algoritmusok fix méretű adatblokkokkal operálnak, a valóságban azonban egy üzenet bármilyen méretű lehet. Esetenként olyan adatfolyamot kell rejtjelezni, amelynek a hossza sem ismert. Ehhez hasonló problémák megoldására a blokkos algoritmust különböző üzemmódokban használhatjuk. Ezek a szabványosított[7] működési módok a gyakorlatban nagyon hasznos segítséget nyújtanak.

Például az elektronikus kódkönyv mód (ECB), melynél a 8 byte méretű blokkokra bontott nyílt információ az algoritmus inputja (DES esetén), s az output lesz a rejtjeles. A rejtjeles blokkjai egymástól függetlenek, vagyis egy potenciális támadó törölheti, beszúrhatja, átrendezheti a rejtjeles blokkokat. A legnagyobb veszélyt ebben a módban az jelenti, ha olyan dokumentumokat rejtjeleznek így, amelyeknek csak egy kicsi része változik, az egyéb struktúrák változatlanok maradnak. Pl. egy szerződés, ahol mindig csak az összeg változik. Ekkor egy kevés oldalinformáció (side information) segítségével összeállítható egy táblázat, hogy melyik rejtjeles blokkhoz milyen összeg tartozik. Használata rövid (egy-két blokk) üzenetek esetén ajánlott.



[7] Information processing – modes for operation for a 64-bit block cipher algorithm-ISO 8372

A következő szabványok a visszacsatolási módok pontos leírását adják:
FIPS 81, December 1980.
SP 800–38A 2001 ED, December 2001.
SP 800–38C, May 2004.

7.2.b.

Az RSA rendszernél (de több más nyilvános kulcsú primitívnél) fontos nagyon nagy *prímszámok generálása*, mely az előkészítő fázishoz tartozik. Ilyen sémához tartozó eljárások a nagy véletlen számokra vonatkozó *valószínűségi tesztek* (pl. Miller-Rabin teszt, Lucas prímteszt, Solovay-Strassen prímteszt), melyek *gyorsak*, de *nem döntik el teljes biztonsággal*, hogy a kérdéses szám prím-e. Azonban a tévedés valószínűsége a teszt többszöri végrehajtásával – ha mindig pozitív a válasz – *tetszőleges küszöbérték alá csökkenthető*. Így ezek a módszerek kriptográfiai célokra – például RSA kulcsgenerálásra – megfelelőek. Az algoritmushoz szükséges paraméterek választási szempontjai, eljárásai már a sémákhoz tartoznak. (Pl. az RSA esetén milyen kiegészítő feltételeknek kell a prímszámoknak eleget tenni. Egyes rendszerekben megkötik, hogy $(p-1)$-nek és $(q-1)$-nek is legyenek nagy prímosztói, sőt ezen nagy prímosztókat eggyel csökkentve ennek is legyenek nagy prímosztói. Egyes gyorsítási ötletekkel szemben az e nyilvános, d titkos kulcsra is különböző alsó korlátokat szokás kikötni, ld. az RSA kalkulációs biztonságáról szóló fejezetet.)

7.2.c.

A kriptográfiai sémákra (az algoritmusok és környezetük egységes szerepének kezelésére) jó példát nyújt az **EESSI-SG** (European Electronic Signature Standardisation Initiative Steering Group) felügyelete alatt dolgozó Algoritmus csoport (**ALGO**) javaslata az elektronikus aláírási készletekre.

Az *elektronikus aláírások* megbízhatósága (sértetlenséget, letagadhatatlanságot biztosító tulajdonságai) alapvetően támaszkodik az egész technológia alapját képező kriptográfiai algoritmusok (és azok paraméterei) megbízhatóságára.

Az elektronikus aláírás biztonságát érintő lehetséges kölcsönhatások miatt az algoritmusok és paraméterek csak előre meghatározott kombinációkban használhatók, melyeket együttesen **aláírás-készletnek** neveznek.

Egy *aláírás-készlet* a következő *elemekből* áll:

- aláíró algoritmus a paramétereivel

  Az aláíró algoritmus az aláírandó dokumentum hash-értékéből képezi az aláírást az aláírás-létrehozó adat (magánkulcs) felhasználásával.

- kulcsgeneráló algoritmus

  A véletlenszám generálásnak az aláírás-létrehozó adat (magánkulcs) generálásánál, illetve bizonyos kriptográfiai algoritmusok (pl. DSA) véletlen paramétereinek generálásakor van jelentősége. Bizonyos esetekben a

hash-függvény feltöltésénél is fontos lehet. Ezért a véletlenszám generálásra vonatkozó kritériumokat mindig a feltöltési módszerekkel és a kulcsgeneráló algoritmusokkal összefüggésben szükséges megadni.

Egy kriptográfiai kulcs megtalálásához, valamint a generátor belső állapotáról bármilyen információ megtalálásához szükséges kipróbálás várható értékének legalább annyinak kell lennie, mint egy **EntropyBits** hosszú véletlen érték megtalálásához.

Egy pszeudo véletlenszám generátort valódi véletlen számmal kell inicializálni.

Ezt a számot kezdőértéknek hívjuk, hossza **SeedLen.**

Négy – az aláíró algoritmushoz rendelt – kulcsgeneráló algoritmust fogadtak el szabványosnak: Rsagen1, Dsagen1, Ecgen1, Ecgen2, melyek valódi véletlen vagy pszeudo véletlen számokból generálnak kulcsokat. Mindegyiknél meghatározott a kulcsgenerálás minimális szabadsági foka: EntropyBits$\geq$128, vagy SeedLen$\geq$128[8].

- feltöltési eljárás

  Bizonyos aláíró algoritmusoknál szükség van a hash-érték kiegészítésére az algoritmus által meghatározott blokk-hosszúságra (pl. RSA modulus hossza). Amennyiben egy aláíró algoritmusnak szüksége van feltöltési eljárásra, akkor ennek ki kell elégítenie a *normatív referenciák*ban található követelményeket.

- kriptográfiai lenyomatolási (hash) függvény.

Ha az aláírás-készlet bármely eleme megbízhatatlan, az egész készlet használata kérdőjeleződik meg.

Ha az aláírás-készlet bármely elemét érvénytelenítik, akkor maga a készlet is érvénytelenné válik. Ha a készlet bármely elemét frissítik, a készletet is frissíteni kell.

Az EU ETSI (ALGO Group) szakbizottsága által biztonságosnak elfogadott aláírás-készletek:

Az elfogadott aláírás-készletek listája az alábbi táblázatban található.

[8] Egy ilyen – a követelményeknek eleget tevő – véletlenszám generátor leírása található [Blum, M and Micali, S: „*How to generate cryptographically strong sequences of pseudo-random bits.*" SIAM Journal on Computing, vol. 4, No. 13, pp. 850–863, 1984]-ben

Készlet sorszáma	Aláíró algoritmus	Aláíró algoritmus paraméterei	Kulcsgeneráló algoritmus	Feltöltő eljárás	Hash-függvény
001-004	Rsa	MinModLen=1020	Rsagen1	emsa-pkcs1-v1_5, vagy emsa-pss	sha1, vagy ripemd160
005	Dsa	PminLen=1024 QminLen=160	Dsagen1	–	sha1
006	Ecdsa-Fp	QminLen=160 r0Min=10^4 MinClass=200	Ecgen1	–	sha1
007	Ecdsa-F2m	QminLen=160 r0Min=10^4 MinClass=200	Ecgen2	–	sha1
008, 009	Ecgdsa-Fp	QminLen=160 r0Min=10^4 MinClass=200	Ecgen1	–	sha1, ripemd160
010, 011	Ecgdsa-F2m	QMinLen=160 r0Min=10^4 MinClass=200	Ecgen2	–	sha1, ripemd160

Megjegyzés: Az RSA algoritmus modulus hosszának minimuma 1020 bit a megszokottabb 1024 bit helyett. Ez lehetővé tesz olyan megvalósításokat, amelyek nem tudják használni a legfelső bit(ek)et.

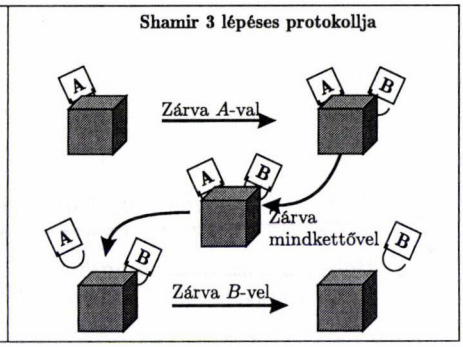
7.3. Kriptográfiai protokollok

Ahogy már említettük, a kriptográfiai protokollokat a partnerek közötti kapcsolat határozza meg. A protokoll a résztvevők közötti egyértelműen meghatározott lépések sorozata, mely két, vagy több résztvevő között zajlik le a biztonsági elvárások maradéktalan teljesülésének érdekében.

A protokolloknak illeszkedniük kell a (megfelelő OSI rétegben aktivizálódó) kommunikációs protokollokhoz. Ezeket meghatározzák a biztonsági elvárások: például a kulcskialakítási és szétosztási lehetőségek és követelmények (ld. kulcscsere protokollok), vagy speciális elvárások (mint a partnerhitelesítési, résztvevői kiegészítő védelmi elvárások, pl. zero-knowledge, secret sharing ...).

7.3.1. Kulcsokkal kapcsolatos protokollok

A kulcsok egyeztetése a legősibb, protokollt igénylő feladatok közé tartozik. Ennek legegyszerűbb formája Shamir híres, három lépéses protokollja:



A nyilvános kulcsú rendszerekkel egyidős a Diffie–Hellmann kulcsegyeztető protokoll, amely először mutatott rá a nyilvános kulcsú kriptográfia lehetőségeire.

Diffie–Hellman féle kulcsegyeztető protokoll:

Legyen a két szereplő $U1$ és $U2$.

Választanak egy csoportban (pl. $GF(p)$) felett egy g primitív (nyilvános) elemet:

Titkos kulcsaik k_{U1} és k_{U2} véletlenül választott pozitív egész számok; ekkor *elküldik egymásnak:* a $P_{U1} = g^{kU1}$ és $P_{U2} = g^{kU2}$. (Azt a DLP megoldásának nehézsége garantálja, hogy g^{kUi} – vagyis a nyilvános kulcsrész – ismeretében egy harmadik fél nem tudja meghatározni k_{Ui}-t, a titkos kulcsot.)

A kulcsegyeztetés:

A Diffie–Hellman protokoll szerint a g^{kU1kU2} érték a közös titok.

Speciális protokollt valósítanak meg a kulcskezeléssel kapcsolatos egyedi elvárások, pl. az alábbiak:

Titkosító kulcsok kezelése

Kulcs letét (*key escrow*): egy titkosító nyilvános kulcs magánkulcs párjának megőrzése a kulcs visszaállítás támogatása érdekében. (Ezt szimmetrikus kulcsú rendszerekben is használják, elsősorban a rendszer működtető szervezet titkosított adatvagyonának visszaállíthatósága érdekében. Az USA-ban Clinton elnök direktívát adott ki a Skipjack algoritmust megvalósító „Clipper Chip" készülékekhez rendelt titkos részkulcsainak két állami szervben megosztott tárolásáról, mely nagy vihart váltott ki, a rendszer nem vált működőképessé. Ld. még **FIPS 185**, Escowed Encryption Standard.)

Kulcs visszaállítás (*key recovery*): egy letétbe helyezett kulcsról másolat készítése, és ezen kulcsmásolat átadása egy erre jogosult kérelmezőnek. A kulcskezelés követelményeiről részletesen olvashatunk a **FIPS 140-2** szabványban: Security Requirements for Cryptographic Modules.

7.3.2. Egyéb, funkcionális protokollok

Mivel a biztonsággal kapcsolatos sokrétű igények a protokollok szintjén jelennek meg, ezért ez a terület rendkívül sokrétű, szerteágazó. Ezért csak néhány példát tudunk mutatni különböző biztonsági feladatot kezelő protokollokra:

- üzenethitelesítő protokoll,
- partnerhitelesítés,
- elektronikus aláírás protokolljai (XAdES, X509v3, CRL lista-kezelés ...),
- zero-knowledge protokoll,
- blind signature protokoll,
- Secret Sharing (titokmegosztási) protokoll.

Terjedelmi okokból ezeket itt nem részletezzük, csak ez utóbbiról példaként adunk rövid áttekintést:

A titokmegosztási protokollok hatékony segítséget nyújtanak informatikai rendszerek kritikus pontjainak védelméhez. A kritikus információ (kulcs, jelszó) szétosztása csökkenti a rendszer függőségét az üzemeltető személyektől, illetve véletlen adatvesztés ellen is védelmet adhat.

A titkos információt olyan módon kell felosztani n személy között, hogy tetszőleges m ($m < n$) személy együttesen rekonstruálni tudja azt, de m-nél kevesebb személy ne legyen képes rekonstruálni a titkot semmilyen esetben sem. A fenti feladatot nevezzük ezután (m, n)-titokmegosztási protokollnak.

Blakley szellemes és szemléletes konstrukciója szerint legyen a titok egy pont (ennek koordinátái) a háromdimenziós térben. Legyenek a titokdarabok a pont vetületei a (x, y), (x, z), (y, z) tengely által meghatározott síkra. (Ez az *eredete az árnyék elnevezésnek.*) Minden ilyen (titok)pont egy egyenest határoz meg, amelyen a titoknak rajta kell lennie. Tehát bármely két titokbirtokos vissza tudja állítani a titkot, ez egy $(2, 3)$ küszöb séma. Természetesen egyéb protokollok is megvalósíthatók a tér és az alterek dimenzióinak megfelelő megválasztásával. Ez a protokoll nem tökéletes, mivel elvárás, hogy nem kellő számú titokbirtokos együttese ne rendelkezzen a titokról szűkítő információval, mely ebben a protokollban nem teljesül, hiszen a titokdarab birtokosa tudja, hogy a titok melyik egyenesen (altérben) van, míg egy kívülálló nem. Megjegyezzük még, hogy itt általában nem euklideszi, hanem véges projektív térben dolgozunk.

A titokmegosztási feladat másik (m, m) modelljét egy p elemű véges testben definiálták. Véges testek felett értelmezett polinomok valamennyi együtthatója a véges test eleme. A titokmegosztási feladatban használt polinom együtthatói véletlenszerűen generáltak és $a_k < p (k = 1, \ldots, m-1)$, illetve a nulladfokú tag együtthatója maga az M titok.

A titok szétosztásához $k_i = (a_{m-1}x_i^{m-1} + a_{m-2}x_i^{m-2} + \ldots + a_1 x_i + M) \pmod p$ értékeket kell kiszámolni az x_i előre meghatározott alappontokban. Célszerű az $1, 2, \ldots, n$ számokat alappontnak kijelölni. A szétosztás során minden résztvevő két releváns adatot kap. A polinom helyettesítési értéke mellett szükség van annak megjegyzésére is, hogy az adott személy mely alappontban kiszámított polinomértéket őrzi. Mindkét adat szükséges a titok visszaállításához.

Már a fenti Shamir-séma is figyelembe tudja venni azt, ha a titokdarabhordozók nem egyformán fontosak. Osszuk az árnyékok hordozóit két csoportra, az első csoport tagjai kapjanak egy titokdarabot, míg a második csoporthoz tartozók kettőt. Ha négy titokdarab szükséges a visszaállításhoz, akkor a második csoportból bármely kettő, az első csoportból bármely négy személy állíthatja vissza a titkot, illetve a harmadik lehetőség az, hogy az

első csoportból kettő, a második csoportból egy titokdarab-birtokosra van szükség a visszaállításhoz.

Általában igaz az, hogy ha egyesével meghatározzuk azokat a részhalmazokat, amelyek jogosultak a titok visszaállítására, akkor konstruálható olyan titokmegosztási séma, amely ezt teljesíti.

7.3.3. Interneten használt komplex (bizalmasságot, integritást, hitelességet biztosító) protokollok

Elterjedtségük miatt külön figyelmet érdemelnek a kriptográfiai primitíveket, sémákat alkalmazó, ***internetes*** kommunikáció biztonságát segítő ***protokollok***.

Az első széles körben elterjedt, biztonságos kommunikációs csatornát megvalósító protokoll a *Netscape* által 1994–95-ben kifejlesztett *SSL (Secure Socket Layer)* volt. 1999-ben az *IETF (Internet Engineering Task Force)* elfogadott szabvány szintre emelte az SSL-t *RFC-t 2246-os számmal*, az SSL továbbfejlesztése a TLS (Transport Layer Security) is internetes szabvány. Ezzel párhuzamosan 1998-ban a *WAP Forum* megjelentette a *WTLS protokoll*t (Wireless Transport Layer Security) tervezetét is, amely a drótnélküli kommunikáció (*WAP, Wireless Application Protocoll*) szabványa. (A WTLS gyakorlatilag a TLS mobil környezetre adoptált változata.)

A főbb követelmények ezekkel a protokollokkal szemben:

- titkosság (secrecy),
- együttműködési képesség (interoperability),
- továbbfejleszthetőség (extensibility),
- relatív hatékonyság (relative efficiency): minél gyorsabb megvalósíthatóság, kevesebb tárolási igény.

7.3.3.a. Az SSH protokoll

A kriptográfiai protokollok a kriptográfia alkalmazásának kiemelt fontosságát mutatják. A kriptográfiai alapelemek ezen protokollok részeként szolgálnak ki egy adott biztonsági elváráshalmazt. A legrégebbi ilyen protokoll az SSH /Secure SHell/, de rendkívül elterjedt az SSL/TLS protokoll is. Az SSH-t részletesebben bemutatjuk, beletekintve a protokoll szerkezetébe, a kriptográfiai alapelemek (primitívek, sémák) beillesztéséhez használt formalizmusba is. Ez a legtöbb protokoll esetében hasonló, ezért a további protokollokat már csak vázlatosabban ismertetjük.

Az SSH protokoll 3 fő részből áll:

A *Transport Layer Protocol* végzi a szerver hitelesítését, és egységes, titkos felületet nyújt. Tartalmazhat tömörítést is. Általában TCP/IP kapcsolaton fut, de bármilyen más megbízható adatfolyamon futhat.

A *User Authentication Protocol* végzi a kliens oldali felhasználó hitelesítését. A Transport Layer Protocol-on fut.

A *Connection Protocol* osztja fel a titkosított csatornát több logikai csatornára. A User Authentication Protocol-on fut.

Két formátum használatos az algoritmus elnevezésekhez:

Az olyan nevek, amelyek nem tartalmaznak „kukacot" (@) azokat az IANA (Internet Assigned Numbers Authority) jelöli ki. Ilyenek például az '3des-cbc', sha-1', 'hmac-sha1' és zlib', az idézőjelek nem részei a neveknek. Ilyen formátumú nevek csak az IANA regisztrációja után használhatók, és nem lehet bennük vessző (,) és kukac (@).

További algoritmusokat bárki elnevezhet a név@domainnév formátum alapján, pl. „ourcipher-cbc@ssh.fi". A formátum @ előtti része tetszőleges, de US-ASCII karakterekből áll és nincs benne vessző és @. A másik rész valós Internet domain név kell, hogy legyen [RFC-1034], amit az adott cég, illetve személy felügyel, aki a nevet definiálta. Az a domaintől függ, hogyan használja a lokális nevet.

Kódolás

A kódoló algoritmust és a kulcsot a kulcs-egyeztetés alatt választja meg a két fél. A kulcs mérete minimum 128 bit kell, hogy legyen. Minden csomag az adott irányban egy adatfolyamnak tekinthető, vagyis az inicializációs vektorok az egyik csomag végéről a másik csomag elejére kell, hogy mutassanak.

A kódolásnak a két irányban egymástól függetlenül kell működnie, és mindkét irányban lehetőséget kell adni különböző kódoló algoritmusok használatára.

Jelenleg az alábbi algoritmusok vannak definiálva:

3des-cbc	kötelező	3 kulcsos DES CBC módban
blowfish-cbc	javasolt	blowfish CBC módban
twofish-cbc	javasolt	twofish CBC módban
aes256-cbc	javasolt	AES (Rijndael) CBC módban, 256 bites kulccsal
aes192-cbc	opcionális	AES 192 bites kulccsal
aes128-cbc	opcionális	AES 128 bites kulccsal
serpent256-cbc	opcionális	serpent CBC módban, 256 bites kulccsal
serpent192-cbc	opcionális	serpent CBC módban, 192 bites kulccsal
serpent128-cbc	opcionális	serpent CBC módban, 128 bites kulccsal
arcfour	opcionális	az ARCFOUR adatfolyam kódoló
idea-cbc	opcionális	IDEA CBC módban
cast128-cbc	opcionális	CAST-128 CBC módban
none	opcionális	nincs kódolás; NEM JAVASOLT

A „3des-cbc" 3 kulcsos DES (kódolás–dekódolás–kódolás), ahol a kulcs első 8 byte-ját használják az első kódolásra, a következő 8-at a dekódolásra és az utolsó 8-at a végső kódolásra. Ehhez tehát 24 byte-nyi kulcs kell (amiből 168 bit kell a kódoláshoz). A CBC mód a külső láncolást jelenti (vagyis hogy csak egy inicializációs vektor van). Ez blokkos kódolás, 8 byte-os blokkokkal.

A „blowfish-cbc" szintén blokkos kódolás, 8 byte-os blokkokkal, CBC módban, 128 bites kulccsal; részletesen megtalálható [7]-ban.

A „twofish-cbc" 256 bites, CBC módban, 16 byte-os blokk-kódolással dolgozik, részletesen tárgyalja a Twofish AES beadvány.

Az „aes256-cbc" az „Advanced Encryption Standard", ami a Rijndael néven algoritmus lesz, CBC módban, 256 bites kulccsal.

Az „aes192-cbc" mint fent, de 192 bites kulccsal.

Az „aes128-cbc" mint fent, de 128 bites kulccsal.

A „serpent256-cbc" CBC módban, 256 bittel dolgozik, részletesen kifejtve a Serpent AES beadványban.

Az „serpent192-cbc" mint fent, de 192 bites kulccsal.

Az „serpent128-cbc" mint fent, de 128 bites kulccsal.

Az „arcfour" az Arcfour 128 bites kulcsú adatfolyam kódoló, elvileg kompatibilis az RC4 kódolással. Az RC4 az RSA Security Inc. védjegye. Az Arcfour (és az RC4) nem biztonságos a gyenge kulcsokkal, ezért figyelmesen kell használni.

A „cast128-cbc" a CAST-128 kódolás CBC módban [RFC-2144].

A „none" azt az algoritmus jelenti, hogy a továbbítás alatt nincs használva semmiféle kódolás. Ebben az esetben az adatok nincsenek védve illetéktelenek elől, ezért lehetőleg kerülni kell a használatát. Bizonyos funkciók ki lehetnek iktatva biztonsági szempontból e kódolás esetén (például a jelszavas hitelesítés).

Egyéb algoritmusokat az előbbiekben meghatározott módszer alapján lehet specifikálni.

Adatintegritás

Minden adatcsomag tartalmaz MAC-ot, ez védi az adat integritását. A MAC-ot egy megosztott, közös csomagsorszámból és a csomag tartalmából számolja ki az algoritmus. Az üzenethitelesítő algoritmust és kulcsot a kulcs-egyeztetés alatt dönti el a két fél. Kezdetben nincs MAC, a hossza így nulla. A kulcs-egyeztetés után, kódolás előtt a MAC-ot a csomag tartalmából kiszámolja az algoritmus:

$$\text{mac} = (\text{MAC kulcs, sorszám}|| \text{ kódolatlan\_csomag}),$$

ahol a kódolatlan_csomag az egész csomagot jelenti MAC nélkül (a hosszmezők, payload, padding), a sorszám egy implicit csomagsorszám uint32-ben ábrázolva. A sorszám 0 az első csomagnál, és minden csomagnál emelkedik (függetlenül attól, hogy MAC vagy kódolás használatban van-e). Nincs sose visszaállítva, még kulcs/algoritmus újraegyeztetéskor se, viszont minden

2^32-dik csomag után újra nulla lesz. A csomag sorszám maga sose továbbítódik, mivel nincs benne a csomagban. A MAC algoritmusok mindkét irányban függetlenül futnak, minden implementációnak meg kell engednie, hogy különböző legyen az algoritmus a két félnél. A MAC algoritmus által előállított MAC byte-ok kódolás nélkül, a csomag végéhez fűzve továbbítódnak. A MAC byte-ok száma a választott algoritmustól függ.

Jelenleg az alábbi algoritmusok vannak definiálva:

hmac-sha1	kötelező	HMAC-SHA1 (digest hossz = kulcshossz = 20)
hmac-sha1-96	javasolt	a HMAC-SHA1 első 96 bitje (digest hossz = 12, kulcs hossz = 20)
hmac-md5	opcionális	HMAC-MD5 (digest hossz = kulcshossz = 16)
hmac-md5-96	opcionális	a HMAC-MD5 első 96 bitje (digest hossz = 12, kulcs hossz = 16)
none	opcionális	nincs MAC; NEM JAVASOLT

A „$hmac-*$" algoritmusok leírása megtalálható: [RFC-2104]. A „$*-n$" MAC algoritmusok az eredmény első n bitjét használják csak.

A hash algoritmusok leírása megtalálható a [7]-ben.

A „none" módszer NEM JAVASOLT, ugyanis ebben az esetben egy aktív támadó módosítani tudja az átvitel alatt lévő csomagot.

Egyéb módszerek definiálhatók az adott specifikációk szerint.

Nyilvános kulcsú algoritmusok

A protokoll képes együttműködni a legtöbb (signature és/vagy kódolás alapú) nyilvános kulcsú formátummal, kódolással és algoritmussal.

Különböző aspektusok alapján lehet definiálni a nyilvános kulcs típusát:

Kulcs formátum: hogyan van kódolva a kulcs, hogyan van reprezentálva a bizonyítvány (certificate).

Signature és/vagy kódolás algoritmusok. Néhány kulcstípus nem támogatja mindkét módszert. A kulcshasználat ugyancsak korlátozva lehet a bizonyítvány alapján. A különféle irányelvekhez különböző kulcstípusokat kell definiálni.

A signature és/vagy a védett adat kódolása.

Az alábbi nyilvános kulcs és/vagy bizonyítvány formátumok vannak jelenleg definiálva:

ssh-dss	kötelező	egyszerű DSS
ssh-rsa	javasolt	egyszerű RSA
x509v3	javasolt	X.509 bizonyítvány
spki	opcionális	SPKI bizonyítvány
pgp	opcionális	OpenPGP bizonyítvány

Egyéb kulcstípusokat az ismert módon lehet definiálni.

A kulcstípusnak explicit módon ismertnek kell lennie (algoritmus egyeztetés vagy egyéb forrás alapján). Általában nem a kulcs részben van.

A bizonyítványok és nyilvános kulcsok az alábbi módon vannak kódolva:

string	bizonyítvány vagy nyilvános kulcs formátum azonosító,
byte[n]	kulcs/bizonyítvány adat.

A bizonyítvány rész lehet 0 hosszúságú string, de a nyilvános kulcs kötelező. Ez a nyilvános kulcs lesz használva a hitelesítéshez.

Ezt a kulcs formátumot használva a jelölés és ellenőrzés a Digital Signature Standard [FIPS-186] alapján történik, az SHA-1-et használva. Leírása megtalálható a [7]-ben.

A kulcs-egyeztetés eredménye

A kulcs-egyeztetés két értéket ad vissza: egy közös titkos K, és egy egyeztetett hash H értékét. A kódoló és hitelesítő kulcsok ezekből származnak. A H, ami az első kulcs-egyeztetésből származik, használatos továbbá még mint session azonosító, amely egy egyedi azonosítója a kapcsolatnak. A hitelesítő eljárások használják az adatnak olyan szignált részeként, ami a privát kulcs létezésének a bizonyítéka. Ha előállt, a session azonosító nem változhat, még akkor se, ha később újabb kulcs-csere történik.

Minden kulcs-egyeztetési módszer meghatároz egy hash függvényt, ami a kulcs-egyeztetés alatt használatos. Ugyanezt a hash algoritmust kell használni a kulcs előállításánál. Jelen esetben HASH néven fog szerepelni.

A kódoló kulcsokat a HASH függvénnyel kell előállítani egy ismert értékből és a K-ból a következőképpen:

– Kezdeti IV kliens ⇒ szerver: HASH(K || H || „A" || session_id) (itt K mpint, „A" byte és a session_id mint sima adat. „A" a sima A karakter, ASCII 65).

– Kezdeti IV szerver ⇒ kliens: HASH(K || H || „B" || session_id).

– Kódoló kulcs kliens ⇒ szerver: HASH(K || H || „C" || session_id).

– Kódoló kulcs szerver ⇒ kliens: HASH(K || H || „D" || session_id).

– Integritás kulcs kliens ⇒ szerver: HASH(K || H || „E" || session_id).

– Integritás kulcs szerver ⇒ kliens: HASH(K || H || „F" || session_id).

A kulcs adatot a hash kimenetének elejéből kell venni. 128 bitet (16 byte-ot) kell használni a változó hosszúságú kódú algoritmusokhoz. Egyéb algoritmusokhoz annyit kell venni az elejéről, amennyire szükség van. Ha a kulcs mérete hosszabb, mint a HASH eredménye, a kulcsot ki kell egészíteni a K, a H és a kulcs összefűzésének HASH értékével. Ezt addig kell ismételni, míg nem lesz elég adat a kulcshoz; a kulcs ennek az eleje lesz. Vagyis:

$K1 = \text{HASH}(K \,||\, H \,||\, X \,||\, \text{session\_id})$ (X mint pl. „A")
$K2 = \text{HASH}(K \,||\, H \,||\, K1)$
$K3 = \text{HASH}(K \,||\, H \,||\, K1 \,||\, K2)$

...
kulcs $= K1 \,||\, K2 \,||\, K3 \,||\, \ldots$

Biztonsági jellemzők

Az SSH protokoll biztonságos csatornát alakít ki egy gyenge védelmű hálózaton. Tartalmaz hitelesítést, kulcs-egyeztetést, kódolást, integritás védelmet. Egyedi session azonosítót generál, amit magasabb szintű protokollok használhatnak.

Előfordulhat, hogy a protokoll úgy van használatban, hogy a host neve és a host kulcsa közti megfeleltetés nem garantált biztonságú. Így nem annyira biztonságos, de megfelelő lehet a nem kritikus biztonság-igényű alkalmazások esetében, és passzív támadások ellen így is védelmet nyújt. Ezt mindenképp figyelembe kell venni implementáláskor.

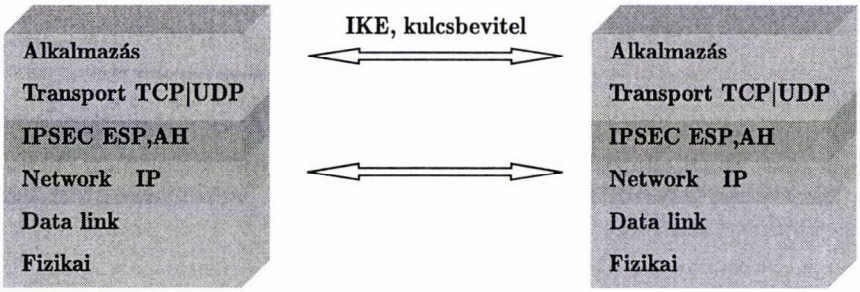
A protokollt átviteli szempontból megbízható hálózaton való használatra tervezték. Ha átviteli hibák vannak, vagy manipulálják az üzeneteket, a kapcsolat lezárul. Ilyenkor a kapcsolatnak újra kell épülnie. Az ilyen típusú Denial-of-Service támadásokat nem védi ki.

A protokoll tartalmaz potenciális rejtett csatornákat. Például a padding, az SSH_MSG_IGNORE üzenetek, és még egyéb helyek a protokollban használhatók rejtett információk átvitelére, és a fogadó fél nem tud megbízható módon meggyőződni arról, hogy ilyen információt fogad.

7.3.3.b. Egyéb fontos protokollok

A hálózati biztonság terén használt de facto standard protokollmegoldások között a két legelterjedtebb a TLS (Transport Layer Security) protokoll és az IPsec (Internet Protocol SECurity). A TLS/SSL protokollt használja a legtöbb web böngésző a http-kapcsolatok védelmére, ezért nagyon jelentős a szerepe pl. az elektronikus kereskedelemmel kapcsolatos kezdeményezésekben. Gyakorlatilag a TLS a világ legtöbbet használt biztonsági protokollja. Ezzel szemben az ***IPSEC*** hostok közötti tetszőleges adatforgalom védelmére szolgál, a virtuális magánhálózatok létrehozásának standard eszköze.

A két protokoll a protokoll stack különböző szintjein dolgozik. A TLS protokoll a TCP réteg fölött helyezkedik el. A gyakorlatban úgy működik, hogy nyit egy új portkapcsolatot, s az eredeti kommunikáció ezen a védett csatornán folyik. Ezzel szemben az IPSEC protokoll a TCP alatt, az IP csomagok szintjén helyezkedik el.



Alapvetően ebből adódnak a két protokoll eltérő tulajdonságai.

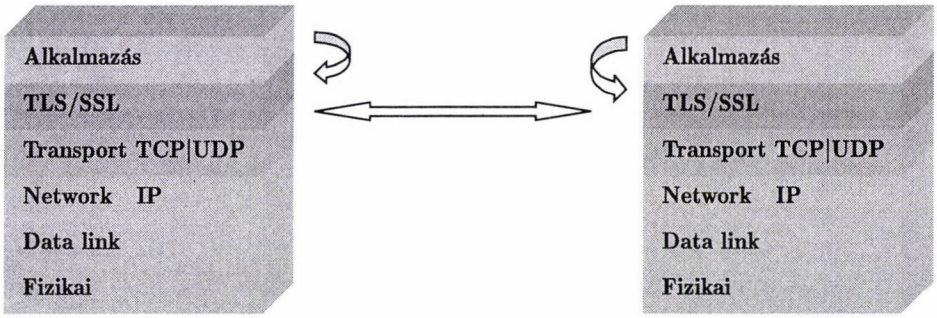
Amiatt, hogy a TLS egy extra protokollként jelenik meg a TCP fölött, a TLS egy TCP session-t képes védeni. Több session védelméhez a protokoll több példányát kell futtatni.

Az IPSEC az IP kiterjesztése, az IP csomagokat védi, ezért egyrészt teljesen transzparens módon minden fölötte folyó TCP (és UDP) kommunikációt véd, másrészt ehhez elég egy IPSEC példányt futtatni.

Mivel az IPSEC az IP szintjén fut, nem tud semmit a fölötte működő alkalmazásokról. Ezért a gyakorlatban nincsenek olyan megoldások, amelyek a felhasználó hitelesítését lehetővé tenné az IPsec alapján, csupán a két kommunikáló host hitelesítése oldható meg.

A TLS ezzel szemben képes a kommunikáló felek kölcsönös hitelesítésére, ami a legtöbb esetben elengedhetetlen (felhasználó azonosítása, jogok hozzárendelése, stb.)

Összegezve elmondható, hogy az IPSEC jelentős térnyerése ellenére ott, ahol a felhasználó hitelesítésére is szükség van, ott az IPSEC helyett a TLS (vagy más hasonló protokoll) használatát kell támogatni.



A fenti protokollokról – több más forrás mellett – a fentitől eltérő szemléletű áttekintés olvasható [2]-ben is.

A Kerberos hitelesítési protokoll

A Kerberos hitelesítési protokollt szintén nagy hálózatok központi hitelesítési mechanizmusaként használják kliens szerver viszonylatban. A Kerberos szerver központilag tárolja mind a kliensek, mind a szerverek titkos jelszavát (itt a titkos jelszó nem az aszimmetrikus kulcsú rendszerek szerint értendő). A Kerberors működése során a DES (Data Encryption Standard) rejtjelező algoritmust használja. A kliens bejelentkezik a Kerberos szerverre, majd közli azonosítóját, és az elérni kívánt szolgáltatás azonosítóját is. A Kerberos szerver véletlenszerűen generál egy ún. session key-t, a továbbiakban majd ezzel rejtjelezve zajlik a kliens és szerver közti kommunikáció. A Kerberos szerver a session key-t és a kliens azonosítóját az elérni kívánt szolgáltatáshoz tartozó titkos kulccsal lerejtjelezi, ez az ún. ticket, amit időpecséttel is ellát. A ticketeknek bizonyos érvényességi idejük van, aminek lejártakor új ticketet kell generálni. Továbbá a session key-t a kliens titkos jelszavával is lerejtjelezi, majd ebből és a ticketből összeállítja a hitelesítési tokent, amelyet a kliensnek visszaküld. A kliens saját titkos jelszava ismeretében a tokenből előállítja a session key-t, majd a ticketet az elérni kívánt szolgáltatást futtató szerverhez küldi. A szerver saját titkos jelszava ismeretében szintén előállítja a session key-t, így mindkét oldalon létrejön a kommunikáció rejtjelezéséhez használandó közös kulcs. A továbbiakban a kliens és szerver között ezzel rejtjelezve folyik a kommunikáció (opcionális), valamint a közös kulcs ismerete bizonyítja a partnerek hitelességét is. A Kerberos támogatja a realm-ok használatát is, ez a Windows NT domain fogalmához hasonló fogalom.

Az S/Key hitelesítési protokoll

Szintén a hálózatok lehallgatásával megszerezhető jelszavak problémája ellen véd az S/Key protokoll. Lényegét tekintve egy alapjelszóból egyszer használatos jelszavakat generál oly módon, hogy a hálózatot figyelő támadó a már megfigyelt jelszavakból nem szerez használható információt a következő esetben használt jelszóra. Az S/KEY jelszavak alapján történő hitelesítés a következőképpen zajlik:

– **Inicializálás**

- A kliens választ egy jelszót és egy véletlen elemet (seed vagy salt.) A véletlen elem szerepe az, hogy a kliens ugyanazt a jelszót több rendszerben is használhassa.
- A jelszót és a seed-et egymáshoz fűzi, s erre alkalmazza az MD4 hash függvényt. Az outputként kapott 128 bit első és második felét modulo 2 összeadva kapja az első, 64 bites jelszót.
- A kliens elkészíti az első száz, esetleg ezer jelszót úgy, hogy az előző jelszóra alkalmazza az MD4 függvényt, s a 128 bit output két felét összeadja.
- Az ezredik jelszót és a seed értéket átadja a szervernek, ezzel az inicializálás lezárult.

– **Hitelesítés**

- A szerver elküldi a seed értékét és a 999 számot a kliensnek, jelezve, hogy a 999. jelszó beírását kéri.
- A kliens megvalósítástól függően vagy kikeresi, vagy kiszámítja a 999. jelszót és elküldi a szervernek.
- A szerver a kapott jelszóra elvégzi az MD4 műveletet, s ha a kapott érték megegyezik a tárolt értékkel, a hitelesítést elfogadja, és a 999. érték váltja fel az ezredik értéket az adatbázisában. Legközelebb a 998. jelszót fogja kérni.

A módszer ereje a hash függvény tulajdonságaiban rejlik. Az ellenőrzés könnyű, hiszen a hash képzés gyors, viszont a következő jelszó megjóslásához fel kell törni a hash függvényt. A módszert az rfc 1760 specifikálja.

A protokoll problémái:

Az S/Key protokoll szótáras támadással támadható. Mivel a módszer nyílt, egy támadó, aki a vonalról leolvassa az üzenetváltást, próbálgatással ellenőrizheti a kliens jelszavára vonatkozó tippjét. Ezen a módon a szokásos, gyakori jelszavak kipróbálásán alapuló módszerek sikerrel alkalmazhatók gyengén választott jelszavak ellen.

A szerveren tárolt adatok védelme. A szerveren tárolt adatok (felhasználónév, számláló, seed, számlálóhoz tartozó jelszó) közvetlenül nem kompromittálják a jelszót, de megismerésük az előzőleg megismert támadási módszert teszi végrehajthatóvá, vagyis a fájlhoz való hozzáférést megfelelően kell beállítani.

Megszemélyesítés. Ha a támadó megszemélyesíti a szervert, és a következő, pl. 998. jelszó helyett a századikat kéri be, akkor ez veszélyes lehet. Ha ugyanis a kliens beírja a századik jelszót, akkor abból a támadó ki tudja számolni az összes száznál nagyobb sorszámú jelszót.

Belépésverseny. Ha a támadó nemcsak lehallgatni, hanem módosítani is képes a vonali forgalmat, elképzelhető, hogy módosítja a kliens válaszát, s így az nem tud belépni, viszont ezután a támadó be tud lépni, mert a rendszer (a sikertelen kísérlet miatt) ugyanazt a sorszámú jelszót kéri ismét.

Hash függvény támadása. A legtöbb rendszer az MD4 hash függvényt használja, amely azonban feltörhetőnek bizonyult (Hans Dobbertin: Cryptoanalysis of MD4, Fast Software Encryption, 1996), így a rendszer MD4 használata esetén alapjaiban rendült meg. Néhány rendszerben az MD5-öt használják, amely lényegesen biztonságosabb, azonban azt is érték már támadások Dobbertin részéről.

A protokoll egyszerű, hatékony, a felvetett problémák a sérülékeny hash kivételével kezelhetők. Megfelelő hash függvény használatát kell elérni az implementációban (pl. SHA-1, HMAC).

7.4. Kriptográfiai alkalmazások

A kriptográfiai alkalmazások adott biztonsági cél megvalósítása érdekében kidolgozott komplex rendszerek.

Ide tartoznak – több más mellett – pl. a következők:

- önálló biztonsági funkciókat megvalósító rendszerek (pl. elektronikus aláírási rendszerek; PGP),
- elektronikus fizetési rendszerek (pl. SET) védelmi alrendszerei,
- kommunikációs rendszerek (pl. GSM rendszer, elektronikus levelezés ...) védelmi alrendszerei.

7.4.1. Önálló biztonsági funkciókat megvalósító rendszerek

A kriptográfiai alkalmazások széleskörű csoportját alkotják azon rendszerek, melyek fő funkciója valamilyen védelmi megoldás felkínálása a felhasználóknak. Csak példaként említjük a jól ismert ***PGP*** /Pretty Good Privacy/ rendszert, mely aszinkron kommunikáció, valamint adattárolás védelmére is szolgál. A védelem nyilvános kulcsú kriptográfiára épül, de ellentétben az elektronikus aláírási rendszerekkel a nyilvános kulcsokat nem ún. CA-k (Certificate Authority-k) hitelesítik, hanem elsősorban az ún. bizalmi háló elvén épül fel a tanúsítványok elfogadása.

Az ***elektronikus aláírási rendszerek*** a magas szintű hitelesítést célul kitűző nagyon komplex alkalmazások, melyek az információs társadalom fejlődésének bizonyos szintjéhez már meghatározott jogszabályi, intézményi és technológiai rendszerben működtetendők. Magyarországon (az EU Direktíva[9] alapján) a 2001. évi XXXV. Törvény (valamint kiegészítő módosítása a 2004. évi LV törvény) az elektronikus aláírásra teremti meg a jogszabályi alapokat, melyekre végrehajtási rendeletek sokasága épült (pl. 1515/2001. Korm. Rendelet, 16/2001 MeHVM Rendelet stb.). Az elektronikus aláírási rendszer komplexitását már a résztvevők és funkciók sokasága is mutatja:

- az aláíró (küldő) és ellenőrző (címzett) közvetlen résztvevők (akiknek a törvény által nevesített ún. minősített aláíráshoz biztonságos aláírás-létrehozó eszközzel /SSCD: Secure Signature Creation Device/, valamint ellenőrzött, nemzetközi követelményeknek /CEN CWA 14170, CWA 14171/ megfelelő aláíró alkalmazásokkal kell rendelkezniük;
- a Hitelesítés-szolgáltatók CA-k, akik a nyilvános kulcsok hitelességét és érvényességét elektronikus tanúsítvány kiállításával (X509v3 tanúsítvány) tanúsítják, bonyolult és drága informatikai és szervezeti rendszerelőírásokkal; a résztvevőkre vonatkozó speciális regisztrációs eljárásokkal, az érvényességi listakezelés szigorú eléjárásaival (OCSP, vagy CRL), az időbélyegzési funkcióra vonatkozó előírásokkal ...

[9] „Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures," December 1999.

- a termékek, rendszerek, szolgáltatások megfelelőségét tanúsító szervezetek (Nemzeti Hírközlési Hatóság, Tanúsító Szervezetek), ezek feladatira jogszabályi előírások ...

7.4.2. Elektronikus fizetési rendszerek védelmi alrendszerei

Az internet szolgáltatások terjedésével egyre nagyobb szerepet kapnak olyan szolgáltatások, melyek kiteljesedéséhez pénzátutalásoknak is kapcsolódni kell. A bonyolult informatikai rendszer része az elektronikus fizetési alrendszer (EPS), mely lehet interneten keresztüli fizetés (pl. SET), vagy ún. elektronikus pénztárca, digitális készpénz[10].

Internet alapú fizetésre iKP néven került kifejlesztésre a – PKI alapú – Internet Keyed Payments Protocol[11]. Mondex néven került kifejlesztésre egy SmartCard alapú fizetési rendszer[12].

SET: Secure Electronic Transaction

A SET kifejlesztését a 90-es években a VISA és a MasterCard végezte azért, hogy elektronikus fizetéshez általánosan elfogadott, biztonságos fizetési módszer álljon rendelkezésre.

A SET résztvevői:

Kártyatulajdonos az elektronikus kereskedelemben a ***fogyasztó***, illetve a ***vevő*** a kereskedőkkel a számítógépeiken keresztül kerülnek kapcsolatba. A kártyatulajdonos egy, a kibocsátótól származó fizetési kártyát használ. A SET biztosítja, hogy a kártyatulajdonos és a kereskedő kapcsolata során a fizetési kártya számlájára vonatkozó információk titkosak maradnak.

Kibocsátó egy olyan pénzügyi intézmény, amely egy kártyatulajdonos számára számlát hoz létre. A kibocsátó garanciát vállal a kártya-használat szabályainak megfelelő tranzakciók tényleges kifizetéséért.

Kereskedő termékeket vagy szolgáltatásokat nyújt fizetség fejében. A SET segítségével a kereskedő a kártyatulajdonosoknak biztonságos elektronikus együttműködést tud nyújtani.

Számlagyűjtő egy olyan pénzügyi intézmény, amely számlát tart fenn egy kereskedő részére, és gondoskodik a fizetési kártya felhatalmazásokról és a kifizetésekről.

Fizetésszervező egy a Számlagyűjtő által működtetett eszköz vagy egy kijelölt harmadik partner, amely a kereskedők fizetéssel kapcsolatos értesítéseit elvégzi, beleértve a kártyatulajdonosok fizetési instrukcióit.

A SET az alábbi kiemelkedően fontos üzleti követelményeket célozta meg:

[10] A rendkívül szerteágazó területről összefoglalás olvasható pl. `http://wwww.w3.org/ECommerce/roadmap.html` címen.

[11] M. Bellare, J. A. Garay, R. Hauser,...: iKP – A Family of Secure Electronic Payment Protocols, 1995., ld. még `http://www.zurich.ibm.com/security/past-projects/ecommmerce/iKP.html`.

[12] `http://www.mondex.com`.

1. A fizetési információk bizalmasságának és a fizetési információkkal együtt továbbításra kerülő rendelési információk bizalmasságának megőrzése.
2. Minden továbbításra kerülő adat integritásának biztosítása.
3. Hitelesítés biztosítása, amely egy kártyatulajdonost a fizetési kártya számlához tartozó legitim felhasználóként hitelesíti.
4. A kereskedő hitelesítésének biztosítása, miszerint ő egy fizetési kártyás tranzakciókat elfogadó személy vagy cég, aki kapcsolatban áll egy számlagyűjtő pénzügyi intézménnyel.
5. Megvédeni az elektronikus kereskedelem tranzakcióinak minden résztvevője érdekeit.
6. Egy protokoll létrehozása, amely nem függ az átvitel biztonsági mechanizmusától, és nem akadályozza azok használatát.

7.4.3. Kommunikációs rendszerek védelmi alrendszerei

Nagyon sokféle kommunikációs rendszerhez fejlesztettek védelmi alrendszereket. Ide sorolhatjuk az internet használat kommunikációs védelmi alrendszereit (melyek közül a SSH, SSL, TLS, IPSEC a protokollokról szóló fejezetben szerepeltek és másokat, pl. HTTPS), a fizetős televíziózás kódolását, az elektronikus levelezés biztonsági alrendszereit, a Wireless technológia védelmi alrendszerét, stb.

7.4.3.a.

A nagyvállalati belső hálózatokban nagyon elterjedt ***Lotus Notes levelező rendszer***.
A rendszer védelmi szintjei:

- az azonosítás eszközei,
- szerver szintű biztonsági eszközök,
- adatbázis szintű biztonsági eszközök,
- adat szintű biztonsági eszközök.

Az *azonosítás* eszköze a Notes ID.
A Notes ID (azonosító) azonosítja a felhasználókat és szervereket. Az ID a felhasználó vagy a szerver regisztrálásakor keletkezik.

Szerver szinten

- port szintű hozzáférés-védelem és port szintű titkosítás; valamint
- szerver dokumentumvédelem (hitelesítés, elérési korlátozások, és felhasználói, „ügynöki") futtatási szabályok vannak.

Az adatbázis védelem titkosítással és hozzáférési listákkal (ACL) megoldott.

Titkosítás a Notes-ban

A Notes az adatok titkosítására az RC2 és RC4 algoritmusokat, a kulcsok kezelésére az RSA algoritmust használja.

Az RSA módszernél minden felhasználóhoz egy egyedi titkosító kulcspárt rendel: egy saját kulcsot, amely csak a felhasználói ID fájlba kerül be, és egy nyilvános kulcsot, amely a felhasználó ID fájlján kívül a közös címjegyzékbe is bekerül, ahol nyilvánosan elérhető.

Ha a felhasználók elfelejtik a jelszavukat vagy elvesztik az ID fájljukat, és nincs róla back-up másolatuk, az összes levél, amit a saját kulcsukkal titkosítottak, örökre elveszett.

Titkosítani lehet hálózati portokat, adatbázisokat, levélfájlokat, dokumentumokat és mezőket.

Aláíráskor a Notes elkészíti a dokumentum 128 bites „ujjlenyomatát", és ezt titkosítja a felhasználó privát kulcsával és hozzácsatolja a dokumentumhoz.

Egy aláírt dokumentum olvasása nincs korlátozva, de ha valaki módosítja a dokumentumot, akkor már nem fog egyezni a titkosított „ujjlenyomat" a dokumentummal, és a Notes nem fogadja el hitelesnek az aláírást.

Amikor egy felhasználó titkosított levelet küld, akkor a Notes generál egy véletlen kulcsot és ezzel titkosítja a levelet. Ezután a Notes a címzett publikus kulcsával titkosítja a véletlen kulcsot. Végül elküldi a titkosított kulcsot a titkosított levéllel együtt. A címzett a saját privát kulcsával tudja ezt a levelet elolvasni.

Adatbiztonság

Az adatok védelmére a Notes egy kombinált rejtjelező rendszert alkalmaz. A különösen érzékeny információkat, az elektronikus pecséteket és aláírásokat, valamint a használat közben generált egyszerűbb kulcsokat az RSA rejtjelező rendszer védi.

RSA alkalmazási területek a Lotus Notes-ban

- Levél és dokumentum kulcs titkosítás, 630 bit.
- Adatbázis kulcs titkosítás, 630 bit.
- Levél, dokumentum és mező digitális aláírás, 630 bit.
- Felhasználó azonosítás, 630 bit.

RC2 és RC4 titkosítás

A Lotus Notes az amerikai és kanadai felhasználók részére 64 bites kulccsal végzi az RC algoritmusú rejtjelzést, míg a nemzetközi felhasználók részére 40 bites rejtjelzést biztosít. Valójában ez utóbbi is 64 bites rejtjelzéssel történik, de a kulcs egy részét (20 bitet) az Amerikai Egyesült Államok importkorlátozó törvényének eleget téve az NSA rendelkezésére bocsájtja. Információink

szerint lehetőség van az erősebb amerikai változat beszerzésére külföldi ügyfeleknek is külön kérés és elbírálás alapján.

RC2 és RC4 felhasználási területek

- ID fájl titkosítása.
- Levél, dokumentum és mező titkosítás, RC2 (64 bit).
- Adatbázis titkosítás, RC2 (64 bit).
- Kapcsolat felépítés, RC2 (64 bit).
- Adatkapcsolati kulcs, RC2 (40 bit).
- Egyéb titkosítási kulcsok, RC2 (64 bit).

Az alábbiakban rövid áttekintést adunk a mobil kommunikáció védelméről:

7.4.3.b. A GSM-hálózatok alkalmazás-szintű biztonsági megoldásai

A GSM egyik kriptográfiai primitívjéről (A5/1 algoritmusról) már írtunk. Több más primitív (A3, A8) is szerepet játszik, de az egész rendszer sokkal komplexebb az algoritmusoknál. A biztonság elemzése is kitekintést igényel a teljes kriptográfiai alkalmazásrendszerre (a gyengeség is ott mutatkozik).

A GSM hálózatban az előfizetőt az úgynevezett IMSI-szám (International Mobile Subscriber Identity – nemzetközi mobilelőfizető-azonosító) azonosítja. A rendszer a telefonszám helyett ezt használja azonosításra a kommunikáció során. Az IMSI a telefonban levő smart kártyán – SIM előfizetői kártyán – tárolódik. Az IMSI-t a telefon bekapcsoláskor elküldi a központnak, s onnan egy TMSI-nek nevezett TMSI (Temporary MSI – ideiglenes MSI) számot kap vissza, s a továbbiakban ezt használják az előfizető azonosítására. Ezt rendszeresen cseréli a rendszer. A fentiek miatt az esetleges lehallgatónak a teljes folyamatot végig kell kísérnie az előfizető azonosításához.

Az azonosítás mellett híváskezdeményezéskor az előfizető hitelesíti magát, majd egy közösen kialakított egyszeri kulccsal védve rejtjelezik is az adatcsatornát, pontosabban annak a telefon és a bázisállomás közötti kapcsolatát. Mindennek kriptográfiai alapját a SIM kártyán, a kártya operációs rendszere által védve tárolt KI kulcs adja.

A hitelesítéshez a telefon a központtól kap egy 128 bites véletlen elemet. (RND) Ebből és a KI kulcsból a kártya az A3 algoritmus segítségével számol ki egy választ, s annak első 32 bitjét küldi vissza. A központ is ismeri a KI kulcsot, ezért ugyanazt a számítást elvégezve ellenőrizni tudja a kapott értéket.

Ugyanebből az RND számból és a KI kulcsból egy A8 nevű algoritmussal készít a kártya egy K 64 bites kulcsot. Sajnos annakidején COCOM előírások miatt ennek a kulcsnak tíz bitjét nullára állították be, s ez a kulcs ezért azóta is 54 bites. Ez a kulcs lesz a kommunikáció titkosításához felhasznált kulcs. A titkosítás csak a „levegőben" működik, kicsit egzaktabbul: a telefon és a

bázisállomás között. A kommunikációt az A5 algoritmussal titkosítják. Az A5 algoritmust a mobiltelefonban implementálják. Az A5-nek két verziója van, az egyiket a fejlett országokban – A5/1, ez az erősebb –, míg a másikat Ázsiában használják.

Az A5 algoritmussal szemben az A3 és A8 algoritmusok a kártyán kerülnek implementálásra. Az A3 és A8 algoritmusok a szolgáltatóhoz kapcsolódnak, azaz akár minden egyes szolgáltatónál más-más algoritmust használhatnak az A3 és az A8 helyén, ez a roamingot nem zavarja.

Az A3 és A8 szerepében sok szolgáltató használja a COMP128 algoritmust.

A GSM és a harmadik generációs UTSM rendszer, valamint a mobil telefonról történő internet elérést segítő WAP /Wireless Application Protocol/ szolgáltatás biztonsági architektúrájáról ajánljuk még a [2]-ben található áttekintést.

7.4.3.c. Wireless technológia

A számítástechnikai hálózatok vezeték nélküli kommunikációja sok felhasználási környezetben nagyon kényelmes, gyorsan terjed. Több más védelem mellett (SSID Bradcasting tiltása, hálózati azonosító /MAC cím/ szerinti szűrés) lényeges kriptográfiai elem a hálózati forgalom titkosítása.

A kezdeti rendszerekben használt titkosítás a WEP (Wired Equivalent Privacy) RC4 eljárást használ, legtöbb esetben 40 bites kulccsal. A kulcs és az ebből képzett inicializáló vektor segítségével kódolnak, de az IV inicializáló vektor csak 24 bites. A kulcs statikus, az IV változik, de rövid, ez az egyik alapja a WEP támadásának.[13]

A gyengeségek kiküszöbölésére fejlesztették ki a WPA (WI_FI Protected Access) eljárást, mely változó kulcs, megnövelt inicializáló vektor mellett több más védelmi elemet is tartalmaz.

A legújabb továbbfejlesztés a WPA2. Ez AES algoritmust használ kódolásra, nagyobb kulcsméretekkel.

7.5. Kriptográfiai alkalmazásokat futtató informatikai rendszerek védelmi alrendszerei

Természetesen a komplex biztonsági (kriptográfiai megoldások hatalmas együttesét) tartalmazó rendszerek (pl. LotusNotes, PGP, elektronikus aláírási rendszer ...) futtatása is általában informatikai rendszerkörnyezetben történik.

A teljes kriptográfiai rendszer biztonsága függhet az általános informatikai rendszer biztonságától (az operációs rendszerek, adatbázis-kezelők, levelező rendszerek, internet és egyéb informatikai szolgáltatók gyenge biztonsága veszélyeztetheti a jól megalapozott védelmet is, pl., ha a gépemben már egy feltételezett támadó az én jogosultságaimmal kezelheti kriptográfiai primitíveimet, kulcsaimat, a sémákat, protokollokat, kriptográfiai alkalmazásokat).

[13] A WEP támadásáról, valamint Budapesten használt rendszerek felméréséről olvashatunk a `www.saveas.hu` honlapon cikkeket.

Ez a terület nagyon szerteágazó: a technológiai (operációs rendszerek, adatbázis-kezelő rendszerek, alkalmazások) biztonsági kérdéseken túl a szervezeti, humán biztonsági kérdések széles skálája is beletartozik.

8. A kriptográfiai primitívek különböző biztonsági megközelítései

A kriptográfiai primitívek biztonságát többféleképpen csoportosíthatjuk:

- *Bizonyítottan nem biztonságos rendszerek.* Ennek két osztályát lehet megkülönböztetni:
  - valamilyen reális feltétel mellett – feltörhető a rejtjelrendszer /ld. pl. David Kahn: The Codebreakers, New York, 1996 c. könyvét, vagy a DES-re vonatkozó támadások leírásait/;
  - publikált támadási technikák vannak, amelyekkel a kulcstér teljes kipróbálásánál hatékonyabban támadható a rendszer, ugyanakkor a módszer használata a legtöbb esetben reálisan nem kivitelezhető (pl. összetartozó 2^{47} db nyílt-rejtjeles blokkpár ismerte a DES kulcs meghatározásához differenciál kripto-analízis módszerével, mely messze van az átlagos eszközökkel kivitelezhető támadástól, de szakmailag a rendszer gyengeségére utal).

- Eddig még nem törték fel az adott (paraméterű) rejtjelrendszert, de *nem kizárható az elmélet és a támadási technikák olyan fejlődése, mellyel az a gyakorlatban is kivitelezhetővé válik.*

  Ezen biztonsági osztály között megkülönböztethetőek az alábbiak:
  - Titkos, vagy nyilvánosságra került, de nem kellően bevizsgált rendszerek.
  - A kriptográfiai szakmai tudományos közösség által különböző támadási technikákkal alaposan vizsgált rendszerek, melyek eddig ellenálltak az ismert technikáknak (pl. TripleDES, IDEA, AES). Ennél az osztálynál érdemes megjegyezni, hogy mit értünk azon a kifejezésen, hogy *egy algoritmus ellenáll az adott támadási technikának.* Például az AES algoritmus is támadható a differenciál kripto-analízis (vagy lineáris kripto-analízis) módszerével, de a támadás igényelt lépésszáma lényegesen nem csökkenti a teljes kulcskipróbáláshoz szükséges lépésszámot. Ekkor azt mondjuk, hogy az algoritmus ellenáll ennek a támadásnak. (Nem hatékonyabb ez a támadás, mintha kipróbálnánk az összes kulcsot. Ez a kijelentés például nem igaz a DES algoritmusra.)
  - Klasszikus, ismert matematikai problémákra visszavezethető kriptográfiai rendszerek, mely problémákra mind a kriptográfiai, mind a matematikai szakmai tudományos közösség még nem talált hatékony megoldó

algoritmusokat (ezek külön osztályát alkotják azok az algoritmusok, melyekre visszavezetve a kriptográfiai rendszert, annak feltörése egyszerre az összes nehéznek tartott (NPC) matematikai probléma megoldását is eredményezné). Ez esetben a kriptográfiai paraméterek méretétől függ a rendszer feltörhetősége. Megadható olyan mérete a paramétereknek, melyek mellett a fenti értelemben biztonságosnak tekinthetőek az algoritmusok (pl. RSA rendszer).

- *Bizonyítottan nem lehetséges a rendszer hatékony – algoritmikai – támadása*, azaz nem várható olyan tudományos eredmény és számítástechnikai fejlődés, melyek lehetővé tennék a támadást (pl. OTP).

8.1. Információelméleti megközelítések, bizonyított biztonság

Shannon adta eddig az egyetlen teljes, információelméleti megközelítésen alapuló bizonyítási módszert az ún. One-Time-Pad /OTP/ rejtjelzésre, mellyel bizonyította, hogy egy esetleges támadó korlátlan erőforrásai mellett sem képes visszaállítani a titkosított üzenetet.

Elméletileg fejthetetlen algoritmus *(tökéletes rejtjelező)*, amelyre teljesül, hogy a kódolt üzenet (y) és az eredeti üzenet (x) közötti kölcsönös információ 0, vagyis $I(x, y) = 0$.

Shannon a fentieknek megfelelő (OTP) kódolás alatt egy valódi véletlen kulcssorozat moduló (jelkészlet) additív felhasználását értette. A One-Time Pad kifejezés onnan származik, hogy régebben a véletlen sorozatokat noteszlapokra írták le („pad"), és minden lapot (kulcselemet) csak egyszer lehetett felhasználni („one-time").

Kiemelt jelentősége van ennek az eredménynek az alábbiak miatt

- A bizonyítási módszer lehetővé teszi az algoritmus fejthetetlenségére vonatkozó állítás feltételrendszerének pontos meghatározását:
  - A kulcssorozat egyenletes eloszlású valódi véletlen, legalább a rejtjelzendő üzenet hosszúságú, karakterkészletével megegyező karakterkészletű (ha pl. a titkosítandó karaktereket bitsorozattá konvertálják titkosítás előtt, akkor a kulcssorozat is bit-sorozat, az additív művelet a mod2-es összeadás; ez a leggyakoribb felhasználási mód).
  - A támadó semmilyen információval nem rendelkezhet a kulcssorozatról (pl. „side-information" nem állhat rendelkezésére, például a rejtjelzés folyamatából).
  - A titkos kulcsok nem juthatnak egy támadó birtokába (őrizni kell).
  - Minden titkos kulcselemet csak egyetlen egyszer szabad felhasználni titkosításra.
- A bizonyítási módszer lehetővé teszi annak megfogalmazását, hogy mit is jelent a támadó bizonyított sikertelensége:

- A támadó az elfogott titkosított üzenetből – a titkosított nyílt üzenet hosszán kívül – semmilyen következtetést nem tud levonni arra vonatkozóan, hogy melyik üzenet a rejtjelszöveg ősképe (bármelyik nyílt szöveg, vagy a szöveg részlete egyenlő valószínűségű).
- Vonatkozik a fejthetetlenség bármilyen inputból kiinduló támadási módszerre /Ciphertext only attack, plaintext and corresponding ciphertext attack, Selected ciphertext and corresponding plaintext attack, Adaptive chosen-ciphertext attack, Selected plaintext and corresponding ciphertext attack/ és bármely algoritmikus módszerre /Brute force attack, Differential criptanalysis attack, Linear criptanalysis attack, .../, abban az értelemben, hogy feltételezett nyílt szöveget /a hosszán kívül/ sem megerősíteni, sem további nyílt szövegekre következtetni nem lehet. De nem feltétlenül vonatkozik a titkosítás műveleteinek kimérését megvalósító támadásokra /Power analysis attack, .../, melyek az előző pontban említett feltételek között kell kizárni.

- Kiemelkedő gyakorlati alkalmazásokban használják, ahol az alkalmazás egyrészt megköveteli a legerősebb titkosítás használatát, másrészt az alkalmazási nehézségek (hatalmas mennyiségű véletlen kulcs gyártása, szétosztása, védett tárolása és felhasználása) elfogadhatóak.

- Az OTP titkosítás gyakorlati problémáinak kezelésére próbálkoztak „pszeudovéletlen" sorozatok felhasználásával. Erre természetesen nem igaz az információelméleti biztonság bizonyítása. Azonban ha egyéb módszerekkel valószínűsíthető, hogy az esetleges támadó nem tudja a generált sorozatot megkülönböztetni a valódi véletlen sorozattól /distinguish attack), akkor már igaz lenne, hogy a nyílt szöveget sem tudja visszaállítani. Ekkor a biztonság mértékét a megkülönböztethetőség garanciális szintje adja, mely nem információelméleti, hanem a következő pontokban tárgyalt redukciós megközelítésen alapul.

- Ez az egyetlen hagyományos titkosítási eljárás, melyre teljesül, hogy ha a támadó eszköztárába kerülnek hatékony kvantumszámítógépek, akkor is ellenáll, a fent részletezett „fejthetetlenségi" fogalomnak megfelelően.

8.2. Redukciós-bonyolultságelméleti biztonság

A kriptográfiai primitívekre számos algoritmust, algoritmusok együttesét alkalmazzák, melyek egy részének biztonsága (és tervezési elve) közös matematikai problémákra vezethető vissza.

A tervezés során biztonsági szempontból megkülönböztethetők

- Intuitív módszerek (legyen minél „bonyolultabb");
- jól struktúrált, klasszikus matematikai problémákra visszavezethető algoritmusok.

Az első típusra példa a DES (Data Enctrition Standard) blokkrejtjelző, ahol utólag dolgoztak ki tervezési elveket (pl. a véletlen permutációk, S-dobozok tervezésére, ezekről részletesen olvashatunk [2]-ben).

A második típusra említhetjük az LFSR-ekből felépített rendszereket, de ez az elv leginkább a nyilvános kulcsú rendszerekre jellemző, a felhasznált klasszikus matematikai problémák, melyekkel próbálják visszavezetni a kriptográfiai biztonságot:

- **IFP** (Integer factoring problem);
  - **RSA_IFP:** A probléma külön osztályaként kezelik az RSA struktúrán alapuló problémát, mely két, ismeretlen nagy prím szorzatának faktorizációját jelenti;
- **SQROOT** (Square roots modulo n): adott egy n összetett szám, meghatározandó x:

$$x^2 \equiv a \pmod{n}$$

- **DLP** (Discret logarithm problem):

Adott egy p prím és α generátor eleme Z_p^* -nak, valamint $\beta \in Z_p^*$, meghatározandó x, $0 \leq x \leq p-2$, melyre $\alpha^x \equiv \beta \pmod{p}$

- **GDLP** (Generalzized discrete logarithm problem): olyan DLP, melynél a mod p osztály helyett egy ciklikus csoportból indulunk ki: Adott egy G véges ciklikus csoport, rendje: n, és α generátor eleme G-nek, valamint $\beta \in G$, meghatározandó x, $0 \leq x \leq n-1$, melyre $\alpha^x \equiv \beta$
  - **DHP, GDHP:** A fenti problémák külön osztályaként kezelik a Diffie–Hellman protokollon alapuló problémákat:
- **SSP, (KSP):** Subset-sum (Knapksacks) problem: adottak $\{a_1, a_2, \ldots, a_n\}$ súlyok, S pozitív egész, meghatározandó (ha létezik) olyan részhalmaza a súlyoknak, melyek összege: S.

(A felosztást részletesebben ld.: [5] 3. Number-Theoretic Reference Problem c. fejezetében.)

A fenti problémáknál a kriptográfiai algoritmust egy nehéznek gondolt feladatra vezetik vissza. A visszavezetés alatt azt kell érteni, hogy olyan lépések sorozatát kell igazolni, hogy amennyiben a titkosító algoritmust valaki fel tudná törni, a leírt lépésekkel a nehéznek vélt matematikai probléma megoldását is megkapná. Olyan nehéz feladatot szoktak választani, amelyre nem ismert hatékony – azaz polinomiális időben végrehajtható – megoldás. Ebből következik, hogy kellő méretű (kulcsterű) probléma mellett gyakorlatilag kivitelezhetetlen lesz a támadás.

Polinom idejű algoritmus (a bemenő paraméterek méretének függvényében):

> Létezik olyan c pozitív egész szám, hogy a legfeljebb n bites egészekkel végzett bitműveletek száma $O(n^c)$, $f(n)$ és $g(n)$ két pozitív értékeket felvevő $fv.$, $f = O(g)$, ha létezik olyan k konstans, hogy elegendően nagy n-re $f(n) \leq k^* g(n)$.

A kutatások az ún. egyirányú függvényekből indultak ki:

Az $f\{0,1\}^* \to \{0,1\}^*$ *fv.* egyirányú, ha a következő két feltétel teljesül:

1. Könnyű kiszámítani: létezik olyan hatékonyan (polinomiális időben) kiszámítható A algoritmus, mely x bemenetre az $f(x)$ kimenetet adja $(A : x \to f(x))$.
2. Nehéz invertálni: tetszőleges hatékony A' algoritmus $(A' : f(x) \to x)$, és tetszőleges $p(n)$ polinom, valamint elegendően nagy n esetén, véletlenszerűen választott x ősképek esetén:

$$Pr\Big\{f\big(A'[f(x), 1^n]\big) = f(x)\Big\} < 1/p(n).$$

Az egyirányú függvények egy osztályát alkotják a „csapda egyirányú függvények", melyeknél valamilyen ún. csapda információ birtokában az invertálás mégis könnyű (polinomiális időben végrehajtható). Ez a csapda tulajdonság szükséges ahhoz, hogy az üzenet címzettje (egy csapda-információ birtokában) a titkosított üzenetből visszakaphassa az eredeti üzenetet.

Tipikus példája ennek az RSA eljárás, ahol a csapda információ a két prímszám, melynek szorzata adja a nyilvános modulust. Amennyiben egy támadó nem ismeri a két prímszámot (csapdát), akkor az üzenet visszafejtése neki „nehéz" probléma, míg a címzettnek könnyű (a modulus méretében polinomiális (köbös) időben végrehajtható).

A nehéz (NP: non-polinomial) problémák külön osztályát alkotják az ún. NP-teljes problémák, amelyek közé azon problémák tartoznak, amelyekről bizonyították, hogy ha megoldhatók lennének polinomiális időben, az összes nehéz probléma megoldható lenne.

A faktorizáció problémája (IFP, RSA_IFP) bár nem NP-teljes probléma, mégis oly sokan vizsgálták évezredek óta, hogy a nehézsége elfogadott.

Még jobbnak tűnne NP-teljes problémát választani. Sikerült is ehhez az osztályhoz tartozó „csapda" konstrukciót választani, mely az SSP problémához csatlakozó **KSP:** Knapksack titkosító eljárás. (Ebben a csapda információt egy szupernövekedő sorozat biztosítja):

Legyenek $a_1, a_2, \ldots, a_n$ „súlyok"

$\underline{x_i} \in \{0,1\}^n$ bitvektor

$$S = \sum_1^n x_i a_i$$

Ekkor S, a_i-k ismeretében „nehéz" x_i-ket meghatározni, ugyanis a

KnapSack (hátizsák) probléma NP-teljes, ha a_i-k véletlenszerűek.

Viszont könnyű a meghatározás, ha a_i-k „szupernövekedő" sorozatot alkotnak (pl. kettes számrendszer).

Az ennek megfelelő kriptográfiai primitív:

Legyen $c_1, c_2, \ldots, c_n$ szupernövekedő sorozat:
és legyen $M > \Sigma c_i$, $(M, U) = 1$ (csapda-paraméterek).
Legyen $a_i \equiv U\ {}^*c_i \pmod{M}$, a_i-**k a nyilvános kulcsok.**

Titkosítás: $x = (x_1, x_2, \ldots x_n)$ az üzenet, a rejtjelzés: $C = \sum_{k=1}^{n} x_k\ {}^*a_k$.

Megoldás (a C rejtjeles sorozatból): Legyen $W : W\ {}^*U \equiv 1 \pmod{M}$-et kielégítő konstans, és

$$C\ {}^*W = \sum_{k=1}^{n} x_k\ {}^*(a_k\ {}^*W) = \sum_{k=1}^{n} x_k\ {}^*c_k,$$

ahonnan az x meghatározható.

Ez az eljárás mutat rá az NP-elméleti megközelítés biztonsági kockázataira, mivel a fenti problémát – az elméleti megközelítés ellenére – lehet támadni. Ennek okai a következőek:

- Az NP-elmélet „csak" a legrosszabb eset megoldásával foglalkozik, ugyanakkor az adatvédelemben nyilvánvalóan nem elegendő, ha sok titkosított üzenetből a támadó néhányat (a számára legrosszabb eseteket) nem tudja visszaállítani;
- Adatvédelemben nem megfelelő, ha közelítő megoldásokat, vagy a megoldás szempontjából ekvivalens megoldások közül egyet meg lehet hatékonyan – polinomiális időben – keresni.

A fenti problémák miatt titkosításra a hátizsák probléma nem elégséges biztonságú. A gyengeség (támadás) részletesen olvasható pl. [6] c. könyvben (itt a módszer javítási kísérleteinek támadásairól is olvashatunk). A kritikai megjegyzések nem az egész NP elmélet használhatóságára, hanem konkrétan a hátizsák algoritmusra vonatkoznak, ugyanakkor rámutatnak az alkalmazással kapcsolatos óvatos (átgondolt) megközelítés fontosságára.

Az NP elmélet kriptográfiai felhasználásáról, a bizonyítási technikákról részletesen olvashatunk [2] könyv IV. részében.

Itt olvashatunk pl. részletes meghatározásokat, eredményeket az alábbi témákról:

- a keménybit fogalma (mely a részleges kulcsfejtés kizárásához szükséges);
- a véletlen algoritmikus megkülönböztetetőség vizsgálata (szükségességét ld. a OTP-nél kifejtett meggondolásnál);
- az ún. szemantikai biztonság;
- üzenet-megkülönböztethetetlenség biztonság;
- rejtjeles szöveg módosíthatatlanság biztonság;
- a biztonságos kriptográfiai függvény-tervezés elvei; ...

fogalmakról, ezekkel kapcsolatos eredményekről.

8.3. Kalkulációs biztonság

A kalkulációs biztonság nem aszimptotikus megközelítését adja a biztonságnak, hanem, hogy ismereteink (gyakorlati vizsgálataink, jobb esetben bizonyítási módszereink) szerint nincs olyan algoritmus és hozzá rendelhető reális (létező eszközzel a futási idő életünkben bevárható) számítási kapacitás, amellyel adott támadás sikerrel elvégezhető lenne.

Ilyen értelmű kalkulációs biztonságot nyújt például az AES algoritmus az ismert támadásokkal szemben (de csak ezekkel szemben!), amelynél bizonyították, hogy pl. a differenciál kripto-analízis, illetve a lineáris kripto-analízis nem lényegesen hatékonyabb a teljes kipróbálás módszerénél, melynél például 128 bites kulcsot feltételezve (ez a minimális használható kulcsméret) a szükséges fejtési lépésszám: c^*2^{128}, ahol $c > 1$ egy kulcsra végzett AES művelet.

Az ismert támadási algoritmusok élesítésével konkrét becsléseket keresünk az algoritmusok szükséges lépésszámára. Így kapjuk, hogy például ez a biztonságfogalom nem teljesül a DES algoritmusra, sőt adott számítási kapacitásokat (memória-műveleti igény párt tekintve) a kétszeresen használt DES-re sem.

A kalkulációs biztonság elemzéséhez két tényező vizsgálata szükséges:

- Strukturális vizsgálatok (a teljes kipróbálás műveletigényét lényegesen lecsökkentő eljárások kutatása);
- Ezek műveletszámának (vagy a teljes kipróbálás műveletigényének) összevetése a számítástechnikai lehetőségekkel.

A fentiekre részletes elemzéseket mutatunk a 8.3.3. pontban az RSA kalkulációs biztonságának áttekintésekor, míg a DES-re vonatkozó megállapítások a második szempont szerint is kijelenthetők (a kétszeres DES nem megfelelő biztonságához szükséges kripto-analitikai megállapításokat is tenni).

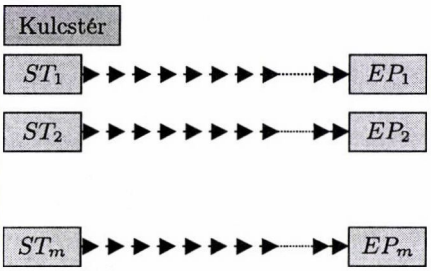
Az alábbiakban példaként néhány olyan eljárást szemléltetünk, amelyek nem felelnek meg a kalkulációs biztonság követelményének, azaz annak, hogy az ismert összes algoritmusra a reálisan létező számítástechnikai kapacitásokkal ne lehessen belátható idő alatt a rejtjelrendszert feltörni.

8.3.1. A DES és a kalkulációs biztonság

A DES-sel kapcsolatban már publikálása után nem sokkal Hellman egy speciális kimerítő támadás lehetőségét vázolta fel (akkoriban olyan technikai eszközparkkal, amelynek költségét akkor 20 millió dollárra becsülte), de kellő (hosszadalmas) prekondicionálási fázis után bármilyen, a feltételeknek eleget tévő rejtjelzett üzenetet hatékonyan – hamar vissza tudott állítani. (Támadási módszere a choosen plaintext attack elvéből indult ki.) Egy blokkra kellett ismerni az összetartozó nyílt-rejtjeles párt (azaz 8 byte-ot), és eszközével az összes olyan kódolt üzenet gyors fejtését lehetővé tette, amelyben az adott nyílt blokk szerepel, és tudjuk (vagy kipróbáljuk), hogy melyik a neki megfelelő rejtjeles blokk.

Hellman módszerének továbbfejlesztése – az azóta nagyon sok rejtjelrendszerre élesített – ún. ***Time-Memory Trade-Off Attack*** (azaz a rendelkezésre álló memória és a használható idő olyan megosztása, mely mellett egy prekondicionálási

szakaszban a kulcstér jelentős részét kipróbálva, ügyesen az eredményeket letárolva, a gyakorlati fejtés gyorsan végrehajtható.



A módszer lényege:
A kulcstérben véletlenszerűen választva „m" db. belépési pontot (ST), „ügyesen" megtéve „t" lépést, le kell tárolni a belépési – (ST) és végpontokat (EP), a végpontok (EP) szerint rendezve az m^*t^*2 (blokkméretnyi) tárban.
Ismeretlen kulcsú rejtjelzés kulcsát (mely valamelyik soron van) átlag $c^*t/2$ rejtjelző, majd $\log_2$m kereső és újból $c^*t/2$ rejtjelző lépésben megtalálhatjuk P valószínűséggel. (P függ a (t, m) pártól, ajánlott választás a $t^*m^2 = N^2$, ahol N a kulcstér mérete).

Például megemlítünk két ezzel foglalkozó előadást:

Philippe Oechslin: *Making a Faster Cryptanalytic Time-Memory Trade-Off,* **CRYPTO 2003 konferencián elhangzott előadásában** a DES törésre 1980-ban Martin Hellman által leírt (ismert P_0, C) párból induló, a kulcstéren lépegető, letároló eljárását gyorsította meg.

A cikkben példát mutat az MS Windows hash jelszótörésére 140 GByte tárhely felhasználásával, az összes alfanumerikus lehetséges hash értékre néhány másodperc alatt.

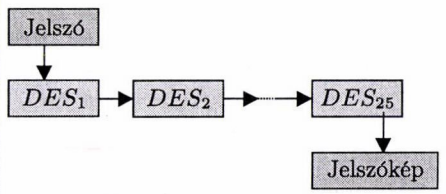
Alex Biryukov and Adi Shamir **Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers** c. ASIACRYPT 2000 konferencián elhangzott előadásában stream-cipher-ekre terjesztik ki ezt a hatékony támadási módszert. Eredménye szerint, ha a támadónak elégséges „d" számú output adat kipróbálása a döntéshez, akkor a $t^*m^{2*}d^2 = N^2$ paraméterválasztás mellett működik a támadás, ahol $d^2 \leq t \leq N$.

Külön említésre érdemes az EFF csoport DES-fejtő projektje (ld: `http://www.eff.org/descracker/`.) Az Electronic Frontier Foundation 1997-ben kezdte a projektet. Speciális FPGA (Field Programmable Gate Array) alapon épített rendszer keresi a DES kulcsokat. Ebben a szükséges hardver modulokról és az eredményes futtatás várható idejéről a következő táblázat szerepel:

Egység	Egységek száma a következő egységben	Másodpercenként kipróbálható kulcsok m száma	Az átlagos keresési idő napokban
Alapegység	24	2,500,000	166,800
Chip	64	60,000,000	6,950
lap	12	3,840,000,000	109
doboz	2	46,080,000,000	9,05
EFF DES Cracker		92,160,000,000	4,524

Azaz a speciális fejlesztésű célhardver minden chipje 24 kereső egységet tartalmaz. 64 ilyen chipet helyeznek el 12 panelen, melyből kettővel számolva másodpercenként 92,2 millió kulcs kipróbálására nyílik lehetősége, így az átlagos megoldási idő 4,5 nap lesz.

A DES támadását olyan szintre emelték, hogy több DES alapú más alkalmazás támadásánál sikerrel jártak. Példa erre a UNIX által használt password egyirányú képének képzése.
Az algoritmust a jobboldali ábra szemlélteti:



Pl. A SHARCS /Special Purpose Hardware for Attacking Cryptographic Systems konferencián 2005 februárjában Párizsban N. Mentes, L. Batina, B. Prencel, I. Verbauewhede: Cracking Unix Password using FPGA Platforms c. előadásukban hatékony célhardvert mutattak be a 25-szörös DES jelszókép ősképének visszaállítására.

A fentiek alapján megállapítható, hogy – bár egy átlagos felhasználó nem rendelkezik elegendő kapacitásokkal, azonban – a *DES nem felel meg a kalkulációs biztonság követelményeinek.* Ugyancsak nem megfelelő a kétszeres DES használat annak ellenére sem, hogy a felhasznált 112 bites kulcshossz önmagában elegendő lenne a kalkulációs biztonság kimondásához (nincs jelenleg elegendő számítástechnikai kapacitás 2^{112} eset végigfuttatásához), azonban a „találkozunk középen módszerrel, két összetartozó nyílt-rejtjeles blokkpár ismeretében a rendszer támadható:

A rejtjelzés képlete:

$$C = E_{K2}\big(E_{K1}(M)\big),$$

ezért

$$D_{K2}(C) = E_{K1}(M).$$

Ismert (M, C) párra kipróbálva – és letárolva – az összes $K1$ értékre az $E_{K1}(M)$ blokkokat, majd egyezést keresve az összes $K2$ kulcsra számolt $D_{K2}(C)$ blokkokkal,

leszűkíthető a $(K1, K2)$ esetszám: $2^{112}/\ 2^{64} = 2^{48}$-ra, amelyből a második összetartozó (M_2, C_2) blokkból nagy valószínűséggel kiválasztható az igazi $(K1, K2)$ kulcspár ($2^{48}/\ 2^{64} = 2^{-16}$ a tévedés valószínűsége).

Természetesen az eljárás során nagy számú technikai (pl. tárolási: 2^{56}* *8 byte, rendezési*) problémával kell megküzdeni, melyhez a Time-Memory Trade-Off módszerhez hasonló technika nyújt segítséget, de a kétszeres DES használat sem felel meg a kalkulációs biztonság követelményének.

8.3.2. Klasszikus stream-cipherek és a kalkulációs biztonság

Szintén nem nyújt megfelelő kalkulációs biztonságot (messze a teljes kipróbálás műveletigénye alatt marad) egy másik klasszikus – több millió készülékben kódolásra használt algoritmus, a GSM mobil telefonban kódoló A5 algoritmus sem.

A sokáig titokban tartott algoritmus internetre kerülése után indult vizsgálatok jelentős támadási eredményeket mutattak. A sok publikáció közül kiemeljük az alábbiakat:

Jovan Dj. Golic: **Cryptanalysis of Alleged A5 Stream Cipher**, EUROCRYPT '97; Alex Biryukov, Adi Shamir, and David Wagner: **Real Time Cryptanalysis of A5/1 on a PC**, mely a Fast Software Encryption, 7th International Workshop, FSE 2000. konferencián hangzott el; Alex Biryukov and Adi Shamir **Time/Memory/Data Tradeoffs for Stream Ciphers** címmel az ASIACRYPT 2000 konferencián, továbbá

Elad Barkan, Eli Biham, and Nathan Keller: **Instant Ciphertext – Only Cryptanalysis of GSM Encrypted Communication** címmel a CRYPTO, 2000 konferencián számoltak be további eredményekről.

Ez utóbbi cikkben foglalkoztak az A5/2 algoritmus hatékony törésével, valamint az A5/1 algoritmusnak pusztán rejtjelszövegből való támadásával, cikkükben az alábbi táblázat szerepel az eredményekről:

Támadáshoz szükséges adat	Felvételi percben	Szükséges 200 Gbyte-os tárolókapacitású HDD-k száma:	Szükséges PC-k száma egy éves előszámítási idővel számolva:	A támadás „reális" időben elvégzéséhez szükséges PC-k száma
2^{12}	5	22	140	1
$2^{6,7}$	8	176	5000	1000
$2^{6,7}$	8	350	5000	200
2^{14}	20	3	35	1

Néhány további példa a gyakorlatban használt, de a kalkulációs biztonsági követelményeket nem teljesítő stream-cipher-es rendszerre. (LFSR-ekből építkező rendszerek; a Bluetooth; több protokollban, pl WEP protokollban használt RC4; a PKZIP algoritmusa; stb.)

Sokszor alkalmazott rendszer a lineáris visszacsatolású shift regiszterekből modulszerűen építkező rejtjelrendszer. Ennek kritikus elemeire mutatnak példát az alábbiak:

- Az EUROCRYPT 2001 konferencián Geffe generátorára Siegenthaler támadási módszerének továbbfejlesztését mutatták be: Anne Canteaut and Eric Filiol „**Ciphertext Only Reconstruction of Stream Ciphers Based on Combination Generators**" c. cikkükben.
- Jovan Dj. Golic and Guglielmo Morgari **„On the Resynchronization Attack"** Fast Software Encryption 2003-as konferencián elhangzott előadásukban a lineáris következő állapotú-függvénnyel és nemlineáris kivételi függvénnyel bíró stream-cipherek egy támadása van leírva.
- AZ EUROCRYPT 2002-es konferencián **„Linear Cryptanalysis of *Bluetooth* Stream Cipher"** címmel Jovan Dj. Golić, Vittorio Bagini, and Guglielmo Morgari tartott előadást.
- **„Statistical Analysis of the Alleged RC4 Keystream Generator"** címmel Scott R. Fluhrer and David A. McGrew tartottak előadást a EUROCRYPT 2001 konferencián.
- **„ZIP Attacks with Reduced Known Plaintext"** címmel Michael Stay tartott előadást a Fast Software Encryption, 2001-es konferencián a PKZIP töréséről.

A fenti példák azt mutatták, hogy a kalkulációs biztonság nagyon sok, széles körben alkalmazott eljárásra nem biztosított. Érdemes azonban megjegyezni, hogy egy „átlagos" felhasználó kockázati szintjének lehet, hogy megfelel valamelyik standard eljárás, csak nagyobb kockázati szint esetén törhető nagyon jelentős erőforrással. Erre különösen jó példákat láthatunk az RSA biztonságával foglalkozó következő fejezetben.

8.3.3. Az RSA és a kalkulációs biztonság

Az RSA kalkulációs biztonságának elemzéséhez az alábbi mindkét tényezőt vizsgálni kell:

- Strukturális támadási vizsgálatok, ezen belül:
  - faktorizáció nélküli támadási irányok;
  - faktorizációs eljárások.
- Ezek műveletszámának (vagy a teljes kipróbálás műveletigényének) összevetése a számítástechnikai lehetőségekkel.

8.3.3.a. Az RSA elleni – nem faktorizácón alapuló – támadási módszerek

Az RSA kezdete óta nagyon sok publikáció jelent meg különböző, a modulus faktorizációját nem igénylő támadási módszerekről (ezekről áttekintés olvasható pl. [6], vagy D. Boneh, **Twenty years of attack on the RSA cryptosystem:** [1]. `http://crypto.stanford.edu/~dabo/abstract/ RSAattack-survey.html`). A támadások kiinduló-pontja hol rosszul megválasztott paraméterek, hol rossz rendszerhasználat; a támadások egyes esetekben az $\langle N, e\rangle$ nyilvános kulcsból a d magán

(titkos) kulcs visszaállítását, egyes esetekben a rejtjelzett üzenetek visszaállítását célozzák meg.

Néhány ismert, speciális esetekben hatékony támadási módszer látható a következő dupla oldalakon.

8.3.3.b. Az RSA faktorizációs támadása

A faktorizációs támadás fejlődése a számítástechnikai fejlődés (gyorsabb processzorok, ezek hatalmas rendszereinek egy feladatra koncentrálása) eredményeképpen, valamint (nagyobb részben) az elméleti módszerek fejlődésének eredményeképpen álltak elő.

Számítástechnikai segítség a faktorizációban

A nagy számok faktorizációja ősidők óta foglalkoztatja a matematikával foglalkozókat. Az algoritmikus módszerek és technikai lehetőségek egyaránt hatalmasat fejlődtek az utóbbi időben.

1874-ben úgy gondolták (Jevons), hogy senki sem tudja majd faktorizálni a 10 decimális jegyű ($10D$) 8616460799 számot. Persze akkoriban nem láthatták, a számítástechnika majd milyen segítséget ad a faktorizációhoz, de ez is mutatja az ilyen „jóslások" kockázatát.

A faktorizációs rekordok (D: a decimális jegyek számában)[14]

1964	1974	1984	1994
$20D$	$45D$	$71D$	$129D$

Kezdetben csak egy-egy gépen folyt a faktorizálás, 1984-ben a faktorizációs feladatra lehetett fordítani az akkori legerősebb Cray X-MP gép 9.5 CPU óra idejét.

A $129D$ faktorizálásában 1600 számítógép dolgozott 8 hónapig. Bár ez utóbbi nagyon sok gépnek számít, mégis pl. a Silicon Graphics kb. 5.000 munkatársa 10.000 workstation erőforrás felett rendelkezik. Mindez 10^5 MY (MY (MIPSYear): másodpercenként millió műveletet végző gép műveletszáma 1 év alatt), azaz 10-szer nagyobb erőforrás, mint az RSA129 projekthez szükséges volt.

A faktorizáció biztonságát befolyásoló, prognosztizált lehetőségeket lehet becsülni a sok átlagos gép erőforrásának együtteséből, illetve céleszközök kapacitásából.

A jelenlegi számítástechnikai lehetőségeket (maximális kooperativitást feltételezve) becsülhetjük az internetre kötött gépek számából (10^9 körül lehet a gépek száma), illetve a „Moore törvény"-ből, mely szerint a számítástechnikai teljesítmény (megfogalmazása szerint az integrált áramköri lapkákon elhelyezett tranzisztorok száma) 18 hónaponként megduplázódik, nagyjából lehet egy becslést adni az általános célú számítógépek teljesítménynövekedésére.

[14] A táblázat nem teljesen precíz, mivel – mint később látni fogjuk – a rekordok függnek a faktorizálandó számok tulajdonságaitól is(pl. 1994-ben faktorizálták a $162D$-ből álló $(12^{151} - 1)/11$ számot); ezért nem folytattuk a táblázatot 2004-re.

Támadás jellemzője	Támadás bemenő adatai	Feltétel	Támadás eredménye	Megjegyzés
Az üzenet rejtjelzés után is olvasható marad	Rejtjeles	M fixpontja az RSA rendszernek	M olvasható	Blakley-Borosh bizonyítása szerint az RSA rendszernek $(e,(p-1)) * (e,(q-1))$ fix pontja van, mely nagy szám is lehet, speciális esetekben (de elenyésző számú, ha $(p-1)$-nek és $(q-1)$-nek van nagy prímosztója. $(1+(e-1,$ $p-1)) * (1+(e-1,q-1))$
Ciklikus Attack	$\langle N,e\rangle$ nyilvános kulcs, C rejtjeles		Az M üzenet: $C^e, C^{e^2}, C^{e^3}, \ldots$ sorozatban megtalálva a $C = C^{e^k}$ egyezést, $M = C^{e^{k-1}}$	A Simmons és Morris által vázolt támadás nem működik, ha $(p-1)$-nek és $(q-1)$-nek van nagy prímfaktora.
Magán kulcs dekonspirálódása	$\langle N,e\rangle$ nyilvános kulcs és a d magán kulcs		N faktorizációja	
Közösen használt magán kulcsok	Több résztvevő által használt közös modulusok: $\langle N_1,e_1\rangle$, $\langle N_2,e_2\rangle$		Az előző állítás szerint meghatározhatják egymás magán kulcsait.	Támadó a rendszer egyik résztvevője
Kicsi nyilvános exponens használata	$\langle N_i,e\rangle$; $i=1,2,\ldots,k$; $C_i = M^e \bmod (N_i)$	Kicsi közös exponenseket használnak; $k \geq e$; $M < \min(N_i)$	Az M üzenet meghatározható	„Hastad's Broadcast Attack" a tétel további erősítését adja
Kicsi nyilvános exponens használata kapcsolódó üzenetekre	$\langle N,3\rangle$; $C_1 = M_1^3$ $C_2 = M_2^3$; f fv.	$e=3$, $f=ax+b$, $M_2 = f(M_1)$	M_1, M_2 meghatározható	Franklin-Reiter Related Message Attack

Kétszer ugyanazt az üzenetet rejtjelzik különböző blokkfeltöltéssel	$\langle N, e\rangle$; $C_1 = M_1^e$, $C_2 = M_2^e$	N : n bit hosszú, M : $n-m$ bit hosszú üzenet, ahol $m = \lfloor n/e^2 \rfloor$; $M_1 = 2^m M + r_1$; $M_2 = 2^m M + r_2$ $r_1 \neq r_2$ $0 \leq r_1$, $r_2 < 2^m$	M meghatározható	Copperrsmith' Short Pad Attack. A támadás hatékony $e = 3$ esetben, ha a közös üzenet hossza a modulus hosszának 9-ed részénél kisebb. Nem hatékony a gyakran használt $e = 65537$ esetben.
Kicsi nyilvános exponens használata	$\langle N, e\rangle$	$N = p * q$, $q < p < 2q$, $d < \frac{1}{3}\sqrt[4]{N}$	d hatékonyan meghatározható	A bizonyítást kiterjesztették $d < N^{0,292}$-re, továbbá sejtés: $d < \sqrt{N}$ esetén is d meghatározható
Részleges titkos kulcs ismeret	$\langle N, e\rangle$; d legkisebb $\lceil n/4 \rceil$ bitje, ahol N n bit hosszú modulus		D meghatározható $e * \log_2 e$-ben lineáris időben.	A tétel általánosítható p $\lceil n/4 \rceil$ számú legkisebb, vagy legnagyobb helyértékű bitjeire is
Kikényszerített elektronikus aláírás	$\langle N, e\rangle$ nyilvános ellenőrző kulcsok? A támadó által összeállított M' üzenet S' aláírása $(S' = (M')^d$	$M' \equiv r^e * M$ mod N, ahol r véletlen szám.	Támadó alá tudja írni az M üzenetét.	
Protokoll elleni támadás	PKCS-1 (Public Key Cryptography Standard-1) blokk, melyet a támadó módosít, és vizsgálja, a címzett elfogadja-e a blokkot.	Vonali módosítás lehetősége, a válasz figyelése	PKCS-1 blokkfelépítése: „02-RANDOM-00-M" Ha a címzettnél nem OK a „02" adat (16 bit), akkor ismétlést kér. Ha a támadó többször módosítja a kódolt üzenetet $C' = r * C$-re, látja, mikor fogadja el a címzett ugyanazt az üzenetet, melyből decryptálni tudja C-t!	Bleichenbacher'Attack on PKCS-1, 1998

Az egy feladatra koncentrálható erőforrásokat lehet az általános célú számítástechnikai eszközök rendszeréből (sok ilyen gép együttes alkalmazásából) becsülni (mint fent, ún. GRID-es technológiával), illetve célgépek felhasználási lehetőségeivel. Erre a jelenlegi lehetőségeket jól mutatják a www.top500.org lapon olvasható erős gépek (vektorszámítógépek) adatai, melyek közül (2005 júliusi adat): pl. a BlueGene/L-eSrever 2005-ben installált gépben 65536 processzor található (440700 MHz-es órajellel /2,8 GFlops/), melynek átlagos teljesítménye 136800 GFlops (136800 milliárd lebegőpontos művelet végrehajtására képes másodpercenként).

A faktorizációs módszerek fejlődése

A faktorizációs algoritmusok hatékonysága függ az n faktorizálandó szám szerkezetétől, valamint méretétől.

- Ha az n szám nem túl nagy, akkor az n szám négyzetgyökéig történő próbálgatás egy lehetséges módszer, melynek műveletigénye: $O(p + \lg n)$, ahol p a második legnagyobb prímosztót jelöli (alulról eddig kell elmenni a próbálgatással). Ismeretes, hogy az $[1, x]$ intervallumban véletlenül választott n számra kb. 0,9 (0,1) a valószínűsége annak, hogy n-nek p_2 második legnagyobb prímfaktorra $p_2 \leq x^{0,359} (p_2 \leq x^{0,0056})$ legyen, mely nagy számokra nagy valószínűséggel hatalmas szám, ráadásul az RSA számokra garantáltan nagy a második prím is.
- Ha N-nek csupa kis prímosztója van (ekkor az egyik leghatékonyabb módszer az ún. „$p - 1$ method").
- Az ún. SNFS (Special Number Field Sieve) eljárás jelenleg a leghatékonyabb az $a^k \pm b$ típusú (pl. Fermat) számokra (például ezzel a módszerrel faktorizálták F_9-et, a 9. Fermat számot, és 1999-ben a $10^{211} - 1$-et).
- Külön figyelmet érdemelnek az RSA számok (melyeknek nincs kis prímosztójuk, két nagy prím szorzatából állnak).
- $120D$ alatt a QS: kavadratikus szita (vagy MPQS: Multiple Polynomial Quadratic Sieve), míg felette a GNSF: General Number Field Sieve) eljárások az ismert leghatékonyabb módszerek.

Az RSA rendszer megalkotása után nem sokkal nyilvánosságra hoztak ún. RSA számokat, melyek két nagy, titokban tartott prímszám szorzatai voltak (ezek faktorizációjára pénzdíjakat is kitűztek, de nagyobb érték volt a hírnév). Mindezt azért tették, hogy lehessen követni az új faktorizációs eljárásokból (illetve a technikai fejlődésből) adódóan az RSA paraméterekkel szemben milyen nagyságrendi követelményeket kell támasztani. Az alábbi táblázat ennek a faktorizációs versenynek néhány állomását mutatja (jelölve a faktorizációs algoritmust, és a szükséges számítási kapacitást MIPS-évben).

Az alábbi táblázatban MY: MIPS-Year-t (Millio Interception Per Year), QS: a Quadratikus szita faktorizációs módszerét, NFS: a Number Field Sieve módszerét jelöli.

Az RSA 155 (1999-ben történt) faktorizációját a EUROCRYPT 2000 konferencián mutatták be „Factorization of a 512-Bit RSA Modulus" címmel, szerzők:

RSA szám:	Decimálisan	Faktorizálás		Műveletigénye	Kitűzött díj
		Dátuma	Módszere		
RSA-100	100	1991.04.	MPQS	7 MY	
RSA-110	110	1992.04.	MPQS	75 MY	
RSA-120	120	1993.06.	MPQS	835 MY	
RSA-129	129	1994.04.	MPQS	5000 MY	
RSA-130	130	1996.04.	NFS	1000 MY	
RSA-140	140	1999.02.	NFS	2000 MY	
RSA-150	150	2004.04.	NFS		
RSA-155	155 (**512** bit)	1999.08.22.	NFS	8400 MY	
RSA-160	160	2003.04.01.	NFS		
RSA-576	174	2003.12.03.	NFS		10.000 $
RSA-200	200	2005.05.09.	NFS		
RSA-640	193				20.000 $
RSA-704	212				30.000 $
RSA-**768**	232				50.000 $
RSA-896	270				75.000 $
RSA-**1024**	309				100.000 $
RSA-1536	463				150.000 $
RSA-**2048**	617				200.000 $

Stefania Cavallar, Bruce Dodson, Arjen K. Lenstra, Walter Lioen, Peter L. Montgomery, Brian Murphy, Herman te Riele, Karen Aardal, Jeff Gilchrist, Gérard Guillerm, Paul Leyland, Joël Marchand, François Morain, Alec Muffett, Chris and Craig Putnam és Paul Zimmermann.

Ez a hosszú sorban azért bizonyult nagy áttörésnek, mivel az 512 bites modulust nagyon sok implementációban használt(j)ák. Szemléltetésül az eredmény:

RSA-155 =
109417386415705274218097073220403576120037329454492059909 13842131476349
984288934784717997257891267332497625752899781833797 0765372440271 467435
31593354333897
Szám faktorizációja, mely az alábbi két prímszám szorzata:
p = 102639592829741105772054196573991675900716567808038066803341933521790711307779
q = 106603488380168454820927220360012878679207958575989291522270608237193062808643

2005 közepén a csúcs a 2005. május 9-én bejelentett, RSA200 faktorizálása GNFS módszerrel, melynél a megfelelő egyenletek keresését (szita lépésről ld. később) 2003 karácsonya előtt kezdték és 2004 októberéig tartott (melyek összességében 55 év CPUidőt vettek volna igénybe egy átlagos 2,2 GHz-es gépen), míg a hatalmas mátrix megoldása 2004 decemberében vette kezdetét. Az eredmény két szám:

353246193440277012127260497819846436867119740019762502364930346877612125367942320005854795652808834 9
és
792586995447833303334708584148005968773797585736421996073433034145576787281815213538140930474018546 7

Faktorizáció Fermat ötletével

Az RSA modulus (két nagy prím szorzata) hatékony faktorizációja meglepően Fermat alapötletén alapul:

Amennyiben találunk olyan a, b számokat, melyekre $a^2 \equiv b^2 \bmod n$, és $a \neq +- b \bmod n$, akkor n osztható $a^2 - b^2 = (a-b) * (a+b)$ kifejezéssel, azaz $gcd(a-b, n)$ n-nek egy osztója.

Meglepő, hogy miért könnyebb találni adott tulajdonságú a, b számokat, melyek négyezetei mod$n$ megegyeznek, mint n-et faktorizálni.

Ehhez dolgozták ki a nagyon hatékonynak bizonyult ún. **kvadratikus szita** módszerét, melynek lényege:

Keresünk sok olyan $r^2 (> n)$ számot, melynek mod$n$ vett értékét tudjuk faktorizálni, és ezek prímfaktorai egy korlátos halmazból valók (viszonylag kicsik).

Sok ilyen

$$(*) \qquad r^2 \equiv y = \prod p_j^{\alpha_j} \bmod n$$

kongruenciából kiválasztjuk – lineáris egyenletrendszert megoldva – azokat, amelyeket összeszorozva a jobb oldalon minden (a korlát alatt lévő prím) páros hatványon szerepel ez adja b^2-et, míg a baloldalak (eleve négyzetszámok) szorzata adja a^2-et.

A fent keresett y számokra használják az ún. ***B-smooth*** fogalmát (B pozitív egész szám), mely szerint egy y szám B-smooth, ha minden prímtényezője kisebb B-nél.

A fenti szellemes eljárás gyengéje, hogy nagyon nagy n-ekre nagyon kicsi a valószínűsége, hogy y-nak csak B-nél kisebb prímosztói legyenek.

D. J. Bernstein: Bounding SmoothIntegers c. Extended Abstract-jában vizsgálta ezen esélyeket.

$\Psi(x, B) = \{\#m : 1 \leq m \leq x\}$ és m B-smooth jelölés mellett leírta $\Psi(x, B)$ nagyságrendjét. Például azt kapta, hogy $1{,}16 \cdot 10^{45} < \Psi(10^{54}, 10^6) < 1{,}19 \cdot 10^{45}$, azaz már $y = 10^{54}$ esetén is $\Psi(x, B)/y \sim 10^{-9}$ nagyon kis szám, sokkal nagyobb y-ra még sokkal kisebb az esély (hogy B-smooth legyen).

A módszert úgy kellett fejleszteni, hogy a (*) kongruencia jobb oldalán (y) nagyobb valószínűséggel legyen B-smooth, melyet úgy értek el, hogy megpróbálták y-t minél kisebb számra kihozni úgy, hogy nem tetszőleges r^2 alakú számként próbálgatunk, hanem olyat, ahol y „kicsi" lesz.

A fent leírt ötlet: *A kvadratikus szita (QS) módszere*

n gyökének alsó egész részéből $\left(m = \lfloor \sqrt{n} \rfloor\right)$ kiindulva számolja a $q(x) = (x+m)^2 - n = x^2 + 2mx + m^2 - n \approx x^2 + 2mx$ számot (mely kicsi n-hez ké-

pest, ha x kis szám); $x = \pm 1, \pm 2, \ldots, \pm S$, ahol S a „Sieve" intervallum paramétere. Mivel $q(x)$ kicsi n-hez képest (kis S-re), ezért nagyobb valószínűséggel ennek csak kis prímfaktorai vannak.

Választva egy B számot, tesztelik, hogy a $\left[q(x) \bmod n\right]$ számok B-smooth-ok-e, azaz ($\left[(x+m)^2 \bmod n\right]$ prím faktorai kisebbek B-nél) – különböző x értékekre (s-ig).

Keresve sok x_i értékhez B-smooth $\left[q(x_i) \bmod n\right]$ értékeket, ezekre

$$(x_i + m)^2 \equiv q(x_i) \mod n \quad (K1)$$

Mivel a $\left[q(x_i) \bmod n\right]$ számok B-smooth-ok, azaz csak „kis" prímfaktorai vannak, ezért olyan i indexű kongruenciákat vesznek, melyek szorzatában minden „kis" prímfaktor páros hatványon szerepel a fenti $K1$ kongruenciák jobb oldalainak összeszorzása után.

A $K1$ kongruenciák szorzatának bal oldalain eleve négyzetszámok szerepelnek, bal oldalon a fenti kiválasztás alapján kapunk négyzetszámokat, így kapják meg a kívánt tulajdonságokkal rendelkező a és b számokat, melyekből n faktorizálható.

(Gyorsítja az eljárást, hogy mivel egy $p(< B)$ prím ha osztja $q(x)$-et, akkor $(x+m)^2 \equiv n \bmod p$, ezért n kvadratikus maradék moduló p. Így csak azokkal a faktor-bázis elemekkel kell foglalkozni, amelyekre a Legrende szimbólum értéke 1.)

Példaként tekintsük az $n = 11111$ számot, legyen $S = 10$, $B = 15$, ezért $m = 105$, a prímek bázishalmaza $= \{-1, 2, 3, 5, 7, 11, 13\}$ (a -1 kiegészítés azért kell, mert $q(x_i)$ lehet negatív.

Ekkor a szita fázisban a következő megfelelő értékeket kapjuk:

$$\begin{aligned}
Q(-4) &= 101^2 - 11111 = -910 = -1 \cdot 2 \cdot 5 \cdot 7 \cdot 13 \\
Q(\ 1) &= 106^2 - 11111 = 125 = 5^3 \\
Q(\ 2) &= 107^2 - 11111 = 338 = 2 \cdot 13^2 \\
Q(\ 4) &= 109^2 - 11111 = 770 = 2 \cdot 5 \cdot 7 \cdot 11 \\
Q(\ 6) &= 111^2 - 11111 = 1210 = 2 \cdot 5 \cdot 11^2
\end{aligned}$$

Meg kell oldani az alábbi kongruenciát:
$= (-1 \cdot 2 \cdot 5 \cdot 7 \cdot 13)^x \cdot (5^3)^y \cdot (2 \cdot 13^2)^u \cdot (2 \cdot 5 \cdot 7 \cdot 11)^v \cdot (2 \cdot 5 \cdot 11^2)^w$
úgy, hogy minden prím páros hatványon szerepeljen:

$$\begin{aligned}
-1 &: \quad x \equiv 0 \pmod 2 \\
2 &: \quad x + u + v + w \equiv 0 \pmod 2 \\
5 &: x + 3 \cdot y + v + w \equiv 0 \pmod 2 \\
7 &: \quad x + v \equiv 0 \pmod 2 \\
11 &: \quad v + 2 \cdot w \equiv 0 \pmod 2 \\
13 &: \quad x + 2 \cdot u \equiv 0 \pmod 2
\end{aligned}$$

A fenti kongruencia-rendszer egy megoldása:

$x = 0$, $y = 1$, $u = 1$, $v = 0$, $w = 1$; ezért

$(106 \cdot 107 \cdot 111)^2 = (2 \cdot 5^2 \cdot 11 \cdot 13)^2$

ezért $a = 106 \cdot 107 \cdot 111$ és $b = 2 \cdot 5^2 \cdot 11 \cdot 13$ esetén

$a^2 \equiv b^2 \bmod n$, így $\text{lnko}\,(a - b, n) = 41$ n osztója.

A fenti kis mintapélda csak az elvet mutatja, lényegesen nagyobb számokra érzékeltetésül megadjuk az eredményes futtatáshoz várható szükséges paraméterek (S, B) méretét:

N jegyei decimálisan	Faktor bázis	Szita (Sieving) intervallum (S)
50	3.000	200.000
60	4.000	2.000.000
70	7.000	5.000.000
80	15.000	6.000.000
90	30.000	8.000.000
100	51.000	14.000.000
110	120.000	16.000.000
120	245.000	26.000.000

(A fenti táblázat Johannes A. Buchmann: Introduction to Cryptography, Springer, 1999 könyvében szerepel.)

A módszer használatában (a B és S paraméterek meghatározása után) két jelentős műveletigényű fázis van:

- találni $(x_i + m)^2 \equiv q(x_i) \bmod n = \prod p_i^{\alpha i} (p_{i<}B)$ kongruenciákat „Sieve Step";
- megoldani egy GF(2) feletti egyenletrendszert, annak meghatározására, hogy mely kongruenciákat kell összeszorozni ahhoz, hogy a baloldalon minden prím páros hatványon szerepeljen „Matrix Reduction Step".

Míg az első feladat jól párhuzamosítható (sok gépre szétoszthető a keresés), a második egy nagy teljesítményű gépet igényel.

Mind a Kvadratikus szita (**QS**), mind továbbfejlesztése, a 120 decimális számnál nagyobb számokra hatékonyabb Számelméleti szita (Number Field Sieve: **NFS**, Generalized Number Field Sieve: **GNFS**) módszerek a Fermat ötletnek megfelelő (mod n négyzetükben kongruens) a és b számokat keresnek. Mindegyiknél van sok gépre szétoszthető (párhuzamosítható) feladatrész: speciális kongruenciák keresése) és hatalmas mátrixot megoldó rész.

A módszerről *olvasható* például az alábbi hivatkozásokban:

A. K. Lenstra and H. W. Lenstra, Jr., *The Development of the Number Field Sieve.* Berlin: Springer-Verlag, 1993.

C. Pomerance, „A Tale of Two Sieves." *Not. Amer. Math. Soc.* **43**, 1473–1485, 1996.

Eric W. Weisstein, „Number Field Sieve." From MathWorld-A Wolfram Web Resource. http://mathworld.wolfram.com/NumberFieldSieve.html

A. K. Lenstra, Bellcore, H. W. Lenstra, Jr., M. S. Manasse, J. M. Pollard, The number field sieve: http://www.std.org/~msm/common/nfspaper.pdf

D. Coppersmith, „Modifications to the Number Field Sieve." *J. Cryptology* **6**, 169–180, 1993.

J. Cowie, B. Dodson, R. M. Elkenbracht-Huizing, A. K. Lenstra, P. L. Montgomery and J. A. Zayer, „World Wide Number Field Sieve Factoring Record: On to 512 Bits." In *Advances in Cryptology-ASIACRYPT '96 (Kyongju)* (Ed. K. Kim and T. Matsumoto.) New York: Springer-Verlag, pp. 382–394, 1996.

Mint említettük az N összetett szám faktorizálására vannak speciális tulajdonságokkal rendelkező számokra hatékony és ilyen tulajdonságra nem kihegyezett módszerek.

Például ilyen, speciális tulajdonságú számokra hatékony módszerek:

- az osztásos próbálkozás;
- Pollard „rho" algoritmusa;
- Pollard „p-1" algoritmusa;
- ECM: Lenstra elliptikus görbéket felhasználó algoritmusa;
- Fermat faktorizációs módszere;
- a „Special Number Field Sieve" módszer.

A módszerek többsége akkor hatékony, ha N-nek sok kis prímosztója van, az utolsó akkor, ha $N = a^k \pm b$ alakú, ahol a, b kis számok (k lehet nagy).

Általános célú faktorizációs módszerek:

- Dixion algoritmusa;
- CFRAC: Continued Fraction Factorization;
- MPQS: Multiple Polynomial Quadratic Sieve;
- GNFS: General Numver Field Sieve.

Az N összetett szám faktorizációhoz szükséges műveletigény egyszerűbb leírásához használják a következő formulát (u, v pozitív egész számok, $N \geq e$):

$$L_N[u, v] = e^{v*(\ln N)^u*(\ln\ln N^{(1-u)})}.$$

A fenti képlet jól jellemzi a futási időt, mivel

- $L_N[0, v] = e^{v*(\ln N)^0*(\ln\ln N^1)} = (\ln N)^v$, amely polinomiális futási időt eredményez, mivel N bináris hossza $\lfloor \log_2 N \rfloor + 1$;
- $L_N[1, v] = e^{v*(\ln N)^1*(\ln\ln N^0)} = e^{v*\ln N}$, amely exponenciális futási időt mutat;
- $0 < u < 1$ esetén a futási idő ún. szubexponenciális.

A fenti jelölésekkel a szakirodalomban található levezetések alapján:
A CFRAC algoritmus műveletigénye: $L_N\left[1/2, c + O(1)\right]$, ahol $c = \sqrt{2} \approx 1{,}4142$.
A MPQS algoritmus műveletigénye: $L_N\left[1/2, 1 + O(1)\right]$.
A CFRAC algoritmus műveletigénye: $L_N\left[1/3, c + O(1)\right]$, ahol

$$\begin{array}{ll} \text{GNFS esetén} & c \approx 1{,}9229, \\ \text{SNFS esetén} & c \approx 1{,}5262. \end{array}$$

Képlettel felírva a szükséges műveletszámokat az alábbi táblázat ad egy összefoglalást, ahol N a faktorizálandó szám, melynek egy p ($\leq \sqrt{N}$) prímfaktorát kell megtalálni. A kétféle módszer, az elsőben a módszer műveletszáma függ p-től, a másodikban nem (pl. az RSA számokra a második esetet kell használni):

Módszer elnevezése	**Függ-e a meghatározandó prímfaktortól**	**Számításigény**
Prímosztásos próbálkozás	Függ	$O\left(p * (\ln N)^2\right)$
Pollard „rho" algoritmusa	Függ	$O\left(p^{1/2} * (\ln N)^2\right)$
ECM: Lenstra Elliptikus görbéken alapuló algoritmusa	Függ	$O\left(e^{\sqrt{c * \ln p * \ln \ln p}} (\ln N)^2\right)$, ahol $c \approx 2$
Lehman algoritmusa (legrosszabb esetre)	Nem függ	$O(N^{1/3})$
MPQS (Multiple Polynomial Quadratic Sieve)	Nem függ	$O\left(e^{\sqrt{c * \ln N * \ln \ln N}}\right)$, ahol $c \approx 1$
NFS (Number Field Sieve) SNFS: Special NFS: $N = a^k \pm b$, a, b kicsi GNFS: General Number Field Sieve	Nem függ	$O\left(e^c * (\ln N)^{1/3} * (\ln \ln N)^{2/3}\right)$, ahol $C = (32/9)^{1/3} \approx 1{,}5262$ $C = (64/9)^{1/3} \approx 1{,}93$

Megjegyzés: Ha $p \approx \sqrt{N}$, akkor az ECM és az MPQS műveletigénye aszimptotikusan közel megegyezik.

A fenti táblázatot Francois Morain: „Thirty Years of Integer Factorization", 2001. febr. 5. cikkéből, valamint Richard P. Brent „Some parallel Algorithms for Integer Factorization", LNCS Vol. 1685 (1999), 1–22 c. cikkéből állítottuk össze.

Az NFS módszer eredményességéhez szükséges faktor bázisok (prekondicionált prímek), a szita módszerhez felhasznált előkészítő kongruenciák, illetve a lineáris kongruencia-rendszer mátrixának méretére Shamir a következő becslést adta különböző méretű RSA számok esetére:

Key-size	Total Time	Factor Base	Sieve Memory	Matrix Memory
428	$5.5 * 10^{17}$	600 K	24 Mbytes	128 Mbytes
465	$2.5 * 10^{18}$	1.2 M	64 Mbytes	825 Mbytes
512	$1.7 * 10^{19} (\sim 2^{64})$	3 M	128 Mbytes	2 Gbytes
768	$1.1 * 10^{23}$	240 M	10 Gbytes	160 Gbytes
1024	$1.3 * 10^{26}$	7.5 G	256 Gbytes	10 Tbytes

Az RSA140-re és RSA155-re faktorizálásánál a mátrix megoldásához szükséges kapacitásadatokat mutatja az alábbi táblázat:

	MIPS igény	Sorok száma a mátrixban	Nem nulla elemek számának átlaga egy sorban	A mátrix megoldási ideje Cray C916-tal
RSA140	2000	$4{,}7 * 10^6$	32	100 óra
RSA155	8000	$6{,}7 * 10^6$	62	224 óra

Az utóbbi időben a kutatások fő iránya az 1024 bites RSA feltörésére alkalmas módszerek és célhardverek előállítása irányában felerősödtek.

Az ilyen kutatások sikerének kockázata, hogy a legtöbb RSA-t használó kriptográfiai rendszerben (pl. **HTTPS, SSH, IPSec, S/MIME, PGP**), de a legtöbb elektronikus aláírási rendszerek ma 1024 bites RSA-t használnak. Vagyis az esetleges sikernek közvetlen, a gyakorlatban használt módszerek biztonságát kompromittáló következménye lenne.

Az ASIACRYPT 2003 konferencián Arjen Lenstra, Eran Tromer, Adi Shamir, Wil Kortsmit, Bruce Dodson, James Hughes, and Paul Leyland adtak elő „Factoring Estimates for a 1024-Bit RSA Modulus" címmel.

A CRYPTO 2003 konferencián Adi Shamir and Eran Tromer adtak elő „Factoring Large Numbers with the TWIRL Device" címmel.

A módszer alapja az NFS (Number Sieve Field) eljárás, 1999-ben a szita része többszáz gépen több hónapig futott. Akkor úgy gondolták, hogy az 1024 bit még 15–20 évre biztos elegendő nagyságú lesz. A cikk bemutat egy 1 GHZ-es VLSI technológián alapuló hardware implementációt, mely 3–4 nagyságrenddel hatékonyabb az összes korábbinál (elnevezése TWINKLE), mely az 512 bites modulus szita lépését 10 perc alatt elvégzi, és az eszköz megvalósítható 10.000 USD-ből. Ugyanez 1024 bitre – egy év alatti szita-lépésre – 10 millió USD-ből jönne ki.

Nyilvánvaló, hogy a jelenlegi matematikai módszerekkel általános célú számítógépekkel az 1024 bites RSA nem törhető. Ezért a kutatások nagyon sok processzor, speciális vezérléssel szervezett célhardverek irányában is intenzíven folynak. Itt megemlítjük a SHARCS (Special PurposeHardware for Attacking Cryptographic Systems) 2005, Párizs konferencián elhangzott két előadást:

- Shamir és E. Tromer: Special-Purpose Hardware for Factoring: the step NFS Sieving Step, melyben a TWINKLE és a TWIRL célhardverek tervezéséről volt szó;
- J. Franke, T. Kleinjung, C. Paar, J. Pelzl, C. Priplata, C. Stahlke: Shark – A Realizable Special Hardware Sieving Device for Factoring 1024-bit integers.

Összefoglalva az RSA kalkulációs biztonságáról szóló részfejezetet elmondhatjuk, hogy rendszerszintű hibát nem elkövetve, az 1000 bit feletti modulusú RSA jelenleg általános célú felhasználásra biztonságosnak tekinthető.

A korábbi, prímekre vonatkozó kikötések (miszerint „$p-1$"-nek, „$q-1$"-nek is legyen nagy prímosztója, ma már nem követelmény. Ennek oka, hogy a korábbi elvárásokhoz képest jelentősen megnövelt modulus miatt, a kapcsolódó támadási módszerek csak nagyon kis valószínűséggel hajthatók végre. Ezt támasztják alá az első és második prímosztóra vonatkozó alábbi eredmények:

Az $[1,x]$ intervallumban véletlenszerűen választott N szám legnagyobb prímosztója: $p_1 \leq x^\alpha$ egy $F(\alpha)$ valószínűség-eloszláshoz tart, ahol

$$F(\alpha) = \int_0^\alpha F\left(\frac{t}{1-t}\right)\frac{dt}{t}, \quad \text{ha} \quad 0 \leq \alpha \leq 1; \quad \text{és} \quad F(\alpha) = 1, \quad \text{ha} \quad \alpha \geq 1.$$

Hasonló integrállal fejezhető ki a második legnagyobb prímosztó $G(\beta)$ eloszlása is (mely pl. a próbálgatásos faktorizáció műveletigényének becslésekor kerül elő). Példaként megadjuk ennek a két sűrűségbecslésnek néhány értékét, mely mutatja, hogy egy x-nél nem nagyobb véletlen szám első, ill. második legnagyobb prímosztója $F(\alpha)$, ill. $G(\beta)$ valószínűséggel lesz x^α, x^B-nél kisebb, ahol:

$F(\alpha), G(\beta)$:	0,9	0,8	0,5	0,35	0,2	0,1	0,01
1. legn. α:	0,9048	0,8187	0,6065	0,5220	0,4430	0,3785	0,2697
2. legn. β:	0,3590	0,3104	0,2117	0,1611	0,1003	0,0531	0,0056

azaz például ha $x = 2^{1024}$, akkor a legnagyobb, második legnagyobb prímosztó 50% valószínűséggel kisebb, mint $x^{0,6065} \approx 2^{621}$, ill. $x^{0,2117} \approx 2^{216}$.

Az EESSI-SG (European Electronic Signature Standardisation Initiative Steering Group) felügyelete alatt dolgozó Algoritmus csoport (ALGO) által meghatározott követelmények RSA algoritmus esetén:

- a modulus $n = pq$ bithossza (ModLen) legalább 1020 bit legyen,
- p és q megközelítőleg azonos hosszú: $0{,}5 < |\log_2 p - \log_2 q| < 30$,
- megfelelően nagyszámú prímszám közül függetlenül kell a két prímet választani, és ezek eloszlása megfelelően egyenletes kell legyen: a véletlen választás, ill. egy véletlen generátor induló értékének legalább 128 bit szabadsági fokúnak kell lennie,
- a prímtesztnél a hiba valószínűsége (tehát annak valószínűsége, hogy p, vagy q mégis összetett szám) kisebb mint 2^{-60} legyen.

8.3.4. Az elektronikus aláíráshoz szabványosított hash eljárások és a kalkulációs biztonság

A gyakorlatban alkalmazott nem megfelelő biztonsági szintű primitívek nem csak a titkos kulcsú rejtjelző primitívek között ismertek, hanem például a hash számoló primitívek között is:

- A Fast Software Encryption, 2003 konferencián „**Cryptanalysis of Block Ciphers Based on SHA-1 and MD5**" címmel Markku-Juhani O. Saarinen tartott előadást, melyben bevezetett „related key attack" módszerével ún. „slid-pairs"-t kerestek (találtak).
- Ugyanezen a konferencián „**New Attacks against Standardized MACs**" címmel Antoine Joux, Guillaume Poupard, and Jacques Stern előadásukban legfeljebb 2^{33} összetartozó „choosen messages"-re támadják a ISO/IEC 9797-1 szabvány szerinti DES-t használó MAC eljárást.
- A CRYPTO 2000 konferencián „**Key Recovery and Forgery Attacks on the MacDES MAC Algorithm**" c. előadásukban Don Coppersmith, Lars R. Knudsen, and Chris J. Mitchell foglakoztak a DES alapú MAC számítás problematikájával.

Az elektronikus aláírási rendszerekben – számítási kapacitás korlátok, valamint a dokumentumokhoz fűzött kiegészítő információk csökkentése okán – nem a teljes dokumentumot írják alá (aláírási primitívvel), hanem csak egy lenyomatát, amit a hash függvénnyel állítanak elő. (A bináris hash függvények: $M \to H(M)\{0,1\}^n \to \{0,1\}^m$ leképezést valósítanak meg egy tetszőleges n ($\geq n_0$) hosszúságú bináris sorozatot egy rögzített m ($\geq m_0$) hosszúságú sorozatra képezve le.) Ekkor – több más követelmény mellett – elengedhetetlen, hogy adott lenyomathoz ne lehessen másik olyan dokumentumot találni, amely azonos lenyomatot (így azonos aláírást) képez. Ilyen tulajdonságokkal már a hagyományos hibajavító kódolók nem rendelkeznek (mint alább látni fogjuk, nem is egyszerű megfelelő Hash függvényt konstruálni).

A szakirodalomban a követelmények többféle megfogalmazása szerepel (és néha nehéz jó magyar kifejezést találni ezekre), melyek három *tulajdonságra* vezethetők vissza:

- *őskép-ellenállóság* (preimage resistance): adott y értékhez „nehéz" találni olyan M üzenetet, amelyre $H(M) = y$;
- *második őskép-ellenállóság* (2-nd preimage resistance): adott M üzenethez (melyre „könnyű" kiszámolni $y = H(M)$-et), nehéz találni olyan M'-t, melyre $H(M) = H(M')$;
- *ütközés-ellenállóság* (collision resistance): ne lehessen két különböző M, M' üzenetet találni, melyekre $H(M) = H(M')$.

(Az egyes tulajdonságok közötti összefüggéseket részletesen vizsgálták. Például az ütközés-ellenállóság nem garantálja az őskép-ellenállóságot.)

A Hash függvényekre használt *követelményeket* a fenti tulajdonságok alapján két nagyobb csoportra szokták osztani:

- **OWHF** /One Way Hash Function/, mely az őskép-ellenállóság és a második őskép-ellenállóság tulajdonságának teljesítését jelenti; valamint
- **CRHF** /Collision Resistant Hash Function/ az ütközésmentesség követelményei, mely a második őskép-elleneállóság és az ütközés-ellenállóság tulajdonságának teljesítését jelenti.

Megjegyzések:

Egyes definíciók az OWHF-re a „weak one-way hash function" kifejezést, míg a CRHF-re a „weak one-way hash function" kifejezést használják.

A „könnyű" kiszámítani fogalom alatt a legtöbb szakirodalom a bemenő paraméterek (M mérete) függvényében polinomiális időben kiszámíthatóságot érti.

Egyes speciális feladatokhoz szokás még más tulajdonságok teljesülését is előírni, ilyenek pl:

- *Közeli ütközés-ellenállóság* (near-collision resistance), amikor nehéz két különböző M, M' üzenetet találni, melyekre $H(M)$, $H(M')$ csak kevés számú bitben különbözik.
- *Részleges őskép-ellenállóság* (partial-preimage resistance), amikor az input egy része és $y = H(M)$ ismert, akkor is „nehéz" találni olyan M üzenetet, amelynek része az ismert input-töredék és $H(M) = y$; (pontosabban, ha az inputból t bit ismert, átlagosan 2^{t-1} hash-művelet szükséges az M input megtalálásához).
- *Korreláció-mentesség (non correlation):* rövid üzeneteknél feltétel (amikor az egyirányú függvényre az input és output nagyságrendje közeli (ilyen felhasználás például a jelszavak egyirányú képének letárolásakor, vagy blokkos rejtjelzéseknek, mint egyirányú függvényeknek a vizsgálatakor jelentkezik).
- *Univerzális egyirányú függvény (UOWHF):* néhány alkalmazásban – az egyszerűbb konstrukciók miatt – a CRHF alternatívájaként jelenik meg az UOWHF, melynél a lenyomatképző függvények egy indexelt halmaza áll rendelkezésre (azonos ősképtérrel és képtérrel), és az üzenetválasztás után választanak a lenyomatképző halmazból egyirányú függvényt.
- *Szabad kezdőérték melletti őskép-ellenállóság (pseudo-primage* resistance), amikor az egymást követő input blokkokon operáló iteratív lenyomatképző algoritmus (ahol a következő blokk képzésébe az előző blokk eredménye behat), véletlen input blokkal kezdődik (ez hat be az első blokk lenyomatképzésébe). Nyilván valamivel könnyebb feladat, ha a támadó is szabadon választhatja meg az első input blokkot, ezért az ilyen támadás kizárását is elő kell írni.

Léteznek ajánlások a képtér méretére [ld. Menezes–Oorschot–Vanstone: Handbook of Applied Cryptography c. könyvének 9.3. fejezet alapján]:

$$\begin{array}{ll} \text{OWHF-re} & n \geq 80, \\ \text{CRHF-re} & n \geq 160. \end{array}$$

Hash függvényekre nagyon sok algoritmust javasoltak, ilyenek például: az MD család (Message Digest rövidítéséből: MD2, MD4, MD5), az SHA család (SHA-0, SHA-1, SHA-256, HSA-384, SHA-512), RIPMD-160, és egyebek (HAVAL, N-HAsh, Snefru, Tiger, Whirpool).

A szakirodalom sokat foglalkozik ezek biztonságával, ld. például az alábbi cikkeket:

B. den Boer, Antoon Bosselaers, Collisions for the Compression Function of MD5, Eurocrypt, 93.

H. Dobbertin, Cryptanalysis of MD4, Fast Software Encryption, LNCS 1039, D., Springer-Verlag, 1996.

H. Dobbertin, Cryptanalysis of MD5 compress, presented at the rump session of EurocrZpt'96.

Hans Dobbertin, RIPEMD with Two-round Compress Function is Not Collision-Free, *J. Cryptology* **10(1)**, 1997.

H. Dobbertin, A. Bosselaers, B. Preneel, „RIPMEMD-160: A Strengthened Version of RIPMMD," Fast

P. R. Kasselman, W. T. Penzhorn, Cryptananlysis od reduced version of HAVAL, Vol. 36, No. 1, Electronic Letters, 2000.

Xiaoyoun Wang, Dengguo Feng, Xuejia Lai, Hongbo Yu „Collisions for Hash Functions, MD4, MD-5, HAVAL-128 and RIPMD" című cikkükben a címben meghatározott hash függvények mindegyikére párokat adtak meg, melyeknek ugyanaz a hash képe, azaz nem teljesül az ütközés-állóság fontos követelménye.

Dan Kaminsky elkészített egy olyan perl scriptet („proof-of-concept" kódot), mellyel az MD5 algoritmust be lehet csapni. Meg is adott 2 file-t (fire.bin és ice.bin néven), melyek MD5 kódja megegyezik, de SHA1 kódja különbözik (ezzel is bizonyítva a tartalmak különbözőségét és az SHA1 „megfelelőségét". Bizonyítása:

fire.bin: md5 = 2fda7b311ce4d4f05256284e41843d87,
ice.bin: md5 = 2fda7b311ce4d4f05256284e41843d87,
fire.bin: sha1 = 5ee5f5edc5744a4803baaf0b4c869d65d6001888,
ice.bin: sha1 = 072c421c6e43db03204665c59ade2eb45827ae6c.

Ekkor még úgy gondolták, hogy az SHA1 megfelelő (ez szerepel az ETSI ALGO munkacsoportja ajánlásai között is).

Nemrég azonban a Shandong egyetem munkatársai (Xiaoyun Wang, Yiqun Lisa Yin, és Hongbo Yu) bejelentették az SHA1 feltörését.

(A kínaiak az összes lehetséges kombináció végigpróbálásához szükséges 2^{80} támadási próbálkozás helyett 2005 elején már 2^{69} művelettel sikerrel jártak, 2005 nyarán a CRYPTO' 2005 konferencián tett bejelentés szerint ezt csökkentették 2^{63}-ra, ami jelentős a 2^{64} határ áttörése miatt.)

8.3.5. Összegzés a kalkulációs biztonsághoz:

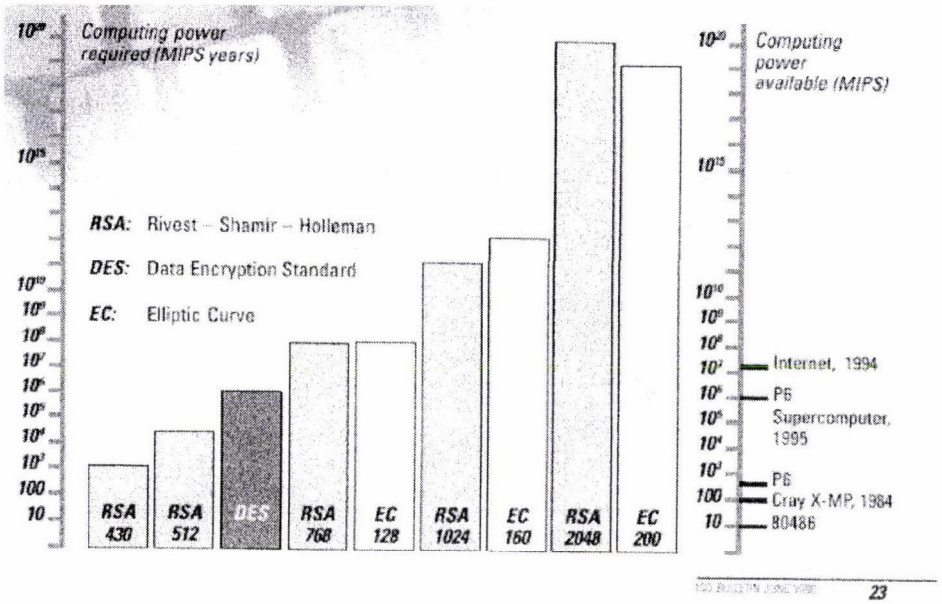
A kalkulációs biztonság elemzéséhez három tényező összevetése szükséges:

- a leghatékonyabb törési algoritmusok keresése;

- az algoritmus futtatásához használt (cél)gépek kapacitása (sebesség, tárhely);
- a feladathoz igénybe vehető gépek száma.

A kalkulációs biztonság megközelítéseihez a töréshez szükséges műveletszámok becslése szükséges.

Az alábbi táblázat az ISO Bulletin 2000. évi számából mutatja az egyes kriptográfiai primitívek (adott paraméterek melletti) támadási időszükségletét:



A táblázat bal oldala logaritmikus skálán mutatja a MIPS year szükségletet, jobb oldala az adott (jelzett) időben rendelkezésre álló kapacitásokat, például az összes internetre kötött számítógép számát jelzi.

Burt Kaliski, RSA Laboratories, IPA/TAO Cryptography Symposium, October 20, 2000, „Cryptography Trends: A US-Based Perspective" c előadásában vázolta az ekvivalens kulcsmérteket (hasonló táblázat több forrásban fellelhető):

Szimmetrikus kulcsú rejtjelzés	ECC	RSA (műveletben)
80	161	1024
96	192	1500
112	225	2048
128	257	3072

Az elektronikus aláírási rendszerekre az intenzív kutatások eredményei megközelítik az ETSI ALGO munkacsoportja által előírt 1020 bites modulus törési lehetőségét. A jelenlegi információk szerint a még (hosszú) kutatási fázisban lévő

TWINKLE és SHARK berendezések nincsenek elérhető közelségben (átlag támadóknak), így a jelenlegi 1020 bit körüli RSA-t használható rendszerek a közeljövőben még használhatók, de az új rendszereknél ajánlatos a nagyobb (például 2048 bites) modulus használata.

Az ETSI munkacsoport – nem hivatalos, informális (2005-ös) – tájékoztatása szerint a szabványos hash függvények (pl. SHA1) még használhatók. A bemutatott eredmények még nem indokolják az ajánlások érvénytelenítését, mivel a kutatási eredmények csak rendkívül számításigényesen használhatók két, aláírási szabványformátum szerint strukturált üzenet generálására, melyek lenyomata (hash képe) megegyezik.

A fentiek mutatják az igényt a kalkulációs biztonság pontosabb meghatározására, az ennek megfelelő garanciák elérésére. A vizsgálatok alapján indokolható, hogy 80 bites kulcsméret (2^{80} esetszám) már akár elegendő is lehet szimmetrikus kulcsú rejtjelzés esetén, de az is érthető hogy ma miért tartják minimumnak a kb. 2^{120} eset kipróbálási igényét.

Az 1990-es évektől új kutatási irányként jelent meg a kriptográfiai szakirodalomban a (NP elméleti módszerek felhasználásával) bizonyított biztonságú primitívek gyakorlati számításigény-becslési megközelítése.

M. Bellare, P. Rogaway: „Random oracles are practical: A paradigm for designing efficient protocols", In Proceedings of the First Annual ACM Conference on Computer and Communications Security, 1993; M. Bellare, P. Rogaway: „Optimal asymetric encryption – How to encrypt with RSA", EUROCRYPT'94; M. Bellare, P. Rogaway: „The exact security of digital signatures: How to sign with RSA and Rabin", EUROCRYPT'96. munkáikban nem aszimptotikus, hanem rögzített támadási erőforrás mellett vizsgálták a módszerek biztonságát.

9. Kvantum-kriptográfia és kriptográfiai biztonság

A klasszikus kriptográfia a makrovilág fogalmaival, a makrovilág lehetőségeit felhasználva tesz kísérletet az információvédelem problémáinak megoldására.

Ebből a keretrendszerből kilépve egy esetleges támadónak olyan lehetőségek állhatnak rendelkezésére, amelyek érvénytelenítik az addigi biztonsági megfontolásokat, illetve a védelem teljesen új dimenziói nyílnak ki.

Alapgondolat: a szubatomikus részecskék bizonyos körülmények között hullámként viselkednek, pl. az elektronok nem szigorúan egyetlen pontban, hanem a térben „elkenődve" helyezkednek el. Pozíciói hullámfüggvénnyel jellemezhetők, mely leírja, hogy a részecske milyen valószínűséggel található a tér különböző pontjaiban, a hullámfüggvények szuperpozíciói pedig egyszerre sok állapotot (műveletet) reprezentálhatnak.

A kvantumelmélet az adatvédelmet (ezen belül elsősorban a kriptográfia módszereit) a szakirodalom szerint három területen segítheti (a jövőben):

- kvantum számítások (quantum computation), kvantum számítógéppel;

- kvantum-kriptográfia (quantum cryptography);
- teleportálás és biztonságos kommunikációs csatorna kialakítása (quantum teleportation and communication).

9.1. Kvantum-számítógépek (számítások)

segítségével a hagyományos komputerekkel exponenciális idő alatt elvégzett számítások polinomiális időben elvégezhetőkké válnak.

A kvantum számítógép alapgondolata:

Az elemi információs egység – a hagyományos bit („binary digit") mintájára – a *qbit* („quantum bit"). Ezt elemi részecskék testesítik meg, amelyeknek bizonyos állapotát – például a forgásuk irányát – feleltetik meg az „1" illetve a „0" állapotnak. Ezek az elemi részecskék a kvantumelmélet törvényeit kihasználva *szuperpozíciós állapotba hozhatók*. Ilyenkor egyszerre veszik fel az „1" és a „0" értéket.

Az ilyen qbiteken végzett műveletek ekkor tulajdonképpen egyszerre minden lehetséges értéken végrehajtódnak. Ez egy qubit esetében 2 művelet egyidejű elvégzését jelenti, míg n qbit esetén már 2^n művelet egy lépésben való elvégzéséről van szó. A kvantumszámítógépekben a qubitek számának lineáris növelése exponenciális mértékben növeli a párhuzamosan elvégezhető műveletek számát.

Papíron sikerült már olyan gépet tervezni (1994, Peter W. Shor – AT&T Bell Labs), ami kvantummechanikai segédlettel képes arra, hogy egy **szám prímfelbontását** a számjegyek számának négyzetével arányos időben elvégezze. Hagyományos módon ez csak exponenciális idejű algoritmussal valósítható meg.

15 prímtényezőkre bontásához egy legalább hét kvantumbites kvantumszámítógép szükséges.

Az IBM kémikusai ezért olyan molekulát hoztak létre, amelyben a hét kvantumbitet hét magspin – öt fluor- és két szénatommag mágneses momentuma – alkotja. A kutatók kis üvegfiolában milliárdszor milliárd (10^{18}) ilyen molekulát tartalmazó oldatban futtatták le a Shor-féle algoritmust. A kvantumszámítógép működése során az információ beírása (a spinek beállítása) rádiófrekvenciás impulzusokkal, az eredmény kiolvasása pedig magmágneses rezonancia módszerrel (NMR) történt. Meg is kapták a helyes eredményt, miszerint 15 prímtényezői: 3 és 5.

9.2. Kvantum-kriptográfia (kulcsegyeztetés)

elsősorban a hagyományos titkosítás kulcsegyeztetése területén hoz teljesen új lehetőségeket:

- lehetséges olyan csatorna, ahol bármilyen mérési kísérlet szükségszerűen, észlelhető módon megzavarja a jelet;
- bizonyos fizikai jellemzők komplementer módon léteznek: az egyik mérése megzavarja a másikat.

Az egyik – egyelőre a kísérletekben – általánosan használt eljárás, a BB84 algoritmus menete:

1. Alice, a küldő – általa ismert – polarizációjú fotonokat generál és küld Bobnak;

2. Bob előre meghatározott méréssorozatot végez a kapott fotonokon;
3. Bob a sikeres mérések típusát nyilvános csatornán elküldi küldőnek;
4. Alice elküldi, melyik mérés típusa volt megfelelő;
5. ezek alapján a megfelelő típusú mérések eredményéből a közös kulcsbitsorozat úgy előállítható, hogy egy illetéktelen belehallgatás detektálható, mivel a csatorna támadója nem tudja egyszerre mérni a kétféle polarizációtípust, így nem minden esetben küld „ugyanolyan" fotont tovább (címzett felé), mint amit kapott.

Ehhez biztonsági szempontból fontos, hogy:

- küldőnek szűrőkkel le kell csökkenteni a fény intenzitását villanásonként átlagosan kevesebb mint egy fotonra (mivel az intenzitás-csökkentéssel négyzetesen csökken a lehallgatás lehetősége);
- csökkenteni kell az ún. „sötét számolás" kockázatát, mivel a jelenleg használatos detektorok néha jeleznek akkor is, amikor valójában nem is érkezett be foton, így az ehhez hasonló hibák „lehallgatást" jeleznek, mely megzavarja a kommunikációt.

A fenti problémák mellett is jelentős gyakorlati eredmények vannak a kvantum kulcsküldés területén:

- 2002. Los Alamos National Laboratory: 10 km, levegőben,
- 2002. Svájc 60 km, optikai szálon,
- 2002. QnetiQ Lab és a müncheni Ludwig Maximilian egyetem 23 km, levegőben.

Az Európai Unió 11 millió eurót különít el négy évre, hogy kifejlesszenek egy biztonságos kommunikációs rendszert, amely a kvantum-kriptográfiára épül. Kereskedelmi forgalomban is kaphatóak már olyan eszközök, amelyek ezen az elven valósítják meg a kulcsegyeztetést (www. idquantique.com).

9.3. Kvantum teleportálás (kommunikáció)

egyik módszere az EPR-hatáson (Einstein-Podolsky-Rosen) alapul:

Léteznek olyan részecskepárok, amelyek mindaddig elválaszthatatlan kötelékben (korrelációban, szuperpozícióban) maradnak, amíg – egy méréssel – „drasztikusan" szét nem választják őket. Az EPR elve:

- egy középpontosan szimmetrikus atom két fotont bocsát ki ellenkező irányba a két megfigyelő felé ismeretlen polarizációval;
- ha küldő és címzett ugyanolyan típusú mérést (+) hajt végre rajtuk, akkor egyforma valószínűséggel kapják az egyik eredményt (−), azonban ekkor a másik pont az ellenkezőjét (|) kapja és viszont;
- mindkét foton polarizációja csak akkor határozódik meg, amikor valamelyikét megmérjük, a távolságtól függetlenül.

- Első próbálkozások: 1982–84-ben voltak.

- Működő prototípus: 1989. IBM.

A módszert *Charles Bennett* 1989-ben Montreálban, az IBM laboratóriumában négy kollégájával együtt a gyakorlatban is megvalósította. Az első prototípusban zöld lézert használtak, és a fotonokat kb. 30 cm távolságra tudták továbbítani.

- 1991-ben az Oxfordi Egyetemen Arthur Ekert az „összetartozó" kvantum rendszerek – mint például az egyszerre létrehozott foton-párok – azon speciális tulajdonságát használta ki, hogy az egyes kvantumok megőrzik bizonyos közös tulajdonságaikat a szétválásuk után is. 1994-ben sikerült egy ezen az elven működő rendszert a gyakorlatban is megvalósítani.

Néhány web-cím, ahol további részletek olvashatók:
`http://www.qubit.org/library/intros/comp/comp.html`
`http://www.qubit.org/library/intros/compSteane/qcintro.html`
`http://www.qubit.org/library/introductions.html`
`http://www.titoktan.hu/_raktar/honlapok.htm`

10. Rendszerszemléletű biztonság

Az eddigiekben elsősorban a kriptográfiai primitívekre vonatkozó biztonsági fogalmakat tekintettük át. Ahogy a korábbi fejezetekben szerepelt, a felhasználói rendszerek kriptográfiai alrendszerének hierarchikusan egymásra épülő elemei:

- a kriptográfiai primitívek;
- kriptográfiai sémák;
- kriptográfiai protokollok;
- kriptográfiai rendszer;
- informatikai rendszer.

Nem elégséges önmagában vizsgálni a primitíveket, hanem azok biztonsága sokszor alkalmazás-függő.

10.1. Sémák és protokollok

Jó példa a primitív és protokoll illesztetlenségéből adódó veszélyekre a „legerősebb" primitív, a OTP és az ún. kétkulcsos láda együttes használata:

Legyen a két kommunikáló fél: A, B.

Tegyük fel, hogy A egy M üzenetet kíván titkosan küldeni B-nek, melynek hossza N.

Mindkét fél generál magának nagy mennyiségű véletlen elemet, mely véletlen sorozatokat kizárólag a generáló fél (A, vagy B) ismer.

Jelölje: Va és Vb a két oldalon generált sorozatból kivett N hosszú véletlen sorozatot.

Ha A küld egy M üzenetet titkosan B-nek, akkor a protokoll szerint a következő lépéseket kell végrehajtaniuk:

Lépés	A számol	irány	Vonalon utazik	B számol
1.	$M + Va$			
2.		$\rightarrow$	$M + Va$	
3.				$M + Va + Vb$
4.		$\leftarrow$	$M + Va + Vb$	
5.	$M + Va + Vb - Va$			
6.		$\rightarrow$	$M + Vb$	
7.				$M + Vb - Va = M$

A protokoll (leszámítva középen aktív támadás módszerét) megfelelő, mégis a jó protokoll és a jó primitív együtt passzív (lefigyeléses) módszerrel is támadható. A 4. és 2. lépés adatainak kivonásával a támadó is megszerezheti Vb-t, amiből a 6. lépésben utazó információból kiszámíthatja M-et!

A nyilvános kulcsú rendszerekkel a legnagyobb probléma az ún. „rosszindulatú postás", az „intruder in the middle", vagy más elnevezéssel az „attack in the middle" támadás kivédhetetlensége. (Ezért olyan bonyolultak az aláírási rendszerek, a két kommunikáló fél mellett be kell vonni egy ún. TTP (Trusted Third Party) résztvevőt, amelyet az aláírási rendszerekben CA-nak (Certificate Authority-nek) neveznek.

A kétkulcsos láda fenti modelljében B-t helyettesítheti a postás (egy intruder), aki először elvégzi a protokoll-lépéseket (B-nek hazudva magát) A-val, majd utána B-vel is. Erről részletesebben majd a későbbiekben az ún. „grand master chess attack"-nál.

Az alábbiakban – példaszerűen – néhány protokolltámadási kutatási irányt említünk meg:

- „New Attacks on PKCS#1 v1.5 Encryption" című, a EUROCRYPT 2000 konferencián elhangzott előadásukban Jean-Sébastien Coron, Marc Joye, David Naccache, and Pascal Paillier foglalkoztak a címben jelzett szabvány eljárás támadási lehetőségével.
- A CRYPTO 2003 konferencián Brice Canvel, Alain Hiltgen, Serge Vaudenay, and Martin Vuagnoux „Password Interception in a SSL/TLS Channel" címmel a MAC használatának CBC módban lévő gyengesége javításával foglalkozik.
- A CRYPTO 2002 konferencián Jakob Jonsson and Burton S. Kaliski Jr. „On the Security of RSA Encryption in TLS" című előadásukban vizsgálták a TLS-ben alkalmazott RSA protokollrészét, a hand-shake alkalmazásának biztonságát.
- Az ASIACRYPT 2001 konferencián Phong Q. Nguyen and Igor E. Shparlinski „On the Insecurity of a Server-Aided RSA Protocol" c előadásukban a SASC protokoll támadásáról szóltak (passiv attack-kal)
- A CRYPTO 2001 konferencián James Manger „A Chosen Ciphertext Attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as Standardized in PKCS #1 v2.0" címmel tartott előadást.

- A szakmai közvéleményt foglalkoztatja a kvantum-kriptográfia jövője, ezen belül az egyre inkább gyakorlati alkalmazási közelségbe kerülő kvantum-kulcscsere biztonsága. Ennek problematikájáról szólt Donald Beaver előadása a EUROCRYPT 2002 konferencián „On Deniability in Quantum Key Exchange" címmel.

10.2. Kriptográfiai alkalmazásokat futtató informatikai rendszerek biztonsága

Nyilvánvaló, hogy biztonságos primitívek, sémák, protokollok sem teljesíthetik védelmi feladataikat nem biztonságos rendszerkörnyezetben.

10.3. Egyéb támadások elleni ellenállóképesség

Az utóbbi időben nagyon sokan vizsgálják az RSA támadását fizikai eszközök igénybevételével is, az ún. power attack, vagy a timing attack segítségével. Ezzel a két támadási típussal hatékonyan támadhatók pl. megfelelő paraméterekkel rendelkező, algoritmikus támadások ellen biztonságos RSA rendszerek is.

Például a „Public Key Cryptography: 5th International Workshop on Practice and Theory in Public Key Cryptosystems, Paris, France, February 2002. konferencián elhangzott *Side-channels attacks* szekcióban ***Roman Novak*** előadó részéről az **„SPA-Based Adaptive Chosen-Ciphertext Attack on RSA Implementation"** c. előadásban szerepelt a jobboldalon látható, a SmartCard energiafelvételének méréséből az RSA működésére, titkos paramétereire következtethető ábra.

Szintén a PKC,2002-es konferencián hangzott el **„A Combined Timing and Power Attack"** címmel ***Werner Schindler*** előadása a két támadási módszer hatékony kombinációjáról.

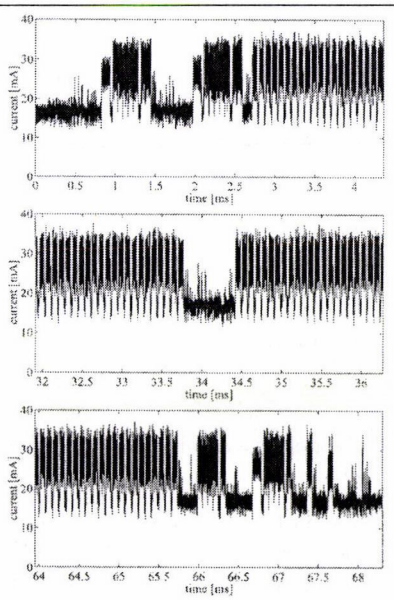


Fig. 1. Typical power consumption patterns during RSA decryption

Ezen támadások ellen is lehet algoritmikus eszközökkel védekezni (a számítási idő rovására), ha kellő programlépéseket kiegészítenek „fal" utasításokkal, melyek az áramfelvételeket, illetve időráfordításokat kiegyensúlyozzák (függetlenítik a kulcsoktól).

AZ ETSI CEN munkacsoportja az elektronikus aláírási követelmények között megjelentette a CWA (Cen Workshop Agreement) 14890–1:2004 előírást, melynek A.5 mellékletében hívja fel a figyelmet a kommunikáló felek közötti protokoll ún. „Grandmaster Chess Attack" sok rendszerben kivédhetetlen módszerére. Ez a módszer közbeékelődő támadással szimulálja mindkét fél felé a lépéseket (mint ha két nagymesterrel egyszerre sakkoznánk, akik nem tudnak egymásról, és az ő lépéseikkel játszunk). A modellt a mellékelt, CWA 14890–1-ből vett ábrával szemléltetjük:

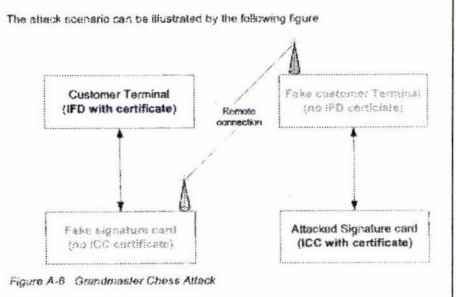


Figure A.6 Grandmaster Chess Attack

Ennek kivédése nagyon bonyolult rendszert igényel. Erre látunk példát I. Zs. Berta, L. Buttyán, I. Vajda: Mitigating The Untrasted Terminal Problem Using Conditional Signatures, IEEE, 2004, april. cikkében, ahol a nembiztonságos terminálok biometriás kiegészítésével, és sűrű időbélyegzéssel nehezítik (teszik lehetetlenné) a közbeékelődő támadó tevékenységét.

Biztonságosnak kell lennie annak a platformnak, ahol a kriptográfiai rendszer működik.

Kritográfiai modulokra a FIPS 140–2 szabvány ír elő követelményeket, általános informatikai környezetre pedig a magyar, európai és világszabvány MSZ ISO/IEC 15408 (Common Criteria: Közös szempontok az informatikai biztonság értékelésére) határozza meg a biztonság elbírálásának szempontjait.

A biztonságos informatikai rendszer (mely nélkül nem működtethető biztonságosan a kriptográfiai alrendszer) a szakirodalom szerint több tényező kezelését igényli:

- *Adminisztratív biztonság*
  - Szervezeti biztonság
  - Személyi biztonság
  - Üzletviteli biztonság

  Kapcsolódó szabványok:

  ISO 9001 (Quality Management systems)

  NIST 800-12 /Kapcsolódó fejezetek: Computer Security Policy, Computer Security Risk Management; Security and Planning in the Computer system Life Cycle; Personal/User issues; Computer Security Incident handling; Awareness, training and Education, Preparing for Contingencies and Disasters/;

  ISO/IEC 17799 / Kapcsolódó fejezetek: Information Security Management: Security Policy; Security Organization; Personal Security; Business Continuity Management/;

- *Fizikai biztonság*

  NIST 800–12 / Kapcsolódó fejezetek:An Introduction to Computer Security; Physical and Enviromental Securiy/

  ISO/IEC 17799 / Kapcsolódó fejezetek: Information Security Management; Physical and Enviromental Securiy/

- *IT Biztonság*

  NIST 800-14

  NIST 800-12 / Kapcsolódó fejezetek: An Introduction to Computer Security/ 16. fejezet Identification and Authentication; 17: Logical Access Control; 18: Audit Trails; 19: Cryptography/

  ISO/IEC 17799 / Kapcsolódó fejezetek: Information Security Management, Communication and Operations Management; Access Control; System Development and Support process/

  Legmeghatározóbb szabvány: MSZ ISO/IEC 15408

Az 1970-es években a DoD és az IBM 40 millió USD-t költött az ún. „tiger team"-ekre, a rendszerek gyenge pontjainak megtalálására.

Ennek alapján állítottak fel biztonsági modelleket, melyek közül a leghíresebb a **Bell–LaPadula modell** (D. E. Bell and L. J. LaPadula, Secure Computer Systems: Mathematical Foundations and Model, Mitrte Corp., Bedford (MA), 1973.), melyben matematikai formalizmussal kezelték a biztonsági elvárásokat.

Ez a modell volt az alapja a **UNIX** biztonsági rendszerének, valamint erős hatással volt a biztonsági követelmények, kiértékelési rendszerek kidolgozására is, és így a TCSEC (Trusted Computer System Evaulation Criteria)-Orange Book sorozata kidolgozására.

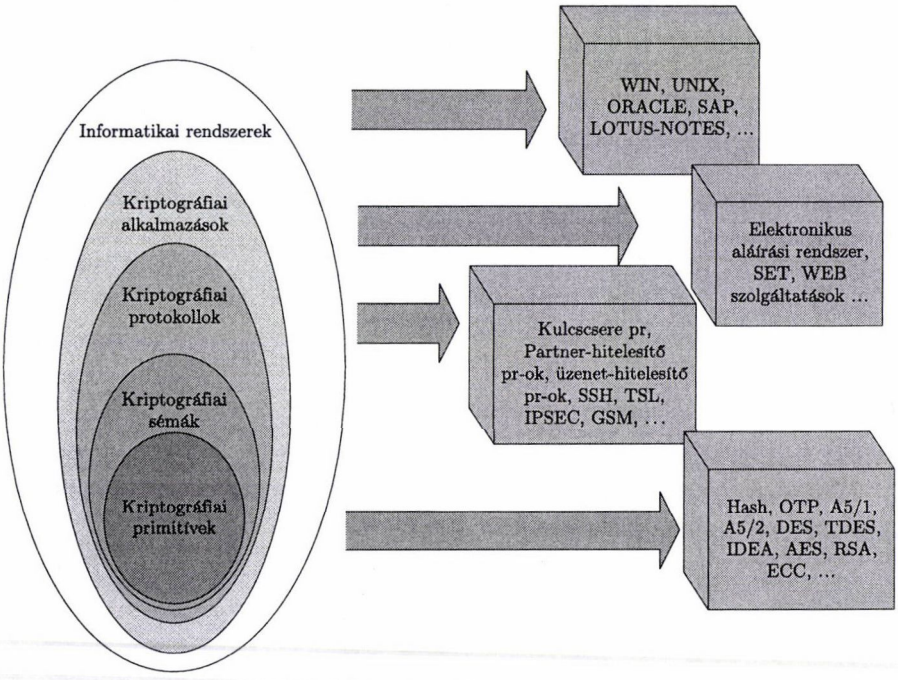
A TCSEC további nemzeti, illetve egyéb közösségi biztonsági módszertanok kidolgozását inspirálta, pl

- **ITSEC:** Information Technology Security Evaulation Criteria (Európai módszertan)
- **FC:** Federal Criteria for Information Technology Security
- **CTCPEC:** Canadian Trusted Computer Product Evaulation Criteria

A fentiekből a nemzetközi testületek összefogásával alakították ki a Common Criteria (röviden CC) követelmény-rendszert, mely magyar, európai és ISO szabvány is lett:

MSZ-ISO/IEC 15408 2002–2003.

11. Összefoglaló ábra



Mellékletek

Kriptográfiai szabványok

Kriptográfiával kapcsolatos szabványokat több szervezet kiadott, ezek között vannak olyan nemzeti szervezetek, melyek szabványait nemzetközi szinten is elfogadják (pl. FIPS), és vannak nemzetközi szervezetek.

Az egyes szervezetek szabványai között jelentős átfedések is vannak, és többségük az interneten ingyenesen hozzáférhető. Ezért itt csak példaként, az eligazodás segítésére említünk néhány szabványt, az érdeklődők a szervezetek honlapjain a teljes listát letölthetik.

ISO: International Organization for Standardization

- ISO 8372 Modes of Operation for 64-bit Cipher
- ISO 9796 Data Integrity Mechanism (MAC)
- ISO 9798 Entity Authetication
- ISO 9779 Register of Cryptographic Algorithm
- ISO 10118 Hash Functions
- ISO 11770 Key Management

- ISO 13888 Non-repudiation
- ISO 14888 Signatures with Appendix

Federal Information Processing Standards Publication

- FIPS PUB 31 Guidelines to ADP Physical Security and Risk Management.
- FIPS PUB 39 Glossary for Computer Systems Security.
- FIPS 46–2 DES
- FIPS 46–3 3DES üzemmódok
- FIPS PUB 73 Guidelines for Security of Computer Applications.
- FIPS PUB 74 Guidelines for Using DES
- FIPS PUB 81 DES Modes of Operation.
- FIPS PUB 87 Guidelines for ADP Contingency Planning.
- FIPS PUB 112 Password Usage.
- FIPS PUB 113 Computer Data Authentication. (CBC MAC)
- FIPS PUB 140–1,–2 Security Requirements for Cryptographic Modules.
- FIPS PUB 185 Key Escrow (Clipper and Skipjack)
- FIPS 186 Secure Hash Algorithm ANSI X9.42, Agreement of Symmetric Keys on Using Diffie–Hellman and MQV Algorithms
- FIPS 196 Entity Authentication (asymetric)

American National Standards Institute

- ANSI X.92 Data Encryption Algorithm (DEA)
- ANSI X3.106 Data Encryption Algorithm (DEA) Modes
- ANSI X9.8 PIN Management and Security
- ANSI X9.9 Message Authentication
- ANSI X9.23 Encription of Messages
- ANSI X9.28 Multi-center Key Management
- ANSI X9.30–1 Digital Signature Algorithm (DSA)
- ANSI X9.30–2 Secure Hash Algorithm (SHA) for DSA
- ANSI X9.31–1 RSA Signature Algorithm
- ANSI X9.31–2 Hashing Algorithm for RSA
- ANSI X9.42 Agreement of Symmetric Keys on Using Diffie–Hellman and MQV Algorithms
- ANSI X9.52 Triple Data Encryption Algorithm Modes of Operation
- ANSI X9.57 Certificate Management

Public Key Criptographic Standards

- RSA PKCS#1 RSA Encription Standard

- RSA PKCS#3 Diffie–Hellman Key-aggreement Szandard
- RSA PKCS#5 Pasword-based Encryption Standard
- RSA PKCS#7 Cryptographic Message Syntax Standard
- RSA PKCS#11 Cryptographic Token Interface Szandard

Request for Comments /Internet „szabványok"/

- RFC 1321 MD5 Hash Function
- RFC 1421 PEM Encription, autjentication
- RFC 1938 One-time Pasword System
- RFC 2246 Tranport Layer Security, TLS V1.0
- RFC 2808 HTTP over TLS
- RFC 2510 Internet X.509 Public Key Infrastructure Certificate Management Protocols
- RFC 2104 HMAC: Keyed Hashing for Message Authentication
- RFC 2402 IPSEC Cryptography
- RFC 2406 IPSEC Authentication

Néhány fontos magyar és egyúttal ISO szabvány az IT rendszerek biztonságához kapcsolódva

- *MSZ ISO/IEC 13888-1:2001* Információtechnika. Biztonságtechnika
- *MSZ ISO/IEC 17799:2002* Informatika. Az informatikai biztonság menedzselésének eljárásrendje
- *MSZ ISO/IEC 9594-8 Informatika. Nyílt rendszerek összekapcsolása. Névtár: Nyilvános kulcs és attribútum tanúsítási keretrendszer*
- *MSZ ISO/IEC TR 13335-1* Informatika. Irányelvek az IT-biztonság menedzseléséhez
- *MSZ ISO/IEC 15408-1:2002 Informatika. Biztonságtechnika. Az informatikai biztonságértékelés közös szempontjai*
- *MSZ EN 45011:1999 Terméktanúsítási rendszereket működtető szervezetre vonatkozó általános követelmények,* és az ezt kiváltó *MSZ ISO/IEC 2700:2006*
- *MSZ EN 17025:2001* Vizsgálólaboratóriumok működésének általános feltételei

Néhány fontosabb hivatkozás

[1] Dan Boneh, Twenty Years of Attack on the RSA Cryptosystem, *Notices of the AMS,* **46(2)** (1999), 2003–213, `http://crypto.stanford.edu/~dabo/abstract/RSAattack-survey.html`

[2] Buttyán Levente, Vajda István, *Kritográfia és alkalmazásai,* Typotex Elektronikus Kiadó Kft. (2004).

[3] Kahn, D., *The Coedebreakers,* New York (1996).

[4] Knuth, D. E., *A számítógép-programozás Művészete 2. Szeminumerikus algoritmusok,* Műszaki Kiadó (Budapest, 1987).

[5] Alfred J. Menezes, Paul C. van Oorshchor, Scott A. Vanstone, *Handbook of Applied Cryptography,* CRC Press (1997), http://www.cacr.math.uwaterloo.ca/hac/

[6] Nemetz Tibor, Vajda István, *Algoritmusos adatvédelem,* Akadémiai Kiadó (Budapest, 1991).

[7] Bruce Schneier, *Applied Cryptography: Protokols, Algorithms, and Source Code in C,* John Wiley & Sons (New York, 2nd edition, 1996).

[8] Bruce Schneier, Adam Shostack, *Breaking Up Is Hard To Do: Modeling Security Threats for Smart Cards,* http://www.counterpane.com/smart-card-threats.html

[9] Shannon, C. E., Communication theory of secrecy systems, *Bell Syst. Techn. J.,* **28** (1949), 656–715.

[10] Virrasztó Tamás, *Titkosítás és adatrejtés, Biztonságos kommunikáció és algoritmikus adatvédelem,* NetAkadémia Oktatóközpont (Budapest, 2004).

DIFFERENT APPROACHES TO THE SECURITY OF CRYPTOGRAPHY – BASED SECURITY MECHANISMS

PÁL PAPP, ISTVÁN SZABÓ

The goal of this paper is to give a broad overview of the main security problems of cryptography, and the different tools we have to handle these problems. In this paper we analyze the security of different crypto mechanisms focusing ont he security of Public Key Infrastructure and its elements. We overview and classify the threats, and the well-known attacking method, we deal with the different ways to measure cryptographic security. We also examine the information-theoretic approaches, and provable security. We also classify different sets of cryptographic primitives. After the theoretic approach, the topics of crypto standards, application, protocols are also dicsussed.