Titkosítási rendszerek CCA-biztonsága

Doktori (PhD) értekezés

szerző: MÁRTON Gyöngyvér

témavezető: Dr. Pethő Attila

Debreceni Egyetem
Természettudományi Doktori Tanács
Informatikai Tudományok Doktori Iskola

Debrecen, 2013

Ezen értekezést a Debreceni Egyetem Természettudományi Doktori Tanács Informatikai Tudományok Doktori Iskola Elméleti számítástudomány, adatvédelem és kriptográfia programja keretében készítettem a Debreceni Egyetem természettudományi doktori (PhD) fokozatának elnyerése céljából.
Debrecen, 20 . . . . . .

. . . . . .
Márton Gyöngyvér
jelölt

Tanúsítom, hogy Márton Gyöngvér doktorjelölt 20 . . . - 20 . . .
között a fent megnevezett Doktori Iskola Elméleti számítástudomány, adatvédelem és kriptográfia programjának keretében irányításommal végezte munkáját. Az értekezésben foglalt eredményekhez a jelölt önálló alkotó tevékenységével meghatározóan hozzájárult. Az értekezés elfogadását javasolom.
Debrecen, 20 . . . . . .

. . . . . .
Dr. Pethő Attila
témavezető

Titkosítási rendszerek CCA-biztonsága

Értekezés a doktori (PhD) fokozat megszerzése érdekében az informatika tudományágban

Írta: Márton Gyöngyvér okleveles informatikus

Készzült a Debreceni Egyetem Informatikai Tudományok Doktori Iskolája, Elméleti számítástudomány, adatvédelem és kriptográfia programja keretében

Témavezető: Dr. Pethő Attila

A doktori szigorlati bizottság:

elnök: Dr. ....

tagok: Dr. ....

Dr. ....

A doktori szigorlat időpontja: 20... ....

Az értekezés bírálói:

Dr. ....

Dr. ....

Dr. ....

A bírálóbizottság:

elnök: Dr. ....

tagok: Dr. ....

Dr. ....

Dr. ....

Dr. ....

Az értekezés védésének időpontja: 20... ....

Tartalomjegyzék

1. fejezet

Bevezető

1.1. Titkosítási alapfogalmak

A kriptográfián belül két, különböző elven működő titkosítási rendszert különböztetünk meg: a szimmetrikus vagy titkos kulcsú (private-key/symmetric encryption) és az aszimmetrikus vagy nyilvános kulcsú titkosítási (public-key/asymmetric encryption) rendszert. A gyakorlatban legtöbbször hibrid rendszereket alkalmaznak, azaz ezek együttes használata merül fel, ha két egymástól távol eső fél (számítógép, táblagép, mobiltelefon) szeretne bizalmas információcserét megvalósítani. Ezt elsősorban azért teszik, mert a szimmetrikus titkosítók hatékonysága messzemenően jobb, mint az aszimmetrikus titkosítókké, a szimmetrikus titkosítók, pedig nem tudják megoldani a két távoli fél biztonságos kulcscseréjét.

Mindkét rendszer esetében a titkosítandó információt a szakirodalom nyílt szövegnek (plaintext) nevezi, a titkosított információt, pedig rejtjelezett szövegnek (ciphertext) ([29], [18], [44]). Ahhoz, hogy a rendszerben a megfelelő titkosított információt létrehozzuk egy kulcsnak nevezett információ kerül felhasználásra, amelynek alkalmazási módja, a fent leírtak alapján két nagy csoportra osztja a titkosító algoritmusokat. A rejtjelezett szöveg előállítását rejtjelezésnek vagy titkosításnak (encryption) nevezzük, a nyílt szöveg visszaállításának folyamatára a visszafejtés (decryption) megnevezést fogjuk használni. A Kerckhoffs-elv szerint [37] a rendszerek által

előírt titkosító algoritmusok nyilvánosak, sőt standard előírások alapján kell őket implementálni, protokollokban felhasználni. A titkosító rendszerek esetében éppen ezért pontosan specifikált formában kell megadni a kulcsgenerálás, titkosítás és visszafejtés algoritmusait, illetve matematikai eszközökkel kell bizonyítani ezen algoritmusok helyes működését és biztonságát.

Titkosító rendszerek esetében, tulajdonképpen kétféle biztonságról beszélhetünk, feltétel nélküli biztonságról vagy információelméleti biztonságról (information-theoretic security) vagy másképpen mondva tökéletes biztonságról (perfect security) és számítástechnikai biztonságról (computational security). Ezen fogalmak alatt a következő bekezdésekben leírtakat értjük ([7], [45]).

Az információelméleti biztonság esetében a kriptorendszert támadó fél számítási kapacitására nem adunk korlátot, azaz azt feltételezzük, hogy a támadó korlátlan számítási kapacitással rendelkezik. A fentiek alapján, informálisan azt mondjuk, hogy egy titkosító rendszer tökéletesen biztonságos, ha egy támadó korlátlan számítási kapacitását feltételezve sem törhető fel a rendszer. Az OTP (one-time pad) titkosító rendszert leszámítva, amelynek egyik változatát Vernam mutatta be 1917-ben [63], és amely implementálás szempontjából számos nehézséget vet fel, a létező titkosító rendszerek egyike sem rendelkezik információelméleti biztonsággal.

Gyakorlati megvalósíthatóság szempontjából a modern kriptográfia feladja az olyan titkosító rendszerek szerkesztésének igényét, amelyek tökéletesen biztonságosak. Az információelméleti biztonság helyett a gyakorlatban is kivitelezhető számítástechnikai biztonságú rendszerek megvalósítása a cél.

A számítástechnikai biztonság azt jelenti, hogy a kriptorendszerek biztonságát olyan feltételek mellett garantálják, amikor a támadónak számítástechnikai kapacitása korlátozott. Ez oly módon valósítható meg, hogy a rendszerek biztonságát olyan matematikai problémák biztosítják, amelyek megoldása nehéz, pontosabban megoldásukra nem ismert polinomiális idejű algoritmus. Számítástechnikai biztonság esetében, természetesen a támadó erőforrásairól is azt feltételezzük, hogy polinomiálisan korlátozottak. Éppen ezért egy kriptorendszer számítástechnikai biztonsága szoros kapcsolatban áll a visszavezethetőség fogalmával, azaz ha

egy kriptorendszer biztonsága visszavezethető valamely jól ismert feladatra, amely nem oldható meg polinomiális időben, akkor azt mondjuk, hogy a kriptorendszer számítástechnikailag biztonságos.

Ezek alapján természetesen merül fel az NP-teljes nehézségű matematikai feladatok alkalmazása kriptográfiai rendszerekben, Az első titkosító rendszerek között, éppen ezért számos olyan rendszert is találunk amelyeknek hátterében NP-teljes problémájú feladat áll. Legismertebb ezek között a hátizsák feladaton alapuló titkosító rendszer [42]. Ezen feladatok alkalmazása, azonban nem mindig vezetett sikerre, mert az NP-teljes feladatok megoldhatósága általában a legrosszabb bemenet esetén számít nehéz feladatnak, a kriptográfiában pedig olyan matematikai problémák alkalmasak, amelyek az átlagos esetben is nehezen megoldhatónak számítanak [57]. A nagy dimenziójú rácsokban, a legrövidebb vektor megkeresésén alapuló kriptorendszerek, azonban kivételt képeznek. Az 1990-es évek végén, több ilyen kriptorendszert is kidolgoztak, például az Ajtai-Dwork kriptorendszert [2]. Ezeknek a rendszereknek a kidolgozása több szempontból is lényeges, az egyik szempont, hogy fontos szélesíteni a nehéz matematikai problémákon alapuló kriptorendszerek területét, a másik szempont, hogy ezek a rendszerek lényegesen gyorsabbak, mint a széleskörben elterjedt, a faktorizáción és diszkrét logaritmuson alapuló kriptorendszerek.

Az értekezés további részében a titkosító rendszerek biztonsága alatt mindig számítástechnikai biztonságot fogunk érteni.

A titkosító rendszerek esetében a támadó lehetőségei különböző számítástechnikai biztonsági szintet határoznak meg, általában négy nagy csoportba szokták felosztani a lehetséges támadási típusokat [40]:

- rejtjelezett szöveg alapú támadás (ciphertext-only attack)
- ismert nyílt szöveg alapú támadás (known-plaintext attack)
- választott nyílt szöveg alapú támadás (chosen-plaintext attack)
- választott rejtjelezett szöveg alapú támadás (chosen-ciphertext attack)

A rejtjelezett szöveg alapú támadás esetén a támadó egyetlen kiindulási pontja a rendelkezésére álló rejtjelezett szöveg, illetve adott esetben a rejtjelezett szövegek. Ez alapján próbálja meghatározni a nyílt szöveget, a rendszerben alkalmazott kulcsot vagy következtetni ezekre.

Az ismert nyílt szöveg alapú támadás esetén a támadó több nyílt szöveg és ezeknek megfelelő rejtjelezett szöveg párral rendelkezik. A támadó ezeknek az információknak a birtokában próbál következtetni a rendszerben alkalmazott kulcsra vagy egy rejtjelezett szöveg tartalmára.

A jelenlegi szabványok szerint, azok a rendszerek, amelyek nem állnak ellen a fenti két támadási típusnak, a legkisebb biztonsági szintet sem biztosítják. A fenti két támadási típust passzív támadásként kategorizálja a kriptográfia ([7], [36], [24]).

A választott nyílt szöveg alapú támadás esetében a támadónak lehetősége van tetszőleges számú rejtjelezést végezni, a saját maga által kiválasztott, szerkesztett nyílt szövegen és ezek alapján próbál következtetni a rendszer bemeneti paramétereire, azaz a kulcsra vagy a nyílt szövegre. A továbbiakban azokat a rendszereket, amelyek biztonságosak a választott nyílt szöveg alapú támadással szemben CPA-biztonságú rendszereknek fogjuk hívni.

A választott rejtjelezett szöveg alapú támadás esetében a támadónak lehetősége van tetszőleges számú visszafejtést végezni, a saját maga által kiválasztott, szerkesztett rejtjelezett szövegen és ezek alapján próbál következtetni a rendszer bemeneti paramétereire, azaz a kulcsra vagy a nyílt szövegre. A továbbiakban azokat a rendszereket, amelyek biztonságosak a választott rejtjelezett szöveg alapú támadással szemben, CCA-biztonságú rendszereknek fogjuk hívni.

Az utolsó két támadási típus esetében megkülönböztetjük ezek adaptív formáját is, mint sokkal erősebb támadási típusokat, ami azt jelenti, hogy a támadónak arra is lehetősége van, hogy a kiválasztásra kerülő nyílt, illetve rejtjelezett szöveget az alapján határozza meg, hogy mi volt a korábbi rejtjelezés, illetve visszafejtés eredménye. Ez utóbbi támadási típust aktív támadásként tartja számon a kriptográfia ([7], [36], [24]).

Fontos megjegyezni, hogy napjainkban is számos olyan titkosító rendszert használnak, amelyek nem állnak ellen a választott rejtjelezett szöveg alapú támadásnak. Sokáig ez a támadási típus, valóban csak az elméleti kriptográfiának jelentett megoldatlan problémát, de az 1998-as RSA elleni sikeres Bleichenbacher-támadás óta [9] az aszimmetrikus titkosító rendszerek alkalmazásakor figyelembe kell venni a választott rejtjelezett szöveg alapú támadással szembeni biztonságot is. Éppen ezért az elmúlt tíz

évben fokozott hangsúly tevődött az aszimmetrikus titkosító rendszerek átalakítására, oly módon hogy azok ellenálljanak a fenti két támadási típusnak, ugyanakkor pedig, hogy megőrizzék a megfelelő hatékonyságukat [50].

Mint említettük, a gyakorlatban titkosító rendszerekként legtöbbször hibrid rendszereket alkalmaznak, ami azt jelenti, hogy a tulajdonképpeni adathalmaz titkosítására szimmetrikus kriptográfiai technikákat alkalmaznak, a titkosító kulcsok cseréjét, pedig aszimmetrikus kriptográfiai technikával oldják meg. Ez utóbbira a szakirodalom bevezette a kulcsbeágyazási mechanizmus (key encapsulation mechanism) elnevezést [21].

Jelen értekezés keretén belül több aszimmetrikus titkosítási rendszert fogunk megvizsgálni, fokozott hangsúlyt fektetve arra is, hogy mit jelent a választott nyílt szöveg alapú, illetve a választott rejtjelezett szöveg alapú támadás egy szimmetrikus titkosítási rendszerben, illetve egy aszimmetrikus titkosítási rendszerben. Mivel a fenti elméleti fogalmakhoz nem kapcsolódik széles körben elfogadott magyar terminológia, az első három fejezetben fokozott hangsúlyt fektetünk az ide kapcsolódó értelmezések és tételek precíz, magyar megfogalmazására.

Az értekezés keretén belül javasolunk egy új kulcsbeágyazási, mechanizmust, amelynek megadjuk a specifikációját, hatékonyságát, bizonyítjuk helyességét, CPA és CCA-biztonságát, és részletezzük implementációs lehetőségeit.

Az értekezésben kitérünk arra is, hogy összehasonlítsuk implementációs lehetőségek, illetve hatékonyság szempontjából az új kulcsbeágyazási mechanizmust az elmúlt években publikált CCA-biztonságú aszimmetrikus rendszerekkel, pontosabban a Cramer–Shoup [20], az RSA–OAEP [6], az SAEP [14], és a Hofheinz- és társai [35] rendszerekkel.

1.2. Áttekintő

Az értekezés első fejezete jelen bevezetőt tartalmazza, amely felvázolja az értekezés főbb mondanivalóját kitérve általános jellegű informális értelmezések, jelölések megadására.

A második fejezet a szimmetrikus titkosítási rendszerek matematikai modelljének bemutatása után rátér a passzív és aktív támadóval szembeni biztonságfogalmak formális értelmezésére. Az aktív támadóval szembeni biztonság esetében különbséget teszünk a választott nyílt szöveg alapú, illetve a választott rejtjelezett szöveg alapú támadások között.

A harmadik fejezetben a hash függvényekkel és üzenethitelesítő kódokkal szembeni követelményeket fogalmazzuk meg, kihangsúlyozva az univerzális hash függvény (universal hash function), a cél ütközésmentes hash függvény (target collision-resistant hash function) és az ütközésmentes hash függvény (collision-resistant hash function) közötti különbségeket, ezt elsősorban azért tesszük, mert az értekezésben tárgyalásra kerülő titkosító rendszerek cél-ütközésmentes hash függvényt használnak.

A negyedik fejezet az aszimmetrikus titkosító rendszerek matematikai modelljének a leírása után megadja az egyirányú függvények értelmezését, majd az értekezésben tárgyalásra kerülő titkosító rendszerek biztonságát meghatározó kriptográfiai feltételezéseket: a faktorizációs feltételezést (factorization assumption), az RSA-feltételezést (RSA assumption), a kvadratikus maradék feltételezést (quadratic residuosity assumption), a diszkrét logaritmus feltételezést (discrete logarithm assumption), és a döntési Diffie–Hellman-feltételezést (Decisional Diffie–Hellman assumption). Az aktív és passzív támadóval szembeni biztonságértelmezések megadása után a hibrid rendszerek biztonságát is tárgyaljuk.

Az ötödik fejezet azokat az aszimmetrikus titkosító rendszereket mutatja be, amelyeknek CCA-biztonságú verzióját implementáltuk, és amelyek hatékonyságát összevetettük az általunk szerkesztett kulcsbeágyazási mechanizmus hatékonyságával. Ezek a rendszerek a következőek:

- az ElGamal- és Cramer–Shoup-rendszer,
- az RSA- és RSA–OAEP-rendszer,
- a Rabin- és SAEP-rendszer,
- a Blum–Goldwasser és Hofheinz- és társai rendszere.

A hatodik fejezet az értekezés legfőbb eredményét az új kulcsbeágyazási mechanizmus CPA, illetve CCA-biztonságú verzióját mutatja be. A kulcsbeágyazási mechanizmus alapjául a Rabin-titkosító [54] és a Hofheinz és

társai [35], által szerkesztett rendszerek szolgáltak. A CCA-biztonságú kulcsbeágyazási mechanizmus a [48] cikkben került publikálásra.

A fejezet keretén belül megadjuk az új mechanizmus specifikációját, helyességét, hatékonyságát, illetve bizonyítjuk a rendszer biztonságát.

A hetedik fejezet az ötödik fejezetben, illetve a hatodik fejezetben leírt CCA-biztonságú rendszerek implementációjára tér ki. Itt megadjuk az implementációhoz használt hardver- és szoftverkonfigurációt, majd rendszerenként kitérünk az implementálás specifikusabb részeire és táblázatokba foglalva feltüntetjük a futási idők eredményeit.

1.3. Definíciók, jelölések

A polinomiális idejű algoritmus, illetve a probabilisztikus és determinisztikus algoritmus fogalmak az értekezés fejezeteiben fontos szerepet fognak játszani. Éppen ezért megadjuk ezek formális értelmezését.

1.1. definíció. *Egy $A(\cdot)$ algoritmus futási ideje, az n bithosszúságú bemenet függvényében polinomiális, ha lépésszáma $\mathcal{O}(n^k)$, ahol n és k nem negatív egész számok. Ekkor azt mondjuk, hogy az algoritmus polinomiális idejű, ([24], [36]).*

1.2. definíció. *Egy $A(\cdot)$ algoritmus, a bemenet függvényében determinisztikus, ha előre meghatározott lépésekből áll, ahol a végrehajtás minden helyzetében egyértelműen meghatározható az aktuális lépés utáni következő lépés, ([24], [36]).*

1.3. definíció. *Egy $A(\cdot)$ algoritmus, az x bemenet függvényében probabilisztikus, ha ([24], [36]):*

- *az x bemenet mellett, egy x-től független, véletlenszerűen generált r bitsorozatot is megadunk, ahol az r bitsorozat a 0 és 1 értékeket egymástól függetlenül, 1/2 valószínűséggel veheti fel,*
- *az $A(x)$ algoritmusnak megadható egy determinisztikus kiterjesztése, amelynek a x bemenet mellett az r bitsorozat is a bemenete lesz, jelöljük ezt $A_D(x, r)$-rel,*

- *az $A(\cdot)$ algoritmus kimenete nem egy konstans érték, hanem egy valószínűségi változó lesz, pontosabban annak az eseménynek a valószínűsége, hogy az A algoritmus az x bemeneten az y kimenetet határozza meg a következő lesz:*

$$\Pr[A(x) = y] = \Pr[\{r | A_D(x, r) = y\}].$$

1.1. megjegyzés. *Egy probabilisztikus algoritmus véletlen döntéseket hozhat a futási ideje alatt, és minden olyan algoritmus, amely véletlen számokat használ probabilisztikus algoritmus.*

1.2. megjegyzés. *A probabilisztikus, illetve determinisztikus algoritmus precízebb definíciója a probabilisztikus, illetve determinisztikus Turing-gép fogalmának bevezetésével adható meg [39], amire jelen értekezésben nem térünk ki.*

Az elhanyagolható függvény fogalma több helyen is megtalálható a kriptográfiai szakirodalomban ([21], [18], [64]), jelen értekezésben ez alatt a következőket fogjuk érteni:

1.4. definíció. *Egy f függvény elhanyagolható (negligible), ha bármely $\epsilon(\cdot)$ polinom esetében létezik egy k_0 egész szám úgy, hogy bármely $k > k_0$ egész számra fennáll: $f(k) < \frac{1}{\epsilon(k)}$.*

Az értekezésben használt jelölések közül, pedig fontosnak tartjuk megemlíteni az alábbiakat.

1. Ha A egy halmaz, akkor $x \xleftarrow{R} A$ jelöli, hogy x véletlenszerűen, egyenletes eloszlás esetén lett kiválasztva.
2. Ha A egy probabilisztikus algoritmus, akkor az $x \xleftarrow{R} A$ jelölés azt jelenti, hogy az A algoritmus x kimenetét véletlenszerűen határozta meg.
3. Az 1^k, k darab egymás mellé írt 1-est jelöl, amelyet a szakirodalomnak megfelelően gyakran fogunk algoritmusoknak megadni bemeneti paraméterként ([21][36]). Ezt a jelölést akkor alkalmazzuk, amikor azt szeretnék jelezni, hogy az algoritmus futási ideje akkor is polinomiális k szerint, ha nincs az algoritmusnak bemenete vagy ez nagyon rövid.

4. A $|\cdot|$ jelöléshez háromféle jelentést is társítunk, melyet az adott helyeken nem jelölünk explicit módon, feltételezve, hogy a kontextus alapján ez kikövetkeztethető. Az említett három jelentéstartalom a következő: abszolútérték, bithossz, és elemszám.
5. Az $||$ operátor összefűzést jelent.

2. fejezet

Szimmetrikus titkosítási rendszerek biztonsága

2.1. Bevezető

Szimmetrikus titkosítás már az ókorban is létezett, azonban a legtöbb XX. század előtti titkosító rendszer (Caesar, Playfair, Vigenere, stb.) a mai számítógépes eszközökkel könnyen feltörhető. A jelenleg, gyakorlatban használt rendszerek (3DES, AES, Salsa20) ([40], [22], [8]) jól meghatározott biztonsági követelményeknek tesznek eleget. Ezen követelmények formális megadásához először szükséges a szimmetrikus rendszerek matematikai értelmezése ([17], [36]).

2.1. definíció. *Jelölje SKE azt a szimmetrikus titkosítási rendszert, amely egy (K, M, C) halmazhármas felett, három algoritmussal értelmezhető, ahol M a nyílt szövegek egy véges halmaza, C a rejtjelezett szövegek egy véges halmaza, K a titkosító kulcsok egy véges halmaza. A három említett algoritmus, pedig a következő:*

- *Gen, a kulcsgeneráló probabilisztikus algoritmus, amely polinomiális időben az 1^k bemeneten meghatározza a key titkosító kulcsot: $key \xleftarrow{R} Gen(1^k)$, ahol $key \in K$, $k \in \mathbb{Z}_{\geq 0}$ a rendszer biztonsági paramétere (legtöbb esetben a generált kulcs bithossza),*

- *Enc_{key} a probabilisztikus rejtjelező algoritmus, amely polinomiális időben végzi a rejtjelezést, a key titkosító kulcs alapján, az $m \in M$ nyílt szövegen, meghatározva a $c \in C$ rejtjelezett szöveget: $c \xleftarrow{R} Enc_{key}(m)$,*
- *Dec_{key} a determinisztikus, visszafejtő algoritmus, amely polinomiális időben végzi a visszafejtést, a key titkosító kulcs alapján a c rejtjelezett szövegen: $m \leftarrow Dec_{key}(c)$.*

A rendszer helyessége érdekében megköveteljük, hogy teljesüljön $Dec_{key}(Enc_{key}(m)) = m$, minden $key \in K$ és minden $m \in M$ esetében.

Fontos megjegyezni, hogy a kulcs és nyílt szöveg értékének a meghatározása egymástól független események, tulajdonképpen előbb rögzítjük a kulcsot, majd aztán kerül meghatározásra a nyílt szöveg.

A szimmetrikus titkosítási rendszerek biztonságának értelmezését többféleképpen is meg lehet adni, elsősorban aszerint csoportosítják a biztonságértelmezéseket, hogy milyen típusú támadóval kell számolni egy adott titkosítási rendszer esetében. A következő fejezetben formálisan megadjuk a passzív támadóval szembeni biztonságértelmezést, majd az aktív támadóval szembeni biztonságértelmezést.

2.2. Passzív támadóval szembeni biztonság

A passzív támadóval szembeni biztonság értelmezéséhez szükség van egy kísérletre. Az alábbi kísérletben az **A** támadó egy probabilisztikus, polinomiális idejű algoritmust jelent, ahol a kísérlet az **A** támadó és környezete, a kihívó között játszódik le.

2.1. kísérlet.

1. *A kihívó a Gen kulcsgeneráló algoritmussal meghatározza a key titkosító kulcsot,*
2. *az **A** támadó, két azonos hosszúságú üzenetet határoz meg, jelöljük ezeket m_0, m_1-gyel, amelyeket átad a kihívónak,*
3. *a kihívó $b \xleftarrow{R} \{0,1\}$ választás alapján az $Enc_{key}(\cdot)$ titkosító algoritmussal meghatározza az m_b titkosított értékét: $c^* \xleftarrow{R} Enc_{key}(m_b)$,*

amelyet átad az **A** *támadónak; nevezzük ezt a* c^* *titkosított értéket kihívó rejtjelezett szövegnek (challenge ciphertext),*

4. *az* **A** *támadó meghatároz egy* $\hat{b} \in \{0,1\}$ *kimeneti értéket.*

2.2. definíció. *Az SKE szimmetrikus titkosítási rendszerben az* **A** *támadó előnye a következőképpen adható meg:* $\mathrm{Adv}^{pas}_{SKE,\mathbf{A}}(k) = |\Pr[b = \hat{b}] - \frac{1}{2}|$, *ahol a* b *és* $\hat{b}$ *értékeket a 2.1. kísérletben határoztuk meg.*

A passzív támadóval szembeni biztonságot a következő értelmezés adja meg [21]:

2.3. definíció. *Az SKE szimmetrikus titkosítási rendszer akkor biztonságos egy passzív támadóval szemben, ha bármely* **A** *probabilisztikus, polinomiális idejű, passzív támadó esetében létezik egy* $f(k)$ *elhanyagolható függvény úgy, hogy*

$$\mathrm{Adv}^{pas}_{SKE,\mathbf{A}}(k) \leq f(k),$$

ahol $\mathrm{Adv}^{pas}_{SKE,\mathbf{A}}(k)$*-t a 2.2. értelmezésnél adtuk meg.*

Az SKE szimmetrikus titkosítási rendszer passzív támadóval szembeni biztonság értelmezésének megadhatjuk egy alternatív definícióját is [21]. Jelöljük a továbbiakban $Expr(0)$-val azt az eseményt, amikor $\hat{b} = 1$, azon feltétel mellett, hogy a kihívónak $b = 0$ volt a választása, és jelöljük $Expr(1)$-gyel azt az eseményt, amikor $\hat{b} = 1$, azon feltétel mellett, hogy a kihívó a $b = 1$ értéket választotta.

Ekkor az **A** támadó előnye a következőképpen értelmezhető:

$$\mathrm{Adv}'^{\,pas}_{SKE,\mathbf{A}}(k) = \tfrac{1}{2}|\Pr[Expr(0)] - \Pr[Expr(1)]|.$$

2.4. definíció. *Az SKE szimmetrikus titkosítási rendszer akkor biztonságos egy passzív támadóval szemben, ha bármely* **A** *probabilisztikus, polinomiális idejű, passzív támadó esetében létezik egy* $f(k)$ *elhanyagolható függvény úgy, hogy*

$$\mathrm{Adv}'^{\,pas}_{SKE,\mathbf{A}}(k) \leq f(k)$$

A két értelmezés közötti ekvivalencia a következő egyenlőség alapján látható be:

$$\begin{aligned}
&\mathrm{Adv}'^{\,pas}_{SKE,\mathbf{A}}(k) && = \\
&\tfrac{1}{2} \cdot |\Pr[\hat{b} = 1 | b = 0] - \Pr[\hat{b} = 1 | b = 1]| && = \\
&\tfrac{1}{2} \cdot |1 - \Pr[\hat{b} = 0 | b = 0] - \Pr[\hat{b} = 1 | b = 1]| && = \\
&\tfrac{1}{2} \cdot |1 - 2 \cdot \Pr[b = \hat{b}]| && = \\
&|\tfrac{1}{2} - \Pr[b = \hat{b}]| && = \\
&\mathrm{Adv}^{pas}_{SKE,\mathbf{A}}(k).
\end{aligned}$$

2.1. megjegyzés. *A későbbi biztonságértelmezések esetében is megadható hasonló alternatív definíció.*

2.3. Aktív támadóval szembeni biztonság

Szimmetrikus titkosítási rendszerek esetében az aktív támadó jelenléte két különböző támadási típust határoz meg, az egyikben a támadó lehetőségei korlátozva vannak, oly módon, hogy csak titkosítást tud végezni, ez lesz a választott nyílt szöveg alapú támadás. A második esetben a támadó tetszőleges rejtjelezett szövegek visszafejtését is tudja kérni, kivéve a kihívó rejtjelezett szöveg visszafejtését. Ezt fogjuk választott rejtjelezett szöveg alapú támadásnak hívni.

Az aktív támadóval szembeni biztonság esetében a támadó, a kihívón keresztül egy titkosító, illetve egy visszafejtő orákulum igényeit veheti igénybe. Ez azt jelenti, hogy a kihívó a megfelelő kulcs felhasználásával, amely természetesen nem ismert a támadó által, megadja a támadó által szolgáltatott bemenetre, a kívánt titkosítás, illetve visszafejtés eredményét.

Megjegyezzük, hogy az orákulumra, mint fekete dobozként viselkedő algoritmusra, a továbbiakban az összes biztonságértelmezésnél szükség van.

2.3.1. Választott nyílt szöveg alapú támadás

A választott nyílt szöveg alapú támadással szembeni biztonság értelmezéséhez szükséges megadni a következő kísérlet lépéssorozatát, amely kísérlet az **A** támadó és környezete, a kihívó között játszódik le.

2.2. kísérlet.

1. *A kihívó a Gen kulcsgeneráló algoritmussal meghatározza a key titkosító kulcsot,*
2. *az* **A** *támadó tetszőleges bemeneten, tetszőleges számú titkosítást végez, ahol a titkosítást a kihívó* $Enc_{key}(\cdot)$ *algoritmusával végzi,*
3. *az* **A** *támadó, két azonos hosszúságú üzenetet határoz meg, jelöljük ezeket* m_0, m_1*-gyel, amelyeket átad a kihívónak,*
4. *a kihívó* $b \xleftarrow{R} \{0,1\}$ *választás alapján meghatározza az* m_b *titkosított értékét:* $c^* \xleftarrow{R} Enc_{key}(m_b)$*, amelyet átad az* **A** *támadónak,*
5. *az* **A** *támadó tetszőleges bemeneten, tetszőleges számú titkosítást végez, ahol a titkosítást a kihívó* $Enc_{key}(\cdot)$ *algoritmusával végzi,*
6. *az* **A** *támadó meghatároz egy* $\hat{b} \in \{0,1\}$ *kimeneti értéket.*

2.5. definíció. *Az SKE szimmetrikus titkosítási rendszerben az* **A** *támadó előnye a következőképpen adható meg:* $\mathrm{Adv}^{cpa}_{SKE,\mathbf{A}}(k) = |\Pr[b = \hat{b}] - \frac{1}{2}|$*, ahol a* b *és* $\hat{b}$ *értékeket a 2.2. kísérletben határoztuk meg.*

A választott nyílt szöveg alapú támadással szembeni CPA-biztonságot a következő értelmezés adja [21]:

2.6. definíció. *Az SKE szimmetrikus titkosítási rendszer CPA-biztonságú, ha bármely* **A** *probabilisztikus, polinomiális idejű támadó esetében létezik egy* $f(k)$ *elhanyagolható függvény úgy, hogy*

$$\mathrm{Adv}^{cpa}_{SKE,\mathbf{A}}(k) \leq f(k),$$

ahol $\mathrm{Adv}^{cpa}_{SKE,\mathbf{A}}(k)$*-t a 2.5. értelmezésnél adtuk meg.*

2.3.2. Választott rejtjelezett szöveg alapú támadás

A választott rejtjelezett szöveg alapú támadással szembeni biztonság értelmezéséhez szükséges megadni a következő kísérlet lépéssorozatát, amely kísérlet az **A** támadó és környezete, a kihívó között játszódik le.

2.3. kísérlet.

1. *A kihívó a Gen kulcsgeneráló algoritmussal meghatározza a key titkosító kulcsot,*
2. *az* **A** *támadó tetszőleges számú titkosítást, illetve visszafejtést végez, a kihívó* $Enc_{key}(\cdot)$ *titkosító és* $Dec_{key}(\cdot)$ *visszafejtő algoritmusával,*
3. *az* **A** *támadó, két azonos hosszúságú üzenetet határoz meg, jelöljük ezeket* m_0, m_1*-gyel, amelyeket átad a kihívónak,*
4. *a kihívó* $b \xleftarrow{R} \{0,1\}$ *választás alapján meghatározza az* m_b *titkosított értékét:* $c^* \xleftarrow{R} Enc_{key}(m_b)$*, amelyet átad az* **A** *támadónak,*
5. *az* **A** *támadó tetszőleges számú titkosítást, illetve visszafejtést végez, a kihívó* $Enc_{key}(\cdot)$ *titkosító és* $Dec_{key}(c)$ *visszafejtő algoritmusával, ahol a visszafejtést különböző c bemenetekre kéri, azzal a megkötéssel, hogy* $c \neq c^*$*,*
6. *az* **A** *támadó meghatároz egy* $\hat{b} \in \{0,1\}$ *kimeneti értéket.*

2.7. definíció. *Az SKE szimmetrikus titkosítási rendszerben az* **A** *támadó előnye a következőképpen adható meg:* $\mathrm{Adv}^{cca}_{SKE,\mathbf{A}}(k) = |\Pr[b = \hat{b}] - \frac{1}{2}|$*, ahol a b és* $\hat{b}$ *értékeket a 2.3. kísérletben határoztuk meg.*

A választott rejtjelezett szöveg alapú támadással szembeni CCA-biztonságot a következő értelmezés adja [21]:

2.8. definíció. *Az SKE szimmetrikus titkosítási rendszer CCA-biztonságú, ha bármely* **A** *probabilisztikus, polinomiális idejű támadó esetében létezik egy* $f(k)$ *elhanyagolható függvény úgy, hogy*

$$\mathrm{Adv}^{cca}_{SKE,\mathbf{A}}(k) \leq f(k),$$

ahol $\mathrm{Adv}^{cca}_{SKE,\mathbf{A}}(k)$*-t a 2.7. értelmezésnél adtuk meg.*

2.4. Szimmetrikus titkosítási rendszerek osztályozása

A szimmetrikus titkosító rendszereken belül megkülönböztetünk folyamtitkosító (stream-cipher) és blokktitkosító (block-cipher) rendszereket. Mindkét rendszer esetében a titkosításhoz szükséges egy titkos adat, a titkosító kulcs. Amiben eltér egymástól a két rendszer az az adathalmaz feldolgozásának módja.

A két különböző titkosítási rendszer természetesen különböző biztonsági előírásokat határoz meg, éppen ezért külön tárgyaljuk a folyamtitkosítókkal, illetve a blokktitkosítókkal szembeni biztonsági követelményeket.

2.4.1. Folyamtitkosító rendszerek

A folyamtitkosító rendszerek alatt azokat a rendszereket értjük, amelyek valamely probabilisztikus algoritmus segítségével, kiindulva egy kezdeti titkos adatból, a titkosító kulcsból, generálnak egy 0 és 1 értékekből álló álvéletlen bitsorozatot. A generált bitsorozatot kulcsfolyamnak nevezik, amelyet aztán legtöbbször bitenkénti $\pmod 2$ összeadást végezve hozzáadnak a nyílt szöveghez. E műveletre a továbbiakban a $\oplus$ jelet fogjuk használni. Megjegyezzük, hogy a kulcsfolyam és nyílt szöveg összeadása történhet $\pmod{2^n}$ szerint is, ahol n pozitív egész szám, pl. a Salsa20 esetében $n = 32$.

Közismert azonban, hogy szoftver szintjén nem lehet véletlen biteket előállítani, ezért azon algoritmusokat amelyekről azt állítjuk, hogy véletlen bitsorozatot állítanak elő úgy hívjuk, hogy álvéletlen bitsorozat generátorok, hiszen az általuk előállított bitsorozatok igazából álvéletlen értékek.

A titkosított rendszer biztonsága függ az álvéletlen bitsorozat generátor kimenetétől, illetve meghatározza a $\oplus$ művelet tulajdonsága. A következő tétel [61] magyarázatot ad arra, hogy titkosító rendszerek esetében miért használható a $\oplus$ művelet:

2.1. tétel. *Legyen $X \in \{0,1\}^k$ és $Y \in \{0,1\}^k$ két független, egyenletes eloszlású valószínűségi változó. Akkor a $Z = X \oplus Y$ valószínűségi változó is egyenletes eloszlású lesz a $\{0,1\}^k$ halmazon.*

A fenti tétel alapján levonhatjuk azt a következtetést, hogy a folyamtitkosítók biztonsága igazából az álvéletlen bitsorozatot generáló algoritmustól függ. Éppen ezért az elmúlt években folyamtitkosító rendszerek alatt azokat az eljárásokat értjük, amelyek meghatározzák az álvéletlen bitsorozat előállításának módját. Ilyen generátorok az RC4 [28], a Salsa20 [8], a Legendre szimbólumot felhasználó Goubin, Mauduit és Sárközy által leírt generátorcsalád [33] vagy a négyzetreemelés függvényen alapuló Blum-Blum-Shub [10] generátor, stb.

A továbbiakban megadjuk az álvéletlen bitsorozat generátor azon értelmezését, amelynek ha eleget tesz az álvéletlen generátor, akkor biztonságosan alkalmazható kriptográfiai rendszerekben ([36], [61]).

2.9. definíció. *Legyen G egy determinisztikus polinomiális idejű algoritmus, amely az $x \in \{0,1\}^k$ bemeneten egy $y \in \{0,1\}^l$ hosszúságú kimeneti bitsorozatot állít elő, ahol $k < l$. Ha bármely probabilisztikus polinomiális idejű D algoritmusra és bármely $r \overset{R}{\leftarrow} \{0,1\}^l$ bitsorozat esetében létezik egy $f(k)$ elhanyagolható függvény, a következő tulajdonsággal:*

$$|\Pr[D(r) = 1] - \Pr[D(G(x)) = 1]| \leq f(k), \text{ ahol } x \overset{R}{\leftarrow} \{0,1\}^k,$$

akkor azt mondjuk, hogy a G algoritmus kriptográfiailag biztonságos álvéletlen bitsorozat generátor.

2.4.2. Blokktitkosító rendszerek

Blokktitkosító rendszerek esetében egyszerre több byte-tömböt titkosítunk, a titkosítás pedig ugyannak a függvénynek vagy permutációnak a többszöri végrehajtását jelenti, a különböző byte-tömbökön. A rendszer biztonságát az alkalmazott függvény vagy permutáció határozza meg, illetve a titkosított byte-tömbök összefűzésének módja.

A byte-tömbök összefűzésének módjára számos jól kidolgozott biztonságos technika létezik [61], amelyre jelen értekezésben nem térünk ki, az alkalmazott függvény vagy permutáció biztonságának vizsgálata, pedig az álvéletlen függvények fogalmához kapcsolódik. Az álvéletlen függvény fogalmának a bevezetéséhez a következőkre van szükség ([7], [24], [36]):

Jelöljük $Func_{X,Y}$-al azon halmazt, amely tartalmazza az összes olyan függvényt, amely az $X = \{0,1\}^N$ bemenetet az Y $= \{0,1\}^M$ kimenetbe képezi le, ahol $|X| = N$ és $|Y| = M$.

Legyen F a következő, két bemeneti paraméterrel rendelkező függvény:

$$F : \{0,1\}^k \times \{0,1\}^N \to \{0,1\}^M.$$

Minden egyes $key \in \{0,1\}^k$-ra, melyet előre rögzítünk meghatározunk egy $F_{key} : \{0,1\}^N \to \{0,1\}^M$ kulcsos függvényt, úgy hogy: $F_{key}(x) = F(key, x)$. A kiválasztott key érték lesz a titkosító kulcs, F_{key} pedig, eleget kell tegyen azon tulajdonságnak, hogy az $F_{key}(x)$, $x \in \{0,1\}^N$ bemenetre polinomiális időben meghatározza az $y \in \{0,1\}^M$ kimeneti értéket. Abban az esetben ha $N = M$, akkor F_{key} permutáció.

Informálisan az F álvéletlen függvény azt jelenti, hogy a függvény polinomiális időben meg tudja határozni tetszőleges bemenete esetén a kimeneti értékét, és nem szerkeszthető olyan polinomiális idejű algoritmus, amely különbséget tudna tenni az F_{key} és egy, a $Func_{X,Y}$ függvénycsaládból véletlenszerűen kiválasztott függvény között.

Az F függvény álvéletlen viselkedését, pedig a következő formális definícióval adhatjuk meg [7]:

2.10. definíció. *Az F függvény álvéletlen függvény, ha bármely probabilisztikus, polinomiális idejű D algoritmusra és bármely $key \xleftarrow{R} \{0,1\}^k$ esetében, létezik egy $f(k)$ elhanyagolható függvény, a következő tulajdonsággal:*

$$|\Pr[D(G(\cdot)) = 1] - \Pr[D(F_{key}(\cdot)) = 1]| \leq f(k),$$

ahol $G \xleftarrow{R} Func_{X,Y}$.

A blokktitkosító rendszereknek a folyamtitkosítókkal szembeni egyik előnye, hogy a titkosítás folyamata párhuzamosítható, a biztonsági kritériumok sokkal jobban tanulmányozottak, hátrányuk, pedig, hogy lassúbbak.

3. fejezet

Hash függvények és üzenethitelesítő kódok

A lehetséges támadási típusok alapján a bizalmas információcsere nem mindig biztosítja egy adott üzenet hiteles voltát. Abban az esetben, ha egy aktív támadóval is számolni kell, akkor a rejtjelezés mellett szükséges mind a küldő fél, mind az elküldött üzenet hitelesítése. Ennek megvalósítására többféle technika is létezik, az egyik a hash függvények alkalmazása, a másik az üzenethitelesítő kódok.

3.1. Hash függvények

A hash függvényeket (hash function) a kriptográfia több területén is alkalmazzák. A kriptográfiában alkalmazott hash függvények az adatok integritásának vizsgálatát biztosítják. Gyakorlatban, leginkább a digitális aláírásoknál alkalmazzák, de a CPA-biztonságú, illetve CCA-biztonságú titkosító rendszereknél is fontos szerepet kapnak.

A hash függvény egy tetszőleges hosszúságú üzenetre egy rögzített hosszúságú kimenetet határoz meg, amelyet ellenőrző összegnek, lenyomatnak is szoktak hívni ([41], [53]).

Jelölje, ahogy a 2.4.2. szakaszban is, $Func_{X,Y}$ az összes olyan függvény halmazát, amelyek értelmezési tartománya X, és értéktartománya Y. Feltételezve, hogy $|X| = N, |Y| = M$ és $N \geq M$, akkor nyilvánvaló, hogy teljesül: $|Func_{X,Y}| = M^N$.

A $H : X \to Y$ hash függvényt mindig a $Func_{X,Y}$ halmazból választjuk ki. Feltételezve, hogy az üzenet, amelynek meg szeretnénk határozni a hash értékét x, akkor $y = H(x)$-szel jelöljük a kiszámolt hash értéket.

A kriptográfiában alkalmazható hash függvények három fontos tulajdonságnak kell eleget tegyenek, ami azt jelenti, hogy az alábbi három feladat egyike se legyen megoldható polinomiális időben:

1. feladat, egyirányú, őskép tulajdonság (preimage, one-way property):
   BEMENET: legyen $H : X \to Y$ hash függvény és $y \in Y$ egy elem.
   KIMENET: az x érték, azzal a tulajdonsággal, hogy $H(x) = y$.
2. feladat, gyengén ütközésmentes, második őskép tulajdonság (second preimage property):
   BEMENET: legyen $H : X \to Y$ hash függvény és $x \in X$ egy elem.
   KIMENET: az x_1 érték, azzal a tulajdonsággal, hogy $x \neq x_1$ és $H(x) = H(x_1)$.
3. feladat, ütközésmentes tulajdonság (collision-resistant property):
   BEMENET: legyen $H : X \to Y$ hash függvény
   KIMENET: az x, x_1 értékek, azzal a tulajdonsággal hogy $x \neq x_1$ és $H(x) = H(x_1)$.

3.1. megjegyzés. *Bármely hash függvény, amely ütközésmentes tulajdonságú ugyanakkor gyengén ütközésmentes tulajdonságú is.*

Bármely hash függvény, amely gyengén ütközésmentes tulajdonságú ugyanakkor egyirányú tulajdonságú is.

A CPA, CCA-biztonság értelmezéseknél a szakirodalom [5] bevezeti a véletlen orákulum modellt, amely a hash függvények egy ideális esetét modellezi. Informálisan ez azt jelenti, hogy a hash függvény, az $Func_{X,Y}$ függvényhalmazból véletlenszerűen kerül kiválasztásra és a hash érték meghatározása egy orákulum megkérdezése során valósítható meg, azaz nem alkalmazható matematikai formula vagy algoritmus a hash érték meghatározására. A valós életben természetesen nem létezik ilyen függvény,

viszont szerkeszthető olyan hash függvény, amelynek viselkedése hasonló a véletlen orákuluméhoz. Formálisan ezt azt jelenti, hogy [61]:

3.1. tétel. *Legyen* $H : X \to Y$ *hash függvény, ahol* $H \xleftarrow{R} Func_{X,Y}$ *és legyen* $X_0 \subseteq X$*. Feltételezzük továbbá, hogy a* $H(x)$ *érték csak az* $x \in X_0$ *esetekben került meghatározásra. Ha a hash függvény viselkedése hasonló a véletlen orákuluméhoz, akkor fennáll, hogy* $\Pr[H(x) = y] = 1/M$*, minden* $x \in X \setminus X_0$ *és* $y \in Y$ *elemre.*

Mivel az ütközésmentes tulajdonság a hash függvények legfontosabb tulajdonsága, az újabb szakirodalom [7] a hash függvények ütközésmentes tulajdonságára vonatkozóan, árnyaltabb értelmezéseket is megad. Ahhoz, hogy ezek formális értelmezését meg lehessen adni, a hash függvény fogalma mellett be kell vezetni a kulcsos hash függvény fogalmát [7].

A kulcsos hash egy olyan függvény, amely a következő hash függvénycsaládból kerül kiválasztásra:

$$Hash : K \times X \to Y.$$

Minden egyes $key \in K$-ra meghatározható egy $H_{key} : X \to Y$ kulcsos hash függvény, ahol a *key* meghatározása egy probabilisztikus, polinomiális idejű algoritmussal történik. Ilyenkor azt mondjuk, hogy a kulcsos hash függvény a függvénycsalád egy példánya.

Egy hash függvénycsalád biztonságának vizsgálatakor legfőképpen azt figyelik, hogy egy támadónak milyen esélye van arra, hogy ütközést találjon a függvénycsalád egy példánya esetén. Aszerint, hogy mikor kerül a támadó birtokába a *key* érték többféle tulajdonságú hash függvény értelmezhető [7].

Ha a *key* érték birtoklása előtt kerülnek kiválasztásra az x, illetve x_1 értékek, akkor univerzális (universal) hash függvény értelmezhető, amelyet a szakirodalom a $CR0$-val jelöl (ahol az x és x_1 értékek jelentése a 3. feladatnak megfelelő). Formálisan az $\mathbf{A}$ támadó előnyét a következő valószínűséggel értelmezhetjük:

$$\begin{aligned}\mathrm{Adv}_{CR0,\mathbf{A}}(H_{key}) = \quad & \Pr[(x, x_1) \xleftarrow{R} \mathbf{A}(\cdot), key \xleftarrow{R} K : \\ & x \neq x_1, H_{key}(x) = H_{key}(x_1)].\end{aligned}$$

3.1. definíció. *Egy $H_{key} : X \to Y$ kulcsos hash függvény univerzális hash függvény, ha bármely probabilisztikus polinomiális idejű* **A** *algoritmus esetén létezik egy $f(k)$ elhanyagolható függvény úgy, hogy $\mathrm{Adv}_{CR0,\mathbf{A}}(H_{key}) \leq f(k)$, ahol az* **A** *algoritmus futási ideje k függvénye.*

Ha az x érték a *key* birtoklása előtt, az x_1 érték pedig a *key* birtoklása után kerül kiválasztásra akkor a cél-ütközésmentes hash (target collision-resistant) függvény értelmezhető, amelyet a szakirodalom $CR1$-gyel jelöl. Formálisan az **A** támadó előnyét a következő valószínűséggel értelmezhetjük:

$$\begin{aligned}\mathrm{Adv}_{CR1,\mathbf{A}}(H_{key}) = \quad & \Pr[(x, st) \xleftarrow{R} \mathbf{A}(\cdot), key \xleftarrow{R} K, x_1 \xleftarrow{R} \mathbf{A}(key, st) : \\ & x \neq x_1, H_{key}(x) = H_{key}(x_1)],\end{aligned}$$

ahol *st* azt az információt jelenti, amelyet a támadó megpróbál megszerezni.

3.2. definíció. *Egy $H_{key} : X \to Y$ kulcsos hash függvény cél-ütközésmentes hash függvény, ha bármely probabilisztikus polinomiális idejű* **A** *algoritmus esetén létezik egy $f(k)$ elhanyagolható függvény úgy, hogy $\mathrm{Adv}_{CR1,\mathbf{A}}(H_{key}) \leq f(k)$, ahol az* **A** *algoritmus futási ideje k függvénye.*

Ha az x és x_1 értékek a *key* birtoklása után kerülnek kiválasztásra, akkor az ütközésmentes hash függvény (collision-resistant) értelmezhető, amelyet a szakirodalom $CR2$-vel jelöl. Formálisan az **A** támadó előnyét a következő valószínűséggel értelmezhetjük:

$$\begin{aligned}\mathrm{Adv}_{CR2,\mathbf{A}}(H_{key}) = \quad & \Pr[key \xleftarrow{R} K, (x, x_1) \xleftarrow{R} \mathbf{A}(key) : \\ & x \neq x_1, H_{key}(x) = H_{key}(x_1)].\end{aligned}$$

3.3. definíció. *Egy $H_{key} : X \to Y$ kulcsos hash függvény ütközésmentes hash függvény, ha bármely probabilisztikus polinomiális idejű* **A** *algoritmus esetén létezik egy $f(k)$ elhanyagolható függvény úgy, hogy $\mathrm{Adv}_{CR2,\mathbf{A}}(H_{key}) \leq f(k)$, ahol az* **A** *algoritmus futási ideje k függvénye.*

Megállapítható, hogy egy CR2 típusú hash függvény egyben CR1 típusú is, egy CR1 típusú, pedig CR0 is egyben, ami alapján fennállnak a következő egyenlőtlenségek:

$$\mathrm{Adv}_{CR0,\mathbf{A}_0}(H_{key}) \leq \mathrm{Adv}_{CR1,\mathbf{A}_1}(H_{key}), \text{ ahol}$$

$\mathbf{A}_0$ tetszőleges támadó algoritmus, amelynek futási ideje ugyanaz mint az $\mathbf{A}_1$ támadó algoritmusnak.

$$\mathrm{Adv}_{CR1,\mathbf{A}_1}(H_{key}) \leq \mathrm{Adv}_{CR2,\mathbf{A}_2}(H_{key}), \text{ ahol}$$

$\mathbf{A}_1$ tetszőleges támadó algoritmus, amelynek futási ideje ugyanaz mint az $\mathbf{A}_2$ támadó algoritmusnak.

A kriptográfiában széles körben alkalmazott SHA hash függvénycsalád példányairól azt feltételezik, hogy CR2 típusú hash függvények.

Ütközésmentes tulajdonsággal rendelkező hash függvényeket norma függvényekből kiindulva Bérczes, Ködmön és Pethő is szerkesztett [16], amelyet a Kripto Kft. Codefish néven hozott kereskedelmi forgalomba. A függvény viselkedését a későbbiekben is tanulmányozták [15] és bizonyították előképellenállóságát, illetve hogy jó statisztikai tulajdonságokkal rendelkezik. Megállapították azt is, hogy a függvény nem rendelkezik a lavinahatás tulajdonsággal a szigorúan vett értelemben, de ezt aszimptotikusan megközelíti.

3.2. Üzenethitelesítő kódok

Az üzenethitelesítő kód (messages authentication code) egy olyan függvény, amely egy titkos információ, a MAC-kulcs alapján, a bemeneti üzenetre (bit-szekvenciára) egy rögzített hosszúságú ellenőrző összeget egy MAC-értéket határoz meg [52]. Az üzenethitelesítő kód egyike a legfontosabb kriptográfiai primitívnek, hiszen a bizalmas információcsere lehetősége mellett az üzenet valódiságát is garantálni kell, amit üzenethitelesítő kódok alkalmazásával lehet elérni. Az üzenethitelesítő kód szerepe tehát, hogy ne lehessen a bemeneti üzenet egyetlen bitjét se megváltoztatni, se kitörölni, se új bitet hozzáadni. A továbbiakban MAC-kódként fogunk hivatkozni az üzenethitelesítő kódra, amelynek formális értelmezése a következő [36]:

3.4. definíció. *Egy MAC-kód három algoritmussal értelmezhető, egy* (K, M, T) *halmazhármas felett, ahol* K *a MAC-kulcsok halmaza,* M *az üzenetek halmaza,* T *a MAC-értékek (a "tag"-ek) halmaza. A három említett algoritmus, pedig a következő:*

- *Gen, a kulcsgeneráló, probabilisztikus algoritmus, amely polinomiális időben az* 1^k *bemeneten meghatározza a MAC-kulcsot, jelöljük a továbbiakban ezt key-el:* $key \xleftarrow{R} Gen(1^k)$*, ahol* $key \in K$*, és* k *a rendszer biztonsági paramétere, a generált kulcs bithossza,*
- Mac_{key}*, az üzenethitelesítő kódot előállító, probabilisztikus algoritmus, a MAC-függvény, amely polinomiális időben a key, MAC-kulcs alapján meghatározza az* $m \in M$ *üzenetre a* t *MAC-értéket:* $t \xleftarrow{R} Mac_{key}(m)$*, ahol* $t \in T$*,*
- Ver_{key}*, a determinisztikus, ellenőrző algoritmus, amely az* key, m, t *bemeneti értékekre leellenőrzi a* t *MAC-érték validitását:* $Ver_{key}(m, t) \rightarrow 1$*, ha az üzenet sértetlen, másképp nullát határoz meg.*

Ha a MAC-függvény bemenetének hossza N bit, akkor a lehetséges bemenetek száma 2^N. Ha a MAC-függvény kimenete egy M-bites érték, akkor ez azt jelenti, hogy 2^M különböző MAC-értéket lehet meghatározni. Mivel N általában sokkal nagyobb, mint M, a lehetséges üzenetek 2^N száma is sokkal nagyobb, mint a lehetséges MAC-értékek 2^M száma. Átlagosan minden MAC-értéket 2^{N-M} különböző bemenet határoz meg.

A MAC-függvény, hasonló egy titkosító függvényhez, azzal a különbséggel, hogy az inverz, a visszafejtő függvényt nem kell értelmezni.

Ezekből a tulajdonságokból kifolyólag a biztonsági előírások egy MAC-függvénnyel szemben mások és nem is olyan szigorúak, mint egy titkosító függvény esetében. A MAC-függvénnyel szemben a következő követelmények állapíthatóak meg [47]:

- A MAC-függvény bemenete egy olyan m üzenet, amelynek hosszára nincs korlát megállapítva, a kimenet hossza, amelyet jelöljünk M-mel, pedig sokkal rövidebb. Biztonsági okokból az $M \geq 128, 256, \ldots$ bit.
- A Kerchoffs-elv alapján a MAC-függvény leírása nyilvános, a rendszer egyetlen titkos információja a *key*.

- Adott MAC-függvény, adott m bemenet és key esetében a $Mac_{key}(m)$ érték polinomiális időben meghatározható kell, hogy legyen.
- Adott MAC-függvény és adott m bemenet esetében sem a $Mac_{key}(m)$ érték, sem a key kulcs nem határozható meg polinomiális időben $1/2^N$-nél nagyobb valószínűséggel; még abban az esetben sem, ha a támadó több $(m_i, Mac_{key}(m_i))$ párral rendelkezik, ahol az m_i bemenetek a támadó által tetszőlegesen szerkesztett üzenetek.

Ahhoz, hogy a kommunikációban résztvevő felek számára egyaránt biztosítva legyen a bizalmas információcsere és az nyílt szöveg sértetlensége, kétféle technika létezik a MAC-érték és a rejtjelezett szöveg együttes meghatározására.

Az első technika szerint a nyílt szöveg után hozzáfűzzük a nyílt szövegre kiszámolt MAC-értéket és ezen új értékre alkalmazunk egy Enc titkosító függvényt.

A második technika szerint előbb alkalmazzuk az Enc titkosító függvényt a nyílt szövegen és az így kapott rejtjelezett szövegre számoljuk ki a MAC-értéket, majd ezt a két értéket egymás után fűzzük.

Mindkét esetben a titkosító függvény és a MAC-függvény kulcsa különböző kell legyen.

A két technika különböző támadási lehetőségeket határoz meg.

A MAC-érték kiszámítása során az üzenetet blokkokra osztják, amelyeket standard előírásoknak megfelelően összeláncolnak [7]. A támadási lehetőségeket az üzenet blokkjaira alkalmazott láncolási technika is meghatározza.

A MAC-függvények elleni támadási lehetőségeket a szakirodalom [60] két csoportra osztja: *kriptoanalízisen* alapuló támadások és *brute-force* típusú támadások.

A *kriptoanalízis* során a támadó a MAC-függvények szerkesztésénél alkalmazott technikák gyengeségeit veszi támadásba.

A *brute-force* típusú támadások esetében a támadó vagy a MAC-kulcsot próbálja meghatározni vagy egy érvényes MAC-értéket akar előállítani a bemeneti üzenet ismeretének hiányában. A támadó keresési technikát alkalmaz az összes lehetséges bemeneti, kimenetei értékpárok között.

Abban az esetben, ha a MAC-kulcs meghatározása a cél, akkor a támadó a következőképpen jár el. Ha feltételezzük, hogy a MAC-kulcs hossza $|key|$ bit és a támadó rendelkezik egy üzenet és a neki megfelelő MAC-érték párral, akkor a támadás abból áll, hogy a támadó meghatározza az összes lehetséges kulcs értékekre a megfelelő MAC-értéket és összehasonlítja, hogy melyik talál az eredeti MAC-értékkel. Ez azt jelenti, hogy minden kulcsértékre meg kell határozza a MAC-értéket, azaz $2^{|key|}$ MAC-értéket kell előállítson. A MAC-érték szerkesztési módjából kifolyólag fennáll az a lehetőség, hogy több kulcsértékre ugyanazt a MAC-értéket kapja, ebben az esetben egy új (üzenet, MAC-érték) párra kell elvégezni a számításokat.

Abban az esetben, ha egy érvényes MAC-értéket akar előállítani, akkor a MAC-érték nagyságrendje határozza meg a brute-force támadás kivitelezhetőségét. Ha a MAC-érték bithossza M, akkor az előállítható MAC-értékek száma 2^M.

Figyelembe véve a számítógépek jelenlegi számítási kapacitását, a minimális biztonsági követelmény egy MAC-kóddal szemben az, hogy fennálljon: $min(|key|, M) \geq 128$ bit.

A MAC-kódok biztonságának formális értelmezése is megadható [36] amelyről azonban fontos megjegyezni, hogy erős megkötést tartalmaz, melyet nem minden gyakorlatban használt MAC-kód teljesít. Az értelmezéshez szükséges megadni egy kísérlet lépéssorozatát, ahol a kísérlet az **A** támadó és környezete, a kihívó között játszódik le.

3.1. kísérlet.

1. *A kihívó a Gen kulcsgeneráló algoritmussal meghatározza az key MAC-kulcsot,*
2. *az* **A** *támadó tetszőleges számú MAC-értéket határoz meg a kihívó* $Mac_{key}(\cdot)$ *algoritmusa segítségével; jelöljük a meghatározott MAC-értékek halmazát* $\mathbb{M}$*-el,*
3. *az* **A** *támadó meghatároz egy* (m, t) *értékpárt, amelyet átad a kihívónak,*
4. *ha* $Ver_{key}(m, t) = 1$ *és* $m \notin \mathbb{M}$*, akkor az* **A** *támadó megadja a* $b = 1$ *értéket, másképp egy* $b \neq 1$ *értéket határoz meg.*

3.5. definíció. *Az* $Adv_{MAC,\mathbf{A}}(k) = Pr[b = 1]$ *valószínűséggel az* **A** *támadó előnyét értelmezzük, ahol a b értéket a 3.1. kísérletben határoztuk meg.*

3.6. definíció. *A MAC-kód biztonságos, ha bármely* **A** *probabilisztikus, polinomiális idejű támadó esetében létezik egy* $f(k)$ *elhanyagolható függvény úgy hogy:*

$$Adv_{MAC,\mathbf{A}}(k) \leq f(k).$$

A gyakorlatban két fajta MAC-függvényt szoktak alkalmazni, az egyik hash függvényeken alapuló MAC-függvény, a HMAC [4], a másik a CBC módban alkalmazott blokktitkosító rendszereken alapuló MAC-függvény [60].

4. fejezet

Aszimmetrikus titkosítási rendszerek biztonsága

4.1. A matematikai modell

Aszimmetrikus vagy nyilvános kulcsú titkosítási rendszereket az 1970-es évek közepe óta alkalmaznak a kriptográfiában, pontosabban Diffie és Hellman 1976-os cikke óta [25]. A legtöbb ma is széles körben alkalmazott rendszer matematikai alapjait ekkor fektették le, viszont a létező rendszerekkel szembeni biztonsági elvárások azóta igencsak megváltoztak [38]. Ezért számos algoritmus, protokoll implementálása az első specifikációk óta, jelenleg más módon történik. Például az RSA esetében a jelenlegi standardok az RSA–OAEP alapján adják meg a titkosító algoritmus specifikációját [6]. Az ElGamal-titkosítónak a Cramer és Shoup által ajánlott rendszer jelenti a biztonságos verzióját [21], a Rabin-titkosító egyik biztonságos verzióját, pedig Boneh írta le [14]. A többszörös titkosítás (multiple encryption) egyik biztonságos használatának módját Dodis és Katz adták meg [26], amelyre a korábbi szakirodalomban többször találunk nem biztonságos ajánlásokat is.

Hogy megadhassuk a jelenlegi biztonsági elvárásoknak is megfelelő biztonsághoz kapcsolódó értelmezéseket először az aszimmetrikus titkosítási rendszerek matematikai értelmezését adjuk meg.

4.1. definíció. *Jelölje PKE azt az aszimmetrikus titkosítási vagy nyilvános kulcsú titkosítási rendszert, amely három algoritmussal értelmezhető, a $((PK, SK), M, C)$ felett, ahol M a nyílt szövegek egy véges halmaza, C a rejtjelezett szövegek egy véges halmaza, PK a nyilvános kulcsok egy véges halmaza, és SK a titkos kulcsok véges halmaza. A három említett algoritmus, pedig a következő:*

- *Gen, a probabilisztikus kulcsgeneráló algoritmus, amely polinomiális időben, az 1^k bemeneten meghatározza a (pk, sk) kulcspárt: $(pk, sk) \xleftarrow{R} Gen(1^k)$, ahol $pk \in PK$ a nyilvános kulcs és $sk \in SK$ a titkos kulcs és $k \in \mathbb{Z}_{\geq 0}$ a rendszer biztonsági paramétere, amely legtöbb esetben a generált kulcs bithossza,*
- *az Enc_{pk} a probabilisztikus rejtjelező algoritmus, amely polinomiális időben végzi a rejtjelezést, az $m \in M$ bemenetre; a pk nyilvános kulcs ismeretében meghatározza a $c \in C$ rejtjelezett szöveget: $c \xleftarrow{R} Enc_{pk}(m)$,*
- *a Dec_{sk} a visszafejtő algoritmus, amely polinomiális időben, determinisztikusan végzi a visszafejtést, az sk titkos kulcs alapján a $c \in C$ rejtjelezett szövegen: $m \leftarrow Dec_{sk}(c)$; hiba esetében, kimenetként visszautasító üzenetet ad.*

A rendszer helyessége érdekében megköveteljük, hogy adott (pk, sk) kulcspár és m nyílt szövegre fennálljon, hogy $Dec_{sk}(Enc_{pk}(m)) = m$. A rendszer helyességét illetően, a jelenlegi szakirodalom [21] ennél sokkal árnyaltabban fogalmaz, mégpedig a következő követelményeket írja elő meg:

- adott $m \in M$ nyílt szöveg és $c \in C$ rejtjelezett szöveg esetében a $(pk, sk) \in PK \times SK$ kulcspár nem megfelelő, ha fennáll: $Dec_{sk}(Enc_{pk}(m)) \neq m$,
- annak a valószínűsége, hogy a *Gen* kulcsgeneráló algoritmus nem megfelelő (pk, sk) kulcspárt generál a k bemenet függvényében elhanyagolhatóan nő.

4.2. Egyirányú függvények

Aszimmetrikus rendszerek szerkesztéséhez egyirányú függvényekre (one-way function) van szükség. Az egyirányú függvények olyan függvények, amelyekre fennáll, hogy a függvény bármely bemeneti értékére hatékony kiértékelési algoritmus adható meg, másfelől az inverz elem meghatározására nem ismert hatékony algoritmus.

Formálisan a következő értelmezés adható meg [18]:

4.2. definíció. *A $g : X \to Y$ függvény egyirányú, ha fennáll a következő két tulajdonság:*

1. *(hatékony kiértékelés) létezik egy polinomiális idejű algoritmus, amely az $x \in X$ bemenetre meghatározza a $g(x) \in Y$ értéket,*
2. *(nehéz az inverz elem meghatározása) bármely probabilisztikus polinomiális idejű* **A** *algoritmus esetében létezik egy $f(k)$ elhanyagolható függvény úgy, hogy:*

$$\Pr[g(\mathbf{A}(g(x), 1^k)) = g(x)] \leq f(k), \text{ ahol } x \overset{R}{\leftarrow} X.$$

A gyakorlatban egyirányú függvények csak bizonyos feltételezések mellett léteznek. Ahhoz, hogy feltételezések nélküli egyirányú függvények létezéséről beszélhessünk szükséges lenne a híres $NP \neq P$ állítás bizonyítása ([39], [56]). Az említett feltételezések az angol kriptográfiai terminológiában basic assumption néven ismertek.

Abban az esetben, ha a g függvény bijektív, és az X értelmezési tartomány megegyezik az Y érték tartománnyal, azt mondjuk, hogy a g egyirányú permutáció.

Kriptorendszerek szerkesztéséhez, azonban nem mindig elégségesek az egyirányú függvények, szükséges az egyirányú csapóajtó függvény (one-way trapdoor function) fogalmának értelmezése.

Az egyirányú csapóajtó függvények olyan függvények, amelyekre igaz, hogy a függvény bármely bemeneti elemére hatékony kiértékelési algoritmus adható meg, másfelől az inverz elem meghatározására nem ismert hatékony algoritmus, csak abban az esetben ha adott egy plusz információ is.

Formálisan a következő értelmezés adható meg [18]:

4.3. definíció. *A $g : X \to Y$ függvény egyirányú csapóajtó függvény, ha fennállnak a következő tulajdonságok:*

1. *létezik egy Gen algoritmus, amely polinomiális időben, az 1^k bemeneten meghatározza a td plusz információt,*
2. *létezik egy polinomiális idejű algoritmus, amely az $x \in X$ bemenetre meghatározza a $g(x) \in Y$ értéket,*
3. *bármely probabilisztikus polinomiális idejű $\mathbf{A}$ algoritmus esetében létezik egy $f(k)$ elhanyagolható függvény úgy, hogy:*

$$\Pr[g(\mathbf{A}(g(x), 1^k)) = g(x)] \leq f(k), \text{ ahol } x \xleftarrow{R} X,$$

4. *létezik egy determinisztikus polinomiális idejű $\mathbf{B}$ algoritmus, amely esetében fennáll:*

$$g(\mathbf{B}(g(x), td, 1^k)) = g(x), \text{ ahol } x \xleftarrow{R} X.$$

4.3. Kriptográfiai feltételezések

A következőkben megadjuk azokat a feltételezéseket, amelyeket az értekezés során tanulmányozott rendszerek esetében használunk ([36], [18]). Ezen feltételezések alapján kijelenthető, hogy létezik egyirányú függvény.

4.3.1. A faktorizációs feltételezés

A faktorizációs feltételezés (factoring assumption) a következőket állítja: legyen Gen_{fact} egy algoritmus, amely polinomiális időben, az 1^k bemeneten meghatározza az (n, p, q) számhármast: $(n, p, q) \xleftarrow{R} Gen_{fact}(1^k)$ úgy, hogy p, q k-bites prímszámok és $n = p \cdot q$.

4.4. definíció. *A faktorizációs feltételezés, a Gen_{fact} algoritmus által generált értékek függvényében, azt jelenti, hogy bármely probabilisztikus, polinomiális idejű $\mathbf{A}$ algoritmus esetében, létezik egy $f(k)$ elhanyagolható függvény úgy, hogy*

$$\Pr[\mathbf{A}(n) = (p, q)] \leq f(k).$$

4.3.2. Az RSA-feltételezés

Az RSA-feltételezés (RSA assumption) a következőket állítja: legyen Gen_{RSA} egy algoritmus, amely polinomiális időben, az 1^k bemeneten meghatározza az (n, p, q, e, d) számötöst: $(n, p, q, e, d) \xleftarrow{R} Gen_{RSA}(1^k)$ úgy, hogy:

- $n = p \cdot q$, és p, q k-bites prímszámok,
- $e \xleftarrow{R} \mathbb{Z}_n^*$, és d-re fennáll: $e \cdot d \equiv 1 \pmod{(p-1)\cdot(q-1)}$.

4.5. definíció. *Az RSA-feltételezés, a Gen_{RSA} algoritmus által generált értékek függvényében, azt jelenti, hogy bármely probabilisztikus, polinomiális idejű* **A** *algoritmus esetében, létezik egy $f(k)$ elhanyagolható függvény úgy, hogy*

$$\Pr[\mathbf{A}(n, e) = d] \leq f(k).$$

4.3.3. A kvadratikus maradék feltételezés

A kvadratikus maradék feltételezés (quadratic residuosity assumption) a következőket állítja: legyen $Gen_{\mathbb{QR}}$ egy algoritmus, amely polinomiális időben, az 1^k bemeneten meghatározza az (n, p, q) számhármast: $(n, p, q) \xleftarrow{R} Gen_{\mathbb{QR}}(1^k)$ úgy, hogy p, q k-bites prímszámok és $n = p \cdot q$.

Jelöljük $\mathbb{QR}_n$-el a kvadratikus maradékok halmazát $\pmod n$ szerint, formálisan:

$$\mathbb{QR}_n = \{x \in \mathbb{Z}_n^* | \exists z \in \mathbb{Z}_n^* : z^2 = x \pmod n\}.$$

Jelöljük $\mathbb{QNR}_n^{+1}$-el azon számok halmazát, amelyek nem kvadratikus maradékok $\pmod p$ és $\pmod q$ szerint sem, ahol formálisan:

$$\mathbb{QNR}_n^{+1} = \{x \in \mathbb{Z}_n^* | \neg\exists z, u \in \mathbb{Z}_n^* : z^2 = x \pmod p \text{ és } u^2 = x \pmod q\}.$$

4.6. definíció. *Annak az eldöntése, hogy egy szám kvadratikus maradék vagy sem, a $Gen_{\mathbb{QR}}$ algoritmus által generált értékek függvényében, azt jelenti, hogy bármely probabilisztikus, polinomiális idejű* **A** *algoritmus esetében, létezik egy $f(k)$ elhanyagolható függvény úgy, hogy*

$$|\Pr[\mathbf{A}(n, x_{qr}) = 1] - \Pr[\mathbf{A}(n, x_{qnr}) = 1]| \leq f(k), \text{ ahol}$$

$$x_{qr} \xleftarrow{R} \mathbb{QR}_n \text{ és } x_{qnr} \xleftarrow{R} \mathbb{QNR}_n^{+1}.$$

4.3.4. A diszkrét logaritmus feltételezés

A diszkrét logaritmus feltételezés (discrete logarithm assumption, DL assumption) a következőket állítja: legyen Gen_{DL} egy algoritmus, amely polinomiális időben, az 1^k bemeneten meghatározza az $(\mathbb{G}, p, g)$ elemeket $(\mathbb{G}, p, g) \xleftarrow{R} Gen_{DL}(1^k)$ úgy, hogy:

- $\mathbb{G}$ egy ciklikus csoport, amelynek rendje egy k-bites szám: p,
- g a $\mathbb{G}$ csoport generátor eleme.

4.7. definíció. *Az diszkrét logaritmus feltételezés, a Gen_{DL} algoritmus által generált értékek függvényében, azt jelenti, hogy bármely probabilisztikus, polinomiális idejű* **A** *algoritmus esetében, létezik egy $f(k)$ elhanyagolható függvény úgy, hogy*

$$\Pr[\mathbf{A}(\mathbb{G}, p, g, h) = x] \leq f(k), \text{ ahol fennáll, hogy } g^x = h.$$

4.3.5. A döntési Diffie–Hellman-feltételezés

A döntési Diffie–Hellman-feltételezés (Decisional Diffie-Hellman assumption, DDH assumption) a következőket állítja: legyen Gen_{DDH} egy algoritmus, amely polinomiális időben, az 1^k bemeneten meghatározza az $(\mathbb{G}, p, g)$ elemeket $(\mathbb{G}, p, g) \xleftarrow{R} Gen_{DDH}(1^k)$ úgy, hogy:

- $\mathbb{G}$ egy ciklikus csoport, amelynek rendje egy k-bites szám: p,
- g a $\mathbb{G}$ csoport generátor eleme.

4.8. definíció. *A döntési Diffie–Hellman-feltételezés, a Gen_{DDH} algoritmus által generált értékek függvényében, azt jelenti, hogy bármely probabilisztikus polinomiális idejű* **A** *algoritmus esetében, létezik egy $f(k)$ elhanyagolható függvény úgy, hogy*

$$|\Pr[\mathbf{A}(\mathbb{G}, p, g, g^x, g^y, g^z) = 1] - \Pr[\mathbf{A}(\mathbb{G}, p, g, g^x, g^y, g^{xy}) = 1]| \leq f(k), \text{ ahol}$$

$$x, y, z \xleftarrow{R} \mathbb{Z}_p.$$

4.4. Biztonságértelmezések

Az aszimmetrikus titkosítók esetében a biztonság értelmezésekor figyelembe kell venni, hogy a támadó rendelkezik a nyilvános kulccsal, ezért bármikor, tetszőleges számú titkosítást tud végrehajtani, amelynek eredményét megvizsgálva sikeres támadást kezdeményezhet.

A támadás során, azon plusz információ, amely a nyilvános kulcs ismeretét jelenti, lényegesen megváltoztatja az aszimmetrikus titkosítás esetében a biztonsági kritériumokat [43]. Ez a plusz információ azt is jelenti, hogy az aszimmetrikus rendszereknél a két legfontosabb támadási típus amelyet meg kell vizsgálni a választott nyílt szöveg alapú támadás és a választott rejtjelezett szöveg alapú támadás. Éppen ezért, jelen értekezésben nem térünk ki a passzív támadóval szembeni biztonsági kritériumokra. Mindkét támadási típus esetében megfogalmazhatóak az általános biztonsági követelmények, megadhatóak a biztonságértelmezések, amelyek azonban sokszor nagyon erős megkötéseket tartalmaznak, így egy adott helyzetben a kriptográfiai rendszer tervezésekor dől el, hogy hogyan érdemes betartani ezeket a biztonsági kritériumokat.

4.4.1. Választott nyílt szöveg alapú támadás

A választott nyílt szöveg alapú támadás esetében feltételezzük, hogy a támadó a titkosító algoritmus segítségével meg tudja határozni egy tetszőleges nyílt szöveg rejtjelezett értékét, ahol a kiválasztásra kerülő nyílt szövegeket a támadó határozza meg. A támadás adaptív formája azt jelenti, hogy a támadó által aszerint kerülnek kiválasztásra a nyílt szövegek, hogy mi volt egy korábbi titkosítás eredménye. Ahogy a szimmetrikus titkosításnál, úgy itt is a biztonságértelmezés megadása egy kísérlet kimeneti eredményének a vizsgálatával történik [21].

A szükséges lépéssorozat, a megfelelő kísérlet esetében a következő, ahol a kísérlet az **A** támadó és környezete, a kihívó között játszódik le.

4.1. kísérlet.

1. *A kihívó a Gen kulcsgeneráló algoritmussal meghatározza a* (pk, sk) *kulcspárt, majd a pk nyilvános kulcsot átadja az* **A** *támadónak,*

2. *az* **A** *támadó tetszőleges számú titkosítást végez a pk nyilvános kulcs ismeretében, ahol a titkosítást a kihívó* $Enc_{pk}(\cdot)$ *algoritmusával végzi,*
3. *az* **A** *támadó, két azonos hosszúságú üzenetet határoz meg, jelöljük ezeket* m_0, m_1*-gyel, amelyeket átad a kihívónak,*
4. *a kihívó* $b \xleftarrow{R} \{0,1\}$ *választás alapján az* $Enc_{pk}(\cdot)$ *algoritmussal meghatározza az* m_b *titkosított értékét:* $c^* \xleftarrow{R} Enc_{pk}(m_b)$*, amelyet átad az* **A** *támadónak; nevezzük ezt a* c^* *titkosított értéket kihívó rejtjelezett szövegnek (challenge ciphertext),*
5. *az* **A** *támadó tetszőleges számú titkosítást végez a pk nyilvános kulcs ismeretében, ahol a titkosítást a kihívó* $Enc_{pk}(\cdot)$ *algoritmusával végzi,*
6. *az* **A** *támadó meghatároz egy* $\hat{b} \in \{0,1\}$ *kimeneti értéket.*

4.9. definíció. *A PKE aszimmetrikus titkosítási rendszerben a* **A** *támadó előnye a következőképpen adható meg:* $\mathrm{Adv}^{cpa}_{PKE,A}(k) = |\Pr[b = \hat{b}] - \frac{1}{2}|$*, ahol a* b *és* $\hat{b}$ *értékeket a 4.1. kísérletben határoztuk meg.*

A választott nyílt szöveg alapú támadással szembeni CPA-biztonságot a következő értelmezés adja:

4.10. definíció. *A PKE aszimmetrikus titkosítási rendszer CPA-biztonságú, ha bármely* **A***, probabilisztikus, polinomiális idejű támadó esetében létezik egy* $f(k)$ *elhanyagolható függvény úgy, hogy*

$$\mathrm{Adv}^{cpa}_{PKE,A}(k) \leq f(k).$$

4.4.2. Választott rejtjelezett szöveg alapú támadás

Egy sokkal erősebb biztonságértelmezés a választott rejtjelezett szöveg alapú támadással szembeni biztonságot értelmezi, amely során a támadó tetszőleges számú rejtjelezést és visszafejtést is el tud végezni egy rejtjelező, illetve visszafejtő orákulum segítségével ([21], [11]). A támadás adaptív formája azt jelenti, hogy a támadónak arra is lehetősége van, hogy a kiválasztásra kerülő nyílt szövegeket, illetve rejtjelezett szövegeket az alapján határozza meg, hogy mi volt a korábbi rejtjelezés, illetve visszafejtés eredménye.

A választott rejtjelezett szöveg alapú támadással szembeni biztonságértelmezéséhez, a megfelelő kísérlet, a következő lépéssorozatból fog állni, ahol a kísérlet az **A** támadó és környezete, a kihívó között játszódik le.

4.2. kísérlet.

1. *A kihívó a Gen kulcsgeneráló algoritmussal meghatározza a* (pk, sk) *kulcspárt, majd a pk nyilvános kulcsot átadja az* **A** *támadónak,*
2. *az* **A** *támadó tetszőleges számú titkosítást, illetve visszafejtést végez a kihívó* $Enc_{pk}(\cdot)$ *titkosító és* $Dec_{sk}(\cdot)$ *visszafejtő algoritmusával,*
3. *az* **A** *támadó, két azonos hosszúságú üzenetet határoz meg, jelöljük ezeket* m_0, m_1*-gyel, amelyeket átad a kihívónak,*
4. *a kihívó* $b \xleftarrow{R} \{0,1\}$ *választás alapján az* $Enc_{pk}(\cdot)$ *titkosító algoritmussal meghatározza az* m_b *titkosított értékét:* $c^* \xleftarrow{R} Enc_{pk}(m_b)$*, amelyet átad az* **A** *támadónak; nevezzük ezt a* c^* *titkosított értéket kihívó rejtjelezett szövegnek (challenge ciphertext),*
5. *az* **A** *támadó tetszőleges számú titkosítást, illetve visszafejtést végez a kihívó* $Enc_{pk}(\cdot)$ *titkosító és* $Dec_{sk}(\cdot)$ *visszafejtő algoritmusával, különböző c bemenetekre, azzal a megkötéssel, hogy* $c \neq c^*$*,*
6. *az* **A** *támadó meghatároz egy* $\hat{b} \in \{0,1\}$ *kimeneti értéket.*

4.11. definíció. *A PKE aszimmetrikus titkosítási rendszerben az* **A** *támadó előnye a következőképpen adható meg:* $\mathrm{Adv}^{cca}_{PKE,\mathbf{A}}(k) = |\Pr[b = \hat{b}] - \frac{1}{2}|$*, ahol a* b *és* $\hat{b}$ *értékeket a 4.2. kísérletben határoztuk meg.*

A választott rejtjelezett szöveg alapú támadással szembeni CCA-biztonságot a következő értelmezés adja:

4.12. definíció. *A PKE aszimmetrikus titkosítási rendszer CCA-biztonságú, ha bármely* **A***, probabilisztikus, polinomiális idejű támadó esetében létezik egy* $f(k)$ *elhanyagolható függvény úgy, hogy*

$$\mathrm{Adv}^{cca}_{PKE,A}(k) \leq f(k).$$

A fenti két biztonságértelmezés alapján a következő következtetéseket vonhatjuk le [32]:

- Azok az aszimmetrikus titkosító rendszerek amelyeknél a titkosító $Enc_{pk}(\cdot)$ algoritmus determinisztikus nem biztosítják se a CPA-biztonságot, se a CCA-biztonságot,
- nem lehet olyan aszimmetrikus titkosító rendszert szerkeszteni, amely tökéletes biztonságú.

4.5. Hibrid rendszerek

A bizalmas információcsere megvalósítása érdekében, a gyakorlatban gyakran alkalmaznak hibrid rendszereket. Ez elsősorban azt jelenti, hogy a tulajdonképpeni információ titkosításához szimmetrikus titkosítási technikát, a titkosításhoz szükséges kulcs, a titkosító kulcs cseréjéhez, pedig aszimmetrikus titkosítási technikát használnak. Ennek szükségessége azért fontos, mert nagy adathalmaz titkosításához nem vehetők igénybe a nagy számítási kapacitást igénylő aszimmetrikus titkosító rendszerek, viszont a szimmetrikus titkosítási technikák nem biztosítják hatékonyan a kulcscserét.

Az aszimmetrikus titkosító rendszerek megoldják a szimmetrikus titkosító rendszerek problémáját: lehetőséget adnak hogy két, egymással, egy nyilvános csatornán keresztül kommunikáló egység kicserélje a titkosításhoz szükséges titkosító kulcsot úgy, hogy az ne kerülhessen illetéktelen egység birtokába.

Titkosítási rendszerek esetében fontos különbséget tenni a következő két eset között:

- megengedett a titkosításhoz generált kulcs többszöri felhasználása,
- nem megengedett a titkosításhoz generált kulcs többszöri felhasználása, azaz minden egyes titkosításhoz más és más kulcsot kell generálni.

A fenti két eset más biztonsági kritériumokat határoz a szimmetrikus, illetve az aszimmetrikus titkosító rendszerek esetében, éppen ezért fontosnak tartjuk a következőket megjegyezni [36]:

4.1. megjegyzés. *Egy determinisztikus, szimmetrikus titkosító rendszer nem tudja biztosítani a passzív támadóval szembeni biztonságot, ha megengedett a kulcs többszöri felhasználása.*

4.2. megjegyzés. *Egy szimmetrikus titkosító rendszer, amely biztonságos egy passzív támadóval szemben, elveszíti passzív támadóval szembeni biztonságát, ha megengedjük a kulcs többszöri felhasználását.*

4.3. megjegyzés. *Egy szimmetrikus titkosító rendszer, amely CPA-biztonságú megőrzi CPA-biztonságát, ha megengedjük a kulcs többszöri felhasználását.*

4.4. megjegyzés. *Egy aszimmetrikus titkosító rendszer, ha biztonságos egy passzív támadóval szemben, akkor CPA-biztonságú is lesz, mivel a támadónak mindig lehetősége van tetszőleges számú titkosítást végezni, hiszen a titkosító kulcs, azaz a nyilvános kulcs publikus. Ez természetesen nem így van a szimmetrikus titkosító rendszerek esetében.*

4.5. megjegyzés. *Egy aszimmetrikus titkosító rendszer, amely CPA-biztonságú megőrzi CPA-biztonságát, ha megengedjük a nyilvános kulcsok többszöri felhasználását.*

4.6. megjegyzés. *Egy determinisztikus, aszimmetrikus titkosító rendszer nem lehet CPA-biztonságú.*

4.13. definíció. *Azokat a rendszereket, ahol minden egyes titkosításhoz más kulcsot generálnak egyszeri titkosító (one-time encryption) rendszereknek hívják.*

Ugyanez alkalmazható a *Mac*-kódokra is, azaz:

4.14. definíció. *Azokat a Mac-kódokat, ahol minden Mac-érték előállításához más kulcsot generálnak egyszeri Mac-kódoknak (one-time message authentication code) hívják.*

Hibrid rendszerek, precíz matematikai definícióját, és CCA-biztonságú hibrid rendszerek szerkesztési módját, többek között Cramer és Shoup is megadták [21]. Az általuk ismertetett eljárás aszimmetrikus titkosítási technikával biztosítja, a kommunikálni kívánó felek között, egy

véletlenszerűen generált érték, a titkosító kulcs megosztását amely kulcs felhasználásával, szimmetrikus titkosítást alkalmazva, megoldható két egység között a bizalmas információcsere. Cikkükben bevezették a kulcsbeágyazási mechanizmus (key encapsulation mechanism) fogalmát, amely a titkosító kulcs generálását és megosztását biztosítja.

Bebizonyították, hogy egy CCA-biztonságú aszimmetrikus titkosítási rendszert kombinálva egy CCA-biztonságú egyszeri szimmetrikus titkosítási rendszerrel CCA-biztonságú hibrid rendszert eredményez.

CCA-biztonságú szimmetrikus rendszerek szerkesztésénél szükség van egyszeri szimmetrikus titkosító rendszerekre, illetve egyszeri Mac-kódokra.

CCA-biztonságú aszimmetrikus rendszerek szerkesztésénél szükség van egyirányú csapóajtó függvényekre és hash függvényekre.

Jelen értekezésben, ahogyan az újabb szakirodalom is teszi, a hibrid rendszereken belül az aszimmetrikus titkosítás illetve visszafejtés megnevezések helyett a kulcsbeágyazási algoritmus (key encapsulation algorithm), illetve kulcskibontási algoritmus (key decapsulation algorithm) neveket fogjuk használni.

Ezek után formálisan is megadható egy hibrid titkosító rendszer matematikai modellje:

4.15. definíció. *Jelölje HKE azt a hibrid titkosítási rendszert, amely három algoritmussal értelmezhető, a $((PK, SK), M, C)$ felett, ahol*

- *Gen, a probabilisztikus kulcsgeneráló algoritmus, amely polinomiális időben, az 1^k bemeneten meghatározza (pk, sk) kulcspárt: $(pk, sk) \xleftarrow{R} Gen(1^k)$, ahol $pk \in PK$ a nyilvános kulcs és $sk \in SK$ a titkos kulcs és $k \in \mathbb{Z}_{\geq 0}$ a rendszer biztonsági paramétere,*
- *Az $Enc_{HKE}(pk, m)$ titkosító a következő két algoritmusból áll:*
  - *az Enc_{KEM} egy CCA-biztonságú kulcsbeágyazási algoritmus, amely polinomiális időben, probabilisztikusan, a pk bemeneten előállít egy értéket, a titkosító kulcsot és annak egy titkosított értékét: $(key, cipher) \leftarrow Enc_{KEM}(pk)$,*
  - *az Enc_{SKE} egy CCA-biztonságú egyszeri szimmetrikus titkosítási rendszer, amely a key és m bemenetre meghatározza az m titkosított értékét: $c \leftarrow Enc_{SKE}(key, m)$.*

  - *Az* $Enc_{HKE}(pk, m)$ *algoritmus kimenete:* $(c, cipher)$.
- *A* $Dec_{HKE}(sk, c, cipher)$ *visszafejtő algoritmus a következőket végzi:*
  - *ha a* $(c, cipher)$ *nem egy helyes titkosított érték, akkor az algoritmus REJECT, elutasító kimeneti értékkel leáll,*
  - *ellenkező esetben meghívja a* Dec_{KEM}*, determinisztikus, kulcskibontási algoritmust:*
    - *ha a* Dec_{KEM} *REJECT, elutasító kimeneti értékkel leáll, akkor a visszafejtő* Dec_{HKE} *algoritmus is leáll REJECT, elutasító kimeneti értékkel,*
    - *ellenkező esetben az* $sk, cipher$ *bemenetre meghatározza a* key *értékét:* $key \leftarrow Dec_{KEM}(sk, cipher)$,
  - *meghívja a determinisztikus, egyszeri szimmetrikus titkosító algoritmust, amely a* key, c *bemenetre meghatározza az* m *értékét:* $m \leftarrow Dec_{SKE}(key, c)$,
  - $Dec_{HKE}(sk, c, cipher)$ *algoritmus kimenete:* m.

A rendszer helyessége érdekében megköveteljük, hogy mind a kulcsbeágyazási mechanizmus, mind a szimmetrikus titkosító rendszer helyes legyen. A szimmetrikus titkosító rendszer helyességét az 2.9. definíciónál adtuk meg, a kulcsbeágyazási mechanizmus helyessége, pedig az aszimmetrikus titkosítók matematikai modelljénél leírt helyesség alapján adható meg.

Egy kulcsbeágyazási mechanizmus esetében, tehát a rendszer helyességét a következő követelmények írják le:

- a $(pk, sk) \in PK \times SK$ kulcspár nem megfelelő, ha bármely $(key, cipher) \overset{R}{\leftarrow} Enc_{KEM}(pk)$ esetében fennáll:

$$Dec_{KEM}(sk, cipher) \neq key$$

- annak a valószínűsége, hogy a Gen kulcsgeneráló algoritmus nem megfelelő (pk, sk) kulcspárt generál, a k bemenet függvényében, elhanyagolhatóan nő.

Ezek után megadható a hibrid rendszerek formális CCA-biztonság értelmezése [21]:

4.1. tétel. *A 4.11. definícióval megadott HKE hibrid rendszer CCA-biztonságú, ha az* Enc_{SKE}, Dec_{SKE} *algoritmusok által definiálható szimmetrikus titkosítási rendszer CCA-biztonságú, és ha a* Gen, Enc_{KEM}, Dec_{KEM} *algoritmusok által definiálható kulcsbeágyazási mechanizmus, azaz aszimmetrikus titkosítási rendszer is CCA-biztonságú.*

4.7. megjegyzés. *Egy CCA-biztonságú szimmetrikus titkosítási rendszer úgy szerkeszthető, ha egy passzív támadóval szemben biztonságos szimmetrikus titkosító rendszerhez hozzácsatolunk egy egyszeri üzenethitelesítő kódot.*

Fontos megjegyezni, azt is hogy a szimmetrikus titkosításhoz használt kulcs és az üzenethitelesítő kód létrehozásához szükséges kulcs különböző kell, hogy legyen.

A CCA-biztonságú kulcsbeágyazási mechanizmusok szerkesztéséhez hash függvényeket alkalmaznak. Aszerint, hogy az alkalmazásra kerülő hash függvény viselkedése milyen feltételezések segítségével van definiálva megkülönböztetjük a véletlen orákulum modellt és a standard modellt.

A véletlen orákulum (random oracle) modellben olyan hash függvény alkalmazásával bizonyítható a CCA-biztonság, amelyről feltételezik, hogy teljesen véletlenszerűen viselkedik. Ebben a modellben nem követelik meg a hash függvénytől, hogy a 3.1. alfejezetben ismertetett tulajdonságoknak eleget tegyen. Éppen ezért a véletlen orákulum modellben bizonyított CCA-biztonságú titkosító rendszerek heurisztikus bizonyítások, és sokszor nem bizonyulnak biztonságosnak. Hatékonyság szempontjából viszont nagyon előnyös az alkalmazásuk.

A standard modellben bebizonyított CCA-biztonság esetén a hash függvénytől azt kell elvárni, hogy a függvény a 3.1. alfejezetben ismertetett tulajdonságoknak tegyen eleget. Ezek a bizonyítások megbízható biztonságot garantálnak csupán a standard feltételezéseket, mint pl. ütközésmentesség, faktorizációs feltételezés, diszkrét logaritmus feltételezés stb. elfogadva. Éppen ezért a standard modellben elért CCA-biztonságú rendszerek sokkal erősebb biztonságot garantálnak, mint a véletlen orákulum modellben bizonyított CCA-biztonságú rendszerek. Hatékonyság szempontjából viszont kevésbé előnyös az alkalmazásuk.

5. fejezet

CCA-biztonságú kulcsbeágyazási mechanizmusok

Ebben a fejezetben azokat az aszimmetrikus titkosító rendszereket ismertetjük, amelyek alapul szolgáltak több, napjainkban használatos CCA-biztonságú kulcsbeágyazási mechanizmusnak. Fontosnak tartjuk megjegyezni, hogy az értekezés 7. fejezetében ezen rendszerek mindegyikének megvizsgáltuk implementációs lehetőségét, összevetve őket hatékonyság szempontjából a 6. fejezetben bemutatásra kerülő új kulcsbeágyazási mechanizmus hatékonyságával. Mindegyik rendszernél bemutatjuk a szükséges algoritmusokat, és azt a sikeres támadási lépéssorozatot, amely alapján kijelenthető, hogy a rendszer nem CCA-biztonságú. Ezeket követően, pedig megadunk egy-egy, az újabb szakirodalom által számon tartott, CCA-biztonságú kulcsbeágyazási verziót ([6], [14], [35], [20]).

5.1. Az RSA-rendszer

Az RSA titkosító rendszert 1978-ban publikálták [55], a rendszer eredeti formájában nem áll ellen se a CPA-támadásnak, se a CCA-támadásnak. Számítástechnikai biztonsága az RSA-feltételezésen alapszik, ahol az ehhez kapcsolódó értelmezést a 4.3.2. szakaszban adtuk meg. Megjelenése

óta számos kitétel létezik a kulcsgeneráló, a titkosító, illetve visszafejtő algoritmusok biztonságos használatára vonatkozóan. Klasszikusan számító kitételek, amelyek részletes tárgyalása megtalálható a ([13], [34], [65]) cikkekben, a következők:

- a kis n modulus problematika,
- a kis e exponens problematika,
- közeli prímek használata,
- kis d exponens használata,
- az n modulus többszöri alkalmazása, stb.

A rendszer [55]-ben publikált verziója alapján és a 4.1. értelmezés szerint a következő algoritmusok adhatók meg:

- Gen, a kulcsgeneráló algoritmus meghatározza az $(e, n, d) \xleftarrow{R} Gen(1^k)$ értékeket, ahol
  - $n = p \cdot q$, és p, q k-bites prímszámok,
  - legyen $e \xleftarrow{R} \mathbb{Z}_n^*$, és meghatározzuk d-t úgy hogy: $e \cdot d = 1 \pmod{(p-1)\cdot(q-1)}$,
  - $pk = (e, n)$, $sk = d$.
- az $Enc_{(e,n)}$ rejtjelező algoritmus polinomiális időben, végzi a rejtjelezést, az $m \in M$ bemenetre elvégez egy moduláris hatványozást: $c \leftarrow m^e \pmod{n}$,
- a Dec_d visszafejtő algoritmus, polinomiális időben végzi a visszafejtést a $c \in C$ rejtjelezett szövegen elvégez egy moduláris hatványozást: $m \leftarrow c^d \pmod{n}$.

Az RSA biztonságos és hatékony implementációjának érdekében szükséges megjegyeznünk a következőket ([30], [44]):

- az RSA kulcsgeneráló algoritmus *nagy* prímek generálását írja elő, amit a probabilisztikus Miler-Rabin vagy Solovay-Strassen algoritmusok implementálásával lehet megoldani,

- a d titkos kulcs meghatározásához a kiterjesztett euklideszi algoritmust lehet alkalmazni,

- a visszafejtés időigényének gyorsítása a kínai maradéktétel alkalmazásával oldható meg,
- a moduláris hatványozás algoritmusának bonyolultsága, ismételt négyzetre emelések és szorzások alkalmazásával: $\mathcal{O}((\log n)^3)$.

A sikeres CCA-támadás lépéssorozata a következő lesz, amely egy polinomiális idejű, probabilisztikus **A** támadó algoritmus és környezete, a kihívó között játszódik le:

5.1. kísérlet.

1. *A kihívó futtatja a Gen kulcsgeneráló algoritmust meghatározza az e, n, d értékeket, majd átadja az (e, n) nyilvános kulcsot az* **A** *támadónak,*
2. *az* **A** *támadó, két azonos hosszúságú üzenetet határoz meg, jelöljük ezeket m_0, m_1-gyel, amelyeket átad a kihívónak,*
3. *a kihívó a $b \xleftarrow{R} \{0,1\}$ választás alapján az $Enc_{(e,n)}$ titkosító algoritmussal meghatározza az m_b titkosított értékét: $c^* \xleftarrow{R} Enc_{(e,n)}(m_b)$, amelyet átad az* **A** *támadónak;*
4. *az* **A** *támadó elvégzi a következőket:*
   - *generál egy m értéket: $m \xleftarrow{R} \mathbb{Z}_n^*$,*
   - *kiszámítja a $c = c^* \cdot m^e$ értéket,*
   - *futtatja a $c \neq c^*$ bemenetre, a kihívó Dec_d visszafejtő algoritmusát, meghatározva ezzel az $\hat{m}$ értéket,*
   - *ha $\hat{m} \cdot m^{-1} = m_0$, akkor $\hat{b} = 0$, másképp $\hat{b} = 1$ lesz az* **A** *támadó választása.*

Az **A** támadó, 4.11. értelmezés szerinti előnye $\frac{1}{2}$, mert fennáll:

$$\hat{m} = (c^* \cdot m^e)^d = (c^*)^d \cdot m^{e \cdot d} = (c^*)^d \cdot m = m_b \cdot m.$$

Formálisan $\mathrm{Adv}^{cca}_{RSA,\mathbf{A}}(k) = |\Pr[b = \hat{b}] - \frac{1}{2}| = |1 - \frac{1}{2}| = \frac{1}{2}$, ahol a b és $\hat{b}$ értékeket az 5.1. kísérletben határoztuk meg.

Azok a titkosító rendszerek, ahol a titkosító algoritmus determinisztikus, nem rendelkezhetnek a CPA, illetve a CCA-biztonság egyikével sem.

Ezért szükséges, hogy a titkosító algoritmust átalakítsuk. Ez a folyamat szükségszerűen a rejtjelezett szöveg adat expanziójával fog járni. Több ilyen biztonságtulajdonsággal rendelkező RSA verzió létezik, de ezek közül az egyik legjelentősebb és leghatékonyabb az RSA–OAEP (RSA-Optimal Asymmetric Encryption Padding). Az RSA–OAEP verziót Bellare és Rogaway dolgozták ki [6], és megadták a véletlen orákulum modellben a CCA-biztonság bizonyítását.

Az RSA–OAEP-titkosító rendszerben alkalmazásra kerül két, egymástól független véletlen orákulum, a G és H függvény, ahol

- $G : \{0,1\}^{l_H} \to \{0,1\}^{l_G}$, egy hash függvényen alapuló ál-véletlenszám generátor,
- $H : \{0,1\}^{l_G} \to \{0,1\}^{l_H}$, egy probabilisztikus hash függvény.

A rendszer három algoritmusa a (P, C, K) halmazhármas felett van értelmezve, ahol $P = \{0,1\}^{l_P}, C = \{0,1\}^{l_H+l_G}, K = \{0,1\}^{l_H+l_G}$, és $l_G,\ l_H$ pozitív egész számok, ahol $l_P < l_G$, $l_G < k$, $l_H < k$ és $k = l_H + l_G$.

- A $Gen(1^k)$ kulcsgeneráló algoritmus, a már leírt módon meghatározza az e, n, d értékeket: $(e, n, d) \xleftarrow{R} Gen(1^k)$, ahol $pk = (e, n)$ és $sk = d$.
- Az $ENC_{(e,n)}(m)$ titkosító algoritmus probabilisztikusan, polinomiális időben meghatározza a c rejtjelezett értéket, ahol
  - $x = m||0^{l_G - l_P}$,
  - $r \xleftarrow{R} \{0,1\}^{l_H}$,
  - $y_1 = x \oplus G(r)$,
  - $y_2 = r \oplus H(x \oplus G(r))$,
  - $c = (y_1||y_2)^e \pmod n$,
- $DEC_{sk}(c)$ visszafejtő algoritmus
  - determinisztikusan, polinomiális időben meghatározza az $y = c^d \pmod n$ értéket, majd felosztja y-t $\hat{y_1}||\hat{y_2}$-re úgy, hogy $|\hat{y_1}| = l_G$ és $|\hat{y_2}| = l_H$,
  - meghatározza $r = H(\hat{y_1}) \oplus \hat{y_2}$,
  - meghatározza $\hat{y} = \hat{y_1} \oplus G(r)$,
  - ellenőrzi, hogy $\hat{y}$ utolsó $l_G - l_P$ bitje nulla-e:

∗ ha nem, akkor REJECT kimeneti értékkel leáll,
∗ ellenkező esetben visszatéríti az első l_P bitjét $\hat{y}$-nak.

A rendszer helyessége elemi számítások során ellenőrizhető, figyelembe véve, az $\oplus$ operátor idempotens tulajdonságát.

A rendszer CCA-biztonságának bizonyítása a következőképpen vázolható fel, részletes bizonyítására, amely megtalálható [6]-ben, jelen értekezés keretén belül nem térünk ki.

Ahhoz, hogy az esetleges támadó az m nyílt szöveg bármilyen bitjére vonatkozóan információt szerezhessen szükséges a $G(r)$ teljes bitszekvenciájának a meghatározása. Ez utóbbi, csak két esetben lehetséges:

- ha sikerül az $r = H(\hat{y_1}) \oplus \hat{y_2}$ meghatározása, azaz ha sikerül invertálni az RSA egyirányú függvényt vagy ha sikerül ütközést találni a H hash függvény esetében,
- ha sikerül ütközést találni a G függvény esetében.

5.2. A Rabin-rendszer

A Rabin titkosító rendszert 1979-ban publikálták [54], a rendszer eredeti formájában nem áll ellen a CCA-támadásnak, ahol a támadás során nem a nyílt szöveget, hanem a titkos kulcsot határozza meg az **A** támadó. Számítástechnikai biztonsága a faktorizációs feltételezésen alapszik, ahol az ehhez kapcsolódó értelmezést a 4.3.1. szakaszban adtuk meg. A rendszer [54]-ben publikált verziója a következő:

- Gen, a kulcsgeneráló algoritmus meghatározza: $(n, p, q) \xleftarrow{R} Gen(1^k)$ értékeket, ahol
  - legyenek p, q k-bites prímszámok úgy, hogy $p \equiv q \equiv 3 \pmod 4$,
  - $n = p \cdot q$,
  - $pk = n$, $sk = (p, q)$.
- az Enc_n rejtjelező algoritmus polinomiális időben, végzi a rejtjelezést, az $m \in M$ bemenetre: $c \leftarrow m^2 \pmod n$,
- a $Dec_{(p,q)}$ visszafejtő algoritmus, polinomiális időben végzi a visszafejtést a $c \in C$ rejtjelezett szövegen: $m \leftarrow \sqrt{c} \pmod n$.

5.1. megjegyzés. *A visszafejtés, azaz $\sqrt{c}$ meghatározása nem egyértelmű: 4 lehetséges visszafejtett szöveg közül kell kiválasztani a megfelelőt. Ennek érdekében alkalmazzuk a kínai maradéktételt, ahol a 4 lehetséges visszafejtett szöveg a következő: $m_1, -m_1, m_2, -m_2$ és ahol:*

- $m_1 = (p^{-1} \cdot p \cdot m_q + q^{-1} \cdot q \cdot m_p) \pmod{n}$,
- $m_2 = (p^{-1} \cdot p \cdot m_q - q^{-1} \cdot q \cdot m_p) \pmod{n}$, *ahol fennáll:*

$$m_p = c^{(p+1)/4} \pmod{p},\ m_q = c^{(q+1)/4} \pmod{q} \text{ és}$$

$$p \cdot p^{-1} = 1 \pmod{q},\ q \cdot q^{-1} = 1 \pmod{p}.$$

A Rabin-titkosító CCA-biztonságú verzióját, a véletlen orákulum modellben Boneh dolgozta ki [14], ahol biztonságát a faktorizációs feltételezés alapján bizonyította. A rendszer SAEP (Simplified Asymmetric Encryption Padding) néven ismert. A rendszer az RSA–OAEP-titkosítóhoz képest jóval egyszerűbb, egyetlen hash függvényt alkalmaz: $H : \{0,1\}^{l_r} \to \{0,1\}^{l_H}$.

A rendszer három algoritmusa a (P, C, K) halmaz hármas felett van értelmezve, ahol $P = \{0,1\}^{l_P}, C = \{0,1\}^{l_H+l_r}, K = \{0,1\}^{l_H+l_r}$, és $l_H,\ l_r$ pozitív egész számok, $l_P < l_H$, és $k = l_H + l_r$.

- A $Gen(1^k)$ kulcsgeneráló algoritmus, meghatároz egy $(k+2)$ bites $n = p \cdot q$ számot, ahol
  - az n legfelsőbb helyértékű két bitje 1 és 0,
  - p, q pedig két $k/2+1$ bites prímszám úgy, hogy: $p \equiv q \equiv 3 \pmod{4}$,
  - $pk = n$ és $sk = (p, q)$.
- Az $ENC_{pk}(m)$ titkosító algoritmus probabilisztikusan, polinomiális időben meghatározza a c rejtjelezett értéket, ahol
  - $x = m||0^{l_H - l_P}$, ahol az $||$ operátor összefűzést jelent,
  - $r \xleftarrow{R} \{0,1\}^{l_r}$,
  - $y = (x \oplus H(r))||r$,
  - $c = y^2 \pmod{n}$,
- $DEC_{sk}(c)$ visszafejtő algoritmus a következőket végzi:

- meghatározza $y_p = c^{(p+1)/4} \pmod p$, $y_q = c^{(q+1)/4} \pmod q$ értékeket, majd alkalmazza a kínai maradéktételt, $y_1, -y_1, y_2, -y_2$ lesz a 4 megoldás:
  - $y_1 = (p^{-1} \cdot p \cdot y_q + q^{-1} \cdot q \cdot y_p) \pmod n$,
  - $y_2 = (p^{-1} \cdot p \cdot y_q - q^{-1} \cdot q \cdot y_p) \pmod n$.
- megvizsgálja, melyik igaz: $y_p^2 = c \pmod p$, $y_q^2 = c \pmod q$,
- ha egyik sem teljesül, akkor REJECT kimeneti értékkel leáll,
- ezután feltételezhető, hogy a 4 lehetséges négyzetgyök közül kettő biztosan nagyobb mint $n/2$, marad 2 lehetséges megoldás, legyenek ezek y_1, y_2,
- ellenőrzi, hogy, ha y_1 és y_2 is nagyobb mint 2^k, akkor REJECT kimeneti értékkel leáll,
- ellenőrzi, hogy ha csak az egyik kisebb mint 2^k, akkor legyen ez y_1, majd a következő pontnál megadott számításszorozatot kell követni, de az y_2 értékkel nem kell számolni,
- ezután feltételezhető, hogy y_1 is és y_2 is kisebb mint 2^k:
  - felírjuk $y_1 = v_1||r_1$ $y_2 = v_2||r_2$ úgy, hogy $r_1, r_2 \in \{0,1\}^{l_r}$
  - meghatározzuk $x_1 = v_1 \oplus H(r_1)$ $x_2 = v_2 \oplus H(r_2)$,
  - felírjuk $x_1 = m_1||t_1$ és $x_2 = m_2||t_2$ úgy, hogy $m_1, m_2 \in \{0,1\}^{l_P}$
  - ha $t_1 = 0^{l_H - l_P}$ és $t_2 = 0^{l_H - l_P}$, akkor REJECT kimeneti értékkel leáll,
  - ha $t_1 \neq 0^{l_H - l_P}$ és $t_2 \neq 0^{l_H - l_P}$, akkor REJECT kimeneti értékkel leáll,
  - ha $t_1 = 0^{l_H - l_P}$, akkor visszatéríti m_1-et, mint visszafejtett értéket,
  - ha $t_2 = 0^{l_H - l_P}$, akkor visszatéríti m_2-et, mint visszafejtett értéket.

A rendszer CCA-biztonságának bizonyítása a következőképpen vázolható fel, részletes bizonyításra jelen értekezés keretén belül nem térünk ki [14].

Ha sikerül egy olyan **A** támadó algoritmust szerkeszteni, amely az $n, l_P, l_H - l_P, l_r, c = y^2 \pmod n$ bemeneti értékekre, ahol $y \xleftarrow{R} \{0, \ldots, 2^k - 1\}$, meghatározza c egy $y^* \neq y$ négyzetgyökét, akkor lehetséges az n

faktorizálása az n és $y - y^*$ értékek legnagyobb közös osztójának a meghatározásával.

Coppersmith egy cikke alapján [19] Boneh-nek sikerült, azt bebizonyítani, hogy elégséges egy olyan **A** támadó algoritmust szerkeszteni, amely ha 1/3-nál nagyobb valószínűséggel megmondja, hogy a $c \overset{R}{\leftarrow} \{0, \ldots n-1\}$ számnak két különböző négyzetgyöke van az $\{0, \ldots 2^k - 1\}$ halmazon $\pmod n$ szerint, akkor 1/6-nál nagyobb a valószínűsége annak, hogy az **A** támadó algoritmus által meghatározott y, y^*-ra fennálljon, hogy: $y \neq y^*$, ami alapján lehetséges az n faktorizálása.

5.3. A Blum–Goldwasser-rendszer

A Blum–Goldwasser titkosító rendszert, amelyre BG-titkosító néven fogunk a továbbiakban hivatkozni, 1985-ban publikálták [12]. A rendszer eredeti formájában CPA-biztonságú, viszont nem áll ellen a választott rejtjelezett szöveg alapú támadásnak. A BG-titkosító egy álvéletlen bitsorozat generátoron alapszik, a Blum-Blum-Shub generátoron, amelyre BBS-generátor néven fogunk a továbbiakban hivatkozni. A BBS generátort 1982-ben publikálták [10] és bebizonyították róla, hogy kriptográfiailag biztonságos. Biztonságát a kvadratikus maradék feltételezés (lásd a 4.3.3. szakaszban), alapján bizonyították.

A 2.9. formális értelmezés egy alternatív megközelítése a következő [61]: egy álvéletlen bitsorozat generátorról, akkor mondjuk, hogy kriptográfiailag biztonságos, ha az első l álvéletlen bit meghatározása után, a kezdeti *seed* (kezdőérték) ismeretének hiányában, nem lehet $1/2 + \varepsilon$-nál nagyobb valószínűséggel megmondani, hogy mi lesz az $(l+1)$-ig bit, ahol ε nem elhanyagolható érték.

A BG-titkosító a BBS által generált byte-okat összeadja a nyílt szöveg byte-jaival, bitenkénti $\pmod 2$ szerinti összeadást végezve.

A továbbiakban előbb a BBS-generátort mutatjuk be.

Legyen $n = p \cdot q$, ahol p, q tetszőleges k-bites prímszámok, azzal a tulajdonsággal, hogy $p \equiv q \equiv 3 \pmod 4$. Jelöljük $\mathbb{QR}_n$-el a kvadratikus maradékok halmazát $\pmod n$ szerint, amelyet formálisan a 4.3.3. fejezetnél adtunk meg.

Legyen $r \overset{R}{\leftarrow} \mathbb{Z}_n^*$, és legyen $r_0 = r^2 \pmod n$, ekkor teljesülni fog, hogy $r_0 \in \mathbb{QR}_n$. Az álvéletlen bitsorozat generátor kiindulva a kezdeti r_0, mint *seed* értékből ismételten alkalmazza a moduláris négyzetreemelés függvényt. A véletlenszerűen generált biteket: $b_1, b_2, \ldots, b_l$-el jelölve, az algoritmus a következőképpen végzi ezek meghatározását:

$$r_{i+1} = r_i^2 \pmod n, \text{ minden } 0 \leq i \leq l,$$

és

$$b_i = r_i \pmod 2, \text{ minden } 1 \leq i \leq l.$$

Jelöljük $BBS(r_0, l)$-el a BBS álvéletlen bitsorozat generátor által meghatározott bitsorozatot, azaz a $b_1||b_2||\ldots||b_l$-t, ahol r_0 a kezdeti *seed* érték és l a generált bitsorozat hossza.

Ezek után megadható a BG-titkosító, a 4.1. értelmezés szerinti három algoritmus alapján:

- A Gen, a kulcsgeneráló algoritmus meghatározza az $(n, p, q) \overset{R}{\leftarrow} Gen(1^k)$ értékeket, ahol
  - legyenek p, q k-bites prímszámok úgy, hogy $p \equiv q \equiv 3 \pmod 4$,
  - $n = p \cdot q$,
  - $pk = n$, $sk = (p, q)$,
- az Enc_n probabilisztikus, rejtjelező algoritmus polinomiális időben, végzi a rejtjelezést, az $m = (m_1||m_2||\ldots||m_l) \in \mathbb{Z}_2^l$ bemenetre, ahol $m_i, i \in \{1, \ldots l\}$ a nyílt szöveg bitjei:

$$Enc_n(m_1||m_2||\ldots||m_l, r_0) = c_1||c_2||\ldots||c_l||r_{l+1}, \text{ ahol}$$

  - $c_i = (m_i \oplus b_i), 1 \leq i \leq l$,
  - r_0 a BBS generátor kezdeti értéke,
  - $b_1, b_2, \ldots, b_l$ a BBS generátor által generált álvéletlen bitek,
  - $r_{l+1} = r_0^{2^{l+1}} \pmod n$,
- a $Dec_{(p,q)}$ determinisztikus, visszafejtő algoritmus, polinomiális időben végzi a visszafejtést a $c = (c_1||c_2||\ldots||c_l||r_{l+1}) \in \mathbb{Z}_2^l \times \mathbb{Z}_n^*$ rejtjelezett szövegen:

$$Dec_{(p,q)}(c_1||c_2||\ldots||c_l||r_{l+1}) = m_1||m_2||\ldots||m_l, \text{ ahol}$$

- $m_i = c_i \oplus b_i$, $1 \leq i \leq l$,
- $b_1, b_2, \ldots, b_l$ a BBS generátor által generált álvéletlen bitek,
- a BBS generátor kezdeti r_0 értéke, alkalmazva a kínai maradéktételt, a következőképpen határozható meg:

$$r_0 = r_{l+1}^{a_1} \pmod{p}, \text{ ahol } a_1 = ((p+1)/4)^{l+1} \pmod{p-1},$$
$$r_0 = r_{l+1}^{a_2} \pmod{q}, \text{ ahol } a_2 = ((q+1)/4)^{l+1} \pmod{q-1}.$$

Bebizonyítható, hogy minden egyes moduláris négyzetre emelés esetén egyetlen bit helyett, akár $\log_2(\log_2(n))$ bit is felhasználható [62].

A Blum–Goldwasser-titkosító CCA-biztonságú verzióját, a standard modellben, mint egy hibrid rendszer kulcsbeágyazási mechanizmusa Hofheinz, Kiltz és Shoup dolgozták ki [35].

A rendszer matematikai alapját a moduláris négyzetreemelés függvény többszöri alkalmazása képezi, amelyet az algoritmus egy véletlenszerűen generált értéken alkalmaz, amelyhez hozzá társul egy konzisztencia ellenőrző elem. A négyzetreemelés függvény bemeneti értékét a $\mathbb{QR}_N^+$ halmazból veszi, ahol a $\mathbb{QR}_N^+$ halmaz a kvadratikus maradékok egy leszűkített csoportját fogja jelenti, és N Blum-egész.

A CCA-biztonságú kulcsbeágyazási mechanizmus esetében a következő *Gen* kulcsgeneráló algoritmusra van szükségünk:

5.1. algoritmus. *A Gen kulcsgeneráló algoritmus az 1^k bemeneten meghatározza az (N, P, Q) számhármast: $(N, P, Q) \xleftarrow{R} Gen(1^k)$ úgy, hogy:*

- *$N = P \cdot Q$, ahol $P \equiv Q \equiv 3 \pmod 4$, és*
- *$P = 2 \cdot p + 1, Q = 2 \cdot q + 1$, ahol p, q prímszámok, azaz P és Q biztonságos prímek lesznek,*
- *a p és a q bithossza $k/2 - 1$,*
- *az így generált N számot Blum-egésznek, a P, Q prímeket biztonságos (safe) prímeknek hívják.*

A már említett, $\mathbb{QR}_N^+$ leszűkített kvadratikus maradékok halmazának formális értelmezéséhez felhasználjuk a 4.3.3. szakasznál értelmezett $\mathbb{QR}_N$ halmazt, ahol N az 5.1. algoritmus által generált összetett szám:

5.1. definíció. $\mathbb{QR}_N^+ = \{|x| : x \in \mathbb{QR}_N\}$, *ahol* $|x|$ *az* x *szám abszolút értékét jelöli.*

A kulcsbeágyazási mechanizmus egy cél-ütközésmentes hash függvényt is használ, amelyet a következőképpen jelölünk: $H : \mathbb{QR}_N^+ \to \{0,1\}^{l_H}$.

A kulcsbeágyazási mechanizmus három algoritmusa, pedig a következő, ahol a műveleteket $\mathbb{QR}_N^+$-ben végezzük:

- A *Gen* kulcsgeneráló algoritmus az 1^k bemeneten meghatározza az (N, P, Q, g, X, α) számötöst, ahol
  - az (N, P, Q) értékeket az 5.1. algoritmussal határoztuk meg,
  - $g \xleftarrow{R} \mathbb{QR}_N^+$, $\alpha \xleftarrow{R} \{1, \ldots, (N-1)/4\}$, és $X = g^{\alpha \cdot 2^{l_H + l_L}}$,
  - az l_H az alkalmazott hash függvény kimenetének bithossza, az l_L érték, pedig a BBS által generált bitek számát jelenti,
  - az algoritmus kimeneti értékei a $pk = (N, g, X)$, és $sk = \alpha$.
- Az Enc_{pk} probabilisztikus kulcsbeágyazási algoritmus, polinomiális időben meghatározza $(key, cipher)$ kimenetet, ahol:
  - $r \xleftarrow{R} \{1, \ldots, (N-1)/4\}$,
  - $K = g^{r \cdot 2^{l_H}}, key = BBS(K, l_L)$,
  - $cipher = (R, S)$, ahol
    $$R = g^{r \cdot 2^{l_H + l_L}},\ h = H(R),\ S = (g^h \cdot X)^r.$$
- A $Dec_{sk}(cipher)$ determinisztikus kulcskibontási algoritmus, polinomiális időben a következőket végzi:
  - ha nem teljesül, hogy $R \in \mathbb{QR}_N^+$, és $S \in \mathbb{QR}_N^+$, akkor REJECT kimeneti értékkel leáll az algoritmus,
  - ellenkező esetben meghatározza $h = H(R)$,
  - ha nem teljesül az
    $$S^{2^{l_H + l_L}} = R^{h + \alpha \cdot 2^{l_H + l_L}} \tag{5.1}$$
    egyenlőség, akkor az algoritmus REJECT kimeneti értékkel leáll,
  - ellenkező esetben meghatározza a K értéket, mint kimeneti értéket a következőképpen:
    $$K = (S^a \cdot R^{b - a\alpha})^{2^{l_H - c}},\ \text{ahol} \tag{5.2}$$

az a, b, c értékeket a következő diofantikus egyenletből, a kiterjesztett euklideszi algoritmussal [3] lehet meghatározni:

$$2^c = a \cdot h + b \cdot 2^{l_H + l_L}.$$

A rendszer helyessége a következő elemi számítások elvégzése során látható be.

Az R és S értékek a következő formában írhatóak fel:

$$\begin{aligned} S &= (g^h \cdot g^{\alpha \cdot 2^{l_H + l_L}})^r, \\ R &= g^{r \cdot 2^{l_H + l_L}}. \end{aligned}$$

Az (5.1) egyenlőség helyességének a belátásához a következők írhatóak fel:

$$\begin{aligned} S^{2^{l_H + l_L}} &= g^{(h + \alpha \cdot 2^{l_H + l_L}) \cdot r \cdot 2^{l_H + l_L}}, \\ R^{h + \alpha \cdot 2^{l_H + l_L}} &= g^{r \cdot 2^{l_H + l_L} \cdot (h + \alpha \cdot 2^{l_H + l_L})}. \end{aligned}$$

A kulcs kiszámításának helyessége a következőképpen ellenőrizhető:

$$\begin{aligned} S^a \cdot R^{b - a \cdot \alpha} &= (S \cdot R^{-\alpha})^a \cdot R^b, \\ S \cdot R^{-\alpha} &= g^{hr} \cdot g^{r\alpha \cdot 2^{l_H + l_L}} \cdot g^{-r\alpha \cdot 2^{l_H + l_L}} = g^{hr}, \\ S^a \cdot R^{b - a \cdot \alpha} &= g^{hra} \cdot g^{br \cdot 2^{l_H + l_L}} = g^{r(ha + b \cdot 2^{l_H + l_L})} = g^{r \cdot 2^c}, \\ (S^a \cdot R^{b - a \cdot \alpha})^{2^{l_H - c}} &= g^{r \cdot 2^c \cdot 2^{l_H - c}} = g^{r l_H}. \end{aligned}$$

A rendszer CCA-biztonságának bizonyítása a következőképpen vázolható fel, részletes bizonyítása, amelyre nem térünk ki, a [35] cikkben található meg.

Ha feltételezzük, hogy létezik egy **D** támadó algoritmus, amely egy véletlenszerűen generált N érték ismeretében $1/2 + \varepsilon$ valószínűséggel, ahol ε nem elhanyagolható érték, meg tudja különböztetni egymástól a BBS által generált biteket egy ugyanolyan l_L hosszú, véletlenszerűen generált bitsorozattól, akkor szerkeszthető egy olyan probabilisztikus, polinomiális **A** támadó algoritmus, amely nem elhanyagolható valószínűséggel faktorizálni tudja az N számot; ezzel, pedig ellent mondunk a faktorizációs feltételezésnek.

5.4. Az ElGamal-rendszer

Az ElGamal titkosító rendszert T. ElGamal publikálta, és bizonyította, hogy a rendszer ellenáll a CPA-támadásnak, viszont a CCA-támadásnak nem [27]. CPA-biztonságát a diszkrét logaritmus feltételezés alapján bizonyította, ahol az ehhez kapcsolódó értelmezést a 4.3.4. szakaszban adtuk meg. A diszkrét logaritmus problémához kapcsolódó felvetések egyik részletes tárgyalása a [51] cikkben is olvasható. A rendszer minden olyan ciklikus algebrai csoportban implementálható, ahol fennáll a döntési Diffie–Hellman-feltételezés, ahol az ehhez kapcsolódó értelmezést pedig a 4.3.5. szakaszban adtuk meg.

A rendszer [27]-ben publikált verziója alapján és a 4.1. értelmezés szerint a következő algoritmusok adhatók meg:

- Gen, a kulcsgeneráló algoritmus meghatároz egy G ciklikus csoportot, amelynek rendje a q, majd
  - meghatározza a g generátor elemet és az $\alpha \xleftarrow{R} \{0, \ldots, q-1\}$ számot,
  - legyen $A = g^\alpha$, és
  - $pk = (g, A)$, $sk = \alpha$.
- az $Enc_{(G,g,A)}$ probabilisztikus rejtjelező algoritmus polinomiális időben végzi a rejtjelezést az $m \in G$ bemenetre: $c \leftarrow Enc_{(G,g,A)}(m)$, ahol
  - $\beta \xleftarrow{R} \{0, \ldots, q-1\}$, $B = g^\beta$,
  - $c = (c_1, c_2)$, ahol $c_1 = B$, $c_2 = A^\beta \cdot m$.
- a Dec_α determinisztikus visszafejtő algoritmus polinomiális időben végzi a visszafejtést a $c \in G$ rejtjelezett szövegen: $m \leftarrow Dec_\alpha(c)$, ahol
  - $m = (c_1^\alpha)^{-1} \cdot c_2$.

A sikeres CCA-támadás lépéssorozata a következő lesz, amely egy polinomiális idejű, probabilisztikus **A** támadó algoritmus és környezete, a kihívó között játszódik le:

5.2. kísérlet.

1. *A kihívó futtatja a Gen kulcsgeneráló algoritmust, meghatározza a* G, q, g, α, A *értékeket, majd átadja az* (G, g, A) *nyilvános adatokat az* **A** *támadónak,*
2. *az* **A** *támadó, két azonos hosszúságú üzenetet határoz meg, jelöljük ezeket* m_0, m_1*-gyel, amelyeket átad a kihívónak,*
3. *a kihívó a* $b \xleftarrow{R} \{0,1\}$ *választás alapján az Enc titkosító algoritmussal meghatározza az* m_b *titkosított értékét:* $c^* = (c_1^*, c_2^*) \xleftarrow{R} Enc_{(G,g,A)}(m_b)$*, amelyet átad az* **A** *támadónak,*
4. *az* **A** *támadó elvégzi a következőket:*
   - *véletlenszerűen generálja* β*-t és* m*-t:*
   $$\beta \xleftarrow{R} \{0, \ldots, q-1\},\ m \xleftarrow{R} \{0, \ldots, q-1\},$$
   - *kiszámítja* $c_1 = c_1^* \cdot g^\beta$*, és* $c_2 = c_2^* \cdot m \cdot A^\beta$ *értékeket,*
   - *futtatja a* $c = (c_1, c_2)$ *bemenetre, ahol* $c \neq c^*$*, a kihívó* $Dec_\alpha(\cdot)$ *visszafejtő algoritmusát, meghatározva ezzel az* $\hat{m}$ *értéket,*
   - *ha* $\hat{m} \cdot m^{-1} = m_0$*, akkor* $\hat{b} = 0$*, másképp* $\hat{b} = 1$ *lesz az* **A** *támadó választása.*

Az **A** támadó 4.11. értelmezés szerinti előnye $\frac{1}{2}$, mert fennáll:

$$\begin{array}{ll} \hat{m} & = \\ (c_1^\alpha)^{-1} \cdot c_2 & = \\ ((c_1^* \cdot g^\beta)^\alpha)^{-1} \cdot c_2^* \cdot m \cdot A^\beta & = \\ ((c_1^*)^\alpha)^{-1} \cdot g^{-\beta \cdot \alpha} \cdot c_2^* \cdot m \cdot g^{\alpha \cdot \beta} & = \\ ((c_1^*)^\alpha)^{-1} \cdot c_2^* \cdot m & = \\ m_b \cdot m. & \end{array}$$

Formálisan $\mathrm{Adv}^{cca}_{ElGamal,\mathbf{A}}(k) = |\Pr[b = \hat{b}] - \frac{1}{2}| = |1 - \frac{1}{2}| = \frac{1}{2}$, ahol a b és $\hat{b}$ értékeket a 5.2. kísérletben határoztuk meg.

Standard modellben az ElGamal CCA-biztonságú verzióját Cramer és Shoup dolgozták ki [21]. Az ElGamal egy másik fontos CCA-biztonságú verziója a DHIES [1]. A Cramer és Shoup által szerkesztett rendszer CCA-biztonságát, a szerzők a döntési Diffie–Hellman-feltételezés alapján bizonyították, illetve feltételezték, hogy az alkalmazott hash függvény cél-ütközésmentes hash függvény (lásd a 3.2. értelmezést). Fontos megjegyezni, hogy ez volt az első hatékony CCA-biztonságú kulcsbeágyazási

mechanizmus. Több verziója is létezik, aszerint hogy az alkalmazásra kerülő hash függvény milyen tulajdonságú: univerzális hash függvény, cél-ütközésmentes hash függvény vagy ütközésmentes hash függvény.

Abban az esetben amikor az alkalmazásra kerülő hash függvény cél-ütközésmentes hash függvény a Cramer és Shoup titkosító három algoritmusa a következő:

- *Gen*, a kulcsgeneráló algoritmus meghatároz egy G ciklikus csoportot, amelynek rendje q, majd:
  - meghatározza a g_1, g_2 generátor elemeket,
  - véletlenszerűen meghatároz öt számot:
    $$(x_1, x_2, y_1, y_2, \alpha) \xleftarrow{R} \{0, \ldots, q-1\},$$
    amelyek alapján kiszámítja a következő értékeket:
    $$e = g_1^{x_1} \cdot g_2^{x_2},\ f = g_1^{y_1} \cdot g_2^{y_2} \text{ és } A = g_1^{\alpha}$$
  - a nyilvános adatok: (G, q, g_1, g_2),
  - a nyilvános kulcs: $pk = (e, f, A)$,
  - a titkos kulcs: $sk = (x_1, x_2, y_1, y_2, \alpha)$.
- az $Enc_{(G,q,g_1,g_2,e,f,A)}$ probabilisztikus rejtjelező algoritmus polinomiális időben, végzi a rejtjelezést az $m \in G$ bemenetre:
  $$(a_1, a_2, c, d) \leftarrow Enc_{(G,q,g_1,g_2,e,f,A)}(m), \text{ ahol}$$
  - legyen $\beta \xleftarrow{R} \{0, \ldots, q-1\}$, $a_1 = g_1^{\beta}$, $a_2 = g_2^{\beta}$,
  - $c = A^{\beta} \cdot m$,
  - $h = H(a_1, a_2, c)$, ahol H cél-ütközésmentes hash függvény,
  - $d = e^{\beta} \cdot f^{\beta \cdot h}$.
- a $Dec_{(x_1,x_2,y_1,y_2,\alpha)}$ determinisztikus visszafejtő algoritmus polinomiális időben végzi a visszafejtést az (a_1, a_2, c, d) rejtjelezett szövegen:
  $$m \leftarrow Dec_{(x_1,x_2,y_1,y_2,\alpha)}(a_1, a_2, c, d), \text{ ahol}$$
  - meghatározza: $h = H(a_1, a_2, c)$,

– ha nem áll fenn a következő egyenlőség:

$$d = a_1^{x_1+y_1\cdot h} \cdot a_2^{x_2+y_2\cdot h}, \tag{5.3}$$

akkor REJECT elutasító kimeneti értékkel leáll az algoritmus, ellenkező esetben

∗ meghatározza:

$$m = c \cdot (a_1^{\alpha})^{-1}. \tag{5.4}$$

A rendszer helyessége néhány elemi számítás során ellenőrizhető, a d értéke, a (5.3) egyenlőségben a következő formában írható fel:

$$\begin{array}{ll} d & = \\ e^{\beta} \cdot f^{\beta\cdot h} & = \\ (g_1^{x_1} \cdot g_2^{x_2})^{\beta} \cdot (g_1^{y_1} \cdot g_2^{y_2})^{\beta\cdot h} & = \\ g_1^{\beta\cdot x_1} \cdot g_2^{\beta\cdot x_2} \cdot g_1^{\beta\cdot y_1\cdot h} \cdot g_2^{\beta\cdot y_2\cdot h} & = \\ a_1^{x_1+y_1\cdot h} \cdot a_2^{x_2+y_2\cdot h}. & \end{array}$$

Az 5.4 egyenlőség pedig a következő formában írható fel:

$$c \cdot (a_1^{\alpha})^{-1} = (A^{\beta} \cdot m) \cdot (a_1^{\alpha})^{-1} = g_1^{\alpha\beta} \cdot m \cdot (g_1^{\beta\alpha})^{-1} = m.$$

A rendszer CCA-biztonságának bizonyítása a következőképpen vázolható fel, részletes bizonyításra jelen értekezés keretén belül nem térünk ki [20].

Ha feltételezzük, hogy létezik egy **A** támadó, amely egy véletlenszerűen generált nyilvános kulcs ismeretében $1/2 + \varepsilon$ valószínűséggel (ahol ε nem elhanyagolható érték) meg tud különböztetni egymástól bizonyos rejtjelezett szövegeket, akkor szerkeszthető egy olyan probabilisztikus, polinomiális **B** támadó algoritmus, amely nem elhanyagolható valószínűséggel különbséget tesz a $(g_1^{\beta}, g_1^{\lambda}, g_1^{\beta\lambda})$ és $(g_1^{\beta}, g_1^{\lambda}, g_1^{z})$ között amellyel ellent mondunk a döntési Diffie–Hellman-feltételezésnek, ahol β, z véletlenszerűen meghatározott értékek.

5.2. megjegyzés. *A CS rendszerben az (5.3) egyenlőség a következő formában is felírható, ahol alkalmazható a $g_2 = g_1^{\lambda}$ jelölés:*

$$(g_1^{\beta})^{x_1+y_1\cdot h} \cdot (g_2^{\beta})^{x_2+y_2\cdot h} = a_1^{x_1+y_1\cdot h} \cdot a_2^{x_2+y_2\cdot h}. \tag{5.5}$$

Az (5.5) egyenlőséget csak azok a rejtjelezett szövegek elégítik ki, amelyek esetében fennáll, hogy $\log_{g_1} a_1 = \log_{g_2} a_2$*, és nagyon kicsi a valószínűsége annak, hogy egy olyan rejtjelezett szöveg is teljesítse az (5.5) egyenlőséget, amelyre ez nem áll fenn, ami alapján megszerkeszthető a* **B** *algoritmus.*

5.3. megjegyzés. *Abban az esetben, ha az* m *értéke valamely hibrid rendszer keretén belül véletlenszerűen kerül kiválasztásra, a fenti titkosító, kulcsbeágyazási mechanizmusnak tekinthető. Ez a korábban tárgyalt titkosító rendszerek mindegyikére igaz.*

6. fejezet

Az új kulcsbeágyazási mechanizmus

6.1. Bevezető

Az értekezés keretén belül bemutatásra kerülő CPA-biztonságú és CCA-biztonságú kulcsbeágyazási mechanizmusok alapjául a Rabin-titkosító és a Hofheinz és társai [35], által szerkesztett rendszerek szolgáltak. A CCA-biztonságú kulcsbeágyazási mechanizmus a [48] cikkben került publikálásra.

Kriptográfiai rendszerek szerkesztésénél Rabin [54] alkalmazta először a négyzetreemelés függvényt és bebizonyította, hogy ennek a függvénynek az invertálása ugyanolyan nehéz matematikai feladat, mint az egész számok faktorizációs problémája. Az 5.2. alfejezetben bemutattuk, hogy az eredeti rendszer teljesen védtelen egy választott rejtjelezett szöveg alapú támadással szemben. A fejezetben bemutatásra kerülő kulcsbeágyazási mechanizmusok szintén a négyzetreemelés függvényen alapulnak, azzal a különbséggel, hogy az alkalmazott egyirányú függvények, bemeneti értéküket egyik esetben a $\mathbb{QR}_N$ halmazból, másik esetben a $\mathbb{QR}_N^+$ halmazból veszik, ahol a $\mathbb{QR}_N^+$ halmazt az 5.1. értelmezéssel adtuk meg. A rendszerek CPA-biztonságát, illetve CCA-biztonságát be tudtuk bizonyítani

a faktorizációs feltételezésből, illetve az alkalmazott hash függvény cél-ütközésmentes tulajdonságából kiindulva. A faktorizációs feltételezésen alapuló titkosító rendszerek biztonságát a [49] cikkben tárgyaltuk.

A továbbiakban először megadjuk a CPA-biztonságú kulcsbeágyazási mechanizmus szerkesztésének lépéssorozatát, majd a fejezet második felében a CCA-biztonságú kulcsbeágyazási mechanizmust mutatjuk be.

Mindkét kulcsbeágyazási mechanizmus esetében a következő *Gen* kulcsgeneráló algoritmusra van szükségünk:

6.1. algoritmus. *A Gen kulcsgeneráló algoritmus az* 1^k *bemeneten meghatározza az* (N, P, Q) *számhármast:* $(N, P, Q) \xleftarrow{R} Gen(1^k)$ *úgy, hogy:*

- $N = P \cdot Q$*, ahol* P, Q *prímszámok, úgy hogy*
  - $P \equiv Q \equiv 3 \pmod 4$*, és*
  - $P = 2 \cdot p + 1, Q = 2 \cdot q + 1$*, ahol* p, q *szintén prímszámok,*
- *A* p *és a* q *bithossza* $k/2 - 1$*,*

A fenti *Gen* kulcsgeneráló algoritmus által generált értékek függvényében a faktorizációs feltételezés a következő formában jelenthető ki:

6.1. definíció. *Bármely probabilisztikus polinomiális idejű* **A** *algoritmus esetében, létezik egy* $f(k)$ *elhanyagolható függvény úgy, hogy*

$$\mathrm{Adv}_{FAC,\mathbf{A}}(k) = \Pr[\mathbf{A}(N) = (P, Q)] \leq f(k),$$

6.2. Az új CPA-biztonságú rendszer

6.2.1. A rendszer leírása

Az új kulcsbeágyazási mechanizmus egy cél-ütközésmentes hash függvényt fog használni, jelöljük ezt $G : \mathbb{QR}_N \to \{0,1\}^{l_G}$-val.

Az új CPA-biztonságú kulcsbeágyazási mechanizmus, a 4.15. értelmezés szerinti három algoritmusa a következő:

- A Gen^{cpa}_{KEM} egy kulcsgeneráló algoritmus, amely meghatározza az $(N, P, Q) \xleftarrow{R} Gen(1^k)$ értékeket, ahol

  - az N, P, Q értékeket a 6.1. algoritmussal határozta meg,
  - $pk = N$, és $sk = (P, Q)$.
- Az Enc^{cpa}_{KEM} kulcsbeágyazási algoritmus az N bemeneten a következőket végzi:
  - generálja r-t: $r \xleftarrow{R} \{1, \ldots, (N-1)\}$,
  - meghatározza: $K = r^2 \pmod N \in \mathbb{QR}_N$, és $key = G(K)$, ahol $G : \mathbb{QR}_N \to \{0,1\}^{l_G}$ cél-ütközésmentes hash függvény,
  - meghatározza: $cipher = K^2 \pmod N$,
  - az algoritmus kimeneti értéke $(key, cipher)$.
- A Dec^{cpa}_{KEM} kulcskibontási algoritmus a (P, Q) és $cipher$ bemeneti értékeken a következőket végzi:
  - a Kínai maradéktételt alkalmazva meghatározza a K értékét, megoldva a következő kongruencia rendszert:
$$K = cipher^{(P+1)/4} \pmod P,$$
$$K = cipher^{(Q+1)/4} \pmod Q,$$
  - az algoritmus kimeneti értéke a $key = G(K)$ lesz.

6.2.2. A rendszer helyessége

A rendszer helyességének bizonyításához szükség van a fő négyzetes gyök fogalmának az értelmezésére [61]:

6.2. definíció. *Az $x \in \{1, \ldots, N\}$ számot az y kvadratikus maradék fő négyzetes gyökének nevezzük, ha x szintén kvadratikus maradék $\pmod N$ szerint.*

Szükséges továbbá a következő tételt is kijelentenünk, amelynek bizonyítása megtalálható [61]-ben.

6.1. tétel. *Ha a P, Q prímszámokra teljesül, hogy $P \equiv Q \equiv 3 \pmod 4$, akkor bármely x kvadratikus maradék esetén egyetlen olyan négyzetes gyöke van az x-nek, amely szintén kvadratikus maradék $\pmod N$ szerint.*

Ezután belátható az új rendszer helyessége a következők alapján:

- mivel $N = P \cdot Q$ és $P \equiv Q \equiv 3 \pmod 4$, igaz, hogy a *cipher* tetszőleges, kvadratikus maradék $\pmod P$ szerinti négyzetes gyöke meghatározható a következő képlettel: $\pm cipher^{(P+1)/4} \pmod P$,
- de *cipher*-nek $\pmod P$ szerint $cipher^{(P+1)/4} \pmod P$ a fő négyzetes gyöke,
- hasonlóan *cipher*-nek $\pmod Q$ szerint $cipher^{(Q+1)/4} \pmod Q$ a fő négyzetes gyöke,
- alkalmazható ezek után a kínai maradéktétel, tehát $K = r^2 \pmod N$ a $cipher = K^2 \pmod N$ fő négyzetes gyöke, $\pmod N$ szerint.

6.2.3. A rendszer biztonsága

Az új rendszer CPA-biztonságának bizonyításához a következő kísérletet értelmezzük, amely kísérlet egy polinomiális idejű, probabilisztikus **A** támadó és a környezete között, a kihívó között játszódik le:

6.1. kísérlet.

1. *A kihívó futtatja a* $Gen_{KEM}^{cpa}(1^k)$ *kulcsgeneráló algoritmust, meghatározza az* $(N, (P, Q))$ *értékeket, majd a az* N *értéket átadja az* **A** *támadónak,*
2. *az* **A** *támadó tetszőleges számú alkalommal meghívja a kihívó* $Enc_{KEM}^{cpa}(N)$ *és* $G(\cdot)$ *algoritmusait,*
3. *a kihívó elvégzi a következőket:*
   - *meghatározza:* $(key^*, cipher^*) \leftarrow Enc_{KEM}^{cpa}(pk)$,
   - *generál egy u értéket:* $u \xleftarrow{R} \mathbb{QR}_N$,
   - *a* (key^*)*-t* c_0*-val jelöli,*
   - *a* $(G(u))$*-t* c_1*-gyel jelöli,*
   - *átadja az* **A** *támadónak a* $(c_b, cipher^*)$ *titkosított értékpárt, ahol* $b \xleftarrow{R} \{0, 1\}$,
4. *az* **A** *támadó tetszőleges számú alkalommal meghívja a kihívó* $Enc_{KEM}^{cpa}(N)$ *és* $G(\cdot)$ *algoritmusait,*
5. *az* **A** *támadó meghatároz egy* $\hat{b} \in \{0, 1\}$ *kimeneti értéket.*

Jelöljük a továbbiakban $Expr(0)$-val azt az eseményt, amikor $\hat{b} = 1$, azon feltétel mellett, hogy a kihívónak $b = 0$ volt a választása, és jelöljük

$Expr(1)$-gyel azt az eseményt, amikor $\hat{b} = 1$ azon feltétel mellett, hogy a kihívó a $b = 1$ értéket választotta.

A továbbiakban a következő függvényt definiáljuk:

$$\mathrm{Adv}^{cpa}_{KEM,\mathbf{A}}(k) = \tfrac{1}{2}|\Pr[Expr(0)] - \Pr[Expr(1)]|,$$

amely alapján a CPA-biztonság értelmezése a következő lesz:

6.3. definíció. *Az új kulcsbeágyazási mechanizmus CPA-biztonságú, ha bármely probabilisztikus, polinomiális idejű* **A** *támadó esetében létezik egy* $f(k)$ *elhanyagolható függvény úgy, hogy*

$$\mathrm{Adv}^{cpa}_{KEM,\mathbf{A}}(k) \leq f(k).$$

Ezek után a következő tételt jelenthetjük ki:

6.2. tétel. *Ha a fenti kísérletben a* G *a 3.2. értelmezés szerint definiált cél-ütközésmentes hash függvény, azaz ha fennáll, hogy bármely probabilisztikus polinomiális* **B** *támadó algoritmus esetén létezik egy* $f_{\mathbf{B}}(k)$ *elhanyagolható függvény úgy, hogy*

$$Adv_{CR1,\mathbf{B}}(G) = Pr[x \xleftarrow{R} \mathbb{QR}_N, y \xleftarrow{R} \mathbf{B}(x) : x \neq y, G(x) = G(y)] \leq f_{\mathbf{B}}(k)$$

és ha a 6.1. értelmezés szerint megadott faktorizációs feltételezést elfogadjuk, akkor létezik egy $f(k)$ *elhanyagolható függvény úgy, hogy*

$$\mathrm{Adv}^{cpa}_{KEM,\mathbf{A}}(k) \leq f(k).$$

Bizonyítás. Minden olyan esetben, amikor az **A** támadó a G hash függvény kiértékelését a K pontra kéri ismernie kell a K értékét, amelyet r^2-ből tud kiszámítani (ez azért igaz, mert r értékét véletlenszerűen határozzuk meg, és mert feltételeztük, hogy G ütközésmentes hash függvény). Ha azonban a fenti esemény nem elhanyagolható valószínűséggel következik be, akkor szintén nem elhanyagolható valószínűséggel következett be, hogy az **A** támadó meghatározta a K értékét K^2 és N ismeretében. Ez viszont akkor következhet be, ha az **A** támadó nem elhanyagolható valószínűséggel oldotta meg a faktorizációs problémát. □

6.2.4. A rendszer hatékonysága

Biztonságos prímek generálása általában időigényes feladat, de gyorsítani lehet az algoritmust a [66] alapján.

A kulcsbeágyazási algoritmus két moduláris négyzetre emelést végez a kulcskibontási algoritmus, pedig két moduláris hatványozást végez.

Következésképpen mindkét algoritmus futási ideje nagyon jó hatékonyságú.

6.3. Az új CCA-biztonságú rendszer

6.3.1. A rendszer leírása

Az új kulcsbeágyazási mechanizmus két cél-ütközésmentes hash függvényt fog használni, jelöljük az egyiket $H : \mathbb{QR}_N^+ \to \{0,1\}^{l_H}$-val, a másikat $G : \mathbb{QR}_N^+ \to \{0,1\}^{l_G}$-vel.

A $\mathbb{QR}_N^+$ halmaznak, amelyet az 5.1. értelmezéssel definiáltunk, több olyan tulajdonsága is van, amely kriptográfiai szempontból hasznos, és amelyekre a további bizonyítások során szükségünk lesz [35]. Ezek a következők:

- $\mathbb{QR}_N^+$ a multiplikatív műveletre nézve ciklikus csoportot alkot, amelynek rendje $\phi(N)/4$, ahol $\phi(N) = (P-1)\cdot(Q-1)$,
- hatékonyan eldönthető, hogy egy elem hozzátartozik a $\mathbb{QR}_N^+$ halmazhoz vagy sem: az adott elemről ellenőrizni kell, hogy benne van-e az $\{1,\ldots,(N-1)/2\}$ halmazban, illetve, hogy a Jacobi szimbóluma $(\mathrm{mod}\ N)$ szerint 1-e.
- a $\mathbb{QR}_N^+$ halmazon belüli kvadratikus maradékok meghatározásának problémájának nehézsége ekvivalens a faktorizációs problémával,
- a $\mathbb{QR}_N^+$ halmazon belüli négyzetreemelés függvény egy permutáció lesz a $\mathbb{QR}_N^+$-ben,
- egy egyenletes eloszlás szerint, véletlenszerűen generált $g \in \mathbb{QR}_N^+$ elem nagy valószínűséggel generátor elem lesz; annak a valószínűsége, hogy a g nem lesz generátor elem, a következő:

$$(p+q-1)/(p\cdot q) \leq 2^{-k+2}.$$

6.1. megjegyzés. *Megjegyezzük, hogy a fenti korlát bizonyításánál fontos szerepet kap, az hogy* $P = 2 \cdot p + 1, Q = 2 \cdot q + 1$, *ahol* p, q *is prímszámok [35] éppen ezért az új CCA-biztonságú rendszer szerkesztéséhez, a fenti módon előállított prímeket kell használni. A szakirodalom, ahogy a 5.1. algoritmusnál is említettük, az* N *számot Blum-egésznek, a* P, Q *prímeket, pedig biztonságos (safe) prímeknek nevezi.*

Az új CCA-biztonságú kulcsbeágyazási mechanizmus, a 4.15. értelmezés szerinti három algoritmusa a következő, ahol a műveleteket $\mathbb{QR}_N^+$-ben végezzük::

A $Gen_{KEM}^{cca}(1^k)$ egy kulcsgeneráló algoritmus, amely egyenletes eloszlás szerint, véletlenszerűen az 1^k bemeneten a következőképpen generálja a pk nyilvános kulcsot és az sk titkos kulcsot:

- generál egy N összetett számot a 6.1. algoritmus szerint,
- generálja g-t és α-t a következőképpen: $g \xleftarrow{R} \mathbb{QR}_N^+$, $\alpha \xleftarrow{R} \{1, \ldots, (N-1)/4\}$, majd meghatározza X-t: $X = (g^{\alpha \cdot 2^{l_H}})^2$
- az algoritmus kimeneti értékei a $pk = (N, g, X)$, és $sk = \alpha$.

Az Enc_{KEM}^{cca} kulcsbeágyazási algoritmus az N, g és X bemeneti értékeken a következőket végzi:

- generálja r-t: $r \xleftarrow{R} \{1, \ldots, (N-1)/4\}$,
- meghatározza: $K = g^{r \cdot 2^{l_H}}$, és $key = G(K)$, ahol $G : \mathbb{QR}_N^+ \to \{0,1\}^{l_G}$ cél-ütközésmentes hash függvény,
- meghatározza $S = (g^h \cdot X)^r$, ahol $h = H(K^2)$, és $H : \mathbb{QR}_N^+ \to \{0,1\}^{l_H}$, cél-ütközésmentes hash függvény,
- meghatározza az $R = K^2$ értéket,
- az algoritmus kimeneti értéke $(key, cipher)$, ahol $cipher = (R, S)$.

A Dec_{KEM}^{cca} kulcskibontási algoritmus az α és $cipher = (R, S)$ bemeneti értékeken a következőket végzi:

- ha a következő tartalmazás nem áll fenn:

$$(R, S) \in \mathbb{QR}_N^+ \times \mathbb{QR}_N^+,$$

akkor az algoritmus REJECT elutasító kimeneti értékkel leáll, ellenkező esetben

- meghatározza a $h = H(R)$ értéket,
- ha a következő egyenlőség nem áll fenn:

$$S^{2^{1+l_H}} = R^{h+\alpha \cdot 2^{1+l_H}}, \tag{6.1}$$

akkor az algoritmus REJECT elutasító kimeneti értékkel leáll, ellenkező esetben

* kiterjesztett euklideszi algoritmussal, meghatározza az a, b, c értékeket a következő diofantikus egyenletből:

$$2^c = a \cdot h + b \cdot 2^{1+l_H},$$

ahol 2^c a h és 2^{1+l_H} legnagyobb közös osztója lesz (a diofantikus egyenlet megoldható lesz, mert fennáll $0 < h < 2^{l_H}$, amiből következik, hogy $c < l_H$),

* az algoritmus kimenete a *key* érték lesz, ahol

$$key = G((S^a \cdot R^{b-a\cdot\alpha})^{2^{l_H - c}}).$$

A Dec^{cca}_{KEM} kulcskibontási algoritmusnak, ha az α és $cipher = (R, S)$ bemeneti értékek mellett megadjuk bemeneti paraméterként a P és Q értékeket is, akkor a (6.1) egyenlőség elvégzése után a *key* értékének, hatékonyabb módon való meghatározása, a következőképpen is történhet:

- a kínai maradéktételt alkalmazva meghatározzuk a K értékét, megoldva a következő kongruencia rendszert:

$$K = R^{(P+1)/4} \pmod{P},$$
$$K = R^{(Q+1)/4} \pmod{Q},$$

- az algoritmus kimeneti értéke a $key = G(K)$ lesz.

6.3.2. A rendszer helyessége

Az új rendszer helyessége néhány elemi számítás során ellenőrizhető.

Egyszerűen belátható, hogy az S és R értékek a következő formában írhatóak fel:

$$\begin{aligned} S &= (g^h \cdot X)^r = g^{r\cdot(h+2\cdot\alpha\cdot 2^{l_H})}, \\ R &= g^{2\cdot r\cdot 2^{l_H}}. \end{aligned}$$

Az (6.1) egyenlőség helyességének a belátásához a következő számítások végezhetőek el:

$$\begin{aligned} S^{2^{1+l_H}} &= g^{2\cdot 2^{l_H}\cdot r\cdot(h+2\cdot\alpha\cdot 2^{l_H})}, \\ R^{h+\alpha\cdot 2^{1+l_H}} &= g^{(h+\alpha\cdot 2\cdot 2^{l_H})\cdot 2\cdot r\cdot 2^{l_H}}. \end{aligned}$$

A kulcs kiszámításának helyessége a következőképpen ellenőrizhető:

$$\begin{aligned} S^a \cdot R^{b-a\cdot\alpha} &= g^{a\cdot r\cdot h+a\cdot r\cdot 2\cdot\alpha\cdot 2^{l_H}} \cdot g^{b\cdot 2\cdot r\cdot 2^{l_H}-a\cdot\alpha\cdot 2\cdot r\cdot 2^{l_H}} \\ &= g^{r\cdot(a\cdot h+b\cdot 2\cdot 2^{l_H})} \\ &= g^{r\cdot 2^c}, \\ (S^a \cdot R^{b-a\cdot\alpha})^{2^{l_H-c}} &= g^{r\cdot 2^{l_H}}. \end{aligned}$$

6.3.3. A rendszer biztonsága

Az új rendszer CCA-biztonságának bizonyításához a következő kísérletet értelmezzük, amely kísérlet egy polinomiális idejű probabilisztikus **A** támadó és a környezete, a kihívó között zajlik:

6.2. kísérlet.

1. *A kihívó futtatja a* $Gen_{KEM}^{cca}(1^k)$ *kulcsgeneráló algoritmust, meghatározza az* N, g, X, α *értékeket, majd az* N, g, X *értékeket, átadja az* **A** *támadónak,*
2. *az* **A** *támadó tetszőleges számú alkalommal meghívja a kihívó* $Dec_{KEM}^{cca}(\alpha, \cdot)$ *és* $G(\cdot)$, $H(\cdot)$ *algoritmusait,*
3. *a kihívó elvégzi a következőket:*
   - *meghatározza* $(key^*, cipher^*) \leftarrow Enc_{KEM}^{cca}(N, g, X)$,
   - *generál egy u értéket:* $u \xleftarrow{R} \mathbb{QR}_N^+$,
   - *a* (key^*)*-t* c_0*-val jelöli,*
   - *a* $(G(u))$*-t* c_1*-gyel jelöli,*
   - *átadja az* **A** *támadónak a* $(c_b, cipher^*)$ *értékpárt, ahol* $b \xleftarrow{R} \{0, 1\}$,

4. *az* **A** *támadó tetszőleges számú alkalommal, különböző cipher értékekre meghívja a kihívó* $Dec^{cca}_{KEM}(\alpha, cipher)$ *és* $G(\cdot)$, $H(\cdot)$ *algoritmusait, azzal a megkötéssel, hogy* $cipher \neq cipher^*$,
5. *az* **A** *támadó meghatároz egy* $\hat{b} \in \{0,1\}$ *kimeneti értéket.*

Jelöljük a továbbiakban $Expr(0)$-val azt az eseményt, amikor a támadó által meghatározott kimeneti érték 1, azon feltétel mellett, hogy a kihívónak $b = 0$ volt a választása, és jelöljük $Expr(1)$-gyel azt az eseményt, amikor a támadó által meghatározott kimeneti érték 1, azzal a feltétellel, hogy a kihívó a $b = 1$ értéket választotta.

A továbbiakban a következő függvényt definiáljuk:

$$\mathrm{Adv}^{cca}_{KEM,\mathbf{A}}(k) = \tfrac{1}{2}|\Pr[Expr(0)] - \Pr[Expr(1)]|,$$

amely alapján a CCA-biztonság értelmezése a következő lesz:

6.4. definíció. *Az új kulcsbeágyazási mechanizmus CCA-biztonságú, ha bármely probabilisztikus, polinomiális idejű* **A** *támadó esetében létezik egy* $f(k)$ *elhanyagolható függvény úgy, hogy*

$$\mathrm{Adv}^{cca}_{KEM,\mathbf{A}}(k) \leq f(k).$$

Ezek után a következő tételt jelenthetjük ki:

6.3. tétel. *Ha a fenti kísérletben a* H *és* G *a cél-ütközésmentes hash függvények, azaz ha fennáll, hogy bármely probabilisztikus polinomiális* $\mathbf{B}_1$ *támadó algoritmus esetén létezik egy* $f_{\mathbf{B}_1}(k)$ *elhanyagolható függvény úgy, hogy*

$$\begin{aligned} Adv_{CR1,\mathbf{B}_1}(H,k) &= Pr[x \xleftarrow{R} \mathbb{QR}_N, y \xleftarrow{R} \mathbf{B}_1(x) : x \neq y, H(x) = H(y)] \\ &\leq f_{\mathbf{B}_1}(k) \end{aligned}$$

és ha fennáll, hogy bármely probabilisztikus polinomiális $\mathbf{B}_2$ *támadó algoritmus esetén létezik egy* $f_{\mathbf{B}_2}(k)$ *elhanyagolható függvény úgy, hogy*

$$\begin{aligned} Adv_{CR1,\mathbf{B}_2}(G,k) &= Pr[x \xleftarrow{R} \mathbb{QR}_N, y \xleftarrow{R} \mathbf{B}_2(x) : x \neq y, G(x) = G(y)] \\ &\leq f_{\mathbf{B}_2}(k) \end{aligned}$$

és ha a 6.1. értelmezés szerint megadott faktorizációs feltételezést elfogadjuk, akkor létezik egy $f(k)$ *elhanyagolható függvény úgy, hogy*

$$\mathrm{Adv}^{cca}_{KEM,A}(k) \leq f(k).$$

Ahhoz, hogy a 6.3. tétel bizonyítását megadhassuk, előbb további két tételt jelentünk ki és bizonyítunk be, a 6.4. és a 6.5. tételt.

6.4. tétel. *Ha elfogadjuk a 6.1. értelmezés szerinti faktorizációs feltételezést, és ha a G hash függvény cél-ütközésmentes tulajdonságú, akkor létezik egy $f(k)$ elhanyagolható függvény úgy, hogy bármely probabilisztikus polinomiális idejű D támadó algoritmus esetén fennáll:*

$$\mathrm{Adv}_{DDS,D}(k) = |\Pr[D(N, K^2, G(K)) = 1] - \Pr[D(N, K^2, G(u)) = 1]| \leq f(k),$$

ahol K a kulcsbeágyazási algoritmus során meghatározott érték és $u \xleftarrow{R} \mathbb{QR}^+_N$.

Informálisan ez azt jelenti, hogy ha szerkesztünk egy olyan D algoritmust, amely különbséget tud tenni az $(N, K^2, G(K))$ és $(N, K^2, G(u))$ számhármasok között, akkor a D algoritmus alkalmas lesz arra, hogy faktorizálja az N értékét.

A továbbiakban ezt a típusú algoritmust D-tulajdonságú algoritmusnak fogjuk hívni.

Bizonyítás. Legyen D egy olyan algoritmus, amelyről feltételezzük, hogy különbséget tud tenni az $(N, K^2, G(K))$ és $(N, K^2, G(u))$ számhármasok között

- bemeneti értékként a D algoritmusnak megadjuk az (N, K^2, V) számhármast, ahol V vagy a $G(K)$ vagy $G(u)$ értéket veszi fel, és ahol $u \xleftarrow{R} \mathbb{QR}^+_N$,
- a D algoritmus több, egyenletes eloszlás szerint véletlenszerűen generált bemeneti értékekre meghatározza a G hash függvény kimenetét,
- minden olyan esetben, amikor a D algoritmus a G függvény kiértékelését a K pontban végzi ismernie kell a K értékét, amelyet K^2-ből tud kiszámítani (ez azért igaz, mert feltételeztük, hogy G ütközésmentes hash függvény),

- ha azonban a fenti esemény nem elhanyagolható valószínűséggel következik be, akkor szintén nem elhanyagolható valószínűséggel következett be az az esemény is, hogy a D algoritmus meghatározta a K értékét K^2 és N ismeretében,
- ha a D algoritmus nem elhanyagolható valószínűséggel határozta meg a K értékét a K^2 és N ismeretében, akkor a D algoritmus nem elhanyagolható valószínűséggel oldotta meg a faktorizációs problémát, ami viszont ellentmond a faktorizációs feltételezésnek.

A fenti gondolatmenet alapján kijelenthető, hogy:

$$\mathrm{Adv}_{DDS,D}(k) \leq \mathrm{Adv}_{FAC,\mathbf{A}}(k) + \mathrm{Adv}_{CR1,\mathbf{B}}(k).$$

azaz a $\mathrm{Adv}_{DDS,D}(k)$ függvény k szerint elhanyagolható. □

6.5. tétel. *Ha* **A** *egy probabilisztikus, polinomiális idejű támadó algoritmus, amely feltöri az új rendszer CCA-biztonságát, akkor szerkeszthető egy probabilisztikus polinomiális idejű D algoritmus, amely D-tulajdonságú vagy szerkeszthető egy probabilisztikus polinomiális idejű B algoritmus, amely feltöri a H függvény ütközésmentes tulajdonságát.*

Formálisan ez a következőket jelenti:

$$\mathrm{Adv}^{cca}_{KEM,A}(k) \leq \mathrm{Adv}_{DDS,D}(k) + \mathrm{Adv}_{CR1,B}(k) + f(k),$$

ahol $f(k)$ elhanyagolható függvény.

Bizonyítás. A tétel bebizonyítása érdekében a következőképpen járunk el: szerkesztünk egy D algoritmust, amely szimulálja az **A** támadó bemenetét.

Tudjuk, hogy a D algoritmus bemenete vagy $(N, K^2, G(K))$ vagy $(N, K^2, G(u)))$, ahol a D algoritmus meghívja az **A** algoritmust a következő bemenetekkel:

- az (N, g, X) nyilvános kulccsal, és a
- $(key^*, cipher^*)$ bemenettel, ha a D bemenete $(N, K^2, G(K))$ vagy a $(G(u), cipher^*)$ bemenettel, ha a D bemenete $(N, K^2, G(u))$, ahol a $g, X, key^*, cipher^*$ értékeket a D határozta meg, a következőképpen:
  - $g \overset{R}{\leftarrow} \mathbb{QR}^+_N$, és $\beta \overset{R}{\leftarrow} \{1, \ldots (N-1)/4)\}$,

– $h^* = H(K^2)$, $X = g^{\beta \cdot 2 \cdot 2^{l_H} - h^*}$,
– $cipher^* = (R^*, S^*)$, ahol $R^* = K^2$, és $S^* = (K^2)^\beta$,
– $key^* = G(K)$.

Mivel $g \in \mathbb{QR}_N^+$ nagy valószínűséggel generátor elem, a 6.3.1. szakaszban megadott tulajdonságok szerint, feltételezhetjük, hogy:

- X felírható mint: $(g^{\alpha \cdot 2^{l_H}})^2$, ahol α ismeretlen, de α felírható $\beta - h^*/(2 \cdot 2^{l_H})$ alakba,
- R^* felírható mint: $g^{r^* \cdot 2 \cdot 2^{l_H}}$, ahol r^* ismeretlen,
- S^* felírható mint: $(g^{h^*} \cdot X)^{r^*}$, mert fennáll:

$$(g^{h^*} \cdot X)^{r^*} = (g^{h^*} \cdot g^{\beta \cdot 2 \cdot 2^{l_H} - h^*})^{r^*}$$

- key^* felírható mint: $g^{r^* \cdot 2^{l_H}}$.

Amikor az **A** elküldi a $cipher = (R, S)$ értékét D-nek, hogy D meghatározza a key értékét, D le tudja ellenőrizni, hogy az $cipher$ érték konzisztens-e vagy sem, elvégezve a következő egyszerű számításokat:

$$S^{2 \cdot 2^{l_H}} = R^{h - h^* + \beta \cdot 2 \cdot 2^{l_H}},$$

ahol a h értéke meghatározható, mert $h = H(R)$.

Ha az (R, S) érték pár nem konzisztens, akkor REJECT visszautasító kimeneti értékkel leáll.

Ha az (R, S) érték pár konzisztens, akkor D két különböző esetet tud megkülönböztetni:

1. $h \neq h^*$. Ebben az esetben a következő diofantikus egyenletből, a kiterjesztett euklideszi algoritmussal, D megtudja határozni az a_1, b_1, c_1 értékeket:

$$2^{c_1} = a_1 \cdot (h - h^*) + b_1 \cdot 2 \cdot 2^{l_H}.$$

   A key meghatározását, pedig a következőképpen végzi:

$$G(S^{a_1} \cdot R^{b_1 - a_1 \cdot \beta})^{2^{l_H - c_1}}.$$

2. $h = h^*$. Ebben az esetben további két alesetet különböztetünk meg:

- Ha fennáll a következő egyenlőség: $R = R^*$, akkor igaz az is hogy $S = S^*$, viszont nem megengedett az (R^*, S^*) bemenet.
- Ha fennáll $R \neq R^*$, akkor a D algoritmus ütközést talált, ami csak kis valószínűséggel következhet be.

Következésképpen a $pk = (N, g, X)$, és (R^*, S^*) értékek eloszlása abban az esetben, amikor D algoritmus szimulálja az $\mathbf{A}$ bemenetét, majdnem mindig ugyanolyan eloszlású, mint a 6.2. kísérlet esetében. Annak a valószínűsége, hogy nem ugyanolyan eloszlásúak a pk, és (R^*, S^*) értékek a következő: 2^{-k+3} [35].

Ezek alapján kijelenthető, hogy:

$$|\Pr[D(N, K^2, G(K)) = 1] - \Pr[Expr(0)]| \leq \mathrm{Adv}_{CR1,B}(k) + 2^{-k+3},$$

$$|\Pr[D(N, K^2, G(u)) = 1] - \Pr[Expr(1)]| \leq \mathrm{Adv}_{CR1,B}(k) + 2^{-k+3},$$

ahol $Expr(0)$ és $Expr(1)$ ugyanazokat az eseményeket jelölik, mint a 6.2. kísérletben.

Ebből következik:

$$\begin{aligned}
\mathrm{Adv}^{cca}_{KEM,A}(k) &= \tfrac{1}{2}|\Pr[Expr(0)] - \Pr[Expr(1)]| \\
&\leq \tfrac{1}{2}(\mathrm{Adv}_{DDS,D}(k) + \\
&+|\Pr[D(N, K^2, G(K)) = 1] - \Pr[Expr(0)]| + \\
&+|\Pr[D(N, K^2, u) = 1] - \Pr[Expr(1)]|) \\
&\leq \mathrm{Adv}_{DDS,D}(k) + \mathrm{Adv}_{CR1,B}(k) + 2^{-k+3}.
\end{aligned}$$

Korábban azonban bebizonyítottuk, hogy:

- $\mathrm{Adv}_{CR1,B}(k)$ elhanyagolható k szerint,
- $\mathrm{Adv}_{DDS,D}(k)$ elhanyagolható k szerint.

Ezeknek következményeként, a 6.4. értelmezésnek megfelelően az új kulcsbeágyazási mechanizmus CCA-biztonságú. □

6.3.4. A rendszer hatékonysága

Biztonságos prímek generálása általában időigényes feladat, de gyorsítani lehet az algoritmust [66] alapján.

A kulcsbeágyazási algoritmus két moduláris hatványozást végez, nagy hatványkitevővel és néhány szorzást, illetve hatványozást kis hatványkitevővel.

A kulcskibontási algoritmus egy moduláris hatványozást végez nagy hatványkitevővel és hasonlóan a kulcsbeágyazási algoritmushoz néhány szorzást, illetve hatványozást kis hatványkitevővel. Az algoritmus futási ideje gyorsítható a kínai maradéktétel alkalmazásával, ha megadjuk a P és Q prímeket, mint bemeneti értékeket az algoritmusnak.

Következésképpen a kulcskibontási algoritmus majdnem kétszer olyan gyors, mint a kulcsbeágyazási algoritmus, ami nagyon alkalmassá teszi az algoritmust olyan alkalmazások számára, ahol a szerver végzi a kulcskibontási folyamatot (mint ahogyan az általában történni szokott).

7. fejezet

Kulcsbeágyazási mechanizmusok implementációja

Ebben a fejezetben több CCA-biztonságú kulcsbeágyazási algoritmus implementációját mutatjuk be, C++ programozási környezetben, illetve összehasonlítjuk ezen rendszerek hatékonyságát, futási időket mérve a kulcsgenerálás, kulcsbeágyazási és kulcskibontási algoritmusok esetében. Konkrétan

- a Cramer–Shoup, a továbbiakban mint CS-rendszer,
- az RSA–OAEP,
- az SAEP,
- a Hofheinz- és társai, a továbbiakban mint HKS-rendszer,
- a 6. a fejezetben bemutatott kulcsbeágyazási mechanizmus implementációjára térünk ki.

Kriptográfia rendszerek implementációjával, pontosabban az RSA-titkosító implementációjával funkcionális programozási nyelvben a [46] cikkben is foglalkoztunk.

Az eredményeket amelyeket itt teszünk közzé a következő konfigurációjú géppel mértük, ahol a rendszer hardverkonfigurációja a következő volt:

processzor: Intel(R)CoreTM i3-2310M 2.10GHz,
a rendszer típusa: 64-bites operációs rendszer,
RAM memóriaméret: 4.00GB.

A rendszer szoftverkonfigurációja:

operációs rendszer: Windows 7 Enterprise, Service Pack 1,
programozási környezet: Visual Studio 2010 Professional, Microsoft Visual C++ 2010.

Az implementáció során a következő nyílt forráskódú programcsomagokat, könyvtárcsomagokat használtuk:

- NTL, static library [58],
- SHA1, SHA2, HMAC and Key Derivation in C [31],
- Crypto++ [23].

A nagy számok ábrázolására és a velük végzett aritmetikai műveletek elvégzésére, a Shoup által fejlesztett, NTL nyílt forráskódú könyvtárcsomagot használtuk. Az NTL magas hatékonyságú C++ könyvtárcsomag, amely lehetővé teszi különböző adatszerkezetek és algoritmusok alkalmazását tetszőleges nagyságrendű egész számok, vektorok, mátrixok, polinomok esetében. A könyvtárcsomag könnyedén installálható Unix, Windows és Macintosh gépekre egyaránt. A tetszőleges hosszúságú, előjeles nagy számok ábrázolására használt osztály a ZZ nevet viseli. Az osztály keretén belül alkalmazhatóak az alapvető aritmetikai műveletek, de a kriptográfiában hasznos, komplexebb függvények egy része is implementálva van, pl: prímtesztelés, moduláris hatványozás, moduláris inverz meghatározás, stb.

A rendszerek implementálása során az SHA-256, illetve SHA-384 hash függvények kerültek alkalmazásra. A biztonság növelése érdekében ezen könnyedén lehet változtatni, nagyobb kimeneti értéket előállító SHA hash függvényt alkalmazva. Megjegyezzük, hogy a jelenlegi standard követelmények a 256, illetve 384 bites kimenti hash értékeket megfelelőnek tartják.

A továbbiakban minden egyes rendszernél külön feltüntetjük, hogy milyen nagyságrendű kimeneti hash értékekkel dolgoztunk, azaz mikor alkalmaztuk az SHA-256, és mikor az SHA-384-es hash függvényeket.

A hash függvény nyílt forráskódja B. Gladman munkája, a hash függvények hatékonysága pedig a fejlesztő nyilvános weboldalán találhatóak meg [31].

A Cryptoo++ kriptográfiai sémák implementálását tartalmazó C++ osztálykönyvtár. A valós alkalmazások esetében használt kriptográfiai primitívek szinte mindegyikét implementálták a fejlesztők. Wei Dai kezdte el, napjainkban azonban többen is bekapcsolódtak a fejlesztésébe. Több platformon futtatható, pl. MSVC 6.0-2012, GCC 3.3 - 4.7, stb. Az osztálykönyvtár több verzióját szabványként fogadta el a NIST. Az általunk implementált rendszerek esetében az osztálykönyvtár primitívei közül az RSA–OAEP implementációt használtuk fel.

A Cramer–Shoup-rendszer

A CS rendszer implementálása során a $\mathbb{Z}_p^*$ ciklikus csoporttal dolgoztunk, ahol a p prímszám, alakja $2 \cdot q+1$, ahol q szintén prímszám. Ez a kitétel a generátor elemek meghatározásánál is fontos szerepet kap. A szakirodalom a p prímet biztonságos (safe) prímnek hívja, a q prímet pedig Sophie–Germain-prímnek. Az NTL könyvtárcsomag több olyan függvénnyel rendelkezik, amely prímeket generál, A következő függvény, amelynek három alakja is használható, egy l hosszúságú Sophie–Germain prímet generál, ahol annak a valószínűsége, hogy sem a q, sem a $2 \cdot q+1$ szám nem lesz összetett kisebb, mint 2^{-err}:

```
void GenGermainPrime(ZZ& q, long l, long err = 80);
ZZ GenGermainPrime_ZZ(long l, long err = 80);
long GenGermainPrime_long(long l, long err = 80);
```

Biztonságos prímek generálása időigényes feladat, ahogy ez a mérési eredményekből is leolvasható, de számos rendszer biztonsága érdekében ez fontos követelmény. A kulcsgenerálás időigényét még az is lassítja, hogy a titkosítás során szükség van két generátor elemre. A $\mathbb{Z}_p^*$ ciklikus csoportban a generátor elem meghatározását a következő tulajdonságot figyelembe véve végeztük: egy $g \neq \pm 1 \pmod p$ szám akkor és csakis akkor lesz generátor elem $\pmod p$ szerint, ha fennáll a következő egyenlőség ([61], [59]):

$$g^q \neq 1 \pmod p.$$

A két generátor elem meghatározása érdekében el kell tehát végezni egy-egy moduláris hatványozást.

A rendszerre vonatkozó mérési eredmények a 7.1. táblázatban találhatóak, ahol a q nagyságrendje a kulcsméret oszlopban feltüntetett érték. A rendszer implementálása során az NTL könyvtárcsomagot és B. Gladman SHA-256 hash függvény implementációját használtuk fel.

A nyílt szöveg egy véletlenszerűen generált p-nél kisebb pozitív egész szám lesz.

Az RSA–OAEP-rendszer

Az RSA–OAEP implementáció, a standardként elismert Crypto++ könyvtárcsomag része. A kulcsgenerálás egy $n = p \cdot q$ összetett számot határoz meg, ahol p, q nem lesz biztonságos-prím, éppen ezért ennél az implementációnál a kulcsgenerálás ideje jóval kisebb nagyságrendű.

A rendszerre vonatkozó mérési eredmények a 7.2. táblázatból olvashatóak le.

A nyílt szöveg egy véletlenszerűen generált, 256 bites pozitív egész szám lesz.

Az SAEP rendszer

A kulcsgenerálás egy $n = p \cdot q$ összetett számot határoz meg, ahol p, q nem lesz biztonságos prím. A rendszer azonban előírja, hogy a p, q prímekre fennálljon: $p \equiv q \equiv 3 \pmod 4$, illetve, hogy az $n = p \cdot q$ összetett szám legfelsőbb helyértékű két bitje 1, illetve 0 legyen. Ezek természetesen lassítják a kulcsgenerálás időigényét.

A rendszer implementálása során az NTL könyvtárcsomagot és B. Gladman SHA-384 hash függvény implementációját használtuk fel. A rendszerre vonatkozó mérési eredmények a 7.3. táblázatban találhatóak.

A nyílt szöveg egy véletlenszerűen generált, 256 bites pozitív, egész szám lesz. A véletlenszerűen generált r érték (ahol az r szerepét a 5.2. fejezetben megadott SAEP rendszernél specifikáltuk) nagyságrendje aszerint fog változni, hogy mekkora k értékkel (kulcs mérettel) dolgozik az algoritmus. Fennáll, hogy $r = k - 384$, ahol a k a 7.3. táblázatban megadott kulcsméret.

A HKS rendszer

A kulcsgenerálás egy $N = P \cdot Q$ összetett számot határoz meg, ahol P, Q biztonságos prímek lesznek, azaz fennáll: $P = 2 \cdot p + 1, Q = 2 \cdot q + 1$ és p, q is prímszámok. A rendszer azt is előírja, hogy a P, Q prímekre fennálljon: $P \equiv Q \equiv 3 \pmod 4$.

A kulcsgenerálás időigényét lassítja az is, hogy a P, Q értékek mellett további két értéket kell meghatározni, az X-et és az α-t, ahol az X esetében szükséges egy moduláris hatványozás elvégzése.

A rendszer implementálása során az NTL könyvtárcsomagot és B. Gladman SHA-256 hash függvény implementációját használtuk fel. A rendszerre vonatkozó mérési eredmények a 7.4. táblázatban találhatóak.

A kulcsbeágyazási algoritmus egy 256 bites hosszúságú kulcsot fog meghatározni.

Az új rendszer

A kulcsgenerálás egy $N = P \cdot Q$ összetett számot határoz meg, ahol P, Q biztonságos prímek lesznek, azaz fennáll: $P = 2 \cdot p + 1, Q = 2 \cdot q + 1$ és p, q is prímszámok. A rendszer azt is előírja, hogy a P, Q prímekre fennálljon: $P \equiv Q \equiv 3 \pmod 4$.

A kulcsgenerálás időigényét, az N értékének a meghatározása mellett az is befolyásolja, hogy szükséges további két érték meghatározása: az X és az α előállítása. Az X esetében szükséges egy moduláris hatványozás elvégzése, kisebb hatványkitevővel, mint a HKS rendszer esetében.

A titkosítás és visszafejtés időigényét a moduláris hatványozáshoz szükséges exponens nagyságrendje határozza meg, ami az új rendszer esetében kisebb nagyságrendű lesz, mint a HKS rendszer esetében.

Az új rendszer implementálása során az NTL könyvtárcsomagot és B. Gladman SHA-256 hash függvény implementációját használtuk fel. A rendszerre vonatkozó mérési eredmények a 7.5. táblázatban találhatóak.

A kulcsbeágyazási algoritmus egy 256 bites hosszúságú kulcsot fog meghatározni.

A 7.1, 7.2, 7.3, 7.4, 7.5. táblázatokban található mérési eredmények 20 bemeneti adat átlagának az értékét mutatják, ahol a kulcsméret bitben, a kulcsgenerálás, a titkosítás és a visszafejtés ideje másodpercben került meghatározásra.

7.1. táblázat. A Cramer–Shoup-rendszer mérési eredményei

kulcsméret	kulcsgenerálás	titkosítás	visszafejtés
256	0.21	0	0.01
512	4.30	0.04	0.05
1024	50.47	0.24	0.26

7.2. táblázat. Az RSA–OAEP-rendszer mérési eredményei

kulcsméret	kulcsgenerálás	titkosítás	visszafejtés
512	0.02	0	0
1024	0.07	0	0
2048	0.31	0	0.01
4096	2.55	0	0.04

7.3. táblázat. Az SAEP-rendszer mérési eredményei

kulcsméret	kulcsgenerálás	titkosítás	visszafejtés
512	0.15	0	0
1024	2.07	0	0.01
2048	11.30	0	0.09

7.4. táblázat. A HKS-rendszer mérési eredményei

kulcsméret	kulcsgenerálás	titkosítás	visszafejtés
512	0.40	0.04	0.02
1024	8.07	0.11	0.11
2048	131.33	0.68	0.53

7.5. táblázat. Az új CCA-biztonságú rendszer mérési eredményei

kulcsméret	kulcsgenerálás	titkosítás	visszafejtés
512	0.39	0.02	0.01
1024	8.13	0.09	0.06
2048	125.59	0.66	0.44

8. fejezet

Összefoglaló

Jelen értekezés a titkosító rendszerek választott nyílt szöveg alapú támadással (chosen plaintext attack), illetve választott rejtjelezett szöveg alapú támadással (chosen ciphertext attack) szembeni biztonságát tárgyalta.

A választott téma aktualitását az adja, hogy a létező titkosító rendszerek átalakítása oly módon, hogy azok ellenálljanak a választott rejtjelezett szöveg típusú támadásnak a jelen kriptográfia aktív kutatási területét képezik ([6], [7], [13], [20], [35]).

Az értekezés első felében megadtuk a témához kapcsolódó értelmezések és tételek formális definícióit, és mivel a tárgyalt fogalmakhoz nincsen széles körben elfogadott magyar terminológia fokozottan oda figyeltünk ezek precíz, magyar megfogalmazására. Konkrétan a következő fogalmakkal foglalkoztunk:

- a szimmetrikus titkosítási rendszerek matematikai modelljével, a passzív és aktív támadóval szembeni biztonság fogalmakkal, ahol az aktív támadóval szembeni biztonság esetében különbséget tettünk a választott nyílt-szöveg alapú, illetve a választott rejtjelezett-szöveg alapú támadások között,
- a hash függvényekkel és üzenethitelesítő kódokkal szembeni követelményekkel, kihangsúlyozva az univerzális hash függvények (universal hash function), a cél ütközésmentes hash függvények (target

collision-resistant hash function) és az ütközésmentes hash függvények (collision-resistant hash function) közötti különbségeket,
- az aszimmetrikus titkosító rendszerek matematikai modelljével, a választott nyílt-szöveg alapú, illetve a választott rejtjelezett-szöveg alapú támadáshoz kapcsolódó biztonságértelmezésekkel,
- a hibrid rendszerek matematikai modelljével és a választott rejtjelezett-szöveg alapú támadáshoz kapcsolódó biztonság értelmezésével.

Az értekezés további részében azokat a matematikai problémákat tárgyaltuk amelyek alapjául szolgálnak a mai biztonságos kriptorendszereknek, és amelyeket a szakirodalomban feltételezések formájában fogalmaztak meg. Jelen értekezésben a következő feltételezések formális definícióit adtuk meg ([18], [24], [36]):

- faktorizációs feltételezés,
- RSA feltételezés,
- kvadratikus maradék feltételezés,
- diszkrét logaritmus feltételezés,
- döntési Diffie–Hellman-feltételezés.

Ezek után megadtunk egy új kulcsbeágyazási mechanizmus CPA, illetve CCA-biztonságú verziót. Megjegyezzük, hogy az új kulcsbeágyazási mechanizmus alapjául a Rabin-titkosító [54] és a Hofheinz és társai [35], által szerkesztett rendszerek szolgáltak. A CCA-biztonságú kulcsbeágyazási mechanizmus a [48] cikkben került publikálásra.

Az értekezés keretén belül megadtuk az új kulcsbeágyazási mechanizmus specifikációját, helyességét, hatékonyságát, illetve bizonyítottuk a rendszer biztonságát. Kitértünk még a szakirodalom által számon tartott CCA-biztonságú kulcsbeágyazási mechanizmusok specifikációira, összehasonlítva ezen rendszereket, hatékonyság szempontjából, az új kulcs beágyazási mechanizmus hatékonyságával. Megadtuk C++ programozási környezetben a rendszerek mindegyikének implementációját, futási időket mérve a megfelelő algoritmusok esetében.

Az implementáció kódja letölthető a következő címről: `http://www.ms.sapientia.ro/~mgyongyi/PhD/PhdSource.zip`

A tárgyalt rendszerek a következőek voltak, ahol vizsgáltuk az eredeti titkosító rendszert, és annak CCA-biztonságú verzióját:

- az ElGamal- és Cramer–Shoup-rendszer ([27], [20]),
- az RSA- és RSA–OAEP-rendszer ([55], [6]),
- a Rabin- és SAEP-rendszer ([54], [14]),
- a Blum–Goldwasser és Hofheinz- és társai rendszere ([12], [35]).

Megjegyezzük, hogy:

- A Cramer–Shoup-rendszer biztonsága a döntési Diffie–Hellman-feltételezésen és az alkalmazott hash függvény ütközésmentes tulajdonságán alapszik.
- Az RSA–OAEP-rendszer biztonsága az RSA feltételezésen és az alkalmazott véletlenszerű hash függvény ütközésmentes tulajdonságán alapszik.
- Az SAEP-rendszer biztonsága a faktorizációs feltételezésen és az alkalmazott véletlenszerű hash függvény ütközésmentes tulajdonságán alapszik.
- A Hofheinz- és társai kulcsbeágyazási mechanizmusa a faktorizációs feltételezésen és az alkalmazott hash függvény ütközésmentes tulajdonságán alapszik.
- Az új kulcsbeágyazási mechanizmus biztonsága a faktorizációs feltételezésen és az alkalmazott hash függvény ütközésmentes tulajdonságán alapszik.

Az új kulcsbeágyazási mechanizmus biztonságát, az újabb szakirodalomban használt bizonyítási módszer alapján adtuk meg, amely szerint definiálni kell egy kísérletet, amely egy polinomiális idejű, probabilisztikus támadó és a környezete, a kihívó között játszódik le. A CCA-biztonság a kísérlet lehetséges kimeneti értékeinek a vizsgálatával bizonyítható.

Következtetésképpen az új kulcsbeágyazási mechanizmusról elmondhatjuk, hogy biztonságos, hatékonyság szempontjából pedig nem marad alul a jelenleg standardként használt rendszerekhez képest.

Fontosnak tartjuk azt is megemlíteni, hogy az új kulcsbeágyazási mechanizmussal szélesítettük a CCA-biztonságú kulcsbeágyazási mechanizmusok terét.

9. fejezet

Summary

This thesis examines the security of encryption systems, it focuses mainly on two types of attacks on chosen plaintext attack and chosen ciphertext attack.

The importance of the chosen topic relies in the fact that transforming encryption systems so as to be secure against chosen ciphertext attack in cryptography is an active research field ([6], [7], [13], [20], [35]).

In the first half of the the thesis we give the formal definitions of the concepts used in the dissertation. Namely we introduce the followings:

- the mathematical model of symmetric encryption systems, the formal definition of security against passive and active attack where, in the case of active attack, we make a distinction between chosen plaintext attack and chosen ciphertext attack,
- the hash functions and message authentication codes and the requirements on them, highlighting the differences between universal hash function, target collision resistant hash function and collision resistant function,
- the mathematical model of asymmetric encryption systems, the formal definition of security against chosen plaintext attack and chosen ciphertext attack,
- the mathematical model of hybrid encryption systems, the formal definition of security against chosen ciphertext attack.

We mention that in most of the protocols asymmetric encryptions are used to generate and interchange the keys of two remote parties. And for the encryption of actual data symmetric encryption technics are used. The newer literature introduces key-encapsulation, respectively the data-encapsulation terms for these.

In the next chapter we introduce the mathematical problems that are the basis of many cryptosystems and are known in cryptography as basic assumptions. We give the formal definition of the following basic assumptions ([18], [24], [36]):

- factorization assumption,
- RSA assumption,
- quadratic residuosity assumption,
- discrete logarithm assumption,
- decisional Diffie–Hellman-assumption.

Consequently we describe the main result of the dissertation, the CPA and CCA version of a new key encapsulation mechanism. We mention that the mechanism is based on Rabin-encryption [54] and on the system constructed by Hofheinz et al [35]. The new CCA-secure key encapsulation mechanism was published in [48].

Within the framework of the thesis we give a detailed description of the proposed mechanism, we present its soundness and efficiency as well as the proof of security. We present some CCA-secure key encapsulation mechanisms, which are considered important in the recent area of cryptography, and we compare these systems with the new key encapsulation mechanism. We give the C++ implementations of the systems and we measure the running times of the corresponding algorithms.

The implementation can be downloaded from the following address: `http://www.ms.sapientia.ro/~mgyongyi/PhD/PhdSource.zip`

The asymmetric encryption systems and the corresponding CCA-secure versions of them implemented are the following:

- the ElGamal, and Cramer–Shoup system ([27], [20]),
- the RSA and RSA–OAEP system ([55], [6]),
- the Rabin and SAEP system ([54], [14]),

- the Blum–Goldwasser and HKS system ([10], [35]).

We mention the following:

- The security of the Cramer–Shoup-cryptosystem relies in the decisional Diffie–Hellman-assumption and the collision-resistant properties of the underlying hash function.
- The security of the RSA–OAEP-scheme relies in the RSA assumption and in the security of hash functions maintained as a random oracle.
- The security of the SAEP-system depends on the factorization assumption and on the security of hash functions maintained as a random oracle.
- The security of the HKS key encapsulation mechanism relies in the factorization assumption and the underlying target collision-resistant hash function.
- The security of the new key encapsulation mechanism relies in the factorization assumption and the underlying target collision-resistant hash function.

The security proof of the new key encapsulation mechanism given in the thesis is based on the proof technic used in the recent literature. So we define an experiment going on between a probabilistic polynomial time adversary and his environment, called a challenger. The CCA-security can be proved by examining the possible outputs of the experiment.

Consequently, the new key encapsulation mechanism is secure and in terms of efficiency does not lag behind the systems accepted as standard.

It is important to mention that the new key encapsulation mechanism broadens the space of CCA-secure key encapsulation mechanisms.

Irodalomjegyzék

[1] M. Abdalla, M. Bellare, and P. Rogaway. The Oracle Diffie-Hellman Assumptions and an Analysis of DHIES. *Topics in Cryptology - CT-RSA 2001, LNCS 2020*, pages 143–158, 2001.

[2] M. Ajtai and C. Dwork. A public-key cryptosystem with worts-case/awerage-case equivalence. *In STOC '97 (El Paso, TX), ACM (electronic)*, pages 284–293, 1999.

[3] A. Bege and Z. Kása. *Algoritmikus kombinatorika és számelmélet.* Presa Universitară Clujeană, 2006.

[4] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. *Advances in Cryptology, CRYPTO '96*, 1996.

[5] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. *Proceedings of 1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[6] M. Bellare and P. Rogaway. Optimal Asymmetric Encryption. *Advances in Cryptology, EUROCRYPT '94*, pages 92–111, 1994.

[7] M. Bellare and P. Rogaway. *Introduction to Modern Cryptography.* 2005.

[8] D.J. Bernstein. The salsa20 family of stream ciphers. http://cr.yp.to/snuffle/salsafamily-20071225.pdf.

[9] D. Bleichenbacher. Chosen Ciphertext Attacks Against Protocols Based on the RSA encryption Standard PKCS#1. *Advances in Cryptology, Proceedings of CRYPTO '98*, pages 1–12, 1998.

[10] L. Blum, M. Blum, and M. Shub. Comparison of two pseudo-random number generators. *Advances in Cryptology, Proceedings of CRYPTO '82*, pages 61–78, 1982.

[11] M. Blum, P. Feldman, and S. Micali. Proving Security Against Chosen Ciphertext Attacks. *Advances in Cryptology, Proceedings of CRYPTO '88*, pages 256–268, 1984.

[12] M. Blum and S. Goldwasser. An effcient probabilistic public-key encryption scheme which hides all partial information. *Advances in Cryptology, Proceedings of CRYPTO '84*, pages 289–302, 1985.

[13] D. Boneh. Twenty Years of attacks on the RSA cryptosystem. *Notices of the American Mathematical Society*, 46 (2):203–213, 1999.

[14] D. Boneh. Simplified OAEP for the RSA and Rabin functions. *Lecture Notes in Computer Science, Proceedings of CRYPTO '01*, 213:275–291, 2001.

[15] A. Bérczes, J. Folláth, and A. Pethő. On a family of collision-free functions. *Tatra Mountains Mathematical Publications*, 47:1–13, 2010.

[16] A. Bérczes, J. Ködmön, and A. Pethő. A one-way function based on norm form equations. *Periodica Mathematica Hungarica*, 49:1–13, 2004.

[17] J.A. Buchmann. *Introduction to cryptography.* Springer, 2002.

[18] L. Buttyán and T. Vajda. *Kriptográfia és alkalmazásai.* Typotex, 2004.

[19] D. Coppersmith. Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities. *Journal of Cryptology*, 10 (4):233–260, 1997.

[20] R. Cramer and V. Shoup. A practical practical public-key cryptosystem provably secure against adaptive chosen ciphertext attack. *Advaces in Cryptology, Proceedings of CRYPTO '98*, 33:13–25, 1998.

[21] R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal of Computing*, pages 167–226, 2003.

[22] J. Daemen and V. Rijmen. The Block Cipher Rijndael. *Proceedings of the The International Conference on Smart Card Research and Applications*, pages 277–284, 2000.

[23] W. Dai. Crypto++ Library. http://www.cryptopp.com/.

[24] H. Delfs and H. Knebl. *Introduction to cryptography. Principles and Applications.* Springer, 2002.

[25] W. Diffie and M.E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT–22:644–654, 1976.

[26] Y. Dodis and J. Katz. Chosen-Ciphertext Security of Multiple Encryption. 3378:188–209, 2005.

[27] T. ElGamal. A Public-Key Cryptosystem and a Signatures Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, IT–31:469–472, 1985.

[28] S.R. Fluhrer, I. Mantin, and A. Shamir. Weaknesses in the Key Scheduling Algorithm of RC4. *Proceeding SAC '01 Revised Papers from the 8th Annual International Workshop on Selected Areas in Cryptography*, pages 1–24, 2001.

[29] J. Folláth, A. Huszti, and A. Pethő. *Informatikai biztonság és kriptográfia.* 2010.
http://www.inf.unideb.hu/∼pethoe/Jegyzet_PA_20110508.pdf.

[30] R. Freud and E. Gyarmati. *Számelmélet.* Nemzeti Tankönyvkiadó, 2000.

[31] B. Gladman. SHA1, SHA2, HMAC and Key Derivation in C. http://gladman.plushost.co.uk/oldsite/cryptography_technology/sha.

[32] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[33] L. Goubin, C. Mauduit, and A. Sárközy. Construction of large families of pseudorandom binary sequences. *J. Number Theory*, 106:59–69, 2004.

[34] J. Hastad. On using RSA with Low Exponent in a Public Key Network. *CRYPTO '85 Advances in Cryptology*, pages 403–408, 1985.

[35] D. Hofheinz, E. Kiltz, and V. Shoup. Practical Chosen Ciphertext Secure Encryption from Factoring. *Journal of Cryptology, Online First^{TM}*, 2011.

[36] J. Katz and Y. Lindell. *Introduction to Modern Cryptograhy*. Chapman & HallCRC Press, 2008.

[37] A. Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, IX:5–38, 1883.

[38] N. Koblitz and A. J. Menezes. A survey of public-key cryptosystems. *SIAM Review*, 46:599–634, 2004.

[39] L. Lovász. *Algoritmusok bonyolultsága*. Nemzeti Tankönyvkiadó, 2001.

[40] A. J. Menezes, P. C. van Oorschot, and S.A. Vanstone. *Handbook of applied cryptography*. CRC Press, Boca Raton, Florida, 1997.

[41] R. C. Merkle. A fast software one-way hash function. *Journal of Cryptology*, 3:43–48, 1990.

[42] R. C. Merkle and M. Hellman. Hiding information and signatures in trapdoor knapsacks. *Information Theory, IEEE Transactions*, 24 (5):525–530, 1978.

[43] S. Micali, C. Rackoff, and B. Sloan. The notion of security for probabilistic cryptosystems. *SIAM Journal on Computing*, pages 412–426, 1988.

[44] Gy. Márton. *Kriptográfiai alapismeretek*. Scientia Kiadó, 2008.

[45] Gy. Márton. Nyilvános kulcsú kriptorendszerek biztonságához kapcsolodó értelmezések. *Számokt 2010 konferenciakötet, 20. Nemzetközi számítástechnika és oktatás konferencia*, pages 183–187, 2010.

[46] Gy. Márton. Public-key cryptography in functional programming context. *Acta Universitatis Sapientiae, Informatica*, 2:99–111, 2010.

[47] Gy. Márton. Üzenethitelesítő kódok és a CCA támadás. *Számokt 2011 konferenciakötet, 21. Nemzetközi számítástechnika és oktatás konferencia*, pages 229–233, 2011.

[48] Gy. Márton. CCA-secure key-encapsulation mechanism based on the factoring assumption. *Tatra Mountains Mathematical Publications*, 53:137–146, 2012.

[49] Gy. Márton. A faktorizációs problémán alapuló titkosító rendszerek biztonsága. *Számokt 2012 konferenciakötet, 22. Nemzetközi számítástechnika és oktatás konferencia*, pages 314–315, 2012.

[50] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attack. *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 427–437, 1990.

[51] A. M. Odlyzko. Discrete logarithms: The past and the future. *Designs, Codes, and Cryptography*, 19:129–145, 2000.

[52] B. Preneel. Analysis and Design of Cryptographic Hash Function. PhD thesis, Katholieke Universiteit Leuven.

[53] B. Preneel. The state of cryptographic hash functions. *Lecture Notes in Computer Science*, 1561:158–182, 1999.

[54] M. O. Rabin. Digitalized Signatures and Public-Key Functions as Intractable as Factorization. *MIT Laboratory for Computer Science LCS/TR-212*, 1979.

[55] R.L. Rivest, A. Shamir, and L.M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21 (2):120–126, 1978.

[56] L. Rónyai, G. Ivanyos, and R. Szabó. *Algoritmusok*. Typotex, 2004.

[57] A. Shamir. A polynomial-time algorithm for breaking the basic Merkle - Hellman cryptosystem. *Information Theory, IEEE Transactions*, 30 (5):699–704, 1984.

[58] V. Shoup. A library for doing number theory. http://www.shoup.net/ntl/.

[59] V. Shoup. Searching for primitive roots in finite fields. *Mathematics of Computation*, 58:369–380, 1992.

[60] W. Stallings. *Cryptography and Network Security. Principles and Practice*. Prentice Hall, 2010.

[61] D. R. Stinson. *Cryptography. Theory and Practice*. Chapman & HallCRC Press, 2006.

[62] U.V. Vazirani and V. Vazirani. Efficient and secure pseudo-random number generation. *Advances in cryptology, Proceedings of CRYPTO 84*, pages 193–202, 1985.

[63] G.S. Vernam. Cipher printing telegraph systems for secret wire and radio telegraphic communications. *Journal of the IEEE*, 55:109–115, 1926.

[64] V. Villányi. RSA signatures schemes with sublimal-free public-key. *Tatra Mountains Mathematical Publications*, 41:19–32, 2008.

[65] M.J. Wiener. Cryptanalysis of short RSA secret exponents. *Information Theory, IEEE Transactions*, 36 (3):553–558, 1990.

[66] M.J. Wiener. Safe prime generation with a combined sieve. *Crypto Eprint Archive entry 2003: 186*, 2003.

A függelék

Referált nemzetközi folyóiratban megjelent cikkek

1. MÁRTON Gyöngyvér: Public-key cryptography in functional programming context, Acta Universitatis Sapientiae Informatica, vol 2. Nr. 1, 2010, 99-112, ISSN 1844-6086 [Zbl 1197.68043]

2. KÁTAI Zoltán, KOVÁCS Lehel István, KÁSA Zoltán, MÁRTON Gyöngyvér, FOGARASI Kinga and FOGARASI Ferenc: Cultivating algorithmic thinking: an important issue for both technical and HUMAN sciences, Teaching Mathematics and Computer Science, 2011, 9/1. [Zbl MathEduc 2012a.00772]

3. MÁRTON Gyöngyvér: CCA-secure key-encapsulation mechanism based on factoring assumption, Tatra Mountains Mathematical Publications, vol 53, 2012, 137–146, ISSN 1210-3195 [Zbl 06135378]

B függelék

Lektorált egyetemi jegyzet

1. MÁRTON Gyöngyvér: Fundamentals of cryptography (in Hungarian), Sapientia-EMTE, Scientia, Cluj-Napoca, 2008, ISBN 978-973-1970-00-4

C függelék

Tudományos konferencia kötetben megjelent cikkek

1. MÁRTON Gyöngyvér, The security of electronic voting, Proceedings of Számokt 2007, 17th International Conference on Computers and Education, Oradea (Nagyvárad), Romania, 2007, ISSN 1842-4546.

2. MÁRTON Gyöngyvér, Recursion, dynamic programming, functional programming, Proceedings of Számokt 2008, 18th International Conference on Computers and Education, Sumuleu Ciuc (Csíksomlyó), Romania, 2008, ISSN 1842-4546.

3. MÁRTON Gyöngyvér, Methods and tools for teaching cryptography, Proceedings of Infodidact 2009, 2th Methodological Conference on Informatics, Szombathely, Hungary, 2009.

4. MÁRTON Gyöngyvér, Eligible problems in cryptography, Proceedings of Számokt 2009, 19th International Conference on Computers and Education, Tg. Mures (Marosvásárhely), Romania, 2009, ISSN 1842-4546.

5. MÁRTON Gyöngyvér, Definitions related to the security of public-key cryptosystems, Proceedings of Számokt 2010, 20th International Conference on Computers and Education, Satu-Mare (Szatmárnémeti), Romania, 2010, ISSN 1842-4546.

6. MÁRTON Gyöngyvér, Message authentication codes and CCA-attack, Proceedings of Számokt 2011, 21th International Conference on Computers and Education, Cluj-Napoca (Kolozsvár), Romania, 2011, ISSN 1842-4546.

7. MÁRTON Gyöngyvér, Security of Encryption Systems based o Factoring Assumptions, Proceedings of Számokt 2012, 22th International Conference on Computers and Education, Alba-Julia (Gyulafehérvár), Romania, 2012, ISSN 1842-4546

D függelék

Tudományos konferenciákon elhangzott előadások

1. MÁRTON Gyöngyvér: Functional programming and cryptography, "The day of science in Transylvania" conference, Miercurea Ciuc, Romania, 2006.

2. MÁRTON Gyöngyvér: Experience in teaching functional programming, "MINICONF" conference, Tg-Mures, Romania, 2007.

3. MÁRTON Gyöngyvér: The role of functional programming in cryptography, Central European Functional Programming School, Cluj Napoca, Romania, 2007.

4. GYŐRFI Eugen, MÁRTON Gyöngyvér: Algebra of the 2D and 3D Cutting/Packing patterns, 6th ESICUP international conference, Valencia, Spain, 2009.

5. MÁRTON Gyöngyvér: Cryptography based on problems of graph theory, 1st "Sapientia MatInfo" International Conference, Tg.Mures, Romania, 2009.

6. MÁRTON Gyöngyvér: Technics used in cryptosystems to achieve CCA-security, 2nd "Sapientia MatInfo" international conference, Tg.Mures, Romania, 2011.

7. MÁRTON Gyöngyvér: Remarks on Blum-Goldwasser public-key encryption system, 11th Central European Conference on Cryptology, Debrecen, Hungary, 2011.

8. MÁRTON Gyöngyvér: Chosen ciphertext security of public-key encryption systems, 9th Joint Conference on Mathematics and Computer Science, Siófok, Hungary, 2012.

9. MÁRTON Gyöngyvér: CCA-secure key-encapsulation mechanism based on the factoring assumption, 12th Central European Conference on Cryptology - TatraCrypt, Smolenice, Slovakia, 2012.

E függelék

Köszönetnyilvánítások

Ezúton szeretném megköszönni mindazon személyeknek a segítségét, akik közvetlenül vagy közvetett módon hozzájárultak PhD értekezésem elkészítéséhez.

Köszönöm Dr. Pethő Attila Tanár Úrnak, hogy témavezetőm volt, hogy számos alkalommal segítette tudományos munkámat úgy szakmai szigorúsággal, mint erkölcsi és anyagi támogatással.

Köszönöm a Sapientia Egyetem vezetőségének és a Matematika-Informatika Tanszék tanárainak azt a kitartó bizalmat mellyel az elmúlt években, mint kollégák támogatták tudományos tevékenységemet.

Köszönöm a Debreceni Egyetemnek, hogy biztosította számomra a tandíjmentességet a Doktori Iskola levelező programjának keretén belül.

Köszönöm a Kutatási Programok Intézetének (Kolozsvár) azon anyagi támogatást, amelyet a Sapientia Doktori Ösztöndíjprogram keretén belül nyújtott számomra.

Köszönöm családomnak azt a felelősségteljes hozzáállást mellyel segítették ezen értekezés elkészülését.