2. Kriptográfiai alapfogalmak

A *kriptológia* klasszikusan a titkos illetve védett kommunikáció tudománya, amelynek két fô ága a *kriptográfia* és a *kriptoanalízis*. A kriptográfia azon algoritmikus módszerekkel foglalkozik, amelyek biztosítják az üzenetek (tárolt információk) titkosságát vagy hitelességét. A kriptoanalízis a titok - általában illetéktelen - megfejtésére ("feltörésére") szolgáló módszerek tudománya.

Az információk védelmi rendszerének az *algoritmikus módszerek* általában csak az egyik pillérét jelentik, amelyek megfelelô *fizikai védelemmel* valamint ügyviteli-*rendszabályi eljárások* alkalmazásával együtt biztosíthatják a valóságos rendszerekben az információvédelmet. Jegyzetünk témaköre az algoritmikus módszerekre terjed ki.

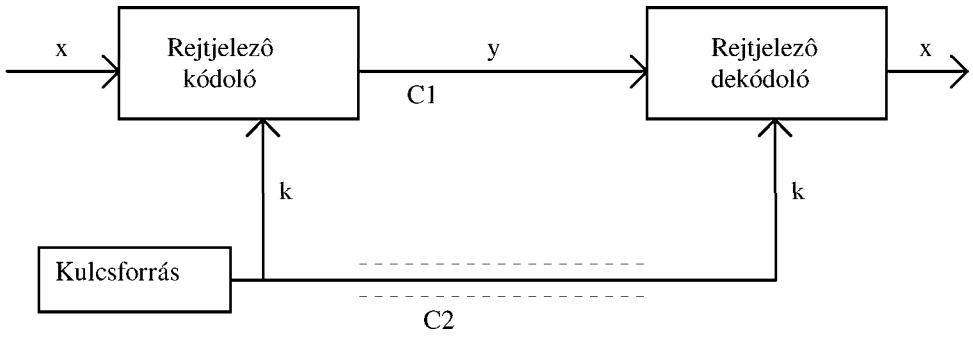
Egy üzenetküldõ szeretné átküldeni üzenetét egy olyan kommunikációs csatornán, amelyrõl feltételezhetõ hogy nem biztonságos, abban az értelemben, hogy egy a csatornán "hallgatózó" illetéktelen személy képes a folyó üzeneteket tartalmát megismerni. Az üzenetet küldõ szeretné ezt megakadályozni, s ezért algoritmikus védelemmel látja el üzenetét, rejtjelezi azt. A csatornán átküldendõ üzenetet *nyílt üzenet*nek (plaintext) a rejtjelezett változatát *rejtett üzenet*nek (ciphertext) hívjuk. Ha a rejtjelezendõ üzenetek illetve a megfeleltetett rejtett üzenetek N bit méretû $x=(x_1,...,x_N)$, illetve $y=(y_1,...,y_N)$ blokkok, akkor blokk kódolásról beszélünk. A rejtjelezõ kódolás (encryption):

$$y = E_k(x), \qquad (2.1)$$

ahol E_k egy $k=(k_1,...,k_K)$ paraméterû *kódoló (rejtjelezõ) transzformáció*, amely invertálható a k paraméter tetszõleges értéke mellett. A k vektor a rejtjelezés kulcsa, az az információ, amely meghatározza (kiválasztja) az aktuális rejtjelezõ transzformációt. A mai rendszerek nagy többségében x, y és k vektorok binárisak. A rejtjelezõ kódolás inverze a *dekódoló transzformáció*, D_k :

$$x = D_k(y) \qquad (2.2)$$

Az alábbi ábrán láthatjuk ezen *konvencionális (rejtett kulcsú) rejtjelezéses* kommunikáció vázlatát:



2.1. A konvencionális rejtjelezés

A rejtett üzenetet a C1 nyilvános csatornán (public channel), míg a kulcsot a C2 titkos csatornán (secret channel) továbbítjuk a dekódoló oldalra. A csatorna nyilvánosságát illetve titkosságát egy támadóval szembeni korlátozott illetve tökéletes védetsége jelenti. Például közhasználatú hírközlési csatornáink nyilvános, míg egy megbízható futár általi kézbesítés titkos csatornának felel meg.

A támadó célja lehet az x nyílt szöveg megállapítása vagy a k kulcs megszerzése.

Alapfeltételezés a támadó ismereteivel kapcsolatosan az, hogy az aktuális k kulcs kivételével a kódolás, dekódolás alkalmazott algoritmusát teljes részletességgel ismeri.

Ez azt is jelenti, hogy egy rejtjelezõ algoritmus által nyújtott védettség nem haladhatja meg a kulcsa védettségének mértékét. A kulcs tehát az a titkos információ, amely gyorsan és kívánatosan gyakran cserélhetõ, míg a rejtjelezõ rendszer többi elemét hosszabb ideig nem szükséges változtatni.

A támadó által alkalmazható módszerek közül csak az algoritmikus típusú támadási módszereket tekintjük a továbbiakban (emellett elképzelhetõk további módszerek, pl. betörés, megvesztegetés, szabotázs stb.). Az algoritmikus típusú támadásnak passzív és aktív módját különböztetjük meg, amelyet a nyivános csatornán hajt végre a támadó.

Passzív módszer a már említett lehallgatás, amikor a támadó a nyilvános csatornán áramló rejtett üzenetek sorozatának birtokába jut. A támadó célja az, hogy egy megfigyelt rejtjelezett kapcsolatból kinyert információk felhasználásával algoritmikus támadást indítson - általában - az aktuális kulcs meghatározására. Ezt az eljárást szokás *rejtjelfejtés*nek nevezni, amelynek eszköztárát a *kriptoanalízis* adja.

A passzív típusú támadásokat azok növekvô ereje szerint klasszikusan a következô kategóriákba soroljuk:

1. Támadás azonos kulccsal kódolt rejtett üzenetek birtokában, amit rejtett szövegû támadásnak nevezünk (ciphertext only attack).

2. Támadás azonos kulccsal kódolt nyílt és rejtett üzenetpárok birtokában, amit ismert nyílt üzenetû támadásnak hívunk (known plaintext attack).

3. Támadás abban az esetben, ha a támadó maga választhatja meg a nyílt (rejtett) üzeneteket, amelynek rejtett (nyílt) párját látni szeretné, amit választott szövegû támadásnak nevezünk (chosen text attack).

Aktív támadási módszer a rejtett üzenetek csatornából történô kivonása, kicserélése, amelynek célja a hozzávetôlegesen ismert tartalmú (struktúrájú) üzenetek támadó szempontjából kedvezô módosítása (üzenetmódosítás). Egy másik módszer az, amikor a támadó megpróbálja egy legális felhasználó szerepét eljátszani azért, hogy valamely másik legális rendszer-elemtôl (-tagtól) információt csaljon ki (megszemélyesítés).

Egy üzenet *titkosság*a (privacy) azt jelenti, hogy csak a legális (kívánatos) partner számára rekonstruálható annak nyílt tartalma, míg egy üzenet *hitelesség*e (authenticity) azt jelenti, hogy azt olyan személy generálta, aki a kulcs legális birtokában van. Lényegében azt mondhatjuk, hogy a titkosság az üzenet tartalmával, míg a hitelesség a küldô személlyel kapcsolatos. Szemléletes példával illusztrálva: ha a postás felbontja a levelet, akkor annak tartalma már nem titok, de ha a levél tartalmát nem módosítja, s úgy kézbesíti, az üzenet még hiteles marad.

Azt mondjuk, hogy a támadó *feltörte a titkosító algoritmust*, ha "gyorsan" meg tudja állapítani egy lehallgatott üzenet nyílt tartalmát, függetlenül attól, hogy éppen melyik kulcsot alkalmazzák. A gyorsasaság olyan idõintevallumot jelent, amelyen belül a támadó sikeresen használhatja céljaira a megszerzett információt.

A rejtjelezés célja alapvetôen a passzív támadások megakadályozása. Az aktív típusú támadásokat algoritmikus eszközökkel megakadályozni nem tudjuk, de megfelelô *kriptoprotokollok* alkalmazásával a támadást észrevethetôvé tehetjük. A protokollok általában véve egy elôre meghatározott üzenetcsere folyamatot jelentenek, amelyet kettô vagy több partner bonyolít le kooperatívan valamely feladat végrehajtására. A kriptográfiai protokollok építô elemként használják a rejtjelezô kódolót, s biztosítják a kapcsolat védett felépülését, a kommunikáció alatti aktív támadások észlelhetôségét, összességében garantálják a partnerek és üzenetfolyamataik hitelességének ellenôrizhetôségét. A rejtjelezô algoritmusok és a protokollok tervezése és analízise a kriptográfia tudományának két fô ágát jelentik.

A konvencionális titkosítási algoritmusok mellett a 70-es évek végétôl kezdôdôen rohamosan fejlôdik a *nyilvános kulcsú algoritmusok* családja, amelyeknek nagy teret szentelünk e jegyzetben is.

3. Szimmetrikus kulcsú kódolás

3.1. One time pad: tökéletes titkosítás

Az információelmélet eszközöket alkalmazó alábbi analízisben azt tételezzük fel, hogy csak rejtett üzenetek állnak az analízis rendelkezésére, vagyis rejtett üzenetû támadást vizsgálunk.

Jelölje X és K a nyílt üzenet és a kulcs valószínûségi változókat, amelyek egy realizációja x és k az aktuális nyílt szöveg és kulcs. Feltesszük, hogy X és K független valószínûségi változók. A Y rejtett üzenet, mint valószínûségi változó a következô:

$$Y = E_K(X) \qquad (3.1.1)$$

azaz determinisztikus függvénye X és K valószínûségi változóknak.

3.1.1. Definíció: *Tökéletes titkosítás*ról akkor beszélünk, ha az X és Y valószínûségi változók statisztikailag függetlenek, azaz kölcsönös információjuk nulla, I(X,Y)=0.

Következésképpen, ha a rejtjelezés tökéletes a lehallgató támadó a megfigyelt rejtett szövegek alapján elméletileg sem képes a nyílt szöveg megfejtésére.

3.1.1. Tétel: Tökéletesen titkosító algoritmus létezik.

Bizonyítás: Legyen kódolás a következô

$$Y=X+K \qquad (3.1.2)$$

ahol X,Y,K bináris N bites vektorok, s az összeadás koordinátánkénti modulo 2 (XOR) mûvelettel történik. Legyen K egyenletes eloszlású az N bites vektorok halmazán. Ekkor

$$P(Y=y|X=x)= P(X+Z=y|X=x)= P(Z=y-x|X=x)= P(Z=y-x)= 2^{-N}$$

ahol felhasználtuk X és Z függetlenségét. Innen

$$P(Y=y) = \Sigma_x P(X+Z=y|X=x)P(X=x) = 2^{-N} = P(Y=y|X=x)$$

adódik, azaz X és Y függetlenek. ♦

Észrevehetjük, hogy X és Y függetelenségéhez nem kellett semmit sem feltételezni X eloszlásáról, továbbá azt, hogy Y eloszlása egyenletes lesz. Azonban ahhoz, hogy a szóbanforgó *one time pad* kódolás (Vernam kódoló) realizálható legyen, minden egyes

üzenet kódolásához új kulcs szükséges. Így az adási és vételi oldalon nagy mennyiségû kulcsot kellene tárolni vagy egy védett átviteli csatornán továbbítani. Pontosabban a szükséges kulcsbitszám azonos az átküldendô nyílt adatfolyam adatbitjeinek számával. Nyilván ezért érdemes tömöríteni az indformációforrás kimenetét.

Ezen tökéletes, de kevésbé praktikus eljáráshoz szorosan kapcsolódik két fontos fogalom, a gyakorlati és a feltétel nélküli titkosság fogalma. Egy rejtjelezô algoritmust gyakorlati titkosságot nyújtónak nevezünk, ha feltörése irreálisan nagy számítási kapacitást kíván, de elméletileg lehetséges (például valamely kimerítô keresés útján). A feltétel nélküli titkosság azt jelenti, hogy a megszerezhetô információ elvileg sem elegendô ahhoz, hogy sikeres feltörést hajtsunk végre bármekkora számítási kapacitás is áll rendelkezésre. Az egyetlen ma ismeretes feltétel nélkül titkos rejtjelezés a fentebb elemzés one time pad.

3.2. Algebrailag zárt blokk rejtjelezõ és támadása

3.2.1 Zártság és csoport tulajdonság

Tekintsük a blokk rejtjelezõ algoritmust, mint transzformációk egy halmazát, amelybõl egy aktuális transzformációt a kulcs aktuális értéke választja ki:

$$E = \{E_k;\ E_k : X \to Y, k \in K\}$$

ahol E a transzformációk halmaza, X a nyílt üzenetek halmaza, Y a rejtett üzenetek halmaza, K a kulcshalmaz. E_k: X → Y invertálható transzformáció. Tipikus gyakorlati esetben X = Y, ahol X és Y bináris n hosszú vektorok halmaza. Ekkor E_k egyben permutáció, s az alábbiakban ezen megszorítást feltételezzük.

Tekintsük azon egymûveletes algebrai strukturát a rejtjelezõ transzformációk E halmazán, ahol a mûvelet két transzformáció egymás utáni alkalmazása, s nevezzük szorzásnak.

Jelölje ezen strukturát T = {E, "·"}.

3.2.1 Definició: *T zárt, ha E·E = E, azaz tetszõleges két, E halmazbeli transzformáció szorzata E-beli transzformáció, formálisan*

$$E_{k_1} \cdot E_{k_2} = E_{k_3}, k_i \in K$$

Nem nehéz belátni, hogy T csoport tulajdonságokkal rendelkezik. Ehhez azt kell belátni, hogy az identitás benne van, s minden elemének létezik az inverze.

3.2.1 Tétel: *A T zárt algebrai struktúra csoport.*

Bizonyítás: Az egyszerûbb jelölés kedvéért E_i álljon E_{k_i} helyett. Tetszõleges E_1, E_2, E_3 esetén, ahol $E_1 \neq E_3$ fennáll, hogy

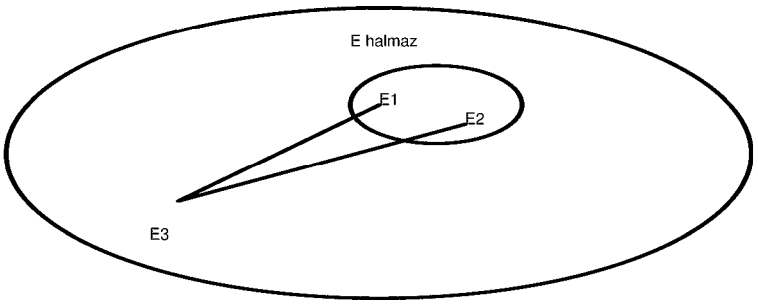
$$E_1E_2 \neq E_3E_2 \qquad (3.2.1)$$

Ha ugyanis (3.2.1) nem állna fenn, E_2 inverzével jobbról szorozva mindkét oldalt ellentmondásra jutnánk. Következésképpen

$$\{E_1E_2,\ E_1 \in \mathrm{E}\} = \mathrm{E} \qquad (3.2.2)$$

Következésképpen tetszõleges E_2 transzformációhoz létezik olyan E_2^* transzformáció, amelyre $E_2^* \cdot E_2 = E_2$, ahol mindkét oldalt jobbról E_2 inverzével szorozva, a jobb oldalon az I identitás transzformációt kapjuk, azaz $E_2^* = I$, s így $I \in \mathrm{E}$. Ha tetszõleges $E_2 \in \mathrm{E}$ transzformációt balról szorzunk a különbözõ E-beli elemekkel, (3.2.2) alapján a szorzat végigfutja az összes E-beli elemet, köztük az I identitást is, azaz létezik $E_2' \in \mathrm{E}$ transzformáció, amelyre $E_2' \cdot E_2 = I$. E_2' az E_2 transzformáció inverze, $E_2' = (E_2)^{-1}$.
■

A zártság tehát esetünkben további algebrai strukturáltságot von maga után. Egy blokk rejtjelezõ algebrai zártságának ellentéte az az eset, amikor E az X feletti permutációk halmazának egy véletlenszerûen választott részhalmaza. Ezen utóbbi esetben két E-beli transzformáció egymás utáni alkalmazása egy, nagy valószínûséggel nem E-beli permutáció.



3.2.1 ábra
Nem zárt rejtjelezõ

A zártságból következõ csoporttulajdonság kedvezõtlen, mert az ún. középen találkozás algoritmikus támadásra ad lehetõséget. A támadás alapja a valószínûségszámításból már jól ismert születésnapi paradoxon.

3.2.2 Születésnapi paradoxon és a középen találkozás támadás

A születésnapi paradoxonnal kapcsolatos kérdés a következõ:
"Mekkora a valószínûsége annak az eseménynek, hogy emberek egy véletlenszerûen választott r fõs csoportjában van legalább két személy, akik azonos napon születtek?"
A paradoxon, vagyis az ember józan sejtésével nehezen érthetõ tény az, hogy ahhoz, hogy a kérdéses valószínûség 1/2 körüli érték legyen elegendõ $r = 23$ személybõl álló csoportot választani. Ugyanis ahhoz, hogy 1/2 körüli valószínûsége legyen annak az eseménynek, hogy egy általam véletlenszerûen válogatott r fõs csoportban legyen egy személy az enyémmel megegyezõ születésnappal, $r = 253$ személybõl álló csoport szükséges.

A kérdéses valószínûség elemi kombinatorikus úton meghatározható:

$$p_r = 1 - \frac{\binom{365}{r} \cdot r!}{365^r}$$

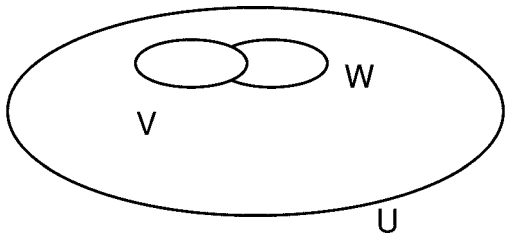
amely jól közelíthetõ a

$$p_r \approx 1 - \exp(-r^2 / (2m)) \qquad (3.2.3)$$

kifejezéssel ($m = 365$). $r = m^{1/2}$ választás esetén $p_r \approx 1 - \exp(-0.5) \approx 0.4$. Tehát ha van egy nagy méretû halmazunk, amelynek elemeit egy egész számmal kifejezett jellemzõvel cimkézzük fel, amely jellemzõ m különbözõ értékû lehet, akkor m négyzetgyökének nagyságrendjébe esõ méretû részhalmazt választva már nagy annak valószínûsége, hogy van benne legalább két, azonos tulajdonsággal felcimkézett elem.

Egy kissé módosított formában is fogjuk használni a paradoxont, s ez a következõ kérdéshez kapcsolódik:

"Mekkora a valószínûsége annak, hogy egy U nagy méretû alaphalmazból véletlenszerûen választva V és W r méretû részhalmazokat, azok metszete nem üres?"



3.2.2 ábra
Születésnapi paradoxon változata

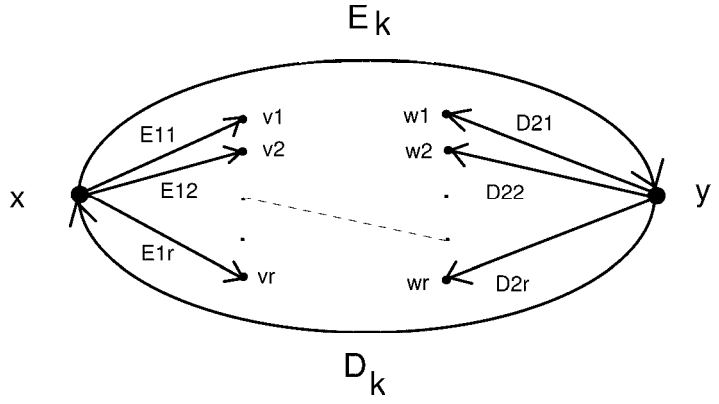
A kérdéses valószínûség és közelítése az alábbi:

$$P(V \cap W \neq 0) = 1 - \frac{\binom{m}{2r}(2r)!}{\left(\binom{m}{r} r!\right)^2} \approx 1 - \exp(-3r^2 / m) \tag{3.2.4}$$

$r = m^{1/2}$ választás esetén $p_r \approx 1$ - exp(-3) ≈ 0.95. A születésnapi paradoxonra épül a zárt strukturájú rejtjelezõk elleni középen találkozás típusú támadás.

3.2.3 Középen találkozás támadás zárt struktúrájú rejtjelezõ ellen

A támadás célja egy, az éppen alkalmazottal ekvivalens transzformáció konstruálása. A támadáshoz szükséges, hogy a támadó rendelkezzen a pillanatnyilag használt kulccsal rejtett nyílt-rejtjelezett blokk-párok - valahogyan megszerzett - halmazával. Azaz egy $Q = \{(x_1, y_1), (x_2, y_2), \ldots (x_s, y_s)\}$ halmazzal, ahol $y_l = E_k(x_l)$, k az ismeretlen kulcs, azaz $E_k \in E$ a nem ismert, pillanatnyilag alkalmazott transzformáció. A támadás megértését segíti az 3.2.3 ábra.



3.2.3.ábra
Középen találkozás támadás

Véletlenszerûen választunk $2r$ kulcsot. A Q halmazból kiveszünk egy párt (az 3.2.3 ábrán (x, y)), s a nyílt x blokkot r számú véletlenszerûen választott kulccsal kódoljuk:

$v_j = E_{1j}(x) \quad j = 1, 2, \ldots r$

hasonlóan az y rejtett blokkot r számú véletlenszerûen választott kulccsal dekódoljuk

$$w_i = D_{2i}(y) \quad i = 1, 2, \ldots r$$

Megvizsgáljuk, hogy van-e a $V = \{v_1, v_2, \ldots v_r\}$ és a $W = \{w_1, w_2, \ldots w_r\}$ halmaznak közös eleme. Ha van, s például $v_j = w_i$ egy ilyen közös elempár, akkor sejtésünk az, hogy

$$E_k = E_{2i}E_{1j} \tag{3.2.5}$$

amely sejtésünket ellenõrizzük további Q halmazbeli párokon, azaz hogy fennáll-e az

$$y_l = E_k(x_l) = E_{2i}[E_{1j}(x_l)] \tag{3.2.6}$$

egyenlõség, $l = 2, \ldots s$. Ha igen, akkor a Q halmaz mérete szerinti biztonsággal igazoltuk sejtésünket. Ha valamelyik Q-beli nyílt-rejtett páron sejtésünk megdõl, akkor elölrõl újrakezdjük az eljárást, újabb $2r$ kulcsot sorsolva.

Kérdés, hogy mekkora legyen r értéke, illetve a Q halmaz mérete. Az r értékkel kacsolatosan a választ a születésnapi paradoxon adja. Az U, V, W halmazok legyenek transzformációk következõ halmazai:

$U = E$
$V = \{E_{11}, E_{12}, \ldots E_{1r}\}$
$W = \{D_{21}E_k, D_{22}E_k, \ldots D_{2r}E_k\}$,

ahol $V \subset U$ állítás nyilvánvaló. Mivel a csoport-tulajdonság miatt $(E_{2i})^{-1} \in E$, azaz $D_{2i} \in E$, $i = 1, 2, \ldots r$ is fennáll, tehát W is részhalmaza U halmaznak. Továbbá nyilván W is véletlen részhalmaza az U halmaznak. Ha van közös eleme V és W részhalmazoknak, azaz ha például

$$E_{1j} = D_{2i}E_k$$

akkor minkét oldalt balról E_{2i} -vel szorozva

$$E_k = E_{2i}E_{1j}$$

adódik, azaz a nem ismert E_k aktuális transzformációt két véletlenül sorsolt transzformáció szorzataként elõállítottuk.

A születésnapi paradoxon változata alapján annak valószínûsége, hogy V és W hamazok metszete nem üres, elegendõen nagy egy sikeres támadáshoz, ha r értéke az U (= E) elemszáma négyzetgyökének nagyságrendjébe esik. Ez ekvivalensen a kulcsok halmaza

méretének négyzetgyökét jelenti. Például egy 60 bites kulcsméret esetén 2^{30} nagyságrendről van szó.

A Q nagyságrendjét illetően azt kell észrevennünk, hogy egy (3.2.6) szerinti sejtésellenőrzési lépésen egy téves sejtés $2^{-|X|}$ valószínűséggel megy át, azaz kisszámú ellenőrző lépés nagy biztonságú ellenőrzést jelent. Az eljárás praktikusságának korlátját tehát az r értéke adja.

Az algebrai zártság tehát veszedelmes tulajdonság lehet. Ugyanakkor azonban kicsi az esélye, hogy egy rejtjelező kódoló ilyen tulajdonsággal rendelkezzen. A fenti gondolatoknak praktikus haszna, ezért elsősorban nem közvetlenül a kódoló transzformációk esetén, hanem a többszörös kódolás esetén van.

3.2.4 Többszörös kódolás

Több különböző kódolás eljárás kombinálásától reméli néha a tervező, hogy egy erősebb kódolóra jut. Az egyik legközvetlenebb ezen irányba tett lépés a kétszeres kódolás, azaz

$$y = E_2[E_1(x)] \qquad (3.2.7)$$
$$x = D_1[D_2(y)]$$

$E_1, E_2 \in \mathrm{E}$, kétszeres kódolási, illetve dekódolási lépés alkalmazása, lépésenként két külön kulcsot használva. Jelölje az egylépéses kódolás kulcshosszát $|k|$. A kulcshossz a kétszeres kódolással formailag kétszeresére nőtt, hiszen két kulcsot használunk. Formailag, mert ténylegesen csak akkor van így, ha a kétszeres kódolással a kétszeres kulcshossznak megfelelő számú különböző transzformációt állítanánk elő. Ha a kódolónk csoport tulajdonságú, akkor ez biztosan nincs így, hiszen ezesetben $E_2[E_1(x)] = E_3(x)$, ahol $E_3 \in \mathrm{E}$, tehát egyáltalán nem változott a különböző transzformációk száma, azaz kétszeres kódolással csak ugyanazon transzformációkat tudjuk előállítani, mint egyszeres kódolással.

A kétszeres kódolás általános hiányossága az, hogy a kimerítő kipróbálásos támadásnál lényegesen egyszerűbb kínálkozik, ugyanis a kétszeres kódolás lehetővé tesz középen találkozás támadást.

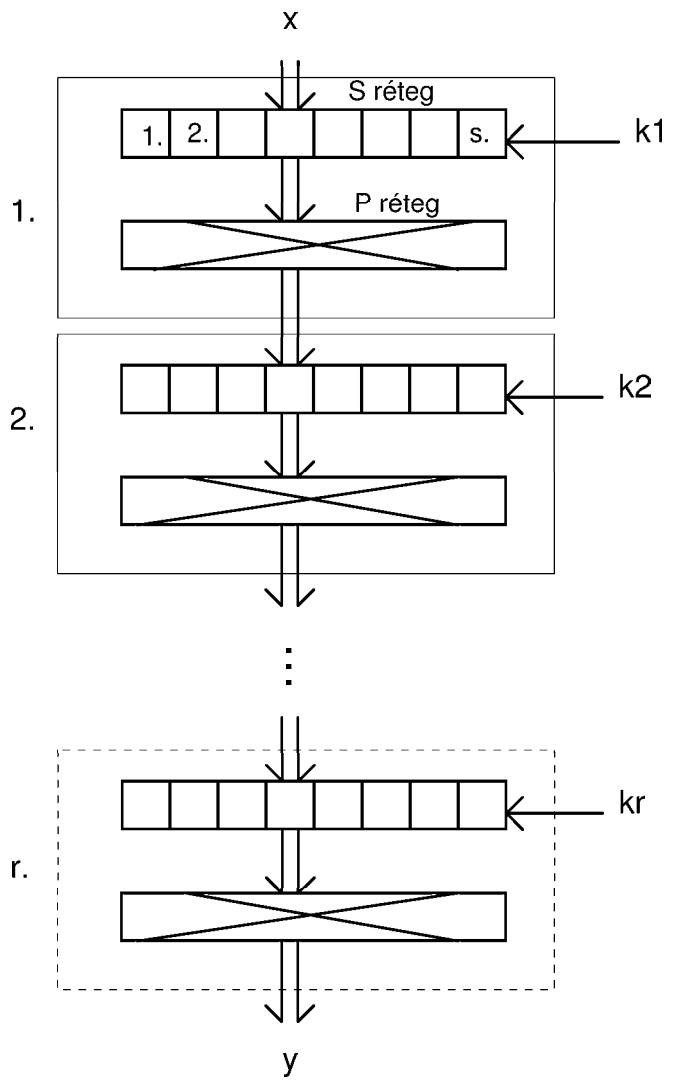
Tudjuk, hogy két "egylépéses" kódoló transzformáció szorzata a nem ismert transzformáció. Az x nyílt blokkot kódoljuk egylépéseses kódolással az összes különböző kulccsal, s hasonlóan y rejtett blokk párját egylépéseses dekódolással az összes különböző kulccsal. A kódolás és dekódolás eredmények halmazának metszetét vizsgáljuk. Ha egy $E_{1i}(x)$ kódolási lépés eredménye egybeesik egy $D_{2j}(y)$ dekódolási lépés eredményével, akkor a keresett ismeretlen kétlépéses transzformációra a sejtésünk $E_{2j}E_{1i}$, amit további nyílt-rejtett blokkokon történő kipróbálással ellenőrzünk. Nyilvánvaló, hogy ezesetben a metszetben benne van a keresett pár is.

3.3. Helyettesítéses-permutációs rejtjelezõk

A kriptanalízis módszerek jelentõs részénél - fõleg a statisztikai alapú támadásoknál - a siker feltétele, hogy a rejtjelezett szöveg magán hordozza az eredeti szöveg speciális statisztikai inhomogenitását. E támadási lehetõséget kézenfekvõen úgy csökkenthetnénk, hogy kulccsal címzett nagyméretû helyettesítõ táblázatok halmazát használnánk, ahol a nyílt szöveg N hosszú blokkjait helyettesítenénk, akkora N értéket alkalmazva, amelynél már egyenleteshez közelivé válik a transzformált nyílt szöveg, azaz a rejtett szöveg eloszlása. Ez az információelméleti terminológiában azt jelenti, hogy a transzformált nyílt szöveg N karakteres blokkjai nagy valószínûséggel tipikus sorozatok lennének, amelyek mindegyikének valószínûsége közel azonos. Másrészt, mivel a lehetséges nyílt szöveg - rejtett szöveg párok halmaza ekkor igen nagy, még több ilyen pár megfigyelése is elenyészõ mennyiségû információt ad a leképezésrõl. Gondoljuk meg azonban, hogy ekkor már $N = 10..30$ karakteres blokkhosszak esetén is megoldhatatlan tárolási probléma elé kerülnénk a helyettesítõ táblázatokat illetõen (még egyet sem tudnánk tárolni közülük). Mivel mindenképpen nagy blokkméretekkel szükséges dolgozni a statisztika "kisimításához", ezért a kivitelezhetõség érdekében le kell mondanunk arról, hogy a helyettesítõ táblázatainkat a lehetséges helyettesítések teljes halmazából válasszuk. Azon - még így is elegendõ méretû - részhalmazból kell szelektálnunk, melynek helyettesítései tömörebben megadható formában függnek a kulcstól. Meg kell még jegyeznünk, hogy ehhez a statisztikai homogenizáláshoz elvileg még az említettnél is nagyobb méretû blokkhosszak lehetnek szükségesek, ha a rejtjelezendõ forrás kimenetén jól strukturált táviratok, dokumentumok sorozata jelenik meg.

A fenti problémák megoldására Shannon az ún. keverõ transzformáció alkalmazását javasolta. Ez a transzformáció kis méretû helyettesítések (S-dobozok), valamint bit permutációk (P-dobozok) többszöri felhasználásával elõállítható helyettesítés. A javasolt megoldás azt sejteti, hogy ilyen módon "erõs" transzformációt lehet elõállítani. A Shannon által javasolt helyettesítéses-permutációs rejtjelezõ felépítése az 3.3.1 ábrán látható.

Azonos struktúrájú kaszkádba kötött r réteg (round) mindegyikében egy helyettesítési (S, Substitution) és egy bit-permutációs (P, Permutation) réteg kaszkádja található. Az S-réteg s darab párhuzamosan mûködõ minihelyettesítõbõl (S-doboz, S-box) áll. Egy S-doboz 2^t számú helyettesítõ táblázatot tartalmaz. Az egyes rétegeken belül az S-dobozok táblázatkészlete különbözõ. Az adott réteg N bites input blokkja (elsõ rétegben a nyílt szöveg) s darab, egyenként N/s méretû részvektorra van felszeletelve, s ezen részvektorok képezik az egyes minihelyettesítõk inputját egyrészrõl, másrészrõl a k_i rétegkulcs (iterációs kulcs) bitvektor egy t bites szelete. A kulcsszelet kiválaszt egy konkrét transzformációt az S-doboz készletébõl. Az egyes rétegek rétegkulcsa más és más, így az egyes rétegekben kiválasztott aktuális S réteg is más. A k kulcsból számítódnak a rétegkulcsok. A k kulcs változtatásával változnak a rétegkulcsok, így változnak az S-rétegek, s ezáltal a teljes $y=E_k(x)$ transzformáció is. A P-rétegek bitperutációt hajtanak végre, azaz átkeverik az N bites blokkbeli bitsorrendet. A P rétegek rögzítettek, kulcsbittel nem vezéreltek.



3.3.1.ábra
A rétegzett struktúra

A sok réteg mûködési mechanizmusára szemléletes illusztráció az úgynevezett lavinahatás létrejötte. Ugyanis például azt is szeretnénk, hogy egy "jó" rejtjelezõ esetén az a nyílt szöveg inputban akár egy bit invertálódása is véletlenszerû változást okozzon a teljes output blokkban. Tegyük fel, hogy az S dobozok helyettesítései duplikálnak egy bemeneti vátozást, azaz ha egy bit billen a bemeneten, akkor a kimeneten tipikusan két bit változzon. A P réteg bemenetére egy S-doboz kimenetérõl érkezõ két bitpozícióbeli változást tipikusan két különbözõ S-doboz bemenetére juttat, mint egy-egy bitpozícióbeli változást. Ezen következõ rétegben már tipikusan négy bitben történik változást, majd a következõben tipikusan nyolc pozícióban s.í.t. A változások egymás hatását is kiolthatják a további rétegekben, s legvégül, az output blokkban "véletlenszerû" bitpozíciókban lesz változás.

Kérdés persze, hogy hogyan lehet olyan S- és P-dobozokat találni, amelyekbõl a fenti feltételeknek eleget tevõ rejtjelezõ építhetõ. Mik az S- és P-doboz tervezés pontos kritériumai, és hogyan lehet a kritériumokat kielégítõ transzformációkat tervezni? Mekkora legyen az S-dobozok mérete, és hány réteget alkalmazzunk? A jegyzetünk keretében az S doboz tervezés néhány alapvetõ kritériumát tekintjük át.

3.3.1. S-doboz tervezési kritériumok

Invertálhatóság, nemlinearitás

Nyilvánvaló alapkövetelmény, hogy a teljes transzformáció invertálható és nemlineáris legyen. Ez közvetlenül teljesíthetõ lenne azáltal, hogy az egyes rétegek ezen két feltételt teljesítenék. Ha ugyanis $F=F_1F_2...F_r$ operátori szorzatként tekintjük a teljes transzformációt, ahol F_i írja le az i-edik réteget, akkor az inverz - könnyen ellenõrizhetõen - $F^{-1}=F_1^{-1}F_2^{-1}...F_r^{-1}$ alakban állítható. Azaz, ha a rétegek invertálhatók, akkor a teljes transzformáció is invertálható. Mivel a bitpermutációs réteg invertálható lineáris transzformáció, ezért ha az S-rétegek invertálhatók, akkor a teljes transzformáció is invertálható.

A helyettesítéses-permutációs rejtjelezõk legfontosabb építõ elemei az S-dobozok. A rejtjelezõ ereje nagy mértékben függ S-dobozainak tulajdonságaitól, ezért ezek tervezése igen nagy körültekintést és tapasztalatot igényel. A megalapozott tervezés elsõ lépése a tervezési kritériumok meghatározása. Az S-doboz tervezés elsõ problémái itt kezdõdnek. Nincs ugyanis olyan elméleti módszer, amely pontos felvilágosítást ad arról, hogy egy S-doboznak milyen kritériumokat kell kielégítenie. A ma ismert kritériumok többsége nem elméleti megfontolásból született. Általában valamilyen ismertté vált támadás elleni védekezés feltételeit sikerült kritérium formájában megfogalmazni. Ezért a kritériumok halmaza "ad hoc" módon bõvült, párhuzamosan az egyre újabb és újabb kriptanalízis módszerek megjelenésével.

A kritériumok nem alkotnak kompatibilis halmazt, azaz vannak közöttük olyanok, melyeket egyszerre nem lehet kielégíteni, illetve bizonyos kritériumok kizárják az optimum elérését más kritériumok szempontjából.

A következõ lépés a kritériumokat kielégítõ S-dobozok keresése. Erre alapvetõen két út áll rendelkezésre. Az egyik a véletlen választás, a másik a konstrukció. Az elsõ esetben arról van szó, hogy véletlenül választunk S-dobozokat, majd az így nyert S-dobozok közül kiválasztjuk azokat, amelyek a tervezés során használt kritériumokat kielégítik. Utóbbi esetben valamilyen matematikai konstrukciós eljárást használunk a kívánt kritériumokat kielégítõ S-dobozok megkeresésére.

A továbbiakban néhány fontos alapfogalom bevezetése után néhány alapvetõ S-doboz tervezési kritériumot tekintünk.

Boole-függvény alatt egy m bitet 1 bitbe képzõ $f:\{0,1\}^m \to \{0,1\}$ függvényt értünk. Egy $f:\{0,1\}^m \to \{0,1\}^n$ leképzést S-doboznak nevezünk, ha $1 < n \le m$. f-et gyakran fogjuk fel úgy, mint n darab Boole-függvény együttesét:

$$f(x) = \left[f_1(x), f_2(x), \ldots f_n(x) \right]$$

ahol az $f_i:\{0,1\}^m \to \{0,1\}$ függvényeket komponens Boole-függvényeknek nevezzük.

Balansz tulajdonság

Egy bináris vektor-vektor transzformáció balansz tulajdonság lényegében azt jelenti, hogy a transzformáció nem torzítja el egy egyenletes eloszlású bemenet gyakoriság-statisztikáját. Egy $f:\{0,1\}^n \to \{0,1\}^n$ függvény esetén az invertálhatóság értelemszerûen biztosítja a balansz tulajdonságot. Általában

Egy $f:\{0,1\}^m \to \{0,1\}^n$ függvény balansz, ha tetszõleges $y \in \{0,1\}^n$ esetén

$$\#\{x \in \{0,1\}^m : f(x) = y\} = 2^{m-n}$$

Egy $f:\{0,1\}^m \to \{0,1\}$ Boole-függvény esetén a balansz tulajdonság azt jelenti, hogy f kimenete pontosan annyi x-re 0, mint amennyire 1, azaz

$$\#\{x \in \{0,1\}^m : f(x) = 0\} = \#\{x \in \{0,1\}^m : f(x) = 1\}$$

Például, egy $f(x) = \mathbf{A} \cdot x + b$, $\mathbf{A}$ egy nxm-es bináris mátrix, $b \in \{0,1\}^n$ lineáris függvény, ahol A sorai lineárisan függetlenek, balansz tulajdonságú, n=1,2,... .

Teljesség

A teljes rejtjelezõ tarnszformáció esetén elvárjuk, hogy mindegyik bemeneti bit hasson mindegyik kimeneti bitre, azaz ne fordulhasson elõ az, hogy léteznek olyan input bitek, bitcsoportok, amelyek semminemû változást nem okoznak bizonyos output bitekben bitcsoportokban, kulcstól, input blokktól függetlenül. Ezen követelményt formalizálja az úgynevezett teljesség követelmény:

Egy $f:\{0,1\}^m \to \{0,1\}^n$ függvény teljes, ha minden $1 \le i \le m$ és $1 \le j \le n$ esetén

$$\#\{x \in \{0,1\}^m : e_n^{(j)} \cdot f(x) \neq e_n^{(j)} \cdot f(x + e_m^{(i)})\} > 0$$

ahol
$x \cdot y$ az x és y bináris skalárszorzata, ha $x, y \in \{0,1\}^n$ és $n > 1$, azaz
$x \cdot y = x_1 \cdot y_1 + x_2 \cdot y_2 + ... + x_n \cdot y_n$ ahol x_i és y_i jelöli az x és y vektorok i. bitjét,

$x + y$ az x és y bitenkénti moduló 2 összege, ha $x, y \in \{0,1\}^n$ és $n > 1$;

$e_n^{(i)}$ n dimenziós egységvektor, mely egyetlen egyest tartalmaz az i. pozícióban.

A teljesség tehát azt jelenti, hogy az $f(x) + f(x + e_m^{(i)})$ függvény j. komponens Boole-függvényének igazságtáblája legalább egy darab 1-et tartalmaz, és ez minden $1 \le i \le m$ és $1 \le j \le n$ esetén igaz. Más szavakkal f minden kimeneti bitje minden bemeneti bittől függ.

Nem nehéz bebizonyítani, hogy ha egy $f: \{0,1\}^n \to \{0,1\}^n$ invertálható transzformáció teljes is, akkor az nem lehet lineáris. azaz nem lehet $y = \mathbf{A} \cdot x + b$ alakú, ahol A egy nxn méretû bináris mátrix, b pedig egy n bites vektor. Az indirekt gondolatmenet az iménti megjegyzésünk alapján arra vezetne, hogy az A mátrix minden eleme 1 kellene legyen, ami azonban ellentmondna az A mátrix, következésképpen az transzformáció invertálhatóságának.

Nemlinearitás mértéke

Mivel egy helyettesítéses-permutációs rejtjelezõ egyetlen nem-lineáris eleme a helyettesítõ réteg, ezért ez a kritérium központi szerepet játszik az S-dobozok tervezése során.

Két Boole-függvény távolságán azon bemenetek számát értjük, melyekre a két függvény kimenete különbözik egymástól. Formálisan egy $f: \{0,1\}^m \to \{0,1\}$ és egy $g: \{0,1\}^m \to \{0,1\}$ Boole-függvény $d(f, g)$ távolsága a következõ:

$$d(f, g) = \#\{x \in \{0,1\}^m : f(x) \neq g(x)\} = w(f + g)$$

Legyen $L_{u,v}: \{0,1\}^m \to \{0,1\}$ a következõ alakú lineáris Boole függvény: $L_{u,v}(x) = u \cdot x + v$ ahol $u, v \in \{0,1\}^m$, $v \in \{0,1\}$.

Ezen definíciókkal egy $f: \{0,1\}^m \to \{0,1\}$ Boole-függvény $N(f)$ nem-linearitásán az lineáris függvények halmazától vett távolságát értjük:

$$N(f) = \min_{u \in \{0,1\}^m, v \in \{0,1\}} d(f, L_{u,v})$$

A definíció alapján nem nehéz belátni, hogy fennáll az

$$N(f) \le 2^{n-1}$$

egyenlõtlenség. Ugyanis, ha $d(f, L_{u,v}) < 2^{n-1}$, akkor $d(f, L_{u,v+1}) \ge 2^{n-1}$.

Ezen definíciót felhasználva $f:\{0,1\}^m \to \{0,1\}^n$, n>1 transzformáció $N(f)$ nemlinearitását komponens Boole függvényei bináris lineárkombinációi nemlinearitásának lineárkombináció szerint képzett minimumával definiálhatjuk:

$$N(f) = \min_{w \in \{0,1\}^m} N(w \cdot f)$$

Egy tulajdonságot feltétlenül elvárunk egy nemlinearitás definíciótól: lineáris transzformációk ne növelhessék a nemlinearitást. Ezzel kapcsolatosan például nem nehéz belátni, hogy ha P és Q invertálható mátrixok a megfelelõ dimenzióval, akkor

$$N(PfQ) = N(f).$$

Egy ilyen tulajdonságot a kriptográfiai mérõszám (kritérium) *invariáns* tulajdonságának, invarianciájának nevezünk. Egy további - szintén a definícióból közvetlenül levezethetõ - tulajdonsága a fentebb definiált nemlinearitásnak az, hogy egy invertálható $f:\{0,1\}^m \to \{0,1\}^m$ transzformációnak és inverzének legyen azonos a nemlinearitása, azaz

$$N(f^{-1}) = N(f).$$

Felmerül az optimum (maximum) nemlinearitás nagyságának kérdése. Boole-függvények esetén a válasz jól ismert:

$$N(f) \le 2^{n-1} - 2^{n/2-1}.$$

Megjegyezzük, hogy a maximális nemlinearitás elérhetõ az úgynevezett a kódelméletben jólismert bent-függvényekkel. Sajnos ezen optimális nemlinearitás ellentmond más optimális kritériumnak, nevezetesen egy optimálisan nemlineáris Boole-függvény nem lehet balansz tulajdonságú.

Lavinahatás

Egy $f:\{0,1\}^m \to \{0,1\}^n$ függvény teljesíti a lavinahatás kritériumot, ha minden $1 \le i \le m$ esetén

$$\frac{1}{2^m} \sum_{x \in \{0,1\}^m} w(f(x) + f(x + e_m^{(i)})) = \frac{n}{2}$$

ahol w() a Hamming súly függvény. A lavinahatás kritérium teljesülése tehát azt jelenti, hogy a bemenet bármelyik bitjét megváltoztatva átlagosan a kimeneti bitek fele is megváltozik. Ugyanakkor ezen kritérium megengedné, hogy az output blokkban legyen olyan bitcsoport, amelyben alig van változás (a teljesség megkövetelése miatt legalább egy input blokk esetén kell változásnak elõállni az i-edik bemeneti bit billentésekor, de elõfordulhat, hogy nincs is több).

Szigorúbb kritérium a szigorú lavinahatás kritérium a SAC (Strict Avalanche Criterion), amelynek definíciója szerint egy $f:\{0,1\}^m \to \{0,1\}^n$ függvény teljesíti a lavinahatás kritériumot, ha minden $1 \le i \le m$ esetén

$$\frac{1}{2^m} \sum_{x \in \{0,1\}^m} (f(x) + f(x + e_m^{(i)})) = (\frac{1}{2}, \frac{1}{2}, \ldots \frac{1}{2})$$

Azaz a SAC azt jelenti, hogy a bemeneten egy bitet megváltoztatva a kimenet 1/2 valószínuséggel változik meg.

Megjegyezzük, hogy egy optimálisan nemlineáris Boole függvény egyben SAC tulajdonságú is.

3.4 Data Encryption Standard (DES)

A DES a legismertebb helyettesítéses-permutációs blokk rejtjelezõ, melyet az IBM szakemberei fejlesztettek ki az USA-ban a hetvenes években. Késõbb a világ több országában szabványként fogadták el, és azóta is használják. A DES kiállta az idõk próbáját, ma sem ismert olyan gyakorlatban hatékonyan alkalmazható módszer, amivel fel lehetne törni.
A DES struktúrája az 3.4.1. ábrán látható. A bemenet és a kimenet mérete 64 bit, a kulcs 56 bites. A rejtjelezõ 16 rétegbõl áll, ahol a rétegek felépítése egy kicsit eltér az 3.3.1 ábra rétegeinek klasszikus felépítésétõl. Itt az egyes rétegek bemenete két részbõl áll, jelöljük ezeket L_i -vel és R_i -vel. L_i és R_i mérete 32 bit. A jobb oldali blokk, R_i áthalad egy kulcs vezérelt, S-dobozokból és különbözõ permutációkból felépülõ F transzformáción. Az *i.* réteg kulcsát jelöljük K_i-vel. K_i 48 bites, és a kulcs ütemezõ állítja elõ az eredeti 56 bites kulcsból. Az F transzformáció kimenetét XOR-oljuk a bal oldali blokkal, L_i -vel, majd a két oldal blokkjait felcseréljük (kivéve az utolsó rétegben). Így kapjuk a következõ réteg bemenetének két blokkját. Formálisan:

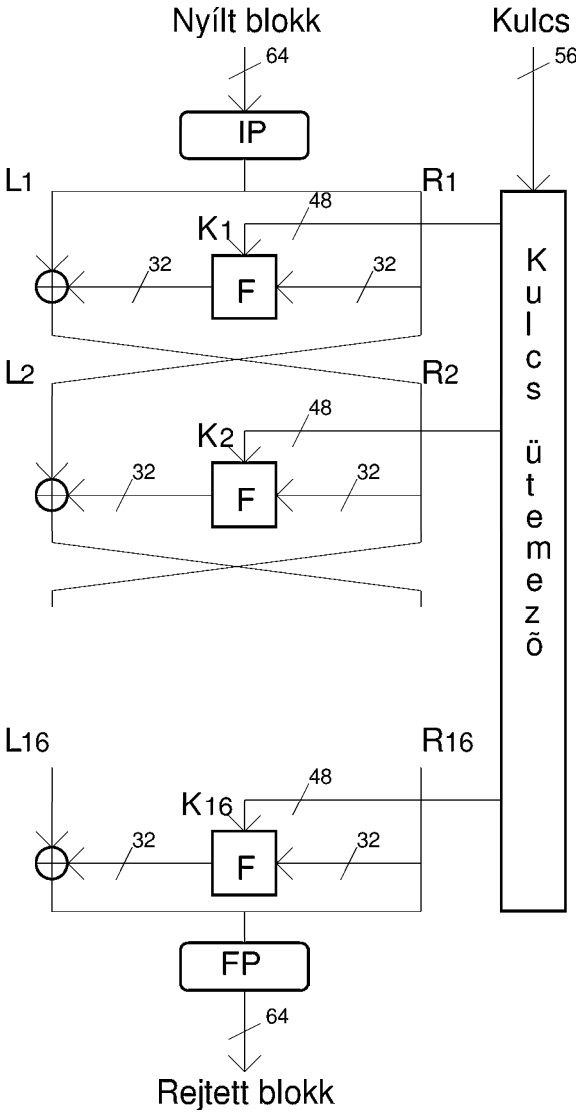
$$L_{i+1} = R_i$$
$$R_{i+1} = L_i + F\ (R_i, K_i) \qquad (3.4.1.)$$

Ezt a struktúrát Feistel-struktúrának nevezik. Az az elõnye, hogy az így felépülõ rejtjelezõ mindenképpen invertálható, függetlenül attól, hogy F invertálható, vagy sem. Vegyük észre ugyanis, hogy L_i és R_i elõállítható L_{i+1} -bõl és R_{i+1} -bõl anélkül, hogy F inverzét használnánk, hiszen (3.4.1.)-bõl:

$$L_i = R_{i+1} + F\ (L_{i+1}\ K_i)$$
$$R_i = L_{i+1} \qquad (3.4.2.)$$

Sõt a DES tervezõi még azt is elérték, hogy a dekódoló gép megegyezzen a kódolóval, csupán a kulcsütemezõt kell fordítva mûködtetni, azaz a kódolásnál az elsõ rétegben alkalmazott 48 bites kulcsot a dekódolásnál az utolsó rétegben kell használni, a kódolásnál a második rétegben alkalmazott 48 bites kulcsot a dekódolásnál az utolsó elõtti rétegben kell használni, s.í.t. Ez egyébként annak a következménye, hogy az utolsó rétegben nem kell a két félblokkot felcserélni.

Vizsgáljuk most meg a DES építõ elemeit részletesen!



3.4.1. ábra
A DES szerkezete

3.4.1 A kezdeti és a végsõ permutáció (IP és FP)

E két 64 bit bemenetû és 64 bit kimenetû bit permutációnak nincs különösebb szerepe, a DES biztonságára nincsenek hatással. Mivel szoftverben viszonylag nehezen implementálhatók, ezért gyakran el is hagyják õket, eltérve az eredeti szabványtól.
A kezdeti permutáció pontos mûködését a következõ táblázat foglalja össze. A táblázatot balról jobbra, fentrõl lefelé kell olvasni. Az IP permutáció a bemenet 58. bitjét például az 1. pozícióba, 50. bitjét a 2. pozícióba mozgatja. A végsõ permutáció a kezdeti permutáció inverze: $FP = IP^{-1}$.

58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6	64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1	59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5	63	55	47	39	31	23	15	7

3.4.1. táblázat
A kezdeti permutáció (IP)

3.4.2 A kulcsütemezõ

A kulcs ütemezõ feladata, hogy az egyes rétegek számára elõállítsa a megfelelõ méretû kulcsot. Ezt a következõképpen teszi. Elõször is az 56 bites kulcsot két 28 bites részre vágja. Mindkét felet balra rotálja egyszer vagy kétszer, attól függõen, hogy melyik réteg kulcsáról van éppen szó. Az egyes rétegekhez tartozó rotálások száma az 3.4.2. táblázatban található. A rotálás után az 3.4.3. táblázatban megadott PC permutáció szerint kiválaszt az 56 bitbõl 48-at. A rotálások következtében minden rétegben más 48 bit kerül kiválasztásra, annak ellenére, hogy a választó PC permutáció minden rétegben azonos. Ellenõrizhetõ, hogy minden bit kb. 14 rétegbe lép be. Vegyük észre továbbá, hogy az összes rotálások száma pontosan 28, így egy blokk rejtjelezése után az 56 bites kulcs az eredeti állapotába áll vissza, és kezdõdhet a következõ blokk kódolása. Ennek elsõsorban hardver megvalósításoknál van szerepe.

Réteg	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Rotálás	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

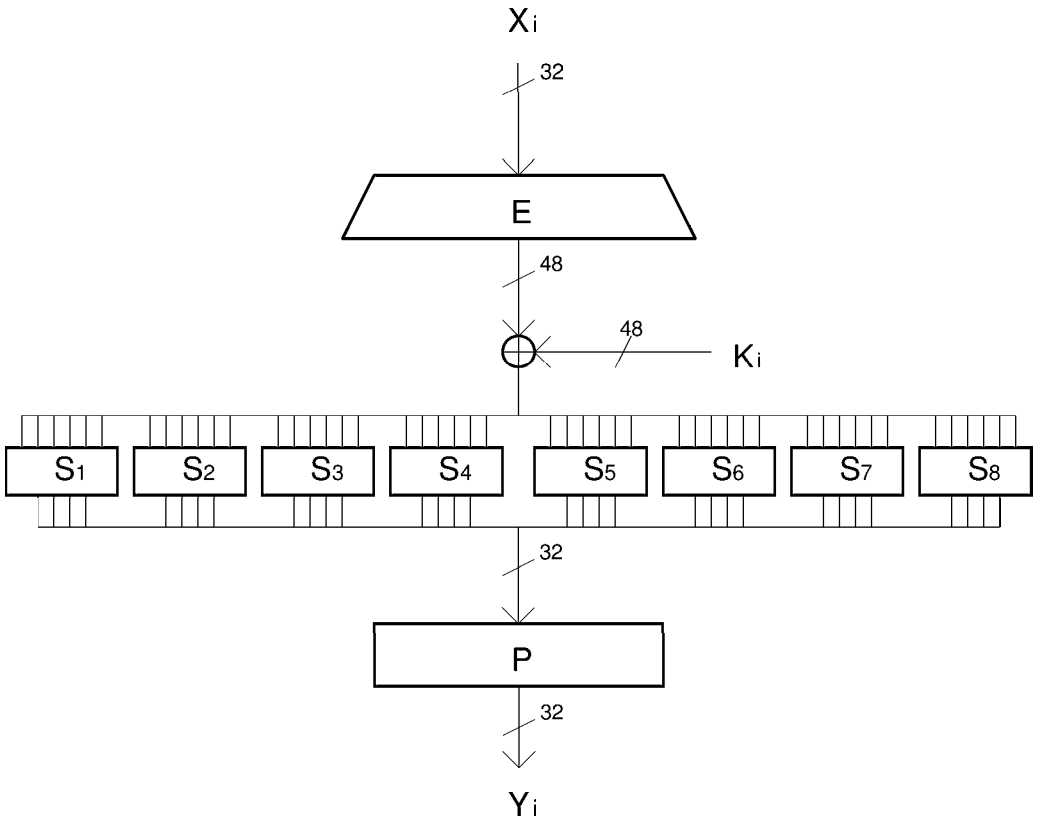
3.4.2. táblázat
Rotálások száma az egyes rétegekben

14	17	11	24	1	5	3	28	15	6	21	10
23	19	12	4	26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

3.4.3. táblázat
Kulcs választó permutáció (PC)

3.4.3 Az *F* függvény

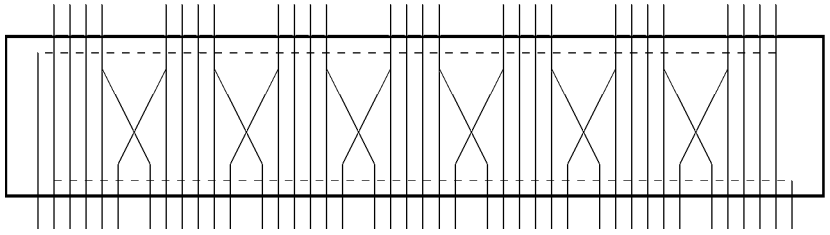
Az *F* függvény felépítését az 3.4.2 ábra szemlélteti. A bemenet elõször egy expanziós transzformáción halad keresztül, mely a 32 bites vektorból egy 48 bites vektort állít elõ. Ezt XOR-oljuk a 48 bites kulcs vektorral. A következõ elem egy S-doboz réteg, melyben 8 darab 6 bitet 4 bitbe képzõ S-doboz található. Az S-doboz réteg kimenete így 32 bites. Az *F* függvény kiemenetét végül egy 32 bites P-doboz állítja elõ.



3.4.2 ábra
Az F függvény felépítése

Az expanziós transzformáció (E)

Ez a transzformáció a 32 bites bemeneti vektorból egy 48 bites vektort állít elõ az 3.4.3 ábra illetve az 3.4.4 táblázat alapján. Látható, hogy vannak olyan bemeneti bitek, amelyek megduplázódnak, és az egyes példányok a késõbbiekben különbözõ S-dobozokba lépnek be. Így egyes bemeneti bitek több S-dobozra is hatással vannak, ami fokozza a lavinahatást, hiszen a bemeneten történõ változások így még gyorsabban terjednek tovább. Ennek következménye, hogy viszonylag kevés réteg után már minden bit minden bemeneti bittõl függ, azaz már a néhány rétegére redukált DES is teljes.



3.4.3. ábra
Az expanziós transzformáció szemléltetése

32	1	2	3	4	5	4	5	6	7	8	9
8	9	10	11	12	13	12	13	14	15	16	17
16	17	18	19	20	21	20	21	22	23	24	25
24	25	26	27	28	29	28	29	30	31	32	1

3.4.4 táblázat
Az expanziós transzformáció (E)

Az S-dobozok (S_i)

Minden S-doboz 4 darab 4 bitet 4 bitbe helyettesítõ táblázatból áll. Ezen táblázatok közül az S-doboz bemenetének két szélsõ bitje választja ki az aktuálisan alkalmazandót. Úgy is elképzelhetjük, hogy az S-dobozok 4 sorból és 16 oszlopból álló táblázatok, és a bemenet két szélsõ (tehát az 1. és a 6.) bitje címzi meg a sort, a középsõ 4 (tehát a 2-5) bit pedig az oszlopot. Az S1 dobozt láthatjuk 3.4.5 táblázatban.

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	5	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

3.4.5 táblázat
Az S_1 doboz

A P-doboz (P)

A P-doboz egy egyszerû bit permutáció, melynek mûködését az 3.4.6 táblázatból olvashatjuk ki. A P-doboz feladata a bitek összekeverése oly módon, hogy egy S-doboz kimeneti bitjei a következõ rétegben más S-dobozok bemeneti bitjei legyenek.

16	7	20	21	29	12	28	17	1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9	19	13	30	6	22	11	4	25

3.4.6 táblázat
A P-doboz (P)

3.4.4 A DES tervezése során alkalmazott tervezési kritériumok

Miután a differenciális kriptanalízis nyilvánosan is ismertté vált (a kilencvenes évek elején), a DES tervezõi elismerték, hogy a DES tervezésekor (a hetvenes évek elején!) õk már ismerték ezt a támadási módszert és mind az S-dobozokat, mind a rétegek számát úgy választották meg, hogy a DES ellenálljon a differenciális kriptanalízisen alapuló támadásnak. Bár nem árulták el, hogy vajon ez volt-e a legerõsebb támadás, aminek tudatában voltak, és ami ellen felkészítették a DES-t, az S- és P-doboz tervezés során használt kritériumokat felfedték:

S-doboz tervezési kritériumok

- Minden S-doboz bemenete 6, kimenete 4 bites legyen. (Az akkori technológia korlátai maximálisan ekkora méretû S-dobozokat engedtek meg. Ezek a méretek lehetõvé tették, hogy a DES-t egyetlen chip-be integrálják.)

- Egyetlen S-doboz egyetlen kimeneti bitje se legyen közel a bemeneti bitek valamely lineáris függvényéhez. (Tehát a nem-linearitás legyen nagy.)

- Ha rögzítjük a két szélsõ bit értékét, és csak a bemenet középsõ négy bitjét változtatjuk folyamatosan, akkor a kimeneten minden 4 bites vektor pontosan egyszer jelenjen meg. (Azaz az S-dobozban található 4 darab 4 bitet 4 bitbe helyettesítõ tábla mindegyike legyen balansz. Ekkor persze maga az S-doboz is balansz, vagyis mineden 4 bites kimeneti vektor pontosan négyszer jelenik meg, ha a bemeneten minden lehetséges értéket végigpörgetünk.)

- Ha az S-doboz bemenetén egyetlen bitet megváltoztatunk, akkor a kimeneten legalább két bit értéke változzon meg. (Lavinahatás.)

- Ha az S-doboz bemenetén a két középsõ bitet megváltoztatjuk, akkor a kimeneten legalább két bit értéke változzon meg.

- Ha két bemeneti vektor elsõ két bitje különbözõ, utolsó két bitje azonos, akkor a megfelelõ kimeneti vektorok nem lehetnek azonosak.

- Tetszõleges, nem nulla bemeneti differencia esetén, az adott differenciával rendelkezõ 32 bemeneti vektor pár közül legfeljebb nyolchoz tartozhat azonos kimeneti differencia. (Azaz a differenciális egyenletesség legyen nagy.)

- Az elõzõhöz hasonló kritérium de egyszerre három S-dobozra.

P-doboz tervezési kritériumok

- A P-doboz legyen olyan, hogy minden S-doboz négy kimeneti bitje közül kettõt a következõ réteg S-dobozainak középsõ bitjeihez, kettõt pedig szélsõ (táblázat választó) bitekhez továbbítson.
- Minden S-doboz négy kimeneti bitje a következõ rétegben hat különbözõ S-dobozra legyen hatással.
- Ha egy S-doboz valamely kimeneti bitje egy másik S-doboz valamely középsõ bitjéhez van vezetve, akkor ez utóbbi S-doboz egyetlen kimenete sem lehet az elõzõ S-doboz középsõ bemeneteihez vezetve.

4. Nyilvános kulcsú titkosítás

4.1 A nyilvános kulcsú titkosítás alapelvei

A nyilvános kulcsú titkosítás alapgondolata, hogy gyakorlati titkosságot tud nyújtani anélkül, hogy a titkos üzenetküldést megelõzõen a kommunikáló partnerek bármiféle titkos kulcsot cseréltek volna egymással, mint ahogy ez az elõzetes kooperáció elõfeltétel a hagyományos titkosítás esetén. Egy nyilvános kulcsú titkosító két kulccsal dolgozik, egy nyilvános k^p és egy titkos k^s kulccsal. A nyilvános kulcsot a kódoláshoz, a titkos kulcsot a dekódoláshoz használjuk. Ha A és B titkosítók között történik titkos üzenetváltás, akkor B az $y = E_A(x)$ $(= E(x, k^p_A))$ rejtett üzenetet küldi A-nak, amit k^p_A nyilvánossága miatt megtehet, s ebbõl a rejtett üzenetbõl $x = D_A(y)$ $(= D(y, k^s_A))$ dekódolással nyeri vissza A a nyílt üzenetet. A kódolás a nyilvános kulcs ismeretében "könnyû" feladat, míg a dekódolás a rejtett kulcs ismeretének hiányában gyakorlatilag nem végrehajtható ("nehéz feladat").

Vegyük észre, hogy ez nem egy elvileg tökéletes kriptorendszer, hiszen elvileg nyilván mód lenne nyilvános kulcs birtokában az összes (nyílt üzenet, rejtett üzenet) pár elõzetes kiszámítására. Ezeket tárolva és a rejtett üzenetek valamilyen sorrendjében felsorolva, tetszõleges rejtett üzenet vételekor kiolvashatnánk a nyílt párját! Ez azt jelenti, hogy a rejtett üzenet minden információt tartalmaz a nyílt üzenetre vonatkozóan.

Egy A, B, C, ... titkosítókat (felhasználókat) tartalmazó rendszerben a k^p_A, k^p_B, ... nyilvános kulcsokat egy nyilvános kulcstárba tesszük le, ahonnan bármelyik felhasználó kiolvashatja annak a felhasználónak a nyilvános kulcsát, akinek rejtett üzenetet kíván küldeni. Az egyes felhasználók maguk generálhatják a (k^p, k^s) kulcspárt, amelybõl a nyilvános részt közzéteszik, a másikat titokban tartják. Nagyon fontos észrevennünk azt a kitételt, miszerint a nyilvános kulcstárból csak olvasni szabad, s védeni kell azt a nyilvános kulcsokkal történõ manipulációktól (pl. cserétõl). Ez azt jelenti, hogy ugyan a kommunikációt megelõzõen nem kell titkos kulcscserét végezni, hiszen a nyilvános kulcstárból olvasás egy nyílt elõzetes információcserének felel meg. Megmarad azonban az a feladat, hogy ezen nyílt elõzetes információ hitelességét biztosítani kell. Ilyen módon tulajdonképpen a nyilvános kulcs hitelességének algoritmikus biztosítása továbbra is feladat marad.

Hitelesítési feladatban jól alkalmazható a D dekódoló transzformáció. Tegyük fel, hogy A az x üzenetet kívánja B-nek elküldeni olyan módon, hogy egyúttal "aláírását" is elhelyezze a rejtett üzenetben. Ezt úgy teheti meg, hogy az

$y = E_B(D_A(x))$

alakú üzenetet küldi el. y alapján a $D_A(x)$ tartalmat csak B tudja dekódolni, s nyilván $D_A(x)$ az a leképezés, amely egyértelmûen kapcsolódik a küldõ személyéhez, A-hoz és az x küldött üzenethez. A dekódoló transzformációnak a nyílt üzenetre történõ alkalmazásával digitális aláírást generálhatunk. Hasonló aláírások hitelesíthetik a nyilvános kulcstár elemeit. Ezen alkalmazásokra a kriptográfiai protokollok kapcsán még visszatérünk.

A. Shamir-tól származik az a rendkívül érdekes eljárás, amelynek a felhasználásával mindennemû elõzetes kulcscsere nélkül titkos üzenetváltás történhet partnerek között, feltéve hogy az esetleges behatoló csak passzív behatolásra, azaz csak a csatornán folyó üzenetváltás lehallgatására képes. Ez az eljárás minõségi különbséget jelent a nyilvános kulcsú algoritmusokhoz képest is, hiszen még nyílvános információt sem kell elõzetesen a partnerek tudomására hozni. Az eljárás elve nagyon szemléletesen is leírható. Képzeljük el, hogy A egy lelakatolható ládába helyezi el az x üzenetet, s lelakatolja saját kulcsával (1. lépés), majd elküldi B-nek. B nem próbálkozik a nyitás számára is lehetetlen feladatával, hanem inkább még a saját kulcsával is lelakatolja a ládát (2. lépés), majd visszaküldi azt A-nak. A leveszi a saját lakatját, s a ládát, amelyen már csak B lakatja maradt, visszaküldi B-nek (3. lépés), aki ezután már könnyen kinyíthatja azt.

Egy ilyen rendszerben a konkrét felhasználók mindegyike egy titkos kulccsal rendelkezik, és felteszik, hogy a rendszerben alkalmazott $E(x)$ kódoló transzformáció kommutatív, azaz tetszõleges x nyílt üzenet, titkos k_A és k_B kulcspár esetén

$$E_A(E_B(x)) = E_B(E_A(x)). \qquad (4.1)$$

Ez azt jelenti, hogy kétszeres kódolás eredménye független a kulcsválasztás sorrendjétõl. Ekkor az ún. "háromlépéses" eljárás alkalmazásával az A felhasználó B-nek a következõképp küldheti el az x üzenetet:

$$\begin{aligned} &1.\ A \rightarrow B:\ y_1 = E_A(x) \qquad (4.2)\\ &2.\ B \rightarrow A:\ y_2 = E_B(E_A(x))\ (= E_A(E_B(x)))\\ &3.\ A \rightarrow B:\ y_3 = D_A(y_2) = E_B(x) \end{aligned}$$

A fentiekbõl úgy tünhet, hogy megtaláltuk a tökéletes megoldást, hiszen valóban nem kellett elõzetes kulcscsere a partnerek között (sem nyilvános, sem titkos). Ezen protokoll praktikus alkalmazhatóságát az gátolja, hogy teljesen ki van szolgáltatva az aktív támadásnak. A fenti szemléletes leírásmód szóhasználatát követve gondoljunk csak arra, hogy a postás, akivel a ládát kívántuk B-hez eljuttatni a saját lakatját teheti a ládára a 2. lépésben, s így adja azt vissza A-nak. Miután A a 3. lépésben leveszi a saját lakatját, nyilván a postás kinyíthatja a ládát.

Az (4.1) kommutativitási követelménynek eleget tesz például az egyszerû

$$E(x) = x+k \pmod 2$$

bitenkénti modulo 2 vektorösszeadást alkamazó kódolás, ugyanis

$(x+k_A)+k_B = (x+k_B)+k_A \pmod 2$,

mivel a mod 2 összeadás kommutattív. Ez a választás különösen jól mutatja a rendszer megválasztása során elengedhetetlen óvatosságot. Láttuk, hogy tökéletes biztonságot jelent mind A, mind B számára a véletlen átkulcsolás alkalmazása. Magában a fenti háromlépéses eljárásban ilyen átkulcsolás azonban még a pusztán rejtett szövegre alapozó passzív támadásnak sem áll ellen. Ugyanis az eljárás 1., 2., 3. lépése során megfigyelve az

$y_1 = x+k_A$,
$y_2 = x+k_A+k_B$,
$y_3 = x+k_B$

rejtett üzeneteket, azok mod 2 összege:

$y_1+y_2+y_3 = x$,

azaz az x nyílt üzenet. Meg kell jegyeznünk, hogy az alkalmazott kódolás nem véletlen átkulcsolás (one time pad), hiszen kétszer is használtuk ugyanazon kulcsot.

Hasonlóan kommutatív egy

$E(x) = x^e \bmod n$

alakú kódolás (modulo hatványozás), ahol x , e , n természetes számok. Nyilván

$(x^{e1})^{e2} = (x^{e2})^{e1} \pmod n$).

Ilyen típusú mûveletet használ az alábbiakban kifejtendõ RSA kódolás.

4.2 Az RSA blokk-titkosítás

4.2.1 Az RSA algoritmus

Ebben a rendszerben a felhasználó maga választja meg a kódoló és a dekódoló kulcsát, mégpedig a következõ szabályok figyelembevételével.

Kulcsválasztás:

1. Elõször véletlenszerûen választ két nagy, p_1 és p_2 primszámot ($p_1 \neq p_2$). Jelenleg "nagynak" a legalább 500 bit bináris méretû számokat tekintik.

2. Kiszámítja az $m = p_1 \cdot p_2$ modulust és a $\Phi(m) = (p_1-1)(p_2-1)$ szorzatot. Választ továbbá egy véletlen e számot, amelyik mind (p_1-1)-hez, mind (p_2-1)-hez, tehát $\Phi(m)$-hez relatív prim.

3. Kiszámítja az e inverzét modulo $\Phi(m)$, azaz keres egy d számot, amelyre $1 < d < \Phi(m)$ és $ed = 1 \pmod{\Phi(m)}$. Ilyen szám mindig létezik, ahogy ezt késõbb látni fogjuk.

Központi nyilvántartás:

A kulcsválasztás a 3. lépéssel be is fejezõdik. A rendszerszerû üzemeltetés során azonban egy központi, minden felhasználó által hozzáférhetõ nyilvántartásra van szükség. Ennek érdekében a kulcs nyilvános részét, esetünkben a

$$k^p = (m, e)$$

számpárt nyilvánosságra hozzák, a kulcs titkos részét, a

$$k^s = (d, p_1, p_2)$$

számhármast, s ezzel együtt a $\Phi(m) = (p_1 - 1)(p_2 - 1)$ számot titokban tartjuk. A rendszer használhatóságához tartozik a kódoló-dekódoló algoritmus központi rögzítése és nyilvános ismerete.

Kódolás:

1. Ha az A felhasználó a B felhasználónak akar üzenni, akkor kikeresi a B nyilvános kulcsát, az $e = e_B$ és $m = m_B$ számokat.

2. Elõkódolja az üzenetet: A nyílt szöveg valamilyen karakterkészlet elemeibõl áll. A rejtjelezéshez ezt a karaktersorozatot olyan nem-negatív egészek sorozatává kell átalakítani, amelyek mindegyike kisebb, mint m. Ez egy kölcsönösen egyértelmû transzformáció, amelyet minden felhasználó ismer.

3. A rejtjelezést az elõkódolt üzenet egymás utáni számain, egyenként hajtjuk végre.

4. Ha az elõkódolt szöveg következõ száma x, akkor a megfelelõ rejtjeles szám:

$y = E_B(x) = x^e \pmod m$, ahol $e = e_B$, $m = m_B$.

Dekódolás:

1. A B felhasználó kap egy rejtjeles üzenetet. Ez az üzenet szükségszerûen a 0 és m_B-1 közti egész számok egy y_1, y_2, ... sorozata.

2. A dekódolást a számsorozat elemein külön-külön hajtja végre.

3. Legyen a rejtjeles szöveg következõ száma y . Ekkor az "elõkódolt" üzenet megfelelõ száma

$x = D_B(y) = y^d \pmod m$, ahol $d = d_B$ és $m = m_B$, $0 < x < m_B - 1$.

Megjegyzés: A hatványozás azonosságait használva:

$y^d = (x^e)^d = x^{ed} = x^{\Phi(m)h + 1}$

teljesül valamely pozitív h egészre. Látni fogjuk, hogy

$x^{\Phi(m)h+1} = x \pmod m$

mindig teljesül, így a dekódolás eredményes.

4. A visszaállított "elõkódolt" $\{x_1, x_2, ...\}$ sorozatból az elõkódolás inverzét alkalmazva adódik a valódi nyílt szöveg.

Amennyiben a "háromlépéses" eljárás kerül alkalmzásra, akkor A a saját nyilvános kulcsával üzen B-nek. B ezt szintén a saját nyilvános kulcsával rejtjelezve küldi vissza A-nak, az alkalmazza a saját titkos dekódoló kulcsát, így az üzenet ezután csak B nyilvános kulcsával lesz rejtjelezve. Ezt B a saját kulcsával már könnyen dekódolhatja. (Meg kell azonban egy kisebb technikai problémát oldani, amely abból adódik, hogy m_A és m_B, a két modulus különbözik.)

Technikai okokból célszerû úgy megválasztani az elõkódolási transzformációt, hogy azt valamennyi felhasználó alkalmazhassa. Ez egyszerûen elérhetõ, ha elõírnak egy M alsó korlátot a megengedett m modulusok számára. Ha ekkor az elõkódolás során 0 és M - 1 közé esõ számokat generálnak, akkor az elõkódolt sorozat univerzálisan tovább kódolható lesz.

A kulcsválasztásnak elfogadható idõn belül kivitelezhetõnek kell lennie. Ezért meg kell vizsgálni, hogy milyen nagy sebességgel tudunk "nagy" prímszámokat generálni. Prímelõállítással a fejezet további részében részletesebben foglalkozunk. Ha olyan prímet választunk e értékéül, mely nagyobb, mint $\max(p_1 - 1, p_2 - 1)$, akkor az automatikusan relatív prím lesz $\Phi(m)$-hez. Enélkül is könnyû eldönteni, hogy $\Phi(m)$ és e relatív prímek-e, amely vizsgálathoz az Euklideszi algoritmus használható. Ugyanakkor vegyük észre, hogy legfeljebb $[\log\Phi(m)]/\Phi(m)$ annak a valószínûsége, hogy egy véletlenszerûen választott $0 < e < \Phi(m)$ egész nem relatív prím $\Phi(m)$-hez.

A véletlenszerûen választott e szám inverzét gyorsan meg kell tudni határozni. Erre a célra az Euklideszi algoritmus használható, amit a következõ pontban ismertetünk. Az algoritmus legfeljebb kétszer annyi lépést igényel, mint a modulus logaritmusa.

A kódolásnak és a dekódoló kulcs ismeretében a dekódolásnak is gyorsnak kell lennie. Ez azt jelenti, hogy a modulo aritmetikában gyorsan kell tudni hatványozni. Az ismételt négyzetreemelés és szorzás módszerét alkalmazva az e-edik hatvány kiszámítása legfeljebb $2\cdot\log_2(e)$ $(2\cdot\log_2(d))$ számú modulo m szorzással megoldható. Például $m \approx 10^{200}$ nagyságrend esetén is legfeljebb $2\cdot200\cdot\log_2(10) < 1400$ szorzást kell végezni.

A nyilvános kulcsú rejtjelezéssel szemben támasztott alapkövetelmény, hogy az illetéktelen fejtés "nehéz" feladat legyen, aminek megoldása gyakorlatilag kivitelezhetelen. Esetünkben ez megköveteli, hogy

- m törzstényezõs elõállítása nehéz legyen,
- d közvetlen meghatározása (e, m) ismeretében gyakorlatilag m törzstényezõs elõállítását igényelje.

Mai ismereteink szerint e és m nyilvános adatok birtokában az RSA-kódoló

$$f(x) = x^e \pmod m$$

leképezés inverzének kiszámítása (azaz egy az $y = f(x) \pmod m$ összefüggésnek megfelelõ x meghatározása) számításigényét tekintve gyakorlatilag megoldhatatlan feladat, ha p_1 és p_2 prímeket elegendõen nagyra választjuk. Erõs sejtés, hogy ez az inverzképzés nehézsége ekvivalens az m prímfaktorokra bontásának nehézségével. A faktorizáló algoritmusok sebessége évrõl évre javul. 1999-ben már az 512 bit modulus sem tekinthetõ biztonságosnak.

A modulo hatványozás a fentebb említett megfelelõ paraméter-nagyságrendek mellett példája az egyirányú függvényeknek:

4.1 Definició: *Az invertálható f függvényt egyirányúnak nevezzük, ha értelmezési tartományának tetszõleges x elemére f(x) értéket könnyû kiszámítani, míg gyakorlatilag irreális feladat egy általános értékkészletbeli y elemhez az y = f(x)-nek megfelelõ x kiszámítása.*

Az RSA kódolás esetén könnyû feladatnak számít a kódolás, míg a dekódolás csak a titkos dekódoló kulcs ismeretében könnyû.

4.2 Definició: *Azon egyirányú függvényeket, amelyeket egy információ birtokában könnyû, de annak hiányában gyakorlatilag lehetetlen invertálni, csapda típusú egyirányú függvényeknek nevezzük.*

Ezzel a szóhasználattal élve csapda típusú egyirányú függvény birtokában elvileg tervezhetünk nyilvános kulcsú titkosító algoritmust.

4.2.2 Példa az RSA titkosításra

Példánk célja a szemléltetés. Ehhez a rejtjelezõi gyakorlat számára elfogadhatatlanul kis paramétereket kellett választanunk.
Két felhasználó, A és B megválasztja a paramétereit:

A: $p_1 = 13,\ p_2 = 19,$
$m_A = 13 \cdot 19 = 247,$
$\Phi(m_A) = 12 \cdot 18 = 216$

B: $p_1 = 11,\ p_2 = 23,$
$m_B = 11 \cdot 23 = 253,$
$\Phi(m_B) = 10 \cdot 22 = 220$

Választanak véletlenszerûen egy-egy nyilvánosságra hozandó *e* komponenst, s kiszámítják a hozzá tartozó titkos komponenst:

$e_A = 65, \quad d_A = 113, \quad 65 \cdot 113 = 7345 = 1 \pmod{226}$
$e_B = 17, \quad d_A = 13, \quad 17 \cdot 13 = 221 = 1 \pmod{220}$

A titkos exponens kiszámítására létezik "gyors" eljárás. Ez a késõbbiekben ismertetésre kerülõ Euklideszi algoritmus.

Így A az $e_A = 65$ és $m_A = 247$, B az $e_B = 17$ és $m_B = 253$, paramétereket hozza nyilvánosságra. Tegyük fel, hogy A a következõ üzenetet akarja B-hez eljuttatni:

"1989. január 24-én 100 USA dollár vételi ára 5197,45 forint lesz."

A rejtjelezésben különösen fontos, hogy tömören, röviden fogalmazzunk a tartalom megváltoztatása nélkül. Ezért A úgy dönt, hogy a

"89.01.24: 100 USA$ = 5197,45 Ft"

üzenetet küldi el. A nyilvántartásból ismeri, hogy B modulusa 253, ezért ezt a sorozatot egyértelmûen dekódolható módon át kell alakítani 0 és 252 közti számok sorozatává. AZ ASCII kód felhasználásával alakítsuk át a szöveget 0 és 255 közti számok sorozatává:

56, 57, 46, 48, 49, 46, 50, 54, 58,
49, 48, 48, 32, 85, 83, 65, 36, 61,
53, 49, 57, 55, 46, 52, 53, 70, 116.

Ezután a 256-os számrendszerbõl a 253-as számrendszerbe kell konvertálni. Ha ezt minden számra külön-külön hajtjuk végre, akkor az eredeti számokat a 253-as számrendszerben két számjeggyel kell leírni, ahol az elsõ jegy három szám kivételével 0 és csak

$$253 = 10|_{253},\quad 254 = 11|_{253},\quad 255 = 12|_{253}$$

esetekben lesz 1. Ugyancsak értelmetlenül növelnénk így kétszeresére az üzenetet. Célszerûbb több számot egyidejûleg konvertálni: Példánkban az egyes sorokat úgy értelmezzük, hogy azok számok 256-os számrendszerben felírt számok számjegyei.
Ebben az értelemben például az elsõ sor az

$$N_1 = 56 + 57\cdot 256 + 46\cdot 256^2 + 48\cdot 256^3 + 49\cdot 256^4 + 46\cdot 256^5 + 50\cdot 256^6 + 54\cdot 256^7 + 58\cdot 256^8$$

számot adja meg. Határozzuk meg a 256 hatványainak 253 alapú számrendszerbeli számjegyeit. Ekkor az alábbi 253-as számrendszerbeli számokat kapjuk :

124, 222, 226, 99, 115, 206, 4, 245, 63
52, 169, 135, 219, 207, 139, 150, 47, 67
96, 171, 194, 163, 39, 16, 144, 195, 127,

ahol egy sor egy szám az üzenet sorainak megfelelõen. Ezután kerülhet sor a rejtjelezés hatványozás lépésére, amelynél a 124, 222,... számok 17 -dik hatványát kell képezni. Az elõkódolt sorozatból ezzel az algoritmussal a következõ rejtjeles sorozat adódik:

141, 194, 228, 88, 69, 68, 71, 240, 57
35, 82, 53, 32, 92, 116, 193, 185, 166
2, 63, 17, 202, 96, 234, 12, 244, 239.

Ezt az üzenetet kapja kézhez a B felhasználó. Az õ megoldó kulcsa $d = 13$, tehát a 13-adik hatványt kell elõállítania a 253 modulus mellett. Ezután a 256-os számrendszerbe, majd az ASCII táblázat segítsével értelmes szöveggé kell visszaalakítania a számokat.

A fenti elõkódolás probléma kevésbé élesen jelentkezik, ha a rendszerben rögzített N bites bináris üzenetblokkokkat kódolunk egy lépésben, s ehhez az eredeti üzenetet ekkora szeletekre bontjuk. Ezt úgy érhetjük el, hogy a p_1 illetve p_2 primeket a $[2^{N/2}, 2^{N/2+1}-1]$ tartományból választjuk véletlenszerûen, mivel ekkor

$$2^N < m < 2^{N+1}-1$$

fennáll, azaz az egy lépésben kódolható üzenet egy 2^N-nél kisebb számnak felel meg.

4.2.3. Számelméleti elemi segédeszközök

Az RSA algoritmus áttekinthetõsége, mûködésének alaposabb megértése érdekében felidézünk néhány elemi számelméleti eredményt. Az RSA algoritmus számítástechnikai megvalósításának fontos eleme a jól ismert maradékos osztás.

4.1 Tétel (A maradékos osztás tétele): *Tetszõleges a és b, $a > 0$, $b > 0$ egészekre egyértelmûen létezik q és r egész, hogy*

$$a = b \cdot q + r \qquad (4.3)$$

ahol $0 < r < b$, $q > 0$.

Bizonyítás: A létezés könnyen látható, hiszen a $q \cdot b < a < (q + 1) \cdot b$ egyenlõtlenségnek eleget tevõ q egészet választhatjuk, s ekkor $r = a - q \cdot b$. Az egyértelmûség igazolásához tegyük fel, hogy q , r páron kívül létezik egy q' , r' pár is, amelyre $a = q' \cdot b + r'$, $0 < r' < b$. Mivel ekkor $q \cdot b + r = q' \cdot b + r'$ állna fenn, ezért az $r - r' = b \cdot (q' - q)$ átrendezéséből ellentmondásra jutunk, hiszen $0 < r, r' < b$ miatt $r - r' < b$, és ugyanakkor teljesülnie kell a $b \mid r - r'$ oszthatóságnak, ami csak $r = r'$ esetén lehetséges. Az $r = r'$ egyenlõségbõl viszont $q = q'$ is következik.
■

Az elemi számelmélet egyik fontos fogalma a közös osztó.

4.3 Definició: *Az a számot a b és c szám közös osztójának nevezzük, ha $a \mid b$ és $a \mid c$ teljesül. Ha b és c közül legalább egyik nem nulla, akkor a közös osztóik legnagyobbikát b és c legnagyobb közös osztójának nevezzük és (b, c)-vel jelöljük.*

4.4 Definició: *Azt mondjuk, hogy b és c relatív primek, ha (b, c) = 1.*

A legnagyobb osztót ránézésre általában nem lehet kitalálni. Ha elkészítenénk mind a b , mind a c prímfelbontását, akkor a közös prímosztók segítségével gyorsan felírhatnánk (b, c)-t. Azonban a prímfelbontás (faktorizálás) bonyolult feladat, így nagy b , c esetén ez az út nem járható. Viszont már a görögök is ismertek célhoz vezetõ eljárást. Ennek alapja a maradékos osztás. Tegyük fel, hogy $b > c$.

A $b = c \cdot q + r$ és az ekvivalens $b - c \cdot q = r$ $(0 < r < c)$ egyenletek közül az elsõbõl láthatóan c és r közös osztói osztói b-nek, míg a másodikból láthatóan b és c közös osztói osztói r -nek. Azaz b és c közös osztóinak halmaza azonos c és r közös osztóinak halmazával, s így ezen halmazok minimális elemei is azonosak, tehát
$(b , c) = (r, c)$.

Ezzel ugyan nem oldottuk meg a feladatot (hacsak nem $r = 0$), de kisebb méretûre vezettük vissza: $c < b$ és $r < c$. Ezt a redukciót mindaddig ismételjük, amíg pozitív maradékot kapunk.

4.2 Tétel (Az euklideszi algoritmus): *Adott b és c > 0 egészekre ismételten alkalmazzuk a maradékos osztást, s ezzel az egyenletek következõ sorozatát kapjuk:*

$$
\begin{aligned}
b &= c \cdot q_1 + r_1, && 0 < r_1 < c \\
c &= r_1 \cdot q_2 + r_2, && 0 < r_2 < r_1 \\
r_1 &= r_2 \cdot q_3 + r_3, && 0 < r_3 < r_2 \\
&\ldots \\
r_{n-2} &= r_{n-1} \cdot q_n + r_n, && 0 < r_n < r_{n-1} \\
r_{n-1} &= r_n \cdot q_{n-1} + 0
\end{aligned}
\qquad (4.4.)
$$

A b és c számok legnagyobb közös osztója r_n , amely az osztási eljárás legutolsó nemnulla maradéka, azaz $(b ,c) = r_n$.

Bizonyítás: r_1, r_2, ... pozitív egészek szigorúan monoton csökkenõ sorozata, ezért léteznie kell olyan egésznek, amelyre $r_{n+1} = 0$. A $b = c \cdot q_1 + r_1$ egyenletre tekintve láthatjuk, hogy ha $x|b$ és $x|c$ fennáll, akkor $x|r_1$ is fenn kell álljon, s viszont, ha $y|c$ és $y|r_1$ akkor ebbõl $y|b$ is következik. Ez pedig azt jelenti, hogy b, c közös osztóinak halmaza megegyezik c, r közös osztóinak halmazával, ezért $(b, c) =$
(c,r_1) is igaz. Az euklideszi algoritmus további egyenleteit is tekintve a

$$(b, c) = (c, r_1) = (r_1, r_2) = \ldots = (r_{n-1}, r_n) = (r_n , 0) = r_n$$

egyenlõségi láncot kapjuk, tehát $(b, c) = r_n$ valóban igaz.
■

Felhívjuk a figyelmet arra az alkalmazási szempontból rendkívül fontos tényre, hogy ilyen algoritmussal (b, c) legfeljebb $2\log(\max\{b, c\})$ maradékos osztással meghatározható.

Az 4.2.2 fejezetbeli példában az e_A illetve e_B exponenseket úgy kellett megválasztani, hogy $\Phi(m_A)$-hoz és $\Phi(m_B)$-hez relatív prímek legyenek, azaz

$(e_A, \Phi(m_A)) = 1$
$(e_B, \Phi(m_B)) = 1$

teljesüljenek. Számpéldaként ellenõrizzük ezeket:

$200 = 17 \cdot 12 + 16$ $\quad$ $216 = 65 \cdot 3 + 21$
$17 = 16 \cdot 1 + 1$ $\quad$ $65 = 21 \cdot 3 + 2$
$16 = 1 \cdot 16 + 0$ $\quad$ $21 = 2 \cdot 10 + 1$
$\quad$ $\quad$ $2 = 1 \cdot 2 + 0$

Tehát minkét számpár relatív prim: $(220, 17) = 1$ és $(216, 65) = 1$.

Az algoritmus alkalmas arra is, hogy két szám legnagyobb közös osztóját a számok lineáris kombinációjaként állítsuk elõ:

Következmény: Tetszõleges b és c egészekre, amelyek közül legalább egyik nem nulla, léteznek s és t egészek, hogy

$$(b, c) = s \cdot b + t \cdot c \qquad (4.5)$$

Bizonyítás: Az euklideszi algoritmus egyenleteibõl egymásba helyettesítésekkel a következõ egyenletsort kaphatjuk:

$r_1 = b - q_1 \cdot c = b + (-q_1) \cdot c$,
$r_2 = c - q_2 \cdot r_1 = (-q_2) \cdot b + (1 + q_1 q_2) \cdot c$,
. . .
$r_n = s \cdot b + t \cdot c$,

azaz az n-edik lépésben a kívánt alakra jutottunk.
■

Fenti számpéldánkat folytatva kapjuk:

$16 = b - 12c$
$1 = c - 16 = c - (b - 12c) =$
$= -b + 13c,$

$21 = b - 3c$
$2 = c - 21 \cdot 3 = c - 3 \cdot (b - 3c) =$
$= -3b + 10c$
$1 = 21 - 2 \cdot 10 = (b - 3c) - 10(-3b + 10c)$
$= 31b - 103c$

azaz

$(220, 17) = -220 + 13 \cdot 17$

$(216, 65) = 31 \cdot 216 - 103 \cdot 65$

Az oszthatóság szempontjából hasonló, azaz kongruens számok közti számolással foglalkozik a moduláris aritmetika.

4.5 Definició: *Ha az m nem nulla egész szám osztja a b - c különbséget,akkor azt mondjuk, hogy a b kongruens c modulo m, és ezt a következõképpen jelöljük: b = c (mod m).*

4.6 Definició: *Azt mondjuk, hogy b modulo m inverze c, ha*

$$b \cdot c = 1 \pmod{m} \qquad (4.6)$$

Az elõzõek szerint 13 modulo 220 inverze 17, míg 65 modulo 216 inverze 113.

4.3 Tétel: *A b szám modulo m inverze akkor és csak akkor létezik, ha (b, m) = 1. Ha létezik inverz, akkor az egyértelmû az m-nél kisebb pozitív egészek között.*

Bizonyítás: Ha létezik olyan c, amelyre $b \cdot c = 1 \pmod{m}$, akkor az ezzel ekvivalens $b \cdot c - q \cdot m = 1$ alakból $(b, m) = 1$ következik. Megfordítva, ha $(b, m) = 1$, akkor a (4.5) elõállítást felhasználva $s \cdot b + t \cdot m = 1$ kapható, ahol $c = s$ választással $c \cdot b = -t \cdot m + 1$, azaz $b\,c = 1 \pmod{m}$ következik.

Az egyértelmûséget a következõképp bizonyíthatjuk. Ha $c' \neq c$, $0 < c, c' < m$ egészekre $b \cdot c = b \cdot c' = 1 \pmod{m}$ adódna, akkor innen $b \cdot (c - c') = q \cdot m$ alakot kaphatnánk, ami az $m \mid b \cdot (c - c')$ oszthatóságot jelentené. De $(b, m) = 1$ miatt csak $m \mid c - c'$ lehetne, ez viszont $0 < |c - c'| < m$ miatt lehetetlen.
■

Az RSA algoritmus alapvetõen használja ki azt a tényt, hogy ha d mod $\Phi(m)$ inverze e-nek, akkor tetszõleges x egész számra

$(x^e)^d = x \pmod{m}$.

Most ezt fogjuk igazolni.

4.4 Tétel (Fermat-tétel): *Ha a c egész nem osztható a p primmel, akkor*

$c^{p-1} = 1 \pmod{p}$

Bizonyítás: Tetszõleges c egészre a c , $2c$, $3c$, ... $(p-1)c$ számok különböznek modulo p . Ha ugyanis $ic = jc \pmod{p}$, $0 < j < i < p$ fennálna, akkor abból, hogy $(i-j)c = q\,p$ és $(i-j) < p$ valamint, hogy p nem osztója c -nek ellentmondásra jutnánk. Így tehát a c , $2c$, ... $(p-1)c$ számok egy és csak egy számmal kongruensek modulo p az 1, 2, ... p -1 számok közül. A kongruencia definiciójából következik, hogy ha $a_i = b_i \pmod{p}$, $i = 1, 2, \ldots n$, akkor

$a_1 a_2 \ldots a_n = b_1 b_2 \ldots b_n \pmod{p}$

is fennáll. Tehát

$1 \cdot 2 \cdot 3 \cdot \ldots \cdot (p-1) = c \cdot 2c \cdot 3c \cdot \ldots \cdot (p-1)c,$

azaz

$$\prod_{i=1}^{p-1} i = c^{p-1} \prod_{i=1}^{p-1} i \pmod{p}$$

fennáll, ahonnan

$$\left(c^{p-1} - 1\right) \prod_{i=1}^{p-1} i = q \cdot p$$

adódik valamely q - ra, ahonnan $p \mid c^{p-1}-1$ következik, s ez már a $c^{p-1} = 1 \pmod{p}$ állítás.
■

Innen már könnyû eljutni ahhoz a bennünket érdeklõ esethez, amikor a modulus két prímszám szorzata.

4.5 Tétel (Fermat-tétel általánosítása): *Ha p_1 és p_2 különbözõ primek, és az c egészre teljesül, hogy $(c, p_1 p_2) = 1$, akkor*

$$c^{(p_1-1)(p_2-1)} = 1 \pmod{p_1 p_2} \qquad (4.7)$$

Bizonyítás: A $(c , p_1 p_2) = 1$ feltétel azt jelenti, hogy sem p_1 sem pedig p_2 nem osztója c-nek, így a Fermat-tétel felhasználásával:

$(c^{(p_1 - 1)})^{(p_2 - 1)} = 1 \pmod{p_1}$

$(c^{(p_2 - 1)})^{(p_1 - 1)} = 1 \pmod{p_2}$

következnek. Ha valamely d egészre, $d = 1 \pmod{p_1}$, $d = 1 \pmod{p_2}$, akkor az ekvivalens $p_1 \mid d - 1$ és $p_2 \mid d - 1$ állításokból $p_1 p_2 \mid d - 1$ következik. A $d = c^{(p_1 - 1)(p_2 - 1)}$ megfeleltetéssel a tétel bizonyítását kaptuk.
■

Ugyanígy lehet az állítást igazolni kettõnél több prímszám esetére is. Ilyen általánosítás Eulertõl származik. Õ vezette be a

$$\Phi(p_1 \cdot p_2 \cdots p_n) = (p_1 - 1)\ldots(p_n - 1)$$

függvényt, innen a függvény, sõt az általánosított tételt is szokás Euler-tételnek nevezni. A bevezetett segédeszközök most már lehetõvé teszik, hogy egyszerûen belássuk az

$$x = D_B(y) = y^d \pmod{m}$$

RSA dekódolási lépés helyességét.

4.6 Tétel: *Ha d az e modulo $\Phi(m)$ vett inverze, akkor bármely x számra fennáll*

$$(x^e)^d = x \pmod{m}. \qquad (4.8)$$

Bizonyítás: Mivel d az e modulo $\Phi(m)$ vett inverze, ezért $ed = q\Phi(m) + 1$, azaz az

$$x = x^{q\,\Phi(m) + 1} \pmod{m} \qquad (4.9)$$

állítást kell belátni. Elõször tetszõleges p prímre belátjuk, hogy

$$x = x^{s(p - 1) + 1} \pmod{p} \qquad (4.10)$$

fennáll. Ha p nem osztója x-nek, akkor a Fermat-tétel szerint

$$(x^{p - 1})^s = 1^s = 1 \pmod{p}$$

tehát (4.9) fennáll. Ha p osztója x-nek, akkor nyilván

$$x = 0 = x^{s(p - 1) + 1} \pmod{p}.$$

Visszatérve az általános esetre, mivel $p_1 - 1 \mid \Phi(m)$ és $p_2 - 1 \mid \Phi(m)$, tehát (4.10) alapján

$x = x^{q\,\Phi(m)+1} \pmod{p_1}$

$x = x^{q\,\Phi(m)+1} \pmod{p_2}$

fennállnak. Innen

$p_1 \mid x^{q\,\Phi(m)+1} - x$

és

$p_2 \mid x^{q\,\Phi(m)+1} - x$

teljesülnek, amelyből $m \mid x^{q\,\Phi(m)+1} - x$ is következik.
■

Az alábbiakban egy újabb számpéldával szemléltetjük a moduláris aritmetika alkalmazását az RSA algoritmus esetén.

Legyen $p_1 = 73$, $p_2 = 151$, így $m = 73 \cdot 151 = 11023$, $\Phi(m) = (73-1)(151-1) = 10800$. Az e paramétert $e = 11$-re választhatjuk, mivel $(10880, 11) = 1$, hiszen $10800 = 2^4 \cdot 3 \cdot 5^2 \cdot 9$. Az e inverzét modulo $\Phi(m)$ az

$e \cdot s + \Phi(m) \cdot t = 1$

elõállításból kaphatjuk:

$10800 = 11 \cdot 981 + 9$
$11 = 9 \cdot 1 + 2$
$9 = 2 \cdot 4 + 1$
$2 = 1 \cdot 2 + 0$

ahonnan

$9 = 10800 - 981 \cdot 11$
$2 = 11 - 9 = 11 - (10800 - 981 \cdot 11) = -10800 + 982 \cdot 11$
$1 = 9 - 2 \cdot 4 = 10800 - 981 \cdot 11 - 4 \cdot (-10800 + 982 \cdot 11) = 5 \cdot 10800 - 4909 \cdot 11,$

így

$-4909 \cdot 11 = 1 \pmod{10800},$

ezért

$d = 10800 - 4909 = 5891 \pmod{10800}$,

tehát $d = 5891$.

Nyilvánosságra hozzuk az $e = 11$, $m = 11023$ egészeket, s titokban tartjuk a $p_1 = 73$, $p_2 = 151$, $d = 5891$ egészeket.

Tegyük fel, hogy az $x = 17$ nyílt üzenetet kívánjuk kódolni, ekkor

$y = 17^{11} \pmod{11023}$,

ahonnan $y = 1782$. A dekódolás az

$$x = 1782^{5891} \pmod{11023} \qquad (4.11)$$

mûvelettel történik. Ezen modulo hatvány kiszámítását egyszerüsíti a "négyzetreemelés és szorzás" módszere. A kitevõt az

$$5891 = 2^0 + 2^1 + 2^8 + 2^9 + 2^{10} + 2^{12}$$

összegre bonthatjuk a bináris ábrázolása alapján. Ennek alapján az (4.11) hatványt az alábbi alakba célszerû átírni:

$$1782^{2^{12}} + 1782^{2^{10}} + 1782^{2^9} + 1782^{2^8} + 1782^{2^1} + 1782 =$$
$$((...(1782)^2)^2 \cdot 1782)^2 \cdot 1782)^2 \cdot 1782)^2)^2)^2)^2)^2)^2 \cdot 1782)^2 \cdot 1782$$

A kiértékelést a legbelsõ modulo négyzetreemeléssel kezdjük, azaz az elsõ néhány lépés:

$1782^2 = 900 \pmod{11023}$
$900^2 = 5321 \pmod{11023}$
$(5321 \cdot 1782) = (2242)^2 = 76 \pmod{11023}$,
...

és az utolsó lépés eredményéül 17-et kapjuk vissza. Így ahelyett, hogy (4.9) mechanikus kiszámításához, azaz a

$$((...(1789 \cdot 1789) \cdot 1789) \cdot 1789) \cdot .....) \cdot 1789$$

szorzáshoz szükséges 5890 darab modulo szorzást végeztük volna el, megúsztuk 17 darab modulo szozással. Könnyen ellenõrizhetõ, hogy ezzel a módszerrel legfeljebb a kitevõ kettes alapú logaritmusa kétszerese számú modulo szorzásra van szükség. Több kísérlet történt a módszer gyorsítására.

4.2.4 Prímszámok keresése

Az RSA rendszerbe történõ belépés kulcsválasztással kezdõdik. Ennek elsõ mozzanata prímszámok keresése. Ez a kérdés már a görög matematikusokat is izgatta. Erasztotenész algoritmusa fokozatosan szûri ki az összetett számokat az összes természetes szám közül. "Szitáján" csak a prímek maradnak meg, ezek azonban hiány nélkül. Módszere azonban számunkra túl lassú, a megkívánt nagyságrendben gyakorlatilag kivihetetlen. Nekünk nincs szükségünk az összes prímszámra, csak néhányra. Az alábbiakban ebbõl a szempontból elemezzük röviden a véletlen primválasztás kérdését. Bizonyítás nélkül utalunk a számelmélet ismert tételére, amely kimondja, hogy $\Pi(n)$, az n pozitív egésznél kisebb primek számának nagyságrendje:

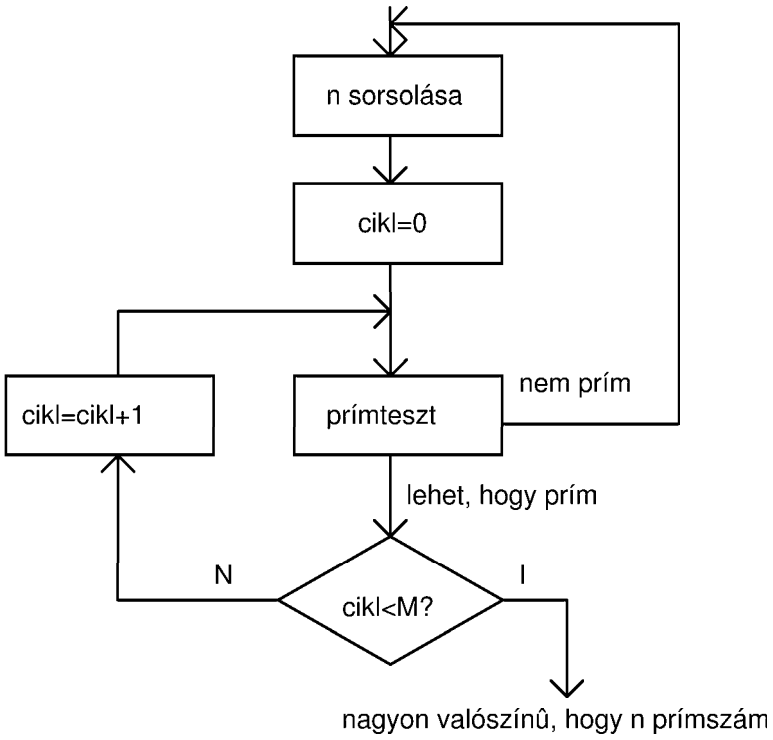
$$\prod(n) \cong \frac{n}{\ln n} \tag{4.12}$$

A mai ismereteink szerint biztonságos az $m = p_1 p_2 = 10^{200}$ nagyságrend, amelyhez $p_I = 10^{100}$ $(= 2^{330})$, $i = 1, 2$ nagyságrendû prímeket választunk. Így annak a valószinûsége, hogy egy véletlenszerûen választott $2^{331} < s < 2^{332}$ (azaz 332 bites) szám prím, az (4.12) alapján approximálható:

$$\frac{\prod(2^{332}) - \prod(2^{331})}{2^{332} - 2^{331}} \cong \frac{2^{331}\dfrac{1}{331 \cdot \ln 2}}{2^{331}} \cong \frac{1}{230}$$

s mivel az adott számtartomány fele páros szám, amelyek nem prímek, elég csak a páratlanok közül válogatnunk, s ezzel 1/115-re duplázzuk meg a találati valószinûséget. Így tehát átlagosan 115 próbálkozásonként jutunk prímszámhoz a 10 nagyságrendû számok között.

Felmerül a kérdés, hogyan dönthetjük el, hogy a véletlenül kisorsolt egész szám prím-e vagy sem. Véletlenül sorsolt számok közül a prímek valószinûségi alapon történõ kiszûrésére több prímteszt algoritmus is létezik. A szûrés folyamatát szemlélteti az 4.1 ábra.



4.1 ábra
Prímszûrés

Fermat-próba

4.7 Definíció: *Egy n összetett szám álprím egy b bázisra nézve, ha*

$$b^{n-1} = 1 \pmod{n} \qquad (4.13)$$

ahol $b \in Z_n$, $Z_n = \{z : 1 < z < n \text{ és } (z, n) = 1\}$.

Az 4.7 Definició a "kis"-Fermat tételen alapul, amely szerint, ha *n* prímszám, akkor (4.13) teljesül. Ha n összetett szám, de mégis teljesíti az (4.13) egyenletet, akkor álprím.

A prímteszt tehát a következõ: ha egy véletlen sorsolt *n* egész egy *b* bázissal nem teljesíti az (4.13) egyenletet, akkor nem prím; ha teljesíti, akkor prímgyanús és továbbvizsgálandó egy következõ bázissal. Ha *M* egymás utáni vizsgálat során prímgyanúsnak bizonyult, akkor nagy valószínûsséggel prímszám. ((4.13) egyenlet teljesülésének ellenõrzését szokás Fermat-próbának is nevezni.)

Kérdés: van-e olyan összetett szám, amely tetszõleges szóbajövõ bázisra "átmegy" a Fermat-próbán? Sajnos van, s ezen számokat Carmichael-számoknak nevezzük. Bizonyságul, a legkisebb ilyen szám az 561.

Felmerül a kérdés, hogy mekkora a valószínûsége annak az eseménynek, hogy egy n összetett szám, amely nem Carmichael-szám, egy véletlenszerûen választott b, $b \in Z_n$ bázissal kielégíti (4.13) egyenletet.

4.7 Tétel: *Ha egy n összetett szám nem Carmichael-szám, akkor legfeljebb n/2 különbözõ b, $b \in Z_n$ bázissal kielégíti (4.13) egyenletet.*

4.1 Lemma: *Ha n álprím b bázisra, akkor álprím b^{-1} (mod n) bázisra is. Ha n álprím b_1 és b_2 bázisokra, akkor álprím ezek modulo n szorzatára azaz $b_1 b_2$ (mod n) bázisra is.*

Bizonyítás: Az 4.7 Definíció alapján az állítás triviális.
■

Bizonyítás (4.7 Tétel): Legyen $B = \{b_1, b_2, \ldots, b_s\}$ azon bázisok halmaza, amelyekre n álprím. B halmaz nem üres, mivel n nem Carmichael-szám. Legyen $b' \in Z_n \setminus B$. Az 4.1 Lemma alapján könnyen látható, hogy $b' \cdot B = \{b' \cdot b_1 \pmod n, b' \cdot b_2 \pmod n, \ldots b' \cdot b_s \pmod n\}$ halmaz elemeire, mint bázisokra n nem álprím. Ha ugyanis $b' \cdot b_i \pmod n$ bázisra álprím lenne, akkor $(b' \cdot b_i) b_i^{-1} \pmod n = b'$ bázisra is annak kellene lennie, ami ellentmondás. Következésképpen azon bázisok száma, amelyekre n nem álprím, legalább $n/2$.
■

4.7 Tétel következménye: *Ha egy n összetett szám Carmichael-szám, továbbá b, $b \in Z_n$ egy véletlenszerûen választott alapszám, akkor annak valószínûsége, hogy n "átmegy" a Fermat-próbán M alkalommal, kisebb, mint 2^{-M}.*

Miller-Rabin-próba

A próba leírásához három jelölést vezetünk be. Adott $n > 1$ páratlan egész számhoz legyenek $u = u(n)$ és $v = v(n)$ azok az egészek, melyekre u páratlan és
$n - 1 = 2^v u$.

Legyen R_n azon $b \in Z_n$ számok halmaza, amelyekre vagy

$$b^u = 1 \pmod n, \qquad (4.14)$$

vagy létezik olyan $0 \le j < v$, melyre

$$b^{2^j u} = -1 \pmod{n} \qquad (4.15)$$

Mivel tetszõleges x egészre

$$(x^{n-1}-1) = (x^{(n-1)/2}+1)(x^{(n-1)/2}-1) =$$
$$(x^{(n-1)/2}+1)[(x^{(n-1)/4}+1)(x^{(n-1)/4}-1)] = \ldots =$$
$$(x^{(n-1)/2}+1)(x^{(n-1)/4}+1)\ldots(x^{(n-1)/2^v}+1)(x^{(n-1)/2^v}-1)$$

azonosság fennáll, ezért, ha n prím, akkor $n \mid (b^{n-1}-1)$ következtében vagy (4.14) vagy (4.15) fennáll, azaz $R_n \in Z_n$. Következésképpen n összetett, ha tetszõleges b esetén $b \notin R_n$.

Ennek alapján a Miller-Rabin próba a következõ. Véletlenszerûen választunk egy b alapszámot. A próba eredménye: n *prímgyanús* , ha $b \in R_n$, illetve n összetett, ha $b \notin R_n$. Bizonyított, hogy annak a valószínûsége, hogy véletlenszerûen választott bázis mellett egy összetett szám megállja a Miller-Rabin-próbát legfeljebb 1/4. Ha az M darab bázist egymástól függetlenül választjuk meg, akkor a téves prímgenerálás valószínûsége 2^{-2M}.

Természetesen merül fel a kérdés: miért kell véletlenszerûen megválogatni az bázisokat. Nem létezik-e univerzálisan jó determinisztikus alapszámhalmaz. A kérdés megválaszolása még várat magára, bár kétségtelenül jelentek meg érdekes eredmények. Igy ha a négy számból álló {2, 5, 7, 13} bázishalmazt használjuk, akkor a Miller-Rabin-próbán egyetlen összetett szám sem megy át $n < 25 \cdot 10^9$ intervallumban, míg a {2, 3, 5, 7} bázishalmaz ebben az intervallumban bázisonként legfeljebb egyet enged meg.

4.2.5 Fejtési próbálkozás példák

1. Tudjuk, hogy m két prímszám szorzata ($m = p_1 p_2$), tehát megkísérelhetjük felbontani. Ha sikerülne olyan x egészet találni, amelyik nem relatív prím n-hez viszonyítva, akkor az Euklideszi algoritmussal megkeresve az (x , m) legnagyobb közös osztót, egyúttal az egyik prímfaktort is megkapnánk. Ilyen x - et jobb híján véletlenszerûen keresünk. Mivel 0 és m - 1 között $\Phi(m)$ szám relatív prím m - hez, ezért annak a valószínûsége, hogy (x , m) > 1

$1 - \Phi(m)/m \approx 1/p_1 + 1/p_2$.

Ha például $p_1, p_2 \approx 10^{100}$, elenyészõen csekély annak a valószínûsége, hogy ez a módszer sikert eredményez.

2. Mint megállapítottuk, faktorizációval nem fejthetõ meg az RSA algoritmus, ha elegendõ nagyra választjuk az m modulust. 1999-ben már nem tekinthetõ biztonságosnak az 512 bit méretû modulus.

3. Simmons és Norris olyan rejtjelfejtõ algoritmust javasolt, amely m faktorizációja nélkül kísérli meg az x üzenet visszaállítását. Módszerük lényege, hogy az adott $y = x^e \pmod m$ rejtjeles számhoz olyan j egészet keresnek, melyre

$$y^{e^j} = y \pmod m$$

Ilyen j szám ismeretében könnyû meghatározni az x üzenetet, hiszen

$$x = y^{e^{j-1}} \pmod m$$

Másrészt j keresése egyszerû iterációt igényel, ismételt hatványozást e kitevõre, míg visszakapjuk y induló értéket. Ha egy adott y esetén nem ismerjük a j értékét, nem tudjuk sikeres támadásra felhasználni a módszert. (Egy olyan x értéket, amelyre $j = 1$ a kódolás fixpontjának nevezzük.)

4. Sikeres támadásra van lehetõségünk, ha a p_1 és p_2 prímek távolsága "nem eléggé nagy". Ugyanis ekkor egy, Fermat-tól származó egyszerû faktorizációs lehetõség adódik az m modulusra. Tekintsünk egy n pozitív összetett egész számot, ahol $n = a \cdot b$, $a \geq b > 0$. Legyen $t = (a + b)/2$ és $s = (a - b)/2$, azaz $a = t+s$, $b = t-s$. Innen $n = t^2 - s^2 = (t+s)(t-s)$ adódik. Ha $a-b$ különbség "kicsi", akkor s is "kicsi ", azaz $t \approx n^{1/2}$. Találgassuk t értékét a következõképpen: számítsuk ki az

$$t_1 = \lfloor n^{1/2} \rfloor + 1,\ t_2 = \lfloor n^{1/2} \rfloor + 2,\ \ldots$$

értékeket, s ellenõrizzük hogy $t_i^2 - n$ különbség négyzetszámot ad-e. Ha igen, akkor a különbség egyenlõ s^2 -tel. A megkapott t és s alapján már könnyen kiszámítható a és b.

Például faktorizáljuk n = 200819-et. n lefelé kerekített négyzetgyöke 448.

$t_1 = 449$, $449^2 - 200819 = 782$ nem négyzetszám,

$t_2 = 450$, $450^2 - 200819 = 1681 = 41^2$ négyzetszám,

ahonnan $200819 = 450^2 - 41^2 = (450+41)(450-41) = 491 \cdot 409$.

4.3. KRIPTOGRÁFIAI PROTOKOLLOK

Algoritmikus szempontból egy titkosító rendszer két fõ komponenst tartalmaz: egyrészt a titkosító kódoló és dekódoló transzformációkat, másrészt kriptográfiai protokollokat. A protokoll algoritmikus lépések sorozata két vagy több résztvevõ partner között valamely feladat végrehajtása céljából.

Szükségesek olyan szabályok, amelyek biztosítják, hogy a titkosító transzformációk egy adott alkalmazásban a megkívánt titkosságot vagy hitelességet nyújtsák. A transzformáció többnyire egy kulcsot használ, de a transzformációt végrehajtó algoritmus nem gondoskodik e kulcs védett célbajuttatásról (kulcskiosztás), a tárolás ideje alatti algoritmikus védelmérõl (pl. hitelességének biztosítása). Aktív támadások ellen egy titkosító transzformáció önmagában nem véd, így megfelelõ szabályokkal kell gondoskodni az üzenetek támadó általi manipulációjának (blokkok nyilvános csatornából történõ kivonása, vagy helyettesítése) felfedhetõségérõl. Protokollok felhasználásával történik a komunikáló partner hitelességének megállapítása, az illetéktelen megszemélyesítés felfedése és megakadályozása is. A legerõsebb titkosító transzformáció sem nyújt védettséget egy hibásan tervezett protokollköryezetben. A kriptográfiai protokollok az algoritmusok igen széles családját foglalják össze. Az gyakorlatban leginkább alkalmazott, alapvetõ protokollok a következõ csoportokba sorolhatók:

- partnerhitelesítés
- kulcskiosztás
- üzenetintegritás
- digitális aláírás
- titokmegosztás

Számos további speciális célú, gyakorlatban kevésbé használatos protokoll ismert, amelyek közül a zero-knowledge protokollok alapelveit bemutatjuk.

4.3.1. Partnerhitelesítés

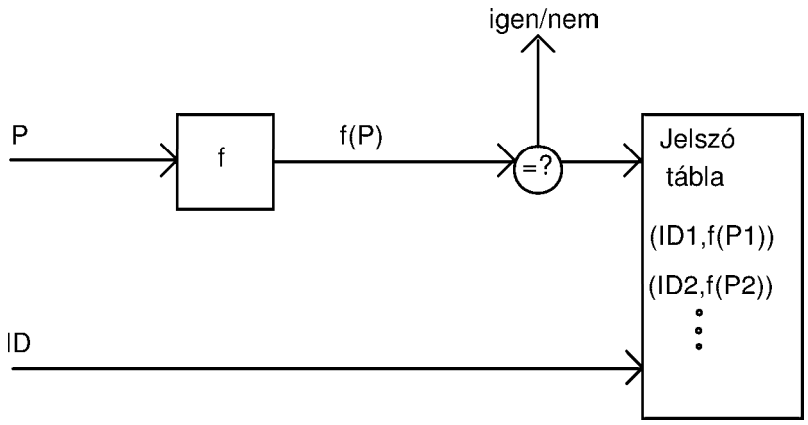
4.3.1.1. Jelszó

A jelszavas partnerhitelesítés a legelterjedtebb, régóta ismert azonosítási eljárás. Hosszú idõ telt el Ali Baba jelszavától a banki PIN kódokig. A jelszó egy titok, amit a felhasználó megoszt azzal az 'erõforrással', amihez alkalmanként hozzá szeretne féri. A jelszavas rendszerek szokásos problémái: a nem megfelelõ jelszóválasztás, a jelszó nyílt alakban történõ továbbítása a rendszerbe jutás pontjától (pl. terminál klaviatúra) az ellenõrzés pontjáig (pl. gazdagép), a jelszavak nem eléggé védett tárolása (mind a felhasználó oldalán, mind pedig a jelszófile tekintetében).

Az alábbiakban több különbözõ protokollon keresztül vizsgáljuk ezen problémák megoldásának vagy enyhítésének módjait.

Egyirányú függvényes leképezés

A jelszóellenõrzés folyamatát láthatjuk a 4.3.1.ábrán.



4.3.1.ábra: Egyirányú jelszóellenõrzés

A jelszavak gazdagép oldali védelmén javít azok egyirányú fügvénnyel történõ leképezése. Ekkor a jelszófile az egyirányú leképezés eredményét tárolja az egyes felhasználókra, a felhasználói azonosítókkal (ID) együtt. A jelszó (P) továbbra is nyílt alakjában érkezik az ellenõrzés helyére, ahol elõbb kiszámításra kerül annak egyirányú függvényes leképezése (f(P)), majd ennek eredménye kerül összevetésre a táblázat bejegyzésével.

Ezzel a megoldással tehát a jelszófile illetéktelen olvasása önmagában nem jelent veszély, hiszen az egyirányú leképezés gyakorlatilag invertálhatatlan tulajdonsága biztosítja, hogy

jelszóhoz a támadó nem férhet hozzá. Ezen utóbbi kijelentés azonban csak feltételek mellett igaz: jelszóméret legyen elég nagy a teljes kipróbálás megakadályozásához, a jelszavakat a teljes jelszótérbõl kerüljenek kiválasztásra.

A nem megfelelõ jelszóválasztás azt jelenti, hogy a felhasználók nem véletlenszerûen választanak az adott hosszúságú alfanumerikus karaktersorozatok közül, hanem pl. értelmes szavakat, jellemzõ dátumokat, ezek triviális kombinációit használják. Ezért a könytár alapú támadások jelentõs százalékú sikerre vezethetnek.

Kétirányú ellenõrzés

Az egyirányú jelszóellenõrzés folyamatában az 'Enter your password:' felszólításra a felhasználó nyíltan a kommunikációs csatornába küld egy titkot, illetve nyíltan átad egy titkot a kommunikációs partnerének, amivel kockázatot is vállal, hiszen a másik fél nem azonosította magát. Egy javított megoldás ellenõrzési folyamata az alábbi, (4.3.1.) protokoll:

1.	A→B:	r1	(4.3.1)
2.	B→A:	y1=f(P2,r1)	
3.	A:	y1=f(P2,r1) ?	
4.	B→A:	r2	
5.	A→B:	y2=f(P1,r2)	
6.	B:	y2=f(P1,r2) ?	

Ezen protokollban mindkét legális fél birtokában van partnere jelszavának, azaz nemcsak egy egyirányú függvényes leképezésének. A jelszavak nem kerülnek nyíltan átvitelre az azonosítási folyamat során. Ehelyett a 'kihívás és válaszvárás' módszerét alkalmazzák. Egy frissen generált véletlen elemet küld át az egyik fél a másiknak, amellyel bõvíti a jelszavát, s ezt a bõvített szót egyirányú függvénnyel leképezi. A leképezés eredményét visszaküldi a 'kihívó' félnek, aki ugyanazon számítást helyben is elvégzi, s a két számítás eredményét egybeveti. A 'kihívás és válaszvárás' mindkét irányban megtörténik. Ezt a módszert ötvözve azzal, hogy a felek egymás jelszavának csak egyirányú leképezéses lenyomatát ismerik egy újabb hasznos protokollhoz jutunk.

A kapcsolatfelvételenként frissített véletlen elemek alkalmazásának célja nyivánvaló, hiszen enélkül nem lenne értelme az egyirányú leképezéses védelemnek, mivel ekkor a leképezés eredménye (y) játszaná tulajdonképpen a nyílt jelszó szerepét. Sajnos a szótár alapú támadás veszélye nem csökkent gyenge jelszóválasztás esetén, hiszen az r véletlen elemet a támadó is láthatja (esetleg a támadó az egyik fél), így az y válaszban csak a P jelszó az ismeretlen.

Fontos észrevétel az, hogy a (4.3.1.) protokoll szerint sajnos kommunikáló páronként kellene egy-egy jelszó-párt egyeztetni, mivel ellenkezõ esetben nyilvánvaló a megszemélyesíthetõség lehetõsége. Így viszont elegendõ lehet felhasználó páronként egy közös "jelszó" elõzetes megbeszélése. Ezzel azonban a jelszó eredeti jelentése elveszett, azaz nem a felhasználókhoz, hanem kommunikációs "irányokhoz" tartozó titokról van szó.

Az egyszerû jelszóprotokoll ugyan nyíltan továbbította a jelszót, ugyanakkor a kiszolgáló oldalán nem igényelte titok tárolását. A fenti kétirányú jelszóellenõrzés esetén ugynakkor mindkét oldalon biztonságosan kell tudni tárolni a titkot.

Egyszer használatos jelszó

Ha a jelszót kapcsolatfelvételenként változtatnánk nyilván semmi értelme nem lenne a jelszó megismerésére irányuló támadásnak, feltéve, hogy a korábbi jelszavakból nem kiszámítható a következõ jelszó. Ekkor tehát az aktuális jelszót nyíltan is átküldhetjük. Egy változó jelszavas protokoll protokoll láthatjuk az alábbiakban:

Ini1.	A:	r generálása	(4.3.2)
Ini2.	A→B:	ID_A, n, $y=f^n(r)$	
1.	A→B:	$P1=f^{n-1}(r)$	
1.1	B:	$y=f(P1)$?	
2.	A→B:	$P2=f^{n-2}(r)$	
2.1	B:	$y=f^2(P2)$?	
...			
i.	A→B:	$Pi=f^{n-i}(r)$	
i.1	B:	$y=f^i(P2)$?	

A protokoll inicializálásakor egy r titkos véletlen elemet A felhasználó az f egyirányú függvény n-szeri alkalmazásával leképez, amelynek y eredményét küldi át B-nek, aki ezt tárolja. Az i-edik bejelentkezéskor használandó Pi jelszó y inverze f^i egyirányú függvényre vonatkozólag. Ezt az inverzet csak A képes kiszámítani, de a számítás helyességét (miután f nyilvános) bárki képes ellenõrizni az f i-szeres alkalmazásával. Az inverz leképezés praktikus futásidejû algoritmusát természetesen A maga sem képes elõállítani, azonban erre nincs is szüksége, mivel r véletlen elem f függvénnyel történõ n-i -szeres leképezése ugyanerre vezet. A két félnek szinkront kell tartani a jelszósorszám vonatkozásában. Ha azonban valamely hiba következtében az átvitel során elveszne egy jelszó, akkor A eggyel továbblép a jelszósorszámban és új jelszót küld, mellette jelezve az egylépéses szinkronhibát.

A módszer biztonságos, hátránya azonban, hogy egy biztonságos külön eszközt is igényel a számítások végrehajtására. Ha kizárólagosan saját használatú számítógéprõl jelentkezünk be ez a probléma megoldható szoftverrel. A másik végletet tekintve, ha egy csak egy 'mindenki' által használt terminál billentyûzetéhez férünk hozzá akkor - és az ellenoldal futtatni képes a dinamikus jelszóprotokollunkat - hozdozható kézi-jelszógenerátor szükséges.

Vegyük észre, hogy a (4.3.2.) protokoll önmagában csak azt biztosítja, hogy az inicializálás két lépését követõen a B félhez érkezõ Pi jelszavakat csakis az tudta elõállítani, aki az inicializálás során a B partnere volt, azaz aki az y üzenetet küldte a számára. Azaz B nem lehet abban biztos, hogy valóban A partnerével áll szemben, hiszen az Ini2. lépésbeli üzenetet egy C támadó is elõállíthatta! Ez persze nem a fenti protokoll hibája, hiszen a klasszikus jelszóellenõrzés is azon alapszik, hogy a legális felek (A és B) elõzetesen - más csatornán, tipikusan személyes találkozás során - megbeszélték a használandó jelszót! A (4.3.2.) protokollnál is el kell végezni egy hitelesítési lépést. Az Ini2. lépésnek hitelesen kell megtörténnie, azaz hitelesen összetartozónak kell lennie y és ID_A mennyiségeknek.

4.3.1.2. Partnerhitelesítés nyilvános kulcsú függvények felhasználával

Tekintsük a következõ, 'kihívás és válaszvárás' típusú partnerazonosítási protokollt, amelyben B kívánja A-t azonosítani:

1.	B→A:	R	(4.3.3)
2.	A→B:	$y=D_A(R)$	
3.	B:	$R=E_A(y)$?	

A protokoll biztonságosnak tûnik, hiszen A nem játszhat vissza korábban felvett 2. lépésbeli üzenetet, továbbá csak A képes y elõállítására a titkos dekódoló kulcsának használatával. A protokoll tehát jónak tûnik, pedig nem feltétlen az. S ez a véletlen elemre adott dekódolási lépéssel kapcsolatos:

Tegyük fel, hogy a leggyakoribb nyilvános kulcsú rejtjelezõ algoritmust az RSA-t kívánjuk használni. Tegyük fel továbbá, hogy C lehallgatott egy korábbi, A számára küldött $y=E_A(x)=x^e$ mod n rejtjelezett blokkot, ahol e az A nyilvános kulcsa az RSA algoritmusnak megfelelõen. Az x üzenetet szeretné megtudni, amit közvetlenül kiszámítani nem tud hiszen, ahol $x=y^d$ mod n , és a d kitevõ az A titkos kulcsa. Tegyük fel, hogy x egy blokk méretû ismert formátumú üzenet, valamint azt, hogy B is óvatos és ellenõrzi a dekódolt blokkok formátumát.

C mindezt tudja, s a következõ ravasz módon jár el. Választ egy R véletlen természetes számot, ahol R<n és (R,n)=1, majd a következõ elõkészítõ számításokat végzi el:

$v=R^e \mod n$
$w=vy \mod n$
$t=R^{-1} \mod n$

Ezután C elküldi w -t A -nak aláírásra. Itt kapcsolódunk vissza a (4.3.3) protokollhoz, ugyanis legyen az (4.3.3) protokollbeli véletlen elem w, amit C küld A -nak 'kihívásként' az 1.lépésben. Az (4.3.3) protokoll 2.lépésében A egy dekódolási lépést hajt végre, azaz visszaküldi a B-t megszemélyesítõ támadónak az

$u=w^d \mod n$

értéket, ami már lehetõvé teszi C számára, hogy megtudja mi is volt az x üzenet, ugyanis:

$tu = R^{-1}w^d = R^{-1}v^d y^d = v^{-d}v^d y^d = y^d = x \mod n,$

ahol felhasználtuk, hogy $R = v^d \mod n$. A következõ módosított változat már biztonságosabb, ahol továbbra is B kívánja A-t azonosítani:

1.	B→A:	R2	(4.3.4)
2.	A→B:	$D_A(R1)$	
3.	A→B:	$z=D_A(R1 \oplus R2)$	
4.	B:	$R1 \oplus R2 = E_A(z)$?	

Ezen módosítás után C már nem képes megszemélyesíteni A-t, mivel nem tudja végrehajtani az 2. lépést, illetve ha az 2.lépésben egy dedódolás nélkül egy véletlen elemet küld át B-nek, akkor nem tudja a 3.lépést elvégezni. Ha A is azonosítani kívánja B-t, akkor ugyanezen protokollt lejátszák fordított szereposztással.

Elkerülhetõ a fenti támadás olyan módon is, hogy rejtjelezésre és azonosításra más kulcskészletet használunk.

4.3.2. Kulcskiosztás protokollok

A konvencionális kódolók közös titkos kulcsot használnak. A jó kulcs valódi véletlen bináris vektor, amelyet nyilván nem lehet szinkronban generálni. Tehát az azt generáló féltõl védetten át kell juttatni a partnerhez. A védelem jelenthet fizikai védelmet (például kulcsszállító hardverben "kézben" visszük át kulcsot a partnerhez), illetve tisztán algoritmikus védelmet, amikor a kulcsot rejtjelezve nyilvános kommunikációs csatornán továbbítjuk. Ezen fejezetben a tisztán algoritmikus módszerekkel foglalkozunk. A konvencionális kódoláson alapuló rejtjelezés ma is alapvetõ módszer, miután a nyilvános kulcsú, rejtjelezés céljára is alkalmas algoritmusok kódolási sebessége sok alkalmazásban nem elegendõen nagy.

Felmerül a kérdés, hogy mindenképpen szükséges-e kulcs-csere, vagy e nélkül is lehetséges titkosan adatot továbbítani nyilvános csatornán keresztül. A válasz az, hogy lehetséges enélkül is, ha csak lehallgató típusú, azaz nem aktív támadás feltételezhetõ. Tegyük fel, hogy A és B kódoló transzformációi kommutatívak, továbbá azonos a nyílt szöveg és rejtett szöveg halmazuk. Shamirtól származik a következõ, háromlépéses eljárás:

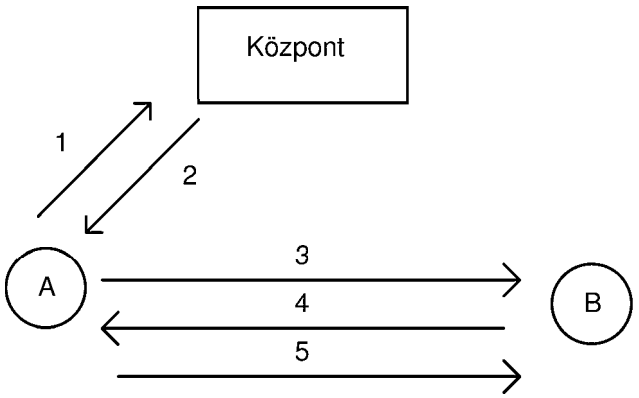
1. A→B: $E_A(x)$
2. B→A: $E_B(E_A(x))$ $(= E_A(E_B(x)))$
   A: $D_A[E_A(E_B(x))]=E_B(x)$
3. A→B: $E_B(x)$
   B: $D_B[E_B(x)]=x$

Nyilvánvaló az aktív támadás lehetõsége, mivel azonosítás (partnerhitelesítés) nincs a protokollban. A C támadó sikeresen megszemélyesítheti B -t mivel egy 2. lépésbeli $E_A(E_C(x))$ üzenetre a 3. lépésben A az $E_C(x)$ üzenetet küldi, amiből a támadó az x nyílt üzenetet visszaállíthatja.

Az alábbiakban tárgyalt kulcskiosztás protokollok mindegyike C aktív támadását próbálja meg kivédeni partnerhitelesítõ protokoll-elemek alklamazásával.

4.3.2.1. Kulcskiosztás konvencionális algoritmussal

A rendszer A,B,C,...felhasználói konvencionális titkosítással kívánnak üzeneteket váltani egymással. Feltesszük, hogy a felhasználók kötõdnek egy-egy terminálhoz és közülük tetszõleges pár tud egymással kommunikálni. A rendszer rendelkezik egy kulcskiosztó központtal, amely lehet egy, a terminálok által elérhetõ számítógép. A további magyarázatot segíti a 4.3.2. szemléltetõ ábra.



4.3.2.ábra: Kulcskiosztás központ felhasználásával

A protokoll a következõ:

1.	A→Kp:	ID_A, ID_B , R1	(4.3.5)
2.	Kp→A:	E_{TKA}(R1, ID_B,DK, E_{TKB}(DK, ID_A))	
3.	A→B:	E_{TKB}(DK, ID_A)	
4.	B→A:	E_{DK}(R2)	
5.	A→B:	E_{DK}(R2 - 1)	

A rendszer háromféle kulcsot használ: a kiosztó központ mesterkulcsát (MK), a felhasználói terminál- kulcsokat (TK_A,TK_B,...) és a kapcsolatkulcsot (DK).

Az R1 és R2 véletlen elemek használatának oka a **visszajátszásos támadás** (replay attack) megakadályozása. Egy ilyen támadásban a C támadó egy korábbi, rögzített üzenetet próbál újrahasználni.

Ha tehát a 2.lépésben A felé az üzenet nem a központtól, hanem C-tõl származna, ezt A rögtön észrevenné, hiszen nem az 1. lépésben általa elküldött véletlen elemet tartalmazná a 2. lépésben megkapott üzenet.

Ha pedig C küldené el A nevében az 1. lépésbeli üzenetet, akkor - nem ismervén a TK_A kulcsot nem tudná dekódolni a 2. lépésben vett üzenetbõl a DK kapcsolatkulcsot.

Ha C a B -t próbálná megszemélyesíteni, nyilván nem tudná dekódolni a 3.lépésbeli üzenetet.

Ha C a A -t próbálná megszemélyesíteni, s a 3. lépéssel indítva egy régebbi 3. lépésbeli üzenetet kíván elküldeni B felé, akkor miután a DK kulcsot ebbõl dekódolni nem tudta, nem lesz képes a 4. lépésbeli 'friss kihívás' 5. lépésbeli megválaszolására.

A 3-4-5. lépések után B azt tudja, hogy olyan féllel áll szemben, aki ismer egy "valamikor" A-nak küldött DK kapcsolatkulcsot. A protokoll nem gondoskodik arról,

hogy A meggyõzõdjön B azonosságáról, hiszen R2 véletlen blokk lévén a nem ellenõrizhetõ, hogy 4.lépésbeli rejtett üzenet valóban egy véletlen blokk rejtjelezésével állt elõ. A protokoll hátránya még, hogy megbízható központ létére támaszkodik, ami nemcsak annak többlet infrastrukturális kiadását jelenti, de annak a veszélyét is, hogy a központ üzemképtelensége vagy sok kérés miatti leterheltsége az egész rendszer üzemképtelenségére vezet.

A protokoll leginkább kifogásolt "gyenge" pontja az, hogy régi, a C támadó által idõközben megismert DK kulcs felhasználható támadásra. A támadást C a protokoll 3.lépésétõl kezdi, s láthatóan B szemében A felet sikeresen megszemélyesítheti (vegyük észre, hogy az $E_{TKB}(DK, ID_A)$ korábbi lehallgatott üzenetet kompletten használhatja fel). Ezen támadással szemben megerõsíthetõ az protokoll, ha a központ $E_{TKB}(DK, ID_A)$ helyett egy idõpecséttel ellátott $E_{TKB}(DK, ID_A,T)$ protokoll-elemet állít elõ. Az idõpecsét azonban az üzenetcsere gyakoriságoknak megfelelõen pontos órát tételez fel, amely biztonságos is abban az értelemben, hogy egy C támadó azt nem képes állítani.

A rendszer üzembe helyezése elõtt megfelelõ védelmi rendszabályok betartása mellett kerülnek elhelyezésre a mester és terminálkulcsok. A központ mesterkulcsával kódolva átlagosan védett tárolóközegen (pl. diszk) tárolhatja az egyes terminálok kulcsait ($E_{MK}(TK_A)$, $E_{MK}(TK_B)$,...). A mester- és terminálkulcsokat kulcstitkosítására, a kapcsolatkulcsot a nyílt adatfolyam titkosítására használják. A legnagyobb védettséget a mesterkulcs igényli, viszonylag a legkisebbet a kapcsolatkulcs kapja, azaz a mesterkucs helyezkedik el a kulcshierarchia tetején. A kapcsolatkulcsok kerülnek leggyakrabban felhasználásra. A hierarchiában alsóbb szintû kulcs kompromittálódása felsõbb szintre nem hat, fordítva viszont igen, hiszen a terminálkulcs megszerzésével dekódolhatjuk az oda érkezõ kapcsolatkulcsot.

Az (4.3.5) protokoll véletlen elemeket használ a régebben felvett protokollrészletek visszajátszásával történõ támadási kisérletek megakadályozására. Miután itt idõpontról van szó, alkalmazhatnánk idõpecséteket is védelemül. Az (4.3.5) kulcscsere protokoll idõpecséteket alkalmazó alábbi változata a kulcscsere és azonosítás kettõs feladatot kívánja megoldani. A javított protokoll az alábbi lépésekbõl áll:

1.	A→Kp:	ID_A, ID_B	(4.3.6)
2.	Kp→A:	$E_{TKA}(T,L,DK, ID_B), E_{TKB}(T,L,DK, ID_A))$	
3.	A→B:	$E_{DK}(T', ID_A), E_{TKB}(T,L,DK, ID_A)$	
4.	B→A:	$E_{DK}(T'+1)$	

A fenti lépésekben T a DK kulcs elõállításának idõpontja, L pedig a DK kulcs élettartama. A 2. lépésben megkapott kódolt üzenetbõl A megállapíthatja, hogy órája szerinti idõpont belül van-e a [T,T+L] intervallumon. Ugyanezt megteszi B is a 3. lépés után. A 3. lépésbeli $E_{DK}(T', ID_A)$ protokoll-elem küldése egyrészt azért szükséges, hogy B meggyõzõdhessen arról, hogy ezen a lépésbeli átvitelt valóban A végzi (vegyük észre, hogy az 1-2. lépésig egy C támadó is eljuthat), másrészt A T' friss elemmel azonosítót

kihívást is küld egyúttal B felé. Az utolsó lépés pedig A felet győzi meg arról, hogy partnere valóban B.

Az ilyen, órát feltételező protokollok ki vannak téve annak a veszélynek, hogy az órák pontos együttfutása (szinkronizáltsága) valamilyen rendszerbeli hiba vagy szándékos beavatkozás miatt megszûnik. Ha a küldő órája siet a vevő órájához képest egy C támadó lehallgatva az üzenetet azt egy visszajátszó támadásban felhasználhatja.

4.3.2.2. Kulcskiosztás nyilvános kulcsú algoritmussal

Ezen pontban további kulcscsere protokollokat vizsgálunk. A előző pontban már tekintett támadási módok, a *passzív hallgatózás* valamint a *visszajátszás*os támadás mellé további két támadás elleni védelmi képességet is vizsgálni fogunk. Ezek a *"támadó középen"* (MIM, Man-In-the-Middle attack) valamint a *megszemélyesítés*es támadás. A *"támadó középen"* támadás a kulcscsere vonatkozásában azt jelenti, hogy a támadó A és B legális felek közé áll, azok kezdődő kulcscsere protokolljába megpróbál bekapcsolódni oly módon, hogy észrevétlen módon rávegye a legális feleket egy általa is ismert kulcs használatára. Ezen támadásnál tehát nem a támadó kezdeményezi a kulcscserét, hanem szinkronban belép A és B beszélgetésébe.

Kulcskiosztás nyilvános kulcsú rejtjelező függvény felhasználásával

A: Hitelesítő központ alkalmazása nélkül:

Feltesszük, hogy nyilvános kulcsú rejtjelező függvények állnak rendelkezésre a protokoll felépítéséhez, továbbá azt, hogy kommunikáló feleknek nincsen módja arra, hogy előzetesen egyeztessék a nyilvános kulcsaikat, nem áll rendelkezésre egy a leendő kommunikációs partnerek nyilvános kulcsait tartalmazó "könyv" , illetve nem támaszkodhatnak egy hiteles központra, mint hiteles harmadik partnerre. Kérdés, mik a támadási siker esélyei ezesetben, a fentebb említett különböző támadási módok lehetőségeit feltételezve.

Megszemélyesítő támadás I.:

Tekintsük elöször a következő egyszerû, egylépéses protokollt:

1. $A \rightarrow B$: $ID_A , E_B(R)$ (4.3.7.)

A egy véletlen R elemet kisorsol, azt B nyilvános kulcsával kódolja. A kódolt véletlen elemet, valamint azonosítóját küldi a B partnerének. B az R véletlen elemet dekódolja az üzenetből, s ezt a véletlen elemet használja A és B közös kulcsként egy szimmetrikus

kulcsú rejtjelezésnél (pl. DES, IDEA) a tényleges adatfolyam rejtjelezésére. A protokoll nagyon egyszerû, könnyen láthatók a támadó lehetõségei:

A kommunikációs csatornában passzívan hallgatózó C támadó nyilván nem képes az R véletlen elem megállapítására.

Ha azonban olyan aktív támadásra is képes, hogy kezdeményezni tud kommunikációt, akkor megszemélyesítheti A-t, azzal hogy õ küldi B-nek az $[ID_A,E_B(R)]$ üzenetet. Kérdés persze, hogy képes-e ezután elhihetõ üzeneteket is cserélni B-vel. Továbbá C azt is megteheti, hogy mind A mind pedig B felé végrehajtja az egylépéses protokollt, s A felé B-nek, B-felé A-nak mutatja magát. Ha eztán A-t és B-t valódi üzenetcserébe tudja ezáltal belevinni, akkor ismerve a kulcsot, azt olvashatja.

Ha nem képes arra, hogy kommunikációt kezdeményezzen (például a legális felek nem fogadnak el tetszõleges idõpontban kezdeményezett kommunikációt, vagy személyek és elõzetesen más csatornán is várnak hiteles jelzést), akkor még megpróbálkozhatna az aktív támadó azzal, hogy a legális felek induló protokolljába kapcsolódjon be, s megpróbálja észrevétlenül eljátszani egy kriptográfiai "protokollkonverter" szerepét. Látható azonban, hogy a támadó az $[ID_A,E_B(R)]$ üzenetet módosítás nélkül kénytelen továbbadni B felé, hogy A-t és B-t az R közös kulcscsal folytatandó üzenetcserébe bevigye. Azonban ezen R kulccsal kódolt üzenetcserét õ nem képes dekódolni, hiszen $E_B(R)$ ismeretében az R kulcsot nem képes megállapítani.

MIM támadás:

A (4.3.7.) egyszerû protokollban az A fél választja a közös kulcsot. B biztonságosabbnak tarthat egy olyan protokollt, amelyben mindkét fél által választott véletlen elemből állítják elõ a közös kulcsot. (Például tart attól, hogy az A fél esetleg nem elég "erõs" véletlen elemeket generál.) Tekintsük az alábbi protokollt:

1. A→B:	ID_A , $k^p{}_A$	(4.3.8.)
2. B→A:	ID_B , $k^p{}_B$	
3. A→B:	$E_B(R1)$	
4. B→A:	$E_A(R2)$	
5. A: , B:	$k=F(R1,R2)$	

A és B az elsõ lépésben elküldik egymásnak azonosítójukat és nyilvános kulcsukat. Eztán rejtjelezve elküldik az általuk választott véletlen elemeket, majd a befejezõ lépésben egy nyilvános F leképezéssel elõállítják a közös kulcsot.

Gondoljuk végig ismét egy C támadó lehetõségeit:

- A kommunikációs csatornában passzívan hallgatózó C támadó nyilván nem képes a véletlen elemek megállapítására.
- C nem képes A (vagy B) megszemélyesítésre, ha helyére áll, s felveszi annak a nyilvános paramétereit.
- Ha A helyett a C támadó küldené a (4.3.8.) protokollbeli üzeneteket az 1. és 3. lépésekben, akkor nyilván nem lenne képes a 4. lépésbeli $E_A(R2)$ kódolt véletlen elemből R2 -t dekódolni, abból a célból, hogy egy közös k kulcsot képezhessen B-vel az 5. lépésben.
- C azonban sikeres, "támadó középen" támadást hajthat végre a (4.3.8.) protokoll ellen a következőképpen. Ha A és B közé áll, és az 1. és 2. lépések helyett a

1. A→C:	ID_A , k^p_A	(4.3.9.)
C→B:	ID_A , k^p_C	
2. B→C:	ID_B , k^p_B	
C→A:	ID_B , k^p_C	

lépések zajlanak, akkor mind A mind pedig B neki fogja elküldeni az R1 és R2 véletlen elemet C nyilvános kulcsával kódolva, amit ő dekódol, majd újrakódol A illetve B nyivános kulcsával, s küldi A illetve B felé. Ezzel ismét olyan k kulcs kerül A-nál és B-nél kiszámításra ami C -nél is rendelkezésre áll.

A (4.3.8.) protokoll "támadó a középen" támadással szembeni védettségét segíti a darabolási technika, ami az **interlock protokoll** néven ismert. A módosított (4.3.8.) protokoll legyen az alábbi:

1. A→B:	ID_A ,k^p_A	(4.3.10.)
2. B→A:	ID_B ,k^p_B	
3. A→B:	$/E_B(R1)/$	
4. B→A:	$/E_A(R2)/$	
5. A→B:	$//E_B(R1)//$	
6. B→A:	$//E_A(R2)//$	
7. A: , B:	k=F(R1,R2)	

Elsõ lépésként A és B elküldik egymásnak a nyilvános kulcsukat. Mindketten elõállítanak egy megfelelõ méretû (kulcsméret) véletlen elemet. A partner nyilvános kulcsával kódolják, de nem küldik el egy lépésben a kódolás eredményét, hanem két félre vágják azt, s elõször az elsõ felét küldik el egymásnak, majd a második felét. A két véletlen elemből képezik egy F leképezéssel a kulcsot (pl. a véletlen bináris vektorként tekintett elemek XOR összegeként).

A (4.3.10.) protokollban egy "középen" támadónak a 3. és 4. lépésben a sikeres támadáshoz az /E_C(R1)/ és /E_C(R2)/ töredékekből elő kellene tudni állítania az /E_A(R1)/ és /E_B(R1)/ töredékeket, amire nyilván nem képes, hiszen ehhez a teljes R1 illetve R2 véletlen elemet kellene elõbb elõállítania az említett /E_C(R1)/ és /E_C(R2)/ töredékekből.

Megszemélyesítõ támadás II.:

Mielõtt megörülnénk az ily módon megerõsített (4.3.10.) protokollnak, sajnos észre kell vennünk mindig van C-nek lehetõsége megszemélyesítéses támadásra. Mi bizonyítja ugyanis B számára a fenti protokollban, hogy A-val beszél. Ha nincs hiteles nyilvános kulcstár, amihez A és B hozzáférhetne (s ezen szakaszban a "hiteles központ" hiánya ezt is jelenti), nem tudja például A észrevenni, hogy egy $[ID_A,k^p_C]$ üzenet hamis. C tehát mégis képes közös szimmetrikus kulcsot megbeszélni külön A-val, külön B-vel, majd a rejtett adatkommunkációs fázisban folyamatosan átkódol egyik kulcsról a másikra. Azaz az interlock annyit még ekkor is nehezít a támadó számára, hogy nem elégséges csak a kulcsmegbeszélés néhány lépésére átkulcsolásokat végezni.

Vegyük azonban észre, hogy a (4.3.10.) kulcscsere protokollunk nem ad lehetõséget egyúttal a partner azonosíthatóságára is. Segíthetünk ezen a problémán azzal, hogy az 1. és 2. lépésben cserélt nyilvános kulcsok cseréjét hitelesíthetõvé tesszük, amihez hiteles központ jelenlétét tételezzük fel:

B: Hitelesítõ központ alkalmazásával (certificate):

Legyen a módosított lépéspár:

1'. A→B: ID_A , k^p_A , C_A

2'. B→A: ID_B , k^p_B , C_B

ahol C_A és C_B a k^p_A és k^p_B nyilvános **kulcsok tanusítványai** (**certificate**), azaz az adott nyilvános kulcsokra egy központ által elõállított digitális aláírások. Ezen aláírást a központ $\{ID_A , k^p_A\}$ együttesére adja. A központ nyilvános kulcsát a rendszer minden résztvevõje ismeri. Mivel ezen aláírást csak a központ képes elõállítani, ezért kizáródik annak a lehetõsége, hogy C támadó A -t megszemélyesítse, mivel ID_A , k^p_C üzenetrészhez kellene egy központ által kiadott aláírást illeszteni, amit nyilván nem megvalósítható C számára. Ezzel nyilvánvalóan egyúttal kizárjuk a 'támadó a középen' támadás lehetõségét is. A módosított teljes protokollunk tehát az alábbi:

1. A→B:	ID_A , k^p_A , C_A	(4.3.11)
2. B→A:	ID_B , k^p_B , C_B	
3. A→B:	$E_B(R1)$	
4. B→A:	$E_A(R2)$	
5. A: , B:	$k=F(R1,R2)$	

Ehhez a megoldáshoz annak árán jutottunk, hogy feltételezzük egy megbízható központ jelenlétét a rendszerünkben.

Kulcskiosztás kommutatív egyirányú függvény felhasználásával

Egyik legismertebb kommutatív egyirányú függvény a diszkrét hatványozás. Ha csak passzív támadásra kell felkészülni, akkor bárminemû kulcscsere illetve központ segítsége nélkül képes két fél - A és B - arra, hogy közös véletlen elemet, azaz közös kapcsolatkulcsot 'megbeszéljen'.

Válasszunk egy 'nagyméretû' véges testet, jelölje ezt GF(q), jelölje g ennek egy - nem titkos - primitív elemét. A protokoll a következõ:

1. A→B:	g^{R1}	(4.3.12)
2. B→A:	g^{R2}	
3. A:	$(g^{R2})^{R1}$	
B:	$(g^{R1})^{R2}$	

Mivel a hatványozás kommutatív mûvelet, ezért a $k=(g^{R2})^{R1}=(g^{R1})^{R2}$ közös kulcsban állapodhat meg a két fél. Az átviteli csatorában hallgatózó C támadó nem képes az R1 illetve R2 véletlen elemeket megállapítani mivel a diszkrét logaritmusképzés nehéz feladat. A C támadó képes "támadó a középen" támadásra. Az A féllel a fenti protokoll szerint megbeszél egy k_A kulcsot, B féllel egy k_B kulcsot, majd összekapcsolja õket olyan módon, hogy konvertálja a rejtett üzeneteket az egyik kulcsról a másikra az üzenet irányának megfelelõen.

Sajnos itt nem segít az interlock ötlet, vagyis az hogy egy kiszámolt hatványnak, mint bináris vektornak csak felét küldjük el egy lépésben. Ugyanis a hatvány nem hordoz semmiféle információt a címzettel kapcsolatban, ellentétben az (4.3.10.) protokoll 3-6 lépésekbeli üzeneteivel. (Azaz nem kell a támadónak dekódolni, hiszen nincs kulcs.) Tehát akár töredékekben, akár egészben kommunikálva a C támadó szeparáltan beszélhet meg egy-egy közös kulcsot A-val illetve B-vel.

4.3.3. Üzenethitelesítés

Az üzenethitelesítés feladata az, hogy a vételi oldalon detektálhatóvá tegyük azon eseményeket, amelyek során az átviteli úton az üzenet valamilyen módosulást szenvedett el. Az alábbiakban a legfontosabb módszerek közül

- kriptográfiai ellenõrzõ összeg (MAC)
- rejtjelezés
- digitális aláírás
- hash függvény

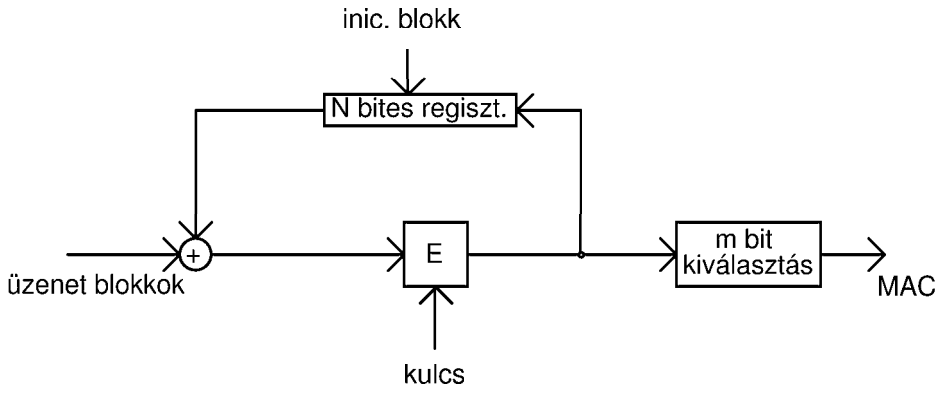
alapú protokollok alapelveit mutatjuk be.

4.3.3.1.Kriptográfiai ellenõrzõ összeg (MAC) alapú üzenethitelesítés

Konvencionális blokk kódolót alkalmazunk. Tegyük fel, hogy n blokkra bontható: $x_1,...,x_n$. Ha az utolsó blokk töredék lenne, egészítsük ki zéró bitekkel teljes blokkra. Blokk kódolónkat CBC (rejtett blokk láncolás) módban használva a kriptográfiai ellenõrzõ összeget (MAC, Message Authentication Code) állítunk elõ, s ezt a nyílt alakú üzenethez fûzve továbbítjuk a csatornán, azaz a továbbított hitelesített üzenet:

$$[x_1, ..., x_n, MAC(x_1,...,x_n)] \qquad (4.3.13)$$

Az eljárást szemléltetjük az 4.3.3. ábrán:



4.3.3. MAC generálás

Formálisan $MAC = g(y_n)$, ahol y_n az

$y_i=E_k(x_i \oplus y_{i-1})$,

i=1,...,n rekurzió n-edik lépésbeli eredménye, továbbá y_0 = I (inicializáló. blokk). A g függvény y_n N bitjéből valamely m bitet választja ki, s az eredmény a kriptográfiai ellenõrzõ összeg. A vételi oldal megismétli az MAC számítást, s az eredményt összehasonlítja a vett MAC-vel. Egyezés esetén fogadja el hitelesnek a vett üzenetet.

Az m értékét akkorára kell választani, hogy elhanyagolható legyen annak a valószínûsége, hogy a C támadó a $[x_1,...,x_n,MAC(x_1,...,x_n)]$ hitelesített üzenetet $[x'_1, ...,x'_n,MAC(x'_1,...,x'_n)]$ hiteles formátumú üzenetre cserélje, ahol $[x'_1, ..., x'_n]$ a támadó céljainak megfelelõ csaló üzenet. Mivel a titkos k kulcsot nem ismeri, azaz MAC számításra nem képes, ezért támadása akkor lehet csak sikeres, kisorsolva egy r m bites vektort, véletlenül $r = MAC(x'_1,...,x'_n)$ egyenlõség fennáll. Az inicializáló blokk lehet nyilvános az adott CBC módú alkalmazásban. Ennek megfelelõen választhatjuk például a csupa zéró blokkot.

4.3.3.2. Konvencionális rejtjelezéssel történõ üzenethitelesítés

Ha konvencionális rejtjelezést alkalmazunk egy nyílt üzenetre és rejtjelezett formában küldjük el a vételi oldalra, nyilván csak a a titkos kulcs birtokában levõ, jogosult vevõ képes annak tartalmát olvasni. Ezzel tehát a hitelesítési feladatot is megoldottuk, ha a nyílt üzenet redundáns, valamely strukturával vagy formátummal, amelyet a vevõ kriptográfián kívüli 'eszközzel' azonosít. Azaz, ha valamely nyelven írott szöveg a nyílt üzenet, akkor ez az 'eszköz' az adott nyelven olvasni tudó ember olvasási képessége. Ha azonban a nyílt üzenet strukturálatlan folyam (legalábbis a rendelkezésre álló 'eszközeink' számára), akkor önmagában a rejtjelezés nem ad lehetõséget üzenethitelesítésre. *Strukturálttá tétel* egyik legegyszerûbb módja valamely lineáris ellenõrzõ összeg (MDC, Manipulation Detection Code) alkalmazása, s a nyílt szöveg ezen összeggel történõ kiegészítése. Ez lehet például az adatátvitelben jólismert ciklikus redundancia karakter (CRC) képzése.

Összegezve: a rejtjelezéssel hitelesítés ötelete, hogy a dekódolt üzenet struktúrája megtörjön egy aktív támadás esetén. (Pl. írott szöveg vagy részletei véletlenszerûvé válnak, beszéd részletei zajjá válnak stb.).

Sajnos ezzel még nem oldottuk meg megnyugtatóan a feladatot. A gondot az jelenti, hogy a nyílt üzenet tipikusan sokszorta hosszabb, mint a konvencionális blokk kódoló blokkmérete, azaz tördelni kell a nyílt üzenetet, s blokkonként kódolni. Ezt azonban nem mindegy hogyan teszzük, ha üzenetmódosító támadásra is gondolnunk kell. Nyilván veszélyes lehet az, ha az x_1, ..., x_n nyílt üzenet blokkokat külön-külön rejtjük (ECB, Electronic Code Book), mert az eredményül kapott $y_1,...,y_n$ rejtett blokkok sorozatának elemei észrevétlenül kihagyhatók, korábbira kicserélhetõk, duplikálhatók lehetnek a nyílt üzenet strukturájától függõen. Ezért alkalmazzuk a láncolás módszerét, így a *rejtett blokk láncolás* (CBC, Cipher Block Chaining) módot. Az 4.3.3. ábrának megfelelõ elrendezést alkalmazzuk, elhagyva a kimeneti m bit kiválasztó elemet. Az I inicializáló blokkot

azonban ezesetben körültekintõen kell kezelni, hogy egy támadási lehetõséget megakadályozzunk. A láncolás elsõ lépése ugyanis

$y_1=E_k(x_1 \oplus I)$,

következésképpen C támadó az I blokkot tudja I' blokkra úgy módosítani, hogy tetszés szerinti x'_1 blokkra módosuljon x_1 elsõ nyílt blokk amellett, hogy $x'_1 \oplus I' = x_1 \oplus I$ fennálljon, azaz hogy az y_1 elsõ rejtett szöveg blokk ne változzon. S mindezt a kulcs ismerete nélkül teheti meg. Hogy ezt a támadását véghezvigye, el kell érnie, hogy a legális vételi oldalon I' inicializáló blokkot alkalmazzanak a dekódolás során. Ha tehát a vételi oldali dekódoló eszköz nem tulajdonítván nagy jelentõséget az I blokk módosítás elleni védelmének akkor potenciális támadásra ad lehetõséget. Tehát elvileg nem baj, ha az inicializáló blokkok tára annyira nyílt, hogy "bárki" által olvasható, de fontos ne legyen illegálisan írható, azaz hiteles maradjon.

4.3.3.3.Digitális aláírással történõ üzenethitelesítés

A konvencionális rejtjelezõk alkalmazásával történõ üzenethitelesítés (MAC illetve CBC módú rejtjelezés) csak a két fél számára ad lehetõséget a hitelesség megállapítására. Így nyilván, hiába mutat fel egy bíróság elõtt például B fél egy konvencionális rejtjelezésû üzenetet azt állítva, hogy azt A küldte, akivel közös titkos kulcsuk volt, a bíróság számára ez nem bizonyíték, hiszen B maga is elõállíthatta azt. Van azonban olyan módszer, amivel detektálhatóvá tehetünk üzenetmódosító támadást, s ugyanakkor egy harmadik fél felé is bizonyítékul szolgálhat. A módszer a digitális aláírást használja. (részletesen elemezzük a digitális aláírás módszereket az 4.3.4. fejezetben)

4.3.3.4. Üzenethitelesítés titkos kulcs nélkül

Egyirányú lenyomatkészítõ függvény, azaz hash függvény felhasználásával is oldhatunk meg üzenethitelesítési feladatot. Ekkor a titkos kulcsból illetve a nyílt üzenetbõl képzett kriptográfiai ellenõrzõösszeg (MAC) helyett csak a nyílt üzenet hash függvényes lenyomatát használjuk ellenõrzõ összegként, azaz ha X jelöli a nyílt üzenetet,

X, Hash(X)

kerül átküldésre A-tól B-hez, ahol Hash egy nyilvánosságra hozott függvény. Érthetetlennek tûnik az állítás, hiszen bárki elõállíthat X',Hash(X') párt tetszõleges, általa választott X' üzenethez. Mégis használható lehet a módszer a gyakorlatban, ha ezen algoritmikus eszközökön túl, A és B között telefonösszeköttetés is rendelkezésre áll. Ekkor ugyanis B felhívja A-t, akinek hangját ismeri, azt kéri, hogy az elküldött hash érték elegendõ számú - hexadecimális - karakterét olvassa be a telefonba.

4.3.4. Digitális aláírás

Míg az üzenet- és partnerhitelesítés protokollok a kommunikáció idõtartamára és a partnerek számára nyújtanak hitelesítési lehetõséget, addig a digitális aláírás az üzenetváltás után és harmadik személy számára is nyújt hitelességellenõrzési lehetõséget. A digitális aláírás protokollok a következõ feladatot oldják meg:

1. az aláírás generálása (az üzenetet küldõ végzi)
2. az aláírás ellenõrzése (az üzenetvevõ által)
3. hitelességgel kapcsolatos vitás kérdések harmadik személy elõtti (pl. bíróság) tisztázása.

A 3. pontban említett vita tárgya lehet: Az aláíró szeretne letagadni egy korábban általa küldött üzenetet, mert tartalma már kedvezõtlen számára. Vád tárgya lehet az is, hogy a címzett hogy saját céljainak megfelelõen módosította a küldött üzenetet. A digitális aláírás utánozni kívánja a valódi kézjegy tulajdonságait, nevezetesen:

i. legyen könnyen generálható,
ii. ne legyen egyik "okmányról" a másikra áthelyezhetõ (hamisítható), azaz csak a tulajdonosa generálhassa,
iii. "bárki" képes legyen ellenõrizni annak hitelességét.

A digitális aláírásnak a fenti közös tulajdonságok mellett van egy igen fontos sajátossága, mégpedig az, hogy nem az üzenet anyagi hordozójához (pl. papír) tartozik, hanem szerves része az aktuális üzenetetnek, azaz *üzenetfüggõ*.

Nyilvános kulcsú titkosító algoritmus felhasználásal egyszerûen készíthetünk digitális aláírást. Tegyük fel, hogy A az x üzenetet kíván B -nek elküldeni olyan módon, hogy egyúttal aláírását is elhelyezze a rejtett üzenetben. Ezt elérheti, ha a titkos kulcsát alkalmazva egy dekódolási lépést hajt végre (tegyük fel egyenlõre, hogy az x üzenet hossza nem nagyobb, mint a dekódoló transzformáció input mérete). Nyilván

$$D_A(x)$$

függvénye mind az x üzenetnek, mind pedig a titkos dekódoló kulcsnak, tehát csak A képes elõállítani azt. Az üzenetvevõ B fél ismerve A nyilvános kulcsát képes x visszaállítására egy

$$x = E_A(D_A(x))$$

kódolási lépéssel. Abban az esetben, ha x egy B által is ismert formátummal (általánosabban redundaniával) rendelkezõ üzenet, akkor $D_A(x)$ önmagában az aláírt üzenet, s nem csak az aláírás. (Egy szokásos redundancia-bevitel, adott számú zérus bittel teljes méretû blokkra történõ kiegészítés.) Ugyanis csak A képes olyan z, dekódoló output

méretû blokkot elõállítani, amelyre $E_A(z)$ nem egy véletlenül választott blokk lesz, hanem olyan, amely megfelelõ formátummal is rendelkezik. Ha azonban nem tételezhetünk fel, hogy B formátum ellenõrzést végez (pl. a B oldali szoftver erre nem készült fel), akkor egyszerûbb, ha a $D_A(x)$ -t csak aláírásnak tekintjük, amit az x üzenethez csatolva küldünk el azaz ekkor

$[x , D_A(x)]$

az aláírással hitelesített üzenetet. B fél ekkor $E_A(D_A(x))$ kódolási lépés eredményét veti össze a nyíltan is megérkezett x üzenettel.

Ezzel teljesítjük a következõ - aláírással szembeni - elvárásokat:

1. az aláírás hitelessége biztonsággal ellenõrizhetõ: B az A nyilvános kulcsát használó kódolási lépéssel bizonyossággal megállapíthatja, hogy a küldõ A volt-e
2. az aláírás nem hamisítható: csak A ismeri a szükséges titkos kulcsot
3. az aláírás nem átvihetõ egy másik dokumentumra: az aláírás függvénye az adott dokumentumnak
4. az aláírt dokumentum már nem változtatható meg: ha megváltoztatják a dokumentumot, ahhoz nem illeszkedik már az aláírás
5. az aláírás letagadhatatlan: B-nek nincs szüksége A-ra, hogy egy harmadik fél számára bebizonyítsa, miszerint A küldte az aláírt dokumentumot.

Mindezen elvi tökéletességek ellenére két ponton tovább érdemes finomítani az aláírás protokollt. Ezek a pontok a
- lenyomatkészítés,
- idõpecsét alkalmazása.

4.3.4.1. Lenyomat aláírása

Célszerû az dokumentum méretétõl függetleníteni az aláírás méretét, s egy alkalmas nyilvános egyirányú dimenziószûkítõ függvény (hash függvény) felhasználásával nem az eredeti dokumentumra, hanem annak lenyomatára adni az aláírást. Ekkor az aláírt üzenet

$[x, D_A(Hash(x))]$

alakú. Egy támadó x megfigyelésével a hash függvény nyilvánossága miatt ki tudja számolni Hash(x) értékét. Sikeres támadáshoz azonban arra van szüksége, hogy egy olyan x' üzenetet találjon, amivel csaló célját elérheti, s ugyanakkor Hash(x)=Hash(x'), mert ezesetben $[x', D_A(Hash(x))]$ is hiteles üzenet B szemében. Csakhogy egy elfogadható hash függvény garantálja, hogy gyakorlatilag nem találhatunk azonos hash értékre vezetõ x' üzenetet. A támadó feladatát még csak tovább nehezíti, hogy egy ilyen azonos hash

értékre vezetõ üzenetnek ráadásul értelmes, sõt céljainak megfelelõ csaló tartalmúnak kell lennie.

Egy születésnapi paradoxon alapú támadás és kivédése példa

Például 64 bites hash érték esetén 2^{-64} annak a valószínûsége, hogy egy megfigyelttel azonos hash értékû üzenetet találjunk. Ha azonban C támadó rá tudja venni A-t, hogy adjon aláírást egy általa összeállított - akár ártatlannak tûnõ szövegû dokumentumra - elvileg van módja ezen valószínûség lényeges megnövelésére a születésnapi paradoxon felhasználásával. Tegyük fel, hogy C elõ tud állítani 2^{32} ártatlan és ugyanennyi csalásra felhasználható dokumentumot (értelmes üzenetet). A - klasszikus - születésnapi paradoxon miatt nagy a valószínûsége annak, hogy a két halmazban van legalább egy pár azonos hash értékkel.
(Megjegyzés: A valószínûsége, hogy a pár mindkét eleme az ártatlan vagy a csaló halmazban legyen 1/4 illetve 1/4, azaz 1/2 a valószínûségû, hogy a pár egyik tagja ártatlan míg a másik csaló legyen. A születésnapi paradoxon esetén az egyesített halmazméret lenne 2^{32} , azaz 2^{64} négyzetgyöke. A mostani példánkban $2 \cdot 2^{32} = 2^{33}$ méretû halmazban keressük a párt, az 1/2-es szorzó kompenzálásául.)

A támadás menete innen már nyilvánvaló: kérjen C a pár ártatlan tagjára aláírást, s ezzel már lesz aláírása a csaló párjára is. Mondhatnánk, hogy egyáltalán hogyan tudnánk 2^{33} (≈8 milliárd) dokumentumot (szöveget) elõállítani illetve tárolni. Szemléltetésül tekintsük az alábbi minipéldát:

"Tisztelt .... Úr!
Ezen levelemmel bemutatom Önnek ..... urat, aki .... naptól .... napig megbízottam .... ügyletekben. Jogosult ..... összeghatárig ..... illetve ..... vásárlására. Az összegeket .... pénznemben bocsássa rendelkezésére .... címletekben. A kettõnk közti elszámolás .... ütemezéssel történik stb. "

A kipontozott helyekre többféle lehetõség szerint választva szavakat nagyszámú levélváltozatot állíthatunk elõ. Például, ha 33 kipontozott helyet hagyunk, s mindegyikre két lehetõség szerint választunk, máris 2^{33} levélváltozatunk van. Egy-egy változatot 33 bittel tudunk kódolni (vessük ezt össze például egy egyoldalas, 2 kbyte méretû szöveg 16384 bitjével, ami felére tömörítve is 8000 bit nagyságrendbe esik.) Mindegyik levélhez 64 bites hash érték tartozik, így egy levélhez 33+64+1=98 bit -et rendelünk, ahol a plusz egy bit jelzi, hogy melyik halmazból való a levél. A párosítást végezzük úgy, hogy a teljes, 2^{33} méretû halmaz elemeit rendezzük sorba hash értékek szerint, s ha egybeesést találunk a rendezés során, akkor a halmazindikátor bit alapján eldöntjük, hogy különbözõ halmazból valók-e. A rendezéshez $33 \cdot 2^{33}$ nagyságrendû számítás szükséges, ami ha nagy érték is, nem lehetetlenül nagy a mai technológiai szinten. (Megjegyezzük, hogy a tárigény 10 Gbyte nagyságrendû.)

A fenti támadást maga a dokumentum szerzõje is végezheti, elõre bebiztosítva maga számára a csalás lehetõségét.
Ha a támadást nem õ végzi, akkor a támadó munkáját nehezíthetjük azzal, hogy minden egyes aláírás elõtt az aláírandó dokumentumhoz illesszünk egy véletlenül sorsolt bináris sorozatot. Ezt a véletlen kiegészítést a C támadó predikálni nem képes, így nem képes a sablon dokumentumváz ismeretében az elõkészítõ számításokat végezni.

4.3.4.2. Idõpecsét az aláírásban és a letagadásvédelem

Tegyük fel, hogy A aláírásával hitelesítetten elküldött B -nek egy szerzõdést, majd egy idõ múlva - a körülmények számára kedvezõtlenné válása miatt - szeretné, ha letagadhatná az aláírt szerzõdés elküldését (repudiation). Elhíreszteli, hogy már elõzõleg kompromittálódott a titkos kulcsa, ezért nem vállal felelõsséget a nevében aláírt szerzõdésért. Helyezzünk el idõpont információt az aláírásban, s tekintsük a következõ protokollt:

$$1.\ A \rightarrow B:\ [x,T,L\ ,\ D_A(\ [Hash(x),T,L]\ )\ ] \qquad (4.3.14)$$

ahol x a dokumentum , T a dátum és idõpont információ, L az aláírás élettartama. Sajnos (4.3.14) protokoll még nem nyújt védelmet az A általi letagadás ellen. (4.3.14) azonban például arra már alkalmas, hogy egy A által kiállított, s digitálisan aláírt csekkel B csak az aláírás élettartamán belül tudhasson pénzt felvenni, ahol L választása lehetõleg zárja ki a többszöri pénzfelvételt.

Ahhoz, hogy A ne tudja letagadni, hogy õ írta alá a dokumentumot, nyilván nem elég az idõpont, hanem egy *harmadik megbízható személy* - részvétele is szükséges a protokollban. Legyen ez a személy G. Egyszerûbb jelölés kedvéért az alábbiakban $S_A(x)$ jelölje az A által digitálisan aláírt x dokumentumot, továbbá $V_A(X)$ jelölje az A általi aláírással elátott $X=S_A(x)$ dokumentum ellenõrzését az aláírás hitelessége szempontjából. Tekintsük a következõ protokollt:

1. A→G:	$U=S_A(u)$ $(u=[\ I\ ,\ v]\ ,\ v=S_A(x))$	(4.3.15)
2. G:	$V_A(U)$	
3. G→A:	$W=S_G(w)$ $(w=[\ T\ ,\ I\ ,\ S_A(x)\ ])$	
G→B:	$W=S_G(w)$ $(w=[\ T\ ,\ I\ ,\ S_A(x)\ ])$	
4. A:	$V_G(W)$	
5. B:	$V_G(W)$, (I) , $V_A(v)$	

Az 1. lépésben A aláírja az x dokumentumot, az aláírt dokumentumot (v) kiegészíti azonosító fejléccel (I), s újra aláírja az eredményt, amellyel U -t kapja. Az azonosító információ minimálisan azt tartalmazza, hogy A B-számára szándékozik aláírt dokumentumot küldeni. A 2. lépésben G ellenõrzi U által hordozott külsõ aláírást, valamint az azonosító I információt. *G az aláírt x dokumentumot*, az I azonosító információt *kiegészíti egy idõpecséttel* (T), majd az eredményt aláírja, s a 3. lépésben elküldi azt mind A -nak mind pedig B-nek. A 4. lépésben A ellenõrzi G-tõl érkezett üzenetet, s ha nem õ küldte az I azonosítójú, $S_A(x)$ aláírt dokumentumot, akkor azonnal jelzi, hogy kompromittálódott titkos kulcsával visszaéltek. Az 5.lépésben B ellenõrzi G aláírást, az I azonosító információt, majd pedig A aláírását.

I azonosító használata némi magyarázatra szorul. Ugyanis úgy gondolhatnánk, hogy a dokumentum (x) elvárhatóan tartalmazza a két fél azonosítóját. Azonban nem feltétlen köthetjük meg, hogy a dokumentumában ki hova tegyen azonosítót, s egy számítógépes, automatikus alírásellenõrzõ rögzített pozíciójú adatmezõkkel nyilván egyszerûbben dolgozik. Továbbá az 5.lépésben B az I információból tudja meg, hogy A küld számára aláírt dokumentumot, s A nyilvános kulcsával kell a $V_A(v)$ verifikációs lépést végrehajtania.

4.3.4.3. Digitálisan aláírt üzenet rejtjelezése

Ha A csak rövid, formátumozott üzeneteket szeretne titkosan B-nek küldeni, amelyek formátumát B ellenõrzi, akkor ezt legegyszerûbben megteheti úgy, hogy egy x nyílt, formátumozott üzenetet

$y=E_B(D_A(x))$

kódolt alakban küldi el. Ez az y kódolt üzenet teljesíti elvárásainkat: csak B képes az x tartalmat megismerni, továbbá azt is ellenõrizni tudja, hogy A volt az üzenet feladója. Például RSA kódoló alkalmazása esetén tipikusan 64-128 byte hosszt értjük egy blokknyi, azaz rövid üzenetnek. Ha az üzenet nem rövid, illetve nem támaszkodunk annak formátumozottságára a vételi oldali verifikáció során, akkor az $U=[x, D_A(Hash(x))]$ aláírással kiegészített hosszabb üzenetet rejtjelezzük B nyilvános kulcsának felhasználásával. Ekkor U -t a kódoló inputjának megfelelõ méretû blokkokra tördeljük, s blokkonként kódoljuk. Nem szükséges blokkláncolás alkalmazása, mivel egy blokk-módosító támadást (blokk-csere, -törlés,-duplikálás) a digitális aláírás ellenõrzése során detektálhatunk.

4.3.4.4. Digitális aláírás konvencionális kódolás felhasználásával

Konvencionális kódolást használva is lehetséges digitális aláírást készíteni egy x üzenethez, ekkor azonban egy megbízható harmadik személyt (G) is be kell vonni a protokollba. A illetve B egy-egy titkos kulcsot oszt meg G-vel. A protokoll alapváltozatához tegyük fel, hogy egy blokknyi, formátumozott üzenetről van szó:

1. A→G:	$v=E_A(x)$	(4.3.16)
2. G:	$D_A(v)$	
3. G→B:	$w=E_B([A , x])$	
4. B:	$D_B(w)$	

G biztos lehet abban, hogy a v üzenetet A küldi, hiszen vele osztotta meg a megfelelõ titkos kulcsot. B is biztos lehet abban, hogy A küldte az x üzenetet, hiszen G meggyõzõdött arról, hogy x az A üzenete, s ezt a B számára úgy jelzi biztonságosan, hogy G és B közös titkos kulcsával rejtjelezve küldi az A azonosítóját és az üzenetet.

Hasonlóan a nyilvános kulcsú rejtjelezõ transzformáció használatával történõ digitális aláíráshoz, itt is tovább finomíthatjuk az eljárást a lenyomatkészítés és idõpecsét alkalmazásával.

Az (4.3.16) konvencionális kódolásra alapuló protokoll hátránya, hogy egy megbízható harmadik személy jelenléte már az alapváltozatban is szükséges. Láttuk azonban, hogy a nyilvános kulcsú eljárás letagadásvédelemmel bõvített protokollja is igényelt egy harmadik felet, illetve általában, ha a nyilvános kulcsok helyben nem állnak rendelkezésre, azaz le kell kérni azokat, akkor hitelesített nyilvános kulcs küldés is feltételez harmadik felet, aki a tanusítványt kiállítja.

4.3.5.Titok megosztása

Mint láttuk, a titkosító transzformációk mindegyike esetén szükséges valamilyen titkos információ, a titkos kulcs, amelyet már nem véd újabb titkosító transzformáció. Ehhez ugyanis újra valamilyen titkos információ kellene s.í.t.. Igy ezt a titkos információt másfajta védelemre kell bízni. Leheteséges pl. valamilyen fizikai védelem alá kell helyezni (memorizálás, felnyitás-biztos dobozba helyezés stb.). Egy másik módszer úgy igyekszik "feldarabolni" a titkos információt N személy között, hogy abból tetszõlegesen választott K személy együttesen rekonstruálni tudja a titkot, de K-nál kevesebb személy sohasem legyen erre képes, ahol K<N. Ez a megoldás nyilván rekonstruálhatóvá tenné a titkot még akkor is, ha annak legfeljebb N-K darabja megsemmisülne.

Kézenfekvõ lenne a binárisan ábrázolt titok valahány, mondjuk T szeletének szétosztása. Ez azonban elfogadhatatlan. Ha ugyanis a szeletek számával egyezõ a személyek száma (N=T), akkor egy rész megsemmisülése is vezethet a titok elvesztéséhez, másrészt az összes személy szükséges a rekonstrukcióhoz. Ha viszont több személynek is adjuk ugyanazt a szeletet (N>T), akkor nem választhatunk tetszõleges K=T személyt a rekonstrukcióhoz. Továbbá, jelentõs információval rendelkezik T-hez közeli számú személy együttese, hiszen közvetlenül a titkos bináris információ részeit kapták meg.

Címezzük meg a lehetséges titkok S halmazának elemeit a 0,1,...,q-1 számokkal.
Egy kicsit javíthatunk a helyzeten a következõképpen. Válasszunk N-1 alkalommal véletlenszerûen egy-egy elemet a {0,1,2,...,q-1} halmazból. Az így kiválasztott $r_1,r_2,...,$ r_{N-1} címek tehát teljesen független, egyenletes eloszlású valószínûségi változók. Jelölje $s \in S$ az aktuális titkot, s az N személynek szétosztandó N számú információ legyen r_1, $r_2,..., r_{N-1}, r_N$, ahol

$$r_N = s - (r_1+r_2+...+ r_{N-1}) \qquad \text{mod } q$$

Ekkor ugyan továbbra is az összes személy szükséges a titok rekonstruálásához, viszont N-nél kevesebb részlet ismerete nem nyújt információt a titokra vonatkozóan. Vegyük azt is észre, hogy ekkor a titok-darabok mindegyikének azonos a mérete a titokéval.
Ha a véletlenítési ötletet ötvözzük a Reed-Solomon kódolással, akkor egy kívánt megoldáshoz jutunk. Nevezetesen tekintsünk egy (N,K) paraméterû RS-kódot GF(q) felett, ahol q legyen prímhatvány (ha q eredetileg nem ilyen lenne bõvítsük ki az fiktív elemekkel S halmazt). Az RS-kódok ismert tulajdonsága szerint a kódszó legalább K elemének ismeretében egyértelmûen dekódolható, amit a hibajavító kódolás elméletében úgy fogalmazunk, hogy ≤ N-K törlés javítására alkalmasak. Ezt a tulajdonságot a titok-szétosztásban a következõképpen kamatoztathatjuk:

Válasszunk K-1 elemet véletlenszerûen GF(q)-ból, amelyeket jelölje r_1, $r_2,...,$ r_{K-1}, továbbá legyen r_0=s. Az

$r=(r_0, r_1,..., r_{K-1})$

vektort tekintsük az RS-kóddal kódolandó üzenetnek, azaz ezt egy G, GF(q) feletti KxN dimenziós generátormátrix-szal leképezzük egy

$c=(c_0, c_1,..., c_{N-1})$

kódszóba a

$c=rG$

lineáris transzformációval. A c kódszó elemeit osztjuk szét az N személy között. A G generátormátrixot választhatjuk a következõképpen:

$$G = \begin{pmatrix} 1 & 1 & 1 & . & 1 \\ 1 & \alpha & \alpha^2 & . & \alpha^{N-1} \\ . & . & . & . & . \\ 1 & \alpha^{K-1} & \alpha^{2(K-1)} & . & \alpha^{(N-1)(K-1)} \end{pmatrix}$$

ahol $\alpha \in GF(q)$ N-edik primitív egységgyök. Bevezetve a

$D(x) = r_0 + r_1x + ... + r_{K-1}x^{K-1}$

GF(q) feletti polinomot, a c kódszó elemei a

$c_i=D(\alpha^i)$,

i=0,1,...,N-1 alakban is elõállíthatók. Ennek a konstrukciónak van egy érdekes interpretációja:

Egy olyan K-1 -edfokú y=D(x) polinom létezik (lehet valós számtest vagy véges test feletti), amely K adott (y_1,x_1), (y_2,x_2),...,(y_K,x_K) ponton keresztül fektethetõ, vagyis amelyre

$y_i=D(x_i)$,

$1 \le i \le K$. Ha tehát N darab egy görbén fekvõ pontot választunk, akkor bármely legalább K pontot tartalmazó részhalmazból a görbe (polinom) rekonstruálható, amelynek a nulladfokú tagja a titok.

4.3.6. Zero knowledge protokollok

4.3.6.1. Fiat-Shamir protokollok partnerazonosításra

Parnerazonosítás két fél, A és B között lezajló protokoll, ahol egyik fél vagy mindegyik azonosságát kívánja igazolni a másik fél számára. Ha például A jelszavát 'mutatja fel' B-nek, akkor B olyan információhoz jut, amellyel egy harmadik fél felé képes A-t megszemélyesíteni. Lehetséges-e olyan azonosítási protokollt konstruálni, amelyben B nem jut olyan információhoz, amelyet késõbb A megszemélyesítésére használhat? Másképp megfogalmazva, képes-e A azonosítani magát B elõtt anélkül, hogy ilyen ismerethez (knowledge) hozzájuttatná B-t. Az alábbiakban bemutatásra kerülõ Fiat-Shamir protokoll egy ilyen, zero-knowledge protokoll. A protokoll a modulo összetett szám szerinti négyzetgyökvonás nehézségén alapul, ezért elõször tömören összefoglaljuk a négyzetgyökvonás feladatot.

Négyzetgyökvonás modulo n=pq

Ha létezik oyan b természetes szám, ahol 0<b<n, amely az

$$x^2 = c \text{ modulo } n \text{ , } 0 < c < n \qquad (4.3.17)$$

egyenlet megoldása, akkor c számot b kvadratikus maradékának, b megoldást pedig a c négyzetgyökének nevezzük modulo n. Több kérdés merül fel az (4.3.17) egyenlettel kapcsolatban: mikor van megoldása, azaz mely c számokból lehet gyököt vonni; ha van megoldása hány különbözõ megoldása van; van-e praktikus algoritmus a gyökök kiszámítására. A válasz nagyban függ az n modulustól. Ennek megfelelõen két esetet tekintünk: az egyik, amikor n primszám, a másik, amikor n két különbözõ primszám szorzata.

Ha n=p , ahol p primszám, akkor ha egy c számra van megoldása az (4.3.17) egyenletnek, s ez a megoldás b, akkor a másik megoldás p-b. Miután ekkor modulo p test feletti egyenletrõl van szó legfeljebb két különbözõ megoldás lehetséges. Ha a (4.3.17) egyenlet megoldható, a megoldásra ismeretes praktikus algoritmus, amely polinomiális futási idejû. Azt is egyszerûen beláthatjuk, hogy (4.3.17) egyenlet a szóbajövõ c értékeknek pontosan felére oldható meg.

4.3.1. lemma: Ha n=p primszám, akkor az (4.3.17) egyenlet pontosan s=(p-1)/2 különbözõ c értékre oldható meg.

Bizonyítás: Ha b megoldása (4.3.17) egyenletnek, akkor p-b is az , ahol 0<b<p. Tehát tehát a kvadratikus maradékok száma legfeljebb s. Ha belátjuk, hogy tetszõleges $b_1 \neq b_2$, $0 < b_1 < b_2 \leq s$ pár esetén $b_1^2 \neq b_2^2 \pmod p$, akkor bebizonyítottuk az állítást. Tegyük fel, hogy $b_1^2 = b_2^2 \pmod p$ fennáll valamely b_1, b_2 párra, azaz $p \mid b_2^2 - b_1^2$. Innen vagy $p \mid b_2 -$

b_1 vagy $p \mid b_1 + b_2$ következik, ami nem lehetséges, mivel $0 < b_2 - b_1 < p$ és $0 < b_2 + b_1 < p$. ♦

Ha $n=pq$, ahol p és q primszámok, akkor ha egy c számra van megoldása a (4.3.17) egyenletnek, akkor négy megoldása van, s a megoldások $\{b_1, n-b_1, b_2, n-b_2\}$, $0 < b_1, b_2 < n$. A négy d^2 megoldást az

$x^2 = c$ modulo p
$x^2 = c$ modulo q

egyenletek megoldásait felhasználva a kinai maradékok tétele segítségével kaphatjuk meg. Mivel a modulo primszám egyenletek polinomiális idejû megoldási algoritmusa ismert, ezért ha ismerjük n két faktorját az (4.3.17) egyenlet megoldását is kiszámíthatjuk polinomiális idõben. Ha viszont ezen faktorokat nem ismerjük, egy nehéz feladattal állunk szemben. Ezen állításunkat egyszerûen beláthatjuk. Válasszunk véletlenszerûen egy d, $0<d<n$ számot és számítsuk ki a négyzetét modulo n. Legyen $c=d^2 \pmod n$. Tegyük fel, hogy a (4.3.17) egyenletnek könnyen ki tudjuk számítani egy d' megoldását. Innen $d^2 - d'^2 = 0 \pmod n$ következik , azaz

$$n \mid (d - d')(d + d'). \qquad (4.3.18)$$

Mivel d számot véletlenszerûen választottuk, ezért 1/2 valószínûsége annak az eseménynek, hogy {d=d' vagy d=n-d'}, azaz d-d'=0 vagy d+d'=n. Ekkor az (4.3.18) oszthatóság triviális. Azonban ugyancsak 1/2 a valószínûsége annak, $d-d' \neq 0 \pmod n$, amely esetben mivel $|d-d'|<n$ és $d+d'<2n$, ezért akár az l.n.k.o.(n,|d-d'|) akár l.n.k.o.(n,d+d') legnagyobb közös osztó számítással n faktorizálásához jutunk, amirõl viszont tudjuk, hogy nehéz feladat.

Az alábbiakban bemutatásra kerülõ protokollok arra épülnek, hogy az (4.3.17) egyenlet megoldása könnyû feladat, ha ismerjük n faktorjait, ellenkezõ esetben azonban nehéz.

A Fiat-Shamir partnerazonosítási protokoll

A protokoll egy kulcskiosztó központot használ. A központ véletlenszerûen választ két primszámot, s azok szorzatából elõállít egy n modulust. Amikor A be kíván lépni az azonosító rendszerbe, a központtól kap egy nyilvános és egy titkos kulcsot. A titkos kulcs, u egy véletlenszerûen választott szám, a nyilvános kulcs ennek négyzete modulo n, azaz $v=u^2 \pmod n$.

A protokoll alapeleme a következõ négy lépés:

1. A→B:	$z=R^2 \pmod n$		(4.3.19)
2. B→A:	b		
3. A→B:	R	,ha b=0	
	$w=R\cdot u \pmod n$	,ha b=1	
4. B:	$z=R^2 \pmod n$?	,ha b=0	
	$w^2=z\cdot v \pmod n$?	,ha b=1	

ahol R, 0<R<n egy véletlen szám, amelyet A az 1.lépést megelõzõen generál, míg b egy véletlen bit, amelyet B a 2.lépést megelõzõen állít elõ. Hogyan próbálhatja egy C támadó megszemélyesíteni A felet:

1.) C végrehajtja az 1. lépést, megfigyeli a 2. lépésbeli b bitet. Ismerve A nyilvános v kulcsát, a feladata b értékétõl függõen vagy R küldése, amit könnyedén megtehet, hiszen õ generálta azt az 1. lépés elõtt, vagy kénytelen megpróbálkozni azzal, hogy gyököt vonjon z·v szorzatból modulo n, ami gyakorlatilag kivitelezhetetlen számára. Azaz 1/2 valószínûséggel tud sikeresen támadni.

2.) C megpróbálja predikálni a 2. lépésben átküldésre kerülõ b bitet. Ha C sejtése b=0:

1. C→B: $z=R^2 \pmod n$
3. C→B: $w=R \pmod n$

C sejtése b=1:

1. C→B: $z=R^2\cdot v^{-1} \pmod n$
3. C→B: $z=R \pmod n$

Ha b bitet valódi véletlen módon sorsoltuk, akkor 1/2 -nél nagyobb valószínûséggel helyesen predikálni nem képes C a b bitet. Következésképpen 1/2 valószínûséggel tudja C támadó az A felet sikeresen megszemélyesíteni.

Ha tehát t számúszor megismételjük az (4.3.19) protokoll-elemet, akkor már csak 2^{-t} annak valószínûsége, hogy C sikeresen tudja A- t megszemélyesíteni.

A módszer egy finomítása az, hogy k számú titkos kulcs - nyilvános kulcs párt ad a központ egy újonnan a rendszerbe lépõnek, azaz egy $(v_1,v_2,...,v_k;u_1,u_2,...,u_k)$ 2k elemû vektort, ahol $v_i=u_i^2 \pmod n$. A 2.lépésben nem egy véletlen bitet küld át B A-nak, hanem k bitet, azaz egy $b_1,b_2,...,b_k$ bináris sorozatot. A 3.lépésben

$$w=R\cdot u_1^{b1} u_2^{b2} \cdots u_k^{bk} \pmod n$$

kerül átküldésre. Ennek megfelelõen a 4.lépésben

$$w^2=z\cdot v_1^{b1} v_2^{b2} \cdots v_k^{bk} \pmod n$$

az ellenõrzõ lépés. Ezen módosítással a protokoll t-szeri alkalmazása esetén már csak 2^{-t} annak valószínûsége, hogy C sikeresen tudja A- t megszemélyesíteni.

A Fiat-Shamir protokoll igen alkalmas intelligens kártyás azonosító rendszerben történõ felhasználásra, ahol a kártya és a tulajdonosa az A fél, akik egy azonosító terminállal, mint B féllel állnak szemben. Tegyük fel hogy az azonosító terminálok üzemeltetõje és a kártya kiállítója nem ugyanaz a szervezet, azaz például nem egy pénzintézet. A terminálok bárki tulajdonában lehetnek, így a kártyának fel kell készülnie arra is, hogy a terminál a partnerazonosítás kapcsán megpróbál jogtalanul titkos ismeretekhez jutni, amelyeket a terminál tulajdonosa esetleg felhasználhat a kártya tulajdonosának megszemélyesítésére. Gondoljunk például arra, hogy a kártya pénzhelyettesítõként funkcionál egy POS (Point Of Sale) terminál felé. A cél tehát az, hogy a rendeltetésüknek megfelelõen használt terminálok biztonságosan azonosíthassák a kártyát (tulajdonosát), míg csalási célú terminálok semmilyen használható titokhoz ne jussanak.

Ezen alkalmazásnál az (4.3.19) protokoll nyilvános v kulcsát v=f(I,c), módon számoljuk, ahol f egy hash függvény, I az ügyfél nyilvános azonosítója (például [neve,számlaszáma,kártyaszáma]), továbbá c valamely kis (nyilvános) érték, amelynek egyetlen megválasztási szempontja az, hogy v kvadratikus maradék legyen. A hashing alkalmazásának célja a v méretének használható méretûvé szûkítése, s emelett annak biztosítása, hogy gyakorlatilag ne fordulhasson elõ, hogy két különbözõ felhasználó azonos v nyilvános kulcsot kap. Az u titkos kulcs a kártyában, ki nem olvasható módon kerül elhelyezésre. A kártya lehetõvé teszi az I és c adatelemek ellenõrzését. A kártya tehát azonosítja a terminál felé a - jogos - tulajdonosát anélkül, hogy az bármit is megtudna a titkos elemmel, azaz a titkos kulccsal kapcsolatban. A kártya helyettesíti jogos tulajdonosát, annak közremûködése nélkül mûködik, következésképpen elvesztése egy csaló megtalálót egyszerû helyzetbe hoz.

A protokoll az RSA hatványozásával szemben csak szorzást alkalmaz, ami jóval kisebb számításigényt jelent.

Felmerülhet az olvasóban, hogy miért nem közvetlenül aknázzuk ki a gyökvonás fentebb taglalt nehézségét, mondjuk a következõ protokoll elem útján:

1. B→A: $R^2 \pmod n$
2. A→B: $R \pmod n$

Ekkor azonban az A félnek gyököt kell vonnia, amit ugyan végre tud hajtani, ugyanakkor ennek a számításigénye jóval meghaladja a Fiat-Shamir protokoll számításigényét, amely ügyesen kikerüli a gyökvonás feladatát.

4.4. A diszkrét hatványozás, mint egyirányú függvény

Kriptográfiai alkalmazásra az egyik elsõ egyirányú függvényként Diffie és Hellman a diszkrét hatványozást (prím modulus melletti moduláris hatványozást) javasolta. Jelen fejezetben röviden ismertetjük javaslatukat, majd a diszkrét logaritmus kiszámítására ismertetünk módszereket. Részletesebben két algoritmust mutatunk be a GF(p)-beli logaritmusképzéssel kapcsolatosan: a Knuth-tól származó alapalgoritmust valamint ennek egy továbbfejlesztett változatát a Silver-Helmann-Pohlig algoritmust. A GF(2^n)-beli logaritmusképzés algoritmusai között röviden megemlítjük az indexkalkulus-algoritmus, valamint a Coppersmith-algoritmus alapelvét. Mindegyik algoritmus egy "logaritmus táblát" készít, majd ennek a segítségével módszeres próbálgatást alkalmaz. A tábla mérete és a próbálgatások száma fordítottan viszonylanak egymáshoz, s "értelmes" kompromisszum alapján kell õket megválasztani. A diszkrét hatványozás legismertebb kriptográfiai alkalmazása a Diffie-Hellman kulcscsere protokoll.

4.4.1. A diszkrét hatványozás, mint egyirányú függvény

Választunk egy "nagy" prímszámot és az üzenetet 1 és p -1 közti x számok sorozatává elõkódoljuk. Ezeket az x számokat aztán egyenként, egymás után képezzük le. A leképezéshez egy további paramétereként választunk egy α primitív elemet, tehát egy olyan α számot, melyre az

$$\alpha, \alpha^2, \ldots \alpha^{p-1}$$

modulo p vett számok mind különböznek, vagy ami ezzel kevivalens, a legkisebb pozitív n egész, amelyre

$$\alpha^n = 1 \pmod{p}$$

pontosan p - 1 (ami a Fermat tétel szerint 1 eredményt adó exponens). Ismeretes, hogy minden p prímre létezik primitív elem. A (p , α) pár képezi a rendszer (titkos) kulcsát, s az x rejtjeles képe

$$y = f(x) = \alpha^x \pmod{p} \qquad (4.4.1)$$

diszkrét hatvány. A primitív elem definíciója miatt $f(x)$ kölcsönösen egyértelmû leképezés, azaz létezik az inverze. Az inverz leképezés a diszkrét logaritmus, amely az $y = \alpha^x$ hatvány ismeretében megadja a kitevõt, azaz

$$x = \log(y) \qquad (4.4.2)$$

A diszkrét hatványozás és logaritmusképzés feladata értelemszerûen általánosítható prím modulus helyett q prím hatványra. Mint láttuk a diszkrét hatvány kiszámítható legfeljebb $2\lceil \log q \rceil$ számú mod q szorzással az ismételt négyzetreemelés és szorzás módszerének alkalmazásával, ahol $\lceil x \rceil$ az x valós szám felfelé kerekített egész értéke.
Az inverz feladat, a logaritmusszámítás sokkal nehezebb. A 70-es évekig a leggyorsabb közismert algoritmus - az alábbiakban ismertetésre kerülõ Knuth-algoritmus - aszimptotikusan $O(p^{1/2} \log p)$ modulo p szorzást, valamint ugyanezen aszimptotika szerinti bitméretû memóriát igényelt. Ebbõl a számításigényből lényegesen sikerült lefaragni a diszkrét logaritmusszámítás gyorsítása kapcsán felpezsdült kutatások eredményeképpen. Egy versenyfutás alakult ki a diszkrét logaritmusképzés nehézségére építõ kriptotranszformációk q paraméterének növelése és az egyre nagyobb q értékeket a logaritmusképzés szempontjából elérhetõ közelségbe hozó matematikai eredmények között. Ennek eredményeképpen a 80-as évek második felére $q = p$ primvásztás esetén $p > 2^{500}$ illetve $p > 2^{1000}$ szükségesnek illetve megnyugtatóan megfelelõnek tûnik, a $q = 2^n$ választás esetére $n > 800$ illetve $n > 1000$ a megfelelõ értékek.

4.4.2 A Knuth algoritmus

Legyen p egy "nagy" prím, és α egy primitív elem modulo p. Szeretnénk adott β ($0 < \beta < p$-1) egész szám esetén a

$$\beta = \alpha^r \pmod p$$

kongruencia r megoldását megtalálni. Ehhez elõször határozzuk meg az

$$m = \lfloor p^{1/2} \rfloor \qquad (4.4.3)$$

egész értéket ($\lfloor x \rfloor$ az x valós szám lefelé kerekített egész értéke), amellyel a keresett r kitevõ maradékos osztással

$$r = c \cdot m + d \qquad (4.4.4)$$

alakba írható egyértelmûen ($0 < d < m$, $c \approx p^{1/2}$). Így $\beta = \alpha^r \pmod p$ hatvány r kitevõjének kiszámítása ekvivalens azon (c, d) pár meghatározásával, amelyre az

$$\alpha^d = \beta \cdot \alpha^{-cm} \pmod p \qquad (4.4.5)$$

modulo egyenlõség teljesül. A megoldás elsõ lépéseként elõre kiszámítjuk az $\alpha^d \pmod p$, $d = 0, 1, \ldots m$-1 hatványokat, és az (α^d, d) párokat rendezetten (a hatványok, mint egész számok növekvõ sorrendjében) tároljuk. Elsõ lépésként tehát egy adatbázist generálunk.

Az aktuális β testelemhez, amelynek a logaritmusára vagyunk kiváncsiak, sorra kiszámítjuk a

$$\beta\cdot\alpha^{-m},\ \beta\cdot\alpha^{-2m},\ \ldots \pmod p$$

értékeket (hatványozás és szorzás), és minden egyes $\beta\cdot\alpha^{-im}$ kiszámítása után megnézzük, hogy szerepel-e adatbázisunkban. Ha igen, akkor megtaláltuk a keresett (c, d) párt, s így (4.4.4) alapján a keresett r logaritmus értéket is.

Az össz-mûveletigény kiszámításakor a következõképpen gondolkodhatunk. Az adatbázis egy elemének kiszámítása az elõzõ hatványból egyetlen szorzással történik, hiszen $\alpha^{d+1}=\alpha^{d}\cdot\alpha$, ami az adatbázisra nézve összesen $m\text{-}1 \approx p^{1/2}$ szorzást jelent. Az elemek rendezése $O(p^{1/2}\log p)$ mûveletet igényel, azaz az adatbázis elõállítása $O(p^{1/2}\log p)$ mûveletigényû. A módszeres próbálgatás során egy-egy újabb $\beta\cdot\alpha^{-im}$ kiszámítása szintén egy szorzást igényel, míg az adatbázisban keresés $O(\log p)$ mûveletigényû, s ezt maximum $p^{1/2}$-szer kell elvégezni. Ezek alapján az adatbázis képzése és egy logaritmus számítása aszimptotikusan $O(p^{1/2}\log p)$ szorzást igényel.

Meggondolásunkban egyetlen β-hoz tartozó logaritmust akartunk meghatározni, ennek megfelelõen választottuk az adatbázis méretét $m \approx p^{1/2}$-nek, ami akkor bizonyíthatóan a legjobb a legjobb választás. Ha sok β szám logaritmusára van szükség, akkor kifizetõdõbb lehet nagyobb adatbázis generálása.

Knuth módszere univerzális, minden p esetén alkalmazható. Bizonyos tulajdonságú prímek esetén azonban gyorsabb eljárást is lehet találni. Ilyen a következõ fejezet algoritmusa, amelyik p-1 prímfelbontását használja ki.

4.4.3 A Pohlig-Hellman-Silver algoritmus

Legyen p tetszõleges prímszám, és p-1 prímfelbontása

$$p-1 = p_1^{n_1} p_2^{n_2} \ldots p_k^{n_k} \quad , p_i < p_{i+1}$$

Az algoritmus az $r = \log_\alpha(\beta)$ szám kiszámítását az

$$r_i = r \pmod{p_i^{n_i}},\ i = 1,2,\ldots,k \qquad (4.4.6)$$

számok kiszámítására vezeti vissza. Ezen számok alapján a kínai maradékok tételének felhasználásával kaphatjuk meg az

$$r' = r \ (\mathrm{mod}(\prod_{i=1}^{k} p_i^{n_i})) = r \ (\mathrm{mod}\, p-1) \qquad (4.4.7)$$

modulo értéket. Mivel $0 < r < p$-2, ezért $r' = r$, tehát megkapjuk a keresett r exponenst. (Vegyük észre, hogy a kiindulásként feltételezi ez a módszer, hogy p-1 faktorizálását el tudjuk végezni. Mivel p>2, ezért 2 valamely hatványa leválasztható. Ha nagyok a további primfaktorok, akkor nem reményteljes a felbontás. Ugyanakkor, mint látni fogjuk, ha nagyok a primfaktorok, akkor ez a diszkrét logaritmusképzõ algoritmus amúgy sem jelentene lényeges áttörést a számításigény vonatkozásában Knuth-féle alapmódszerhez képest.)

(**Kínai maradékok tétele**: *Ha az $m_1, m_2, \ldots m_r$ pozitív egészek páronként relatív prímek és $a_1, a_2, \ldots a_r$ tetszõleges egész számok, akkor az $x = a_i \ (\mathrm{mod}\, m_i)$ $i = 1, 2, \ldots r$ kongruenciarendszernek van közös megoldása. Bármely két megoldás kongruens modulo $m_1 m_2 \ldots m_r$.*)

Az algoritmus idõigénye szempontjából az r_i számok elõállítása a meghatározó. Az r_i ($0 < r_i < p_i^{n_i}$) értékek kiszámítása a következõképpen történik. Az r_i egyértelmûen elõállítható p_i alapú számrendszerben, tehát

$$r_i = \sum_{j=0}^{n_i - 1} b_{i,j} \cdot p_i^j \qquad (4.4.8)$$

alakban. A cél a $b_{i,j}$ ($j = 0, 1, \ldots n_i$-1) "számjegyek" meghatározása. Ehhez vezessük be a

$$\gamma_i = \alpha^{(p-1)/p_i}$$

elemet, amely primitív p_i-edik egységgyök, azaz p_i az a legkisebb pozitív egész s hatvány, amelyre $\gamma_i^s = 1$ adódik eredményül. Ezzel

$$\beta^{(p-1)/p_i} = \alpha^{r(p-1)/p_i} = \gamma_i^{r} = \gamma_i^{r_i} =$$
$$\gamma_i^{b_{i,0} p_i^0} \gamma_i^{b_{i,1} p_i^1} \ldots \gamma_i^{b_{i,n_i-1} p_i^{n_i-1}} = \gamma_i^{b_{i,0}}$$

ahol felhasználtuk, hogy az (4.4.6) definícióból $r_i = r \ (\mathrm{mod}\, p_i)$ következik. Ha tehát elõre elkészítjük a

$$\gamma_i^s, \ s = 0,1,\ldots, p_i - 1$$

rendezett adatbázist (logaritmustábla) , akkor a

$\gamma_i^{b_{i,0}}$

hatványhoz megkereshetjük a $b_{i,0}$ kitevõt. A $b_{i,0}$ kiszámítása után a

$\beta' = \beta\alpha^{-b_{i,0}}$

számot állítjuk elõ, amelybõl $b_{i,1}$ -et számíthatjuk ki a következõ hatványozás mûvelettel

$\beta'^{(p-1)/p_i^2} = \gamma_i^{b_{i,1}}$

ahonnan ismét a logaritmustábla alapján kapjuk meg a $b_{i,1}$ "számjegyet". A többi "számjegy" számítása az algoritmus értelemszerû folytatásával történik.

Az algoritmus aszimptotikus számításigénye a Knuth-algoritmusnál látott gondolatmenet alapján

$$O(\sum_{i=1}^{k} n_i\sqrt{p_i}\,\log p_i) \qquad (4.4.9.)$$

nagyságrendû. Azaz, ez az algoritmus elsõsorban figyelmeztetõ jellegû, hiszen ha a p-1 faktorai nem mind kicsik, akkor nem kapunk lényeges nyereséget a Knuth-algoritmushoz képest.

Abban az esetben azonban, ha p-1 primfaktorai relatíve kicsik, a diszkrét logaritmust igen gyorsan kiszámíthatóvá teszi. A legrosszabb választás az, ha $p = 2^t+1$. Ekkor ugyanis p-$1 = 2^t$, azaz $p_1 = 2$, s így a logaritmusképzés $O((\log p)^2)$ mûveletigényû, azaz lényegében a hatványozással azonos. Ekkor a diszkrét hatványozás nyilván nem egyirányú leképezés. A titkosító számára legkedvezõbb választás ebbõl a szempontból, ha $p = 2p'+1$, ahol p' is prim. Általában akkor, ha a p-1 faktorai nem mind kicsik, akkor nem kapunk lényeges nyereséget a Knuth-algoritmushoz képest.

A fentiekben bemutatott két eljárás körültekintõ p választás esetére $O(e^{c_1 n})$, $n = \lfloor \log p \rfloor$ futási idejû. Prím modulus esetére elõször Adleman adott n-ben exponenciálisnál jobb eljárást, nevezetesen $O(\exp(c_2\cdot(n \log n)^{1/2})$ futási idõvel. Ezt Hellman és Reyneri adaptálták hasonló futási idõvel 2 hatvány esetére. Ez az eljárás az Adleman-Hellman-Reyneri- vagy index-kalkulus algoritmus. Igazán lényeges áttörést Coppershmidt algoritmusa hozott, amely felgyorsította az index-kalkulus eljárást, és $O(\exp(c_3\cdot(n \log^2 n)^{1/3})$ aszimptotikus futási idõt ért el.