Kriptográfia

Hatodik előadás
Nyilvános kulcsú kriptográfia I.
Az RSA

Németh L. Zoltán
SZTE, Számítástudomány Alapjai Tanszék
2008 ősz

Titkos kulcsú kriptográfia (Private-Key Cryptography)

- a hagyományos: szimmetrikus/titkos/egy kulcsú kriptográfiában **egy** titkosító/megfejtő kulcs van
- ha nem is szó szerint egyezik meg a kettő, a titkosító és a megfejtő kulcs, egymásból könnyen kiszámítható
- a kulcsot csak a feladó és a címzett ismeri
- a kulcs titokban tartásán alapszik a biztonság
- a feleknek előzetesen kommunikálni kell egymással a titkos kulcsot
- ez szimmetrikus, a felek szerepe egyenrangú: mindketten tudnak titkosítani és megfejteni is
- ezért nem védi a feladót a címzettel szemben attól, hogy a címzett a kapott üzenetet meghamísítva azt állítsa, hogy az a feladótól jött

Nyilvános kulcsú kriptográfia (Public-Key Cryptography)

- talán a legjelentősebb találmány a kriptográfia 3000 éves történetében
- **két kulcs van**
  - egy **nyilvános (public key)**
  - egy **magán (private key)** /néhol: **saját kulcs** v. **titkos kulcs**/
- **a nyilvános kulccsal lehet titkosítani**
- de az üzenetet **csak a magánkulccsal lehet megfejteni**
- így például maga a küldő sem tudja visszafejteni az üzenetet, ha mondjuk elfelejtette, hogy mit titkosított

Nyilvános kulcsú kriptográfia II (Public-Key Cryptography)

- **a nyilvános kulcsot nyilvánosságra lehet hozni** és legalább a küldő számára nyilvánosságra kell hozni (de ez nem igényeli hogy biztonságos kommunikáció legyen)
- bárki lehet feladó, aki a nyilvános kulcsot megkapja
- a számelmélet számítási szempontból ,,egyik irányban nehéz -- másik irányban könnyű'' problémáin alapszik (pl. faktorizáció, diszkrét log.)
- kiegészíti és nem helyettesíti a titkosított kulcsú kriptográfiát

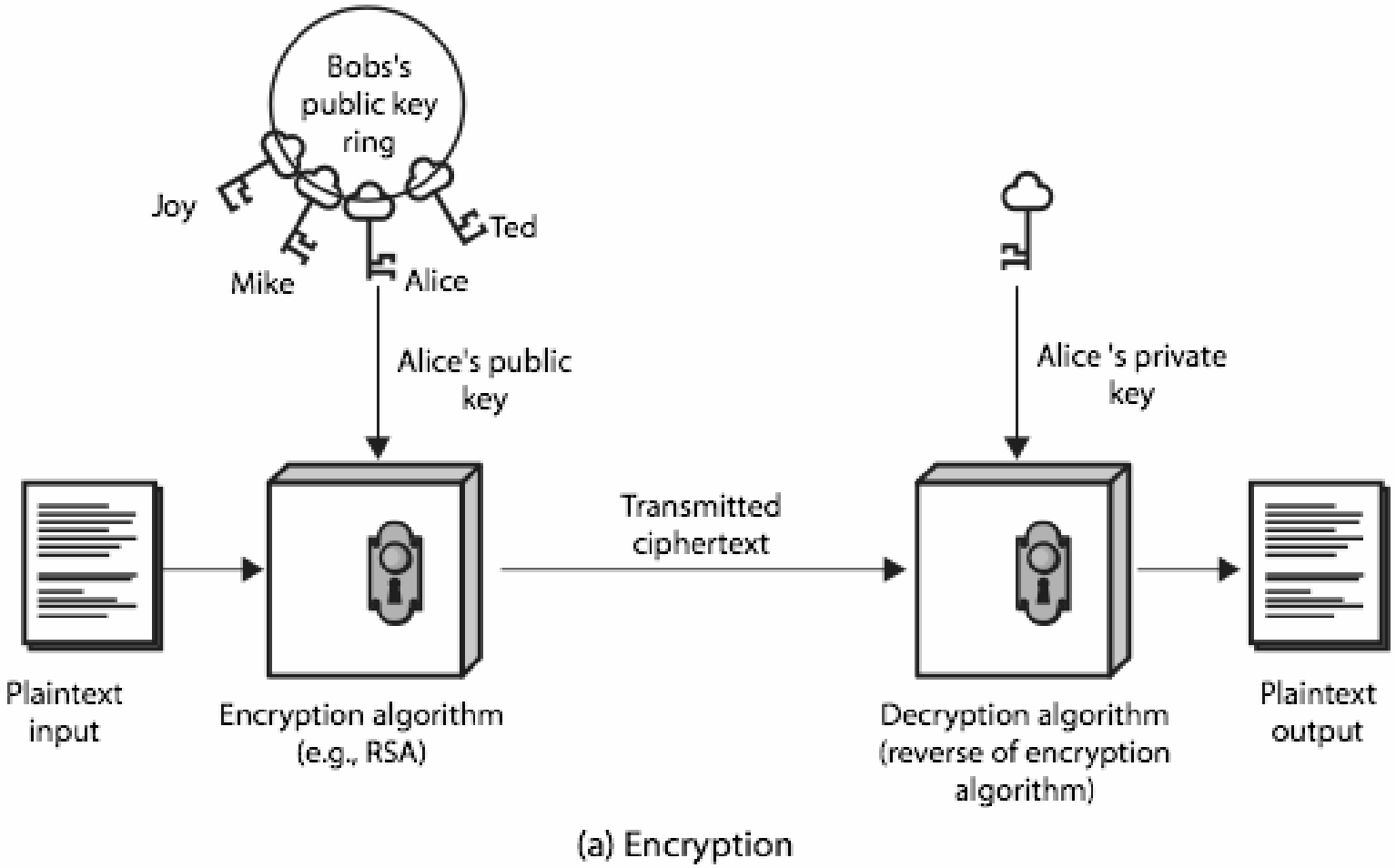
Miért jó a nyilvános kulcsú kriptográfia?

- a titkos kulcsú kriptográfia két alap problémájára ad választ:
  - **kulcselosztás**
  - **elektronikus aláírások**
- az első **nyilvános** publikációja:
  Whitfield Diffie és Martin Hellman (Stanford), 1976
  - ismert volt, bár titokban tartották 1999-ig: James Ellis (UK), 1970
  - sőt állítólag az NSA már a 60-as évek közepén ismerte

Public-Key Cryptography

- nyílt kulcsú/két kulcsú/aszimmetrikus titkosításnak is nevezik
- a kulcsok szerepe:
  - a nyilvános kulcsot **titkosításra** és a magánkulccsal készített **aláírás ellenőrzésére** lehet használni
  - a magánkulccsal (amit csak a címzett ismer) a **megfejteni** lehet, és **aláírást készíteni** természetesen másik irányú titkos vagy nem titkos üzenetküldéshez
- **a felek szerepe aszimmetrikus**:
  a nyilvános kulcs tulajdonosa (a feladó)
  - **csak titkosítani és aláírást ellenőrizni tud**,
  - **megfejteni vagy aláírni nem**
- ezért aláíráshoz (saját névre szóló) magánkulcs kell, de üzenet titkosításához elég a küldő nyilvános kulcsát ismerni

A nyilvános kulcsú titkosítás vázlata

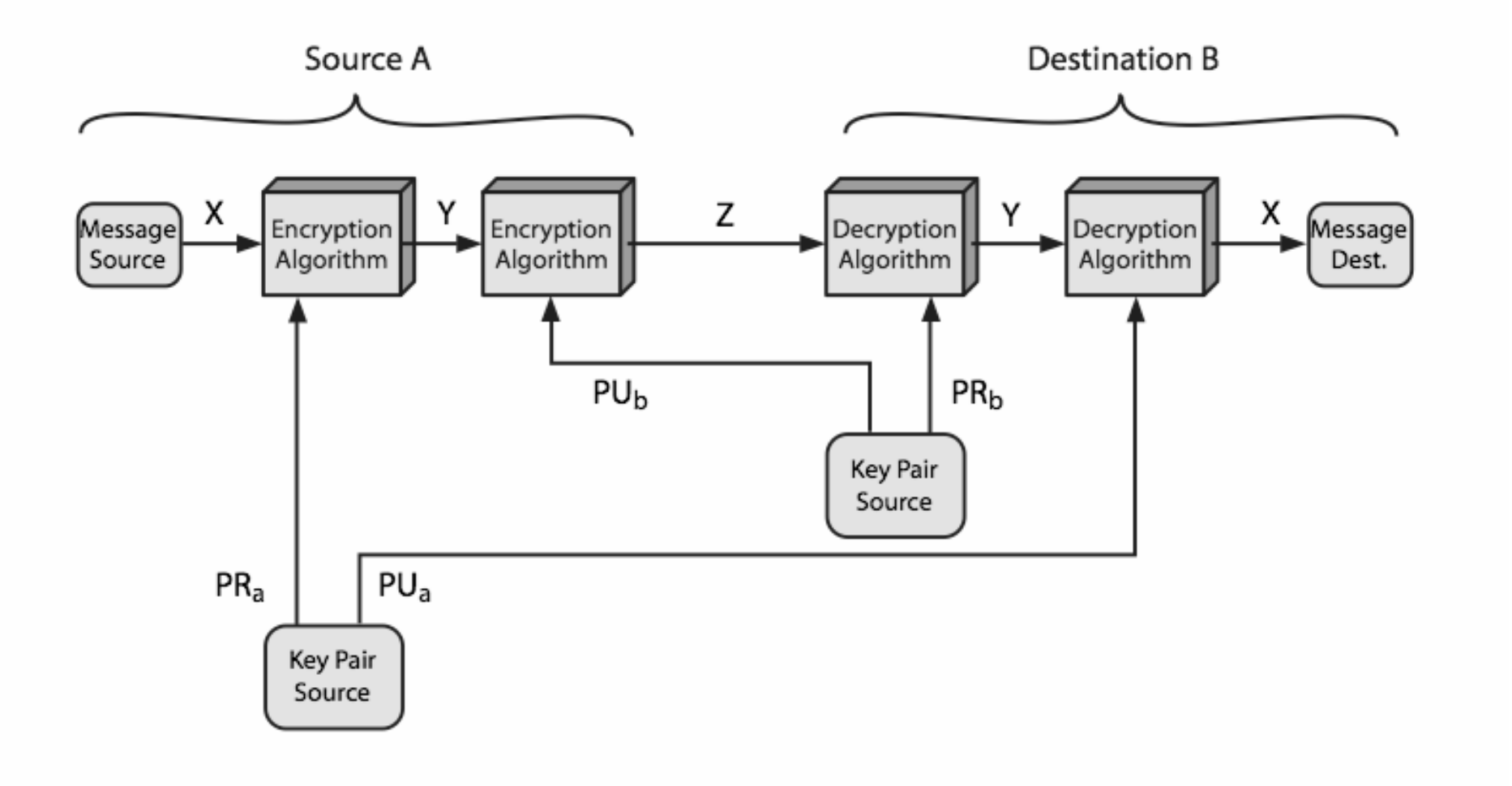


(a) Encryption

A két kulcs viszonya

- Feltételek a nyilvános kulcsú titkosítás működéshez:
  - a nyilvános kulcs ismeretében hatékonyan lehessen titkosítani
  - a magánkulcs ismeretében hatékonyan lehessen az üzenetet megfejteni
  - jelenlegi algoritmusainkkal reménytelenül sok ideig tartson a nyilvános kulcsból a magánkulcsot kiszámítani (a titkosító/megfejtő algoritmust ismeretét persze feltételezzük)
  - a magán kulcs ismerete nélkül szintén reménytelen számítási feladat legyen az üzenet megfejtése
  - hatékonyan tudjunk véletlen nyilvános-magán kulcspárokat generálni
- néhány algoritmusnál (pl. RSA) hasznos, hogy
  - a magánkulccsal is lehessen titkosítani, ami csak a nyilvános kulccsal fejthető meg (ezen alapul az aláírás)

Nyilvános kulcsú titkosítás és aláírás (elvi vázlat)



PR = private, magánkulcs, PU = public, nyilvános kulcs

A nyilvános kulcsú kriptográfia alkalmazásai

- 3 kategóriába osztható:
  - **titkosítás/megfejtés** (bizalmasságot ad)
  - **elektronikus aláírások** (hitelesítést ad)
  - **kulcscsere**
    (kapcsolatkulcsok (session keys) cseréjére)
- néhány algoritmus mindhárom feladatra alkalmas, mások csak egy-két célra használhatók

A nyilvános kulcsú rendszerek biztonsága I

- nem biztonságosabbak vagy kevésbé biztonságosabbak mint a titkos kulcsú rendszerek; a biztonság az alkalmazott algoritmustól, kulcs hossztól stb. függ nem a módszer típusától
- mint a titkos kulcsú rendszereknél itt is a **teljes kipróbálás** (brute force) feltörés legalább is elméletben mindig lehetséges
- de a gyakorlatban használt kulcsok a teljes kipróbálás meghiúsításánál jóval hosszabbak (> 512 bitesek)
- mert a titkosítás alapját képező számelméleti probléma nehéz irányának (pl. faktorizáció) kiszámítása ellen kell védekeznünk
- pl. 512-bit RSA ≈ 64-bit DES, 1024-bit RSA ≈ 80-bit DES

A nyilvános kulcsú rendszerek biztonsága II

- a biztonság a „könnyű irány” (titkosítás) és a „nehéz irány” (feltörés) közötti **elég nagy számítási különbségen** alapszik
- pl. RSA esetében
  - könnyű irány = szorzás, (illetve hatványozás mod p)
  - nehéz irány = faktorizáció (prímtényezőkre bontás)
- általánosságban a ,,nehéz irány’’ is algoritmussal megoldható, de **elég nehézzé kell tennünk** ahhoz, hogy a gyakorlatban kivitelezhetetlen legyen
- ehhez nagy (több százjegyű) számokra van szükség
- ezért a nyilvános kulcsú kriptográfia **jóval lassabb a titkos kulcsúnál**

RSA

- **R**ivest, **S**hamir & **A**dleman (MIT), 1977
- a legismertebb és legelterjedtebb nyilvános kulcsú algoritmus
- véges (Galois) test feletti hatványozáson alapszik valamilyen **`n`** modulusra nézve
  - $\mathbf{a^b \pmod n}$ kiszámításának időigénye $\mathbf{O((\log n)^3)}$ ez **polinomiális** a bemenet hosszának (**`log n`**) függvényében /könnyű/
- a modulus nagy szám (pl. 1024 bit)
- a biztonságát a faktorizáció nehézsége adja
  - erre ma csak **superpolinomiális algoritmusok ismertek** /nehéz/, pl. GNFS időigénye $O\left(\exp\left(\left(\frac{64}{9}n\right)^{\frac{1}{3}}(\log n)^{\frac{2}{3}}\right)\right)$ ahol `n` a bemenet hossza

RSA Kulcsgenerálás

Minden felhasználónak saját nyilvános/magán kulcspárra van szüksége, amit így generálhatunk:

1. válasszunk két nagy prímszámot véletlenszerűen: p, q
2. számítsuk ki a modulust $n=p \cdot q$
3. ekkor $\varphi(n)=(p-1)(q-1)$
   - ahol $\varphi(n)$ az Euler féle φ függvény az $\{1,\ldots,n\}$ számok közül az n-hez relatív prímek száma, pl $\varphi(6)=2$
4. válasszunk egy olyan e számot, melyre
   - $1<e<\varphi(n),\ (e,\varphi(n))=1$
5. számítsuk ki azt a d értékét, melyre
   - $e \cdot d=1 \bmod \varphi(n)$ and $0 \le d \le n$
6. a nyilvános kulcs: **PU={e,n}**
7. a titkos kulcs: **PR={d,n}**

➢ p és q értékét szintén titokban kell tartani, vagy meg kell semmisíteni, bár sebesség szempontjából, még jó jöhet

Az RSA használata

- az nyílt szöveget először 1 és n közötti számokkal kell kódolnunk, legyen M a titkosítandó blokk értéke, ahol $0 \leq M < n$
- az M üzenet titkosításához a küldő
  - a címzett $PU=\{e,n\}$ nyilvános kulcsával
  - kiszámítja a $C = M^e \bmod n$ értéket
- a C üzenet megfejtéséhez a címzett
  - a $PR=\{d,n\}$ magánkulcsát használva
  - kiszámítja az $C^d \bmod n$ értéket, ami éppen M

Miért működik ?

- az Euler-Fermat tétel szerint
  - $a^{\varphi(n)} \bmod n = 1$, amennyiben $(a,n)=1$
- az RSA esetében:
  - $n=p \cdot q$
  - $\varphi(n)=(p-1)(q-1)$
  - e és d egymás inverzei $\bmod \varphi(n)$
  - ezért $e \cdot d=1+k \cdot \varphi(n)$ valamilyen k-ra
- ezért ha $(M,n) = 1$, akkor

$$C^d = M^{e \cdot d} = M^{1+k \cdot \varphi(n)} = M^1 \cdot (M^{\varphi(n)})^k$$

$$= M^1 \cdot (1)^k = M^1 = M \bmod n$$

- az $(M,n) \neq 1$, azaz az $M=p$ vagy $M=q$ eset hasonlóan igazolható (HF!)

RSA minipélda - Kulcsgenerálás

1. Választott prímek: $p=17$ és $q=11$
2. Számítás: $n = pq = 17 \cdot 11 = 187$
3. Számítás: $\varphi(n)=(p-1)\cdot(q-1)=16\cdot 10=160$
4. Választás e: melyre $(e,160)=1$; legyen $e=7$
5. Számítás d: melyre $de=1 \bmod 160$ és $d < 160$
   Most $d=23$ mert $23\cdot 7=161=10\cdot 160+1$
6. Nyilvános kulcs: **PU={7,187}**
7. Magánkulcs: **PR={23,187}**

RSA minipélda – titkosítás/megfejtés

- a példát folytatva
- legyen a nyílt szöveg $M = 88$
- M szabályos blokk, mert $88<187$
- titkosítás:

  $C = 88^7 \bmod 187 = 11$

- megfejtés:

  $M = 11^{23} \bmod 187 = 88$

Gyors hatványozás modulo n

- az iterált négyzetreemelések módszerével (Square and Multiply Algorithm)
- gyors hatékony algoritmus (moduláris) hatványozásra
- a kitevő bináris reprezentációját tekintjük
- az alapot ismételten négyzetre emeljük
- és ezen hatványok közül azokat szorozzuk össze, amelyek a hatványérték kiszámításhoz valóban kellenek, mert a kitevő megfelelő bitje 1-es
- így $O(\log_2 n)$ szorzás kell, ha a kitevő legfeljebb n
  - pl. $3^{135} = 3^{128} \cdot 3^4 \cdot 3^1 = 5 \cdot 4 \cdot 3 = 5 \bmod 11$
  - mert $3^1=3$, $3^2=9$, $3^4=4$, $3^8=5$, $3^{16}=3$, $3^{32}=9$, $3^{64}=4$, $3^{64}=5$, $3^{128}=5$

a^b mod n kiszámítása, ahol $b=b_k \ldots b_0$

```
c = 0; f = 1
for i = k downto 0
    do c = 2 * c
       f = (f * f) mod n
    if b_i == 1 then
       c = c + 1
       f = (f * a) mod n
 return f
```

Megj: c –re nincs szükség csak szemlélteti a kitevő aktuális értékét

A titkosítás hatékonyan megvalósítható

- a titkosítás `e` kitevőre hatványozás mod `n`
- Ezért ha `e` kicsi, a titkosítás gyorsabb
  - gyakori választás: e=65537=(2^{16}+1)
  - vagy szóba jöhet: e=3 or e=17
- de ha `e` túl kicsi (pl. `e=3`) akkor feltörhető
- ha e-t rögzítjük, akkor **`(e, φ(n))=1`**-t **`n`** megválasztásával kell biztosítanunk,
  - pl. elutasítva minden **`p`**-t és **`q`**-t melyekre **`p-1`** és **`q-1`** nem relatív prímek **`e`**-hez

A megfejtés is hatékony

- a megfejtés **d**-dik hatványra emelés
  - **d** valószínűleg nagy szám, különben a rendszer nem biztonságos
  - alkalmazható a kínai maradéktétel a hatványérték **mod p** majd **mod q** külön kiszámításához, majd a részeredmények összekombinálásához
  - ez kb. 4 –szer gyorsabb mint a direkt módszer
  - csak a titkos kulcs, pontosabban **p** és **q** ismerője tudja ezt a gyorsítást használni

RSA kulcsgenerálás elmélete I

Az RSA kulcsaihoz szükséges:

- két nagy véletlen prímszám meghatározása: **p** és **q**
- e vagy d egyikének kiválasztása, és a másik kiszámítása
- **p** és **q** nem lehet az **n = p·q** szorzatból könnyen kiszámítható pl.
  - nem lehet az egyik sem túl kicsi
  - nem eshetnek közel a **n** négyzetgyökéhez
  - **p-1**-nek és **q-1**-nek is kell, hogy legyen nagy prímosztója
  - de **(p-1,q-1)** kicsi legyen

RSA kulcsgenerálás elélete II

- általában a prímeket véletlen generálással + valószínűségi teszteléssel állítják elő
  - Fermat teszt (néha hibás eredményt ad)
  - Miller-Rabin teszt (valószínűségi teszt)
  - Solovay-Strassen teszt (valószínűségi teszt)
  - AKS teszt (determinisztikus, elméleti) ezért **PRÍMEK∈P** (2004-ben!)

  Ld: bonyolultságelmélet illetve
  http://en.wikipedia.org/wiki/Primality_test

- a kitevők **e, d** egymás inverzei $\bmod \varphi(n)$ egymásból és $\varphi(n)$-ből az általánosított Euklideszi algoritmussal számíthatók

Az RSA biztonsága

- lehetséges támadási típusok ellene:
  - a kulcsok teljes kipróbálása (kivitelezhetetlen a számok nagysága miatt)
  - matematikai támadások
  - időméréses támadások (a megfejtő algoritmuson)
  - választott titkos szöveg alapú támadások

Matematikai támadások

- matematikai támadások fajtái:
  - faktorizáció (**n**–ből **p** és **q** meghatározása)
  - **φ(n)** kiszámítása
  - **d** direkt kiszámítása
- az első kettő egyforma nehéz:
  - p és q ismeretében $\varphi(n)=(p-1)(q-1)$
  - $\varphi(n)$ **és** n ismeretében
    $n = pq$ **és** $\varphi(n)=(p-1)(q-1)$
    p-re és **q**-ra megoldható, mert másodfokú egyenletrendszer
- sejtés, hogy **d** kiszámítása is a faktorizációval polinomiálisan ekvivalens

A faktorizáció nehézsége

- a faktorizáló algoritmusok csak lassú ütemben fejlődnek (QS -> GNFS -> LS)
- Ld: Paul Zimmermann's factoring page
  - 2005 májusában a rekord 200 jegyű (663 bites) szám felbontása LS (Lattice Sieve) algoritmussal (55 processzorév lenne egyetlen 2.2 GHz Opteron CPU-val.)
  - pedig díjakat is lehetett nyerni érte: Ld. RSA factoring challange: http://www.rsasecurity.com/rsalabs/node.asp?id=2879
- így ma egy 1024 vagy inkább **2048** bites RSA biztonságos feltéve, hogy
  - `p` és `q`, `e` és `d` a feltételeknek megfelelően
  - és valóban véletlenül választott
  - az algoritmus biztonságosan implementált
  - a titkos kulcs biztonságosan tárolt

Időméréses támadások (Timing Attacks)

- feltalálójuk Paul Kocher a 90-es évek közepén
- különböző műveletek időigénye eltérő
  - pl. kis vagy nagy számmal való szorzás
  - az IF-ek mely ága kerül végrehajtásra
- így a használt titkos kulcstól (kitevő!) függ a megfejtés végrehajtásának ideje
- az RSA erre igen érzékeny, mert a megfejtéskor a titkos kulcs a kitevő, erősen hat az időigényre
- védekezés ellene
  - konstans idejű hatványozás implementálása (lassít)
  - véletlen szünetek beiktatása
  - blinding (megvakítás): véletlen értékkel szorzás a hatványozás előtt, majd korrigálás (csak + 2–10 %)

Választott titkos szöveg alapú támadás

- az RSA választott titkos szöveg alapú támadásokra sérülékeny
- a támadó a feltöréshez titkos szövegeket jelölhet ki, melyek megfejtését megkapja
- választhatjuk titkos szövegeket úgy, hogy azok elősegítsék a kriptoanalízist
- ellenszere: a nyíltszöveg véletlen szöveggel való feltöltése (padding)
- nevezetesen: Optimal Asymmetric Encryption Padding (OASP) /bonyolult/

Felhasznált irodalom

- Virrasztó Tamás: Titkosítás és adatrejtés: Biztonságos kommunikáció és algoritmikus adatvédelem, NetAcademia Kft., Budapest, 2004. Online elérhető: http://www.netacademia.net/book.aspx?id=1#

- William Stallings: Cryptography and Network Security, 4th Edition, Prentice Hall, 2006. (Chapter 9)
- Lawrie Brown előadás fóliái (Chapter 9)
- http://en.wikipedia.org/wiki/RSA