

A nyílt kulcsú titkosítás és a digitális aláírás

Készítette: Fuszenecker Róbert
Konzulens: Dr. Tuzson Tibor, docens

Budapest Műszaki Főiskola
Kandó Kálmán Műszaki Főiskolai Kar
Műszertechnikai és Automatizálási Intézet

2006. november

Ez a dokumentum szabad szoftver, szabadon terjeszthető és/vagy módosítható a GNU GPL 2.0-ban leírtak szerint.

Az előadás vázlata

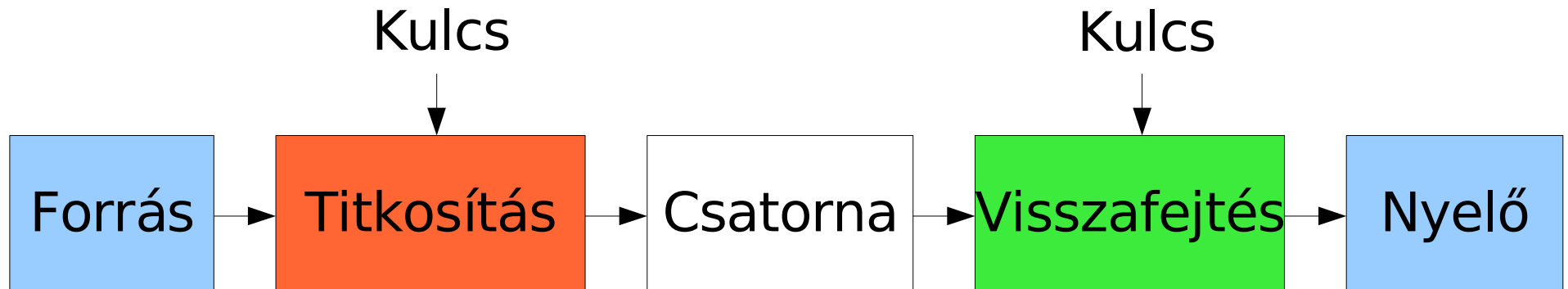
- Kommunikáció, a titkosítás definíciója
- A szimmetrikus titkosítás, előnye és hátránya
- Diffie-Hellman kulcscsere algoritmus, matematikai háttér, a „középen levő ember” problémája
- RSA algoritmus: matematikai háttér, titkosítás és aláírás
- Lehetőségek a DH és az RSA törésére
- A nyílt kulcsú titkosítás gyakorlati felhasználása
- Képernyőképek

Kommunikáció titkosítatlan csatornán



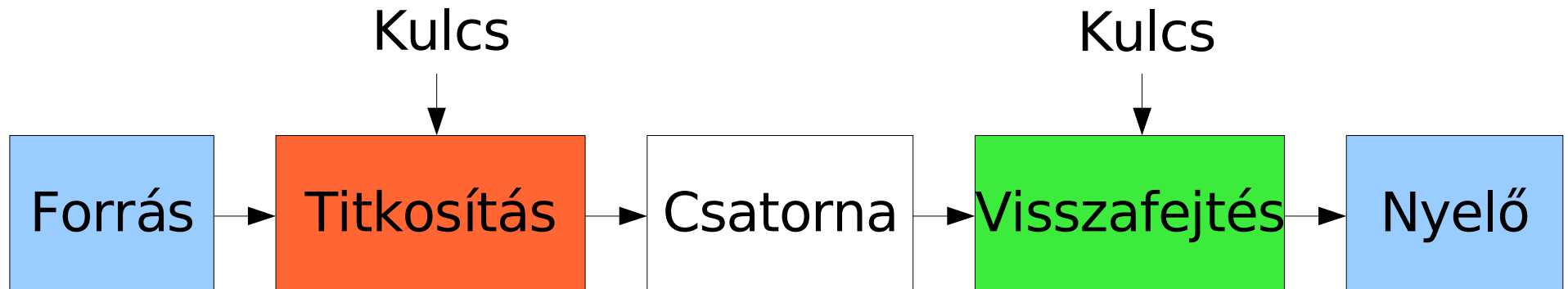
- A kommunikáció során az információ a **csatornán** keresztül jut el a forrástól a nyelőig (célállomásig)
- A csatornát gyakran többen használják, így egymást zavarhatják, „lehallgathatják”

Kommunikáció titkosított csatornán



- A titkosítás során az információt úgy alakítjuk át, hogy az csak a megfelelő állomás(ok) számára legyen értelmezhető
- A visszafejtés a titkosított információt visszaalakítását jelenti

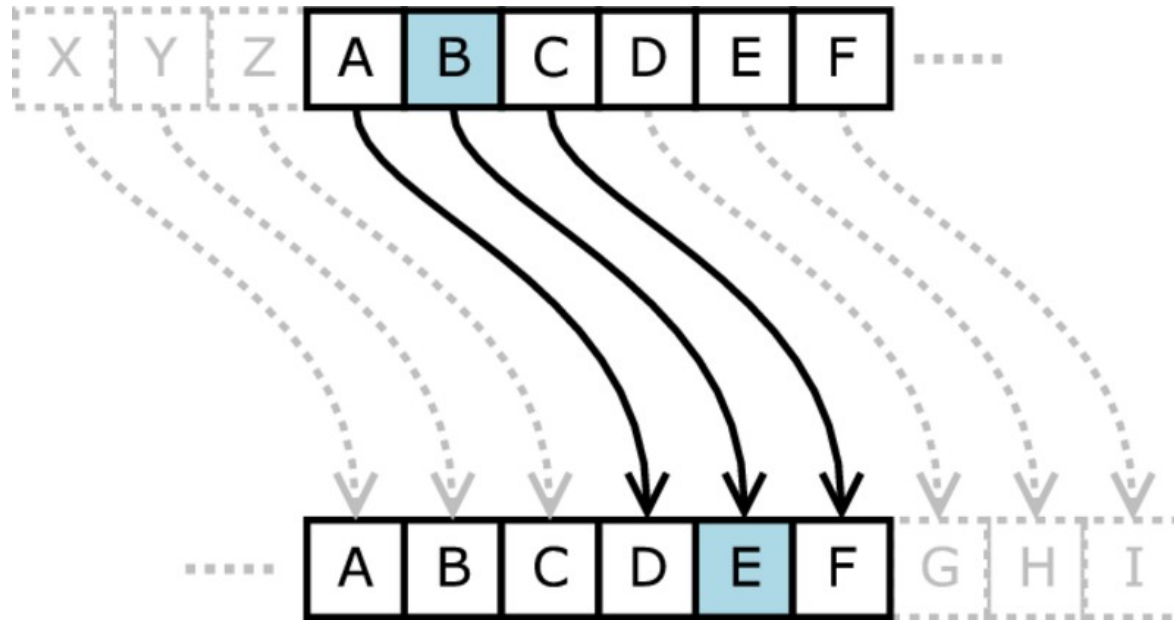
Szimmetrikus (kulcsú) titkosítás



A visszafejtés a titkosítás inverz művelete, pl: ha a titkosítás összeadás, akkor a visszafejtés a kivonás

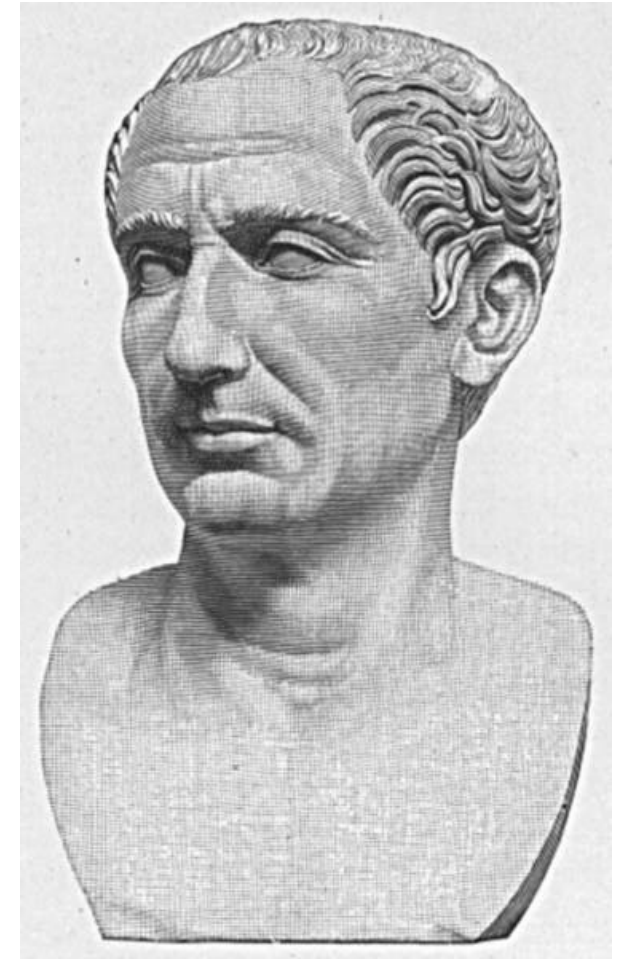
Példa: Caesar-módszer

Caesar-módszer



Helyettesítő algoritmus, a **kulcs** értéke 3.

Könnyen törhető → gyakoriság elemzés



Caius Iulius Caesar
Kr. e. 100. – Kr. e. 44.

Egyéb szimmetrikus algoritmusok

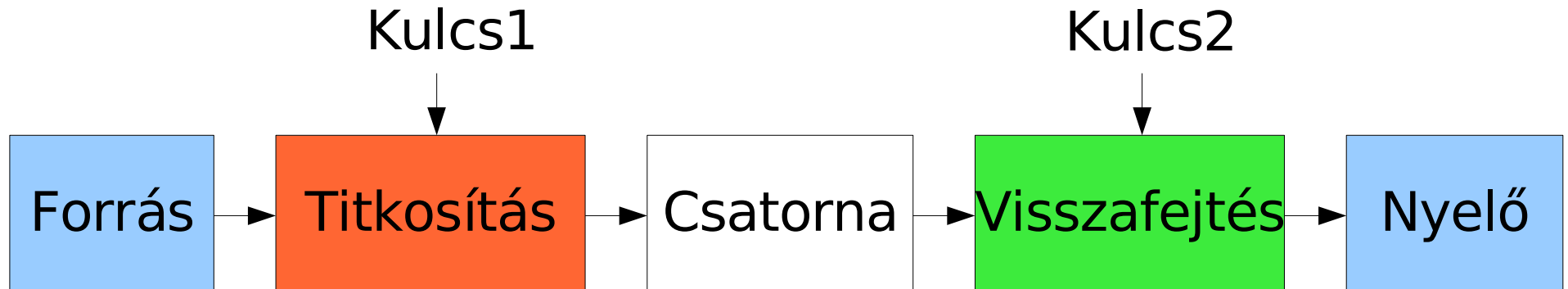
- DES, 3DES: Data Encryption Standard (történelem)
- AES: Advanced Encryption Standard, eredetileg Rijndael
- CAST: a tervezők után
(Carlisle Adams and Stafford Tavares)
- Twofish: nehezen törhető, szabadon felhasználható
- XTEA: eXtended Tiny Encryption Algorithm

További algoritmusok a http://en.wikipedia.org/wiki/Block_cipher címen találhatók.

Szimmetrikus titkosítás hátránya

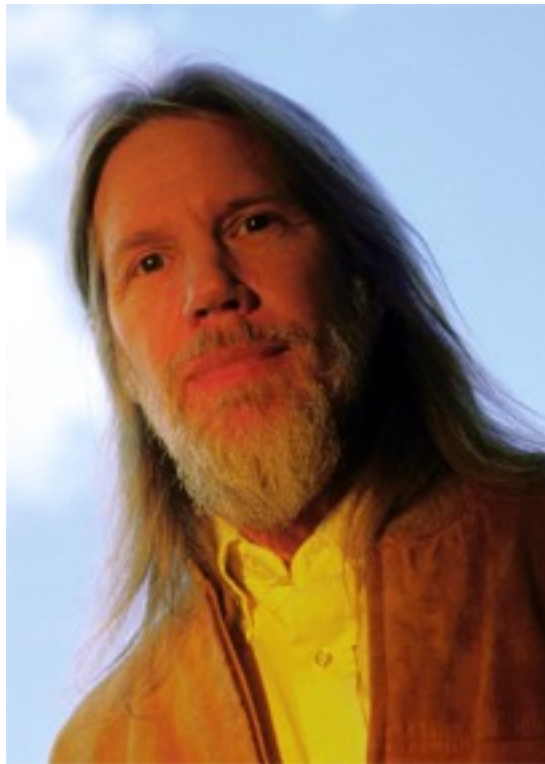
- Gondoskodni kell a kulcsnak (titok, jelszó, stb.) a nyelőhöz történő biztonságos eljuttatásáról
- Ez nehezen valósítható meg, mert a csatorna nem biztonságos, más átviteli közeg pedig általában nem áll rendelkezésre, vagy nehezen használható
- A kulcsokat a biztonság növelése érdekében időnként meg kell változtatni

A nyílt kulcsú titkosítás



A „visszafejtés” nem inverz művelete a „titkosítás”-nak, hanem valamilyen speciális matematikai azonosságot használunk ki

A Diffie-Hellman kulcscsere (1976)



Whitfield Diffie



Martin Hellman

A Diffie-Hellman kulcscsere algoritmus

Nem az információt titkosítjuk, hanem kiszámítunk egy ideiglenes kulcsot úgy, hogy:

- a számítás végén a forrás és a nyelő oldalán ugyanazt az eredményt kapjuk
- a kulcs sose haladjon keresztül a csatornán, legfeljebb a számítás részeredményei
- az információt a valamilyen szimmetrikus algoritmussal titkosítjuk, a kiszámított kulcs felhasználásával

A Diffie-Hellman kulcscsere algoritmus

„**A**” állomás



Csatorna



„**B**” állomás

Közös paraméterek: **g** (generátor) és **p** (prímszám)

Titkos kulcs: **a**

Titkos kulcs: **b**

Nyilvános kulcsok: **A** és **B**, ahol **$A = g^a \bmod p$** és **$B = g^b \bmod p$**
Ezeket egymással kicserélik a csatornán keresztül

$K = B^a \bmod p$

$K = A^b \bmod p$

A Diffie-Hellman kulcscsere algoritmus

Az „A” oldal számításai:

$$\mathbf{K_A = B^a \bmod p = (g^b \bmod p)^a \bmod p = g^{ba} \bmod p}$$

A „B” oldal számításai:

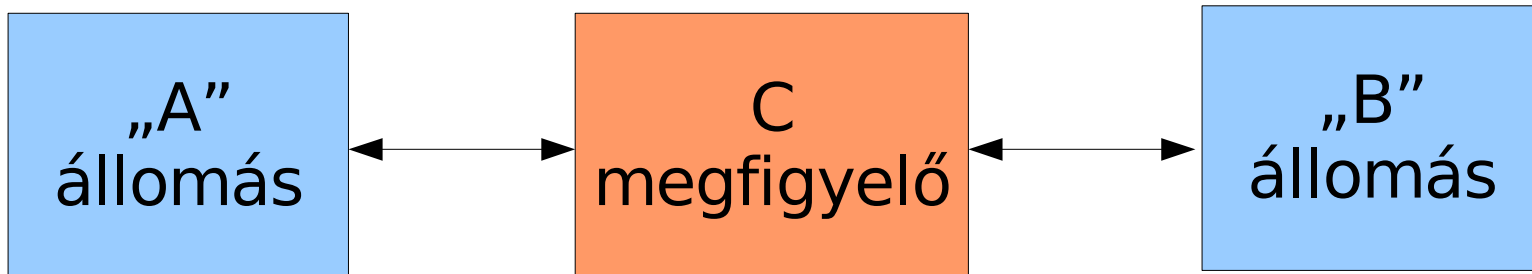
$$\mathbf{K_B = A^b \bmod p = (g^a \bmod p)^b \bmod p = g^{ab} \bmod p}$$

Ha $\mathbf{ab = ba}$, akkor $\mathbf{g^{ab} \bmod p = g^{ba} \bmod p}$, így

$\mathbf{K_A = K_B = K} \rightarrow$ ez a kulcs a szimmetrikus titkosításhoz

A Diffie-Hellman kulcscsere algoritmus

A „középen levő ember” (man in the middle) támadás:



Ha „C” azt állítja „A”-nak, hogy ő valójában „B”, „B”-nek pedig azt, hogy ő „A”, akkor megfigyelheti – lehallgathatja – az „A” és „B” közötti kommunikációt.

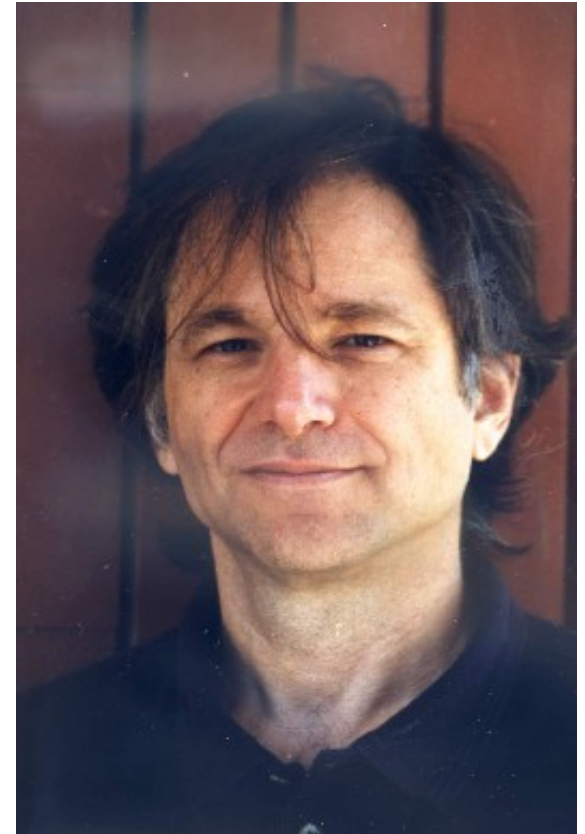
Az RSA algoritmusa (1977)



Ron Rivest



Adi Shamir



Leonard Adleman

Az RSA algoritmus

Az algoritmus Fermat kis tételére épül:

$$a^p \bmod p = a$$
$$8^{13} \bmod 13 = 549755813888 \bmod 13 = 8$$

ahol

- **p** egy prímszám, és **a** egy tetszőleges egész szám
- **a** és **p** relatív prímek

Az RSA algoritmusa

Ha $a^{p-1} \bmod p = 1$ és $a^{q-1} \bmod p = 1$, akkor a kínai maradék-tétel szerint

$$\begin{aligned} a^{(p-1)(q-1)} \bmod pq &= 1 \\ a^{k(p-1)(q-1)} \bmod pq &= 1^k = 1 \\ a^{k(p-1)(q-1)+1} \bmod pq &= a \end{aligned}$$

ahol

- p és q különböző prímszámok, és a és k tetszőleges egész számok
- a , p és q relatív prímek

Az RSA algoritmusa

Bontsuk fel a $k(p-1)(q-1)+1$ értékét két szám szorzatára:

$$k(p-1)(q-1)+1 = ed$$

ahol **k**, **e** és **d** egész számok. Nevezzük az **e** számot titkosító kitevőnek, míg **d**-t pedig visszafejtő kitevőnek. A nyilvános kulcsot (**e** és **pq**) tegyük elérhetővé mindenki számára, a titkos kulcsot (**d** és **pq**) tartsuk titokban.

Az RSA algoritmus

Mivel **e**-t és **pq**-t mindenki megszerezheti (hiszen nyilvánosak), ezért bárki küldhet nekünk titkosított üzenetet:

$$\mathbf{M} = \mathbf{m}^e \bmod pq,$$

amit mi könnyen visszafejthetünk:

$$\begin{aligned} \mathbf{m}^* &= \mathbf{M}^d \bmod pq = \mathbf{m}^{ed} \bmod pq = \\ &= \mathbf{m}^{k(p-1)(q-1)+1} \bmod pq = \mathbf{m}, \end{aligned}$$

ahol **m** a titkosítandó (és visszafejtett) és **M** a titkosított üzenet

Digitális aláírás készítése RSA algoritmussal

A digitális aláírás egy dokumentum hitelességét igazolja.
A dokumentumból képzett szám (**h**) segítségével számítsuk ki

$$S = h^d \bmod pq$$

értékét. A **h** általában valamilyen ellenőrző összeg (pl. a betűk, számjegyek, írásjelek kódjának összege), vagy egy speciális (hash) függvény értéke (pl. MD5, SHA1, SHA256), de mindenképpen a dokumentumra jellemző **egyedi** szám!

Digitális aláírás ellenőrzése

A digitális aláírást (**S**) csatoljuk a dokumentumhoz.

Ellenőrzéskor (az ellenőrző félnek) újra ki kell számítani a dokumentumra jellemző egyedi számot (MD5, SHA1, SHA256, stb., továbbiakban: **j**-t), és vissza kell fejtenie az általunk küldött azonosítót):

$$\mathbf{j^* = S^e \bmod pq}$$

Digitális aláírás ellenőrzése

A dokumentum akkor hiteles, ha $j = j^*$, és biztos, hogy a nyilvános kulcs tőlünk származik. A nyilvános kulcsok tárolhatók kulcs-szervereken.

Mivel az aláírást csak mi állíthattuk elő (mert **csak nekünk van meg a nyilvános kulcs „párja”**), ezért az általunk megadott ellenőrző összeg biztosan tőlünk származik. Ha a dokumentum jelenlegi ellenőrző összege is ugyanez az érték, akkor a dokumentum változatlanul jutott át a csatornán, tehát hiteles.

Egyéb nyílt kulcsú titkosító rendszerek

- DSA: csak aláírásra használható
- ECDH: elliptikus görbéken értelmezett Diffie-Hellman, csak titkosításra
- ECDSA: elliptikus görbéken értelmezett DSA, csak aláírásra
- NTRU: titkosításra és aláírásra
- stb.

Miért nehéz megtörni a DH-t?

Diffie-Hellman kulcscsere esetén a részeredmények áthaladnak a csatornán, így azokat bárki megszerezheti. A harmadik félnek akkor sikerülne a **K** kulcs értékét kiszámítani, ha megszerezné **a** vagy **b** értékét. Ehhez azonban meg kell oldania a következő egyenletet:

$$\mathbf{A} = \mathbf{g}^a \bmod \mathbf{p}$$

$$\mathbf{g}^a = \mathbf{k}\mathbf{p} + \mathbf{A}$$

Ez nagyon nehéz, hiszen **k** és **a** ismeretlen → kétismeretlenes egyenlet
(**p** legyen igen nagy [2048-4096 bit hosszú]).

Miért nehéz megtörni a RSA-t?

Az RSA ugyanúgy a hatványozásra épül, mint a DH algoritmus, így a kitevők meghatározása ebben az esetben is zsákutca.

Az RSA törése „könnyebb”, ha a **pq** szorzatból próbáljuk kiszámolni **p** és **q** értékét, azaz megpróbáljuk faktorizálni **pq** értékét. **p** és **q** ismeretében könnyen megtalálhatjuk a megfelelő titkos kulcsot.

Az RSA akkor törhető nehezen, ha **p** és **q** igen nagy, több ezer bites, tipikusan 1024-2048 bit hosszúak.

Mire használhatjuk a nyílt kulcsú titkosítást?

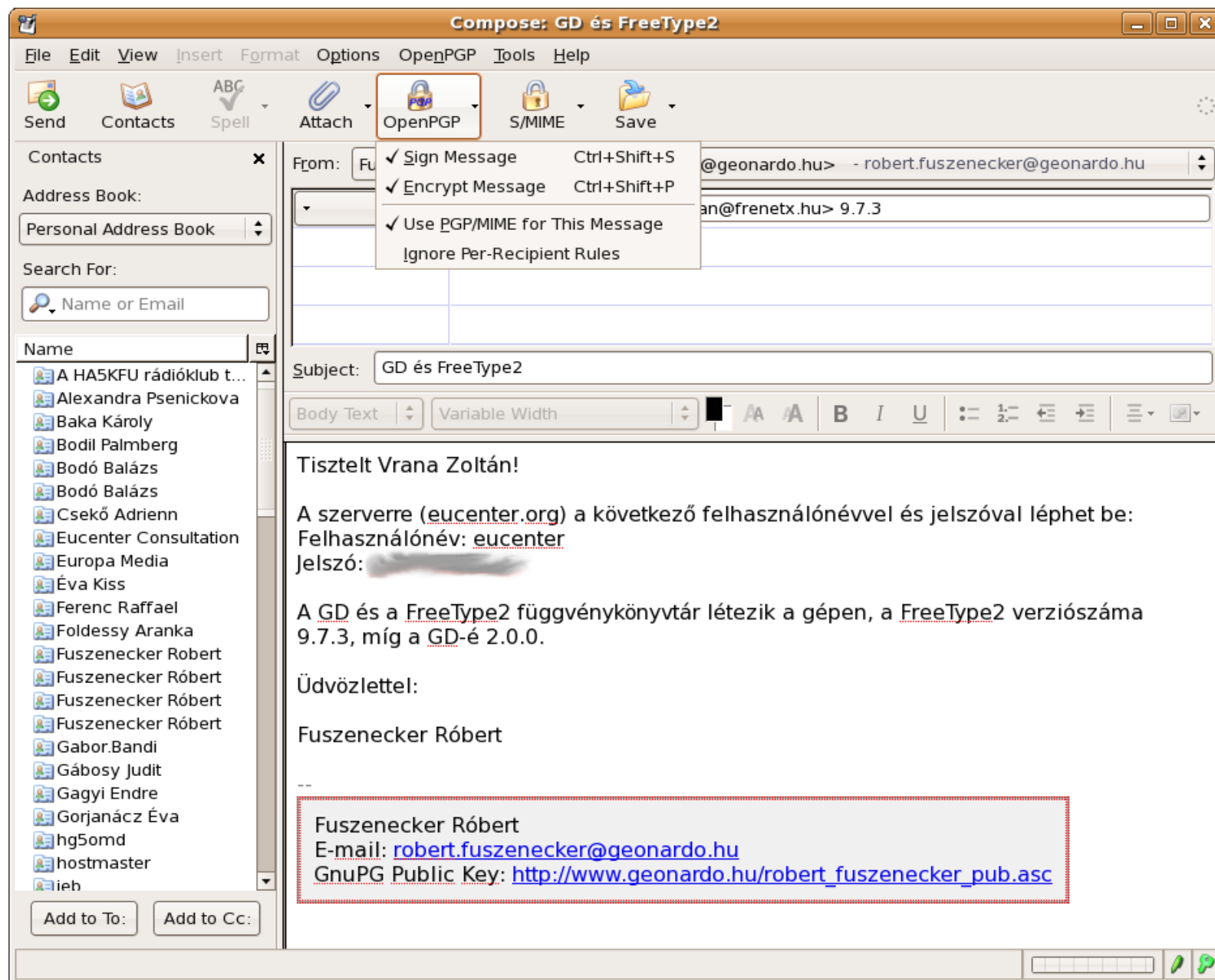
A titkosítást:

- levelezés: bizalmas információkat titkosíthatunk (pgp, OpenPGP, GnuPG, enigmmail)
- biztonságos belépés távoli számítógépre (ssh)
- interneten történő vásárlás, távoli adminisztrálás (https)

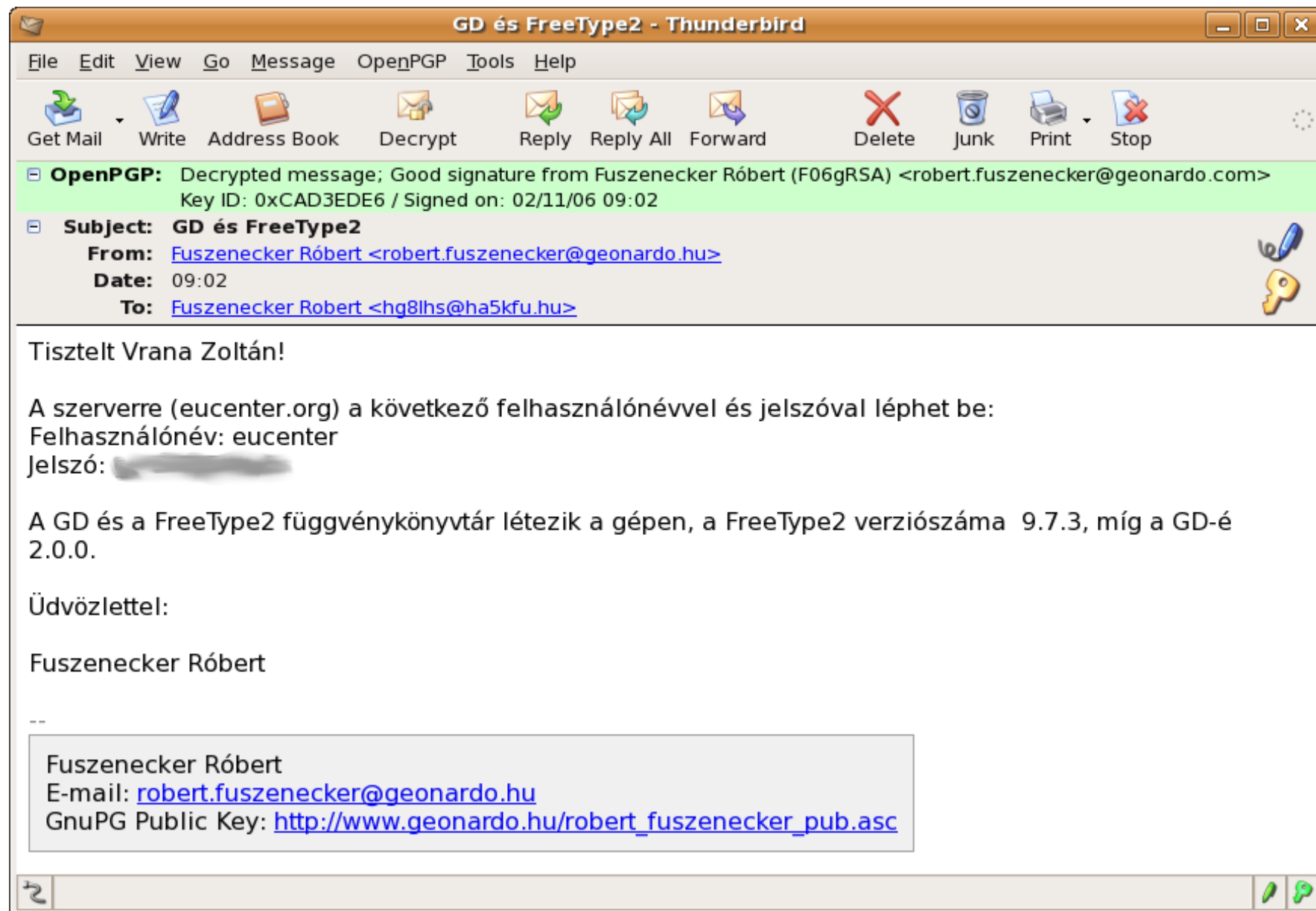
A digitális aláírást:

- levelezés: eredetiség igazolása
- internetes ügyintézés

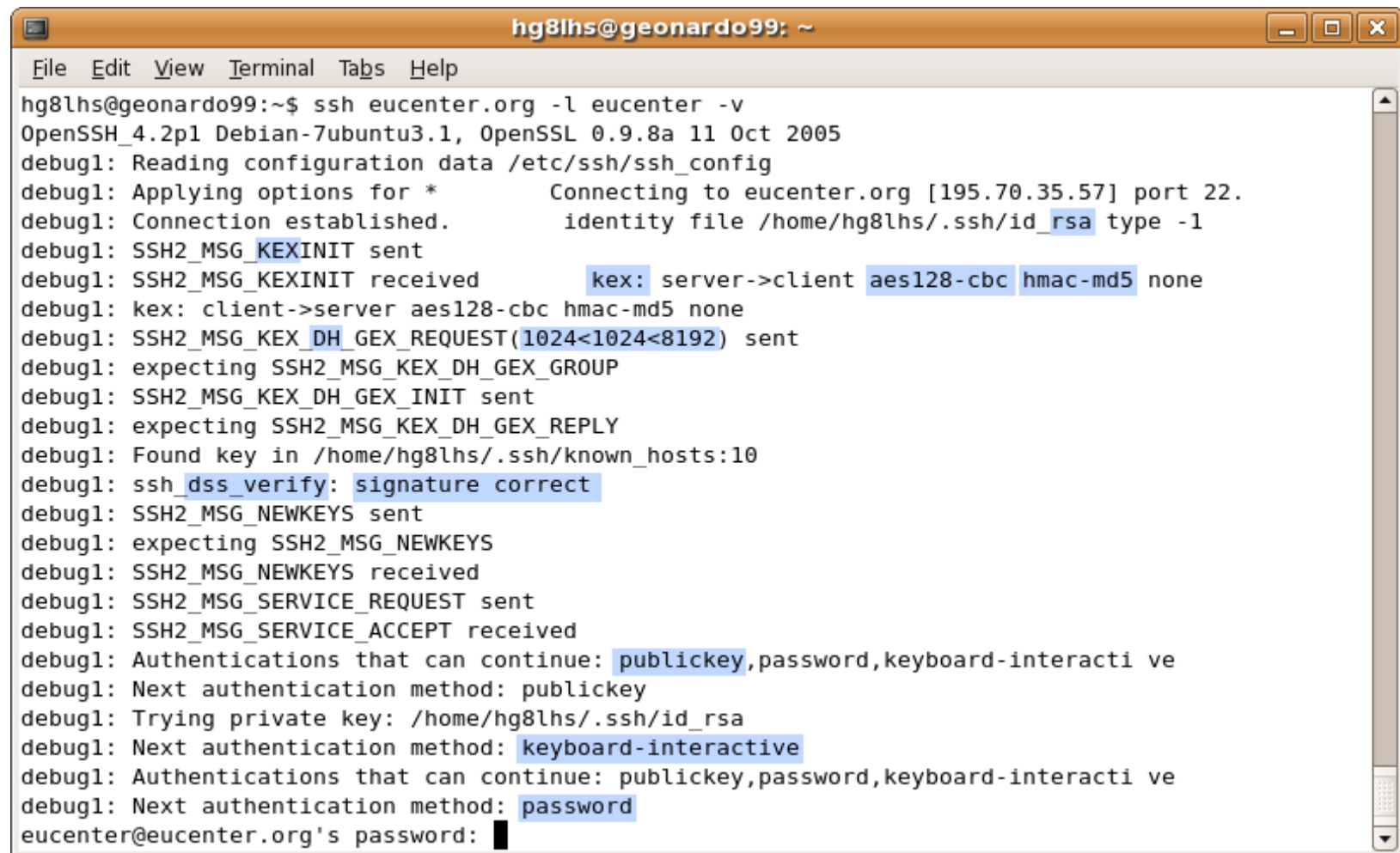
A nyílt kulcsú titkosítás és a digitális aláírás



A nyílt kulcsú titkosítás és a digitális aláírás



A nyílt kulcsú titkosítás és a digitális aláírás



```
hg8lhs@geonardo99: ~  
File Edit View Terminal Tabs Help  
hg8lhs@geonardo99:~$ ssh eucenter.org -l eucenter -v  
OpenSSH_4.2p1 Debian-7ubuntu3.1, OpenSSL 0.9.8a 11 Oct 2005  
debug1: Reading configuration data /etc/ssh/ssh_config  
debug1: Applying options for *      Connecting to eucenter.org [195.70.35.57] port 22.  
debug1: Connection established.      identity file /home/hg8lhs/.ssh/id_rsa type -1  
debug1: SSH2_MSG_KEXINIT sent  
debug1: SSH2_MSG_KEXINIT received      kex: server->client aes128-cbc hmac-md5 none  
debug1: kex: client->server aes128-cbc hmac-md5 none  
debug1: SSH2_MSG_KEX_DH_GEX_REQUEST(1024<1024<8192) sent  
debug1: expecting SSH2_MSG_KEX_DH_GEX_GROUP  
debug1: SSH2_MSG_KEX_DH_GEX_INIT sent  
debug1: expecting SSH2_MSG_KEX_DH_GEX_REPLY  
debug1: Found key in /home/hg8lhs/.ssh/known_hosts:10  
debug1: ssh_dss_verify: signature correct  
debug1: SSH2_MSG_NEWKEYS sent  
debug1: expecting SSH2_MSG_NEWKEYS  
debug1: SSH2_MSG_NEWKEYS received  
debug1: SSH2_MSG_SERVICE_REQUEST sent  
debug1: SSH2_MSG_SERVICE_ACCEPT received  
debug1: Authentications that can continue: publickey,password,keyboard-interactive  
debug1: Next authentication method: publickey  
debug1: Trying private key: /home/hg8lhs/.ssh/id_rsa  
debug1: Next authentication method: keyboard-interactive  
debug1: Authentications that can continue: publickey,password,keyboard-interactive  
debug1: Next authentication method: password  
eucenter@eucenter.org's password: █
```

A nyílt kulcsú titkosítás és a digitális aláírás

geonardo.hu / localhost / geonardo_eu2007 / opinion | phpMyAdmin 2.9.0.2 - Mozilla Firefox

Fájl Szerkesztés Nézet Előzmények Könyvtárak Eszközök Súgó

https://geonardo.hu:2083/3rdparty/phpMyAdmin/index.php

Gmail Google HUP Wikipedia EN SZTAKI Szótár: szot... Elektro Helyek NEPTUN Mindentudás egyet...

Google Keresés PageRank Helyesírás Feedek Beállítások

cPanel X geonardo.hu / localhost ...

phpMyAdmin

Adatbázis
_eu2007 (3)

geonardo_eu2007 (3)

- opinion
- quiz
- updates

Szerver: localhost ▶ Adatbázis: geonardo_eu2007 ▶ Tábla: opinion

Tartalom Struktúra SQL Keresés Beszúr Export Import

Tevékenységek Kiürít Eldob

Sorok megjelenítése 0 - 0 (1 Összesen, A lekérés lefutott 0.0003 másodperc alatt)

SQL-kérés:

```
SELECT *  
FROM `opinion`  
LIMIT 0, 30
```

[Szerkeszt] [SQL magyarázat] [PHP kód készítése] [Frissítés]

Query results operations

Nyomtatási nézet Nyomtatási nézet (összes szöveggel) Export

Mutat : 30 sor kezdve ezzel: 0

vízszintes módon, a fejléccet 100 soronként megismételve

	id	opinion1	opinion2	opinion3	opinion4
<input type="checkbox"/>	1	9	12	6	4

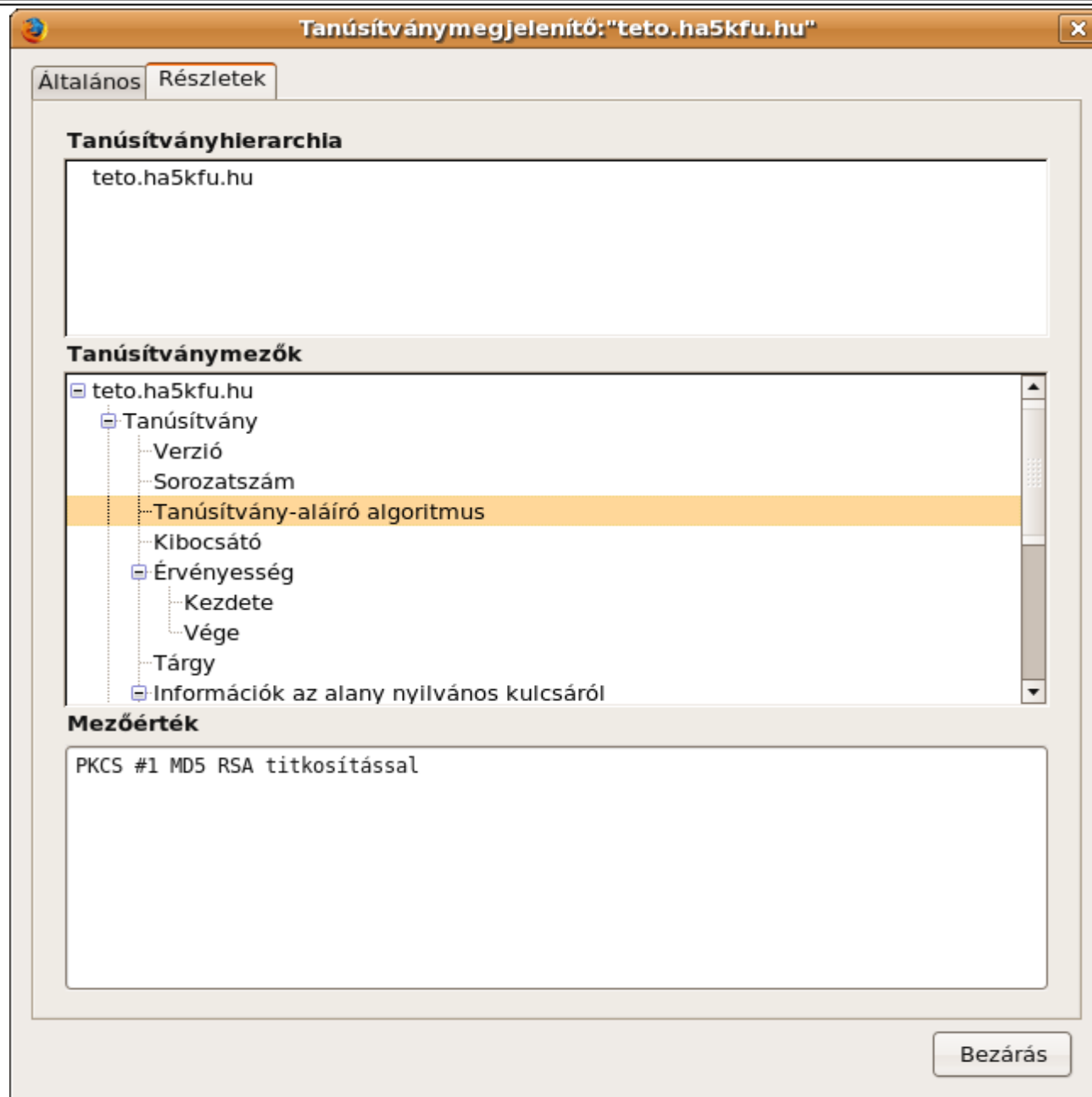
↑ Összeset kijelöli / Összeset törli A kijelöltekkel végzendő művelet: ☐ ☒ ☐

Mutat : 30 sor kezdve ezzel: 0

vízszintes módon, a fejléccet 100 soronként megismételve

Kész geonardo.hu:2083

A nyílt kulcsú titkosítás és a digitális aláírás



Összefoglalás

- Kommunikáció: titkosítatlan és titkosított
- A szimmetrikus titkosítás: Caesar-módszer hátránya: a kulcs átvitele
- Diffie-Hellman kulcscsere algoritmus
probléma: a „középen levő ember” problémája
megoldás: a nyilvános kulcs aláírása
- RSA algoritmus: matematikai háttér, titkosítás és aláírás
- Lehetőségek a DH és az RSA törésére: nagyon nehéz, szinte lehetetlen
- A nyílt kulcsú titkosítás gyakorlati felhasználása: levelezés, internetes vásárlás, stb.

Köszönöm a figyelmet!

Ez az előadás letölthető a <http://hg8lhs.ham.hu/tdk> oldalról.

Ez a dokumentum szabad szoftver, szabadon terjeszthető és/vagy módosítható a GNU GPL 2.0-ban leírtak szerint.