Bevezetés a kriptográfiába

Sulyok András Attila

2018. 09. 27.[1]

Bevezetés a számítástechnikába

[1] Utoljára módosítva: 2018. október 4.

Módszerek

Biztonságos rendszert tervezni nehéz. Saját kriptográfiai algoritmust írni **még nehezebb**.

Ötlet:
találjunk egy matematikai problémát, amelyet könnyű megoldani, de az inverzét nagyon nehéz.
(legalábbis sok időbe telik)

Pl.: prímszámfaktorizáció

Titkosítás (encryption)

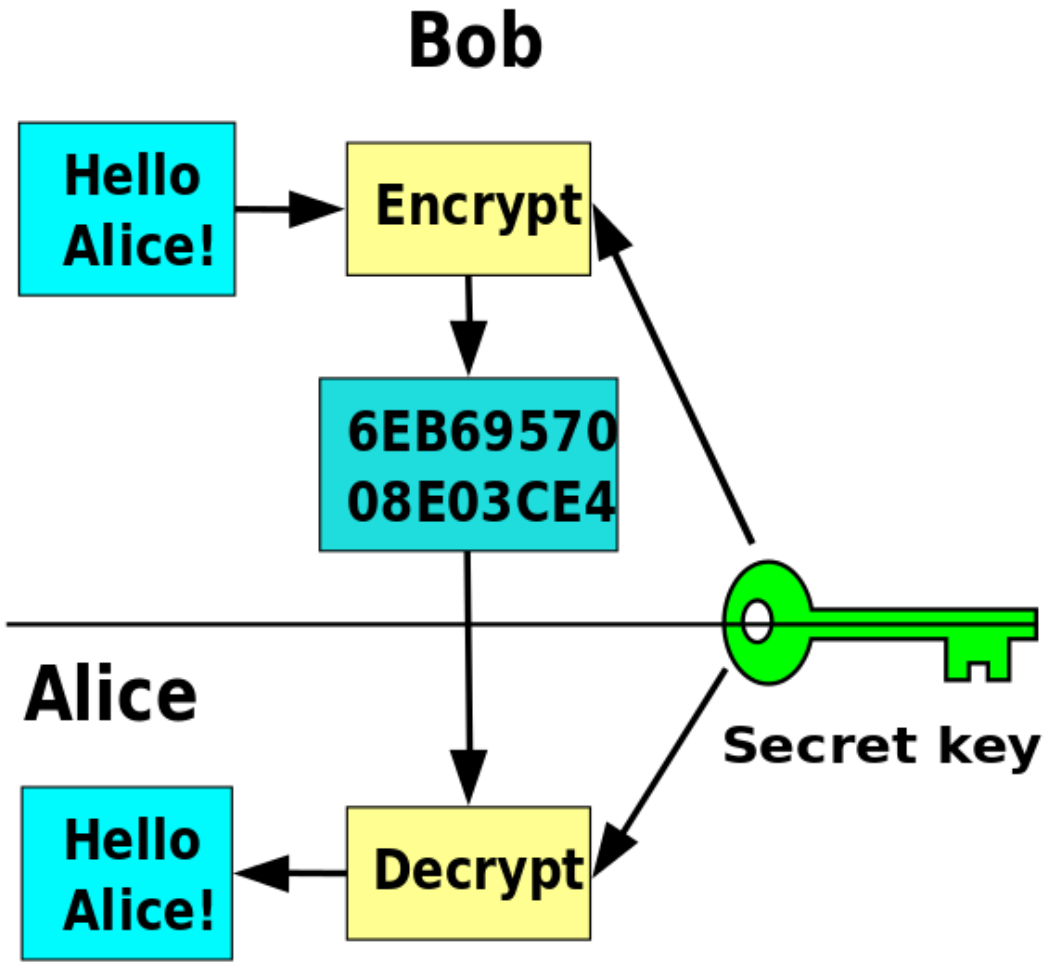
Problémafelvetés: az adatot csak egy kulcs birtokában lehessen elolvasni (visszafejteni)

Alkalmazás:

- HTTPS
- fájlrendszer titkosítása
- adatbázisok titkosítása

Szimmetrikus titkosítás

A kódolás és a visszafejtés ugyanazzal a kulccsal történik.



Kép forrása: Wikipedia

Szimmetrikus titkosítás

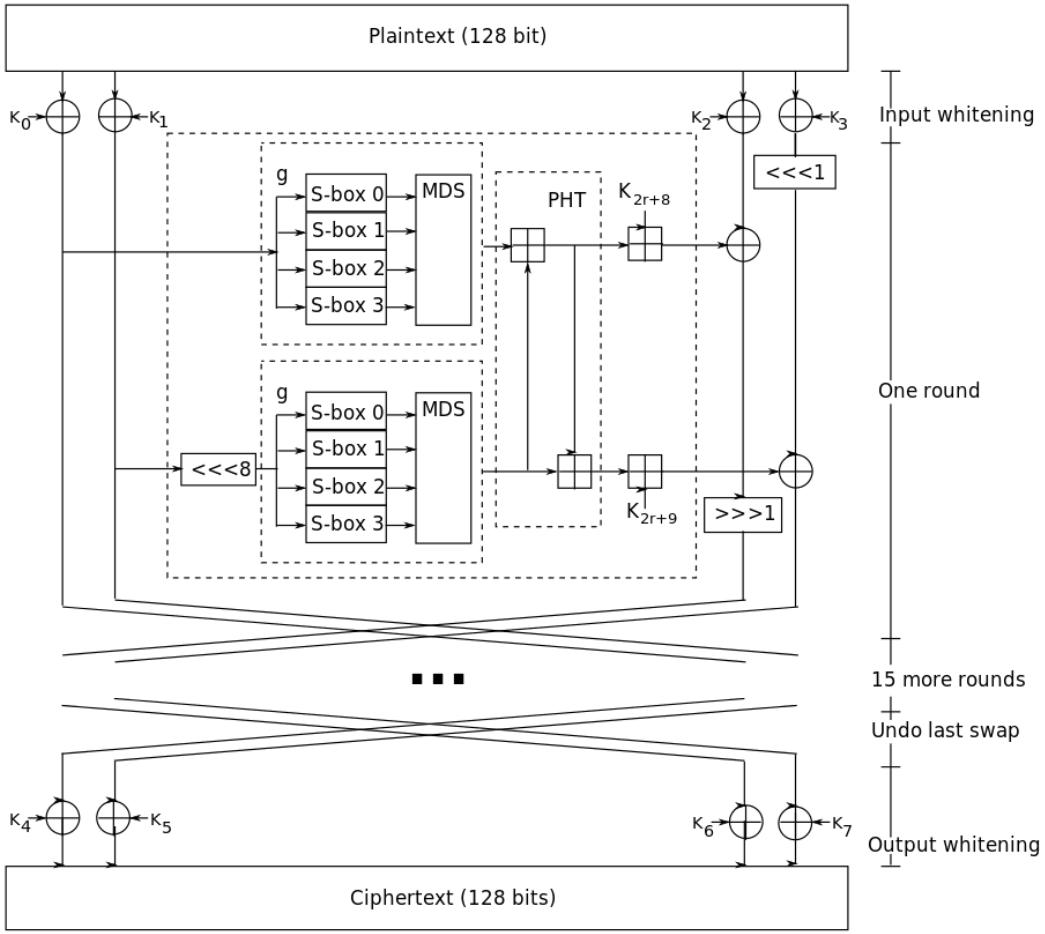
A kódolás és a visszafejtés ugyanazzal a kulccsal történik.

Példák:

- rotation cypher
  `https://www.xarg.org/tools/caesar-cipher/`
- one-time pad
- AES
  `https://aesencryption.net/`
- `gpg --symmetric -o cypher.gpg plain`

Szimmetrikus titkosítás

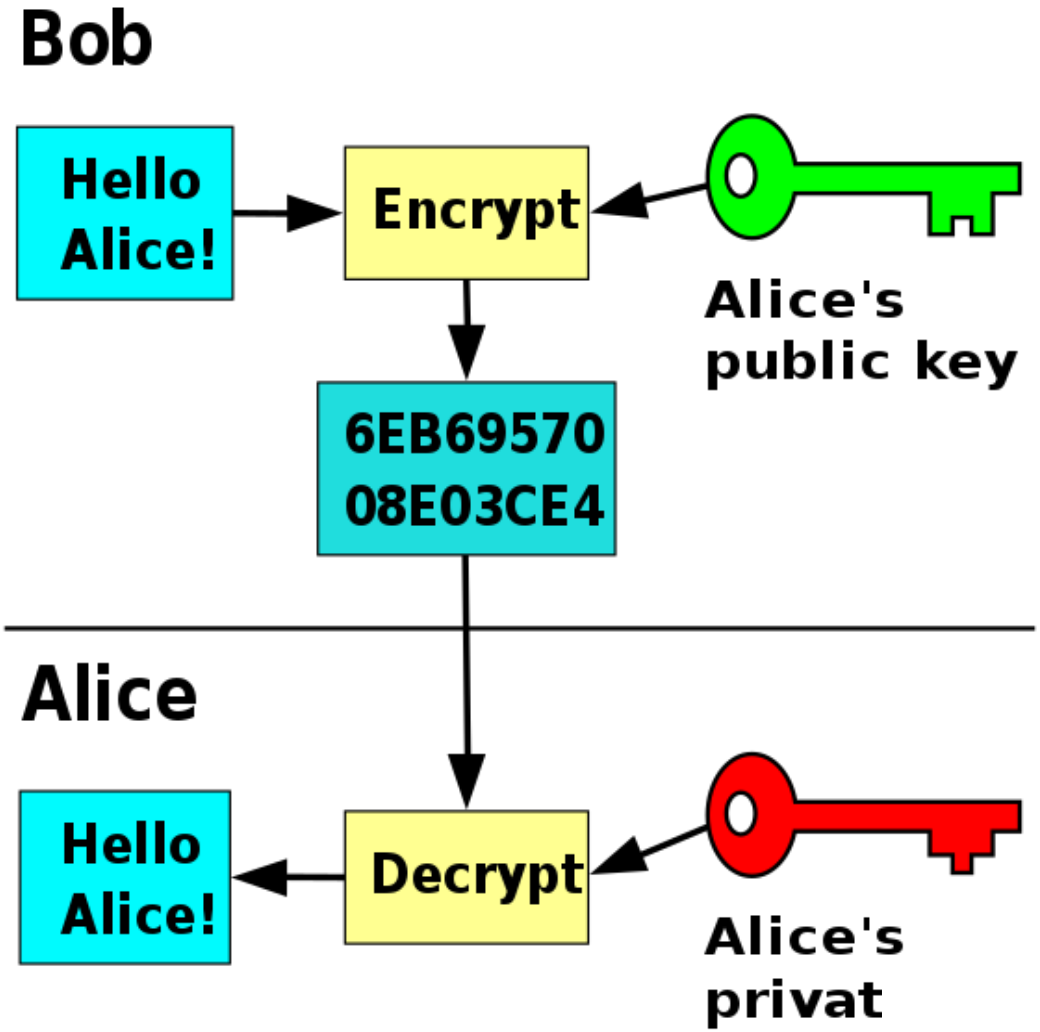
A CAST128 algoritmus egy részlete



Kép forrása: Wikipedia

Asszimmetrikus titkosítás

Hogyan juttassuk el a kulcsot a címzettnek?



Kép forrása: Wikipedia

Asszimmetrikus titkosítás

Hogyan juttassuk el a kulcsot a címzettnek?

Trükk: kulcspár

- Az egyikkel kódolni, a másikkal visszafejteni lehet.
  - Nem lehet egy kódolt üzenetet a kódoláshoz használt kulccsal visszafejteni.
  - Egyik kulcsból sem lehet a másikat kiszámolni.
- Az elsőt mindenki publikussá teszi, a másodikat megtartja magának (privát kulcs).
- A címzett publikus kulcsával titkosítom az üzenetet
- A nekem címzetteket csak én tudom elolvasni, mert csak nekem van meg a privát kulcs.

Rivest-Shamir-Adleman

Ilyen például az RSA algoritmus,
amelynek matematikai háttere a prímfaktorizáció nehézsége.
(Egy adott számról kellene megmondani, hogy melyik két prím szorzata.
Összeszorozni viszont könnyű a két prímet.)

`http://travistidwell.com/jsencrypt/demo/`

Olyan kapcsolat esetén, amikor mindkét fél jelen van egyszerre, általában a kommunikáció elején megbeszélnek egy közös kulcsot, és szimmetrikus titkosítással folytatják (gyorsabb).

Rivest-Shamir-Adleman

Demo

```
# Toltsd le a pelda kulcsot
wget https://users.itk.ppke.hu/~sulan/bevtech/kripto/test.private.asc
gpg --import test.private.asc

# Titkosits
gpg --recipient szami.tastec.hnika@pelda.domain.ppke.hu \
    --output cypher.gpg \
    --encrypt plain.txt

# Vizsgald meg a fajl tipusat
file cypher.gpg

# Fejtsd vissza
gpg --decrypt cypher.gpg
```

Digitális aláírás (Digital signature)

Mire jó még a publikus-privát kulcs?

Digitális aláírás (Digital signature)

Mire jó még a publikus-privát kulcs?

Ha az üzenet küldöje a privát kulcsával ír alá
akkor bárki el tudja olvasni a publikus kulcs birtokában
összevetve az eredetivel érvényesíthető az aláírás.

Tehát a digitális aláírás két dolgot igazol:

- a küldőt
- és a dokumentumot.

(Vesd össze a papír alapú aláírással.)
Az RSA-n kívűl egyéb aláíró algoritmusok pl.: DSA, Ed25519,
Merkle-fa alapú aláírás.

Digitális aláírás (Digital signature)

Demo

```
# Kell hozza egy importalt kulcs
# Alairas
gpg --local-user szami.tastec.hnika@pelda.domain.ppke.hu \
    --output plain.asc \
    --clear-sign plain.txt
# vagy kulon fajlba
gpg --output plain.sig \
    --detach-sign plain.txt

# Titkositas es alairas
gpg --recipient szami.tastec.hnika@pelda.domain.ppke.hu \
    --sign \
    --output cypher.gpg \
    --encrypt plain.txt
```

Hash függvény

Mi van akkor, ha az üzenetet titkosítani és aláírni is szeretnénk?

Megoldás: hash függvények

Ezek olyan függvények, amelyek egy (tipikusan tömb típusú) adatból egy rövid számot generálnak; viszonylag rövid idő alatt.Ez jellemző az adatra:
ha az adat egy resze megváltozik, a számnak is meg kell.Hívják még ellenőrző összegnek is.

Pl.:

$$\left(\sum_{i=0}^{N-1} i \oplus DATA[i]\right) \bmod \left(10^9 + 7\right)$$

Kriptografikus hash

A kriprográfiában használatos hash függvényektől elvárják, hogy egyirányú legyen:

- az eredményből nem lehet az üzenetet visszakövetkeztetni
- nem lehet két olyan üzenetet találni, amelyeknek ugyanaz a hash-e

Digitális aláírás során a küldő csak az üzenet lenyomatát írja alá.

Példák: SHA-2, SHA-3

```
sha512sum plain.txt
```

Tanúsítványok (certificates)

Honnan tudjuk, hogy egy publikus kulcs valóban a címzetté?

Public Key Infrastructure:
egy hiteles személy aláírta a publikus kulcsot
ezt hívjuk tanúsítványnak
Két módszer:

- Web of Trust:
  Néhány embert és kulcsát személyesen ellenőrzök (key-signing parties)
  Megbízom azokban is, akiket ők aláírnak
- Tanúsítvány-hatóság:
  Megbízom a néhány cég által kiadott root-tanúsítványokban
  Azok más cégeket tanúsítanak
  Azok pedig az adott publikus kulcsot
  És ezt az egész láncot mellékelik a kulcs mellé

Tanúsítványok (certificates)

Responsible Behavior

Kép forrása: XKCD

Hogyan számoljuk a véletlent?

> *"Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin."*
>
> *John von Neumann*

Számolni csak álvéletlent lehet:
Pseudorandom number generator (PRNG)
Algoritmus, amely olyan számokat (bitsorozatot) állít elő, amelyről:

- statisztikailag nem lehet megállapítani, hogy nem véletlen;
- nem lehet az előző vagy következő elemeket kitalálni;
- és nem lehet a belső állapotából következtetni az addigi kimenetekre.

Példa ilyen statisztikai tesztre: DieHarder
`https://webhome.phy.duke.edu/~rgb/General/rand_rate.php`

Linear congruential generator

Az egyik legegyszerűbb álvéletlen algoritmus.

Nem használható kriptográfiára, „nem eléggé véletlen”.

Egy belső állapotot tart fenn:

$$X_{t+1} = (aX_n + c) \mod m$$

valamilyen a, c, m konstansokkal.
A glibc 2.26 verziójában például a `rand()` függvény a következőket használja:

$$a = 1\,103\,515\,245, \quad c = 12\,345, \quad m = 2^{31},$$

a kimenet pedig X_t.

Valódi véletlen előállítása

Hardware generátorok, amelyek különböző, nehezen predikálható fizikai jelenségek mérését használják fel.
Pl.: hőmérsékleti zaj, fotoelektrikus hatások, kvantum jelenségek

Ezeket át kell alakítani, hogy az eloszlás egyenletes legyen.

Linuxon a `/dev/random` fájl olvasásával ilyen véletlen szekvenciát olvashatunk.

Alkalmazások

- Transport Layer Security (TLS)
  Internetes kommunikációban (lásd HTTPS) titkosít és azonosítja a szervert
- Pretty Good Privacy (PGP)
  Emailek és fájlok titkosítására és verifikációjára szolgál
  A tanúsítványok nem hierarchiát alkotnak, hanem egy hálózatot
  Alternatív implementáció: GNU Privacy Guard
- Secure Shell (SSH)
  Távoli bejelentkezésre használják
  A szervert (publikus kulcs) és a klienst (jelszó vagy publikus kulcs) is ellenőrzi

További irodalom

- Adatbiztonság és kriptográfia tárgy
  `https://wiki.itk.ppke.hu/twiki/pub/PPKE/Adatbiztons%c3%a1g%c3%89sKriptogr%c3%a1fia`
- `https://stribika.github.io/2015/01/04/secure-secure-shell.html`
- `https://cr.yp.to/`