

Hálózatok építése, konfigurálása és működtetése

Biztonság alapok



Incidensek a közelmúltban

Hálózatos incidensek a közelmúltban

- ▶ 2019. augusztus: Cybercrime az űrben
- ▶ 2019. március-április: Több száz millió Facebook, Instagram felhasználó érintett a hibás jelszó tárolásban. 540 millió Facebook felhasználó adat szivárogt ki
- ▶ 2018. április és november: BGP támadások
- ▶ 2018. Memcached DDoS. 17 Tbps. 15 byte kérés, 750 KB válasz
- ▶ 2018. január: Spectre és Meltdown
- ▶ 2017. május: KRACKs: WPA2 sebezhetőség a kulcscsere protokollban
 - ▶ Linux (Android) rendszerek könnyű sebezhetősége
- ▶ 2017. február: A Freedom Hosting II darknet web szerver feltörése (A teljes darknet 20%-át adja)
 - ▶ 381.000 anonim felhasználó adatainak publikálása
- ▶ 2016. október: DDOS támadás DNS szolgáltató ellen
 - ▶ 20.000 IoT eszköz kompromitálása (1 Tbps forgalom!)
- ▶ 2016. szeptember: fél millió YAHOO felhasználó adatait lopták el
 - ▶ A YAHOO valószínűleg már augusztus óta tudta ezt
- ▶ 2014. április: Heartbleed bug
 - ▶ OPENSSL implementációs hiba miatt a szerverek memóriájában lehet olvasgatni (jelszavak, kulcsok...)

Kriptográfiai alapok

Gyorstalpaló

Titkosítás

▶ Szimmetrikus titkosítás

- ▶ A kulcs megegyezik a titkosítás és a feloldás esetében
- ▶ Ma biztonságosnak tekintjük a 100 bitnél nagyobb kulcsokat
- ▶ A jó algoritmus nyílt, sokan megvizsgálták már
 - ▶ Security by obscurity vs Security by design

▶ Blokk titkosító

- ▶ Blokknyi információ titkosítása (általában 64 bit / 128 bit)
 - ▶ A blokkokat egymás után kell fűzni. Többféle mód is ismert. Jobb esetben párhuzamosítható és véletlen elérésű. Sokszor szintén szükséges a blokkok feltöltése is
- ▶ Általában több egyszerű művelet kombinációja
- ▶ A ma legjobbnak tartott blokktitkosító: AES - Advanced Encryption Standard
- ▶ Népszerű és jó titkosítók még: BlowFish, 3DES

Titkosítás 2.

- ▶ **Folyam titkosító**
 - ▶ Bitenként titkosítás, leggyakrabban XOR művelet egy hosszú álvéletlen bitsorozattal
 - ▶ A titkosító algoritmus az álvéletlen sorozatot állítja elő, egy rövid kulcsból
 - ▶ Gyorsabbak, mint a blokktitkosítók. Még keressük a legjobb folyamtitkosítót!
 - ▶ RC4, mint ismert folyamtitkosító. Kódja kb. 4 egyszerű sor
 - ▶ Chacha, Salsa (chacha20-poly1305)
- ▶ **Egyetlen bit megváltoztatása titkosított adatban:**
 - ▶ Blokktitkosító esetén a teljes felfedett blokk megváltozik
 - ▶ Folyamtitkosító esetén a felfedett adatban csak az adott bit változik meg

Titkosítás 3.

▶ Aszimmetrikus titkosítás

- ▶ A titkosítás és felfedés külön kulcsokat használ. Privát és publikus kulcsok. A kulcsokat egymáshoz gyártják.
- ▶ A működése egy olyan matematikai problémára vezethető vissza, ahol a megoldást kimerítően kell keresni, de az ellenőrzés egyszerűen és gyorsan el lehet végezni
- ▶ Legismertebb az RSA algoritmus
 - ▶ A prímtényezőzés felbontás problémájára épül. 2 nagy prímszámból jön a kulcs. Titkosítás és felfedés a hatványozás művelettel
 - ▶ Kulcsok mérete ma 2Kbit, de folyamatosan nő, a közeljövőben már 4Kbit lesz
- ▶ Jobb megoldást ad az ECC (elliptikus görbéken alapuló)
 - ▶ Kisebb kulcsok, gyorsabb műveletek
- ▶ A lassúsága miatt kerüljük az aszimmetrikus titkosítást. Ha szükséges, akkor csak egy szimmetrikus titkosítás kulcsát titkosítjuk aszimmetrikus titkosítóval
- ▶ A titkosító/felfedő folyamat időigénye leleplezheti a titkos kulcsot. Ennek kivédésére a „blinding” technika szolgál

Titkosítás 4.

- ▶ Az RSA nagyon lassú
 - ▶ Csak a szimmetrikus kulcsot titkosítjuk vele! (128-256 bit)
- ▶ Léteznek gyorsabb megoldások, jelenleg ez irányba haladunk
 - ▶ El Gamal titkosító
 - ▶ Elliptikus görbe alapú kriptográfia

ECC modulus	AES	RSA modulus	RSA:ECC
112	56	512	5:1
161	80	1024	6:1
256	128	3072	12:1
384	192	7680	20:1
512	256	15630	30:1

RC4 titkosító kód

▶ RC4 – Ron's Code 4 – Folyamtitkosító (1987)

▶ Inicializálás

```
for i = 0 to 255:
```

```
  Si = i;
```

```
  j = 0;
```

```
for i = 0 to 255:
```

```
  j = (j + Si + Ki mod 1) mod 256
```

```
  swap Si, Sj
```

▶ Kulcsfolyam generálás

```
i = (i + 1) mod 256
```

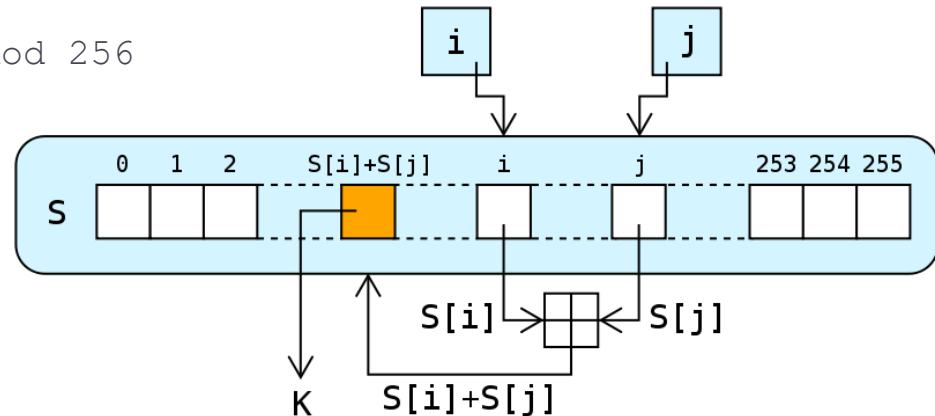
```
j = (j + Si) mod 256
```

```
swap Si, Sj
```

```
Out = S[(Si + Sj) mod 256]
```

$$C = P \text{ XOR } K$$

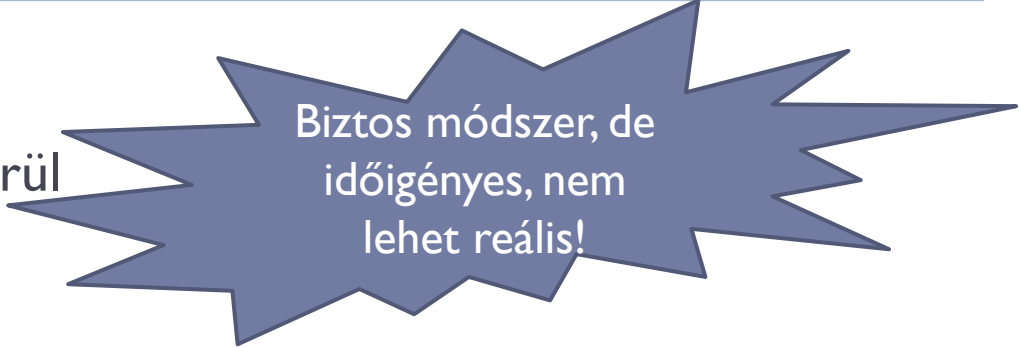
$$P = C \text{ XOR } K$$



Titkosítások felfedése

▶ Brute-force technológia

- ▶ Minden lehetséges kulcsot kipróbálunk, amíg nem sikerül a törés



Biztos módszer, de időigényes, nem lehet reális!

▶ Szótáras támadás

- ▶ A népszerű kulcsokat összegyűjtik és csak azokat teszteljük
 - ▶ Kiegészíthetik minimális átalakításokkal

▶ Védekezésképpen a kicsi kulcsteret nyilvános, de véletlen érték segítségével, nagyobbra lehet növelni

- ▶ salting

Titkosítás a gyakorlatban

- ▶ Dokumentumok, fájlok titkosítása
- ▶ Kommunikáció titkosítása
- ▶ Hitelesítés, digitális aláírás

Más nem ismerheti meg

Biztosan nem változott meg

- ▶ Sokszor a titkosítás kevés, szükséges az integritás védelme és a hitelesítés is!

Ismert az eredete

Kivonatképzés

- ▶ **Hash: Tetszőleges méretű információ leképzése fix, kis méretű kivonatra**
 - ▶ Tipikusan 128, 160, 256 (...) bit hosszúak
 - ▶ Nem használunk kulcsot
 - ▶ Iteratív működés. Egyetlen bit változásával változik minden a kimeneten
 - ▶ Fontos, hogy egyirányú, illetve gyakorlatilag nincsenek ütközések (egyező hash értékek)

- ▶ MD5 sokáig használtuk, de hiba van benne! SHA1 talán hibás és emiatt már nem kéne használni. Helyettük SHA-3 (Keccak), ami teljesen más felépítésű, mint elődei

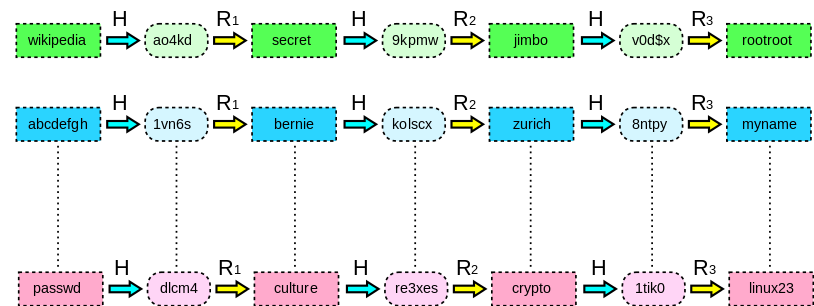
Kivonatképzés 2.

- ▶ Integritás ellenőrzésre használjuk
 - ▶ Üzenetek esetén csak akkor használható, ha a kivonat nem számolható újra a támadó által
 - ▶ Kulcsos hash, a kulcsot nem ismerhetik mások
 - ▶ Biztonságos megoldás a HMAC kivonat, ahol duplán képzünk kivonatot a biztonság érdekében
- ▶ Salting
 - ▶ A *salt* publikus, de véletlen szám
 - ▶ $H(\text{data})$ helyett $H(\text{salt}|\text{data})$. Főleg jelszavak esetén hasznos

Kivonatképzés támadása

▶ Szivárvány táblák

- ▶ Több, előre elvégzett és feljegyzett művelet segítségével a találgatás felgyorsítható. A műveletek több lépés kombinációját tartalmazzák, így nem szükséges minden eredményt eltárolni
- ▶ Hosszú hash láncokat készítenek és azok végeredményét tárolják csak
 - ▶ A láncok hurokmentesek (mert minden lépésnél más paraméter) és nem egyeznek más láncokkal
 - ▶ Ha a keresett hash érték újabb hashelések után megegyezik valamelyik lánc végével, akkor annak már megtalálható az eredete, azaz a hash előtti értéke



- ▶ TB méretű szivárvány táblák adott algoritmushoz és adott karakterkészlethez, hosszhoz
 - ▶ Kiszámított találati valószínűség

Hitelesítés

▶ Üzenet hitelesítése

- ▶ Sok esetben visszavezethető a kulcs ismeretére
 - ▶ Feltételezzük, hogy csak a másik fél ismeri a kulcsot
 - ▶ Ha helyesen használja a kulcsot, akkor a másik fél ismert
 - ▶ Ellenőrző összeg: pl. titkosított hash

▶ Küldő/másik fél hitelesítése

- ▶ Kulcsot igénylő művelet elvégzése friss feladványon
- ▶ Aszimmetrikus kulcsú kriptográfia használata

Kulcscsere

- ▶ **Kulcsok védett szétoztása a résztvevő felek között**
 - ▶ Sok esetben „csak” kulcskiosztás. Az egyik fél biztonságosan szétoztja a kulcsokat
- ▶ **Diffie-Hellman algoritmus: Kulcscsere publikus csatornán két idegen között**
 - ▶ Ha nincs hitelesítés, akkor támadható! Közbeékelődéses támadás
- ▶ **Egy megbízható harmadik fél is segíthet a kulcscserében**
 - ▶ „Ismert” kulcscsere: Kerberos

Kriptográfiai egzotikumok

▶ Titokmegosztás

- ▶ Egy titkot sokfelé oszthatok, ahol előírható, hogy hány résztitkosból áll össze a teljes titok

Kommunikáció biztonsága

