Kriptográfia és Információbiztonság
2. előadás

MÁRTON Gyöngyvér

Sapientia Egyetem, Matematika-Informatika Tanszék
Marosvásárhely, Románia
mgyongyi@ms.sapientia.ro

2023

Miről volt szó az elmúlt előadáson?

- Követelmények, osztályozás
- Bevezető, félévi áttekintő
- Könyvészet
- Történelmi háttér
- Klasszikus kriptográfiai rendszerek:
  - Eltolásos rejtjelezések: Caesar-titkosító, Keyword Caesar
  - Helyettesítéses rejtjelezés: affin-titkosító

Miről lesz szó?

- a Hill rejtjelezés
- Klasszikus kriptográfiai rendszerek
  - feltörési módszerek,
  - mátrixos rejtjelezés: Hill titkosító,
  - matematikai modell,
- titkos kulcsú rendszerek: matematikai modell
- az OTP titkosítási rendszer
- tökéletes biztonság
- biztonság-értelmezések, osztályozások

A Hill rejtjelezés

1929-ben publikálta Lester S. Hill, polialfabetikus titkosító, az első blokk titkosítók egyike. Lineáris algebrán alapszik, mátrix műveleteket végez. A következő leírás során legyen a blokkméret $d = 2$:

- az egyszerre titkosítható szimbólumok száma 2,
- $M = C = \mathbb{Z}_{26}^2$,
- *Gen*: a kulcs egy $d \times d$-es mátrix, elemei $\in \mathbb{Z}_{26}$ és fenn kell álljon, hogy $gcd(det_{key}, 26) = 1$, legyen

$$key = \begin{pmatrix} k_{1,1} & k_{1,2} \\ k_{2,1} & k_{2,2} \end{pmatrix}.$$

- Titkosítás, visszafejtés:
  - $Enc_{key}(m) = key \cdot m \rightarrow c$
  - legyen $m = (m_1, m_2)$ és $c = (c_1, c_2)$, ekkor:

$$(c_1, c_2) = \begin{pmatrix} k_{1,1} & k_{1,2} \\ k_{2,1} & k_{2,2} \end{pmatrix} \cdot \begin{pmatrix} m_1 \\ m_2 \end{pmatrix}$$

  - $Dec_{key}(c) = key^{-1} \cdot c$, ahol key^{-1} a key inverze.

A Hill rejtjelezés, kulcsgenerálás

A visszafejtéshez tehát meg kell határozni a kulcs inverzét. Ehhez szükséges kiszámolni a determinánst, a determináns multplikatív inverzét (mod 26) szerint, és az inverz mátrix értékét:

- jelöljük a determinánst: det_{key}-el,
- jelöljük a determináns inverzét: det_{key}^{-1}-el, amelyet meg lehetett határozni, mert $gcd(det_{key}, 26) = 1$
- jelöljük az adjungált mátrixot: adj_{key}, ekkor

$$adj_{key} = \begin{pmatrix} k_{2,2} & -k_{1,2} \\ -k_{2,1} & k_{1,1} \end{pmatrix}$$

- jelöljük az inverz mátrixot: key^{-1}-el, ekkor $key^{-1} = det_{key}^{-1} \cdot adj_{key}$.

Példa:

- legyen

$$key = \begin{pmatrix} 3 & 3 \\ -2 & 1 \end{pmatrix},$$

- $det_{key} = 1 \cdot 3 - (-2) \cdot 3 = 9$,
- $det_{key}^{-1} = 3$, mert $9 \cdot 3 = 1 \pmod{26}$,

$$adj_{key} = \begin{pmatrix} 1 & -3 \\ 2 & 3 \end{pmatrix}, \quad key^{-1} = det_{key}^{-1} \cdot adj_{key} = 3 \cdot \begin{pmatrix} 1 & -3 \\ 2 & 3 \end{pmatrix} = \begin{pmatrix} 3 & -9 \\ 6 & 9 \end{pmatrix}.$$

A Hill rejtjelezés, példa ha $d = 2$

Titkosítás, visszafejtés:

- ha a nyílt szöveg:

$$AM\ AT\ HE\ MA\ TI\ CI\ AN,$$

akkor a titkosított szöveg:

$$KM\ FT\ HQ\ KC\ DW\ EE\ NN,$$

- az első két betű-tömb titkosítása:

$$\begin{pmatrix} 3 & 3 \\ -2 & 1 \end{pmatrix} \cdot \begin{pmatrix} A \\ M \end{pmatrix} = \begin{pmatrix} 3 & 3 \\ -2 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 12 \end{pmatrix} = \begin{pmatrix} 10 \\ 12 \end{pmatrix} = \begin{pmatrix} K \\ M \end{pmatrix},$$

$$\begin{pmatrix} 3 & 3 \\ -2 & 1 \end{pmatrix} \cdot \begin{pmatrix} A \\ T \end{pmatrix} = \begin{pmatrix} 3 & 3 \\ -2 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 19 \end{pmatrix} = \begin{pmatrix} 5 \\ 19 \end{pmatrix} = \begin{pmatrix} F \\ T \end{pmatrix}.$$

- az első két betű-tömb visszafejtése:

$$\begin{pmatrix} 3 & -9 \\ 6 & 9 \end{pmatrix} \cdot \begin{pmatrix} K \\ M \end{pmatrix} = \begin{pmatrix} 3 & -9 \\ 6 & 9 \end{pmatrix} \cdot \begin{pmatrix} 10 \\ 12 \end{pmatrix} = \begin{pmatrix} 0 \\ 12 \end{pmatrix} = \begin{pmatrix} A \\ M \end{pmatrix},$$

$$\begin{pmatrix} 3 & -9 \\ 6 & 9 \end{pmatrix} \cdot \begin{pmatrix} F \\ T \end{pmatrix} = \begin{pmatrix} 3 & -9 \\ 6 & 9 \end{pmatrix} \cdot \begin{pmatrix} 5 \\ 19 \end{pmatrix} = \begin{pmatrix} 0 \\ 19 \end{pmatrix} = \begin{pmatrix} A \\ T \end{pmatrix}.$$

A Hill rejtjelezés, általános esetben

- $M = C = \mathbb{Z}_p^d$,
- *Gen*, kulcsgenerálás: a *key* egy $d \times d$-es mátrix, elemei $\in \mathbb{Z}_p$, jelöljük a mátrix i, j elemét $k_{i,j}$-vel, ahol fenn kell álljon, hogy $gcd(det_{key}, p) = 1$, mert a visszafejtéshez meg kell határozni key^{-1}-t
  - $key^{-1} = det_{key}^{-1} \cdot adj_{key} \pmod{p}$, ahol adj_{key} az adjungált mátrix,
- legyen $m = (m_1, m_2, \ldots m_d) \in M$ a nyílt szöveg egy d hosszúságú blokkja, melyet egy lépésben fogunk titkosítani,
- a titkosítás során meghatározzuk a nyílt szöveg egy lineáris transzformációját: $c = (c_1, c_2, \ldots c_d)$-t,

A Hill rejtjelezés, általános esetben

- $Enc_{key}(m) : c = key \cdot m$

$$(c_1, c_2, \ldots c_d) = \begin{pmatrix} k_{1,1} & k_{1,2} & \ldots & k_{1,d} \\ k_{2,1} & k_{2,2} & \ldots & k_{2,d} \\ \vdots & \vdots & & \vdots \\ k_{d,1} & k_{d,2} & \ldots & k_{d,d} \end{pmatrix} \cdot \begin{pmatrix} m_1 \\ m_2 \\ \vdots \\ m_d \end{pmatrix}$$

- $Dec_{key}(c) = key^{-1} \cdot c$
- ha $p = 256$, akkor bájtok felett végezhető a titkosítás/visszafejtés

Megjegyzések:

- ha nyílt szöveg hossza nem osztható a blokkmérettel, akkor szükséges kiegészíteni a bemenetet: paddingolási technikák,
- könyvtárcsomagok a mátrix műveletekkel kapcsolatos algoritmusokhoz:
  - C/C++-ban *NTL - Library for doing Number Theory*
  - Python: *SymPy - Library for Symbolic Matematics*

Padding-olás, a nyílt szöveg kiegészítése

Padding: a blokk titkosítók esetében ha a nyílt szöveg nem osztható a d blokk mérettel, akkor a nyílt szöveg elejét vagy végét ki kell egészíteni valamely standard eljárás szerint.

- `PKCS#7`: bájtokkal egészítjük ki a nyílt szöveg végét
  - ha `N` darab bájtot kell hozzáadni, akkor az `N` értékét adjuk hozzá `N`-szer
  - ha osztható a nyílt szöveg a blokkmérettel, akkor is kiegészítésre kerül a nyílt szöveg: egy teljes blokknyi bájtot adunk hozzá, ahol mindegyik bájt értéke d lesz

Példa:
Egy `plainTextSize` bájtból álló nyílt szöveg paddingolása:

```
N = d - plainTextSize % d;
for (i = 0; i < N; i++)
    plainText[plaintextSize + i] = N;
```

A visszafejtésnél az utolsó blokk alapján tudjuk levágni a megfelelő számú bájtokat:

```
N = cipherTextLastB[d - 1];
for (i = 0; i < d - N; i++)
    newCipherTextLastB[i] = cipherTextLastB[i];
```

Klasszikus kriptográfiai rendszerek

A következő feltörési módszerek mindegyike kivitelezhető, éppen ezért a klasszikus kriptorendszerek egyikét sem használják a gyakorlatban:

- az összes lehetséges kulcs kipróbálása (exhaustive key search): a kulcsok száma legtöbbször kicsi a mai rendszerek tár és feldolgozó kapacitáshoz képest,
- betűgyakoriság vizsgálat (ciphertext-only attack): a rejtjelezett szöveg ugyanolyan statisztikai tulajdonságokkal rendelkezik, mint a nyílt szöveg,
- ismert nyílt-szöveg támadás (known plaintext attack): ha rendelkezünk néhány betű rejtjelezett értékével, akkor meghatározható az eredeti szöveg, gyakran a kulcs is.
  - Caesar titkosító esetében elég tudni egy betűnek a rejtjelét, ahhoz, hogy a rendszerben használt kulcsot meghatározhassuk,
  - az Affin titkosító esetében elég két betűnek tudni a rejtjelezett értékét ahhoz, hogy a rendszerben használt kulcsot meghatározhassuk,
  - a Hill titkosító esetében, ha a blokkméret d, akkor elég ha ismert d darab blokk rejtjelezett értéke ahhoz hogy, meghatározzuk a rendszerben használt kulcsot.

Az Affin rejtjelezés - ismert nyílt-szöveg támadás

- tudva hogy az m_1 rejtjele c_1, és az m_2 rejtjele c_2, akkor a következő kongruencia-rendszer megoldásával, megállapítható a titkosításhoz használt (a, b) kulcs, ahol $(m_1 - m_2)^{-1}$ az $m_1 - m_2$ multiplikatív inverze:

$$\begin{aligned} m_1 \cdot a + b &= c_1 \pmod{26} \\ m_2 \cdot a + b &= c_2 \pmod{26}. \end{aligned}$$

$$(m_1 - m_2) \cdot a = (c_1 - c_2) \pmod{26}$$

$$\begin{aligned} a &= (m_1 - m_2)^{-1} \cdot (c_1 - c_2) \pmod{26} \\ b &= (c_1 - m_1 \cdot a) \pmod{26} \end{aligned}$$

vagy

$$b = (c_2 - m_2 \cdot a) \pmod{26}$$

Az Affin rejtjelező (az 1.előadásból):

- $M = C = \{0, 1, \ldots, 25\}^* = \mathbb{Z}_{\mathbf{26}}, K = \{(a, b) \in \mathbb{Z}_{\mathbf{26}}, gcd(a, 26) = 1\}$,
- $key = (a, b)$
- $Enc_{(a,b)}(m) = (a \cdot m + b) \pmod{26} \Rightarrow c$,
- $Dec_{(a,b)}(c) = a^{-1} \cdot (c + 26 - b) \pmod{26}$.

Hill módszere - ismert nyílt-szöveg támadás, d = 2

- Ha ismertek az $m, \hat{m}, c, \hat{c}$, tömb-párok értékei, ahol az m rejtjelezett értéke c és az $\hat{m}$ rejtjelezett értéke $\hat{c}$,
- akkor a következő rendszer megoldásával, megállapítható a titkosításhoz használt *key* kulcs,
- legyen $m = \begin{pmatrix} m_1 \\ m_2 \end{pmatrix}$, $c = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$, $\hat{m} = \begin{pmatrix} \hat{m}_1 \\ \hat{m}_2 \end{pmatrix}$, $\hat{c} = \begin{pmatrix} \hat{c}_1 \\ \hat{c}_2 \end{pmatrix}$,
- felírható:

$$key \cdot \begin{pmatrix} m_1 & \hat{m}_1 \\ m_2 & \hat{m}_2 \end{pmatrix} = \begin{pmatrix} c_1 & \hat{c}_1 \\ c_2 & \hat{c}_2 \end{pmatrix}$$

$$key = \begin{pmatrix} c_1 & \hat{c}_1 \\ c_2 & \hat{c}_2 \end{pmatrix} \cdot \begin{pmatrix} m_1 & \hat{m}_1 \\ m_2 & \hat{m}_2 \end{pmatrix}^{-1}$$

- hasonló támadási stratégia alkalmazható nagyobb blokkméret esetében.

A klasszikus titkosítás matematikai modellje

Klasszikus titkosítók, egyéb megnevezések: szimmetrikus kriptográfia (classical cryptography). Három algoritmust szükséges értelmezni, ahol K a kulcsok, M az üzenetek halmaza:

- Gen, a kulcs-generáló algoritmus, meghatározza a key kulcsot,
- $Enc(key, \cdot)$ a rejtjelező algoritmus, a key kulcs alapján, meghatározza az $m \in M$ nyílt-szöveg rejtjelezett értékét:

$$c \leftarrow Enc(key, m),$$

- $Dec(key, \cdot)$ a visszafejtő algoritmus, a key kulcs alapján visszafejti a c rejtjelezett-szöveget:

$$m \leftarrow Dec(key, c).$$

A helyesség fennáll, ha minden $m \in M$ esetében:

$$Dec(key, Enc(key, m)) = m$$

Számos klasszikus titkosítási rendszer létezik: Caesar, Vigenere, Palyfair, Hill, stb.

Modern kriptográfia - területek, felosztása

Titkos kulcsú kriptográfia: a tulajdonképpeni adatfolyam titkosítása

- folyamtitkosítók
- blokktitkosítók

Publikus kulcsú kriptográfia: az adatfolyam titkosításánál alkalmazott kulcs megosztása, a kulcsmegosztáshoz használt publikus kulcs/publikus információ hitelesítése

- publikus kulcsú titkosítók
- kulcscserék
- digitális aláírások

Egyéb kriptográfia primitívek:

- pszedudo-random és random számgenerátorok
- hash függvények
- üzenethitelesítő kódok (MAC - Message Authentication Code)

A titkos kulcsú rendszerek matematikai modellje

Titkos kulcsú titkosítók, egyéb megnevezések: szimmetrikus kriptográfia (secret-key encryption, symmetric cryptography). Három algoritmust szükséges értelmezni, ahol K a kulcsok, M az üzenetek halmaza:

- Gen, a kulcs-generáló algoritmus, polinom idejű, véletlenszerű:

$$key \xleftarrow{R} Gen(1^k),$$

  ahol $key \in K$ és k a rendszer biztonsági paramétere, legtöbb esetben a generált kulcs bit-hossza, és fennáll: $k \in \mathbb{Z}_{\geq 0}$

- $Enc(key, \cdot)$ a rejtjelező algoritmus, polinom idejű, véletlenszerű:

$$c \xleftarrow{R} Enc(key, m),$$

- a $Dec(key, \cdot)$ a visszafejtő algoritmus, polinom idejű, determinisztikus:

$$m \leftarrow Dec(key, c),$$

A helyesség fennáll, ha minden $m \in M$ esetében:

$$Dec(key, (Enc(key, m)) = m.$$

A publikus kulcsú titkosítók matematikai modellje

Publikus kulcsú titkosítók, egyéb megnevezések: nyilvános kulcsú titkosítók, aszimmetrikus titkosítók (public-key encryption, asymmetric cryptography). Három algoritmust szükséges értelmezni, ahol K a kulcsok, és $M = f(K)$ az üzenetek halmaza.

- *Gen*, a kulcs-generáló algoritmus, polinom idejű, véletlenszerű:

$$(pk, sk) \xleftarrow{R} Gen(1^k),$$

  ahol $(pk, sk) \in K$ és k a rendszer biztonsági paramétere, legtöbb esetben a generált kulcs bit-hossza, és fennáll: $k \in \mathbb{Z}_{\geq 0}$

- $Enc(pk, \cdot)$ a rejtjelező algoritmus, polinom idejű, véletlenszerű:

$$c \xleftarrow{R} Enc(pk, m),$$

- a $Dec(sk, \cdot)$ a visszafejtő algoritmus, polinom idejű, determinisztikus:

$$m \leftarrow Dec(sk, c),$$

A helyesség fennáll, ha minden $m \in M$ esetében:

$$Dec(sk, Enc(pk, m)) = m$$

A digitális aláírás matematikai modellje

Digitális aláírások, egyéb megnevezések: publikus kulcsú tanúsítványok (digital signature, public key certificates). Három algoritmust szükséges értelmezni, ahol K a kulcsok, és $M = f(K)$ az üzenetek halmaza.

- Gen, a kulcs-generáló algoritmus, polinom idejű, lehet véletlenszerű is:

$$(pk, sk) \xleftarrow{R} Gen(1^k),$$

  ahol $(pk, sk) \in K$ és k a rendszer biztonsági paramétere, legtöbb esetben a generált kulcs bit-hossza, és fennáll: $k \in \mathbb{Z}_{\geq 0}$

- $Aut(sk, \cdot)$ a hitelesítő algoritmus, polinom idejű, véletlenszerű:

$$(m, c) \xleftarrow{R} Aut(sk, m),$$

- a $Ver(pk, \cdot)$ az ellenőrző algoritmus, polinom idejű, determinisztikus, amelynek `True` vagy `False` a kimenete aszerint, hogy $m = \hat{m}_1$-el vagy sem:

$$\hat{m} \leftarrow Ver(pk, c).$$

A helyesség fennáll, ha minden $m \in M$ esetében:

$$Ver(pk, Aut(sk, m)) = m.$$

A kulcscsere mechanizmusok matematikai modellje

Kulcscsere mechanizmusok (key exchange mechanism): ez egy interaktív protokoll, amikor a két kommunikáló fél egyidőben kell online legyen. Feltételezzük, hogy a két kommunikáló fél **A** és **B**. Három algoritmust szükséges értelmezni, ahol K a kulcsok, és $M = f(K)$ az üzenetek halmaza.

- *Gen*, a publikus információt/kulcsot generáló algoritmus, polinom idejű, véletlenszerű:

$$pk \xleftarrow{R} Gen(1^k),$$

  $pk \in K$, ahol k a rendszer biztonsági paramétere, a generált kulcs bithossza
- a $PGen_{\mathbf{X}}(pk, \cdot)$, $\mathbf{X} \in \{\mathbf{A}, \mathbf{B}\}$ algoritmus polinom idejű, véletlenszerű:
  - ezt mindkét fél végrehajtja, a kapott A, B értékeket megosztják egymással, az a, b értékek azonban szigorúan titkosak maradnak:

$$A \xleftarrow{R} PGen_{\mathbf{A}}(pk, a), \quad a \xleftarrow{R} M$$
$$B \xleftarrow{R} PGen_{\mathbf{B}}(pk, b), \quad b \xleftarrow{R} M$$

- a $KGen_{\mathbf{X}}(x, \cdot)$, $\mathbf{X} \in \{\mathbf{A}, \mathbf{B}\}$, $x \in \{a, b\}$ algoritmus polinom idejű, determinisztikus, és amelyet mindkét fél végre kell hajtson:

$$key \leftarrow KGen_{\mathbf{A}}(a, B), \quad key \leftarrow KGen_{\mathbf{B}}(b, A)$$

A helyesség fennáll, ha minden $a, b \in M$-re:

$$KGen_{\mathbf{A}}(a, B) = KGen_{\mathbf{A}}(a, PGen_{\mathbf{B}}(pk, b)) = KGen_{\mathbf{B}}(b, PGen_{\mathbf{A}}(pk, a)) = KGen_{\mathbf{B}}(b, A).$$

A titkos kulcsú rendszerek matematikai modellje

A gyakorlatban

- a kulcsok, az üzenetek a titkosított szövegek bájt illetve bit szekvenciák
- a kulcs mérete legalább 128 bit (16 bájt), az üzenetek, pedig lehetnek 1GB (gigabájtnyi) videó file-ok, 10 MB zene file-ok, 1KB email adat, vagy egyetlen bitnyi adat ami megfelel pl. egy szavazási rendszerben az `igen` vagy `nem` szavazati értéknek.
- a polinom futásidejű algoritmus hatékony algoritmust jelent, például elvárjuk, hogy 1 GB adat titkosítását 1 perc alatt végezze el a rendszer
- a kulcsok, az üzenetek, a titkosított adatok különböző típusú matematikai objektumok is lehetnek, pl. értékpárok, mátrixok, polinomok, görbék pontjai, stb. Minden objektum esetben megadható kell legyen az, hogy hogyan ábrázoljuk, alakítjuk át ezeket bit, illetve bájt szekvenciákká.

Tökéletes biztonság

Shannon biztonság-elmélete, 1949 $\Rightarrow$ információelméleti biztonság, amely szerint egy *Gen*, *Enc*, *Dec* algoritmusok által értelmezett *SKE* titkosító rendszerről kijelenthető:

1. értelmezés

Az SKE, amelynek biztonsági paramétere k, tökéletesen biztonságos (perfectly secret), ha egy véletlenszerűen generált $key \in K$, bármely $m_0, m_1 \in M$, és bármely $c \in C$ esetében fennáll:

$$Pr[Enc_{key}(m_0) = c] = Pr[Enc_{key}(m_1) = c].$$

1. tétel

Az SKE tökéletesen biztonságos akkor és csakis akkor, ha M bármely valószínűség eloszlása esetében bármely $m \in M$, és bármely $c \in C$ esetében, ahol $Pr[C = c] > 0$ az m valószínűsége ugyanaz mint az m valószínűsége feltételezve, hogy a c következett be, azaz fennáll:

$$Pr[M = m] = Pr[M = m|C = c].$$

Tökéletes biztonság

Megjegyzések:

- intuitive: egy adott rejtjelezett szöveg egyforma valószínűséggel lehet bármelyik nyílt szöveg rejtjelezett értéke
- a kulcs ismeretének hiányában nem lehet eldönteni, hogy melyik nyílt szöveg került titkosításra
- a nyílt szöveg hossza rögzített: a rejtjelezés nem titkosítja a nyílt-szöveg hosszát

2. tétel

Ha egy titkosító rendszer tökéletes biztonságú, akkor $|K| \geq |M|$.

Tehát a gyakorlatban szinte lehetetlen olyan titkosítót létrehozni, amely tökéletes biztonságú, ezért más biztonság-értelmezésre van szükség: számítástechnikai biztonság.

Számítástechnikai biztonság

2. értelmezés

Egy Gen, Enc, Dec algoritmusok által értelmezett titkosító rendszer, amelynek biztonsági paramétere k számítástechnikai biztonságú (computational secure), ha bármely polinom idejű, véletlenszerű algoritmus csak nagyon kis/elhanyagolható valószínűséggel tudja feltörni a rendszert.

A számítástechnikai biztonság, feladja a tökéletes biztonságot, feltételezi, hogy a támadó számítás-kapacitása (idő, tár) korlátos, és hogy a támadás csak nagyon kicsi valószínűséggel lehet sikeres.

Számítástechnikai biztonság

Példa: ha egy támadó egy titkosító algoritmus esetében az összes kulcs kipróbálásának módszerét választja, ahol a kulcsméret k, azaz a kulcsok száma 2^k, akkor a következő állapítható meg:

- ha egy kulcs teszteléséhez c időegységre van szükséges, összesen pedig K időegység áll a támadás kivitelezésére, akkor $\frac{K}{c}$ darab kulcs vizsgálható meg,
- ha $\frac{K}{c} \ll 2^k$, akkor a kulcs meghatározásának valószínűsége $\frac{K}{c2^k}$,
- ha $c = 1$ és 2GHz CPU gépünk van, amely 1 évig számol, akkor
$$K = 365 \cdot 24 \cdot 60 \cdot 60 \cdot 2 \cdot 10^9 = 63072000000000000 < 2^{56},$$
ami azt jelenti, hogy kb. 2^{55} darab kulcs vizsgálható meg,
- ha a kulcsméret 128 bit, akkor a kulcs meghatározásának valószínűsége: $\frac{2^{55}}{2^{128}} = 2^{-73}$!!!

Az OTP rendszer

- OTP - One-time Pad rendszer, 1917, Vernam rendszerének egy változata,
- $M = C = K = \mathbb{Z}_2^k$,
- legyen $m = (m_1, \ldots, m_k) \in M$, $c = (c_1, \ldots, c_k) \in C$,
  $key = (key_1, \ldots, key_k) \in K$,
- $Gen \xrightarrow{R} key$, (véletlenszerűen, egyenletes eloszlással)
- $Enc_{key}(m) : c = (m_1 \oplus key_1, \ldots m_k \oplus key_k)$,
- $Dec_{key}(c) : m = (c_1 \oplus key_1, \ldots c_k \oplus key_k)$.
- az $\oplus$ az XOR műveletet jelöli, ami tulajdonképpen mod 2 szerinti összeadás,
- az $\oplus$ művelet tulajdonságai:

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

$x \oplus y = y \oplus x$,
$x \oplus (y \oplus z) = (x \oplus y) \oplus z$,
$x \oplus 0 = x, x \oplus x = 0$.

- a fenti tulajdonságok alapján a titkosítási folyamat helyessége egyértelmű:
  $Dec_{key}(Enc_{key}(m)) = Dec_{key}(m \oplus key) = (m \oplus key) \oplus key = m$.

Az OTP rendszer

Példa, titkosításra, ahol a kulcs és a nyílt szöveg is 7 bites:

$$\begin{array}{lcccccccc} m & = & (1 & 1 & 0 & 0 & 1 & 0 & 0) & \oplus \\ key & = & (0 & 0 & 1 & 0 & 1 & 0 & 1) & \\ \hline c & = & (1 & 1 & 1 & 0 & 0 & 0 & 1) & \end{array}$$

Példa, visszafejtésre, ahol a kulcs és a rejtjelezett szöveg is 7 bites:

$$\begin{array}{lcccccccc} c & = & (1 & 1 & 1 & 0 & 0 & 0 & 1) & \oplus \\ key & = & (0 & 0 & 1 & 0 & 1 & 0 & 1) & \\ \hline m & = & (1 & 1 & 0 & 0 & 1 & 0 & 0) & \end{array}$$

Az OTP rendszer, feltörési lehetőségek

A kulcs többszöri felhasználásának problémája:

$$
\begin{array}{lcccccccc}
m_1 & = & (1 & 1 & 0 & 0 & 1 & 0 & 0) & \oplus \\
key & = & (0 & 0 & 1 & 0 & 1 & 0 & 1) & \\
\hline
c_1 & = & (1 & 1 & 1 & 0 & 0 & 0 & 1) &
\end{array}
$$

$$
\begin{array}{lcccccccc}
m_2 & = & (1 & 0 & 0 & 1 & 1 & 1 & 0) & \oplus \\
key & = & (0 & 0 & 1 & 0 & 1 & 0 & 1) & \\
\hline
c_2 & = & (1 & 0 & 1 & 1 & 0 & 1 & 1) &
\end{array}
$$

Known plaintext attack: m_2, c_2 ismeretében meg lehet határozni a *key* értékét, majd a *key* és a c_1 alapján meg lehet határozni m_1-t:

$$
\begin{array}{lcccccccc}
m_2 & = & (1 & 0 & 0 & 1 & 1 & 1 & 0) & \oplus \\
c_2 & = & (1 & 0 & 1 & 1 & 0 & 1 & 1) & \\
\hline
key & = & (0 & 0 & 1 & 0 & 1 & 0 & 1) &
\end{array}
$$

$$
\begin{array}{lcccccccc}
key & = & (0 & 0 & 1 & 0 & 1 & 0 & 1) & \oplus \\
c_1 & = & (1 & 1 & 1 & 0 & 0 & 0 & 1) & \\
\hline
m_1 & = & (1 & 1 & 0 & 0 & 1 & 0 & 0) &
\end{array}
$$

Az OTP rendszer, feltörési lehetőségek

A kulcs többszöri felhasználásának problémája:

$$
\begin{array}{lcccccccc}
m_1 & = & (1 & 1 & 0 & 0 & 1 & 0 & 0) & \oplus \\
key & = & (0 & 0 & 1 & 0 & 1 & 0 & 1) & \\
\hline
c_1 & = & (1 & 1 & 1 & 0 & 0 & 0 & 1) &
\end{array}
$$

$$
\begin{array}{lcccccccc}
m_2 & = & (1 & 0 & 0 & 1 & 1 & 1 & 0) & \oplus \\
key & = & (0 & 0 & 1 & 0 & 1 & 0 & 1) & \\
\hline
c_2 & = & (1 & 0 & 1 & 1 & 0 & 1 & 1) &
\end{array}
$$

c_1, c_2 ismeretében meg lehet határozni $m_1 \oplus m_2$ értékét:

$$
\begin{array}{lcccccccc}
c_1 & = & (1 & 1 & 1 & 0 & 0 & 0 & 1) & \oplus \\
c_2 & = & (1 & 0 & 1 & 1 & 0 & 1 & 1) & \\
\hline
m_1 \oplus m_2 & = & (0 & 1 & 0 & 1 & 0 & 1 & 0) &
\end{array}
$$

Az OTP rendszer

3. tétel

Az OTP titkosító rendszer tökéletes biztonságú.

Megjegyzések:

- nagyon gyors a titkosítás és visszafejtés folyamata,
- nem lehet észrevenni, ha módosult a rejtjelezett-szöveg, ezt üzenet-hitelesítő kódok alkalmazásával lehet megoldani,
- a gyakorlatban felmerülő problémák:
  - a generált kulcsot tilos többször felhasználni: ha ismert egy nyílt-szöveg és titkosított szöveg pár, akkor meg lehet határozni a kulcsot,
  - minden egyes titkosításhoz más kulcsot kell használni, amely független a korábbi kulcsoktól,
  - a generált kulcs ugyanolyan hosszú kell legyen, mint a nyílt szöveg,
- gyakorlatban egy véletlenszerűen előállított rövid (pár bájtos) kulcs alapján véletlenszerűen generálnak tetszőleges hosszúságú kulcsfolyamnak (key stream) nevezett bájt/bit szekvenciát

Random számok

- az adatbiztonságban számos helyen van szükség véletlenszerűen előállított bájtokra/bitekre (random bits/random numbers),
- különbség van a valódi (true) és az ál (pseudo) véletlen számok között,
- valódi random biteket manuálisan, illetve hardver eszközök segítségével lehet generálni, viszonylag lassúak, és megbízhatóság szempontjából számos kritika éri őket,
- a pseudorandom biteket/számokat determinisztikus algoritmusokkal generálják, amelyeknek bemenetként meg kell adni egy kezdeti, rövid *seed*-nek nevezett random értéket, és amelyek eredményként random biteknek/számoknak tűnő hosszú szekvenciát adnak,
- a megfelelő biztonság eléréséhez szükséges megadni a random bit generáló algoritmusok matematikai definícióit.

Random számok

3. értelmezés

Egy valódi random bit generátor egy olyan eljárás/egység, amely egy random bit szekvenciát generál, ahol a megfelelő $X_1, X_2, \ldots$ bináris valószínűségi változók szekvenciája a következő tulajdonsággal rendelkezik:

- $Pr[X_n = 0] = Pr[X_n = 1] = \frac{1}{2}$, *minden* $n \in \mathbb{N}$ *értékre*
- $X_1, X_2, \ldots, X_n$ *egymástól független valószínűségi változók, minden* $n \in \mathbb{N}$*-re*

4. értelmezés

Egy G pseudorandom bit generátor egy olyan polinom idejű, determinisztikus algoritmus, amely

- *egy kezdeti* $s \in \{0,1\}^n$*, seed alapján egy* $G(s) \in \{0,1\}^{l(n)}$ *kimeneti bitszekvenciát állít elő, ahol* $l(\cdot)$ *egy polinom, és* $l(n) > n$ *bármely* $n \in \mathbb{N}$*-re*
- *esetében polinom időben nem állapítható meg különbség a kimeneti bitszekvencia és egy egyenletes eloszlású véletlenszerűen generált bitszekvencia között.*

Random számok

- egy pseudorandom generátor által előállított bitszekvencia soha nem lehet egyenletes eloszlású, mert $l(n) > n$, számos olyan $l(n)$ hosszúságú bitszekvencia létezik, amelyek nem szerepelnek a G bemenetében
- egy pseudorandom generátor szerkesztése nem egy triviális feladat
- *next-bit test*: egy támadó ismerve egy generátor első i darab kimeneti értékét, 50%-nál nagyobb valószínűséggel nem határozhatja meg polinom időben az $(i+1)$-ik bit értékét
- a pseudorandom generátorok számára a NIST készített egy statisztikai *tesztcsomagot*: ha egy generátor megfelel mindegyik tesztnek az még nem bizonyítja, hogy a generátor pszeudorandom, ellenben ha egy teszten nem megy át, akkor biztosan nem pszeudorandom generátor

A titkos kulcsú rendszerek osztályozása

- nagy adathalmaz titkosítására alkalmasak,
- biztonságuk számítástechnikai szempontból elfogadható.
- nincs megoldva a felek közötti kulcscsere, ezt a publikus kulcsú kriptográfia végzi,
- nincs megoldva a felek hitelesítése, ezt a publikus kulcsú kriptográfia végzi,
- két nagy csoportra oszthatóak:
  - folyam-titkosító rendszerek,
  - blokk-titkosító rendszerek.

Folyam-titkosító rendszerek

- véletlenszerűen generálnak egy bitsort, a kulcsfolyamot, amelyet legtöbbször a $\oplus$ művelettel hozzáadnak a nyílt szöveghez,
- a kulcsfolyamot egyetlenegyszer használják,
- a rendszer biztonságát a kulcsfolyamot generáló algoritmus határozza meg $\Rightarrow$ álvéletlen-szám generáló algoritmusok (pseudo-random number generators),
- nem minden ál-véletlenszám generátor alkalmas a kriptográfiában,
- pl: az $x_n = a \cdot x_{n-1} + b \pmod{p}$ lineáris kongruenciával értelmezett generátor sem alkalmas.

Blokk-titkosító rendszerek

- bájt-tömbönként alkalmaznak egy álvéletlen függvényt/permutációt,
- a kulcsot bájt-tömbönként újra használják $\Rightarrow$ blokk titkosítási módok,
- ha nyílt szöveg hossza nem osztható a blokkmérettel, akkor szükséges kiegészíteni a bemenetet: paddingolási technikák,
- a rendszer biztonságát az alkalmazott álvéletlen függvény (pseudo-random function), illetve álvéletlen permutáció (pseudo-random permutation) határozza meg,
- egy blokktitkosító rendszer átalakítható folyamtitkosító rendszerré, ha valamilyen blokk-titkosító módot használunk, pl. CFB, CTR.