Kriptográfia és Információbiztonság
2. előadás

MÁRTON Gyöngyvér

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

2022

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?

- Klasszikus kriptográfiai rendszerek
  - Mátrixos rejtjelezés: Hill módszere
  - matematikai modell
- titkos-kulcsú rendszerek: matematikai modell
- az OTP titkosítási rendszer
- tökéletes biztonság
- biztonság-értelmezések, osztályozások

Hill módszere, példa, titkosítás

- 1929-ben publikálta Lester S. Hill,
- polialfabetikus rendszer, az első blokk titkosítók egyike,

Példa:

- Legyen $d = 2$, az egyszerre titkosítható szimbólumok száma, $M = C = \mathbb{Z}_{26}^2$, és
$$key = \begin{pmatrix} 3 & 3 \\ -2 & 1 \end{pmatrix},$$
- 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:
$$\begin{pmatrix} 3 & 3 \\ -2 & 1 \end{pmatrix} \cdot \begin{pmatrix} A \\ M \end{pmatrix} = \begin{pmatrix} 3 & 3 \\ -2 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 12 \end{pmatrix} = \begin{pmatrix} 10 \\ 12 \end{pmatrix} = \begin{pmatrix} K \\ M \end{pmatrix},$$
$$\begin{pmatrix} 3 & 3 \\ -2 & 1 \end{pmatrix} \cdot \begin{pmatrix} A \\ T \end{pmatrix} = \begin{pmatrix} 3 & 3 \\ -2 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 19 \end{pmatrix} = \begin{pmatrix} 5 \\ 19 \end{pmatrix} = \begin{pmatrix} F \\ T \end{pmatrix}.$$

Hill módszere, példa, visszafejtés

A visszafejtéshez meg kell határoznunk a $key = \begin{pmatrix} 3 & 3 \\ -2 & 1 \end{pmatrix}$ kulcs inverzét, ehhez szükséges kiszámolni a determinánst, a determináns inverzét $\pmod{26}$ szerint, és az inverz mátrix értékét:

- a determináns: $(det\ key) = 1 \cdot 3 - (-2) \cdot 3 = 9$,
- a determináns inverze: $(\det key)^{-1} = 3$, mert $9 \cdot 3 = 1 \pmod{26}$, amelyet meg lehetett határozni, mert, $gcd(9,\ 26) = 1$,
- az adjungált mátrix: $(\mathrm{adj}\ key) = \begin{pmatrix} 1 & -3 \\ 2 & 3 \end{pmatrix}$.
- az inverz mátrix:
  $key^{-1} = (\det key)^{-1} \cdot (\mathrm{adj}\ key) = 3 \cdot \begin{pmatrix} 1 & -3 \\ 2 & 3 \end{pmatrix} = \begin{pmatrix} 3 & -9 \\ 6 & 9 \end{pmatrix}$.

Az első két betű-tömb visszafejtése:

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

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

Hill módszere, általános esetben

- $M = C = \mathbb{Z}_p^d$,
- a *key* egy $d \times d$-es mátrix, melynek elemei $\in \mathbb{Z}_p$, jelöljük a mátrix i, j elemét $k_{i,j}$-vel,
- 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,
- ha $p = 256$, akkor bájtok felett végezhető titkosítás/visszafejtés
- a mátrix műveletekkel kapcsolatos algoritmusok implementációja: C/C++-ban Shoup NTL könyvtárcsomagja: *https://libntl.org/*

Hill módszere, általános esetben

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

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

- $Dec_{key}(c) = key^{-1} \cdot c$
- key^{-1} létezik, ha $(det\ key) \cdot (det\ key)^{-1} = 1 \pmod p$,
- $key^{-1} = (det\ key)^{-1} \cdot (\mathrm{adj}\ key) \pmod p$, ahol $(\mathrm{adj}\ key)$ az adjungált mátrix.
- ha $d = 2$, akkor egyszerűen meghatározható az inverz mátrix

$$key = \begin{pmatrix} k_{1,1} & k_{1,2} \\ k_{2,1} & k_{2,2} \end{pmatrix} \text{ és } key^{-1} = (det\ key)^{-1} \cdot \begin{pmatrix} k_{2,2} & -k_{1,2} \\ -k_{2,1} & k_{1,1} \end{pmatrix}$$

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 blokk méret esetében.

Az Affin rejtjelezés - ismert nyílt-szöveg támadás

Feltörési módszerek:

- gyakoriság vizsgálat
- az összes lehetséges kulcs kipróbálása, kulcsok száma: 312
- ismert nyílt-szöveg támadás (known plaintext attack): rendelkezünk két betű rejtjelezett értékével,
  - 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:

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

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

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

A klasszikus titkosítás matematikai modellje

Legyen

- M a nyílt-szövegek egy véges halmaza,
- C a rejtjelezett-szövegek egy véges halmaza,
- K a kulcsok egy véges halmaza.

Három algoritmust értelmezünk:

- Gen, a kulcs-generáló algoritmus, meghatározza a key kulcsot,
- Enc_{key} a rejtjelező algoritmus, a key kulcs alapján, meghatározza az $m \in M$ nyílt-szöveg rejtjelezett értékét: $c \leftarrow Enc_{key}(m)$,
- Dec_{key} a visszafejtő algoritmus, a key kulcs alapján visszafejti a c rejtjelezett-szöveget: $m \leftarrow Dec_{key}(c)$.
- A rendszer helyessége érdekében megköveteljük: $Dec_{key}(Enc_{key}(m)) = m$, minden $m \in M$ esetében.

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

A titkos-kulcsú rendszerek matematikai modellje

- megnevezések: titkos-kulcsú rendszerek, szimmetrikus rendszerek (private-key, symmetric crypto),
- jelölés: SKE-vel, a (K, M, C) halmaz-hármas felett értelmezzük,
- 3 algoritmust szükséges értelmezni:
  - 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} a rejtjelező algoritmus, polinom idejű, véletlenszerű:
    $$c \xleftarrow{R} Enc_{key}(m),$$
  - a Dec_{key} a visszafejtő algoritmus, polinom idejű, determinisztikus:
    $$m \leftarrow Dec_{key}(c),$$
- Helyesség: $Dec_{key}(Enc_{key}(m)) = m$, minden $m \in M$ esetében.

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,

1. értelmezés

Egy Gen, Enc, Dec algoritmusok által értelmezett titkosító rendszer, amelynek biztonsági paramétere (kulcsmérete) k, tökéletesen biztonságos (perfectly secret), ha M bármely valószínűség eloszlása esetében és bármely $m \in M$, $c \in C$ esetében, fennáll:

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

ahol $Pr[C = c] > 0$.

- 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,
- több, a fentivel ekvivalens definíció létezik

Biztonság-értelmezések

1. tétel

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

- a tétel azt jelenti, hogy nehéz gyakorlatilag olyan titkosító rendszert létrehozni, amely tökéletes biztonságú.
- más biztonság-értelmezés: 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.

Biztonság-értelmezések

A számítástechnikai biztonság, feladja a tökéletes biztonságot, feltételezi, hogy:

- a támadó számítás-kapacitása (idő, tár) korlátos,
- a támadás csak nagyon kicsi valószínűséggel lehet sikeres.

A szakirodalom a különböző támadási típusokat figyelembe véve különböző biztonság értelmezéseket ad meg, ahol a támadási típusokat a következők szerint lehet osztályozni:

- ismert nyílt szövegű támadás (Known Plaintext Attack)
- választott nyílt szövegű támadás (Chosen Plaintext Attack)
- választott rejtett szövegű támadás (Chosen Ciphertext Attack)

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 $\oplus$ szimbólummal az XOR műveletet jelöltük, ami tulajdonképpen mod 2 szerinti összeadást végez.
- az $\oplus$ művelet algebrai tulajdonságai:

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

- 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:

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

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

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

2. 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

- különbség van a valódi (true) és az ál (pseudo) véletlen számok között
- az adatbiztonságban számos helyen van szükség véletlenszerűen előállított bájtokra/bitekre (random bits/random numbers)
- 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

Random számok

3. értelmezés

Egy 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. Fennáll továbbá, hogy 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 kulcs-csere, ezt a nyilvános kulcsú kriptográfia végzi,
- nincs megoldva a felek hitelesítése, ezt a nyilvános 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,
- 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.