Kriptográfia és Információbiztonság
2. előadás

MÁRTON Gyöngyvér

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

2024

Miről volt szó az elmúlt előadáson?

- Követelmények, osztályozás
- Bevezető: alkalmazási terület, törzsanyag
- Könyvészet
- Történelmi háttér
- Kriptográfiai alapfogalmak

Miről lesz szó?

- Klasszikus kriptográfiai rendszerek
  - eltolásos rejtjelezések: Caesar-titkosító, Keyword Caesar
  - helyettesítő rejtjelezés: affin-titkosító
  - blokk titkosítók: Hill titkosító,
  - feltörési módszerek,
  - matematikai modell
- a modern kriptográfia: tervezési szempontok, követelmények
- tökéletes biztonság
- számítástechnikai biztonság
- az OTP titkosítási rendszer

A Caesar-titkosító

- a titkosító azért viseli a Caesar nevet mert a $key = 3$ esetet adó rejtjelezést Julius Caesar római császár használta,
- a titkos kulcs már korábban megosztásra került,
- a klasszikus kriptográfia fizikai úton továbbította a rejtjelezéshez használt kulcsokat,
- a kulcsokat hosszú ideig változatlanul használták és szigorúan őrizték,
- a Caesar-titkosító nem biztonságos több feltörési módszer is kivitelezhető:
  - az összes lehetséges kulcs kipróbálása (exhaustive key search): a titkosító összesen 26 különböző kulcsot tud kezelni
  - statisztikai elemzések: betű, betű-pár, betű-hármas, vagy szavak gyakoriság vizsgálatát kell elvégezni.

Caesar titkosító

- az üzenetek halmaza: $M = C = \{0, 1, \ldots, 25\}^*$, az angol ábécé 26 betűjének megfelelő számkód:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*

- a kulcsok halmaza: $K = \{0, 1, \ldots, 25\}$,
- kulcsgenerálás *Gen*: kiválasztunk egy $key \in K$ értéket,
- titkosítás: $Enc_{key}(m) = (m + key) \pmod{26} \rightarrow c$, ahol $m \in M$,
- visszafejtés: $Dec_{key}(c) = (c + 26 - key) \pmod{26} \rightarrow m$.
- Megjegyzés: $key = 0$ nincs titkosítás, $key = 3$, az eredeti Caesar titkosító
- a $26 - key$ értéket a *key* additív inverzének hívjuk (mod 26) szerint
- Feltörési módszer: az összes lehetséges kulcs kipróbálása (exhaustive key search): 26 lehetséges kulcs van.

Példa:

- Ha a nyílt szöveg a következő: *THISISAPLAINTEXT*
- Ha a kulcs 5, akkor a titkosított szöveg: *YMNXNXFUQFNSYJCY*
- A megfelelő titkosító tábla:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*
F	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*

Helyettesítő rejtjelezés

- $M = C = \{0, 1, \ldots, 25\}$
- K elemszáma: a 26 szimbólum összes lehetséges permutációja, *key* egy lehetséges permutáció,
- *Gen*: kiválasztunk egy permutációt,
- $Enc_{key}(m) = Perm(m) \rightarrow c$,
- $Dec_{key}(c) = Perm^{-1}(c) \rightarrow m$.
- Feltörési módszerek:
  - az összes lehetséges kulcs kipróbálása nem működik, mert a lehetséges kulcsok száma: 26! = 403291461126605635584000000
  - működik: betű, betű-pár, betű-hármas, szavak gyakoriság vizsgálat.

Példa, Keyword Caesar titkosító, ahol a kulcs két értékből tevődik össze:

- Ha a nyílt szöveg a következő: *THISISAPLAINTEXT*
- Ha a kulcs 7, *PASWD*, akkor a titkosított szöveg: *PLMZMZEVQEMTPIDP*
- A megfelelő titkosító tábla, ahol vegyük észre, hogy $26 - 19 = 7$:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*
E	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*Q*	*R*	*T*	*U*	*V*	*X*	*Y*	*Z*	*P*	*A*	*S*	*W*	*D*	*B*	*C*

Az Affin rejtjelezés

- helyettesítéses, monoalfabetikus rejtjelezés,
- $M = C = \{0, 1, \ldots, 25\}^* = \mathbb{Z}_{26}, K = \{(a, b) \in \mathbb{Z}_{26}, gcd(a, 26) = 1\}$,
- $key = (a, b)$
- $Enc_{(a,b)}(m) = (a \cdot m + b) \pmod{26} \rightarrow c$,
- $Dec_{(a,b)}(c) = a^{-1} \cdot (c + 26 - b) \pmod{26}$.
- a^{-1}, az a multiplikatív inverze $\pmod{26}$ szerint, ami azt jelenti, hogy meg kell határozni azt az a^{-1} értéket, amelyre fennáll:

$$a \cdot a^{-1} = 1 \pmod{26}.$$

- a multiplikatív inverz akkor létezik, ha $gcd(a, 26) = 1$, ahol gcd - greatest common division/legnagyobb közös osztó
- Ha létezik multiplikatív inverz, akkor az meghatározható:
  - az összes érték kipróbálásával: 12 szorzás szükséges,
  - a kiterjesztett Eukleidészi algoritmussal,
  - az Euler tételt alkalmazva: $a^{-1} = a^{\phi(26)-1} = a^{11} \pmod{26}$, mert $\phi(26) = \phi(2) \cdot \phi(13) = 12$.

Az Affin rejtjelezés - példa

- Ha a kulcs $(5, 2)$ és a nyílt-szöveg a következő:

$$AMATHEMATICIAN,$$

- akkor a titkosított-szöveg:

$$CKCTLWKCTQMQCP,$$

- ahol a titkosított-szöveg első 6 karakterét a következőképpen határoztuk meg:

$$\begin{array}{llllll} A & -> & C: & (5 \cdot 0 + 2) & = & 2 \pmod{26} \\ M & -> & K: & (5 \cdot 12 + 2) & = & 10 \pmod{26} \\ A & -> & C: & (5 \cdot 0 + 2) & = & 2 \pmod{26} \\ T & -> & T: & (5 \cdot 19 + 2) & = & 19 \pmod{26} \\ H & -> & L: & (5 \cdot 7 + 2) & = & 11 \pmod{26} \\ E & -> & W: & (5 \cdot 4 + 2) & = & 22 \pmod{26} \end{array}$$

- A visszafejtéshez szükséges kulcs: $(21, 2)$, mert $5 \cdot 21 = 105 = 1 \pmod{26}$, azaz, 5 multiplikatív inverze $\pmod{26}$ szerint 21.

Az Affin rejtjelezés - ismert nyílt-szöveg támadás

Feltörési módszerek:

- gyakoriság vizsgálat
- az összes lehetséges kulcs kipróbálása, kulcsok száma: $12 \cdot 26 = 312$
- 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 Hill-titkosító

- 1929-ben publikálta Lester S. Hill,
- polialfabetikus titkosító: ugyanannak a betűnek nem mindig ugyanaz lesz a rejtjele
- blokktitkosító: egyszerre több betűt is képes titkosítani
- lineáris algebrán alapszik, mátrix műveleteket végez
- az üzenetek halmazát az angol/latin ábécé 26 betűjének megfelelő számkódok alkotják,
- a *key* kulcs egy $d \times d$-es mátrix lesz, d a blokkméretet,
- ha az üzenet nem osztható a blokkmérettel, akkor az üzenetet kiegészítik, azaz padding-olják.

A Hill-titkosító

- a titkosítási, visszafejtési képletet d betűnként kell alkalmazni,
- a mátrix az elemeit az $\{0, 1, \ldots, 25\}$ halmazból veszi fel
- fenn kell álljon: $gcd(det_{key}, 26) = 1$, ahol det_{key}-el a key mátrix determinánsa,
- egy $m \in M = \{0, 1, \ldots, 25\}^d$, d betűs üzenet titkosítása:

$$Enc_{key}(m) = key \cdot m \rightarrow c.$$

- a visszafejtés:

$$Dec_{key}(c) = key^{-1} \cdot c \rightarrow m,$$

  ahol key^{-1} a key inverze
- meg kell határozni a kulcs inverzét: kell a key determinánsa, a determináns multplikatív inverze, és az inverz mátrix értéke, $\pmod{26}$ szerint

A Hill-titkosító

- ha a blokkméret $d = 2$, akkor egyszerre 2 betűt lehet titkosítani. Legyen

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

- ha $m = (m_1, m_2)$ és $c = (c_1, c_2)$, akkor a titkosítás:

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

- legyen a key determináns inverze det_{key}^{-1}, ekkor fenn kell álljon: $gcd(det_{key}, 26) = 1$,
- legyen az adjungált mátrix adj_{key}, ekkor:

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

- a visszafejtés:

$$key^{-1} \cdot \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} \longrightarrow (m_1, m_2).$$

A Hill-titkosító, kulcsgenerálás

Legyen

$$key = \begin{pmatrix} 3 & 3 \\ -2 & 1 \end{pmatrix}, \text{ és } d = 2.$$

Ekkor

$$\begin{aligned} det_{key} &= 1 \cdot 3 - (-2) \cdot 3 = 9, \\ det_{key}^{-1} &= 3, \text{ mert } 9 \cdot 3 \equiv 1 \pmod{26}, \\ adj_{key} &= \begin{pmatrix} 1 & -3 \\ 2 & 3 \end{pmatrix}, \\ key^{-1} &= det_{key}^{-1} \cdot adj_{key} = 3 \cdot \begin{pmatrix} 1 & -3 \\ 2 & 3 \end{pmatrix} = \begin{pmatrix} 3 & -9 \\ 6 & 9 \end{pmatrix}. \end{aligned}$$

A Hill-titkosító, 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-titkosító, általános eset

- az $m = (m_1, m_2, \ldots m_d) \in M$ nyílt szöveg egy d hosszúságú blokkja egy lépésben lesz titkosítva,
- legyen a bemeneti ábécé elemszáma p, és álljon az első p darab nem negatív egész számból, a számításokat $\pmod p$ szerint kell végezni,
- a *key* mátrix i, j elemét $k_{i,j}$-vel jelölve a titkosítás a következő:

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

- amikor a bemeneti ábécé az első 256 darab nem negatív egész számból áll, akkor a titkosítást bájtokon végezzük, a műveleteket $\pmod{256}$-al kell végezni,
- a visszafejtéshez szükség lesz a $d \times d$ mátrix inverzére: $gcd(det_{key}, p) = 1$,
- a Hill kulcsgeneráló algoritmusa bonyolult,
- az inverz mátrix meghatározásához használhatjuk a Python *SymPy* (Library for Symbolic Matematics) könyvtárcsomagját,
- a titkosítás során a bemenet egy lineáris transformációját határozzuk meg, a titkosító nem biztonságos.

A Hill-titkosító - 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.

Klasszikus kriptográfiai rendszerek - feltörési módszerek

Klasszikus kriptográfiai rendszerek esetében 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.

A klasszikus titkosítás matematikai modellje

Klasszikus titkosítók (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.

A modern kriptográfia tervezési szempontok

- kriptográfiai primitívek:
  - álvéletlenszám generáló algoritmusok,
  - folyamtitkosítók,
  - blokktitkosítók,
  - hash függvények,
  - üzenet hitelesítő kódok,
  - a publikus kulcsú kriptográfia algoritmusai
- a kriptográfiai primitívek tervezése az elméleti kriptográfia területéhez tartozik,
- a kriptográfiai primitívek segítségével kriptográfiai protokollok építhetők,
- azért mert egy kriptográfiai primitív biztonságos, nem biztos, hogy a belőlük felépített kriptográfiai protokoll biztonságos,
- a kriptográfiai protokollok tervezése is az elméleti kriptográfia tárgyát képezik,

A modern kriptográfia tervezési szempontok

- a kriptográfiai protokollok mellett a biztonsági rendszerek működését is protokollok alapján kell megadni,
- a szakirodalom mindkét esetben a protokoll megnevezéssel él, de a kettő között különbség van,
- a valós rendszerek biztonságát is protokollok/alkalmazások teszik lehetővé, ezek tervezése, működtetése azonban az adatbiztonság területéhez tartozik,
- példa, kriptográfiai protokoll:
  - végpontok közötti titkosítás
  - hitelesített kulcscsere
  - zero-knowledge protokollok, partner-hitelesítés, stb.
- példa, biztonsági protokoll: SSL/TLS, Signal, PGP

A modern kriptográfia követelmények

- a kriptográfiában használt primitívek standard nyílt forráskódú algoritmusok kell legyenek: a standardizáló bizottság már korábban biztonsági szempontból megvizsgált és elfogadott,
- Kerckhoff elv (1883): egy kriptorendszer működési elve teljesen publikus kell legyen, egyedül a titkosításhoz használt kulcsot kell szigorúan titokban tartani, azt csak a kommunikáló felek ismerhetik,
- Claude Shannon (1949) amerikai matematikus: a kriptorendszereket, úgy kell tervezni, hogy szem előtt kell tartani, hogy egy támadó azt azonnal ki fogja ismerni,
- számítástechnikai szempontból nehéz feladatokra kell alapozni, de az ilyen feladatok nem olyan gyakoriak

A modern kriptográfia követelmények

"Nehéz" feladatok, azaz nem ismert rájuk hatékony algoritmus:

- NP-teljes feladatok:
  - nem lehet felhasználni titkosító algoritmusoknál,
  - a legrosszabb esetben nehezek, de az átlagos esetben könnyűnek számítanak,
  - például sokáig a hátizsák feladattal próbálkoztak: Határozzuk meg, hogy N tárgy közül, melyek lesznek azok a tárgyak, amelyeket belepakolunk egy hátizsákba, ismerve a hátizsák maximális kapacitását, mindegyik tárgy súlyát, illetve használati értékét.
- számelméleti feladatok:
  - faktorizációs probléma: két szám szorzatának a meghatározása gyorsan megadható azonban, ha csak a szorzat áll a rendelkezésünkre, akkor a szorzótényezők meghatározására máig sem ismert hatékony algoritmus,
- nem létezik sok alkalmas feladat.

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 szükség, 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 egy 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 egy év alatt,
- 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 $m = (m_1, \ldots, m_k) \in M$, $c = (c_1, \ldots, c_k) \in C$,
  $key = (key_1, \ldots, key_k) \in K$,
- $Gen \xrightarrow{R} key$, (egyenletes eloszlás, véletlenszerű, egyszer használt)
- $Enc_{key}(m) : (m_1 \oplus key_1, \ldots m_k \oplus key_k) \to c$,
- $Dec_{key}(c) : (c_1 \oplus key_1, \ldots c_k \oplus key_k) \to m$.
- 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