

1 ☐ **Adatvédelem, Adatbiztonság**

Alapvető titkosítási technológiák, titkosítási alapfogalmak, véletlen szám generálása

2 ☐ **Titkosítási alapfogalmak**

- Plaintext: Egyszerű szöveg, ami titkosításra kerül.
- Ciphertext: Titkosított szöveg, amit algoritmus állít elő a bemeneti szövegből.
- Encryption, titkosítás: A folyamat, amely létrehozza az egyszerű szövegből a titkosított szöveget.

3 ☐ **Titkosítási alapfogalmak**

- Titkosító algoritmus: A titkosítás műveletsora. Mindig két bemenete van: titkosítandó szöveg és titkosító kulcs.
- Visszafejtés, Deciphering, Decryption: Folyamat, amely során titkosított szövegből egyszerű szöveg lesz.
- Visszafejtő algoritmus: Visszafejtés műveletsora. Két bemenete: titkosított szöveg és titkosítás feloldó kulcs.
-

4 ☐ **Titkosítási alapfogalmak**

- Titkos kulcs: titkosító és titkosítást feloldó kulcs elnevezése, ha megegyeznek. Szokás szimmetrikus kulcsnak is nevezni.
- Kriptográfia, Cryptography: Titkosításokkal foglalkozó tudomány.
- Kriptoanalízis: Kriptográfiai rendszerek támadásával foglalkozó tudomány.
- Kriptológia: Kriptográfia + Kriptoanalízis

5 ☐ **Titkosítások fajtái**

- Szimmetrikus:
 - Azonos kulcs használatos a titkosításhoz és a titkosítás feloldásához is.
 - Blokk alapú: Fix méretű blokkal dolgozik az adatfolyam végéig.
 - Stream, folyamatos: A titkosítandó adatot bitenként dolgozza fel.
- Aszimmetrikus
 - 1 kulcs a titkosításhoz és egy másik a titkosítás feloldásához.

6 ☐ **Szimmetrikus titkosítási rendszerek**

- Minden klasszikus titkosítási modell ezen alapul
- Mai napig is a legelterjedtebb megoldás
- 1970-es évekig, az aszimmetrikus rendszerek feltalálásáig másmilyen titkosítás nem is létezett.

7 ☐ **Szimmetrikus titkosítás matematikailag, mert szeretjük ©**

$$Y = E_K(X) \quad \text{vagy} \quad Y = E(K, X) \\ X = D_K(Y) \quad \text{vagy} \quad X = D(K, Y)$$

- X = Bemeneti szöveg
- Y = Titkosított szöveg
- K = Titkosító kulcs
- E = Titkosító algoritmus
- D = Titkosítást feloldó algoritmus
- E és D nyílt, bárki által megtekinthető, megismerhető

8 ☐ **Kriptoanalízis**

- Cél: Titkosított szövegből bemenet kinyerése, vagy jobb esetben titkosító kulcs meghatározása.
- Kerkhoff elv: A támadó mindent tud a titkosításról, csak a titkosító kulcsot nem.
- 2 támadási módszer:
 - Nyers erő, Brute Force
 - Kriptoanalitikus támadás

9 ☐ **Brute Force Módszer**

10 ☐ **Brute Force módszer**

- Minden lehetséges titkos kulcs végigpróbálása.
- Cél az lenne, hogy az algoritmusunk csak így legyen támadható.
- Átlagosan csak a lehetséges kulcsok felét kell kipróbálni a sikeres töréshez.

11 ☐ **Brute Force időigénye**

12 ☐ **Kriptoanalitikai támadások**

- Következő csoportokra bontható:
 - Titkosított szöveg támadás
 - Ismert bemenet alapú támadás
 - Választott bemenet alapú támadás
 - Választott titkosított szöveg támadás

13 ☐ **Titkosított szöveg alapú támadás**

- Adott C titkosító kulcs.
- Kérdés: Mi a C-hez tartozó M bemeneti szöveg ?
- Egy titkosítási algoritmus totál használhatatlan, ha nem tud ellenállni titkosított szöveg alapú támadásoknak

14 ☐ **Ismert bemeneti szöveg támadás**

- Adott $(m_1, c_1), (m_2, c_2), \dots, (m_k, c_k)$
- Ezeknek segítségével próbálunk rájönni a C-t előállító algoritmus gyengéire, valamint a titkosító kulcsra.
-

15 ☐ **Választott bemenet alapú támadás**

- Adott: $(m_1, c_1), (m_2, c_2), \dots, (m_k, c_k)$, ahol m_1, m_2, \dots, m_k a támadó által választott, valamint egy ismeretlen C
- Cél:Ebből rájönni a C mögött álló M-re, vagy a használt kulcsra

16 ☐ **Választott bemenet alapú támadás példa**

- 1942-ben az Amerikai Haditengerészet lehallgatott egy japán üzenetet, amiben az állt, hogy „AF” jelzésű célpont elleni támadás tervben van.
- Feltételezés az volt, hogy „AF” = Midway sziget
- A feltételezés megerősítésére azt a rádió üzenetet küldték, hogy a Midway szigeten lévő katonák utánpótlásai kifogyóban vannak.
- Nem sokkal később lehallgatott japán üzenetben az állt, hogy „AF” készletei fogytán.

17 ☐ **Választott Titkosított szöveg támadás**

- Adott: $(m_1, c_1), (m_2, c_2), \dots, (m_k, c_k)$, ahol c_1, c_2, \dots, c_k a támadó által választott, valamint egy ismeretlen C .
- Cél: Ebből rájönni a C mögött álló M -re, vagy a használt kulcsra
-

18 ☐ **Klasszikus titkosítási módszerek**

19 ☐ **Klasszikus titkosítási módszerek**

- Nagyjából 1970-es évekig használtak
- Fő alkalmazási terület: Harcászat
- A titkosítandó szöveg elemek sorozataként van kezelve (vagy bitenként, vagy karakterek sorozataként)

20 ☐ **Klasszikus Titkosítási módszerek**

- Algoritmus minták:
 - Helyettesítéses – bemeneti szöveg elemei valamilyen szabály szerint cserélődnek egy másik elemre
 - Transzpozíciós/permutációs – bemeneti szöveg elemeinek sorrendje keveredik valamilyen minta szerint
 - Eredmény – a bemeneti szöveg titkosított változata több lépés eredményeképpen áll elő.

21 ☐ **Caesar titkosítás (adatfolyam)**

- Legöregebb ismert cserés titkosítási algoritmus
- Julius Caesar találta fel.
- Titkosítás elve: Minden betű 3 hellyel van eltolva az ABC-ben.
 - be: 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
 - ki: D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
 - Példa: A titkosítás jo -> D WLWNRVLWDV MR

22 ☐ **Caesar titkosítás matematikailag ☺**

- Betűkhöz számok hozzárendelése $A = 0, B = -1, C = -2, \dots, Z = -25$
- Ekkor a következőképpen írható le az algoritmus:
 - $c = E_k(p) = (p + k) \bmod 26$
 - $p = D_k(c) = (c - k) \bmod 26$
- Általánosítható bármilyen ABC-re

23 ☐ **CAESAR Titkosítás Kriptoanalízise**

- Kulcsere $\{0 \dots 25\}$ között mozog angol ABC esetén
- Brute Force törési módszerrel viszonylag könnyen visszafejthető
 - Visszafejtőnek fel kell ismernie, hogy mikor kész a szöveg.
 - Gond lehet, ha a bemenet olyan nyelven van írva, amit a törő nem ismer. Pl: Klingon

24 ☐ **Caesar titkosítás általánosítása**

- Monoalfabetikus keveréses titkosítás
- Lényege, hogy a bemeneti szöveg karaktereit, valamilyen logika szerint kicseréljük egy másik betűre.
- Példa:
 - be: abcdefghijklmnopqrstuvwxyz

ki: DKVQFIBJWPESCXHTMYAUOLRGZN

25 ☐ **Monoalfabetikus keveréses titkosítás**

- Kulcs: 26! (angol ABC esetén)
- Brute Force ellen védett, de kriptanalitikus szempontból gyenge.
- Gond: A nyelv karakterisztikája. Egyes betűk jóval többször szerepelnek, mint mások.
- További gond: betű ismétlődések és összetett írásjelek. Pl. magyarban: cs, sz, zs

26 ☐ **Az angol ABC betűinek ismétlődési gyakorisága**

27 ☐ **Monoalfabetikus keverés támadása**

- Kulcs koncepció: A keverés nem változtatja meg a relatív betűismétlődéseket.
- Támadás menete:
 - Titkosított szövegben betűk ismétlődési gyakoriságának kiszámítása
 - Eloszlás hasonlítása egy ismert eloszláshoz

28 ☐ **Playfair titkosítás (Blok)**

- Monoalfabetikus rendszereknél a nagy kulcs: sem véd.
- Kiküszöbölhető a hiba, ha egy lépésben több betűt titkosítunk.
- Charles Weastone találta ki 1854-ben, de a barátja, Playfair báró után kapta nevét

29 ☐ **Algoritmus menete**

- 5x5-ös mátrix rajzolása az ABC számára
- ABC beírása, ez lesz a kulcs
- Maradék hely kitöltése egyéb betűkkel
- Példa kulcs:

30 ☐ **Algoritmus menete**

- A titkosítandó szöveg felbontása két karakteres párokra.
- J betűk cserélése I betűkre
- Ha a két karakteres pár mindkét karakterje azonos, akkor kitöltő karaktert kell alkalmazni, ami: x
- Ha a szöveg végén kell még betű a két karakteres párhoz, akkor Z betűt kell alkalmazni

31 ☐ **Algoritmus menete**

- Ezután szöveg titkosítása a következő szabályok betartásával:
 - Ha két karakter ugyanabban a sorban szerepel az ABC táblában, akkor a karakter mellett jobbra található karakter lesz használva (körkörösen!)
 - Ha két karakter ugyanabban az oszlopban szerepel, akkor az egyel alatta lévő karakter lesz használva. (körkörösen!)
 - Ha a fenti két eset nem áll fent, akkor kereszt cserét kell csinálni.

32 ☐ **Kereszt csere**

- Példán keresztül elmagyarázni a legkönnyebb:
 -
 -
 -
 -
 -
- Pl. BD karakterpár a KH párra fordul, mivel a B betűhöz a D betű alatt lévő karaktert kell

- rendelni
► A D betűhöz meg a B alatti betűt.
- 33 ☐ **Ha az oszlop vagy sor egyezik példa**
► Pl. AM pár LE-re fordul, ME pedig EP-re
► RH pár BV-re, a HV pedig VP-re.
- 34 ☐ **Komplex példa**
► Példa szöveg: Lord Granville's letter
► Bontás után:
lo rd gr an vi lx le sl et te rz
► Titkosítás után:
MT TB BN ES WH TL MP TA LN NV
► Visszafejteti az ABC és a szabályok fordított alkalmazásával lehet.
- 35 ☐ **Playfair biztonsága**
► Kulcs tere 26×26 karakter = 676 karakter egy üzenet esetén.
► Valaha feltörhetetlen kódnak hitték, de mint kiderült, fel lehet törni, mivel a módszer néhány betűismétlődést és összetett szót figyelmen kívül hagy.
► Széles körben használt volt az I. és II. világháború alatt az Amerikai és Brit hadsereg által.
- 36 ☐ **Polialfabetikus titkosítás**
► Monoalfabetikus titkosítási kulcsok használtak körönként a titkosításban
► A kulcs határozza meg, hogy melyik betű melyik titkosítás szerint lesz titkosítva
► Kriptoanalízist nehezíti, mivel az eloszlás függvény sokkal egyenletesebb lesz.
- 37 ☐ **Vigenère titkosítás**
► A legegyszerűbb polialfabetikus cserés titkosítás
► Tételezzük fel, hogy adott az összes Caesar ABC variáció: $\{ C_a, C_b, C_c, \dots, C_z \}$
► Kulcs legyen pl.: biztonság
► Betűk titkosítása a következő elven $C_b, C_i, C_z, C_t, C_o, C_n, C_s, C_a, C_g$
► Ismétlés C_g után
► A titkosítás feloldása visszafelé történik
►
- 38 ☐ **Vigenère titkosítás biztonsága**
► Minden betűhöz több titkosított betű tartozik (Kérdés, hogy mennyi ?)
► A betűk ismétlődési statisztikái majdnem használhatatlanok.
- 39 ☐ **Vigenère titkosítás biztonsága**
► Feltörése:
► Ki kell találni a kulcs hosszúságát
► Ha a kulcs hosszúsága N, akkor az algoritmus N Caesar titkosítást használ.
► K pozícióban lévő és $N+k, 2N+k$, valamint $3N+k$ betűk ugyanazzal a módszerrel vannak titkosítva.
► Különálló Caesar kulcsok törése úgy, mint eddig.
►
- 40 ☐ **Vigenère titkosítás a Gyakorlatban**
► Elektromechanikus titkosító gépek.

- Német Enigma gép
 - 3-10 tárcsás titkosítás
 - 3 tárcsa esetén a kulcskombinációk száma: 26^3
 - 5 tárcsa esetén: $26^5 = \sim 12$ millió
 - 10 tárcsa esetén: 26^{10}

41 ☐ **Enigma – Első alkalommal volt részletesen**

42 ☐ **Permutációs titkosítás**

- A betűk keverték valamilyen logika alapján
- A használt betűk nem változnak
- Példa: Sor transzpozíciós titkosítás

43 ☐ **Sor transzpozíciós titkosítás**

- Titkosítandó szöveg beírása egy mátrixba, majd a kulcssorozat szerint oszlopok kiválasztása és tartalmuk leírása.
- Példa:
- Kulcs:
1 4 6 3 7 2 5
- Eredmény:
ttygskadepselbytsoduigoczozogw

44 ☐ **Eredmény titkosítások**

- Több egyszerűbb titkosítási algoritmus összességét használják
- Híd a modern kriptográfia felé
- Elvénél fogva sokkal nehezebben törhető, mint a hagyományos titkosítási módszerek.

45 ☐ **Feltétlen biztonság és számítási biztonság**

- Egy algoritmus feltétlenül biztonságosnak nevezett, ha nem számít, hogy a támadónak mennyi ideje és erőforrása van.
- Egy algoritmus számításiag biztonságosnak tekintett, ha a feltöréséhez annyi erőforrás és idő szükséges (pl. 1000 év), ami nem áll senki rendelkezésére.
- Az eddig ismert algoritmusok nem felelnek meg a feltétlen biztonság követelményének.

46 ☐ **Feltétlen biztonságnak megfelelő titkosítás**

- Vernam-féle titkosítás.
- Elve:
 - $(A \text{ xor } K) \text{ xor } K = A$
 - Biztonságos, amennyiben K egyszer használatos, véletlenszerű és legalább olyan hosszú, mint a titkosítandó adat.
 - Amennyiben nem ilyen hosszú, akkor könnyű feltörni.

47 ☐ **Véletlen számok generálása**

48 ☐ **Véletlen számok generálása**

- A véletlenszám generátor (RNG) egy olyan berendezés / szoftver, ami olyan számok sorozatát képes generálni, amiben nincsen minta.
- Nem egyszerű feladat.
- Neumann János: „Bárki, aki aritmetikai módszerekkel akar előállítani egy véletlen

számot, a bűn állapotában leledzik.”

49 ☐ **Véletlen számok generálása**

- A kriptográfia számára megfelelő véletlenszám generátorokat kriptográfiailag biztonságos pszeudó véletlenszám generátornak nevezzük (CSPRNG)
- Elvárások a CSPRNG algoritmusokkal szemben:
 - Át kell mennie a következő bit teszten
 - Ki kell állnia az állapot kompromittálódási tesztet.

50 ☐ **Következő bit teszt**

- Egy bitsorozat átmegy a következő bit teszten, ha a támadó a bitsorozat bármely i -edik bit előtti sorozatot ismerve nem képes megmondani belátható időn belül $i+1$ bit értékét matematikai számításokkal levezetve.

51 ☐ **Állapot kompromittálódási teszt**

- Amennyiben a támadónak sikerült kitalálnia a véletlenszám generátor jelenlegi állapotát, és ennek ismeretében sikerül kitalálnia a következő állapotot, akkor az algoritmus megbukott az állapot kompromittálódási teszten.

52 ☐ **Állapot kompromittálódási teszt**

- Példa:
 - Tételezzük fel, hogy a véletlenszám generátor algoritmusom a π számjegyeit használja véletlenszerűen megválasztott kezdőpont alapján.
 - Tudjuk a π -ről, hogy végtelen és nincs benne minta.
 - Viszont, ha kitalálom, hogy mi a generálási kezdőpont, onnantól kezdve az algoritmus alapján ki tudom számítani bármelyik számjegyet.

53 ☐ **Véletlenszám generátorok**

- A legtöbb nem kriptográfiai véletlenszám generátor algoritmus hasonló elven működik.
- Adott egy generáló függvény, amit inicializálni kell egy kezdőértékkel.
- Inicializálás után „véletlen” számok sorozatát köpi ki az algoritmus.
- Azonban, ha tudom a kezdőértéket, akkor már nem is véletlen...
-

54 ☐ **Véletlenszám generátorok**

- Ideális esetben az inicializációs értéknek is véletlennek kellene lennie, de ez felveti a „tyúk vagy tojás volt előbb” klasszikus problémát.
- A megoldás:
 - Idő alapján inicializálunk
 - Környezetből mintavételezünk adatot.

55 ☐ **Idő alapján inicializálunk**

- Sci-Fi irodalomtól eltekintve a fizika jelenlegi állása szerint az idő csak egy irányban halad.
- Ennek alapján, ha egy másodperc részt felbontok nagyon kicsi részegységekre, akkor jó közelítéssel alkalmazhatom ezt inicializálásra.
- Egy modern számítógép az időt másodpercekben méri, általában 1970.01.01 00:00-tól kezdve (EPOCH Time) 32 vagy 64 biten.

56 ☐ **Idő alapján inicializálunk**

- Egy másodpercet további 32 bitre bontunk, ami 233 pikomásodperc pontosságot (233×10^{-12}) eredményez.
- Ez jó közelítéssel kitalálhatatlan, de elméletileg lehetséges, illetve az óra pontossága piszkálható.

57 ☐ **Környezetből mintavételezés**

- Az összes analóg/digitális mintavételezésnek van némi hibája a technológiából adódóan.
- Ez általában problémát jelent, mivel ez zajként jelentkezik a digitalizálás során.
- Ez a zaj ténylegesen véletlen mintázatot követ kitalálhatatlan.
- Ha kifejezetten zajt mintavételezünk, akkor ténylegesen véletlen számokat kapunk.

58 ☐ **Környezetből mintavételezés**

- A zaj forrása:
 - A CMOS áramköri gyártás során FET tranzisztorokat alkalmaznak, amelyek feszültségvezéreltek.
 - Ezen tranzisztorok gate elektródája teljesen szigetelt a kapcsoló rétegtől, hogy ne folyjék áram, vagyis a fogyasztás minimális legyen.
 - A dolog hátulütője, hogy így bemeneti kapacitás keletkezik, vagyis a statikus töltések már működésbe hozzák a tranzisztort.

59 ☐ **Környezetből mintavételezés**

- Mivel az A/D átalakítók is CMOS áramkörök, így ha egy A/D áramkör bemenetére egy antennát teszünk, akkor összeszedi a környezet statikus elektromosságát, ami bizonyos részeit működésbe hozza.
- Így zajt digitalizálunk.

60 ☐ **Na jó, ok, de honnan jön a zaj?**

- Folyamatosan ott van körülöttünk.
- Rádióhullámok, elektromos vezetékek, kozmikus sugárzás, stb...
- Probléma annyi, hogy bizonyos helyek nagyobb zajforrással rendelkeznek, mint más területek.
- Így lényegében ez a megoldás sem teljesen tökéletes.

61 ☐ **Környezetből mintavételezés**

- Mégis számos helyen alkalmazott
- Olyan hardverelemek esetén is, ahol a belső környezet nem igen piszkálható meg.
- Legújabb kutatási eredmények a kvantummechanikát is belekeverik a történetbe.

62 ☐ **Kvantum mechanikai véletlenszerűség**

- Vákuum kvantum fluktuációjának mérésén alapul.
- Kvantummechanikai definíció szerint a vákuum egy olyan hely, amely anyagtól és fotonoktól mentes.
- Ilyen környezetben azt lehet tapasztalni, hogy részecskék keletkeznek és tűnnek el időről időre.

63 ☐ **Kvantum mechanikai véletlenszerűség**

- Ez annak a következménye, hogy a vákuum rendelkezik nullponti energiával
- Ennek mintavételezése alapján valódi véletlen számok kaphatóak.
- Interneten is elérhető ilyen:

► <http://qrng.anu.edu.au/>

64  **Köszönöm a figyelmet**