Információbiztonság és Adatvédelem

2025/2026

László Ernő

Félév teljesítése

- **Évközi feladatok házi feladatok**
  - nem teljesítés esetén nem lehet vizsgázni

- Vizsga / ZH
  - Teszt feladat – elmélet (40%)
  - Gyakorlati feladat (60%)

Miről lesz szó

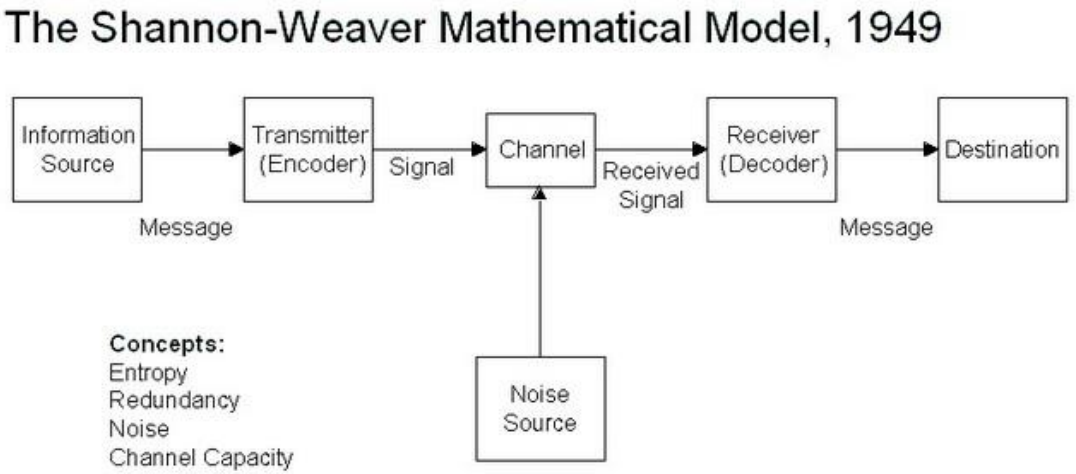
- **Információelmélet alapfogalmak** – Shannon, entropia, tömörítés
- **Klasszikus kriptográfia** – Caesar, Vigenère, Enigma
- **Hibajavítás és csatornamodellek** – Hamming-kód, Reed–Solomon
- **Modern kriptográfia** – AES, RSA, hash-ek, digitális aláírás
- **Hálózati biztonság** – SSL, TLS, VPN, MITM támadások
- **Adatvédelem, GDPR** – anonimizálás, nyomkövetés, metaadat
- **Támadások és védekezés** – brute force, phishing, ransomware, sql injection
- **Jövő és kutatási irányok** – kvantumkriptográfia, AI biztonság

Gyakorlati feladatok

Információ

- **Információ = bizonytalanság csökkenése**.(Shannon) (https://hu.wikipedia.org/wiki/Shannon%E2%80%93Weaver-modell)

- Matematikai megközelítés (mérhető)
  - Bit
  - Redundancia
  - Entrópia



Példák

- Holnap vizsga lesz!

Mekkora az információ tartalma?

- Minél kevésbé várt egy üzenet, annál több információt hordoz.

$$I(x) = -log_2(P(x))$$

- **I(x)** az x esemény információtartalma
- **P(x)** az esemény valószínűsége
  - Ha valami **valószínűtlen** (pl. 1/1000), akkor **sok bit** információt hordoz.
  - Ha valami **nagyon valószínű** (pl. 0.99), akkor **kevés bitet** (közel 0).

Példák

- Feldobott érme információ tartalma: 1 bit (-log2(0.5))
- Ha az érme mindig fejet mutatna akkor nincs információ tartalma mert nem közöl váratlant

Miért fontos ez?

- Pl kódolásnál Morsee kód . E a leggyakoribb betű ezért ez a legrövidebb mert a legkisebb az információ tartalma
- Q ritkább ezért hosszabb kód.

Esemény	Valószínűség	Információ (bit)	Megjegyzés
Napsütés nyáron	0.9	~0.15 bit	Nem sok új, várt esemény
Hóesés nyáron	0.01	~6.6 bit	Nagyon váratlan → sok információ
Érmefeldobás (fej/írás)	0.5	1 bit	Klasszikus példa
Új jelszó megadása	1/1,000,000	~20 bit	Ritka → magas információtartalom

Miért számít a biztonság?

- Ti hol tároltok jelszavakat?
- Törték már fel bármilyen online fiókodat?
- Mi baj származhat belőle?

Az információelmélet és az adatbiztonság kapcsolata

- Információelmélet: tömörítés, hatékony adatátvitel → **mit tudunk megspórolni?**
- Adatbiztonság: titkosítás, integritás, hitelesítés → **mit kell megvédeni?**
- A kettő közös alapja: **mintázatok felismerése és elrejtése**.

- Példa:
  - Tömörítés = a *mintázat* alapján eldobunk redundáns adatot (ZIP, JPEG)
  - Titkosítás = a *mintázat* elrejtése, hogy még az se értse meg, aki elfogja.

Adatszivárgás, adatlopás esetek (1)

– Enigma feltörése – mit okozott?

Év	Esemény
1918	Enigma-gép első változata (német fejlesztés)
1932	Lengyel matematikusok feltörik az Enigma egyszerűbb verzióját
1939	Turing csatlakozik a Bletchley Parkhoz (UK)
1940	Bombe gép működni kezd (Turing és Welchman)
1941	Daily kód feltörve – fordulópont
1945	Háború vége – a munka titkos marad évtizedekig

Adatszivárgás, adatlopás esetek (2)

- Enigma ismerttő: https://www.youtube.com/watch?v=G2_Q9FoD-oQ

- Titkosító eljárások kipróbálása: https://cryptii.com/pipes/enigma-machine

Adatlopás céljai (1)

- **1. Pénzügyi nyereségszerzés**
  - **Bankkártyaadatok** ellopása → közvetlen pénzlehívás, online vásárlás
  - **Ransomware**: fájlok titkosítása után váltságdíj követelése
  - **Adatok eladása** a dark weben (pl. jelszavak, email listák)
- **2. Kémkedés, hírszerzés**
  - Állami szintű kibertámadások → katonai, diplomáciai vagy gazdasági titkok megszerzése
  - Ipari kémkedés: versenytárs szellemi tulajdonának megszerzése
- **3. Profilalkotás és manipuláció**
  - Felhasználói szokások elemzése marketing vagy politikai célokra
  - Algoritmusok tanítása (AI modellek) ellopott adatokkal
  - Facebook / Cambridge Analytica → szavazók manipulálása

Adatlopás céljai (2)

- **4. Személyes zsarolás vagy bosszú**
  - Intim vagy kényes adatok kiszivárogtatása (pl. Ashley Madison hack)
  - „Doxing” – valaki személyes adatainak nyilvánosságra hozása
- **5. Szabotázs / rombolás**
  - Adatbázis törlés, módosítás → működésképtelenné tenni egy rendszert
  - Bizalom lerombolása (pl. nyilvánosságra hozott ügyféladatok)

Adatlopás által okozott károk

- **1. Gazdasági kár**
  - Váltságdíjak, bírságok (pl. GDPR → akár éves árbevétel 4%-a)
  - IT-rendszerek helyreállítási költsége
  - Ügyfelek elvesztése → bevételkiesés
- **2. Reputációs kár**
  - Ügyfelek, partnerek, befektetők **bizalmának elvesztése**
  - Negatív sajtómegjelenések → márka értékének csökkenése
  - Cégérték esése (pl. Yahoo felvásárlási ára is csökkent emiatt)
- **3. Személyes következmények**
  - Az érintett személyek:
    - elveszíthetik állásukat (pl. bizalmas e-mailek szivárgása)
    - pszichológiai terhelést élnek meg (megfélemlítés, stressz)
    - anyagi károkat szenvednek el (identitáslopás)

Adatlopás által okozott károk

- **4. Jogi következmények**
  - GDPR / adatvédelmi hatósági eljárás
  - Peres eljárások (kártérítési igények)
  - Hatósági vizsgálatok, törvényi felelősség
- **5. Szolgáltatás-kimaradás / leállás**
  - Adatlopás után a rendszer működése **leállhat** (pl. zsarolóvírus)
  - Kórházak, tömegközlekedés, közigazgatás működését is érintheti

Adatlopás által okozott károk

- **4. Jogi következmények**
  - GDPR / adatvédelmi hatósági eljárás
  - Peres eljárások (kártérítési igények)
  - Hatósági vizsgálatok, törvényi felelősség
- **5. Szolgáltatás-kimaradás / leállás**
  - Adatlopás után a rendszer működése **leállhat** (pl. zsarolóvírus)
  - Kórházak, tömegközlekedés, közigazgatás működését is érintheti

Adatszivárgás, adatlopás esetek

1. Feladat

Mindenki lehetőleg különböző adatszivárgás esetről írjon egy 1-2 oldalas ismertetőt.

- Mi történt?
- Mi volt a cél?
- Elkerülhető lett volna?
- Használták az információt?

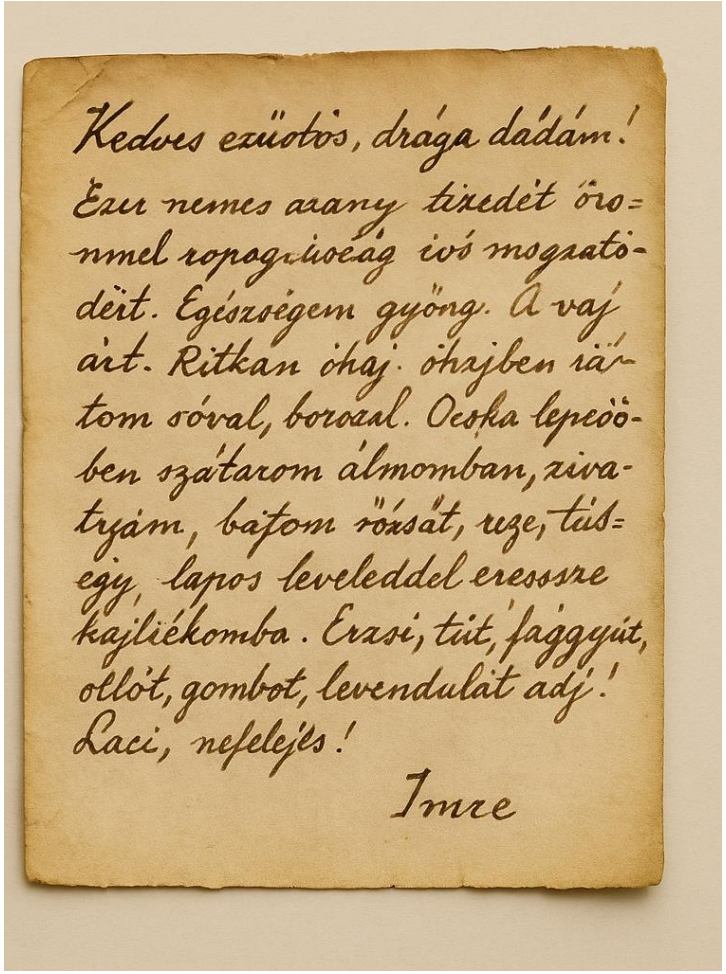
Szteganográfia

- Görög eredetű szó: *steganos* = fedett, *graphein* = írni
- Jelentése: információ elrejtése, nem csak titkosítása
- Cél: ne is derüljön ki, hogy üzenet van
- Kriptográfiával szemben nem a tartalom védelmét, hanem a **kommunikáció tényének** elrejtését célozza

Történelmi példák

- Hérodotosz: *viasztábla* (a viasz alá karcolt üzenet)
- Hisztiaiosz: rabszolga fejére tetovált titkos üzenet
- Láthatatlan tinta: citromlé, tej → meleg hatására válik láthatóvá
- Gárdonyi Géza: *Egy magyar rab levele* – akrosztichon (első betűk rejtik a valódi üzenetet)

Gárdonyi Géza – Egy rab levele


Kedves ezüstös, drága dádám!
Ezer nemes arany tizedét örömmel ropogtasdörök keserűség ivó magzatodért. Egészségem gyöngy. A vaj árt. Ritkán óhajtomsóval, borssal. Ócska lepedőben szárítkozomálmomban, zivataros estén. Matyi bátyám, egypár rózsát, rezet, ezüstöt, libát egy lapos leveleddel eressze hajlékomba. Erzsi, tűt, faggyút, ollót, gombot, levendulát adj!
Laci, nefelejts!
Imre

„Kedves ezüstös, drága dádém!
Ezer nemes arany tizedét örömmel ropogtasdörök keserűség ivó magzatodért.
Egészségem gyöngy. A vaj árt. Ritkán óhajtomsóval, borssal. Ócska lepedőben szárítkozomálmomban, zivataros estén. Matyi bátyám, egypár rózsát, rezet, ezüstöt, libát egy lapos leveleddel eressze hajlékomba.
Erzsi, tűt, faggyút, ollót, gombot, levendulát adj!
Laci, nefelejts!
Imre”

Gárdonyi Géza – Egy rab levele

„Kedves ezüstös, drága dádém!

Ezer nemes arany tizedét örömmel ropogtasd örök keserűség ivó magzatodért.

Egészségem gyöngy. A vaj árt. Ritkán óhajtom sóval, borssal. Ócska lepedőben szárítkozom álmomban, zivataros estén. Matyi bátyám, egypár rózsát, rezet, ezüstöt, libát egy lapos leveleddel eressze hajlékomba.

Erzsi, tűt, faggyút, ollót, gombot, levendulát adj!

Laci, nefelejts!

Imre”

Gárdonyi Géza – Egy rab levele

„Kedves ezüstös, drága dádém!

Ezer nemes arany tizedét örömmel ropogtasd örök keserűség ivó magzatodért.

Egészségem gyöngy. A vaj árt. Ritkán óhajtom sóval, borssal. Ócska lepedőben szárítkozom álmomban, zivataros estén. Matyi bátyám, egypár rózsát, rezet, ezüstöt, libát egy lapos leveleddel eressze hajlékomba.

Erzsi, tűt, faggyút, ollót, gombot, levendulát adj!

Laci, nefelejts!

Imre”

Kedden a török kimegy a városból. Száz emberrel el lehet foglalni.

Modern szteganográfia

- Digitális képek: pixelértékek legalacsonyabb bitjeiben rejthető adat (LSB)
- Hangfájlok: frekvenciák manipulálása az információ elrejtésére
- Videók, dokumentumok: metaadatok, fájlszerkezet használata
- Tipikus formátumok: PNG, WAV, MP4

Alkalmazási területek és kihívások

- Cenzúra megkerülése (diktatúrákban)
- Digitális vízjelek (pl. szerzői jogvédelem)
- Malware rejtése (kibertámadások)
- **Kihívások**: felismerés nehézsége, automatikus detektálás, ellenőrzés

- **Kriptográfiával** kombinálva hatékonyabb
- Szubliminális üzenetek - 25. képkocka
  - Éhes vagy!, Igyál kólát! => 58%-al nőtt a kóla és 18%-al a pattogatott kukorica eladása

Kriptográfia

- Ógörög eredetű: κρυπτός (kryptós) = „rejtett”, γράφειν (gráphein) = „írni”, tehát „titkosírás”
- Kriptográfia: információrejtés
- Kriptoanalízis: visszafejtés
- Kriptológia: kriptográfia + kriptoanalízis
- Állandó „harc”: rejtjelezők vs. kódfeltö

Kriptográfia célja

- **Titkosítás (Confidentiality)** – csak az olvassa, akinek szabad
- **Integritás (Integrity)** – ne lehessen észrevétlenül megváltoztatni
- **Hitelesítés (Authentication)** – tudjuk, ki küldte
- **Letagadhatatlanság (Non-repudiation)** – ne lehessen utólag letagadni

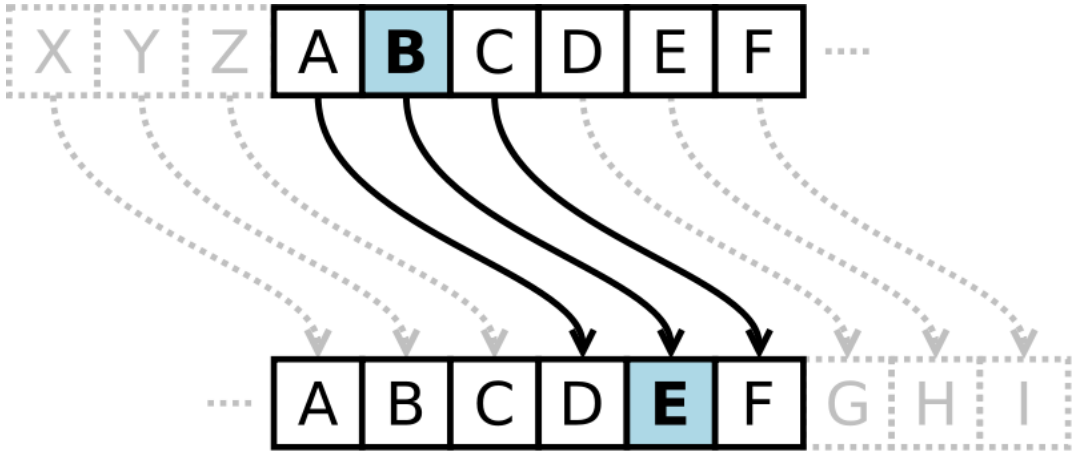
Klasszikus Kriptográfia

- **Példák:**
  - **Caesar-kód**: betűk eltolása egy fix kulccsal
    Pl. HELLO → KHOOR (3 pozícióval eltolva)
  - **Vigenère-kód**: kulcsszó alapján váltakozó eltolások
  - **Enigma-gép**: bonyolult rotoros mechanikus rendszer (WW2)
- könnyen megérthető, de könnyen feltörhető a mai technikával.

Klasszikus Kriptográfia

– **Caesar-kód**

Minden betűt kicserél egy,
az ABC-bentőle *k* távolságra
lévő betűvel



Általánosítva:

Minden betű helyett egy másikat használunk => bonyolultabb

26! Lehetőség

Ezt biztos nem lehet megfejteni, hiszen rengeteg párosítást kell végignézni… gondolták hosszú évszázadokig

Klasszikus Kriptográfia

Iszlám világ – Korán – több verzió – Mohamed szóban terjesztette később írták le

Arab tudósok vizsgálták mely részek származnak Mohamedtől és melyek nem.

Szavak előfordulását elemezték, majd a betűket is vizsgálták

Megszületett a **gyakoriságelemzés**

Klasszikus Kriptográfia

Smidla József tanár úr jegyzetei

Miről lesz szó

- ✅ **Információelmélet alapfogalmak** – Shannon, entropia, tömörítés
- ✅ **Klasszikus kriptográfia** – Caesar, Vigenère, Enigma
- **Hibajavítás és csatornamodellek** – Hamming-kód, Reed–Solomon
- **Modern kriptográfia** – AES, RSA, hash-ek, digitális aláírás
- **Hálózati biztonság** – SSL, TLS, VPN, MITM támadások
- **Adatvédelem, GDPR** – anonimizálás, nyomkövetés, metaadat
- **Támadások és védekezés** – brute force, phishing, ransomware, sql injection
- **Jövő és kutatási irányok** – kvantumkriptográfia, AI biztonság

Gyakorlati feladatok

A csatorna kihívásai (1)

IT megjelenése előtt

A futárt elfogják, lelövik, kicserélik az üzenetet

Informatika megjelenésével mi változott?

SEMMI

Csomagküldés – (nagyon titkos információ küldése) (1)

Küldő:

1) **Bőröndbe rakod a titkot**
2) **Gyártasz egy kulcsot a bőröndhöz**
3) **Bezárod a kulccsal a bőröndöt**
4) **A kulcsot egy számzáras „borítékba” rakod amit a vevőtől kaptál**
5) **A bőröndöt és a borítékot belerakod egy dobozba ami hungarocellel van kipárnázva (sérülések ellen)**

Csomagküldés – (nagyon titkos információ küldése) (2)

Vevő:

1) **Megkapja a dobozt**
2) **Ellenőrzi a sérüléseket és javítja (ha tudja)**
3) **A borítékot ki tudja nyitni mert ismeri a kódot**
4) **A borítékban lévő kulccsal kinyitja a bőröndöt**
5) **Elolvassa a titkos üzenetet**

Csomagküldés – IT oldalról

- **Küldő oldal (Te)**

- **Van egy adatod** (pl. egy .docx fájl)
- **Véletlenszerűen generálsz egy AES kulcsot** (olyan, mint egy zár a fájlodhoz)
- **AES-sel titkosítod a fájlt**
  → most már olvashatatlan, mint egy lakatolt bőrönd
- **RSA publikus kulccsal titkosítod az AES kulcsot**
  → csak a címzett tudja majd visszafejteni (ő a kulcs tulajdonosa)
- **Reed–Solomon hibajavító kódot generálsz a teljes csomagra**
  → így ha néhány byte sérül útközben, a vevő helyre tudja állítani
- **Elküldöd a következőket:**
  - AES-sel titkosított adat (a bőrönd)
  - RSA-val titkosított kulcs (a lakat kulcsa, titkos rekeszben)
  - Reed–Solomon ellenőrző kód (védőcsomagolás)

Vevő oldal (barátod)

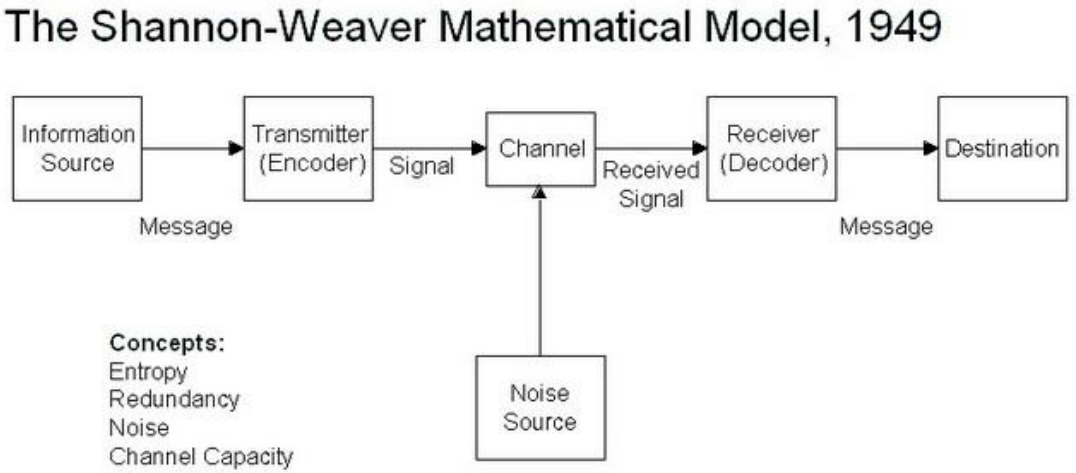
Megkapja az adatot + hibajavító kódokat
Reed–Solomon hibajavítással kijavítja az esetleges sérült byte-okat
RSA privát kulcsával visszafejti az AES kulcsot
AES kulccsal visszafejti az adatot
Megkapja az eredeti fájlt, teljesen helyreállítva és titoktartva

A csatorna kihívásai (2)

Információ átadás biztosítása minden téren

Adat sérülés megelőzése

Pl. Hamming kód



Hamming-kód (1)

Egyetlen hiba javítására képes kódolás

Motiváció: Ha a csatorna jó minőségű (pl. vezetékes összeköttetés), akkor a többszörös hibák generálásának valószínűsége kicsi. Ezért elegendő minden egyszeres hibát javítani, mert ezek fordulnak elő nagy valószínűséggel.

Több hibát észlel de nem tud javítani

Hamming-kód (2)

Hamming perfekt kódok: C(n,k)

$$n = 2^{n-k} - 1 \quad \Longleftrightarrow \quad \sum_{i=0}^{1} \binom{n}{i} = 2^{n-k}$$

Felépítésük:

1) a H paritás ellenörző mátrix oszlopait úgy választjuk meg, hogy mind különböző legyen és a csupa nulla oszlop ne szerepeljen köztük
2) meghatározzuk a generátormátrixot
3) megtervezzük az illesztő kapukat a szindróma dekódoláshoz
4) implementáljuk az egész sémát.

C(7,4) Hamming-kód (1)

$$n = 7, k = 4, tehát\ n + 1 = 8 = 2^{7-4}\ teljesül$$

A paritás ellenörző mátrix konstrukciója:

$$H = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

A generátormátrix:

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

A számolástól most eltekintünk, jegyzetben megtalálható

C(7,4) Hamming-kód (2)

- **Adatbitek**: d1, d2, d3, d4
- **Paritásbitek (ellenőrző bitek)**: p1, p2, p3

a paritásbiteket mindig a **2 hatványainak pozícióira** tesszük: 1, 2, 4, 8, stb

Elküldött csomag:

p1 p2 d1 p3 d2 d3 d4

C(7,4) Hamming-kód (2)

Paritásbitek kiszámítása:

$$P1 = D1 \oplus D2 \oplus D4$$
$$P2 = D1 \oplus D3 \oplus D4$$
$$P3 = D2 \oplus D3 \oplus D4$$

$$\oplus = \boldsymbol{XOR}$$

- **0 ⊕ 0 = 0**
- **1 ⊕ 0 = 1**
- **0 ⊕ 1 = 1**
- **1 ⊕ 1 = 0**

C(7,4) Hamming-kód példa (1)

Adat: 1 0 1 1

Pozíció	1	2	3	4	5	6	7
Bit	p1	p2	d1	p3	d2	d3	d4

paritásbitek:

$$P1 = D1 \oplus D2 \oplus D4 = 1 \oplus 0 \oplus 1 = 0$$
$$P2 = D1 \oplus D3 \oplus D4 = 1 \oplus 1 \oplus 1 = 1$$
$$P3 = D2 \oplus D3 \oplus D4 = 0 \oplus 1 \oplus 1 = 0$$

Végleges kód:

0 1 1 0 0 1 1

C(7,4) Hamming-kód példa (2)

Eredeti üzenet:

Pozíció	1	2	3	4	5	6	7
Bit	0	1	1	0	0	1	1

Hiba elhelyezése:

Pozíció	1	2	3	4	5	6	7
Bit	0	1	1	0	0	0	1

C(7,4) Hamming-kód példa (3)

Kapott üzenet:

Pozíció	1	2	3	4	5	6	7
Bit	0	1	1	0	0	0	1

Hiba detektálása:

$$\boldsymbol{P1 \oplus D1 \oplus D2 \oplus D4 = 0 \oplus 1 \oplus 0 \oplus 1 = 0}$$
$$\boldsymbol{P2 \oplus D1 \oplus D3 \oplus D4 = 1 \oplus 1 \oplus 0 \oplus 1 = 1}$$
$$\boldsymbol{P3 \oplus D2 \oplus D3 \oplus D4 = 0 \oplus 0 \oplus 0 \oplus 1 = 1}$$

Hibahely meghatározása:

P3 P2 P1 = 1 1 0 = 4 + 2 + 0 = 6 => a 6. pozíció a hibás fordítsuk meg

0 1 1 0 0 1 1

C(7,4) Hamming-kód példa (4)

Kapott üzenet 2 hibával:

Pozíció	1	2	3	4	5	6	7
Bit	0	1	1	0	1	0	1

Hiba detektálása:

$$\boldsymbol{P1 \oplus D1 \oplus D2 \oplus D4 = 0 \oplus 1 \oplus 1 \oplus 1 = 1}$$
$$\boldsymbol{P2 \oplus D1 \oplus D3 \oplus D4 = 1 \oplus 1 \oplus 0 \oplus 1 = 1}$$
$$\boldsymbol{P3 \oplus D2 \oplus D3 \oplus D4 = 0 \oplus 1 \oplus 0 \oplus 1 = 0}$$

Hibahely meghatározása:

P3 P2 P1 = 0 1 1 = 0 + 2 + 1 = 3 => a 3. pozíció a hibás fordítsuk meg

0 1 0 0 1 0 1 => rossz adat lesz

Hamming-kód variációk

Hamming-kód	Adatbitek (k)	Paritásbitek (r)	Teljes hossz (n)	Javítási képesség
(7,4)	4	3	7	1 bit javítható
(15,11)	11	4	15	1 bit javítható
(31,26)	26	5	31	1 bit javítható
(63,57)	57	6	63	1 bit javítható

A Hamming-kód egy fontos kiterjesztése a **SECDED**:
Single **E**rror **C**orrection **D**ouble **E**rror **D**etection
Ez úgy jön létre, hogy a Hamming-kódhoz még egy **globális paritásbitet** adunk hozzá (összes bitre), így:
Pl. **(8,4)** kód → az eredeti (7,4) Hamming + 1 extra paritásbit
Képes 2 bit hibát felismerni, és 1-et javítani

Alkalmazásuk
(7,4) – oktatás, egyszerű RAM védelem
(15,11) – memóriák, hibajavító modulok, kommunikációs protokollok (pl. űrkutatásban is használták)
SECDED – ECC RAM, processzorok, tárolók (BIOS, SSD)

Hamming-kód feladatok

Egy bináris csatorna bit hiba valószínűsége $P_b = 0.001$

1) Átküldünk egy 57 bit hosszú üzenetet a csatornán kódolás nélkül. Mennyi a blokk hiba valószínűsége (vagyis annak az esélye, hogy az üzenet hibásan érkezik meg)?

$$(1 - P_b)^{57} \approx 0.9446$$

blokk hiba valószínűsége

$$1 - (1 - P_b)^{57} \approx 0.0554$$

Hamming-kód feladatok

Egy bináris csatorna bit hiba valószínűsége $P_b = 0.001$

2) Átküldünk egy 57 bit hosszú üzenetet a csatornán C(63,57) Hamming-kóddal kódolva. Mennyi a blokk hiba valószínűsége (vagyis annak az esélye, hogy hibás az üzenet dekódolása)?

Hamming kód használata esetén a dekódolás helyes lesz, ha 0 vagy 1 hiba van a vett kódszóban. A helyes dekódolás valószínűsége

Hamming-kód feladatok

Egy bináris csatorna bit hiba valószínűsége $P_b = 0.001$

2) Átküldünk egy 57 bit hosszú üzenetet a csatornán C(63,57) Hamming-kóddal kódolva. Mennyi a blokk hiba valószínűsége (vagyis annak az esélye, hogy hibás az üzenet dekódolása)?

$$(1 - P_b)^{63} + 63(1 - P_b)^{62} * P_b \approx 0.99812$$

blokk hiba valószínűsége

$$1 - (1 - P_b)^{63} + 63(1 - P_b)^{62} * P_b \approx 0.00188$$

Hamming-kód feladatok

C(63,57) Hamming kód használatával csökkentettük a blokk hiba valószínűségét 0.0554-ről 0.00188-re

Ennek ára pedig az, hogy a kód ráta 57/63, vagyis az effektív csatorna kapacitás az eredeti kapacitás 57/63 részére csökkent.

Hamming távolság

két azonos hosszúságú bitlánc között **az eltérő bitek száma**

A: 1011011

B: 1001001

Hamming távolság d = 2

$$Maximálisan\ javítható\ hiba:\ t = \frac{d-1}{2}$$

$$Maximálisan\ detektálható\ hiba: d-1$$

Milyen kódolást használjak?

- Mekkora az adatblokkom (hány bit)?
- Hány hibát akarok biztosan kijavítani vagy legalább észrevenni?
- Mekkora a maximális redundancia, amit el tudok viselni?
- Mekkora az átviteli vagy tárolási hibák valószínűsége?
- Kell-e a hibajavítás valós időben?

Reed-Solomon (1)

- nem bitenként, hanem **szimbólumonként** dolgozik (pl. 1 szimbólum = 1 byte = 8 bit)
- képes egyszerre **több hiba javítására** – akár **több egymást követő hibás szimbólumot** is
- **Javít és detektál is**, anélkül hogy újraküldésre lenne szükség

Reed-Solomon (2)

k = eredeti adat szimbólumok száma

n = teljes kódolt szimbólumok száma

n − k = redundáns szimbólumok száma (hibajavításra szolgál)

t szimbólum javítására képes

$$t = \frac{n-k}{2}$$

RS(255,223)

223 byte adat, 32 byte hibajavító kód = 255 byte adat küldve

16 byte hibás adat javítása lehetséges

Reed-Solomon vs. Hamming kód

	Hamming-kód	Reed–Solomon-kód
Bit vagy szimbólum	Biteken dolgozik	Szimbólumokon (ált. byte)
Hibatípus	Egyedi bit	Szimbólumhibák / blokkok
Javítási képesség	1 bit	t szimbólumhibát is javít
Használat	RAM, egyszerű rendszerek	QR-kód, DVD, űrtávközlés, mobil
Rugalmasság	Alacsony	Magas (paraméterezhető)

Reed-Solomon működése

– Mi lenne, ha egy QR-kódban 15 négyzet elmaszatolódna – szerintetek még olvasható maradna?

Igen, mert RS kódoslást használ

Reed-Solomon kódolása (1)

Adat = [15, 22, 7]

$$P(x)=a^0+a_1x^1+a_2x^2$$

$$P(x) = 15 + 22x + 7x^2$$

(lehet fordítva is)

Kiértékeljük a polinomot több x értékre pl. X = 1,2,3,4,5,6

x	P(x) = 15 + 22x + 7x²
1	15 + 22×1 + 7×1 = 44
2	15 + 44 + 28 = 87
3	15 + 66 + 63 = 144
4	15 + 88 + 112 = 215
5	15 + 110 + 175 = 300
6	15 + 132 + 252 = 399

Reed-Solomon kódolása (2)

x	P(x)	P(x) mod 256
1	44	44
2	87	87
3	144	144
4	215	215
5	300	**44**
6	399	**143**

Mivel az RS egy véges testben dolgozik ezért (pl GF(256) – 0-255 közötti érték) minden eredmény **mod(256)-**al kell venni

Reed-Solomon kódolása (3)

Elküldött adat = [44, 87, 144, 215, 44, 143]

- A vevő tudja, hogy a Reed–Solomon kódolás k = 3 adatpontból állt
- Ezért a polinom **legfeljebb fokszáma = k – 1 = 2**

3 ép pont elég lenne az egyértelmű visszafejtéshez
Ha van 4–5–6 pont, az segít az esetleges hibák javításában

$$P(x)=a^0+a_1x^1+a_2x^2$$

Jelen esetben 6 egyenletből pl Lagrange interpolációval visszafejti, hogy

$$P(x) = 15 + 22x + 7x^2$$

Egy ilyen 6-os kódolással 2 hiba egyértelműen javítható

Reed-Solomon hatékonysága

Hibajavítási cél	Adatszimbólumok	Hibajavító szimbólumok	Összesen (n)	RS(n,k)
1 hiba javítása	3	2	5	RS(5,3)
2 hiba javítása	4	4	8	RS(8,4)
3 hiba javítása	10	6	16	RS(16,10)
16 hiba javítása	223	32	255	RS(255,223)

Ha valakit érdekel több infó: https://www.pclviewer.com/rs2/calculator.html

Reed-Solomon használata

Ma is használt eljárás leginkább ott ahol

- Fizikai használatból adódó sérülések léphetnek fel
- Nem lehet az adatot újrakülden
- Részben streaming

Miről lesz szó

- ✅ **Információelmélet alapfogalmak** – Shannon, entropia, tömörítés
- ✅ **Klasszikus kriptográfia** – Caesar, Vigenère, Enigma
- ✅ **Hibajavítás és csatornamodellek** – Hamming-kód, Reed–Solomon
- **Modern kriptográfia** – AES, RSA, hash-ek, digitális aláírás
- **Hálózati biztonság** – SSL, TLS, VPN, MITM támadások
- **Adatvédelem, GDPR** – anonimizálás, nyomkövetés, metaadat
- **Támadások és védekezés** – brute force, phishing, ransomware, sql injection
- **Jövő és kutatási irányok** – kvantumkriptográfia, AI biztonság

Gyakorlati feladatok

Átvitel költségének csökkentése

Átvitel során törekedni kell a redundancia és az adatmennyiség minimalizálásra.
Ez tömörítéssel érhető el a legjobban.

Miért van szükség tömörítésre?

 **Csökkenti az adatmennyiséget** → kevesebb tárhely, kisebb fájl
 **Gyorsabb átvitel** → kisebb adat = gyorsabb hálózati továbbítás
 **Gazdaságosabb tárolás** → SSD/HDD, memóriakezelés optimalizálása
 **Titkosítás előtt célszerű** → a tömörített adat hatékonyabban titkosítható
 **Bizonyos formátumok alapja** → pl. PNG, ZIP, MP3 részben

Huffman kódolás

TXT állomány tartalma: AABACAA

File mérete 7 karakter x 8 bit = 56 bit

A A B A C A A
↓ ↓ ↓ ↓ ↓ ↓ ↓
0 0 10 0 11 0 0 → 0010001100

10 bit + fa = kb. 20-30 bit

Miért van szükség tömörítésre

Miért van szükség tömörítésre

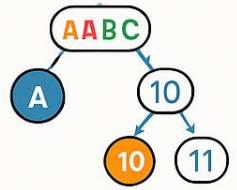
- Adatmennyiség csökkentése
- Gyorsabb átvitel
- Költséghatékony tárkosítás
- Bizonyos formátumok alapja

Huffman-kódolás

- Elemzi a karakterek gyakoriságát
- Gyakori karakter → rövidebb kód
- Bináris fa alapján kódol
- Veszteségmentes

Egyszerű példa

Karakter	Gyakoriság	Kód
A	5	0
B	1	10
C	1	11



A A B A C A A . . > 0 0 1 0 0 1 1 0 0

A 0 10 10 0 11 11 00110011010

Hol használják?

Huffman kódolás algoritmusa (1)

Kódolandó szöveg : ABCBBAAAAAADDZBB

Karakter	Gyakoriság	S orrend	Bináris kód
A	7	0	0
B	5	1	1
C	1	2	10
D	2	3	11
Z	1	4	100
	16		

Első három karakter kódolva: 0 1 1 0

Ez lehet visszafejtve: A B C de lehet A B B A is !!!

de lehet A B B A is !!!

? Hogyan Javíthatnánk ?

Huffman kódolás algoritmusa (2)

Javított kód tábla: a bináris kódot a leghosszabbhoz igazítjuk!

Karakter	Gyakoriság	S orrend	Bináris kód
A	7	0	000
B	5	1	001
C	1	2	010
D	2	3	011
Z	1	4	100
	16	5	101

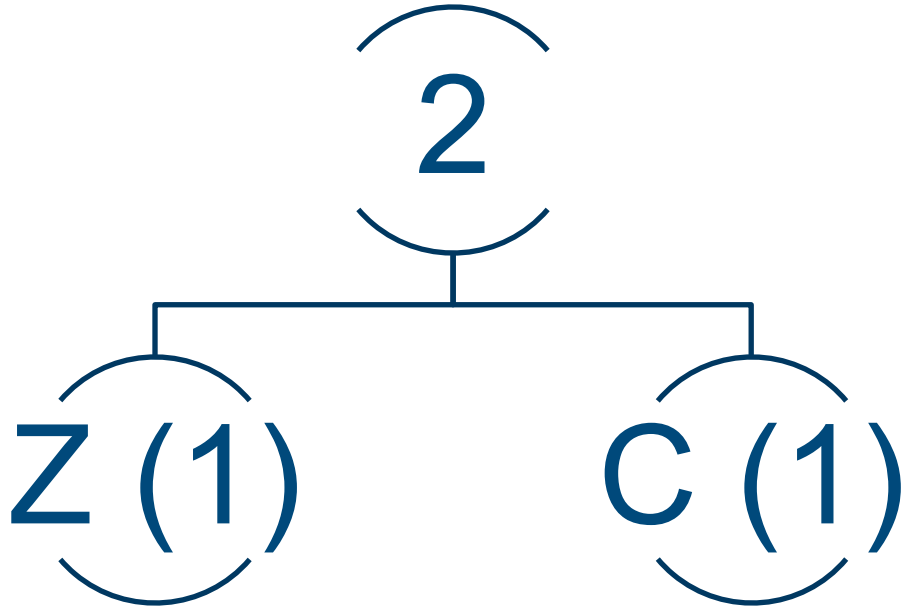
Nem hatékony mert minden karakterre 3 bit jut ebben az esetben
Hosszabb szöveg esetében pedig még több

Jelen esetben 16 x 3 bit = 48 bit de ez még nem tömörítés csak értelmezési szótár rövidítés

Huffman kódolás algoritmusa (3)

Karakter	Gyakoriság
A	7
B	5
C	1
D	2
Z	1
	16

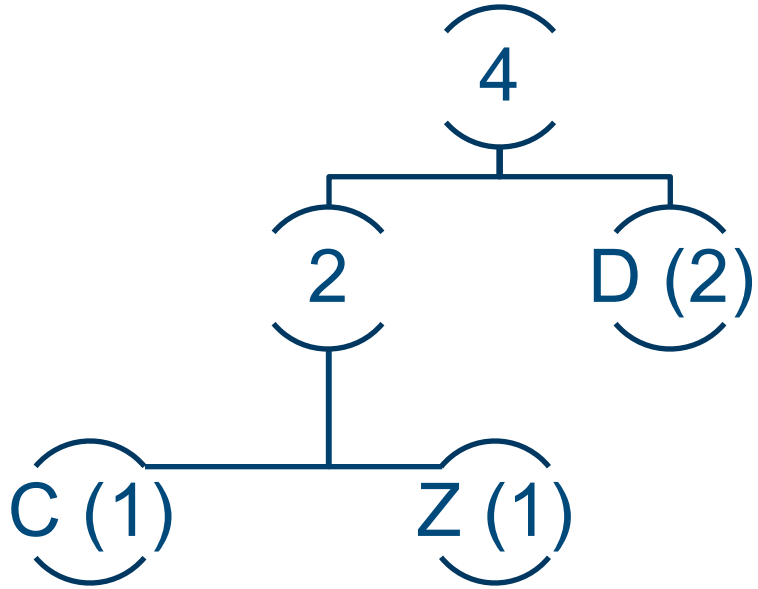
Építsünk bináris fát
1) Vegyük a két legalacsonyabb gyakoriságú elemet
2) A fában legyenek egymás szomszédjai



Huffman kódolás algoritmusa (4)

Karakter	Gyakoriság
A	7
B	5
C	1
D	2
Z	1
	16

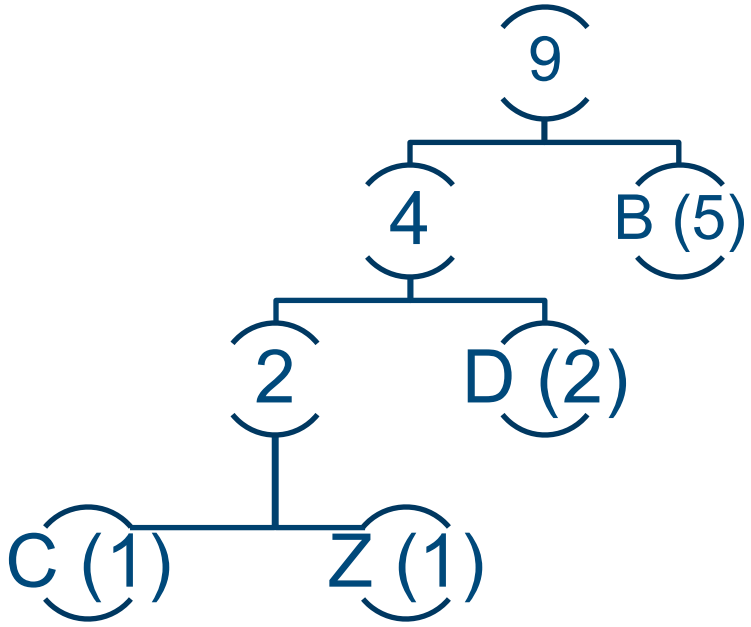
Építsünk bináris fát
1) Vegyük a két legalacsonyabb gyakoriságú elemet
2) A fában legyenek egymás szomszédjai
3) Vegyük a sorban következő gyakoriságot (D) és legyen a szülő szomszédja



Huffman kódolás algoritmusa (5)

Karakter	Gyakoriság
A	7
B	5
C	1
D	2
Z	1
	16

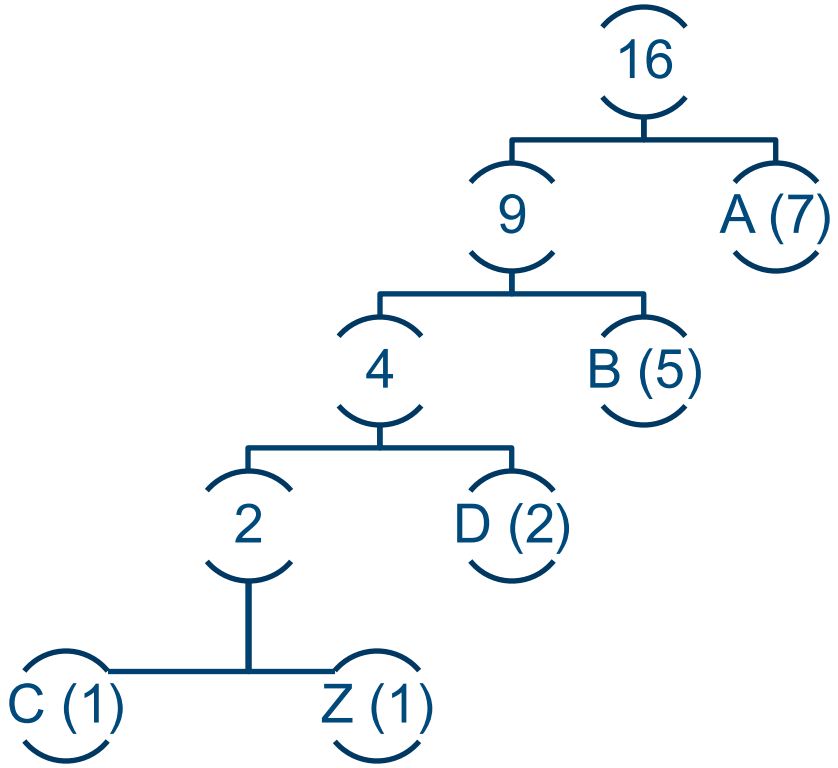
Folytatjuk a logika szerint



Huffman kódolás algoritmusa (6)

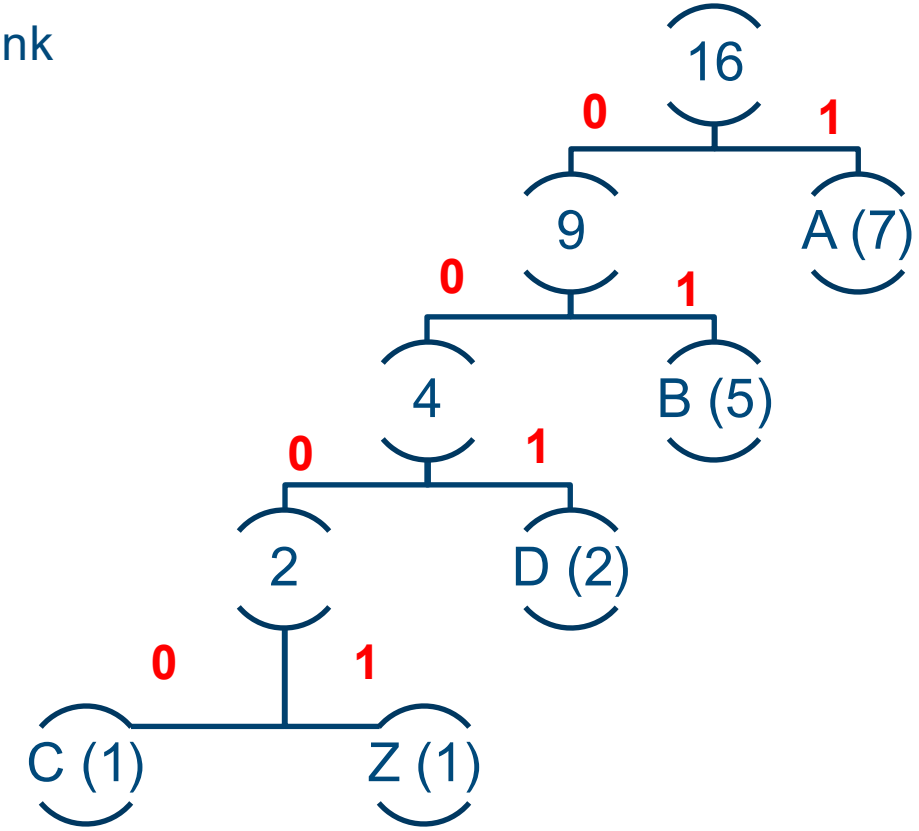
Karakter	Gyakoriság
A	7
B	5
C	1
D	2
Z	1
	16

Folytatjuk a logika szerint



Huffman kódolás algoritmusa (7)

Bal oldalra 0-t írunk



A = 1
B = 01
D = 001
C = 0000
Z = 0001

Huffman kódolás algoritmusa (8)

Javított kód tábla: a bináris kódot a leghosszabbhoz igazítjuk!

Karakter	Gyakoriság	Bináris kód	Méret
A	7	1	7 bit
B	5	01	10 bit
C	1	0000	4 bit
D	2	001	6 bit
Z	1	0001	4 bit
	16	5	31 bit

Tömörítés példa (1)

Elemér elment a boltba elemért.

31 karakter:

- ASCII – 7 bit
- UTF-8 / 16 / 32 – 8, 16 , 32 bit.

8 bittel számolva = 248 bit adat

Karakter	Előfordulás
E	1
l	4
e	5
m	3
é	2
r	2
(szóköz)	4
n	1
t	3
a	2
b	2
o	1
.	1
	31

Tömörítés példa (2)

Rendezzük csökkenő sorba

Karakter	Előfordulás
e	5
l	4
(szóköz)	4
m	3
t	3
é	2
r	2
a	2
b	2
E	1
n	1
o	1
.	1
	31

Tömörítés példa (3) – fa készítés

Karakter	Előfordulás
e	5
l	4
(szóköz)	4
m	3
t	3
é	2
r	2
a	2
b	2
E	1
n	1
o	1
.	1
	31

Titkosítások - AES (Advanced Encryption Standard)

Biztonságos, gyors és hatékony titkosítás biztosítása mind szoftveres,

mind hardveres környezetekben.

Tulajdonság	Érték
Titkosítás típusa	Szimmetrikus (ugyanazzal a kulccsal titkosít és dekódol)
Blokkméret	128 bit (16 bájt)
Kulcshossz	128, 192 vagy 256 bit
Rounds (körök száma)	10 (128-bit), 12 (192-bit), 14 (256-bit)

Titkosítások - AES – működési elv

Milyen adatot titkosít az AES?

- **Blokkalapú**: minden 128 bit (16 bájt) hosszú adatblokkot külön kezel.
- Ha az üzenet nem osztható pontosan 16-tal, **padding**-et használunk (pl. PKCS#7).
- Nagyobb adatokhoz külön **üzemmódokat (módokat)** használunk

A titkosítás **teljes biztonsága** azon múlik, hogy **a kulcs titokban maradjon**

Kulcsok kezelése lehet:

- Helyi fájlban (nem ajánlott)
- Hardveres tárolás (TPM, HSM)
- Kulcskezelő rendszerek (pl. AWS KMS, Azure Key Vault)

Titkosítások - AES – működési elv (128 bit)

AES 128 esetén **10 körös transzformáció történik** egy 16 bájtos adatblokkra. A fő lépések minden körben:

🔷 **0. Előkészítés (kulcs hozzáadása – AddRoundKey)**

🔁 **1–9. kör:**
1.SubBytes – minden bájtot lecserél egy előre definiált S-box alapján (nemlineáris helyettesítés)
2.ShiftRows – a sorok balra eltolása (elkeverés)
3.MixColumns – oszlopok mátrixszorzása (diffúzió növelése)
4.AddRoundKey – XOR a körhöz tartozó kulccsal

🔚 **10. kör (végső kör):**
•Ugyanaz, **de nincs MixColumns**

Titkosítások - AES – előnyei

- **Széles körben elterjedt**: VPN-ek, HTTPS, fájltitkosítás, diszktitkosítás
- **Gyors hardveres támogatás** (pl. Intel AES-NI)
- **Erős biztonság** – nincs ismert gyakorlatban működő támadás (pl. brute-force ellenálló)
- **Szimmetrikus titkosítás** – egyszerűbb és gyorsabb, mint az RSA

Titkosítások - AES – használata

Alkalmazás	Szerepe
HTTPS / TLS	Böngészős adatkapcsolat titkosítása
VPN	Internetes alagút védelme
Wi-Fi WPA2/WPA3	Hálózati adatvédelem
ZIP/7z/GPG fájlok	Fájlok titkosítása
BitLocker, VeraCrypt	Teljes diszk titkosítása
Mobil OS (iOS/Android)	Adattitkosítás eszközszinten

Titkosítások - AES – példa

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad
from Crypto.Random import get_random_bytes

# Bemeneti adatok
data = b"Hello AES!"                # 9 bájt → padding szükséges
key = get_random_bytes(16)          # 128 bites véletlen kulcs
iv = get_random_bytes(16)           # Inicializációs vektor

# 2 Titkosítás
cipher = AES.new(key, AES.MODE_CBC, iv)
ciphertext = cipher.encrypt(pad(data, AES.block_size))

print("Titkosított adat (hex):", ciphertext.hex())

# 3 Visszafejtés
decipher = AES.new(key, AES.MODE_CBC, iv)
plaintext = unpad(decipher.decrypt(ciphertext), AES.block_size)

print("Visszafejtett szöveg:", plaintext.decode())
```

Kimenet:

Titkosított adat (hex):
1a7e6c234a3f3e9960a59c0edc7dd9e9

Visszafejtett szöveg: Hello AES!

Miről lesz szó

- ✅ **Információelmélet alapfogalmak** – Shannon, entropia, tömörítés
- ✅ **Klasszikus kriptográfia** – Caesar, Vigenère, Enigma
- ✅ **Hibajavítás és csatornamodellek** – Hamming-kód, Reed–Solomon
- **Modern kriptográfia** – ✅ AES, RSA, hash-ek, digitális aláírás
- **Hálózati biztonság** – SSL, TLS, VPN, MITM támadások
- **Adatvédelem, GDPR** – anonimizálás, nyomkövetés, metaadat
- **Támadások és védekezés** – brute force, phishing, ransomware, sql injection
- **Jövő és kutatási irányok** – kvantumkriptográfia, AI biztonság

Gyakorlati feladatok

Egykulcsos kódolás problémája (1)

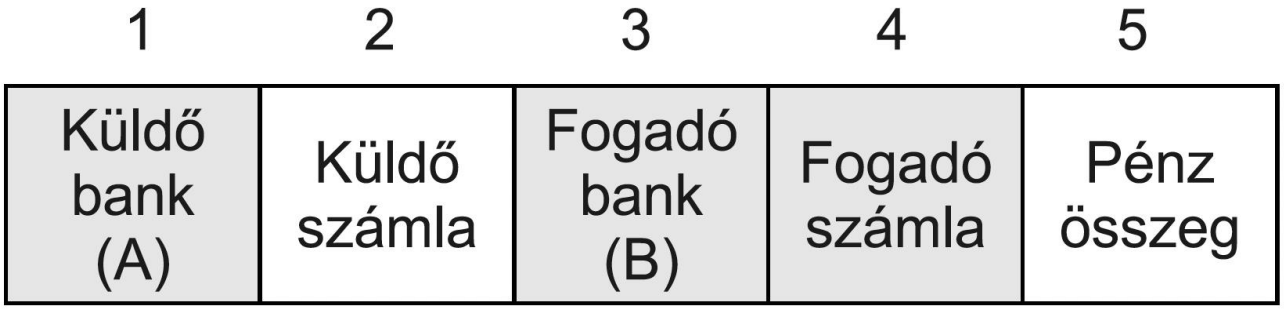
Az A bankból pénzt utalunk B bankba, az utalást az alábbi üzenettel végezzük el:

1	2	3	4	5
Küldő bank (A)	Küldő számla	Fogadó bank (B)	Fogadó számla	Pénz összeg

Tegyük fel, hogy

- A blokkok egyenként 16 bájtosak, és AES-el kódoljuk őket
- A bankok egy hétig minden üzenethez ugyanazt a 256 bites AES kulcsot használják
- Mi viszont nagyon gyorsan meg szeretnénk gazdagodni

Egykulcsos kódolás problémája (2)

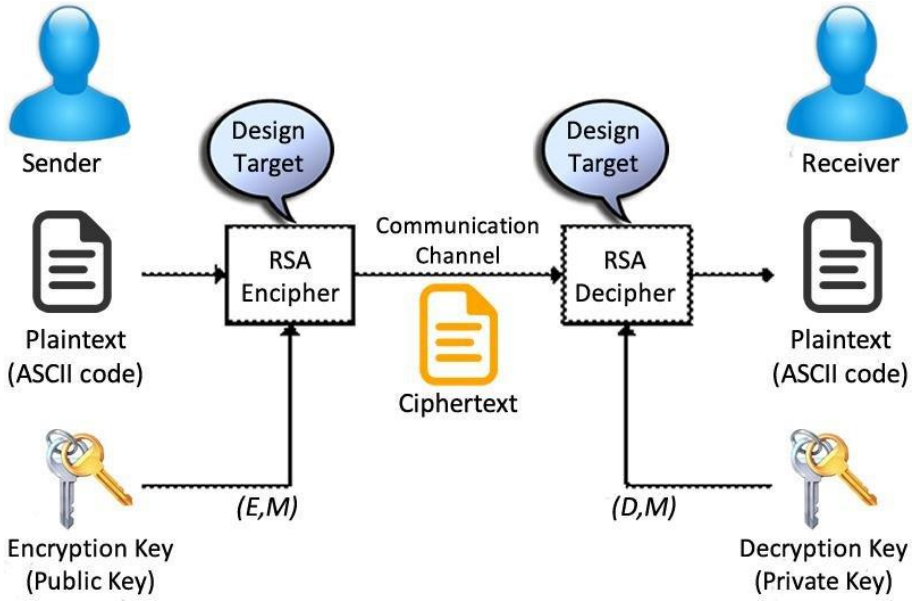


1) Nyissunk számlát A és B banknál is
2) Utaljunk 1$-t A számláról B számlára többször
3) Észrevesszük, hogy pl. 5 blokk van.
4) Lementjük az 1., 3. és 4. blokk tartalmát
5) Elkapunk minden más üzenetet is
6) Ha az elfogott üzenetek 1., 3. blokkja megegyezik azzal amit mi eltároltunk akkor a 4. blokkot kicseréljük arra amit mi eltároltunk
7) Ezt megismételjük párszor de valódi utalás nélkül

Aszimmetrikus kulcsú titkosítás

Kettős kulcs segítségével:

- Egyik kulcs a titkosításhoz (public key)
- Másik kulcs a dekodolásához (private key)



Aszimmetrikus kulcsú titkosítás

A kódoláshoz és dekódoláshoz használt kulcsok eltérőek

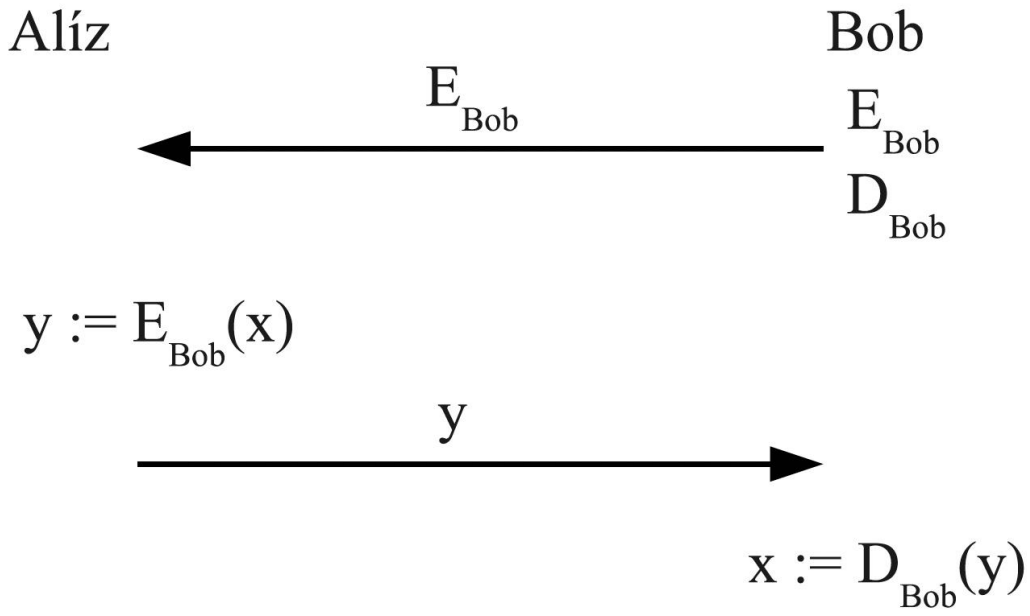
Mindenkinél van egy **privát** és **publikus** kulcs

Nagyságrendekkel lassabb mint a szimmetrikus kódolás

Aszimmetrikus kulcsú titkosítás (2)

Alíz szeretne üzenetet (x) küldeni Bobnak:

- Bob nyilvános kulcsa : E_b
- Bob privát kulcsa : D_b



Egy kis matek

LNKO – Legnagyobb közös osztó

A és B egész számok legnagyobb közös osztója az a legnagyobb szám, amely osztója A-nak és B-nek is

Lnko(84, 30) = ?

84 = 2 * 2 * 3 * 7

30 = 2 * 3 * 5

A közös osztó: 2 * 3 = 6

A prímtényezős felbontás nagy számoknál lassú és nehéz

Euklideszi algoritmus

A és B egész számok (A > B)

Iteratív módon minden ciklusban:

1. R = A mod B
2. Ha R == 0 akkor LNKO = B
3. A = B
4. B = R

A = 84
B = 30

A = 84, B = 30

1. R = 84 mod 30 = 24
   - A = 30, B = 24
2. R = 30 mod 24 = 6
   - A = 24, B = 6
3. R = 24 mod 6 = 0
   - Vége, legnagyobb közös osztó = 6

Multiplikatív inverz

Ha van két szám (**A** és **B**), akkor **B az A inverze modulo M szerint**, ha:

$$AB \equiv 1 \pmod{M}$$

Példa:
Tegyük fel, hogy:

$$A = 3$$
$$M = 7$$

Akkor keressük azt a **B** számot, amire igaz:

$$3B \equiv 1 \pmod{7}$$

Próbálgatással:

$$3 \times 5 = 15 \equiv 1 \pmod{7}$$

$\rightarrow$ Tehát $B = 5$, és így **5 a 3 inverze modulo 7 szerint.**

Multiplikatív inverz

Legyen A = 23, M = 120

Mi az A szám multiplkiatív inverze modulo 120 esetén?

Mi a valódi kérdés?
Mivel kell megszorozzam a 23-at, hogy az eredményt 120-al osztva 1 maradékot kapjak?

B = 47
A*B = 1081
1081 mod 120 = 1

Tehát 23 multiplikatív inverze modulo 120-ban 47

Multiplikatív inverz meghatározása (1)

Cél:
Szeretnénk megtalálni egy olyan számot (**x**), amely kielégíti az alábbi egyenletet:

$$\boldsymbol{ax \equiv 1 \pmod m}$$

Kiterjesztett euklideszi azonosság
A klasszikus Euklideszi algoritmus az **a** és **b** számok legnagyobb közös osztóját (lnko) adja meg.

A **kiterjesztett euklideszi algoritmus** ennél többet tud: megmondja azokat az egész számokat (**x** és **y**), amelyekre igaz, hogy:

$$ax + by = lnko(a, b)$$

Ezt nevezzük **Bézout-azonosságnak**.

(két egész szám, a és b legnagyobb közös osztója (lnko), előáll **a** és **b** egész együtthatós lineáris kombinációjaként)

Multiplikatív inverz meghatározása (2)

Minek kell teljesülnie ahhoz hogy egy számnak legyen inverze modulo m-ben?

Ahhoz, hogy az **a számnak legyen inverze modulo m**, az kell, hogy:

$$\text{lnko}(a, m) = 1$$

Ez azt jelenti, hogy **a és m relatív prímek**. (Nincs közös osztójuk az 1-en kívül.)

Multiplikatív inverz meghatározása (3)

Kapcsoljuk össze a kettőt:
Az inverz definíciója:

$$ax \equiv 1 \pmod{m}$$

Ez átalakítható **normál egyenletté** úgy, hogy a kongruencia jelentése az, hogy az egyik szám **m-mel osztva** ugyanazt az eredményt adja, mint a másik, vagyis:

$$ax - 1 = qm \text{ (q valamilyen egesz szam)}$$

Ebből:

$$ax - qm = 1$$

Ez egy **egyenlet két ismeretlennel (x és q)**, és ez pont olyan alakú, mint a Bézout-azonosság:

$$ax + (-m)q = 1$$

ez alapján a kiterjesztett euklideszi algoritmus segítségével megtalálható az **x** (az inverz), mivel:

$$\text{lnko}(a, m) = 1 \Rightarrow \text{letezik megoldas}$$

Multiplikatív inverz meghatározása - példa (1)

A = 120
B = 23
Lnko = ?

Keressük meg az **lnko(120, 23)** értéket, és vele együtt azokat az **egész számokat** x és y, melyekre igaz:

$$120x + 23y = 1$$

Ez különösen akkor hasznos, ha pl. 23 inverzét keressük modulo 120 szerint (RSA algoritmushoz)

Multiplikatív inverz meghatározása - példa (2)

Osztásos algoritmus – euklideszi algoritmus:

Osztandó	Osztó	Hányados (q)	Maradék (r)
120	23	5	5
23	5	4	3
5	3	1	2
3	2	1	1
2	1	2	0

Mit látunk itt?

Az utolsó **nem nulla maradék** az **lnko(120, 23) = 1**
Ez azt jelenti, hogy **létezik** inverz.

Multiplikatív inverz meghatározása - példa (3)

Lineáris kombinációk meghatározása

A cél, hogy minden r_i-t ki tudjunk fejezni az a = 120 és b = 23 segítségével:

$$r_i = a \cdot x_i + b \cdot y_i$$

Egy rekurzív képlet segítségével számoljuk a x_i és y_i értékeket:

$$x_i = x_{i-2} - q_i \cdot x_{i-1}$$
$$y_i = y_{i-2} - q_i \cdot y_{i-1}$$

Kezdőértékek:

$$x_1 = 1,\ y_1 = 0$$
$$x_2 = 0,\ y_2 = 1$$

Multiplikatív inverz meghatározása - példa (4)

i	q_i	r_i	x_i	y_i	Egyenlet
1	–	120	1	0	120 = 120×1 + 23×0
2	–	23	0	1	23 = 120×0 + 23×1

Az első két sor adott a kezdőértékek miatt:

$$x_1 = 1,\ y_1 = 0$$
$$x_2 = 0,\ y_2 = 1$$

Multiplikatív inverz meghatározása - példa (5)

i	q_i	r_i	x_i	y_i	Egyenlet
1	–	120	1	0	120 = 120×1 + 23×0
2	–	23	0	1	23 = 120×0 + 23×1
3	5	5	1	-5	5 = 120×1 + 23×(-5)

A harmadik sorhoz (i = 3) az alábbi képleteket használjuk:

$$x_3 = x_1 - q_3 \cdot x_2$$
$$y_3 = y_1 - q_3 \cdot y_2$$

Tudjuk, hogy:

$q_3 = 5$ (euklideszi táblázat 3. sora)

$$x_1 = 1,\ x_2 = 0$$
$$y_1 = 0,\ y_2 = 1$$

Multiplikatív inverz meghatározása - példa (6)

i	q_i	r_i	x_i	y_i	Egyenlet
1	–	120	1	0	120 = 120×1 + 23×0
2	–	23	0	1	23 = 120×0 + 23×1
3	5	5	1	-5	5 = 120×1 + 23×(-5)
4	4	3	-4	21	3 = 120×(-4) + 23×21
5	1	2	5	-26	2 = 120×5 + 23×(-26)
6	1	1	-9	47	1 = 120×(-9) + 23×47

$x_1 = 1,\ y_1 = 0$

$x_2 = 0,\ y_2 = 1$

Multiplikatív inverz meghatározása - példa (7)

A **1** értéknél azt kaptuk:

$$\mathbf{1 = 120 \cdot (-9) + 23 \cdot 47}$$

Vagyis:
x = -9,
y = 47

$$\text{lnko}(a, m) = 1$$
$$ax + by = \textit{lnko}(a, b)$$

A cél az volt, hogy megtaláljuk a
23 moduló 120 szerinti inverzét.

Mivel: $23 \cdot 47 \equiv 1 \pmod{120}$
ezért: **23^{-1} mod 120 = 47**

tehát 47 az **inverze** 23-nak modulo 120 szerint.

Euler-függvény (1)

Relatív prímek: A és B egész számok relatív prímek, ha a lnko(A,B) = 1

Az **Euler-függvény φ(n)** megadja, hogy **hány darab olyan pozitív egész szám van, ami kisebb, mint n és relatív prím vele** (vagyis az **lnko(n, a) = 1**).

$$\varphi(n)=|\{ a:\text{lnko}(n,a)=1 \text{ és } a<n \}|$$

***ϕ*(9):**
Nézzük meg a 9-nél kisebb számokat:
1, 2, 3, 4, 5, 6, 7, 8
Vizsgáljuk meg, melyek **relatív prímek** a 9-cel (vagyis lnko = 1):

lnko(9,1) = 1 ✅
lnko(9,2) = 1 ✅
lnko(9,3) = 3 ❌
lnko(9,4) = 1 ✅
lnko(9,5) = 1 ✅
lnko(9,6) = 3 ❌
lnko(9,7) = 1 ✅
lnko(9,8) = 1 ✅

Ezek relatív prímek: 1, 2, 4, 5, 7, 8 → összesen **6 darab**
Tehát:

$$\varphi(9) = 6$$

Euler-függvény (2)

Mi történik ha **n prím szám**?

$$\varphi(p)=p-1$$

Példa φ(240)=?

Prímtényezős felbontás: $240 = 2^4 \cdot 3^1 \cdot 5^1$

Euler-függvény képlete prímtényezőkre: $\varphi(p^e) = p^e - p^{e-1} = \boldsymbol{p^e\left(1-\frac{1}{p}\right)}$

Általános formában: $$\varphi(m) = \prod_{i=1}^{n}(p_i^{e_i} - p_i^{e_i-1})$$

Euler-függvény (3)

Példa φ(240)=?

Prímtényezős felbontás: $240 = 2^4 \cdot 3^1 \cdot 5^1$

Euler-függvény képlete prímtényezőkre:

$$\boldsymbol{\varphi(240) = \left(2^4 - 2^3\right) \times \left(3^1 - 3^0\right) \times \left(5^1 - 5^0\right) = 8 \times 2 \times 4 = 64}$$

240-nél 64 db kisebb olyan szám van amely relatív prím 240-el.

A kis Fermat - tétel

Legyen **a egész** és **p prím** szám, ekkor igaz a következő állítás:

$$a^p \equiv a \ (mod\ p)$$

Ha pedig $a \not\equiv 0 \mod p$ akkor**:**

$$a^{p-1} \equiv 1 \ (mod\ p)$$

Mit jelent ez?

Ez azt mondja ki, hogy **prím modulus** esetén az a szám hatványai **mod p** szerint nagyon **jó viselkedésűek**:

- Az első állítás: ha bármilyen **a**-t fogsz és **p**-edik hatványra emeled, majd veszed modulo p, akkor **visszakapod a-t**.
- A második állítás: ha **a** nem osztható **p**-vel, akkor a a^{p-1} hatványra ≡ 1 (mod p) esetén.

A kis Fermat – tétel példa(1)

Példa 1: $a = 34,\ p = 37$

Ez a kis Fermat-tétel 1. alakja szerint:

$$34^{37} \equiv 34 \pmod{37}$$

A kimenet:

$$34^{37} = 46208293553660808371261784090350221471870358813721042944$$

$$46208293553660808371261784090350221471870358813721042944 \bmod 37 = 34$$

Miért működik?

Mert **37 prímszám**, tehát a kis Fermat-tétel szerint:

$$a^p \equiv a \pmod{p} \Rightarrow 34^{37} \equiv 34 \pmod{37}$$

Mire volt ez a kis matek jó?

1976-ban Ron **R**ivest, Adi **S**hamir és Len **A**dleman fejlesztették ki

RSA

A módszer azt használja ki, hogy egy nagy szám prímtényezőkre bontására nem ismert gyors algoritmus

Lépései:

1. Kulcsválasztás
2. Kódolás
3. Dekódolás

RSA - Lépései

Készítünk egy nyilvános és titkos kulcsot

1. választunk két nagyon nagy prím számot *p* és *q*. $p \neq q$
   Titkosak és nehéz legyen kitalálni őket.

2. Modulus meghatározása $N = p \times q$
   Euler függvény segítségével meghatározzuk $\boldsymbol{\varphi(N) = (p-1)\cdot(q-1)}$

3. Választunk egy $\boldsymbol{e}$ számot $\boldsymbol{1 < e < \varphi(N)}$ között és $\boldsymbol{lnko(e, \varphi(N)) = 1}$
   Ez lesz a **nyilvános kulcs** kitevője

4. **Privát kulcs** komponens **d** meghatározása: $\boldsymbol{e \times d \equiv 1 \; mod \; \varphi(N)}$ (euklideszi alg.)

Egy üzenetet (x) így **titkosítunk**:

$$y = x^e \mod N$$

x: a titkosítandó szám (pl. ASCII)
e, N: a **nyilvános kulcs**
y: a **titkosított üzenet**

A titkosított y értéket így lehet **visszafejteni**:

$$x = y^d \mod N$$

d: a **privát kulcs**, amit csak a tulajdonos ismer
N: ugyanaz a modulus
x: a visszafejtett eredeti üzenet

Mitől lesz biztonságos?

Egy üzenetet (x) így **titkosítunk**:

$$y = x^e \mod N$$

x: a titkosítandó szám (pl. ASCII)
e, N: a **nyilvános kulcs**
y: a **titkosított üzenet**

A titkosított y értéket így lehet **visszafejteni**:

$$x = y^d \mod N$$

d: a **privát kulcs**, amit csak a tulajdonos ismer
N: ugyanaz a modulus
x: a visszafejtett eredeti üzenet

A matematikai **biztonság alapja**:
A számítás azon alapul, hogy:

$$(x^e)^d \equiv x \pmod N$$

Ez az Euler−tételen, pontosabban a **kis Fermat−tételen** alapul

A támadó:

- Ismeri a nyilvános kulcsot: `(N, e)`
- **DE nem ismeri `d`-t**
- Hogy `d`-t kiszámolja, **tudnia kéne φ(N)` értékét**
- Viszont φ(N) csak akkor számolható, ha ismerjük `p`-t és `q`-t
- Ez a szám **nagyon nehezen faktorizálható**, ha elég nagy számokat választunk (pl. 1024–4096 bit)

RSA példa

Kulcsgenerálás

p = 11, q = 13

N = 11 * 13 = 143

φ(N) = (11-1)(13-1) = 120

Választunk e = 7 (mert lnko(7,120) = 1)

Megkeressük d-t, hogy:

$$7 \cdot d \equiv 1 \pmod{120} \Rightarrow d = 103$$

Nyilvános kulcs: (143, 7)

Privát kulcs: 103

Titkosítsuk be az X = 9-et

$$y = 9^7 \, mod \, 143 = 48$$

Az titkosított üzenet 48 lesz

Fejtsük vissza a 48-at:

$$x = 48^{103} \, mod \, 143 = 9$$

A kapott visszafejtett üzenet = 9

Egy életszerűbb példa

$p = 14155210439216370810694657014 97$

$q = 17182746842346720870110327533 31$

$N =$
$243225397477198435382244489307218274890236227899958127843650 7$

$\varphi(N) =$
$243225397477198435382244489306904895317420596983150077998168 0$

$e = 7$

$d =$
$138985941415541963075568279603945654467097483990371473141810 3$

Titkosítsuk az $x = 74628364326326832684327 32$ üzenetet:

$y = 57698412740031832079012086993516050689823397875730202393137 5$

RSA algoritmus bizonyítása (1)

Állítás: Bármely x egész számra igaz, hogy

$$(x^e)^d \equiv x \ (\mod N)$$

Tudjuk, hogy d az e-nek mod $\varphi(N)$ inverze, azaz

$$ed \equiv 1 \ (\mod \varphi(N))$$

Tehát: ed egy olyan szám, amit ha elosztok $\varphi(N)$-el, akkor 1-et kapok maradékul, az egész osztás eredménye legyen v:

$$ed = v\varphi(N) + 1$$

Tehát a következőt kell belátnunk:

$$x^{v\varphi(N)+1} \equiv x \ (\mod N) \qquad (1)$$

RSA algoritmus bizonyítása (2)

Bizonyítsuk be, hogy tetszőleges x és s értékekre igaz az alábbi állítás, ha u prím:

$$x^{s(u-1)+1} \equiv x \ (\ \mathrm{mod\ u}\ ) \tag{2}$$

Nézzük azt az esetet, mikor x-et nem osztja u, ekkor a kis Fermat-tétel szerint:

$$x^{u-1} \equiv 1 \ (\ \mathrm{mod\ u}\ ) \rightarrow (x^{u-1})^s \equiv 1 \ (\ \mathrm{mod\ u}\ ) \rightarrow x(x^{u-1})^s \equiv x \ (\ \mathrm{mod\ u}\ )$$

Tehát (2) ebben az esetben igaz.
A másik eset az, mikor x-et osztja u. Ekkor A maradék 0, tehát:

$$x = 0 = x^{s(u-1)+1} \equiv x \ (\ \mathrm{mod\ u}\ )$$

Azaz (2) ekkor is igaz lesz.

RSA algoritmus bizonyítása (3)

- Ezt akarjuk belátni: $x^{v\varphi(N)+1} \equiv x \pmod{N}$
- Tudjuk, hogy $x^{s(u-1)+1} \equiv x \pmod{u}$ ha u prím
- Valamint $ed = v\varphi(N) + 1 = v(p-1)(q-1) + 1$
- Legyen $s_1 = v(p-1)$, valamint $s_2 = v(q-1)$

Ezek alapján igazak a következők:

$$x^{s_1(q-1)+1} \equiv x \pmod{q}$$

$$x^{s_2(p-1)+1} \equiv x \pmod{p}$$

Vagyis q osztja $x^{s_1(q-1)+1} - x$-et és p osztja $x^{s_2(p-1)+1} - x$-et

$$\Downarrow$$

$pq = N$ osztja $x^{v\varphi(N)+1} - x$-et $\rightarrow x^{v\varphi(N)+1} \equiv x \pmod{N}$

RSA előnyei

- Gyorsan lehet kulcsot generálni: prímtesztelésre ismertek gyors eljárások (eldönteni egy számról, hogy prím-e)
- Prímszámok viszonylag sűrűn vannak: prímszámtétel szerint az N nagyságrendű számok között átlagosan lg(N)-edik szám prímszám
- Random 4096 bites szám – több mint 600 számjegy!! – kellően nagy.
- Lnko és az $ax \equiv 1 \pmod{m}$ gyorsan számolható
- Euler-tétel bizonyítása miatt az eljárás igazoltan működik.

- Jelenleg nincs nagy számok faktorizálására ismert számítógépes algoritmus (Nobel díj esély!) => RSA feltörhető
- Shor-algoritmus kvantumszámítógépeken... Talán a közeljövőben

Miről lesz szó

- ✅ **Információelmélet alapfogalmak** – Shannon, entropia, tömörítés
- ✅ **Klasszikus kriptográfia** – Caesar, Vigenère, Enigma
- ✅ **Hibajavítás és csatornamodellek** – Hamming-kód, Reed–Solomon
- **Modern kriptográfia** – ✅ AES, ✅ RSA, hash-ek, digitális aláírás
- **Hálózati biztonság** – SSL, TLS, VPN, MITM támadások
- **Adatvédelem, GDPR** – anonimizálás, nyomkövetés, metaadat
- **Támadások és védekezés** – brute force, phishing, ransomware, sql injection
- **Jövő és kutatási irányok** – kvantumkriptográfia, AI biztonság

Gyakorlati feladatok

Előadások

– **Skriba Izabella:**

DES, kulcs csere, hitelesítési algoritmusok, elliptikus görbe algoritmus

- **Szőcs-Nagy Medárd**

MD5/SHA1/SHA256

- **Gaál Gergő**

phising, brute force, ransomware, sql injection

Digitális aláírás

Mi az a digitális aláírás?

- **Matematikai művelet**
– egy személy vagy rendszer **hitelesíthet egy adatot vagy dokumentumot**,
– **bizonyítja**, hogy ő küldte (vagy írta alá),
– **és azt sem módosították** azóta.

Digitális aláírás – hogyan működik?

- **1. Aláírás:**

A feladó a **saját privát kulcsával** (pl. RSA, ECDSA) aláír egy üzenetet vagy annak **hash-ét**.

- **2. Ellenőrzés:**

A vevő a **feladó nyilvános kulcsával** ellenőrzi, hogy az aláírás érvényes-e az üzenetre.

Titkosítás vs. Digitális aláírás

Titkosítás	Digitális aláírás
Titokban tartja az üzenetet	Hitelesíti az üzenetet
Cél: bizalmasság	Cél: hitelesség + integritás
Olvasás csak privát kulccsal	Ellenőrzés csak nyilvános kulccsal
Pl. RSA, AES, ECC	Pl. RSA-PSS, ECDSA, EdDSA

Hol találkozhatunk digitális aláírásokkal

- **E-mail aláírás (S/MIME, PGP)**
- **Dokumentumok (pl. PDF aláírás)**
- **Kódaláírás** (szoftverfejlesztés, Windows/.exe/.apk/.app)
- **SSL/TLS tanúsítványok** (weboldalak, HTTPS)
- **Blokkláncok, kriptovaluták** (pl. Bitcoin, Ethereum: ECDSA/EdDSA)

Milyen algoritmusokkal működik

- **RSA-PSS** – biztonságosabb RSA-alapú aláírás
- **ECDSA** – elliptikus görbéken alapuló aláírás
- **EdDSA (Ed25519)** – modern, gyors, biztonságos séma
- **Post-quantum (pl. SPHINCS+, Falcon, Dilithium)** – kvantumbiztos jövő

Digitális aláírások részletesebben

- Aláírás pontos működése (hash-elés, kulcsok szerepe, ellenőrzés lépései)
- RSA-alapú aláírás
- ECDSA és Ed25519 – modern megközelítés
- Aláírás támadásai (pl. replay, forgery, forgatás, kulcsszivárgás)
- Digitális tanúsítványok és PKI (Public Key Infrastructure)
- Valódi példák (OpenSSL, Python-kód, tanúsítvány ellenőrzés, PDF aláírás)

Digitális aláírások működése (1)

Digitális aláírás = két kulccsal végzett művelet:

- A **privát kulccsal** aláírsz egy üzenetet
- A **nyilvános kulccsal** bárki ellenőrizheti, hogy az aláírás érvényes

Digitális aláírások működése (2) - küldés

1. Üzenet:

msg = "Az utalás összege: 10 000 Ft”

2. Hash generálás

h=Hash(msg)

Hash algoritmus lehet

pl. SHA-256 → 256 bites érték

Miért hashelünk?

- Mert hosszú üzenetek aláírása drága lenne
- Hash egyirányú és ütközésálló: ha valaki megváltoztatja az üzenetet, a hash is teljesen más lesz

3. **Aláírás létrehozása:**

σ=Sign(h,privát kulcs)

Ez lesz az aláírás (signature) amit az üzenettel elküldünk

Digitális aláírások működése (3) - fogadás

A fogadó fél a következőt kapja ill ismeri

- Üzenet
- Aláírás
- Ismeri a küldő nyílvános kulcsát

1. Az üzenet újra hashelése: h′=Hash(msg)
2. Megnézi, hogy az üzenetből kinyerhető-e ugyanaz a hash:

Verify(h′,σ,nyilva´nos kulcs)⇒?true

Digitális aláírások működése (3) - példa kód

Python kód

Digitális aláírások támadási lehetőségei

1. Replay támadás:

Az aláírt üzenetet egy támadó egyszerűen újraküldi (replay), hogy újabb hatást/ügyletet idézzen elő azért, mert az aláírt üzenet önmagában érvényes parancs (pl. „küldj 1000 Ft”).

Miért működhet?

Mert az ellenőrzés csak az aláírás és az üzenet egyezését nézi, nincs állapot vagy időbélyeg.

Digitális aláírások támadási lehetőségei

Replay támadás kivédése:

- Üzenetbe beépített, szerver által ellenőrzött **nonce/egyedi tranzakció-azonosító**.
- **Timestamp + rövid élettartam** (token érvényesség).
- Szerveroldali állapot: egyszer felhasznált azonosítók (replay-ID) ne legyenek újra elfogadva.
- Titkos csatorna + session (például TLS + autentikáció).
- Kétlépcsős jóváhagyás (humán verifikáció) pénzügyi műveleteknél.

Digitális aláírások támadási lehetőségei

2. Forgery támadás (hamisítás):

a) Existential forgery

Támadó képes találni egy (üzenet, aláírás) párt, ami érvényes, de lehet, hogy az üzenet nem releváns. Lehet „könnyű” egy gyengébb séma esetén.

b) Selective forgery

Támadó képes konkrét célzott üzenetre hamis aláírást létrehozni (pl. „átutalás 1M Ft”).

c) Universal forgery

Támadó képes bármely üzenetre hiteles aláírást generálni (ez a legsúlyosabb).

Digitális aláírások támadási lehetőségei

Forgery támadás kivédése:

Támadás lehetőségét biztosító okok:

- Gyenge aláírási algoritmus vagy rossz implementáció.
- Gyenge hash (ütközés esetén lehet hamisítani).
- Hiányzó vagy rossz padding (pl. régi PKCS1 v1.5 problémák).
- Old school „homomorf” sémák hibás használata.

Védelem:

- **erős, jól elemzett sémák használata** (RSA-PSS, Ed25519, ECDSA helyesen).
- Erős hash (SHA-256+).
- Ne használj saját „home-made” paddingot; használj szabványos sémákat.
- Védelem a választott-üzenet támadások ellen (pl. aláírásnál soha ne add aláírást olyan üzenetre, amit a támadó választhat).

Digitális aláírások támadási lehetőségei

3. Nonce / random újrahasználás (ECDSA/DSA sémák)

ECDSA/DSA aláírásoknál minden aláíráshoz egy véletlen *k* értéket kell generálni. Ha ez az k **újra felhasználásra kerül** két különböző aláírásnál (vagy részben kiszivárog), akkor a privát kulcs kiszámolható.

Matematikai lényeg (röviden):
ECDSA aláírás: $s = k^{-1}(z + rd) \pmod{n}$.
Két aláírásnál (azonos k):

$$s_1 = k^{-1}(z_1 + rd),$$
$$s_2 = k^{-1}(z_2 + rd).$$

Sony PS3
Bitcoin wallet hibák

Kivonva:

$$s_1 - s_2 = k^{-1}(z_1 - z_2) \Rightarrow k = (z_1 - z_2)/(s_1 - s_2) \pmod{n}.$$

Ha k ismert → privát kulcs kiszámítható:

$$d = (s_1 k - z_1) r^{-1} \pmod{n}.$$

Digitális aláírások támadási lehetőségei

Nonce / random újrahasználás kivédése:

- **determinisztikus nonce használata** (RFC6979):
  k lesz Hash(privkey, msg) — megbízható módon,
  de nem újrafelhasználható sértés nélkül.
- RNG **kriptográfiailag erős** (OS RNG, os.urandom, /dev/urandom, libsodium).
- Edge: Ed25519 megoldja — a séma belül gondoskodik a determinisztikus, biztonságos eljárásról (nincs k kezelés a felhasználó részéről).
- Ne ossz meg partiális k-információkat, és védd a kulcsot.

Digitális aláírások támadási lehetőségei

4. Forgatás / signature malleability (aláírás módosíthatósága):

Egy érvényes aláírásból a támadó konstrukciósan előállíthat egy másik, eltérő, de még mindig érvényes aláírást ugyanarra az üzenetre (signature malleability). Ez zavarhat tranzakció-azonosítókat (pl. blockchain), log-ellenőrzést, protokollok egyezését.

Példa: ECDSA-nál (r, s)és $(r, -s \bmod n)$is érvényes.

Bitcoin emiatt gyakran canonicalizálja s-t a kisebb félsúlyra (low-S rule) hogy eltérés ne forduljon elő.

Digitális aláírások támadási lehetőségei

Forgatás / signature malleability kivédése:

- Aláírás canonicalizálása (pl. mindig s a kisebb fele legyen).
- olyan sémát használjunk, ami nem malleábilis (Ed25519 determinisztikus, nem ilyen jellegű problémák).
- A protokoll tervezésekor ne az aláírás biteit használd kulcs-azonosítóként.

Digitális aláírások támadási lehetőségei

5. Kulcsszivárgás:

A privát kulcs elvesztése / kiszivárgása: a támadó képes hamis aláírásokat készíteni, visszavonni nem tudod, ergo teljes kompromisszum.

Megszerezhető:

- Rosszul védett fájl, rossz jogosultságok.
- Mentés felhőben titkosítás nélkül.
- Side-channel: időzítés, power analízis, cache-based támadások.
- Fejlesztői hiba (kulcs beégetve kódban).
- Gyenge jelszóval védett PEM (brute force).

Digitális aláírások támadási lehetőségei

Kulcsszivárgás kivédése:

- **HSM / TPM / hardware token** (kulcs soha nem hagyja el a hardvert).
- Kulcs titkosítása erős jelszóval + scrypt/Argon2 KDF a jelszó-kulcsban.
- Kulcsfrissítés / rotáció és rövid élettartamú kulcsok.
- Korlátozott jogosultság, audit naplózás, hozzáférés-kontroll.
- Több aláíró (threshold signatures, multisig) kritikus műveleteknél.
- Side-channel védelem: constant-time implementációk, no branching on secret, blinding (RSA), hardveres védelem.

Digitális aláírások támadási lehetőségei

6. Chosen-message és chosen-plaintext támadások

A támadó interaktívan kérhet aláírást tetszőleges üzenetekre (pl. rajta keresztül fut egy „aláíró szervezet”), majd ezekből megpróbál létrehozni hamis aláírást egy célzott üzenethez.

Kivédése:

- Soha ne adj aláírást „nyilvános aláíró szervezetnént” kontroll nélkül.
- Használj kontextus-kötött aláírásokat (pl. domain-separation: az aláírandó adatban tüntesd fel a célprotokollt).
- Hívj be ellenőrző lépéseket, ha szolgáltatás kulcsokkal aláír – limitáld a hozzáférést.

Digitális aláírások támadási lehetőségei

7. Side-channel (időzítés, táplálás, csatorna) támadások

A támadó fizikai vagy időbeli információt gyűjt az aláírási műveletről (mérések), ami visszafejtheti a privát kulcsot.

Kivédése:

- **Constant-time** algoritmusok és könyvtárak.
- Hardveres védelem: HSM, TPM.
- Masking, blinding (RSA-nál véletlen blinding).
- Fizikai védelmek, korlátozott hozzáférés.

Digitális aláírások támadási lehetőségei

8. Tanúsítvány-infrastruktúra sebezhetőségek (PKI)

Ha a CA kompromittálódik, hamis tanúsítványokat adhat ki, amelyekkel támadók aláírhatnak hamis kulcsokat és szerepet hamisíthatnak.

Kivédése:

- Több CA (cross-validation), Certificate Transparency logok, OCSP/CRL és rövid életű tanúsítványok.
- Pinning (alkalmazáson belüli), ellenőrzés.

Hasznos tanácsok

- **Algoritmusok:** RSA-PSS (ha RSA), Ed25519 (ajánlott modern), ECDSA csak helyesen és jó nonce-kezeléssel.
- **Hash:** SHA-256/384/512
- **Padding:** mindig használd a szabványos paddinget (RSA-PSS - salt).
- **Kulcsvédelem:** HSM/TPM vagy legalább erős titkosítás és biztonságos tárolás; rövid kulcslétezés; rotáció.
- **RNG:** OS vg. libsodium; ne saját PRNG.
- **Protokoł tervezés:** üzenetbe nonce, timestamp, tranID; szerveroldali állapot; CSRF-szerű védelmek tranzakcióknál.
- **Monitoring & Audit:** aláírási események naplózása, anomália-detektálás.
- **Frissítések:** használd a jól karbantartott könyvtárakat (cryptography, libsodium), és alkalmazd a security patch-eket.
- **Malleability kezelés:** canonicalizálás, használj algoritmust, ami nem malleábilis.
- **Jog és procedúra:** kulcsrevocation, incident response terv.

Minta: rsa_compare_pss_sajat.py

Egyszerű példák

- Github / Példák