Titkosírás
Biztos, hogy titkos? – Biztonság növelése véletlennel
Wettl Ferenc előadása
2010 december 7.

Szabó Tanár Úr két héttel ezelőtti előadásának folytatása, ahol klasszikus titkosító rendszerekről volt szó, melynek kapcsán Wettl Tanár úr Verne Gyula *800 méter az Amazonason* című regényét ajánlotta. A regény lényege egy titkosított szöveg megfejtése, mely a Vigenére-titkosítás segítségével készült. A regény azért is érdekes, mert a) kb. akkor íródott, amikor kitalálták azt a matematikai eljárást, amellyel ez a titkos-írás visszafejthető, b) a regényből kiderül, hogy egy normál-tudású embernek – jelen esetben Vernének, az írónak – nemcsak a titkosírás megfejtése rejtély, hanem magát a titkosító eljárást sem igazán érti, miközben egy izgalmas regényt ír annak alapján.

A Vigenére-titkosítás

Szükséges hozzá: a kódolandó üzenet;
egy kulcs, ami lehet tetszőleges hosszú betű- és számkombináció;

A kódolt szöveget úgy kapjuk, hogy a kulcsot sokszor egymás után téve egy olyan hosszú betűláncot csinálunk, mint az üzenet szóközök nélkül, majd a titkosított szöveg *n*-edik betűje a (szóközök nélküli) üzenet *n*-edik betűjétől számított annyiadik betű lesz, ahányadik a kód-betűlánc *n*-edik betűje az abc-ben.

Példa:	Kódolandó szöveg:	ALMA
	Kulcs:	BUDA
	Üzenet:	BHPA

Mint a példán is látható, ciklikusan (modulo 25) számolunk, ezért lesz az L betű kódja egy *korábbi* betű, a H. Verne az említett regényében leír egy gondolatmenetet, amelyből kiderül, hogy nem volt tisztában ezzel a ciklikussággal, vagyis azzal, hogy egy, az ábécében korábban álló betű jelezhet egy későbbit is.
Visszafejtés: inverz-kulcsot alkalmazunk – a kulcs ismeretében meghatározzuk azt a kulcsot amit az üzenetre alkalmazva az eredeti szöveget kapjuk.

A Vigenére-titkosítás un. polialfabetikus titkosítás, vagyis nem ugyanannyi hellyel tologatunk egy minden betűt, hanem egy ciklus szerint különböző értékekkel.

Hosszú ideig rendkívül biztonságosnak tartották.

(**SAGE**: Mathematica-hoz hasonló computer-algebra program, szimbolikus számításokra képes, a keresőt használja grafikus felületnek, ingyenes – továbbiakban ezen dolgozunk.)

Vigenére-titkosítást Blaise de Vigenére fedezte fel a XVI. században, Kasiskiról nevezték el Kasiski-tesztnek egy bizonyos megfejtési módját (1863), amit valamivel korábban (1854-ben) már C. Babbage is kitalált. Az ötlet alapja, hogy ismétlődő mintákat keresünk a kriptoszövegben (a titkosított szövegben). Például az angol nyelvben rengetegszer szerepel a „the” betűsor, nemcsak névelőként, hanem például a „they”, „their” szóban is. => Ha az

eredeti szövegben egy adott „szó” (= betűsorozat) sokkal többször szerepelt, mint a kulcs hossza, akkor a skatulya-elv alapján a kriptoszövegben lesznek azonos kulcsrészre eső azonos részletek. Márpedig ezek ugyanúgy lesznek kódolva. Másrészt tudjuk, hogy a kulcs hosszának egész számú többszörös távolságára vannak egymástól. Ha sok ilyen van, akkor a távolságaik legnagyobb közös osztójának osztója lehet a kulcs hossza. Innen megsejthetjük a kulcs hosszát. A kulcs hosszát megkapva (*k*), minden *k*. elemet kivéve azokra nyelvi statisztika alkalmazásával, valamint egyéb ügyeskedésekkel viszonylag gyorsan ki lehetett hozni a szöveget. Verne a regényét 1881-ben írta, tehát tkp. már ismerhette volna a kód megfejtésének módját, de ez nem róható fel neki, hiszen a megfejtés nem volt annyira ismert.

1920: Wolfe Friedman szinte teljesen számítógépesítette az eljárást, még a nyelvi statisztikákra sem igazán van szükség és sejtéseket sem kell kipróbálni, mint a korábbi eljárások során. Módszerét az előadáson Kehlmann: A világ felmérése c. könyvéből vett idézeten mutatta be az előadó. Matematikai lényege a következő:

Definíció: Koincidenciaindex annak a valószínűségét jelenti, hogy egy karakterláncban (betűsorozatban) két véletlenül választott karakter azonos. Ezt a valószínűséget *n* hosszú karakterlánc esetében a következő képlet adja meg:

$$I_c = \frac{\sum_{i=0}^{25}\binom{b_i}{2}}{\binom{n}{2}} = \sum_{i=0}^{25}\frac{b_i(b_i-1)}{n(n-1)}$$

itt b_i az *i*-edik betű gyakorisága a *c* karakterláncban. (Tehát például b_0 az a betű gyakorisága, b_1 a b betűjé, stb.) A képlet könnyen bizonyítható, hiszen a jó esetek száma van a számlálóban (minden *i*-re összeadjuk azt, hogy az *i*-edik betűt hányféleképp választhatom ki két különböző helyről, ezek a jó esetek, majd a jó esetek számát elosztjuk az összes esettel).

Amennyiben c véletlen karakterlánc, I_c~1:26~0,03846, hiszen ebben az esetben minden b_i nagyjából ugyanannyi lesz, *n*/26. Nagy n esetén lényegében $(1/26)^2$-et kell összeadnunk 26-szor.

Amennyiben ez egy értelmes angol szöveg, a koincidenciaindex kb 0,065. Ez jelentős különbség.

A Vigenére-titkosítás elmossa az adott nyelvre jellemző speciális különbségeket, vagyis a szövegre számolt koincidenciaindex csökkenni fog. Ennek oka, hogy a mivel a kódolás során egy fajta betűből többféle is lehet, nem marad meg a karakterek számának egymáshoz viszonyított aránya (még a betűk felcserélésével sem), így a gyakori betűk aránya csökkenni fog.

Az eljárás során vesszük az *l*. betűk koincidenciaindexét, tehát minden *l*. betűből készítünk egy új karakterláncot és kiszámoljuk ennek a koincidenciaindexét. Ha $l \geq 1$, akkor több ilyen is van, ahol több van, ott azok átlagát vesszük. (minden első betű, minden második betű az elejétől, minden második betű a másodiktól, minden harmadik az elsőtől, stb).

Ha l megegyezik a kulcs hosszával, minden l. betű ugyanazzal a számmal van eltolva, így azok kódolt koincidenciaindexe megegyezik a nyers betűkre számolt koincidenciaindexszel.

Ezek közül az az l, melyre a koinciaindex kiugró nagyságú, valószínűleg a kulcs hossza, vagy annak többszöröse, ugyanis könnyen belátható, hogy azokhoz is ugyanolyan kiugró koincidenciaindex társul: tegyük fel, hogy az eredeti kulcsunk „BUDA" volt, a definícióból látszik, hogy a „BUDA" kulccsal kódolni ugyanaz, mintha pl. a „BUDABUDA" kulccsal tennénk azt, így tulajdonképpen egy megtöbbszörözött kulcs hosszához jutunk.

Az így kapott l szeletet most egymással összehasonlítva próbáljuk megkapni a rájuk jellemző eltolást (a kulcsot nem kaphatjuk meg, csak a benne lévő betűk távolságát).

Feladat:

Adott $p_1 \geq p_2 \geq p_3 \geq \ldots \geq p_n > 0$ valószínűségi eloszlás, valamint $q_1, q_2, \ldots, q_n$ is valószínűségi eloszlás.

BBH $\sum_{i=1}^{n} p_i q_i$ maximális $\Leftrightarrow q_1 \geq q_2 \geq q_3 \geq \ldots \geq q_n$.

Rendezési elv, avagy „Szűcs Adolf" egyenlőtlenség. (Magyarországon azért hívják így, mert Sz. A. bizonyította hogy ebből szinte minden ismert egyenlőtlenség kijön)

Mi reméljük, hogy az adott szeleteinkben (adott l. betűkben) az egyes betűk gyakorisága közel azonos. Ebben az esetben vesszük egy szeletben a betűk gyakoriságát, valamint egy másik szelet betűinek minden ciklikus permutáltját (vagyis az összes betűt sorra 1,2, … , 25 hellyel eltolva), ezek páronkénti szorzatának, vagyis gyakorlatilag relatív koincidenciájának maximumát keressük. Ez azt jelenti, hogy a második szeletben lévő betűket úgy próbáljuk meg eltolni, hogy a titkosítás valamint a jelen mozgatás alatt összesen annyival térjenek el eredetijüktől, mint az első szeletbeliek. Amennyiben ez sikerül, ott a leggyakoribb betű kódoltja a második szeletben azonos lesz mint az elsőben, a második, harmadik leggyakoribb, illetve a legritkább betűé szintén. Amikor nézzük e permutációk és az első szelet relatív koincidenciaindexét, tulajdonképpen a fenti rendezési egyenlőtlenség q_i egyes permutációit (a ciklikusakat) vizsgáljuk. Az egyenlőtlenség szerint ez az összeg akkor maximális, ha az első, valamint elforgatott szeletünkben azonos betűk azonosan gyakoriak, vagyis ahol a maximum található, az az eltolás (a ciklikus permutáció eltolásának száma) lesz a kulcs két betűjének távolsága.

A gyakorlatban az előadáson a kulcs a „KORTE" szó volt, az eljárás elvégzésével az alábbi mátrixot kaptuk:

0	4	7	9	20	Ahol az i. sor j. eleme a kulcs
22	0	3	5	16	j. betűjének távolsága az i.-től.
19	23	0	2	13	Amennyiben valahol a távok nem egyeznek, az
17	21	24	0	11	azért van mert a függvény véletlenül máshol vett
6	10	13	15	0	fel maximumot, ezt korrigálhatjuk.

Mostanra a nyelv használata nélkül gépiesen eljutottunk odáig, hogy tudjuk a kulcs hosszát, valamint amennyiben egy betűjét rögzítjük, az egész kulcsot. Innentől már csak tényleg az adott nyelv nyelvi statisztikáit használva haladhatunk tovább, de az így keletkezett 26 lehetőséget fejben is meg tudjuk vizsgálni.

Ez a titkosítási rendszer lényegében a legbiztonságosabbnak, megfejthetetlennek gondolt rendszer volt az 1800-as évekig.

Felmerül a kérdés: létezik-e tökéletesen biztonságos rendszer? Egyáltalán miből következik, hogy tökéletesen biztonságos?

ONE TIME PAD

- Tökéletesen biztonságos kriptorendszer,
- Legalább az üzenet hosszával megegyező hosszúságú 0-1 kulcsot igényel, mely egyszer használható, ugyanis ha többször használnánk, az előző titkosírások szövege alapján lenne olyan információnk, ami segítségével az új titkosított szöveg ismerete információt nyújt az eredeti üzenetről. (Vagyis a titkosítás definíció szerint nem lenne tökéletesen biztonságos.)
- Viszonylag kevés esetben van lehetőség az üzenet hosszával megegyező kulcsot megbeszélni előre, így a One Time Pad kissé irreálisnak mondható. (Egy bizonytalan csatornán akarunk üzenetet küldeni, de előtte egy teljesen biztonságos csatornán kellene egy ugyanolyan hosszú szövegben megállapodnunk – erre aránylag ritkán van esély.)
- Az eljárás a következő:
  nyílt szöveg: 0010110011100
  kulcs (egy teljesen véletlenszerűen generált 0-1 sorozat): 1110110001011
  kriptoszöveg: 1100000010111
  Tehát mod 2 összeadjuk a kriptoszöveg és a kulcs *i*. jegyét, vagyis 0-ból 1, 1-ből 0 lesz.
- Könnyen látszik, hogy visszafejtésénél a kulcsot ugyanúgy kell alkalmazni a kriptoszövegre, mint ahogyan a nyílt szövegre tettük. (A kódolásnál minden karakter az ellentettjére változik, ezt megismételve az eredetihez jutunk.)

Magát a titkosítási eljárást nem nagyon használják, de az ismertetett *műveletet* (a mod 2 összeadást) egyszerű visszafejthetősége miatt gyakran használják titkosításoknál.

Matematikailag mit jelent, hogy „teljesen biztonságos egy kriptorendszer”? Ezt csak Shanon fogalmazta meg az 50-es években:

Definíció: Egy kriptorendszer akkor tökéletesen biztonságos, ha
P(*x* üzenetet küldték el)=P(*x* üzenetet küldték el | *y* üzenetet kaptuk)

Vagyis akkor tekinthetünk egy kriptorendszert tökéletesen biztonságosnak, hogyha a kódolt üzenetet megkapva ugyanannyit tudunk az eredeti szövegről, mint annak ismerete nélkül. A fenti eljárásról bebizonyítható, hogy ebben az értelemben tökéletesen biztonságos.

– Érdekes tény, hogy Julius Caesarnak az előző előadásban említett titkosítási módszere (melyben a szöveg minden betűjét azonos *n* hellyel toljuk el) *egy karakter hosszúságú üzenet esetén* teljesen biztonságos, hiszen minden eltolási hossznak azonos esélye van, így az üzenetben kapott karakter bármi lehet.

Börtönüzenet-fejtés

Wettl Tanár úr egy 2009-es cikket mutatott (Diaconis írta), mely szerint amerikai börtönpszichológusok egy állami börtönben titkosírásos üzeneteket kaptak el, melyet a Stanfordi egyetem tudósainak nyíltnapjára vittek be. (A Stanford egyetem hetente egy nyílt

napot tart, amikor mindenki elviheti valamely felvetődött matematikai problémáját, és az egyetem matematikusai megpróbálják megoldani azt.) Az egyetem tudósai hamar észrevették, hogy primitív helyettesítéses, betűpermutációs titkosírásról van szó (vagyis minden betűnek egy jel (egy „krix-krax") volt megfeleltetve). A nagy részben manuális visszafejtéshez nyilván szükség lenne a latin-amerikai börtönszleng ismeretére, és ebben a Stanford matematikusai nem jeleskednek. Ezért a levelek teljesen gépesített megfejtésére törekedtek. A módszerüket az ún Markov-láncokra és a Metropolis-algoritmusra építették. Nézzük először, mi is az a Markov-lánc.

Definíció: Markov-lánc: olyan sztochasztikus (véletlen) folyamat, amelynek lényege az, hogy az $n+1$-edik időpontbeli állapota csak az n-edik időpontbeli állapottól (és persze a véletlentől) függ, a korábbiaktól nem. Mi most az időben homogén fajtáiról beszélünk. Ez azt jelenti, hogy annak a valószínűsége, hogy valamely állapotból egy másikba jut a folyamat, időben állandó. (Ha valaki ismeri a „ferde foci" nevű valószínűségi játékot, az egy aránylag egyszerű példa erre.)
Az időben homogén Markov-láncok – ha az állapotok száma véges – leírhatók egy mátrixszal, melyben az i, sor j. eleme azt jelöli, hogy mely valószínűséggel jutunk a j. állapotba az i.-ből.

Az ötlet a következő. A tudósok Markov-láncnak tekintették a szöveget, melynek állapotai azok a lehetséges szövegek, amelyeket úgy kapunk, ha az angol ábécé és a kriptoszövegben szereplő „krix-kraxok" között veszünk egy 1-1 értelmű megfeleltetést és ezt „ráeresztjük" a kriptoszövegre. Minden ilyen „megfejtésnek" van egy plauzibilitása: ezt abból számoljuk ki, hogy milyen valószínűséggel követheti egymást két, a szövegben egymás után álló betű az angol nyelvben. (Például nyilván sokkal kisebb a valószínűsége, hogy egy „u" után egy „i" jön, mint hogy egy „u" után egy „n" jön.) Ez persze durva feltételezés: nem igaz, hogy egy betű után melyik másik következik, az csak a közvetlenül előtte álló betűtől függ. Ha például már megfejtettük, hogy „lekváros kenyé", akkor elenyésző a valószínűsége annak, hogy *ez* után az „é" után nem „r" következik. A tudósok mégis ezzel a durva feltételezéssel, mondhatnánk: rossz feltételezéssel éltek és – csodálatosan működött. „Illetve – tette hozzá Wettl tanár úr –, később elmesélem azt is, hogy nekem nem működött. Nagyon tanulságos, hogy milyen hibákat követtem el." (A cikk bemutatta, hogy a híres Hamlet-monológ első sorait a „Monte Carlo" algoritmussal „megkeverve" előállított szöveget a Markov-lánc feltételezésen alapuló algoritmusuk milyen iramban fejtette meg: a 2000. iterálás után már lényegében jó szöveget kaptak. Az 1800. után még „To be or sot to be that is the quentios" volt az aktuális szöveg.)

Az algoritmus ezek után a következő. Veszünk egy, a fenti módon kapott „megfejtést" és kiszámítjuk a plauzibilitását. Ezután két véletlenszerűen választott betűt kicserélünk az eredeti, az ábécé és a krix-kraxok közötti megfeleltetésben. Az ebből az új megfeleltetésből kapott új „megfejtés" plauzibilitását is kiszámoljuk és ha ez nagyobb, akkor kicseréljük rá a megfejtésünket. Ezután megint véletlenszerűen kicserélünk két betűt és a kapott új „megfejtés" plauzibilitását is kiszámoljuk, ha nagyobb, elfogadjuk az új megfejtést. Ezt folytatjuk addig, amíg nem jön egy rosszabb „megfejtés".
És itt jön a Metropolisz algoritmus érdekessége: *a rosszabb megfejtést sem vetjük el automatikusan.*
Ennek az oka a következő: mi a *legjobb* megoldást keressük, és nem akarunk leragadni egy lokálisan legjobb megoldásnál. Ha ugyanis a rosszabb megfejtést azonnal elvetnénk, akkor nem biztos, hogy eljutunk a legjobb megoldáshoz. Úgy kell elképzelni (sok dimenziós térben), mint ha egy olyan terepen járnánk, ahol van – reményeink szerint egyetlen – kimagasló csúcs és sok kisebb domb körülötte. A veszély az, hogy egy ilyen dombon

elakadunk, mert onnan csak lefele vezet út. A Metropolis-algoritmus idővel remélhetőleg kimozdít az ilyenekből. Ha tehát az új szöveg nem lett jobb, a következőt tesszük: Feldobunk egy súlyozott pénzt, ahol a fej és írás úgy van súlyozva, hogy arányuk megegyezik a „régi" (jobb) és az „új" (rosszabb) megfejtés plauzibilitásának az arányával. Ha fej, a „jobbat", ha írás, a „rosszabbat" választjuk. Így ha a rosszabb valószínűsége elenyésző a jobbhoz képest, akkor csak elenyésző annak a valószínűsége, hogy a rosszabbat választjuk, ha közel van egymáshoz a valószínűségük, akkor közel egyforma valószínűséggel választjuk a kettőt.

Ez tehát a M**etropolis-algoritmus**. (Metropolistól és szerzőtársaitól származik, a szerzőtársak között ott van Teller Ede is.) Nagyon egyszerű, teljesen gépiesített, bár tény, hogy a plauzibilitási index kiszámításához nyelvi statisztikára van szükség.

Magyar titkosírásfejtő program készítése

Wettl Tanár úr a hétvégéjét az előadásra készülve a fenti program magyar nyelvre való megírásával töltötte. A már egyszer szereplő szöveget a betűkre alkalmazott véletlen algoritmussal „megkeverte", majd a Metropolis-algoritmus magyar változatával akarta megfejteni. A program lényegesen nem tér el az angol verziótól, csupán a nyelvi statisztikát kellett magyar nyelven is megcsinálni.

A program lépései:

- Magyar nyelvi statisztika készítése
    - Arany János művéből csinálta Wettl Tanár úr
    - Megszámolta hogy adott betűből hány van a szövegrészletben, ezáltal megkapta a betű előfordulásának valószínűségét.
- A program többi része az angolhoz hasonló

⇒ Valamiért nem működött jól:

- Volt egy „megoldás", amely nagyjából a „legjobbnak" nézett ki – és kiderült, hogy ennek a plauzibilitása nagyobb, mint az eredeti (vagyis: a keresett) szövegé!
    - Ennek oka, hogy Arany János *Toldi estéje* című műve nem elég reprezentatív példa egy mai magyar szöveg megfejtéséhez (aminek más az összetétele, mint régen – a kapott megoldások hangzásban hasonlítottak a Toldi stílusához),
    - Valamint hogy a szóközöktől, ékezetektől a statisztika készítésénél eltekintett.
    - ⇒ ezek túl nagy különbséget okoznak.

RSA – Nyilvános kulcsú titkosítás

- Az RSA titkosítást Szabó Tanár Úr két héttel ezelőtti előadásán már ismertette, így itt csak az új megjegyzéseket szerepeltetjük, mely az RSA tökéletesen biztonságossá tételéről szóltak:
- RSA titkosítási módszer a Jakobi-jelet megőrzi szemantikailag nem biztonságos (a Jakobi jel 0-át vagy 1-et társít a számhoz, az üzenethez ugyanannyit itt, mint a kriptoszöveghez. Ezt nem fejtettük ki részletesebben.). Ez az egy bites információ itt jelenlegi tudásunk szerint nem segít a visszafejtésben, de Sharon feltételét így már nem teljesíti (azaz a kriptoszöveg ismerete a Jakobi jele miatt információt ad nekünk az eredeti üzenetről), valamint van olyan szintén egy bites információ ami alapján visszafejthető.

Házi Feladat:

x szövegnek *y* a titkosítása
Half(*y*) = 0, ha $0 < x < n/2$
1, ha $n/2 < x \leq n - 1$.
($n = pq$)

Hogyan törjük fel ennek a függvénynek a segítségével az RSAt?
(ez egy un orákulumos algoritmus, vagyis ha van egy olyan orákulum, aki polinom időben (nagyon gyorsan) megmondja *y*-hoz half(*y*)-t, akkor az RSA nagyon gyorsan feltörhető.
Segítség: x^e hogy viselkedik kül. X-ekre? $X_1^e X_2^e$ szorzatot hogy írhatjuk fel másképp?
Gondoljunk felezésre!)

Jelenleg nem tudjuk a h-t megkeresni, de ha meglenne, a házi feladatban megtalált módszerrel feltörhetnénk az RSA-t.

Hogyan lehetne az RSA-t teljesen biztonságossá tenni?

f(x): $x \Rightarrow x^e \bmod n$
Ahol r egy véletlen szám, p és q az RSA létrehozásához szükséges két nagy prím, n=p*q.
G: r => G(r) véletlen, egyirányú, vagyis a G-t úgy kapjuk, hogy egy orákulumtól kérdezzük mi a G(x), ő kitalál egy random számot és megjegyzi. Ez eléggé bonyolult, nem lett kifejtve részletesebben, lényeg, hogy véletlenszerűen viselkedő és egyirányú.

Ebben a módszerben x-hez, vagyis a szöveghez
$X \Rightarrow (f(r), G(r) \oplus x) = (y_1, y_2)$ számot rendeljük hozzá.

A visszafejtést $G(f^{-1}(y_1)) \oplus y_2 = x$ függvénnyel végezzük.

Ezt az orákulum modellt használva az RSA szemantikailag is teljesen biztonságos.