SZEMÉLYES VÉLETLENSZÁM GENERÁLÁS[1]

FERENCZI ZOLTÁN – NEMETZ TIBOR
Győri Széchenyi István Egyetem – MTA Rényi Alfréd Mat. Int.

Digitális dokumentumok, szerződések aláírásához az aláíró személyétől függő, általa generált véletlen véletlenszámokat célszerű használni. Véletlen számok generálásának erről az oldaláról számolunk be a jelen munkában.

Hagyományos és digitális dokumentum

A közelmúlt tinta-papír, írógép-papír világában törvény szabályozta az okmányok formáját és tartalmát. Törvényi előírás határozta meg, hogy mikor szabályos egy aláírás, egy cégszerű aláírás, amelyet közjegyző hitelesíthet a törvény számára. Az elektronikus hírközlés világában ugyanilyen törvényi szabályozásra került sor. A hagyományos papír- alapú világban az aláírás azonosításának, hitelesítésének több ,,kelléke" is volt, így maga az aláírás, pecsét, tanúk, illetve a közjegyző. Az aláírás célja a hitelesítés. Az, hogy igazolja: *a jogügylet meghatározott tartalommal, adott helyen és időben létrejött.* Az aláírás tehát azonosítható, és egyetlen személyhez kapcsolódik. A közjegyző feladata a hitelesítés, illetve hiteles példány őrzése és arról másolat kiadása.

Ugyanez a feladat az elektronikus társadalomban, az elektronikus üzletvitel világában kihívást jelent: *az értékesítési folyamat elektronikus eszközök, internet segítségével zajlik,* ami számos megoldandó kérdést vet fel, kezdve a személyes adatok védelmétől a tartalom biztonságáig, a hozzáférési jogokig, és persze a ,,digitális közjegyző" által hitelesített elektronikus aláírásig.

Elvárásunk egy digitális dokumentummal, okirattal szemben az, hogy akkor legyen érvényes, ha teljesül, hogy

- Szerzője és tartalma letagadhatatlan
- Megállapítható, hogy mikor készült (keltezés-igazolás).
- Sem szerzője, sem más nem változtathatja meg (sértetlenség).
- Jogtalan hozzáférő ne tudja elolvasni (bizalmasság).
- Meg lehessen győződni arról, hogy az készítette, akinek vallja magát (hitelesség).

Megoldás: sűrítmény (digitális lenyomat) készítés.

[1]Beérkezett: 2005. szeptember 18. E-mail: z.ferenczi@invitel.hu.

Sűrítmény (digitális lenyomat)

A digitális hírközlés során többnyire hosszabb dokumentumok cserélnek gazdát. Ezek egy részében nem követelmény, hogy azok tartalma titokban maradjon. Sokkal gyakrabban és erősebben jelenik meg az a követelmény, hogy senki ne piszkálhasson bele a dokumentumba, vagy ha belenyúlnak, akkor az legyen gyorsan felismerhető. Erre a célra hosszú dokumentumokból rövid lenyomatokat állítanak elő, amelyek nagyon erősen függnek az eredeti dokumentumoktól. Ezek a rövid stringek olyanok, hogy gyakorlatilag lehetetlen hozzájuk olyan új dokumentumot konstruálni, amelyiknek ugyanez a lenyomata. Ily módon lehetetlen két olyan dokumentumot konstruálni, amelyeknek azonos a lenyomata. Ha a dokumentumban legalább egy bitet megváltoztatunk, akkor a megfelelő lenyomatok sok bitben fognak különbözni. A lenyomatokat úgy definiálják, hogy előállításuk könnyű feladat legyen. Azt a leképezést, amely egy ilyen leképezést előállít, *hash függvény*nek nevezik. A hash függvény lényegében véve egy olyan transzformáció, amely egy tetszőleges hosszú szöveg digitális ujjlenyomatát készíti el. Az ujjlenyomat fix hosszú bitsorozat, amely jellemző az adott szövegre, és amely *digitális aláírás protokoll* szerves alkotórésze.

A hash függvényeket eredetileg adattárolásra és adatbankokban való keresésre dolgozták ki. Lényegében az ott elért eredményeket használják fel a kriptográfiában is, ahol azonban további feltételekre is figyelemmel kell lenni.

A hash függvénnyel szembeni elvárásokat a következő követelmények összegezik:

1. Gyakorlatilag lehetetlen egy adott outputhoz olyan dokumentumot konstruálni, amelyikhez a hash függvény ugyanazt az outputot rendeli.

2. Gyakorlatilag lehetetlen 2 olyan dokumentumot konstruálni, amelyek azonos hash értéket eredményeznek.

3. Ha legalább egy bitet egy dokumentumban megváltoztatunk, akkor a megfelelő hash értékek több bitben különböznek.

4. Elegendő hosszúságú bementi dokumentum esetén a *sűrítmény véletlenszerűen* viselkedik. 5-6 darab 32 bites szóból álló bemenet már ,,elegendően" hosszúnak tekinthető.

A 4. tulajdonság, a *véletlenszerűség* külön kommentárt igényel. Az irodalomban nem található ,,szép" hivatkozás arra, hogy ez a tulajdonság valóban teljesül. A konkrét alkalmazáshoz arra van szükség, hogy a sűrítmény *bármilyen eloszlású* bemeneti string esetén is egyenletes eloszlású stringnek legyen tekinthető. Igazából ennél kevesebb is elegendő, elég úgynevezett felcserélhető valószínűségi változókból álló bemeneti változókra igazolni (tehát egy adott sorozatnak ugyanakkora a valószínűsége, mint a sorrend bármely permutálásával adódó sorozatoknak). E mögött az állítás mögött egy határértéktétel áll: Ha (majdnem) független, (majdnem) azonos eloszlású valószínűségi változó sorozatot nézünk egy modulusban, és egy blokkon belüli valószínűségi

változókat modulo aritmetikában összeadunk, akkor enyhe (mindig teljesülő) feltételek mellett az összeg határértékben (a blokkhossz növekedése esetén) egyenletes eloszlású lesz.

Véletlenszámok generálása

A kriptográfiában véletlenszám alatt heurisztikusan a következőt szokás érteni. Valamilyen k egész szám mellett tekintsük a $\{0, 1, \ldots, k-1\}$ értékkészletet. Tekintsük ebbe a halmazba tartozó számok olyan sorozatát, amelyeknek mindegyike lényegében ugyanannyiszor fordul elő a sorozatban, mégpedig bármelyik helyen bármelyik szám egyenlő eséllyel fordul elő (*egyenletesség*), függetlenül attól, hogy mi volt előtte, vagy mik jönnek utána (*függetlenség*). Az ilyen sorozatokat véletlenszámok sorozatának nevezik. Az értékkészlet k számosságának szokásos megválasztása $k = 2$, amikor *bináris véletlenszámokról* beszélünk. Ezek központi jelentőségét az információelméletben figyelhetjük meg. Ugyancsak gyakori a $k = 10$ választás, amikor *decimális véletlenszámokról* beszélünk. A $\{0, 1, \ldots, k-1\}$ értékkészlet helyett egy nyelv ábécéjét is lehet értékkészletnek tekinteni. Ekkor is véletlenszámokról beszélnek.

Véletlenszámok manuális előállításához kockát, pénzérmét, kártyát, sorsolást szoktak használni. Ez azonban egy rendkívül lassú folyamat. Ezért keresnek más lehetőségeket is. A 20. század elején a statisztikai következtetésekhez a véletlenszámokat manuálisan állították elő és táblázatolták. Az első ,,nagy" táblázatot az angol Tippet készítette 1927-ben: 40 ezer véletlen számjegyet állított elő népszámlálási adatokból.

Véletlenszámok fizikai generálása

Kriptográfiai kulcsok generálásakor a pszeudó véletlen számgenerátorok előnyei rendkívüli hátrányokká válnak. Ilyenkor különösen fontos, hogy

- az elkészült sorozat ne legyen reprodukálható
- ,,információtartalma" megegyezzen a méretével.

Fentiek miatt pusztán fizikai véletlenszám generátorokat használnak a rejtjelkulcsok készítésére. *A fizikai véletlenszám generátorok alapjául véletlen jeleket adó természeti jelenség szolgál.*

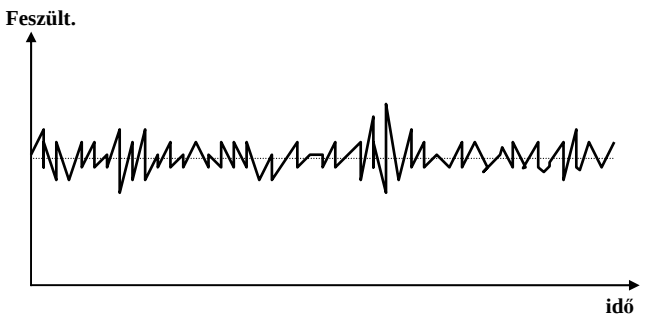
A gyakorlatban legtöbbször valamilyen rádiótechnikai eszköz a jelforrás. Segítségével építhető olyan számítógépes bővítőkártya, amelyben a fizikai folyamatot erősítik, mintavételezik és amely a kapott digitális jeleket átadja egy számítógépnek. A generált sorozatot folyamatosan ellenőrizni kell, mert a fizikai folyamatot befolyásolja a környezete, amely enyhébb esetben az eloszlás megváltozását, súlyosabb esetben degenerációt okoz.

A következőkben leírjuk egy PC-be illeszthető fizikai generátorkártya felépítését. Ennek a fizikai véletlenszám generátornak az alapja, a zajforrás

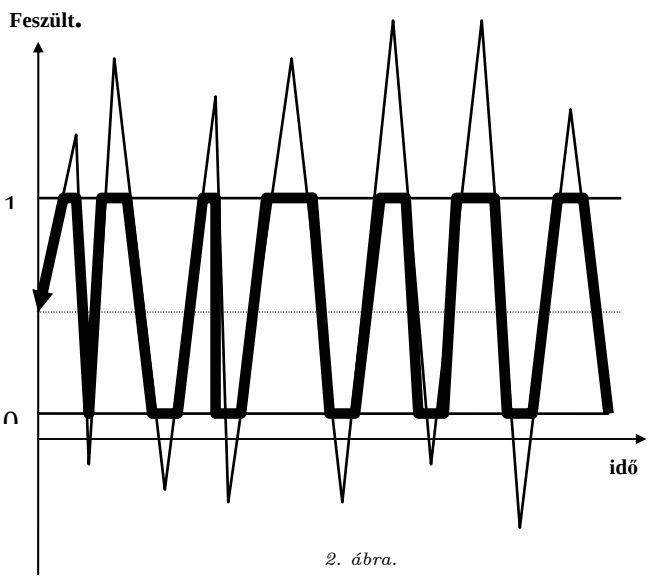
egy ún. *Zener dióda*. Az ezen átfolyó áram feszültsége alapvető fizikai törvényszerűségek miatt kismértékben ugyan, de véletlenszerűen ingadozik. Az ingadozás „frekvenciája" 0 és néhány száz (200) kHz között változik (*1. ábra*).

A kártyára épített elektronika ezt az ingadozást több fokozatban erősíti, majd a jel „közepén" szimmetrikusan kijelöl egy intervallumot, amelynek két szélső értéke lesz a 0 és az 1 érték (*2. ábra*).

A jelet ezután négyszögesítjük. Ezután kerül sor a mintavételezésre, amelynek frekvenciája néhány ezer bit másodpercenként, vagyis a mintavételezés frekvenciája nagyságrendekkel kisebb a kiinduló jel frekvenciájánál, ami a szomszédos minták függetlenségét biztosítja.



1. ábra.



2. ábra.

A generátorkártyán két független jelforrás és mintavevő rendszer működik, a két forrásból származó biteket modul 2 (xor) összeadva kapjuk azt a bitet, amely a generátor outputja lesz. A PC-be illesztett kártyához egy, a PC operációs rendszerén futó driver tartozik, amely statisztikai ellenőrzéseket is végez az outputra, s szélsőséges esetben riaszt. Szükséges még egy applikáció, amely segíti a kártya installálás utáni kalibrálását, és statisztikai teszteket végez, amelyek eredményét a felhasználó számára meg is jeleníti. Többszörösen hangsúlyozzuk *statisztikai tesztek folyamatos elvégzésének* szükségességét, lásd [3].

Pszeudo-véletlenszámok generálása

Az atombomba gyártásánál felmerült nagyon nagy mennyiségű véletlenszám előállításának az igénye. Fából vaskarika kívánalomként az is felmerült, hogy ezek reprodukálhatók legyenek. Neumann János 1944-ben javasolta azt, hogy erre a célra álvéletlen számokat használjanak. Valódi véletlenszámok helyett determinisztikus számsorozatot állítsanak elő, amelyek *statisztikailag ugyanúgy viselkednek, mintha valódi véletlen sorozatok lennének.* Ezeket a számsorozatokat ismételten elő lehet állítani, reprodukálni, ami a számítógépek hőskorában rendkívül fontos követelmény volt. Az algoritmikusan előállítható, véletlenszerűen viselkedő determinisztikus számsorozatokat *pszeudovéletlen* (álvéletlen) sorozatoknak nevezik.

A Neumann János által javasolt eljárás nagyon egyszerű volt. Kiindulásként választott egy n jegyű számot kezdőértéknek. A pszeudo-véletlen sorozat következő számának előállításához az utolsónak generált n jegyű számot négyzetre emelte, majd az így nyert négyzet középső n jegyéből álló szám lett a sorozat új száma. (Az n jegy szükség esetén nullákkal kezdődhet.)

A négyzetközép módszernek ma már csak történelmi jelentősége van. Pszeudo véletlen számsorozatok előállítására a leggyakrabban a *lineáris kongruencia generátorokat* használják. Ezt a módszert D. A. Lehmer javasolta 1949-ben. Ehhez választanak egy $R(0)$ *egész kezdőértéket, egy A multiplikatív és egy B additív konstanst*, valamint egy nagy M *modulust.* Ezek segítségével az utolsónak előállított $R(N)$ számból kiszámítják az

$$R(N+1) = A * R(N) + B \pmod{M}$$

egész számot. Ebből egy további $U(N)$ álvéletlen számot M-mel való osztással kapunk. Ez az osztás egy bizonyos fajta standardizálást szolgál: az $U(N)$ számok mindig 0 és 1 közé esnek. A számítógépek RND generátora ilyen számok sorozatát szolgáltatja. Innen a $\{0, 1, \ldots, k-1\}$ halmazba eső egyenletes eloszlású egész számokat az

$$[X(N) = k * U(N)]$$

képlet ad meg (adja vissza).

A paraméterek megválasztásával és az előállított sorozat statisztikai vizsgálatával kapcsolatos kérdésekről D. Knuth (1987) 2. kötete ad kimerítő tájékoztatást.

Az algoritmikus eljárások gyors végrehajtására különböző célgépeket szerkesztettek, melyek valódi véletlen sorozatok helyett determinisztikus, de véletlenszerűen viselkedő sorozatokat alkalmaznak. Gyors hardver megoldást tesznek lehetővé a maximális periódusú lineáris visszacsatolt shift regiszterek (angol rövidítéssel LFSR).

Hibrid generátorok

Már a véletlenszámok tömeges előállításának hőskorában megszületett az az ötlet, hogy bármilyen nagy munkával is, de célszerű *egyszer* nagymennyiségű véletlenszámot előállítani, és ezeket közkinccsé téve belőlük bárki, bármikor tetszőlegesen sokat felhasználhat, lásd [2]. Természetesen ez nem egy megbízható megoldás a kriptográfusok számára. Hiszen éppen a legfontosabb követelmény, a kiismerhetetlenség sérül meg. Ezt kell orvosolni ahhoz, hogy ilyen típusú megoldás számukra is használhatóvá váljon. Ezt több lehetőség is segítheti:

- A CD-ROM tárolók megjelenése az általában egyetlen alkalmazáshoz szükséges véletlenszám mennyiség rendkívül sokszorosának megőrzését teszi lehetővé.

- A CD-ROM-okat abszolút biztonságos körülmények között lehet teleírni valódi véletlen (fizikai) generátorok segítségével, miközben a teleírás történhet szubjektív (személyi) beavatkozás nélkül.

- Véletlenszerű kezdettel kis blokkokat úgy lehet kiemelni a CD-ROM-ról, hogy még a kezelő sem képes felismerni, honnan származnak, tehát védelmet lehet biztosítani a saját kezelő személyzettel szemben.

Billentyűleütéssel generált véletlenszám

Javaslunk egy olyan fizikai véletlenszám generátort, amely az internettől függetlenül, „kézi" munkával, a feladat megengedte sebesség (lassúság) mellett állít elő véletlenszámokat. *Az újdonságot az adja meg, hogy az alkalmazást igyekszünk az ad-hoc elemektől megszabadítva, lehetőség szerint szabatossá tenni. Ezért ez újnak tekintendő generálási módszert is eredményez.*

Ebben az eljárásban a véletlenszerűséget a klaviatúrán két egymás után leütött billentyű leütése között eltelt idő adja meg. Ezt az eltelt időt mérjük meg a számítógép beépített órája által megengedett pontossággal. A nyert számot elosztjuk 16-tal, maradékos osztással. A maradékot tekintjük generált (hexadecimális) véletlenszámnak.

Kissé formálisabban a következő lesz az algoritmusunk (amelyben a számítógép pszeudo-véletlenszám generátorát is felhasználjuk).

Algoritmus

1. lépés. Megadunk egy nyomtatható (képernyőre írható) karakterekből álló $ABC\$$ stringet (vagy mátrixot). Legyen a stringben (mátrixban) szereplő jelek száma M, ami a program paramétere. A stringben szereplő karakterek ismeretén kívűl bizonytalansági faktort jelenthet a benne szereplő jelek sorrendje is.

2. lépés. A program deklarálja, hogy csak windos alatt futtatható. Közli, hogy a generált véletlenszámokat mindig az `rndout.doc` fájlba menti. Felszólítja a felhasználót, hogy gondoskodjon az esetleges ilyen nevű fájl tartalmának a mentéséről, mert ezt a program felülírja.

3. lépés. A program rákérdez a generálandó hexadecimális jegyek N számára. Ez a szám 1 és 500 közti tetszőleges egész lehet.

4. lépés. Állítsuk be a PC pszeudovéletlen generátorát egy véletlen kezdőértékre, és generáljunk egymás után N pszeudovéletlen számot az $1, 2, \ldots, M$ egész számok közül.

5. lépés. Minden egyes generált pszeudovéletlen egész esetén írjuk ki a képernyőre az $ABC\$$ string ennyiedik betűjét. (Ez szervezhető $N = 280$ esetén például 4 darab 70 betűs sorba). Ezzel kezdődik a véletlen-hexa generálás.

6. lépés. Üssük le a billentyűzeten egymás után a képernyőn levő N karaktert, miközben a PC méri és kódolja az egymás utáni leütések közti időt, és leírja (feljegyzi, tárolja) a generált hexadecimális számot. A kiírás megint szervezhető 4×70-es mátrixba, ahogy ezt a konkrét példánk teszi is.

Teljesen lényegtelen, hogy a számítógép milyen pszeudo-véletlenszám generátort alkalmaz. A program működésének bemutatására $N = 4 \times 70$ paraméter mellett megadunk egy *leütendő sorozatot*, majd a kiírt sorozat leütésekor létrejövő véletlenszám sorozatot.

```
3gkmh%g7wl g%zi3$6e=, wl3jz-bpfm t6ibcbp8nt 2=a,%p4=ma 24hrcrcp08 ym7yms5x59
ecx4d-tpzn 3$@h5tpa60 37pfakprwu a34-yx@0pv 0jh68$#e7! bbizueg,kw cynu.%r6$8
mesw%wxubn w6b3phq9pz d!a$%ny,x0 kwkzl!%=rf 03ap4xz9%3 za4yrct3o@ xmg0bu@d!1
su=a%cyv!2 .,9uton6yx oo9@x$zcph gmazktp3q3 t1zhw%!t-j ok,t$dp8,v r4wzg8%,xo
```

A leütéssel generált véletlenszám sorozat:

```
82E5F343C5 687D4A1574 F2D60BE321 5B105D9595 70551049C7 65D2EF56BB 9D1A8E27A7
7264AA6BC8 5E906AD00C C6BF053066 7012CC702A C0B118E671 A35D4EDED6 C0EB221A4C
9BBBE0043F 13272E40B2 8781BCCBC9 AF3294E75D 32086A97B2 145276860C 65EB2B9161
96CE07F282 427882DB76 DBC9E59E11 DCB0956BF8 CBF5559E5C 37CFB0CE5D 6ABB3668E0
```

A generált sorozat elemeinek *gyakoriság-eloszlása:*

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
22	17	22	11	12	24	23	19	14	15	12	25	21	14	19	10

1. táblázat.

A χ^2 próba-statisztika értéke: 21,4857. A 15 szabadsági fokhoz tartozó kritikus értékek

P	0,1	0,05	0,01
χ^2 érték	22,3	25,0	30,6

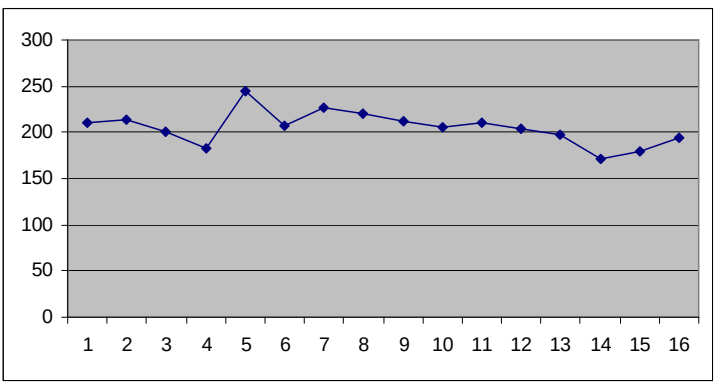
2. táblázat.

Az Excel program segítségével adatainkat grafikusan szemléltethetjük is. Illusztrációként megadunk egy további 4×70-es „valódi" véletlen „leütéses" mátrixnak megfelelő gyakorisági grafikont.

Az $N = 280$ elemű minta gyakoriság-eloszlása a 3. táblázatban, grafikonja pedig a *3. ábrán* látható.

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
16	15	20	17	19	19	16	18	17	16	19	19	19	15	15	20

3. táblázat.



3. ábra. Gyakorisági poligon

Az empirikusan generált véletlen sorozatokat kötelező felhasználás előtt statisztikailag ellenőrizni. Erre a szokásos tesztek megfelelnek, de a generálás módját figyelembe vevő speciális tesztek is szükségesek lehetnek.

Ellenőrizendők a következő feltételezések:

- *Egyenletesség*: Bármely időpontban a 0-k és 1-ek addigi előfordulási száma várhatóan megegyezik.
- *Skálázhatóság*: Minden univerzálisan kiválasztott részsorozat is kielégíti a statisztikai teszteket.
- *Konzisztencia*: Egy pszeudo véletlen generátor statisztikailag nem függhet az algoritmus kezdőértékétől.

Összefoglalás

Digitális dokumentumok azonosítására használt algoritmusok véletlenszámokat alkalmaznak. Ezek a véletlenszámok az alkalmazótól és a dokumentumtól függenek. Ezért az alkalmazónak magának kell generálni ezeket. A dolgozat erre javasol egy olyan módszert, amely semmilyen eszközt nem igényel. A módszer azt használja ki, hogy egy PC billentyűzetén két billentyű leütése közt eltelt idő véletlenszerűen változik.

Irodalom

1. Knuth, D. E. (1987): *A számítógép-programozás művészete 2. Szeminumerikus algoritmusok.* Műszaki Kiadó, Budapest
2. Nemetz, T.–Papp, P. (1998): Hybrid Random Byte Generators, in: *Global IT Security*, Papp, Gy.-Posch, R. (ed), Proc. of the 14th Int. Conf. on Info. Security, Wien, 366–380
3. NIST (2001): *Special Publication 800-22* Statisztikai próbák a kriptográfiai alkalmazásoknál használt véletlen- és pszeudo-véletlenszám generátorokra. (Javított kiadás: 2001. május 15.)

GENERATING PERSONAL RANDOM NUMBERS

Algorithms for the identification of digital documents apply random numbers. These random numbers depend on the user and the document hence the user himself must generate them. In this paper we suggest an algorithm to this task which does not require any other tools. Our method exploits the observation that the length of the time interval between two presses of keys on a keyboard of a PC varies randomly.