Könnyű álmok (13. rész)

A titkosítás titokzatos titkai…

Ebben a hónapban egy kis kitérőt teszünk. Utolsó két cikkünk az Interneten leggyakrabban használt hálózati protokollok sajátosságait tárta fel. Ebbe az irányba haladva mindenképpen szót kell ejtenünk azokról a protokollokról és protokollkiegészítésekről, amelyek az átvitt adatokat valamilyen formában titkosítják. Az elkövetkező néhány cikk a titkosításról és annak gyakorlati alkalmazásairól szól. A téma elérhető szakirodalma kimerítő, sőt magyar nyelvű források is léteznek, ennek ellenére a felhasználók jelentős része úgy tekint a kriptográfiára, mint az űrtudományra – messziről. A titkosítás tudományának eredményeiben vakon bízik, de a mögötte dolgozó algoritmusok és eljárások nagyfokú bonyolultsága visszariasztja a közelebbi ismeretségtől. Fontosnak tartjuk, hogy végre egy olyan összefoglaló szülessen, amely nem igényel komoly matematikai előképzettséget, de a fontosabb eljárások alapelveit mégis meg lehet belőle érteni. Tévedés ne essék, ez a pár oldal a tárgyalt téma pontos, mérhető adatokra alapozott leírása és bizonyítása híján csak tudományos ismeretterjesztő cikknek tekinthető. Reméljük azonban, hogy az itt leírtak sokakban kedvet ébresztenek a titkosítás tudományának behatóbb tanulmányozására. Az érdeklődők az ajánlott irodalomban minden kérdésére kimerítő választ találnak. Akik pedig nem válnak kriptográfussá, azok betekintést nyernek ebbe a komoly és érdekes tudományba. Helyezkedjenek el kényelmesen, és jó szórakozást!

A titkosításról általában

A titkosítás szép, hosszú története helyhiány miatt ma kimarad. Inkább foglalkozzunk a gyakorlattal. Igyekszünk csak arról szólni, ami működésének megértéséhez elengedhetetlenül szükséges. Mi is a titkosítás? Az adatok titkosságával, hitelességével és védettségével foglalkozó tudomány. Két alapvető ága van: a kriptográfia, amely a korábban felvetett kérdések megoldásával foglalkozik; és a kriptoanalízis, amelynek a célja a kriptográfia által adott megoldások hibáinak feltárása. A kriptográfia további két fontosabb részre osztható. Az egyik ága a titkosító algoritmusokkal foglalkozik, a másik azok használati körülményeivel, ahol ezen eljárásokat összefoglaló néven kriptográfiai protokolloknak hívják.

A kriptográfia alapfogalmai

A kriptográfiának, mint minden tudománynak, megvannak a saját szakkifejezései. Ezeket a későbbiek egyértelműsége és a vonatkozó szakirodalom értő tanulmányozásához előnyös megismerni. Az adatok továbbítása általában két fél között történik: a küldő (sender) és a fogadó (receiver) között. Az átvivendő szöveg eredeti formájában a nyílt szöveg (plaintext), titkosítva pedig kriptoszöveg (ciphertext). A nyílt szöveget kriptoszöveggé alakító eljárás a titkosítás (encryption), ellentéte pedig a visszafejtés (decryption). A korszerű titkosító eljárások egy vagy több úgynevezett kulcsot (key) használnak. A kulcs egy, az algoritmus által megkövetelt módon előállított, általában kis méretű adatdarab. A jó titkosító algoritmusok alapvető tulajdonsága, hogy a kulcs ismerete nélkül a nyílt szöveget a kriptoszövegből még az algoritmus ismeretében sem lehet visszanyerni.

A jobb érthetőség végett szerepeljenek a kapcsolattartás lépései a fenti kifejezések használatával:

- A ***küldő*** egy üzenetet megbízhatatlan (lehallgatható) csatornán szeretne a ***fogadónak*** továbbítani.
- A ***nyílt szöveget*** valamilyen eljárással, a ***kulcs*** segítségével ***titkosítja***, így előáll a ***kriptoszöveg***.
- A ***kriptoszöveget*** már nyugodtan elküldheti a megbízhatatlan csatornán a ***fogadónak***, mivel azt – reményeink szerint – más nem tudja elolvasni.
- A ***fogadó*** a kapott ***kriptoszöveget*** a ***kulcs*** segítségével ***visszafejti*** (amely nem feltétlenül egyezik meg a titkosító kulccsal), így a ***nyílt szöveget*** már elolvashatja.

A titkosító eljárások kezdetben semmilyen kulcsokat nem igényeltek, de a népek hamar rájöttek, hogy az ilyen eljárások megbízhatósága nem kielégítő, ráadásul megfejtésük esetén nehéz őket lecserélni. A későbbiekben olyan eljárások kidolgozására törekedtek, ahol a titkosítás megfejtéséhez egy titkos szó vagy könyv, illetve valamilyen eszköz birtoklására is szükség volt. Az ilyen eljárások nagy előnye, hogy ha nem az algoritmus hibás, hanem csak a titkos „kulcs” vált ismertté, annak cseréjével a titkosság helyreállt. A haladó titkosító eljárások is használnak ilyen kulcsokat, de itt a kulcs már valamilyen előre meghatározott tulajdonságokkal bíró adathalmaz, amely elengedhetetlen az algoritmus végrehajtásához. A korábbi kulcsalapú módszerek mai megfelelői, a szimmetrikus titkosítási eljárások egyetlen titkos kulcsot használnak. E módszerek komoly hátránya, hogy a küldőnek és a fogadónak előre meg kell állapodnia a kulcsban. Itt a kulcs biztos továbbítása jelenti a gondot. Ennek kiküszöbölésére hozták létre az aszimmetrikus algoritmusokat, amelyek már nem igénylik a titkos kulcscserét. Mindkét félnek van egy titkos és egy nyilvános kulcsa, és ezen kulcspár segítségével meg tudják oldani a kulcscserét, adott esetben magát az adatátvitelt is. A fent leírtak kissé „matematikusabb” írásmóddal így néznek ki:

Szimmetrikus eljárások (egyetlen kulccsal)

- E(p,k) = c
- D(c,k) = p

tehát

- D(E(p,k),k) = p

ahol

- E – titkosítás (encrypt)
- D – visszafejtés (decrypt)
- p – nyílt szöveg (plaintext)

Eltolási táblázat az angol ábécére:

```
abcdefghijklmnopqrstuvwxyz
efghijklmnopqrstuvwxyzabcd
```

A nyílt szöveg:

Kínos, ha az embert fejbe vágják egy lédús citromszelettel, amit egy aranytéglára erősítettek.

A kriptoszöveg:

omrsw, le ed iqfivx jinfi zekneo ikc pihyw gmxvsqwdipixxip, eqmx ikc evercxikpeve ivswmxixxio.

1. ábra A Caesar-féle titkosítás működése

- c – kriptoszöveg (ciphertext)
- k – kulcs (key)

Aszimmetrikus algoritmusok (több kulccsal)

- E(p,k1) = c
- D(c,k2) = p

tehát

- D(E(p,k2),k1) = p

A lenyomatokról (hash)

És most valami egészen más. A titkosítás kihívásainak megoldása közben olyan eljárások kidolgozása is szükségessé vált, amelyek egy adathalmazról (akár egy nyílt szövegről) képesek lenyomatot (hash) képezni. Ezt az eljárást egyirányú függvénynek vagy kriptográfiai ellenőrzőösszeg-képzésnek, magát a lenyomatot pedig más néven kivonatnak, ujjlenyomatnak vagy kriptográfiai ellenőrző összegnek is hívják. Mire jó ez a gyakorlatban? Tételezzük fel, hogy van egy olyan eljárásunk, amelynek segítségével egy tetszőlegesen nagy méretű adatot át tudunk alakítani egy olyan lényegesen kisebb kötött méretűvé, amelyre igazak az alábbi állítások:

- A lenyomatból az eredeti adatra vonatkozóan semmilyen következtetést nem lehet levonni, vagyis ***egyirányú***.
- Nagyon nehezen található olyan másik adat, amelynek lenyomata megegyezik az eredeti adat lenyomatával, tehát nagymértékben ***ütközésmentes***.
- Az eredeti adat kismértékű változása nagy változást okoz a lenyomatban. Ezt a szakirodalom ***lavinahatásnak*** hívja.

Ha ilyen eljárással képzünk lenyomatot a nyílt szövegről, a lenyomat titkosított változatát az átvitt szöveg mellé csatolva a másik oldal meggyőződhet róla, hogy az üzenetet valóban mi küldtük, és az átvitel során az nem módosult. Sokakban felmerülhet a kérdés, hogy miért nem a teljes átvivendő adatot titkosítottuk. Ennek két oka van: az egyik, hogy bizonyos esetekben az üzenet titkosítására nincs szükség, csak feladójának egyértelmű azonosítása a cél. A másik ok, hogy a titkosító algoritmusok jelentős része nem gyors, sőt kifejezetten lassú. Egy mai átlagos gépen a később tárgyalt RSA algoritmus alkalmazása egy 10 MB méretű állományra néhány perc.
A jó lenyomatképző algoritmusok fontosabb elvárt tulajdonságait fent már felsoroltuk. Tisztázzuk azonban, hogy pontosabban mit jelent az ütközésmentes kifejezés. Mivel

A betűhelyettesítő táblázat

```
abcdefghijklmnopqrstuvwxyz
bcdefghijklmnopqrstuvwxyza
cdefghijklmnopqrstuvwxyzab
defghijklmnopqrstuvwxyzabc
efghijklmnopqrstuvwxyzabcd
fghijklmnopqrstuvwxyzabcde
ghijklmnopqrstuvwxyzabcdef
hijklmnopqrstuvwxyzabcdefg
ijklmnopqrstuvwxyzabcdefgh
jklmnopqrstuvwxyzabcdefghi
klmnopqrstuvwxyzabcdefghij
lmnopqrstuvwxyzabcdefghijk
mnopqrstuvwxyzabcdefghijkl
nopqrstuvwxyzabcdefghijklm
opqrstuvwxyzabcdefghijklmn
pqrstuvwxyzabcdefghijklmno
qrstuvwxyzabcdefghijklmnop
rstuvwxyzabcdefghijklmnopq
stuvwxyzabcdefghijklmnopqr
uvwxyzabcdefghijklmnopqrst
vwxyzabcdefghijklmnopqrstu
wxyzabcdefghijklmnopqrstuv
xyzabcdefghijklmnopqrstuvw
yzabcdefghijklmnopqrstuvwx
zabcdefghijklmnopqrstuvwxy
```

A nyílt szöveg:

Frodó úr szerzett egy gyűrűt, és mindenhová magával hurcolta, ha kellett, ha nem.

A titkosító jelszó:

cirok se prucirok sep ruciro ks epruciroks eprucir oksepruc ir oksepru ci rok

A titkos szöveg:

hzfry mv hqythvhd wkn xswzlh, ok qxexgvycfs qpxuxic vejgdcnc, pr yodptkn, ji esw.

2. ábra De Vignére többábécés módszerének működése

a lenyomat az esetek jelentős részében nagyobb, mint az eredeti adat, a leképezéssel – leegyszerűsítve – egy nagyobb elemszámú halmaz elemeit képezzük le egy kisebb elemszámú halmazba. Elkerülhetetlen tehát, hogy a kiinduló halmaz több eleméhez is ugyanaz az eredmény tartozzon. Ezt nevezik lenyomatütközésnek.

Egyszerű titkosító algoritmusok

A korszerűbb, összetettebb megoldások megértéséhez szükség van az egyszerűbb eljárások és hibáik ismeretére. A titkosítás használatának kezdeteinél a tudományok nem álltak még a mai szinten. Az akkoriban feltörhetetlennek tűnő algoritmusokat a tudomány folyamatos fejlődésének eredményeképpen ma könnyedén fel tudjuk törni. A titkosítás elmélete és gyakorlata a matematika, az információelmélet és az algoritmuselmélet fejlődésével folyamatosan változik, fejlődik. A fontosabb korai titkosítási módszereknek és hibáiknak alapos megismerésével lényegesen jobban átlátható, hogy milyen típusú támadásokkal kell számolnia a titkosítással foglalkozónak.

1. lista Mini titkosítónk: megatitok_3000.pl

```
#!/usr/bin/perl -w

sub help();
use Getopt::Std;
my %changetable = ( "Æ" => "a", "Ø" => "e",
➥" " => "i", " " => "o",
    ➥"o" => "o", " " => "o", "œ" => "u", "u"
=> "u", " " => "u",
    ➥"`" => "a", " " => "e", "´" => "i", " "
=> "o", "O" => "o",
    ➥" " => "o", " " => "u", "U" => "u", " "
=> "u" );

my %opts;
getopts("edp:", \%opts);
if (! defined($opts{p}))
{
    print("A jelsz  sajnos nincs megadva.\n");
    help();
}
$pwd = $opts{p};
$plen = length($pwd);

if (! (defined($opts{e}) or defined($opts{d})))
{
    $opts{e}=1;
}
elsif (defined($opts{e}) and defined($opts{d}))
{
    print("Egyszerre nem tudok titkos tani
    ➥Øs visszafejteni.\n");
    help();
}

my $letters = 0;
while(<>)
{
    chomp;
    for ($i=0; $i<length; $i++)
    {
  my $l = lc(substr($_, $i, 1));
  if (defined($changetable{$l}))
  {
      $l = $changetable{$l};
  }

  if($l =~ /\w/)
  {
      if(defined($opts{e}))
      {
    # encrypt
    print(chr(((ord($l)-97 +
       ➥ord(substr($pwd, $letters % $plen,
       ➥1))-97) % 26)+97));
      }
      else
      {
    # decrypt
    my $char = ord($l)-97 -
        ➥ord(substr($pwd, $letters %
        ➥$plen, 1))+97;
    if ($char < 0)
    {
        $char += 26;
    }
    print(chr($char+97));
      }
      $letters++;
  }
  else
  {
      print($l)
  }
    }
    print("\n");
}

sub help()
{
    print <<EOF;
HasznÆlat: de_vigenere -p <jelsz > [-e|-d]
    A program Blaise De VigenØre titkos t
    algoritmusÆval titkos t vagy fejt
    vissza, a megadott jelsz val. A jelsz t
    a -p paramØterben kell megadni, a
    titkos tÆshoz a -e, a visszafejtØshez
    pedig a -d paramØtert kell megadni.
    Ha egyik sincs megadva, akkor
    a program titkos t.

EOF
    exit(1);
}
```

Julius Ceasar eljárása

A következő eljárást a történelemkönyvek szerint ***Julius Caesar*** alkalmazta először, ezért róla nevezték el. Meglepő módon a neve: Ceasar-féle titkosítás. Lényege a következő: a nyílt szöveg betűit úgy titkosítjuk, hogy azokat az ábécé szerint megadott betűvel eltoljuk. Ha az eltolás négy betű, akkor a nyílt szöveg „a” betűjéből a kriptoszövegben „e” betű lesz, a „b”-ből „f”, és így tovább. Ennél az algoritmusnál a kulcs az eltolás mértéke, jelen példánkban a négy. Az így kapott kriptoszöveg első ránézésre betűk összefüggéstelen halmaza. Ezt az eljárást szemlélteti az *1. ábra*. Az ábrán is jól látható azonban ennek a módszernek a hibája. Mivel a nyílt szöveg bizonyos szabályosságokat mutat, a módszer viszonylag könnyen felismerhető, megfejthető. Minden nyelvre jellemző például a betűk előfordulási valószínűsége, a betűtöbbszöröződések, a betűkettősök és -hármasok átlagos száma. Mivel maga az alapmódszer csak a betűk számának megfelelő eltolási lehetőséget kínál, a titok próbálgatással hamar kideríthető. Ha nem egyszerűen eltoljuk a betűket, hanem a helyettesítésnél valamilyen véletlenszerű sorrendet használunk, az a próbálgatásos feltörést kissé nehezebbé teszi (a lehetőségek száma: hozzávetőleg a 25 faktoriálisa).

	Kis ábécé szóköz nélkül	Kis ábécé szóközzel	Ékezetes betűk szóköz nélkül
A	11,55	10,07	9,35
Á	-	-	3,72
B	2,38	2,12	1,72
C	0,63	0,54	0,60
D	1,79	1,42	1,71
E	14,26	11,86	9,71
É	-	-	3,87
F	0,94	0,83	0,88
G	3,22	2,87	3,55
H	1,68	1,37	1,23
I	5,48	4,84	4,39
J	1,05	0,90	1,21
K	5,84	5,26	5,35
L	6,23	5,44	6,30
M	3,65	3,29	3,92
N	5,47	4,69	5,47
O	6,87	6,04	4,47
Ö	-	-	2,14
P	1,09	0,88	1,04
Q	0,00	0,00	0,00
R	3,76	3,23	4,22
S	5,89	5,10	6,57
T	7,35	6,12	7,87
U	2,47	2,24	1,29
Ü	-	-	0,93
V	1,66	1,42	1,81
W	0,00	0,00	0,00
X	0,02	0,01	0,01
Y	1,92	1,64	2,21
Z	4,79	4,14	4,46
Szóköz	-	13,70	-

A magyar nyelv betűgyakoriságai 10 000 betűs újságszöveg alapján

Ezt a módszert nevezik egyábécés titkosítási módszernek. A módszer használata mellett a küldő és a fogadó oldaláról a hozzárendelési táblázatnak ismertnek kell lennie (ebben az esetben a táblázat a kulcs). A módszer hibája, hogy nem fedi el a nyílt szöveg szabályszerűségeit. Mivel minden betűírásos nyelvben meghatározhatók a betűk előfordulási valószínűségei, a titokfejtőnek nem kell próbálgatással piszmognia. Nem kell mást tennie, mint meghatározni az egyes betűk (jelek) számát a kriptoszövegben, és azok előfordulási arányai alapján a titkosítási táblázat könnyen létrehozható.

De Vignére többábécés módszere

Többek közt a fent ismertetett eljárás néhány továbbfejlesztett változatát ismertette ***Blaise De Vigenére*** a középkorban. Az ő eljárásának az a lényege, hogy a nyílt szöveg betűinek helyettesítése a szövegben elfoglalt helyüktől is függ. Ehhez egy betűtáblázatot állít fel, amely egymás alatt az újra és újra eggyel eltolt ábécét tartalmazza. Az eljárás működési elvét a *2. ábra* és az *1. listán* látható egyszerű kis Perl-program szemlélteti. A nyílt szöveg minden betűjének egy jelszóbetűt feleltet meg. Amennyiben a jelszó rövidebb, mint a titkosítandó szöveg, a jelszó ismételgetésével nyújtja meg a megfelelő hosszúságúra. A kriptoszöveg egy betűjét a táblázatban a nyílt szöveg megfelelő betűje és a hozzá tartozó jelszóbetű határozza meg. Ez a gyakorlatban azt jelenti, hogy a jelszó betűi határozzák meg, hogy az adott betű melyik eltolási táblázat szerint lesz módosítva. Ez az eljárás kismértékben képes elfedni a nyílt szöveg szabályszerűségeit, de megadott felhasználása sem védett a statisztikai alapú támadások ellen. Egyértelmű, hogy ha a titkos kulcsszót ismételgetve használjuk a betűk eltolásának meghatározására, akkor a betűk minden ismétlésnél a korábbival megegyezően lesznek eltolva. Ha tehát meg tudnánk határozni a titkos jelszó hosszát, akkor minden periódusra meghatározhatnánk a korábbiakban említett betűvalószínűségeket, így a nyílt szöveg – és ezáltal a jelszó – minden egyes betűjét meg tudnánk határozni. Ennek kivitelezése a következő: mivel a módszer a betűk előfordulási valószínűségét a kriptoszövegben egyenletesebbé teszi, ha meg tudnánk határozni, hogy a szöveg milyen periódusai térnek el leginkább az egyenletestől – tehát mely szakaszok mutatják a leginkább a természetes nyelv jellegzetességeit –, akkor megtudnánk a jelszó hosszát. Rendre meghatározzuk minden második, harmadik, negyedik stb. betű csoportjának betűeloszlását, és amelyik a legjobban egyezik a természetes nyelv betűeloszlásával, annál ellenőrizzük, hogy a periódus minden betűcsoportja a magyar nyelv betűvalószínűségeit hozza-e. A fenti elemzés csak akkor tud a megfelelő hatékonysággal működni, ha elegendő kriptoszöveg áll a rendelkezésünkre. Nagy előnye, hogy nemcsak ezt a klasszikus eljárást lehet hatékonyan törni vele, hanem egy általánosabb módszert is. Ha ugyanis a titkosító táblázat soraiban nem eltolt, hanem kevert betűk vannak, az egyes sorok betűsorrendjét is meg kell találnunk. Ezt pedig a jelszó minden betűjére el kell végeznünk. Ennél a módszernél lényegesen jobb védettséget ad, ha az első, a jelszóval titkosított blokkot használjuk fel mint a soron következő blokk jelszavát és így tovább. Nagyon hasonló alapelven működnek a modern szimmetrikus algoritmusok is.

Mini titkosítónk

Az 1. listán látható példaprogram nagyon egyszerű megvalósítása a fent tárgyalt De Vignére-féle algoritmus egy formájának. Az egyszerűség kedvéért csak a betűket titkosítja vagy fejti vissza, az egyéb karaktereket figyelmen kívül hagyja. További egyszerűsítés, hogy minden betűt kisbetűssé alakít és az ékezetes betűket ékezetmentesekre cseréli le. Használata nagyon egyszerű. A program a `-p` kapcsoló után várja a titkosító jelszót, ha mást nem adunk meg, akkor titkosít. Ha a `-d` kapcsolót is megadjuk, akkor a megadott jelszó felhasználásával visszafejti a kriptoszöveget. A nyílt vagy kriptoszöveget a szabványos bemenetén várja, és a szabvány kimenetén jelenik meg az eredmény. Ez az egyszerű program így elég karcsú, de kis módosítással tökéletes titkosító készíthető belőle. Elegendő úgy átírni, hogy a jelszót egy állományból vegye. Az állományban lévő jelszónak természetesen meg kell felelnie a korábban leírt feltételeknek. További érdekesség, hogy ha egyetlen betűs

titkosító jelszót választunk, a korábban leírt Caesar-féle eljárást kapjuk, mivel minden betűt ugyannyival tolunk el.
A program használatával lelkes olvasóink érdekes feladat elé állíthatják magukat: írjanak programot a korábban ismertetett fejtési eljárás önműködővé tételére. Ennek megkönnyítésére *táblázatunkban* közöljük a magyar nyelv betűgyakoriságait, a következő forrás alapján: [fulop]. Vájt fülűek természetesen maguk is elkészíthetik a betűgyakorisági táblázatot az Interneten található bőséges magyar nyelvű írott anyag és egy egyszerű kis program segítségével. Az ***Igazi Perl-guru*** így oldaná meg:

```
perl -ne '@l=split(//);

foreach(@l){if(/[[:alpha:]ØÆ    ıœß `ˇ       ]/)
➥{print"$_\n";}}' nagy_magyar_szoveg.txt |
    ➥sort | uniq -c > atya_betugyakorisag.txt
```

A tökéletes titkosítás

A klasszikus Vigénere-féle módszerrel az a baj, hogy a jelszó adott hosszúságú, így a támadó a periodikusságot kihasználva megfejtheti a titkot. A feladat megoldása egyszerű, mint a pofon. Minden üzenet továbbítására egyszer használatos jelszót használjunk, amely hosszát tekintve nem rövidebb, mint maga az üzenet, és betűi semmilyen módon nem függenek egymástól. Ez az egyszerű, mégis zseniális megoldás 1917-ben született meg ***Gilbert Vernam*** fejében. A szakirodalomban ***véletlen átkulcsolásnak*** (one time pad) vagy Vernam-féle titkosítóeljárásnak (Vernam cipher) hívják, és húsz esztendővel később matematikai módszerekkel is bebizonyították, hogy valóban nagyon jó. Sőt: tökéletes. Ezt a titkosítást semmilyen módszerrel nem lehet feltörni, ugyanis a véletlen jelszó olyan szinten függetleníti a kriptoszöveget a nyílt szövegtől, hogy köztük semmilyen összefüggés nem tárható fel. Van azonban néhány bökkenő.

A véletlenszámokról

A feltételeknek megfelelő kulcsok előállítása egyáltalán nem egyszerű, hiszen véletlen számok olyan sorozatát kellene előállítani, amelyek függetlenek egymástól. Mindezt általában számítógéppel, amely – ebből a szempontból – túlzottan szabályos működésű szerkezet. A megfelelő véletlen számok, véletlenszám-sorozatok előállítása a titkosítás nagyon sok modern alkalmazásánál elengedhetetlen, így a titkosítás elmélete komoly figyelmet szentel ennek a témakörnek [rand]. A Unix-rendszerek általában úgy segítik e kérdés megoldását, hogy a felhasználói, hálózati és egyéb tevékenység véletlenszerű adatait gyűjtik, és két különleges eszközön keresztül elérhetővé teszik. Ezek a ***/dev/random*** és a ***/dev/urandom***. A kettő közti különbség, hogy a `random` nagyon jó minőségű véletlen adatokat ad, de a teljesítménye behatárolt. Addig nem lehet véletlen adatokat kiolvasni belőle, amíg nincs elegendő véletlen forrásadat (például felhasználói billentyűleütések). Így ezt nem célszerű olyan alkalmazásnál használni, ahol nagy mennyiségű véletlenszámra van szükség, és a véletlenszám-előállító időszakos bedugulása az alkalmazás működésképtelenségét okozhatja. Ha nagy mennyiségű véletlenszámra van szükség, az `urandom` eszköz használható.
Sajnos, itt is igaz az általános igazság: a mennyiség növelése szinte minden esetben a minőség romlásával jár. Az `urandom` eszköz nem tud kifogyni.
Az általa szolgáltatott véletlen számok esetében azonban nem lehetetlen, hogy valamilyen külső körülménnyel (például szélsőséges méretű, állandó hálózati terhelés) befolyásolhatóak. Ezért csak ennek a lehetőségnek a mérlegelése után szabad felhasználni. Abban az esetben, ha a kissé bizonytalanabb minőségű, de nagy teljesítményű véletlenszám-előállító nem felel meg, a kereskedelemben is kapható eszközök használhatók. Ezek az egyedi eszközök éppen erre a feladatra vannak kifejlesztve – valamilyen fizikai folyamat véletlenszerűségén alapulva állítanak elő nagy mennyiségű, jó minőségű véletlen számot.

Még egy kis véletlen átkulcsolás, zárszavak

Tételezzük fel, hogy előállítottuk a megfelelő tulajdonságokkal bíró kulcsot. Mind a küldőnek, mind a fogadónak a kulcs birtokában kell lennie. A kulcs továbbítása azonban ugyanazt a gondot veti fel, mint az eredeti üzeneté. Meg kell találni hozzá a megfelelően biztonságos csatornát. Így hiába a tökéletes titkosítási módszer, ha a gyakorlatban nehezen használható. Talán egyetlen dologra jó lehet: ha a nyílt szöveget visszük át véletlen átkulcsolással, a kulcsot pedig valamilyen más algoritmussal titkosítjuk, némi védelmet nyerünk, mivel a véletlen kulcsnak nincsenek statisztikai módszerekkel támadható sajátosságai.
A cikksorozat további részeiben szót ejtünk a legfontosabb szimmetrikus (egykulcsos) titkosító algoritmusokról, a DES-ről és leszármazottairól, valamint az új szabványnak tekinthető AES-ről. Utána kerül sorra az aszimmetrikus csapatból a ***Diffie-Hellman*** javasolta módszer, az RSA és a DSA algoritmus.

Irodalomjegyzék

Néhány fontosabb könyv és honlap a téma behatóbb tanulmányozásához:

[kahn] Kahn, David: The codebreakers, McMillan, New York, 1996; ISBN: 0684831309
[schneier] Bruce Schneier: Applied Cryptography (second edition), John Wiley & Sons, 1995; ISBN: 0471117099
[cryptweb1] Cryptography and Information Security Group
➲ http://theory.lcs.mit.edu/~cis/homepage.html
[cryptweb2] John Savard: Cryptographic Compendium
➲ http://home.ecn.ab.ca/~jsavard/crypto/entry.htm
[rand] Internet RFC 1750 – A véletlenszám–generálásról
[krithun1] Ködmön József: Kriptográfia, ComputerBooks, 1999; ISBN: 9636182248
[krithun2] Nemetz Tibor és Vajda István: Algoritmikus Adatvédelem, Akadémiai Kiadó, 1991; ISBN: 9630560933
[fulop] Fülöp Géza: Az információ, ELTE Könyvtártudományi – Informatikai Tanszék, 1996
➲ http://www.mek.iif.hu/porta/szint/tarsad/konyvtar/informat/azinform/

Mátó Péter (atya@andrews.hu), informatikus mérnök és tanár. Biztonsági rendszerek ellenőrzésével és telepítésével, valamint oktatással foglalkozik. 1995-ben találkozott először linuxos rendszerrel. Ha teheti, kirándul vagy olvas.

Borbély Zoltán (bozo@andrews.hu), okleveles mérnök-informatikus. Főként Linuxon futó számítógépes biztonsági rendszerek tervezésével és fejlesztésével foglalkozik.
A 1.0.9-es rendszermag ideje óta linuxozik. Szabadidejét barátaival tölti.