GnuPG tippek trükkök

Kevesen tudják, de a GnuPG a levelek titkosításánál és visszafejtésénél valójában sokkal többre is képes...

Különösen a kezdő felhasználók gyakran érzik úgy, hogy a kriptográfia határozottan érdekes és hasznos téma ugyan, de egyben riasztóan összetett is. Eleve számos különböző szoftvercsomag létezik, aztán ott vannak a jelszavak, a kulcsok, a kulcskarikák, a tanúsítványok és digitális ujjlenyomatok. Az egész egy nagy zűrzavar. Azt már kevesebben sejtik, hogy nincs szükségünk az összes fura nevű dologra ahhoz, hogy használni tudjuk a titkosítási lehetőségeket. A *GnuPG*-vel gyakorlatilag minden különösebb előismeret nélkül is végezhetünk titkosítást, sőt könnyen lehet, hogy még a telepítésével se kell fáradnunk, mert a legtöbb terjesztésnek alapértelmezésként is része.

A GnuPG és az OpenPGP

A *GnuPG* az *OpenPGP* nyílt forrású (*GNU Project*) megvalósítása, amit *GNU Privacy Guard* néven is ismernek. A *GnuPG* egy meglehetősen kifinomult, nyilvános kulcsokon alapuló titkosítási rendszer, amelynek több mint 70 parancssori kapcsolója van, sőt rendelkezik egy belső parancssorral és egy menüvezérelt környezettel is. Számos különböző operációs rendszerre lefordítható és eleve számos rendszerhez létezik bináris csomagja is. Ezek letöltéséhez látogassunk el a *GnuPG* hivatalos weblapjára (lásd az elektronikus forrásokat). Akár a többi *GNU* szoftvert, ezt a csomagot is szabadon használhatjuk, hiszen ugyanúgy a *GNU General Public License* vonatkozik rá.
Az *RFC 2440*-ben rögzített *OpenPGP* szabvány *Phil Zimmermann* 1991-ben közzétett *Pretty Good Privacy* rendszere alapján készült. Ennek a leírásnak számos más, kereskedelmi termék titkosítási rendszer is megfelel, vagyis nyugodtan nevezhetjük általánosan elfogadottnak. Ami azt illeti, a jelenleg használatban levő titkosítási rendszerek között az *OpenPGP* rendszerek a leggyakoribbak.

Vágjunk bele!

Először ismerkedjünk meg a *GnuPG* néhány olyan szolgáltatásával, amelyekhez nincs szükség jelszóra. Ez után majd kitalálunk egy kiváló jelszót, és titkosítunk is vele valamit. Elöljáróban érdemes megjegyezni, hogy bár magát a rendszert *GnuPG*-nek hívják, az elindításához szükséges parancs a `gpg`.
A következő paranccsal győződjünk meg róla, hogy a *GnuPG* telepítve van rendszerünkön, illetve hogy végrehajtható állományának helyes szerepel az alapértelmezett útvonalak között:

```
gpg --version
```

Valami ilyesminek kell megjelennie a képernyőn:

```
gpg (GnuPG) 1.4.1
Copyright (C) 2005 Free
➥Software Foundation, Inc.
This program comes with
➥ABSOLUTELY NO WARRANTY.
This is free software, and you
➥are welcome to
redistribute it under certain
➥conditions.
See the file COPYING for
➥details.
Home: ~/.gnupg
Supported algorithms:
Pubkey: RSA, RSA-E, RSA-S,
➥ELG-E, DSA
Cipher: 3DES, CAST5, BLOWFISH,
➥AES, AES192, AES256, TWOFISH
Hash: MD5, SHA1, RIPEMD160,
➥SHA256, SHA384, SHA512
Compression: Uncompressed, ZIP,
➥ZLIB, BZIP2
```

A változatszám, a dátum és egyes más részletek természetesen eltérhetnek, de a kimenet nagyjából ilyen. Az ebben a cikkben bemutatott példáknak elvileg működniük kell a *GnuPG* legfrissebb, és minden jövőben megjelenő változatával is. Nos, akkor adjuk ki a következő parancsot:

```
gpg /dev/null
```

Erre a következő kimenetet kapjuk:

```
gpg: /home/you/.gnupg:
➥directory created
gpg: new configuration file
➥`/home/you/.gnupg/gpg.conf'
➥created
gpg: WARNING: options in
➥`/home/you/.gnupg/gpg.conf'
➥are not yet active during
➥this run
gpg: keyring `/home/you/.gnupg/
➥secring.gpg' created
gpg: keyring `/home/you/.gnupg/
➥pubring.gpg' created
gpg: processing message failed:
➥eof
```

A figyelmeztetés és ez a rengeteg tájékoztató adat teljesen normális, ha a *GnuPG*-t először futtatjuk. Ha valakinek mégsem ez jelenne meg a képernyőjén, annak sem kell semmitől tartania. Egyszerűen csak arról van szó, hogy a jelek szerint korábban már futtatta ezt a parancsot, és az létrehozta a *.gnupg* könyvtárat.

Bináris fájlok kódolása

A legtöbb e-mail program támogatja a csatolt fájlok küldését, bár a csak parancssorból működtethetők, mint például a `/bin/mail` történetesen nem. Ez néha nem is különösebben nagy baj, mert lehetnek olyan esetek, amikor célszerű minden adatot a levél törzsébe tenni. Csakhogy a leveleket továbbító hálózat nem képes megbirkózni bináris fájlokkal, így azokat előbb *ASCII* adatokká kell átalakítani. (Ha nem így teszünk, garantáltan hibás lesz az átvitel.) Talán sokan próbálták már a `uuencode` programot, és sokan szembesültek az összetettségével, vagy azzal, hogy néha egyszerűen nem működik. Ráadásul nincs is minden rendszerben parancssorból működtethető *MIME* kódoló. Ilyenkor jöhet jól a *GnuPG*, amelynek – kissé talán meglepő módon – van ilyen szolgáltatása. Ráadásul ez a funkció működését tekintve nagyon hasonló a *MIME* kódolókhoz, de a használata teljesen egyszerű és problémamentes. Ha *ASCII* adatokká akarunk alakítani egy fájlt, gépeljük a következőt:

```
$ gpg --enarmor < filename.bin
➥> filename.txt
```

A kódolt tartalmat a következő paranccsal lehet kibontani:

```
$ gpg --dearmor < filename.txt
➥> filename.bin
```

Figyelem! Bár a *GnuPG* alapvetően egy titkosításra szolgáló program, az *OpenPGP*-vel előállított *ASCII* fájlok semmiféle biztonsági védelemmel nem rendelkeznek. Ezen a helyzeten persze könnyen változtathatunk, amint az hamarosan ki is fog derülni.

Ellenőrzőösszegek hatékonyabban

Mit tegyünk, ha azt gyanítjuk, hogy egy épp most kapott vagy letöltött bináris fájl hibás. Ilyenkor általában a `sum` vagy `cksum` programokat szokás használni az átvitel előtt és után is. A kimenetek egyszerű összehasonlításával viszonylag nagy biztonsággal eldönthető, hogy sikeres volt-e az átvitel. Ugyanakkor sajnos ezeknek a programoknak három különböző, egymással nem kompatibilis változata létezik, sőt az is megeshet, hogy ugyanaz a változat különböző összeget számol ki ugyanabból a bemenetből különböző gépeken futtatva. Ezt utóbbi jelenséget egyébként az eltérő bájtsorrend (endianness) szokta okozni. És ami a legrosszabba az egészben az az, hogy bizonyos eseteben a fenti két program nem is képes kimutatni a hibát. A `sum` és a `cksum` programok kimenete egyaránt mindössze 32 bites, ami egyszerűen túl kevés a megbízhatósághoz. Simán előfordulhat, hogy bár egyezik a változatszám és nincs eltérés a két végpontok működő architektúrák között sem, mégis különböző bemenetekre ugyanazt a kimenetet kapjuk. Az *SSH v1* rendszer elleni „népszerű” *CRC-32* kompenzációs támadást éppen ez a probléma teszi lehetővé.
Minderre az első lehetséges megoldás az `md5sum` parancs használata, amelynek azonban szintén megvannak a maga problémái. A legbosszantóbb talán az, hogy a különböző változatok mind egy kicsit máshogy formázzák a kimenetet, máshogy adják meg a fájl nevét, vagy más módon jelenítik meg a hexadecimális értékeket. Ezek sajnos nem könnyítik meg az automatikus hibaszűrést, hiszen a `diff` parancs ezektől az eltérésektől akkor sem tud simán lefutni, ha amúgy nincs semmi gond. A dolgot csak tetézi, hogy az `md5sum` által használt *MD5* hasítóalgoritmusnak is vannak ismert sebezhetőségi pontjai. És akkor még nem is említettük azt a triviális lehetőséget, hogy a rendszergazda egyszerűen elfelejtette telepíteni a programot.
A *GnuPG*-vel az összes fent említett probléma megoldható, hiszen ez operációs rendszertől és változatszámtól függetlenül mindig ugyanazt a kimenetet adja ugyanarra a bemenetre. A *GnuPG* ráadásul támogatja az újabb és ezért biztonságosabb algoritmusokat is:

```
$ gpg --print-md sha1 filename
filename: E83A 42B9 BC84 31A6
➥6450  99BE 50B6 341A 35D3
➥DCEB
```

Megadhatjuk egyszerre több fájl nevét is:

```
$ gpg --print-md sha1 *.txt
test.txt: E0D6 3F44 4253 CED5
➥9205  4047 4AA6 4E0F FD0F
➥130D
test2.txt: 32AC 34F9 B7AF 1972
➥C015  E5EE 456E 89BD CC3C
➥7246
```

Ha valamiért mégis az *MD5* algoritmust szeretnénk használni, az se gond, mert a program ezt is ismeri:

```
$ gpg --print-md md5 filename
filename: 26 E9 85 5F 8A D6 A5
➥90  6F EA 12 12 83 C7 29 C4
```

A *GnuPG* újabb változatai támogatják az olyan újabb, kiemelten biztonságos hasítóalgoritmusokat is mint a *SHA-512*:

```
$ gpg --print-md sha512
➥filename
filename: FC37410D 9336DD60
➥22AEB6A2 A42E82F1 2EA3470D
➥4982E958 B35C14A0
          CF381CD2 3C4CBA35
          ➥BE5F11CB 05505ED2
          ➥DBF1C7A0 397EFF75
          ➥007FAEBB
          30B43B30 6514990D
```

Apropó a fenti kimeneteket és a `--print-md` példákat egész egyszerűen ellenőrizhetjük saját gépünkön is: Hozzunk létre egy egysoros fájlt, ami a `The Linux Journal` szöveget tartalmazza, és ezt adjuk meg bemenetként a *GnuPG*-nek.
A hash értékeknek pontosan meg kell egyezniük a bemutatottakkal.

Gyors és egyszerű titkosítás

Aki titkosítani akar egy fájlt, de nem tudja hogy fogjon hozzá, annak valószínűleg jól jön a következő, *GnuPG*-re alapozott gyorstalpaló:

```
$ gpg -c test.txt
Enter passphrase:
Repeat passphrase:
```

Kódolásnál a *GnuPG* kétszer bekér egy jelszót, pont ugyanúgy, mint amikor a bejelentkezési jelszavunkat átállítjuk. Az új, kódolt tartalmat egy ugyanolyan nevű fájl fogja tartalmazni, amelynek azonban immár *.gpg* kiterjesztése lesz. Az eredeti fájl érintetlen marad.
A `-c` kapcsoló a szokványos (conventional) titkosítást jelöli, amit más néven szimmetrikus titkosításnak is szoktak nevezni. A *GnuPG* alapértelmezett titkosítási módszere a nyilvános kulcsú architektúra használata, de mi egyelőre nem állítottunk elő egyetlen kulcspárt sem, így most kénytelenek leszünk az egyszerűbb módszert alkalmazni.

1. táblázat *A különböző szerkezetű és hosszúságú jelszavak erőssége (összehasonlítási alap a feltöréshez szükséges becsült idő)*

Típus	Hosszúság	Bitek száma	Bitek teljes száma	A feltöréshez szükséges idő
Egyetlen szó bármilyen nyelven	8 karakter	24	24	Másodpercek
Véletlenszerűen kiválasztott karaktersorozat (csak egyféle betű)	8 karakter	4.7	37	Percek
Véletlenszerűen kiválasztott karaktersorozat (csak egyféle betű)	16 karakter	4.7	75	Évtizedek
base64 [A-Za-z0-9+/=]	10 karakter	6	60	Hónapok
base64 [A-Za-z0-9+/=]	20 karakter	6	120	Törhetetlen?
Teljesen véletlenszerű nyomtatható karakterek	6 karakter	6.5	40	Percek
Teljesen véletlenszerű nyomtatható karakterek	8 karakter	6.5	52	Órák
Completely random printable	12 karakter	6.5	78	Évtizedek
Teljesen véletlenszerű nyomtatható karakterek	15 karakter	6.5	97	Évszázadok
Teljesen véletlenszerű nyomtatható karakterek	20 karakter	6.5	130	Törhetetlen?
Diceware jelszó	2 szó	12.9	26	Másodpercek
Diceware jelszó	4 szó	12.9	51	órák
Diceware jelszó	6 szó	12.9	78	Évtizedek
Diceware jelszó	8 szó	12.9	120	Törhetetlen?

Ez a titkosítási módszer egyébként akkor a leghasznosabb, ha csak mi magunk akarjuk visszafejteni a kérdéses tartalmat, de nem bízunk abban, hogy az a hely, ahol azt tároljuk, biztonságos. A könnyen elveszíthető vagy ellopható tárolóeszközök tartalmát például célszerű szimmetrikus kódolással titkosítani. Szintén hasznos lehet ez a védelem a telephelyen kívül tartott biztonsági másolatok védelmére.
A kódolt fájl visszafejtéséhez adjuk ki a következő parancsot:

```
$ gpg filename.gpg
```

A *GnuPG* automatikus detektálja, hogy szimmetrikus kódolással titkosított tartalomról van szó, és magától rákérdez a jelszóra. A visszafejtett adatokat egy ugyanolyan nevű fájlba írja de levágja a *.gpg* kiterjesztést. Akárcsak a kódolásnál, az eredeti fájl most is érintetlen marad. Ha a kimenetet egy eltérő nevű fájlba, vagy más helyre szeretnénk irányítani, használjuk a szabványos átirányítást ugyanúgy, mint a `--dearmor` kapcsoló esetében.

Fontos megjegyezni, hogy ilyenkor mind a bemenetet, mind a kimenetet átirányítással kell megadnunk, ellenkező esetben *GnuPG* összezavarodik:

```
$ gpg < filename.gpg >
➥filename.txt
```

Ha azt akarjuk, hogy a titkosított anyaghoz más is hozzáférjen, el kell árulnunk neki a kódolás során használt jelszót anélkül, hogy az kiszivárogna. Ennek a legegyszerűbb és legbiztonságosabb módja természetesen a személyes átadás. Erre most nyugodtan mondhatja valaki azt is, hogy ha már úgyis találkozunk az illetővel, akkor ezzel az erővel átadhatjuk neki magát a tartalmat is mindenféle titkosítás nélkül: Ne felejtsük el azonban, hogy ugyanazt a jelszót több alkalommal is használhatjuk, tehát a dolog nem föltétlen értelmetlen. Ugyanakkor a titkosításhoz használt jelszavakat időnként ugyanúgy le kell cserélni, mint a bejelentkezési kódot. Ezen kívül alapszabály, hogy soha nem használjuk ugyanazt a jelszót különböző emberekkel való kapcsolattartásra, hacsak nem akarjuk, hogy közülük bárki bármilyen általunk küldött tartalomhoz hozzáférhessen.
Van itt még egy talán nem lényegtelen megjegyzés. A következő figyelmeztető üzenet megjelenése teljesen normális, ha a *GnuPG*-t jelszavas (szimmetrikus) titkosításra használjuk:

```
gpg: WARNING: message was not
➥integrity protected
```

Ha nem akarjuk többé látni, használjunk nyilvános kulcsú titkosítást.

Jelszavak

A jelszó olyan titok, amely segít más dolgokat titokban tartani. Ebből következőleg a *GnuPG* teljes működési folyamatának egyik leglényegesebb az alkalmazott jelszó. Sajnos ez egyben azt is jelenti, hogy a rendszer legsebezhetőbb pontja is maga a jelszó. Ennek pedig egyszerűen az az oka, hogy igazán jó jelszavakat egyrészt nehéz előállítani, másrészt ha egy jelszó valóban jó, azt nehéz fejben tartani. Erősen javasolt a *Diceware* használata,

2. táblázat *A GnuPG e cikkben említett parancsainak rövid összefoglalása*

A kapcsoló rövid formája	Hosszú forma	Leírás
	`--version`	A változatszám és a támogatott algoritmusok kiíratása
	`--help`	Súgó
`-a`	`--armor`	ASCII kódolás bekapcsolása titkosítás közben
	`--enarmor`	Bináris bemenet ASCII kimenetté való átkódolása
	`--dearmor`	ASCII bemenet bináris kimenetté való visszakódolása
	`--print-md HASH`	Kivonat készítése az üzenetből a megadott hash alapján
`-c`	`--symmetric`	Hagyományos, szimmetrikus kulcson alapuló titkosítás jelszóval
`-o`	`--output`	A kimeneti fájl nevének megadása. A szabványos kimenetet - jelöli.

de ha ez valakinek nem tetszik, érdemes megnézni az elektronikus források között említett ***Wikipedia*** cikket, vagy rákeresni a interneten erős jelszavakat előállító weboldalakra. Függetlenül attól, hogy melyik módszer használata mellett döntünk, az alapvető irányelv azonos: a hosszabb jelszó jobb (lásd az *1. Táblázatot*). Látható, hogy az *1. Táblázat* adatai több nagyságrendet fognak át. Ennek alapvetően az az oka, hogy a törhetőség kérdésének két, nagyjából azonos jelentőségű összetevője van, amelyekkel szabadon lehet játszani: az idő és a pénz. A számítási teljesítmény – amint az közismert – egyre olcsóbb, így a titkosítások feltöréséhez szükséges idő egyre rövidül. Ami a költségeket illeti, bizonyos esetekben a dolog akár ingyen is megoldható, a felső határ azonban a „csillagos ég”. Mindettől függetlenül általánosságban kijelenthetjük, hogy ha valaki elfelejti a *GnuPG* jelszavát, az azzal kódolt adatokról feltehetőleg örökre lemondhat. A *GnuPG*-hez sem ismert hátsó kapu, sem a jelszó visszanyerésére szolgáló egyszerű módszer nem létezik. Ha valaki mégis belevág, a visszafejtéshez szükséges idő attól függ, mennyire jó jelszót választottunk. Egy igazán jó 20 karakterből álló jelszó feltörése úgy néhány milliárd évig tart, nemcsak a jelenleg elérhető gépeken, de valószínűleg a jövőben megjelenőkön is.

Jelszó előállítása

Van egy egyszerű trükk arra, hogy magával a *GnuPG*-vel állítsunk elő biztonságos jelszót. Ez persze nem lesz egy könnyen fejben tartható, vagy könnyen begépelhető darab, viszont garantáltan nagyon biztonságos lesz. Először egy 16 véletlenszerű bájtból álló sorozatot állítunk elő, majd ezt – ismét a *GnuPG*-t használva – base64 kódolásnak vetjük alá. Végül a `sed` segítségével levágjuk a kimenet fejlécét. Az így keletkezett karaktersorozatot nyugodtan használhatjuk jelszó gyanánt:

```
gpg --gen-random 1 16 | gpg
➥--enarmor | sed -n 5p
```

Kódolt tar fájlok előállítása

A *tar* archivumok tömörítésére a szokásos *gzip* helyett a *GnuPG*-t is használhatjuk. A végeredmény ebben az esetben egy körülbelül ugyanakkora fájl lesz, de a tartalom immár titkos. Apropó senkinek nem ajánlom, hogy titkosított fájlok tömörítésével töltse az idejét, az ilyen adatok ugyanis rendszerint tömöríthetetlenek. Ennek – erősen leegyszerűsítve – az az oka, hogy a tömörítés és a titkosítás matematikailag egymáshoz meglehetősen közel álló dolgok. Éppen ezért a legtöbb titkosító rendszer – és ez alól a *GnuPG* sem kivétel – a titkosítás előtt automatikusan tömöríti a kódolandó adatokat. Ráadásul ez valamelyest a biztonságot is növeli.

Archívum titkosításakor a rendszer ugyanúgy be fog kérni egy jelszót, mint amikor egy közönséges fájlt titkosítunk:

```
tar -cf - these files here |
➥gpg -c > these-files-here.tgp
```

A visszafejtéskor természetesen ugyanezt a jelszót kell majd megadnunk:

```
gpg < these-files-here.tgp  |
➥tar -xvf -
```

A GnuPG automatizálása

Ha a *GnuPG*-t egy szkripten belül akarjuk használni, és nem szeretnénk, ha minden egyes futtatásnál bekérné a jelszót, akkor rögzíthetjük azt egy szövegfájlban is (példánkban `passphrase.txt`), aminek a nevét adjuk meg paraméterként:

```
$ cat passphrase.txt | gpg
➥--passphrase-fd 0 -c <
➥filename.txt > filename.gpg
```

A visszafejtés majdnem ugyanígy megy, csak a `-c` kapcsolót kell használnunk, és értelemszerűen át kell írni a fájlneveket:

```
$ cat passphrase.txt | gpg
➥--passphrase-fd 0 <
➥filename.gpg > filename.txt
```

Ha a titkosított fájl e-mailben akarjuk elküldeni valakinek, esetleg egy a telephelyen kívüli címre, akkor érdemes a `-a` kapcsolót is használni, amivel bekapcsoljuk az ***ASCII*** kódolást. A végeredmény pontosan ugyanaz lesz, mint mikor korábban az `--enarmor` kapcsolót használtuk, de a tartalom most titkosítva is lesz. Kellemes mellékhatásként a fájlméret is kisebb lesz mint a `uuencode`-ot vagy ***MIME*** kódolást használva, hiszen a *GnuPG* – mint már említettem – automatikusan tömörít, mielőtt elvégezné a titkosítást. A dolog betetőzéseként a levélküldést magából a szkriptből is megoldhatjuk. Figyeljük meg, hogy ilyenkor szükség van a `-o` kapcsolóra, ami a *GnuPG*-t a szabványos kimenet használatára kényszeríti:

```
$ cat passphrase.txt | gpg
➥--passphrase-fd 0 -ac -o
➥- filename.txt | mail
➥user@example.com
```

Apropó a jelszónak egy szövegfájlban való elhelyezése meglehetősen veszélyes dolog. Bárki, aki képes ezt a fájlt megszerezni, vissza tudja fejteni az összes, ezzel titkosított anyagunkat. Mi több, az illető akár új, titkosított fájlokat is létrehozhat, amelyek megkülönböztethetetlenek lesznek a sajátjainktól. Mielőtt tehát alkalmazzuk az itt bemutatott módszert, győződjünk meg róla, hogy a jelszót tartalmazó fájl, illetve az egész gép kellően biztonságos helyen van.
A dolgok automatizálása eleve azzal jár, hogy a folyamatból kizárjuk az emberi tényezőt, így az imént említett biztonsági problémát igen nehéz kiküszöbölni. Persze a *GnuPG*-nek még erre is van megoldása, hiszen használhatunk nyilvános kulcsú titkosítást.

GnuPG gondok

Megeshet, hogy egyszer csak kapunk valakitől egy *OpenPGP*-vel titkosított fájlt, de nincs hozzá kulcskarika, így első közelítésben fogalmunk sincs, mit kellene vele kezdeni. Lehet, hogy aki küldte nekünk, úgy gondolta, hogy számunkra a dolog egyértelmű lesz, de tévedett, mert mégsem az. Aztán az sem kizárt, hogy a küldő se tudja, mivel mit is kellene tenni ahhoz hogy mi legyen.
Ha a fájlnak *.pgp* vagy *.gpg* a kiterjesztése, megpróbálhatjuk visszafejteni a *GnuPG*-vel. Belenézhetünk egy szövegszerkesztő segítségével is, és ellenőrizhetjük, hogy a tartalma hasonlít-e a következőre:

```
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.2.5
➥(GNU/Linux)
jA0EAwMCwg21r1fAW+5gyS0KR/bkeI8
➥qPwwQo/NOaFL2LMXEYZEV9E7PBLjj
➥Gm7Y
DGG4QnWD5HSNOvdaqXg=
=j5Jy
-----END PGP MESSAGE-----
```

Ha igen, akkor egy *ASCII* kódolású *PGP*-vel titkosított fájlunk van.
A fenti példában a titkosított tartalom ugyanaz, mint amit a `--print-md` kapcsoló bemutatásakor használtunk, és a jelszó is azonos az ott használttal. Abból is számos hasznos dolgot leszűrhetünk, ha egyszerűen csak lefuttatjuk a *GnuPG*-t egy ismeretlen bemeneti fájllal. Ha jelszót kér a program, akkor például biztos, hogy titkosított tartalomról van szó, tehát vagy megszerezzük, vagy kitaláljuk azt, különben semmit nem tudunk vele kezdeni:

```
gpg unknown_file
```

Ha a kapott anyag nem is *OpenPGP* fájl, csak úgy néz ki, a következő üzenetet kapjuk:

```
gpg: no valid OpenPGP data
➥found.
gpg: processing message failed:
➥eof
```

Az ettől eltérő üzenet arra utalhat, hogy a fájlt egy olyan nyilvános kulccsal titkosították, aminek nem rendelkezünk a titkos párjával. Aztán az is előfordulhat, hogy a fájl egyszerűen sérült. Az egyik leggyakoribb hiba az, amikor bináris tartalmat küldenek kódolás nélkül e-mailben, vagy *FTP*-n *ASCII* módban töltik le azt.
A *GnuPG*-nek egy speciális diagnosztikai funkciója is van, ami segít a hibák kiszűrésében. Egy *OpenPGP* üzenet belül csomagokra oszlik, amelyekről a `--list-packtets` kapcsolóval információt is kérhetünk:

```
gpg --list-packets
➥unknown_file.gpg
```

A szabványos információ mellett ez a kapcsoló kiíratja a titkosításhoz használt nyilvános kulcs teljes azonosítóját (már amennyiben így titkosították a fájlt), valamint a titkosító algoritmus nevét is. Előfordulhat, hogy az anyagot a *PGP 2.x* nyilvános változatával titkosították (ezt szokás hagyományos kulcsnak is hívni). Sajnos a *PGP 2.x*. Nem felel meg mindenben az *OpenPGP* szabványnak, így a *GnuPG* nem tudja visszakódolni az így titkosított üzeneteket. A *PGP* legtöbb megvalósítása – különösen az utóbbi években készítettek – megfelel az *OpenPGP* szabványnak, így ha ilyesmivel találkozunk, általában az is elegendő, ha megkérjük a küldő felet, hogy mentse a tartalmat *OpenPGP*-vel kompatibilis formában, és küldje el újra.
Az *OpenPGP* szabvány ezen kívül többféle titkosító algoritmust támogat, amelyek közül egyesek esetleg nincsenek megvalósítva a *PGP* bizonyos változataiban. Hogy saját rendszerünkkel milyen algoritmusok használhatók, azt a

```
gpg --version
```

paranccsal jeleníthetjük meg.
A csomagformátummal és az algoritmusok belső számozásával kapcsolatos részleteket megtaláljuk az *RFC 2440*-es dokumentumban.
A *GnuPG* legtöbb kapcsolójának csak hosszú formája létezik, de néhol használhatunk rövid, egy betűs alakot is. Ezzel akadhat némi probléma is, ugyanis az eredeti PGP program egybetűs parancsai nem mindig ugyanazt jelentik, mint a *GnuPG*-nél.
A `-v` például az egyik helyen a `--verbose`, míg a másikon a `--version` rövidítése.

Hogyan tovább?

A *GnuPG* általános funkciói több helyen is egészen jól össze vannak foglalva. A hozzáférhető leírások közül az egyik legjobb a *GnuPG MiniHOWTO*, amit *Brenni de Winter* írt, és megtalálható a *GnuPG* webhelyén. Ez és még számos más dokumentum is bemutatja, miként használhatjuk a *GnuPG* általános, nyilvános kulcsokon alapuló szolgáltatásit.
A *GnuPG* levelezési listája szintén kiváló információforrás lehet, sőt a *GnuPG* webhelyén a teljes archívumát is megtaláljuk. Ezen a listán amúgy *Werner Koch*, a *GnuPG* vezető fejlesztője is gyakran publikál érdekes dolgokat.

Linux Journal 2006. 143. szám

Tony Stieber
UNIX rendszerekre, kriptológiára és fizikai biztonságra szakosodott információbiztonsági szakértő. 1999 óta foglalkozik Linuxszal, a UNIX-szal pedig 1987-ben találkozott. Számítógépet már 1980 előtt is látott. Ezzel együtt nem igazán tudja, mit hoz majd a következő évtized.

KAPCSOLÓDÓ CÍMEK

A cikkhez tartozó elektronikus források a következő helyen találhatók:
➲ www.linuxjournal.com/article/8743