Levelezés titkosítása a GPG segítségével

Szabad szoftver keretrendszer

Készítette a Közigazgatási és Igazságügyi minisztérium E-közigazgatási
Szabad Szoftver Kompetencia Központja
Budapest, 2013

Kódszám: EKOP–1.2.15

Tartalomjegyzék

Levelezés titkosítása a GPG segítségével

Bevezetés

Miért is merülhet fel az igény a levelek titkosítására? A válasz roppant egyszerű: bizonyos munkakörökben (tipikusan ilyen a rendszergazda, programozó, vagy bárki aki hozzáféréseket, vagy csak bizalmas adatokat szeretne e-mailben közvetíteni) előfordulhat az, hogy külsős szakértővel, vagy csak külsős e-mail címmel rendelkezővel kell olyan adatot egyeztetni, vagy belsős konfigurációt küldeni, amely esetleges nyilvánosságra kerülése esetén veszélyezteti az adatbiztonságot. Jellemzően ilyen adat lehet egy megfelelő jelszó- és felhasználónév-páros, amelyet gyakorlatilag bevett szokás szerint sima titkosításmentes e-mailben küldünk, vagy egy tűzfalbejegyzés, vagy -konfiguráció. A legtöbb esetben már a küldés során kompromittálódhat a levél és a benne lévő adat, hiszen az MTA[1]-k a legtöbb esetben sima titkosításmentes közegben cserélnek adatot egymás között (bár lehetnek kivételek az erre vonatkozó RFC szerint[2]). Ha a felhasználó már IMAPS vagy POP3S segítségével (azaz SSL vagy TLS titkosítást használva) éri is el a leveleit, és a küldő fél is használja az SSL küldés lehetőségét (amennyiben ez adott), akkor is a köztes kommunikáció és a legtöbb esetben a köztes tárolás (Mbox, Maildir) is mások által hozzáférhető területen történik. Így tehát amíg nem titkosítunk, addig még a saját területileg illetékes szerverünkön tárolt levél hitelességét sem tudhatjuk biztosan (ilyen eset az, amikor a levelezőszerver házon belül van, és mind a küldő, mind a fogadó házon belül levelezik). Így tehát indokolt, hogy főként a kiemelt esetekben, de célszerűen úgy általában a levelezésünk tekintetében is bevezessük a legelterjedtebb és legrégebben jelenlévő

[1] http://en.wikipedia.org/wiki/Message_transfer_agent
[2] http://tools.ietf.org/html/rfc2476

MAGYARORSZÁG MEGÚJUL

A projekt az Európai Unió támogatásával, az Európai Regionális Fejlesztési Alap társfinanszírozásával valósul meg.

nyílt kulcsú alkalmazást, a GPG[3]-t. A GPG a népszerű PGP program egyik utódverziója, amely természetesen szabadon használható. A PGP-t Phil R. Zimmermann bocsájtotta útjára 1991-ben, azzal a céllal, hogy legyen az emberek kezében egy olyan kódoló algoritmus, amelyet még az NSA[4] sem képes visszafejteni. Ez a törekvése olyan jól sikerült, hogy az FBI hosszú éveken keresztül zaklatta, ugyanis az USA jogszabályai szerint a kódoló és dekódoló programok készítése egy megítélés alá esik a fegyverkereskedelemmel, és ezek exportja eleve fegyverkereskedelemnek számít. Kezdetekben Zimmermann az RSA és a DES algoritmusait választotta, azonban az RSA-tól nem kért hozzájárulást vagy vásárolt licencet. Később ez visszaütött, hiszen az RSA beperelte, majd látva a PGP hihetetlen térhódítását és a felhasználók számának több millióra való gyarapodását, elállt a perektől. Később a PGP-t megvásárolták, és készült belőle csak az USA területén használható és nemzetközi változat is. Ma számos kereskedelmi termék alapját képezi. A GNU GPG, mint a nevében is szerepel, GNU alapokon használja fel annak a tudásnak a java részét, amelyet Phil R. Zimmermann a PGP kódjának megnyitásával tett lehetővé.

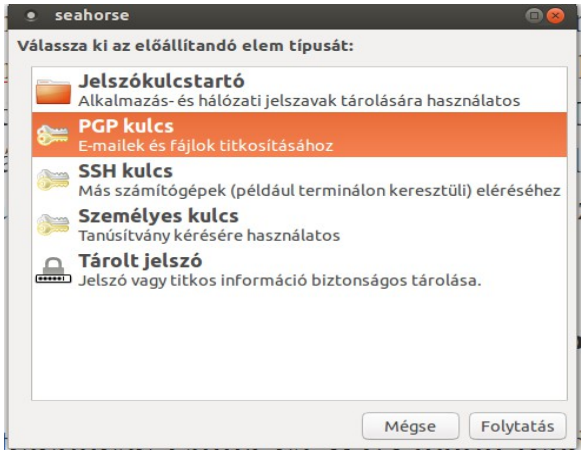
A GPG általános tudása

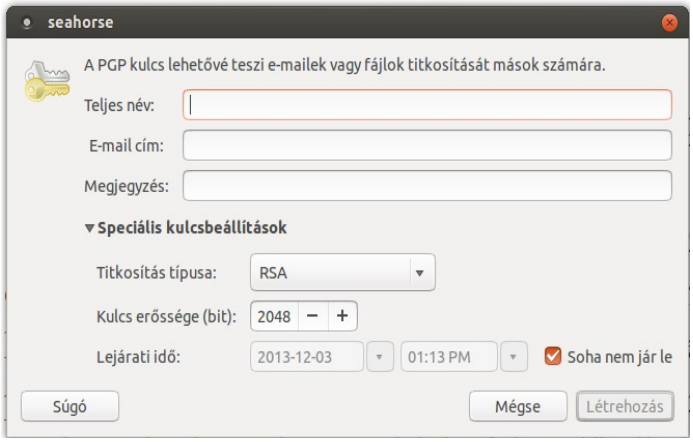
Biztonsági szempontból az egyik legnagyszerűbb szabad szoftver, természetesen GPL licenc alatt érhető el. Legfontosabb funkciója a titkosítás. Segítségével állományokat, leveleket lehet titkosítani oly módon, hogy az harmadik fél számára ne legyen visszafejthető. Nyílt kulcsos titkosítást használ, amely eljárásnál két kulcsot használunk.[5] Az egyik egy publikus (nyilvános) kulcs, amit titkosítás nélkül kell közzétennünk a másik fél számára – ezt bárki megszerezheti (pont ez a lényege). A másik kulcs a privát (titkos) kulcs, amelyet jelszóval védve biztonságban kell tárolnunk. A rendszer alapelve, hogy a publikus kulcsból nem határozható meg a privát kulcs (és fordítva sem). A másik nagy előnye ennek a titkosítási módnak, hogy már akkor tudok egy általam ismeretlen embernek titkosítottan levelet küldeni, ha még nem is találkoztunk, és akár nem is ismerem. Ennek egyetlen feltétele, hogy az interneten található kulcsszerverek egyikén, vagy a másik ember publikusan elérhető weboldalán szerepeljen a publikus kulcsa, amelyet felfűzve a saját kulcskarikámra máris tudok neki titkos levelet írni, vagy állományt kódolni úgy, hogy azt csak és kizárólag ő fogja tudni kibontani, elolvasni. Ideális akár lokális jelszótárolásra is. Alkalmas továbbá bizalmi körök kialakítására a saját aláírási rendszerével. Fontos funkciója, hogy a kulcsaink rendelkeznek úgynevezett egyedi ujjlenyomattal (*fingerprint*). Azaz, ha kulcsot akarok cserélni egy általam még ismeretlen emberrel, akkor első teendő az ujjlenyomat egyeztetése a másik féllel (egy azonosításra alkalmas csatornán, pl. telefon, videobeszélgetés, aláíró parti stb.) majd ha az ujjlenyomatok stimmelnek, akkor a kulcs felfűzése a saját karikánkra. *A Debian alapú rendszerek csomagjai a biztonsági csapat GPG-aláírásával vannak ellátva, így azok telepítésnél ellenőrzésre kerülnek. Az MD5 szignó mellett alkalmazandó remek és megbízható eszköz.*

A GPG használata

A GNUPGP telepítése Linux és Windows alapú rendszerekre is igen egyszerű. A legtöbb Linux terjesztés csomagként tartalmazza, Windows rendszerekre pedig a GPG4WIN[6] csomag letöltése javasolt. A következő lépcső a kulcspár elkészítése, amelyet vagy a gpg --gen-key parancs kiadásával,

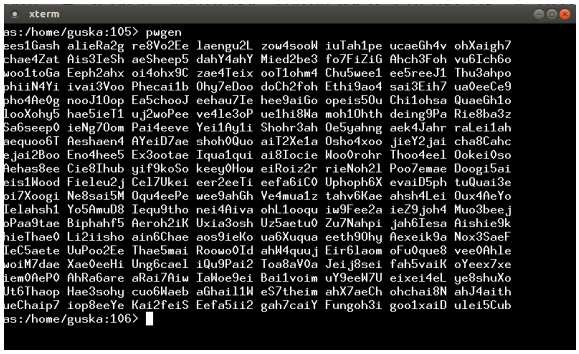
[3] http://www.gnupg.org/
[4] http://wiki.hup.hu/index.php/NSA
[5] Valójában GPG a PGP-nek, illetve az OpenPGP nyílt szabványnak is megfelelő hibrid algoritmust használ: a nyílt kulcsú (aszimmetrikus) hitelesítés mellett csak titkos kulcsú (szimmetrikus) titkosítást az adatok gyorsabb kódolásához. A PGP-vel ellentétben viszont a GPG nem alkalmaz szabadalmi vagy egyéb korlátozás alá eső algoritmusokat.
[6] http://gpg4win.org/

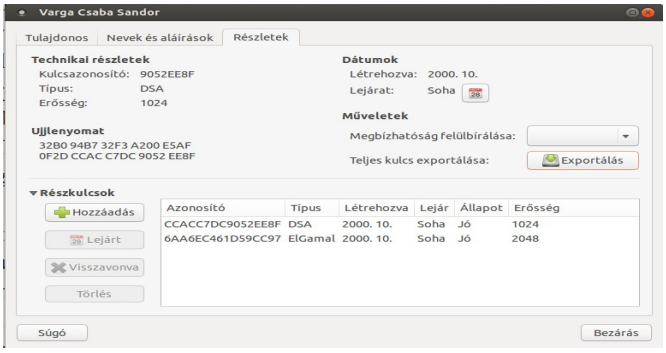




vagy a megfelelő GUI-n belüli menüpont elindításával tudunk elérni. Ubuntu rendszerek alatt a központi jelszókezelő alkalmazás (Seahorse[7], a menüben „Jelszavak és kulcsok” néven szerepel) is kezel GPG-kulcsokat, így közvetlenül onnan is el lehet készíteni.

Az alapvető adatok megadása után (név, e-mail cím) célszerű a legmagasabb bitrátájú (a képen: erősségű) kódolást kiválasztani, és többnyire célszerű a kulcs lejáratát végtelenre állítani. A lejárat átállítása „Soha nem jár le” opcióra alapvetően ellenkezik a biztonsági alapelvekkel, de praktikus okokból célszerűbb ezt később visszavonó kulcs kiadásával orvosolni, mint egy elfelejtett dátum miatt lejárt kulcsok birtokában maradni. Jelmondatnak érdemes olyan jelszót választani, ami általánosságban megfelel az elérni kívánt célnak. Azaz, mivel a GPG-vel aláírt levél felér egy hivatalosan aláírt papíralapú levéllel, ezért a jelszó javasolhatóan legalább 8-12 karakter legyen, ahol a kis-nagy betűk és az ASCII karakterek (számok, írásjelek) széles választékát érdemes felvonultatni. Ajánlott eleve a pwgen[8] alkalmazással készíteni hozzá a jelszót (Linux terjesztések legtöbbje ezt szintén csomagként tartalmazza).





Ha ezzel megvagyunk, akkor érdemes egy biztonsági másolatot készíteni vagy az egész .gnupg könyvtárról, vagy legalább a titkos és a publikus kulcsról. Természetesen a legjobb, ha a GPG kulcskarikánkat eleve titkosított (vagy Loop-AES, vagy egyéb LUKS vagy Ubuntu lvm+crypto) területen tároljuk. A továbbiakban begyűjtjük a mások által elkészített publikus kulcsokat, és mi is publikáljuk a saját kulcsunkat. A saját publikus kulcsunkat a gpg -a --export felhaszn@intezmeny.hu parancs segítségével tudjuk exportálni, amely eredményét egy sima .txt állományban elhelyezve, vagy weboldalra felrakva tudjuk publikálni. Mindezek mellett célszerű a gpg --send-keys ID parancs segítségével az aktuálisan használt kulcsszerverre is elküldeni a kulcsunkat. Jelen esetben az ID a generálásnál nekünk kiosztott egyedi kulcs azonosítója legyen. A GPG-kulcs lenyomatát a gpg --finger felhaszn@intezmeny.hu parancs segítségével nyerhetjük ki, amely információt érdemes magunknál tartanunk, pl. elhelyeztetni a saját névjegyünkön az e-mail cím mellett. Az ujjlenyomat se-

[7] http://projects.gnome.org/seahorse/
[8] http://pwgen-win.sourceforge.net/

gíteni fog olyan esetekben a saját kulcsunkat azonosítani egy idegen számára, amikor hitelesítenünk kell egymás kulcsait. A kulcsok cseréje ugyanis egymást jól és személyesen ismerő emberek esetében triviális lehet, azonban ha egy tárgyalás során a saját ujjlenyomatunkat átadtuk egy ismeretlennek, akkor ő már tudni fogja, hogy az ellenőrző szám a miénk (rajta volt a névjegyen). Így később, amikor a kulcsunkat megkapja e-mailben, akkor az ellenőrző szám egyeztetése után már biztos lehet benne, hogy a kulcs valóban ahhoz tartozik, akitől kapta. Ugyanez vonatkozik a weben elhelyezett nyilvános kulcsokra. Ha két ember – aki előtte sosem cserélt kulcsot és nem találkozott, kulcsot kíván cserélni, akkor mindenképpen szükséges akár telefonon, akár más biztosított csatornán az első lépésként megtett ujjlenyomat egyeztetése. Csak így tudhatjuk biztosan, hogy kinek a kulcsát fogadtuk be.

Alternatív e-mail címek felvétele a kulcsunkhoz

Amennyiben több címünk van, abban az esetben érdemes mindegyiket e-mail álnévként felvenni a fő címünkhöz, amelyhez a kulcsot készítettük. Ennek akkor van jelentősége, ha a másik oldal egy olyan címről kap tőlünk üzenetet, amely nem volt felvéve, akkor manuálisan kell majd a kliense számára megmutatni, hogy azt az alap kulcsunkkal oldja fel, amíg ha az összes címünk felvettük, ezzel már továbbiakban probléma nem lesz. Az alternatív e-mail címek felvételének triviális módja, ha a Seahorse alkalmazásban hozzáadjuk, azonban a gpg –edit-key ID (majd adduid) parancs segítségével is megtehetjük.

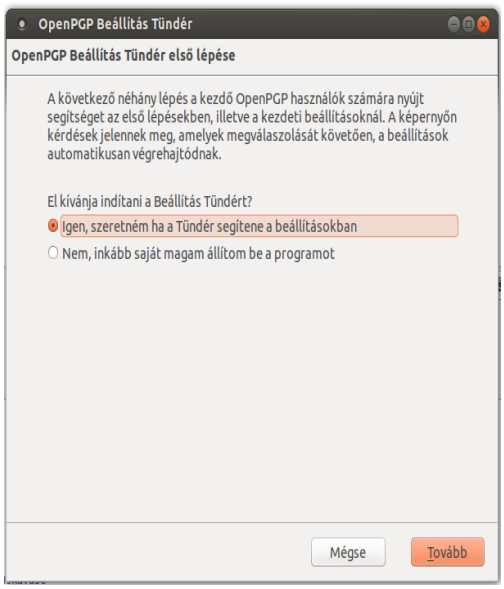
Partner kulcsának felvétele a karikára

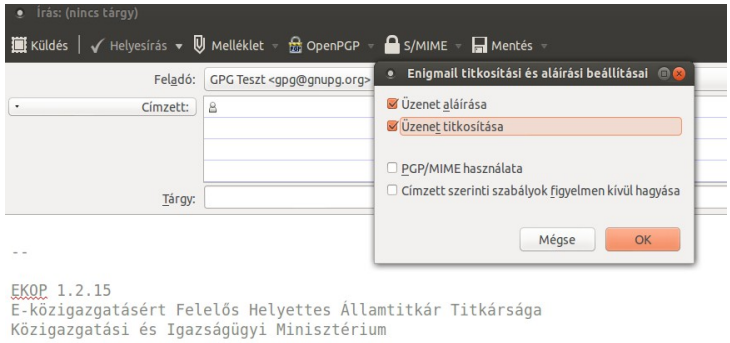
A partnereink GPG nyilvános kulcsának importálása egy szükséges feltétele a titkos levelezésnek. A folyamat rendkívül egyszerű, az Enigmail kiterjesztés használói a menüből tudnak felvenni e-mail cím alapján új partnert a kulcsszerverek segítségével, vagy pedig kézzel lehet hozzáadni. A parancssorból pedig a gpg –import kulcs.asc parancs segítségével tudjuk beilleszteni a kulcsot. Előtte a gpg -i kulcs.asc parancs segítségével tudjuk ellenőrizni, hogy kinek a kulcsát fogjuk illeszteni. A GPG felépítéséből következik, hogy már akkor is tudunk valakinek titkos levelet küldeni, ha a fogadó fél a mi címünk és publikus kulcsunk nem ismeri, hiszen mi az ő publikus kulcsával fogjuk elkódolni a levelet, amit ő a saját titkos kulcsával fog feloldani. Egyetlen feltétel, hogy valamelyik nyilvános kulcsszerverről, vagy a honlapjáról előtte le kell szednünk az ő aktuális kulcsát. Ilyen kulcsszerver pl. a http://wwwkeys.pgp.net vagy a http://pgp.mit.edu/ is. Éppen ezért a saját kulcsunkat érdemes minden változás után (pl. plusz e-mail cím hozzáadása) frissítenünk a szerveren is, és a saját weblapunkra is érdemes felrakni.

Thunderbird integráció

Mint azt láthattuk, a gpg segédprogram az oda-vissza titkosítás részét végzi a folyamatnak. Ahhoz, hogy kényelmesen tudjuk használni a GPG-t, a levelezőkliensünk számára is elérhetővé kell tenni. A legtöbb szabad szoftveres levelezőkliens vagy natív módon vagy kiegészítő (plugin) segítségével tehető alkalmassá a titkosított levelezésre. A Thunderbird – mint az egyik leginkább elterjedt levelezőkliens – esetében ez a kiegészítő az Enigmail.[9] A kiegészítőt a Thunderbird saját belső hitelesítéssel ellátott rendszerén belül az Eszközök/Kiegészítők menü alól érdemes feltelepíteni. A levelező újraindítás után egy OpenPGP menüvel gyarapodik, ahol a Beállítások között megtalálhatjuk a Beállítás tündért, amely végigvezet minket az alapvető beállításokon, és itt tudjuk kijelölni, hogy melyik kulcsot akarjuk használni a levelezéshez. Ezután minden olyan levélnél, amely aláírva vagy titkosítva érkezik hozzánk, a jelszókérő ablak jelenik meg, és az Enigmail a gpg parancs segítségével ki fogja kódolni nekünk az üzenetet. Illetve ha írni akarunk titkosított levelet, akkor a levél-

[9] http://www.enigmail.net





írás menüben a lakatra kattintva dönthetünk, hogy mit is szeretnénk: aláírni vagy titkosítani (vagy mind a kettőt).

A PGP/MIME használata

Mivel többnyire nem tudhatjuk, hogy a fogadó oldalon, ahol a levelünk fogják dekódolni, milyen kliens található ezért célszerű a PGP/MIME[10] opciót alkalmazni. Ezt megtehetjük akár levelenként is, amikor egy levélről eldöntjük, hogy az titkosított lesz, akkor a PGP/MIME opciót szintén bejelöljük, vagy egyszerűbb ezt globálisan beállítani a Thunderbird Postafiók beállításai menü, OpenPGP Adatbiztonság almenü, Mindig használjon PGP/MIME-ot kiválasztásával.

A bizalmi körök

A GPG-t, csakúgy mint elődjét, úgy tervezték meg, hogy ne csak titkosítani lehessen vele állományokat, e-maileket vagy a segítségével pl. lemezeket (loop-aes+gpg), hanem ki lehet vele építeni bizalmi hálózatot is. Ennek alapjait általánosságban Whitfield Diffie, Martin Hellman és Ralph Merkle nevéhez kötik, akik a módszert az 1970-es években fejlesztették ki (a módszer lényege, hogy ismeretlen emberek is építhetnek bizalmi hálózatot az interneten). A GPG, mint azt már taglaltuk, a 90-es évek közepén terjedt el igazán, és már az első implementációi is használták a kulcsok hitelesíthetőségét, azaz a bizalmi hálózatok lehetőségét. Ez egy igen előremutató gondolkodásmód, és egy olyan technológia alap, amely a mai napig megfelelően teszi a dolgát. Gondoljunk csak bele, a ’90-es évek itthoni világában még nagyrészt BBS[11]-en leveleztek az emberek, és a technológia iránt érdeklődő hasonló gondolkodású emberek leginkább csak demó partikon vagy a nyomtatott sajtón keresztül (pl. Alaplap Magazin stb.) érintkezhettek egymással. Nem voltak fejlett közösségi hálózatok, ahol ma percek alatt nagyjából le lehet ellenőrizni valakit, még ha csak felületesen is. A cél tehát nem a közösségi hálózatok egy korai modellje volt, hanem pl. fejlesztők között kialakítani egy bizalmi kapcsolatot anélkül, hogy azok valaha is találkoztak volna egymással. Hiszen pont a fejlesztés volt az már a '90-es években is, amely jellemzően egyetemi berkeken belül zajlott interneten keresztül. Ma is teljesen jellemző, hogy pl. a szabad szoftveres fejlesztéseken számtalan országból dolgoznak együtt a fejlesztők. Pont egy szoftver fejlesztésénél létfontosságú (gondoljunk csak a Linux rendszermag, vagy egy SSH fejlesztésre), hogy a fejlesztők mindig tudják, hogy a fejlesztő

[10] http://hu.wikipedia.org/wiki/MIME
[11] http://hu.wikipedia.org/wiki/FidoNet

társuk kódját elfogadhatják-e, mivel tudják, hogy valaki szavatolt érte, vagy esetleg a GPG segítségével ők maguk győződhetnek meg róla, hogy ki is az adott fejlesztő. Nézzük, hogyan működik ez a gyakorlatban. Parancssorból a gpg --edit-key felhasználó parancs segítségével tudjuk szerkeszteni valakinek a kulcsát. Pontosan úgy, mintha a sajátunkat akarnánk szerkeszteni. Majd a trust parancs kiadásával határozhatjuk meg, hogy mennyire is ismerjük mi az adott felhasználót. Az itt kapott lehetőségek választásával dönthetünk:

```
1 = I don't know or won't say
2 = I do NOT trust
3 = I trust marginally
4 = I trust fully
5 = I trust ultimately
m = back to the main menu
```

Miután eldöntöttük, kiben mennyire bízunk, a kulcsszerkesztésből kilépve a gpg –sign-key felhasználó parancs segítségével elkészítjük az aláírást, majd már csak egy dolgunk maradt, feltölteni az illető kulcsát a publikus szerverre, pl. a gpg –send-key KEY ID segítségével. Ha utána megnézzük a kulcsszerveren, akkor látni fogjuk, hogy a szerver a többi aláírót és a mi aláírásunkat egybegyűjtve a kulcsot megjeleníti. Így akár két teljesen idegen ember is megbízhat egymás kulcsában, ha azt látja, hogy a másik fél kulcsát több olyan ember is aláírta, akikben ő megbízik. Hogy érthetőbb legyen: Péter és Zoltán ismeri egymást a kulcsszerveren, és bíznak is egymásban. Én Péter kulcsában szeretnék bízni, de nem ismerem, viszont Zoltán személyesen is, és benne bízom. Zoltán aláírta Péter kulcsát, így én el fogom hinni, hogy az tényleg annak a Péternek a kulcsa, akit én keresek, mivel Zoltán hitelesítette. Nyilvánvalóan minél több ilyen hitelesítés történik egy adott kulcsra, annál inkább nő annak az esélye, hogy egy idegen ellenőrizni tudja a hitelességét, egy közbeiktatott személy segítségével. Itthon is rendszeresen rendeznek kulcsaláíró alkalmakat, ahol ismeretségi, munkatársi, egyesületi stb. alapon írunk kulcsokat egymásnak alá, papíron és gép segítéségével is, és persze a személyi igazolvány bemutatása után.

GPG elérése böngészőből

Sokakban már maga a cím is erős ellenérzést kelhet, hiszen igazán a GPG program az, ami szavatolja a kulcsok hitelességét. Már a ráépített GUI-k is kétkedést válthatnak ki az igazi biztonságmániásokból. Azonban sok esetben hasznos lehet, ha egy munkahelyi környezetben, ahol csak egy webes levelezőt használunk, ott is elérhetjük a GPG-t. Akár esetleg egy másik kulcs (nem az általánosan használt) segítségével. Gyakori igényt tehát, hogy egy olyan területen, ahol a gépet nem uraljuk 100%-os tekintetben (munkahelyi környezet, ahol mások üzemeltetik a lokális gépeket) is szeretnénk GPG-t használni. Mint már korábban említettük, a GPG esetében az egyik legfontosabb dolog a titkos kulcs integritásának megőrzése (hozzáférhetetlenség biztosítása). Így a példa kedvéért a munkahelyünkön (MS környezetben) magát a GPG4WIN-t feltelepítjük, de a kulcsokat a Truecryptben vagy külön USB-kulcson tároljuk. A céges levelezőbe nem tudjuk vagy akarjuk feltelepíteni az Enigmail kiterjesztést, viszont a saját webes levelezőnkben el akarjuk olvasni a GPG-vel érkező leveleket. Erre egy remek megoldás a Firefox kiterjesztéseként elérhető WebPG (http://webpg.org/) program, amely segítségével akár egy Gmailben lévő levelet is vissza tudunk fejteni, vagy írni tudunk egyet. Természetesen a GPG felhasználása alapvetően nem így javasolt, de kényszerhelyzet esetén egy ilyen lehetőség adott.

Hasznos kulcs parancsok és ötletek

Beállítások

A .gnupg/options állományban találhatóak meg a GPG alapvető beállításai. Ezek közül érdemes párat fixen rögzíteni, mint például:

– encrypt-to PUBKEYID: a későbbi visszaolvashatóság érdekében ez az egyik legfontosabb opció. A GPG ugyanis alapesetben egy levél elküldésekor csak annak a félnek fogja titkosítani a levelet, aki a TO: illetve több címzett esetében akik a CC: és BCC: mezőkben szerepelnek. Ha az encrypt-to PUBKEYID szerepel (ahol a saját publikus kulcsunk ID-je kell hogy legyen a to után, pl. így: encrypt-to 9052EE8F), akkor minden elküldött levél a saját kulcsunkkal is titkosítva lesz. Így később hónapok, évek múlva a sent-items vagy sent-mail mappánk titkosítottlevél-tartalmát is el fogjuk tudni olvasni. Ami egyébként másként nem lenne lehetséges.

– keyring pubring.gpg: itt tudjuk rögzíteni, hogy hol tároljuk a publikus kulcsokat. Ennek a mentésnél lehet szerepe.

– keyserver wwwkeys.pgp.net: megmondhatjuk egy vagy több kulcsszerver nevét, hogy ezektől kérdezze meg a kulcsokat, illetve oda aktualizáljon vissza.

– secret-keyring ~/.gnupg/secring.gpg: itt tudjuk rögzíteni, hogy hol tároljuk a titkos kulcsokat. Ennek a mentésnél lehet szerepe.

– default-key 32B094B732F3A200E5AF0F2DCCACC7DC9052EE8F: megmondjuk, hogy több kulcs esetén melyik az alap kulcsunk. A paraméter, mint az látható, a teljes ujjlenyomat.

Állományok titkosítása

A gpg -e fájlnév.txt parancs segítségével tudunk állományt titkosítani, majd a gpg -d fájlnév.txt parancs a kioldására szolgál.

VIM integráció

A VI felhasználóknak igazi hiánypótló alkalmazás, hogy a GPG-t és a VIM-et össze lehet párosítani. Így a VIM natív módon meg tudja nyitni egy jelszó bekérése után a GPG-vel titkosított szöveges állományokat, majd a szerkesztés után menteni is tudja. A bővítmény neve gnupg.vim[12] és a ~.vim/plugin/ könyvtárba másolása után azonnal működőképes. Ez hatalmas segítség lehet, ha a jelszavakat egy sima szöveges állományba mentjük, majd titkosítjuk. Később ha olvasni vagy írni akarjuk, akkor nem kell állandóan kézzel ki-be titkosítani, hanem elég csak a VI használatával megnyitni.

Gpg-agent[13]

A jelenleg használt Ubuntu asztali telepítésekben alapértelmezetten be van kapcsolva, és használhatjuk is. Számtalan opciója van, többek között nem csak a GPG jelszavunkat tudja megjegyezni, de rá lehet bízni az SSH jelszavak és kulcsok kezelését is. Használata praktikus lehet, ha sokszor kell kézzel megadnunk egy adott idő intervallumon belül a gpg kódmondatot. Azonban érvényességi körét és idejét érdemes erősen korlátozni, azaz csak az adott munkamenet tartalmára jegyezze meg a jelszót, illetve használata esetén (egyébként is) szokjunk hozzá, hogy az automatikus képernyőzár be van kapcsolva, valamint fontos, hogy amint elhagyjuk a gépünket, akár csak 2 percre is, azonnal zároljuk a munkaállomást. Ubuntuban a Ctrl+Alt+L kombinációval ezt könnyen megtehetjük.

[12] http://www.vim.org/scripts/script.php?script_id=661
[13] http://manpages.ubuntu.com/manpages/hardy/man1/gpg-agent.1.html

Seahorse-nautilus[14]

Egy igazán remek kiegészítés az amúgy is gyakran használt Nautilus fájlkezelőhöz. Segítségével a Nautilus menüjében megjelenik egy Encrypt menü, amely segítségével parancssor mellőzésével tudunk titkosítani vagy kibontani állományokat. Igazi jelentősége akkor mutatkozik meg, amikor 8-10 különböző ember számára kell titkosítani egy állományt és felrakni pl. egy megosztásra. Ilyenkor parancssor esetén a gpg -r vagy --recipient megadásával egyenként felsorolhatjuk mind a 10 érintettet. A kiegészítés esetében viszont egy listából tudunk hozzáadni ID-ket, amely már 2-3 ember esetében is nagyságrendekkel egyszerűbb, mint kézzel felsorolni. Nem beszélve arról, hogy ha éppen csak a nevének az első vagy második tagját tudjuk, az e-mail címét, vagy a kulcs ID-jét nem. A Nautilusban akár egy egész könyvtár tartalmát szervezhetjük egy kattintással egy db tömörített, majd ismét egy kattintással egy db titkosított állományba, majd egy újabb kattintás segítségével pedig elküldhetjük FTP, vagy egyéb megosztás irányába. Használata tehát bizonyos körülmények között (*user experience*) kifejezetten ajánlott.

Kulcs lekérése szerverről, illetve visszatöltése

– A gpg --search-keys segítségével tudunk szerverről kulcsot keresni
– A gpg --recv-keys key ID segítségével tudjuk a már ismert kulcsot letölteni a szerverről
– A gpg –refresh-keys a parancs futtatásával tudjuk szinkronba hozni a kulcsszerveren és a karikánkon lévő kulcsokat.

Kulcs visszavonása

A gpg –gen-revoke ID kiadásával tudunk olyan tanúsítványt készíteni, amelynek segítségével vissza tudjuk vonni a kulcsunkat. Ez egy nagyon fontos lépés, ugyanis ha a titkos kulcsunk vagy jelszavunk kikerült a kizárólagos felügyeletünk alól, akkor nagyon fontos, hogy azt a kulcsot nyilvánosan is visszavonjuk.

```
Create a revocation certificate for this key? (y/N) y
Kérem, válassza ki a visszavonás okát:
  0 = Nincs megadva ok.
  1 = A kulcs kompromittálódott.
  2 = A kulcsot lecserélték.
  3 = A kulcs már nem használatos.
  Q = Mégsem
(Valószínűleg a(z) 1. lehetőséget akarja választani.)
Mit választ?
```

Az elkészült tanúsítványt publikálni, azaz terjeszteni kell, pl. a kulcsszerverek felé is. Ha a kulcsunk lejárati idő miatt kerül visszavonásra, akkor erre a lépésre nincs feltétlen szükség. **A visszavonó kulcsot csak indokolt estben készítsük el.** Illetőleg van egy olyan nézet, mely szerint ha pl. a jelszót elfelejtjük, akkor a legjobb ha az első kulcs generálásánál már elkészítjük a visszavonó kulcsot is, mivel ilyenkor csak azzal tudunk kulcsot érvényteleníteni, hiszen a jelszót már nem tudjuk. **Ebben az esetben nagyon fontos a visszavonó kulcs extra biztonságos őrzése. Azaz a megfelelő helyen tárolása, pl. nyomtatásban egy széfben.**

[14] http://git.gnome.org/browse/seahorse-nautilus/

Egy valós biztonsági rés a GPG-ben

Hosszú idő óta keresik a GPG programban és az algoritmusban rejlő rést és az illetéktelen visszafejtés lehetőségét. Egy megfelelően erősre állított kulcs visszafejtése a tudomány jelenlegi állása szerint nem lehetséges.[15] Azonban a hosszú évek alatt nyilvánosságot látott egy valós probléma. Amikor titkosított levelet küldünk, akkor a GPG aláírni és titkosítani a levél törzsét és a csatolmányt fogja csak. A feladó, a címzett, a tárgy érintetlen marad, hiszen a megfelelő RFC kimondja ezt. Ha tehát egy illetéktelen hozzáfér a levelezésünkhöz, esetleg még mielőtt mi látnánk azt (pl. rendszergazdai jogokkal fájl szinten) akkor át tudja írni ezeket az értékeket. Mást nem tud kezdeni vele, mivel ha a titkosított részhez hozzányúl, akkor megsérül a kód integritása és számunkra is olvashatatlan lesz a titkos tartalom. Védekezni ez ellen úgy tudunk, hogy minden titkosított levél törzsébe (BODY) belemásoljuk a Feladó, a Címzett és a Tárgy sorokat is. Így címzett ellenőrizni tudja majd ezek hitelességét is, hiszen a titkosított tartalmon belül tároltuk.

Készítette a Közigazgatási és Igazságügyi minisztérium
E-közigazgatási Szabad Szoftver Kompetencia Központja, 2013. Budapest
Varga Csaba Sándor
1.1 verzió

[15] A kvantum-számítógépes Shor-algoritmus (http://en.wikipedia.org/wiki/Shor%27s_algorithm) elméletileg igen, a mai kis kapacitású kvantumszámítógépek gyakorlatilag nem képesek a nagy számok prímtényezőkre való hatékony felbontására, amely lehetővé tenné a prímszámfaktorizáció nagy algoritmikus bonyolultságán alapuló nyílt kulcsú titkosítások feltörését (a jelenleg ismert kvantum-számítógépes rekord mindössze a 21 felbontása prímtényezőkre). 2011-ben 128 kvantumbites, 2013-ban pedig már 512 kvantumbites kvantumszámítógép is gazdára talált (http://en.wikipedia.org/wiki/D-Wave_Systems), de a nyílt kulcsú titkosításban ma már kicsinek számító 300 jegyű szám felbontásához az ideális 10 ezer helyett akár 10 millió kvantumbites gépre is szükség lehet a kvantum-dekoherencia okozta hibák javítása miatt (http://en.wikipedia.org/wiki/Quantum_computer#Quantum decoherence). Mivel a kvantumszámítógépek fenyegetést jelenthetnek a jövőben a jelenlegi nyílt kulcsú titkosításokra, megkezdődött az ellen is védő, már jelenleg is létező titkosítási algoritmusok vizsgálata, javítása, l. http://en.wikipedia.org/wiki/Post-quantum cryptography.