

Podstawy pracy z pakietem gpg4win (wersja 2.1.0)

Idea pary kluczy publicznego i prywatnego

Zasada działania szyfrowania asymetrycznego opiera się na idei dwóch kluczy: publicznego i prywatnego. Dane zaszyfrowane jednym z kluczy można odszyfrować wyłącznie drugim kluczem z pary. (np. zaszyfrować kluczem publicznym można, a potem odszyfrować kluczem prywatnym). Jak wskazują nazwy klucz prywatny powinien być dostępny tylko właścicielowi, natomiast klucz publiczny może być dostępny dla każdego. Siła szyfrowania asymetrycznego tkwi w złożoności danych działań matematycznych względem działań do nich odwrotnych. Pomnożenie dwóch dużych liczb pierwszych jest dużo mniej wymagające obliczeniowo, niż uzyskanie czynników z tak uzyskanego iloczynu. Weźmy dwie 128-bitowe liczby pierwsze, przeciętny komputer w ciągu niespełna sekundy dokona mnożenia takich liczb i zwróci liczbę 256-bitową. Odwrócenie działania, czyli uzyskanie z liczby 256-bitowej dwóch liczb 128-bitowych, będzie dużo bardziej wymagające. Klucz publiczny może być właśnie tak powstałą liczbą 256-bitową, natomiast klucz prywatny dwiema liczbami 128-bitowymi, z których powstał klucz publiczny.

Przykładowo rozłożenie na czynniki pierwsze liczby 15 jest proste: 3 5, natomiast liczba 91723948197324 sprawi już trochę więcej kłopotu, tym bardziej liczba 155- czy 200-cyfrowa. Rozłożenie liczby na czynniki pierwsze składa się z pewnej ilości kroków, która to ilość zwiększa się wykładniczo wraz ze wzrostem wielkości liczby. Oznacza to, że jeśli liczba jest wystarczająco duża, jej faktoryzacja może zająć lata.

Najbardziej istotne jest to, że klucz deszyfrujący (prywatny) znacząco różni się od klucza szyfrującego (publicznego) oraz że dla odpowiednio długiego klucza publicznego nie jest możliwe, w rozsądnym czasie, wygenerowanie klucza prywatnego.

Standard OpenPGP i jego implementacje

Wraz z rozwojem Internetu oraz związanych z nim niebezpieczeństw i brakiem prywatności, szyfrowanie przesyłanych danych nabrało szczególnego znaczenia.

W 1991 roku Phil Zimmerman zapoczątkował projekt PGP - program umożliwiający szyfrowanie i cyfrowe podpisywanie wszelkiego rodzaju dokumentów elektronicznych. PGP umożliwiał bezpieczną wymianę oraz autentykację wszelkiego rodzaju dokumentów elektronicznych. Początkowo był dostępny wraz kodem źródłowym, począwszy jednak od wersji 5.0 stał się oprogramowaniem zamkniętym i komercyjnym.

Z początkiem 1997 roku została powołana w ramach IETF (Internet Engineering Task Force) grupa, która zdefiniowała otwarty standard szyfrowania poczty elektronicznej z wykorzystaniem kryptografii klucza publicznego, bazując na zamkniętym programie PGP i zapewniając kompatybilność z nim. Tak powstał standard OpenPGP, opisany w dokumencie RFC 2440. Określa on wzorcowe formaty zaszyfrowanych wiadomości i podpisów. Od tej pory praktycznie każdy może stworzyć własną implementację standardu i wymieniać informacje z użytkownikami innych, jak choćby PGP.

Najpopularniejszą otwartą implementacją protokołu OpenPGP jest Gnu Privacy Guard (GPG). Choć oferuje nieco mniejsze możliwości niż komercyjny PGP, znakomicie nadaje się do zastosowań w małych firmach oraz wśród użytkowników domowych.

Instalacja GnuPG i narzędzi wspomagających

Gnu Privacy Guard w wersji w przygotowanej na platformę Windows można pobrać ze strony projektu <http://www.gpg4win.org/>. W pakiecie oprócz GnuPG znajduje się kilka aplikacji ułatwiających korzystanie z niego:

- rozszerzenia systemowego menu podręcznego

- GNU Privacy Assistant (GPA) – narzędzie do zarządzania kluczami
- Kleopatra – narzędzie do szyfrowania i podpisywania oraz zarządzania kluczami PGP i certyfikatami X.509
- Claws-Mail – klient poczty, z wtyczkami PGP

Po instalacji możemy korzystać z gpg w powłoce systemowej (*Start -> Uruchom -> cmd*).

Po wpisaniu:

gpg --version

powinniśmy ujrzeć numer wersji zainstalowanego GPG oraz wspierane algorytmy kryptograficzne.

```

C:\>gpg --version
gpg (GnuPG) 2.0.17 (Gpg4win 2.1.0)
libgcrypt 1.4.6
Copyright (C) 2011 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Home: C:/Documents and Settings/student/Dane aplikacji/gnupg
Obsługiwane algorytmy:
Asymetryczne: RSA, ELG, DSA
Symetryczne: 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH, CAMELLIA128,
              CAMELLIA192, CAMELLIA256
Skrótów: MD5, SHA1, RIPEMD160, SHA256, SHA384, SHA512, SHA224
Kompresji: Nieskompresowany, ZIP, ZLIB, BZIP2
C:\>_
  
```

Rys.1 Wynik wywołania polecenia gpg --version

Generowanie kluczy

Kiedy utworzymy parę kluczy, zarówno publiczny jak i prywatny będą przechowywane na dysku naszego komputera. Stwarza to pewne ryzyko – każdy kto uzyska dostęp do naszego komputera może zdobyć nasz klucz prywatny i odszyfrować nasze wiadomości lub podszyć się pod nas. Próba zapamiętania klucza w pamięci również skazana jest na niepowodzenie – jest on po prostu bardzo długi. Praktycznym rozwiązaniem tego problemu jest zaszyfrowanie klucza prywatnego hasłem. Przy każdej próbie użycia klucza prywatnego gpg zapyta o nie, a po poprawnym podaniu odszyfruje klucz w pamięci i skorzysta z niego. Należy dbać by hasło miało odpowiednią długość i złożoność – na pewno złym pomysłem będzie używanie haseł słownikowych. Zaleca się hasła zawierające małe i duże litery, cyfry oraz znaki specjalne o długości conajmniej ośmiu znaków.

Aby utworzyć parę kluczy, w powłoce systemowej wpisujemy:

gpg --gen-key

Zostaniemy poproszeni o wybranie rodzaju klucza. Wybierzmy RSA.

Następne pytanie dotyczy długości klucza. Wartość ta powinna być kompromisem – krótszy klucz łatwiej złamać, dłuższy klucz będzie wymagał więcej czasu na każdą operację, która go wykorzystuje.

Wzrost mocy obliczeniowej komputerów oraz rozwój badań w dziedzinie kryptoanalizy powoduje, że wiele algorytmów, uważanych niegdyś za niemożliwe do złamania w krótkim czasie, zostało skompromitowanych w ostatnich latach. Na przykład w ostatnim czasie złamano algorytm RSA o długości klucza 576 bitów (100 maszyn w 3 miesiące).

Kolejne pytanie dotyczy długości czasu ważności klucza. Ustawienie wygaśnięcia klucza po pewnym czasie zwiększa nieco bezpieczeństwo – po przekroczeniu daty ważności nie będzie można z niego korzystać. Wiąże się z tym pewna niedogodność – będziemy musieli rozesłać nasz nowy klucz publiczny wszystkim, którzy dotychczas korzystali ze starego.

[illegible]

Rys. 2 Generowanie pary kluczy.

Teraz program zapyta o tożsamość i adres email. Są to informacje, które pozwolą innym zidentyfikować do kogo należy klucz. Jeśli troszczymy się o naszą prywatność, nie musimy podawać naszych prawdziwych danych, zamiast tego możemy wybrać nazwę unikalną na tyle, by nasz klucz nie został pomyłony z należącym do kogoś innego. Możemy również uzupełnić pole

komentarz lub pozostawić je puste.

Ostatnia rzecz o jaką zapyta proces generowania klucza jest hasło, którym będziemy go chronić. Podczas podawania hasła nie drukuje się ono na ekranie (ani żadne inne znaki sygnalizujące wpisywanie).

Po podaniu danych rozpoczyna się właściwy proces generowania klucza. Aby był on wygenerowany w sposób losowy potrzebna jest duża ilość entropii. Podczas trwania tego procesu zaleca się by intensywnie wykorzystywać dysk twardy lub aktywnie przeglądać strony internetowe. Aby dostarczyć odpowiednią entropię można włączyć kopiowanie dużego pliku, z różnymi odstępami czasowymi naciskać klawisze i losowo przesuwając kursor myszy.

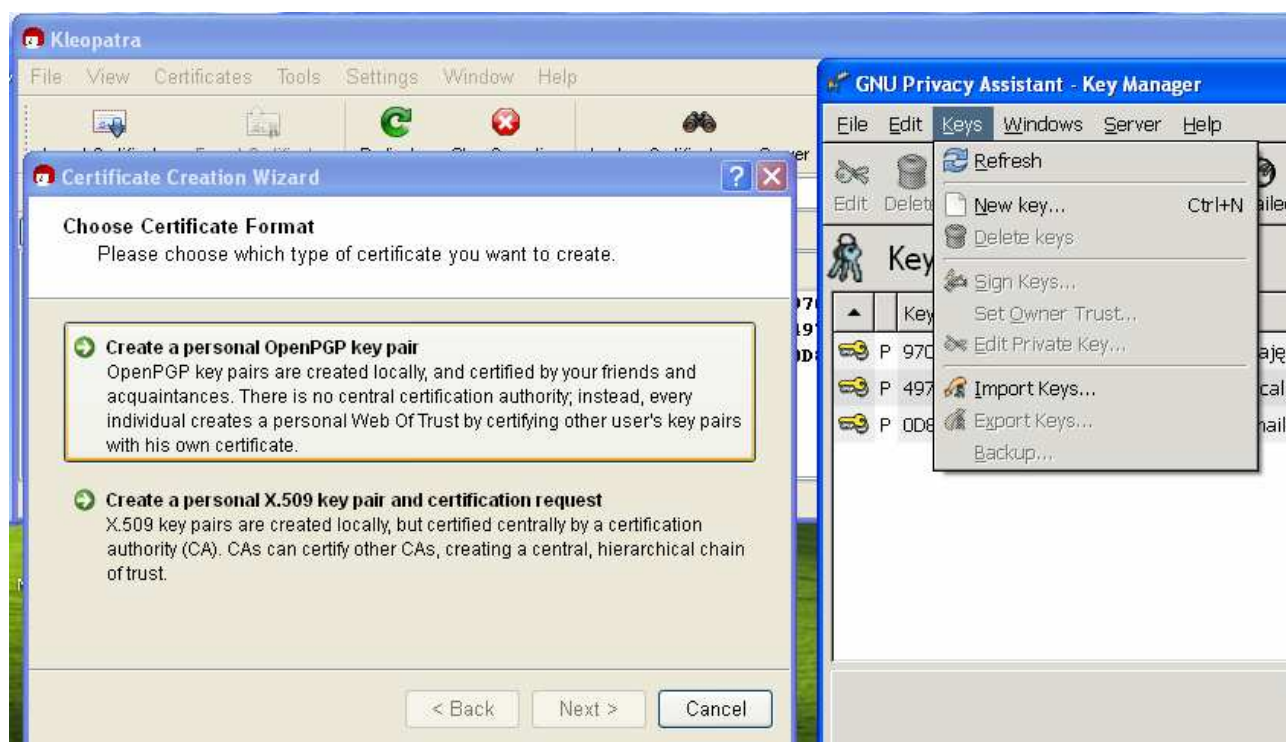
Po wygenerowaniu nasza para kluczy znajduje się w katalogu:

`C:\Documents and Settings\%USERNAME%\Dane aplikacji\GnuPG`

Klucz publiczny to *pubring.gpg* a prywatny *secring.gpg*. Ten drugi należy szczególnie chronić.

Wygenerowane klucze należy po zakończonych zajęciach bezwzględnie usunąć z komputerów laboratoryjnych!

Alternatywnie, tą samą operację generowania klucza możemy wykonać korzystając z graficznych narzędzi do zarządzania kluczami, instalowanymi wraz z pakietem *Gpg4win*, to jest *GPA* lub *Kleopatra*.



Rys. 3 Generowanie kluczy w narzędziach graficznych.

Eksportowanie oraz importowanie kluczy

Wygenerowany klucz publiczny jest bezużyteczny jeśli nie dostarczymy go osobom, z którymi chcemy wymieniać szyfrowane informacje. Możemy więc umieścić go na naszej stronie internetowej lub wysłać pocztą elektroniczną. Jeszcze innym, a prawdopodobnie

najpraktyczniejszym, sposobem jest umieszczenie klucza na specjalnym serwerze, który służy do ich przechowywania.

Eksport klucza do pliku

By wyeksportować klucz do formatu tekstowego użyjemy w konsoli systemowej polecenia:

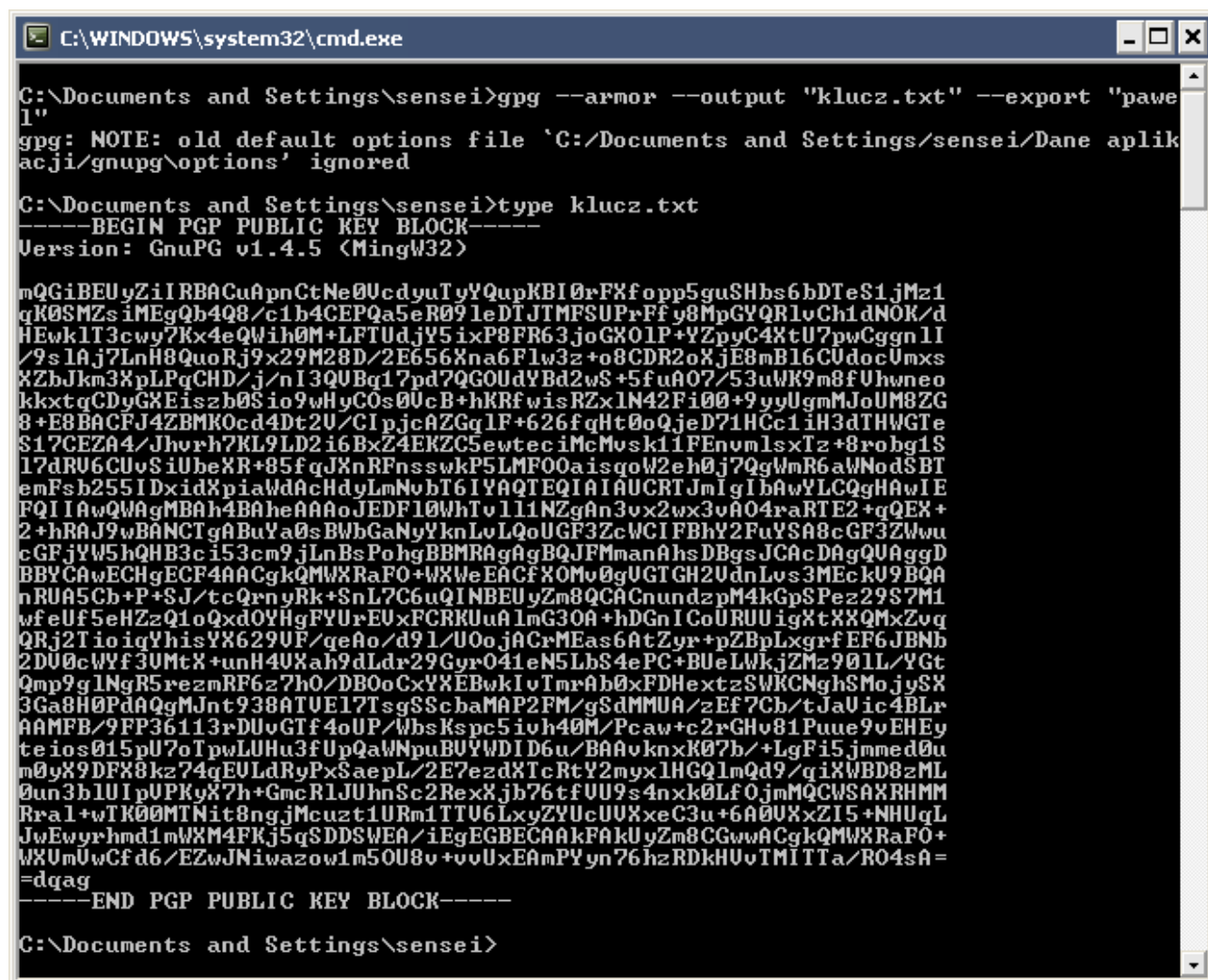
```
gpg --armor --output "klucz.txt" --export "NAZWA"
```

gdzie w miejsce nazwa wpisujemy adres email wpisany przy generacji klucza lub naszą 'nazwę' która identyfikuje klucz (np. imię i nazwisko). Możemy także podać część nazwy jeśli jednoznacznie wskazuje ona klucz (np. jedynie nazwa domeny adresu email jeśli jest unikatowa w zbiorze kluczy).

Tak przygotowany klucz jest gotowy do publikowania. Ważne jest by umieszczać go w całości (razem z komentarzami, od początkowego do końcowego):

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
```

```
-----END PGP PUBLIC KEY BLOCK-----
```



```
C:\WINDOWS\system32\cmd.exe

C:\Documents and Settings\sensei>gpg --armor --output "klucz.txt" --export "pawe
l"
gpg: NOTE: old default options file 'C:/Documents and Settings/sensei/Dane aplik
acji/gnupg/options' ignored

C:\Documents and Settings\sensei>type klucz.txt
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.5 (MingW32)

mQGIBEUyZiIRBAcuApnCtNe0UcdyuTyYQupKBI0rFXfopp5guSHbs6bDTes1jMz1
qK0SMZsIMEgQb4Q8/c1b4CEPQa5eR091eDTJTMFSUPrFfy8MpGYQR1vCh1dNOK/d
HEwklT3cwy7Kx4eQWih0M+LFTUdjY5ixP8FR63joGX01P+YZpyC4XtU7pwCggn1I
/9slAj7LnH8QuorJ9x29M28D/2E656Xna6Flw3z+o8CDR2oXje8mB16CUdocUmxs
XZbJkm3XpLPqCHD/j/n13QUBg17pd7QGOUdYBd2wS+5fuA07/53uWK9m8fUhwne0
kkxtqCdYGXeiszb0Sio9wHyC0s0UcB+hKrfwisRZx1N42Fi00+9yyUgmMJoUM8ZG
8+E8BACFJ4ZBMK0cd4Dt2U/C1pjcaZGq1F+626fqHt0oQjeD71HCc1iH3dTHWGT
S17CEZA4/Jhurh7KL9LD2i6BxZ4EKZC5ewtecIMcMusk11FEnumlsxTz+8robg1S
17dRU6CUvSiUbeXR+85fqJXnRFnsswkP5LMP00aisqoW2eh0j7QgWmR6aWNoDSBT
emFsb255IDxidXpiaWdAcHdyLmNubT6lYAQTEQIAIAUCRTJmIgiBawYLCQgHAuIE
FQIIAwQWAgMBAh4BAheAAaoJEDF10WhTvl11NZgAn3vx2wx3va04raRTE2+qQEX+
2+hRAJ9wBANCTgABuYa0sBwbGaNyYknLvLQoUGF3ZcWCI FBhY2FuYSA8cGF3ZWuu
cGFjYW5hQHB3ci53cm9jLnBsPohgBBMRAgAgBQJFMmanAhsDBgsJCAcDAgQUAggD
BBYCAwECHgECF4AAcGkQMWRaFO+WXWeEACfXOMv0gUGTGH2UdnLvs3MEckU9BQA
nRUA5Cb+P+SJ/tcQrnyRk+Snl7C6uQINBEUyZm8QCAcGundzPM4kGpSPez29S7M1
wfeUf5eHZZq1oQxdOYHgFYUrEUxPCRKUuAlmG30A+hDgnICoURUuigXtXXQMxZvq
QRj2TioigYhisYX629UF/geAo/d9L/U0oJAcrMEas6AtZyr+pZBpLxgrfEF6JBnb
2DU0cWYf3UMtX+unH4UXah9dLdr29Gyr041eN5LbS4ePC+BUeLWkjZmz901L/YGt
Qmp9glNgR5rezmRF6z7h0/DB0oCxYXEBuklvTmrAb0xFDHextzSWKCNghSMojySx
3Ga8H0PdAQgMJnt938ATUE17TsgSScbAMAP2FM/gSdMMUA/zEf7Cb/tJaUic4BLr
AAMFB/9FP36113rDUvGTF4oUP/WbsKspc5ivh40M/Pcaw+c2rGHv81Puue9vEHEy
teios015pU7oTpWLUHu3fUpQaWNPuBUYYWDID6u/BAAvknxK07b/+LgFi5jmed0u
m0yX9DFX8kz74qEULdRyPxSaepL/2E7ezdXtCrtY2myx1HGQ1mQd9/qiXWBD8zML
0un3b1UIpUPKyx7h+GmcR1JUhnSc2RexXjb76tfUU9s4nxk0Lf0jmMQCWSAXRHMM
Rral+wTK00MTNIt8ngjMcuzt1URm1TTU6LxyZYUcUvXxeC3u+6A0UXXZI5+NHUqL
JwEwyrhmd1mXXM4FKj5qSDDSWEA/iEgEBECAAKFAkUyZm8CGwAACgkQMWRaFO+
WXUuUwCfd6/EZwJNiawow1m5OU8v+uvUxEXAMPYyn76hzRDkHUVTMIITa/RO4sA=
=dqag
-----END PGP PUBLIC KEY BLOCK-----

C:\Documents and Settings\sensei>
```

Rys. 4 Wyeksportowany klucz w postaci znaków ASCII.

By dokonać tego samego z nakładki GPA należy w oknie głównym zaznaczyć interesujący nas

klucz, a następnie wybrać bądź to z menu *Keys* bądź z menu podręcznego (prawy klawisz myszy) *Export Key*.

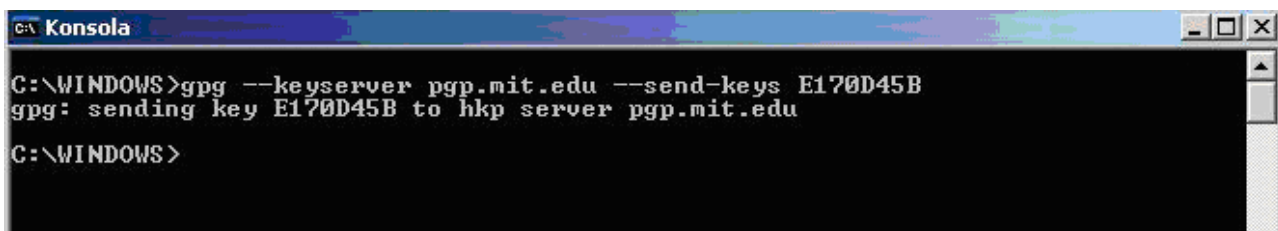
Eksport klucza do serwera kluczy

Serwery kluczy przechowują klucze publiczne i udostępniają je każdemu, kto o nie poprosi. Jest to rozwiązanie wygodne, zwłaszcza jeśli chcemy rozdystrybuować klucz większej liczbie osób. Istnieje wiele serwerów kluczy, jednak nie musimy wysyłać naszego klucza do każdego z nich. Zdecydowana większość, a przynajmniej popularne serwery, synchronizuje się ze sobą. W efekcie po pewnym czasie odpytanie któregośkolwiek z nich o wyeksportowany klucz zwróci spodziewany wynik.

W celu wyeksportowania klucza na serwer należy użyć polecenia:

gpg --keyserver SERWER_KLUCZY --send-keys ID_KLUCZA

gdzie ID_KLUCZA to jego identyfikator, a SERWER_KLUCZY jest nazwą domenową lub adresem wybranego serwera kluczy (jeżeli go nie wskażemy, to klucz zostanie wysłany na serwer domyślny).

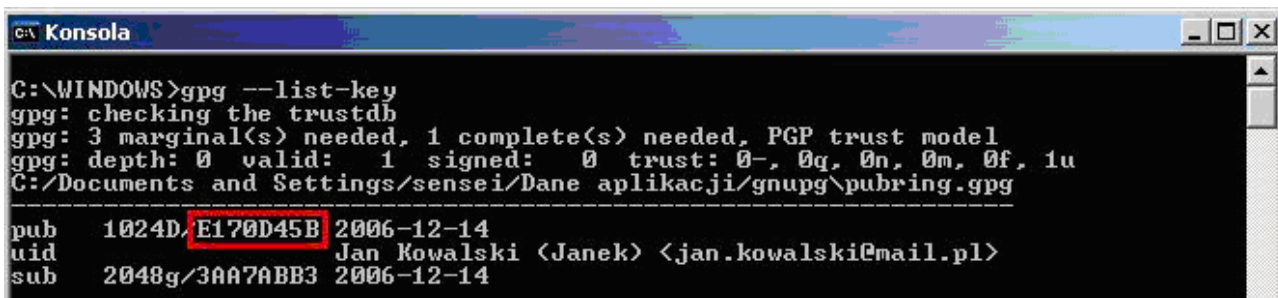


```
C:\WINDOWS>gpg --keyserver pgp.mit.edu --send-keys E170D45B
gpg: sending key E170D45B to hkp server pgp.mit.edu
C:\WINDOWS>
```

Rys.5 Eksport klucza na serwer kluczy.

Listę wszystkich kluczy wraz z identyfikatorami uzyskamy wywołując:

gpg --list-key



```
C:\WINDOWS>gpg --list-key
gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0 valid: 1 signed: 0 trust: 0-, 0q, 0n, 0m, 0f, 1u
C:/Documents and Settings/sensei/Dane aplikacji/gnupg/pubring.gpg
-----
pub   1024D, E170D45B  2006-12-14
uid           Jan Kowalski <Janek> <jan.kowalski@mail.pl>
sub   2048g/3AA7ABB3  2006-12-14
```

Rys. 6 Listowanie dostępnych kluczy. W ramce ID klucza

Popularne serwery kluczy

- pgp.mit.edu
- wwwkeys.eu.pgp.net
- keyserver.veridis.com
- keyserver.pgp.com

Operację eksportu klucza na serwer można dokonać także za pomocą GPA. W tym celu otworzymy menu podręczne na interesującym nas kluczu i wybierzmy pozycję *Send Key*. Listę serwerów i serwer domyślny można modyfikować w menu *Edit -> Preferences*.

Import klucza z pliku

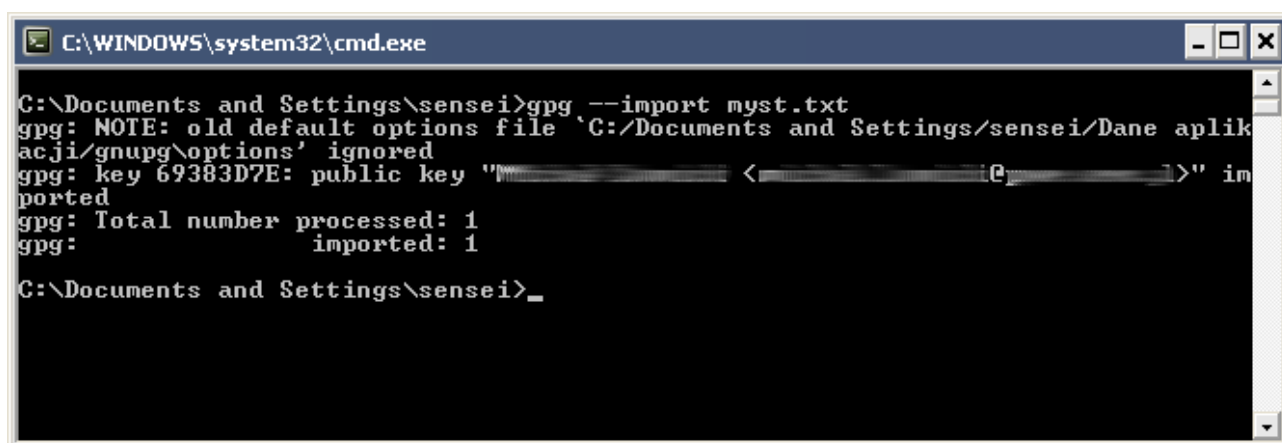
Jeśli chcemy przesłać jakiejś osobie zaszyfrowaną wiadomość, to także musimy zdobyć jej klucz publiczny. Jeśli dysponujemy już takim, zapisanym w pliku tekstowym to by go zaimportować do naszego GPG wystarczy zrobić:

gpg --import zapisany_klucz_publiczny.txt

Naturalnie należy podać poprawną ścieżkę do pliku lub musimy znajdować się w katalogu gdzie jest klucz. Co ciekawe w pliku zawierającym klucz mogą znajdować się także inne dane – wczytany zostanie jedynie blok pomiędzy znacznikami.

----- BEGIN PGP PUBLICKEY BLOCK-----

----- END PGP PUBLICKEY BLOCK-----



```
C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings\sensei>gpg --import myst.txt
gpg: NOTE: old default options file 'C:/Documents and Settings/sensei/Dane aplik
acji/gnupg/options' ignored
gpg: key 69383D7E: public key "[redacted]" imported
gpg: Total number processed: 1
gpg:          imported: 1
C:\Documents and Settings\sensei>_
```

Rys. 7. Import klucza zapisanego w pliku tekstowym.

Te same operacje wykonamy z GPA. Odpowiednią opcję służącą do importu obcego klucza publicznego z pliku tekstowego znajdziemy w menu kontekstowym lub *Keys -> Import Key*.

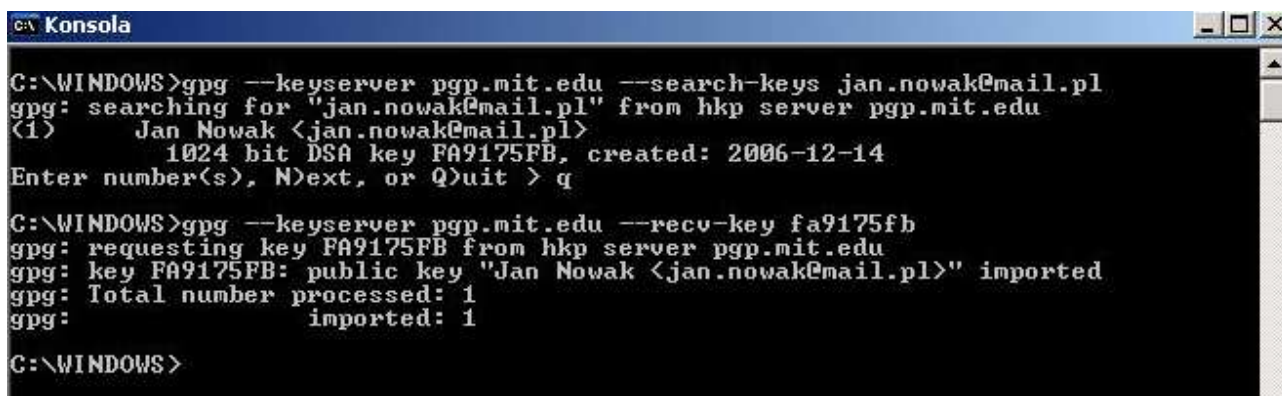
Import klucza z serwera kluczy

Importowanie klucza z serwerów jest znacznie wygodniejsze, nie wymaga uprzedniego dostarczenia klucza w formacie tekstowym. Najpierw musimy zlokalizować interesujący nas klucz. Załóżmy, że chcielibyśmy nawiązać bezpieczną korespondencję z Janem Nowakiem. Odnajdźmy najpierw jego klucz:

gpg --keyserver SERWER_KLUCZY --search-key adres_email

Gdy odnajdziemy właściwy klucz i poznamy jego ID, możemy go zaimportować:

gpg --keyserver SERWER_KLUCZY --recv-key ID_KLUCZA



```
C:\WINDOWS>gpg --keyserver pgp.mit.edu --search-keys jan.nowak@mail.pl
gpg: searching for "jan.nowak@mail.pl" from hkps server pgp.mit.edu
(1) Jan Nowak <jan.nowak@mail.pl>
    1024 bit DSA key FA9175FB, created: 2006-12-14
Enter number(s), N)ext, or Q)uit > q

C:\WINDOWS>gpg --keyserver pgp.mit.edu --recv-key fa9175fb
gpg: requesting key FA9175FB from hkps server pgp.mit.edu
gpg: key FA9175FB: public key "Jan Nowak <jan.nowak@mail.pl>" imported
gpg: Total number processed: 1
gpg:          imported: 1
C:\WINDOWS>
```

Rys. 8 Wyszukanie i import klucza z serwera.

Ta sama operacja w GPA sprowadza się do wywołania menu *Server -> Retrieve Key*. Uruchomi się wtedy nowe okienko w którym możemy wpisać identyfikator szukanego klucza.

Określanie poziomu zaufania do klucza

Jest jeszcze jedna czynność, którą powinniśmy wykonać po zaimportowaniu. Taki klucz domyślnie nie należy do grupy zaufanych. Jeśli będziemy z niego korzystać, otrzymamy ostrzeżenia, że jego autentyczność nie została potwierdzona. Jeśli jednak jesteśmy przekonani co do jego pochodzenia, to by pozbyć się ostrzeżeń wystarczy ustawić odpowiedni poziom zaufania względem niego. W konsoli wpiszmy:

gpg --edit-key NAZWA_KLUCZA

Wejdziemy w tryb interaktywny. Wydajmy w nim polecenie:

trust

Zostaniemy poproszeni o wybranie poziomu zaufania. Wybierzmy opcję przedostatnią, tj. *5 = I trust ultimately*.

Aby wyjść z trybu interaktywnego i zapisać wprowadzone zmiany wpiszmy:

quit

Alternatywnie ta sama czynność w GPA polega na wybraniu klucza z listy, a po uruchomieniu na nim menu podręcznego – wybraniu opcji *Set owner trust*.

Podpisywanie i weryfikacja podpisu cyfrowego

Dzięki OpenPGP otrzymujemy mechanizm weryfikacji poprawności i autorstwa istotnych dla nas danych. Wystarczy, że nadawca podpisze cyfrowo swoim kluczem prywatnym dane, których autentyczność sprawdzimy my, korzystając z jego klucza publicznego. Procedura wygląda następująco: obliczania jest wartość funkcji mieszającej dla danego zestawu danych, następnie ta wartość jest szyfrowana kluczem prywatnym nadawcy. Można ją odszyfrować tylko kluczem publicznym tego nadawcy. To upewnia nas o pochodzeniu. Po odszyfrowaniu możemy porównać wartość funkcji mieszającej nadawcy z wynikiem takiego haszowania u nas. Jeśli są zgodne, to znaczy że dane nie uległy zmianie. Mamy wtedy już całkowitą pewność, że to co otrzymaliśmy, po przejściu przez kanał komunikacyjny, zgadza się z oryginałem.

Podpisywanie pliku (podpis zostanie umieszczony na końcu pliku):

gpg --clearsign PLIK

Zostanie do tego użyty nasz domyślny klucz. Program poprosi nas o podanie naszego hasła do klucza, a wynikiem jego działania będzie plik PLIK.asc. Może on wyglądać następująco:


```
C:\WINDOWS\system32\cmd.exe

C:\>gpg --local-user Kowalski --clearsign test.txt

You need a passphrase to unlock the secret key for
user: "Jan Kowalski <Janek> <jan.kowalski@mail.pl>"
1024-bit DSA key, ID E170D45B, created 2006-12-14

C:\>type test.txt.asc
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

To jest wiadomosc testowa.
Ma dwie linijki.
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.5 (MingW32)

iD8DBQFFhRcyBpUJsuFw1FsRAcK9AJ9XDodRyS/oRf/3f2UUZOquXN4mkwCgge lw
D1nDnr1TeSKg4zR0cS6hXzE=
=i4PA
-----END PGP SIGNATURE-----

C:\>_
```

Rys. 9 Podpisywanie wiadomości wybranym kluczem. Listing wiadomości.

Jeżeli chcemy użyć do podpisu innego klucza niż nasz domyślny, wtedy dołączmy parametr `--local-user`. Jako argument przyjmuje nazwę wybranego klucza, tj. adres email wpisany przy generacji klucza, imię i nazwisko lub identyfikator.

gpg --local-user NAZWA --clearsign PLIK

Powyższe sposoby sprawdzają się kiedy podpisujemy pliki tekstowe. Jednak taka modyfikacja jak dodanie opisu do pliku binarnego uszkodzi go. Wtedy powinniśmy stosować podpis cyfrowy w osobnym pliku:

gpg --output PLIK_PODPISU.sig --detach-sign PLIK

Utworzony podpis będzie w formacie binarnym, nieczytelny dla człowieka. Jeśli chcemy by podpis był w formacie pliku ASCII dodajmy przełącznik

--armor

```
C:\WINDOWS\system32\cmd.exe

C:\>gpg --armor --output "texlive.iso.asc" --detach-sign "texlive.iso"

You need a passphrase to unlock the secret key for
user: "Jan Kowalski <Janek> <jan.kowalski@mail.pl>"
1024-bit DSA key, ID E170D45B, created 2006-12-14

C:\>type texlive.iso.asc
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.5 (MingW32)

iD8DBQBFhTUIBpUJsuFw1FsRApt6AJ9DMkcZTM+TY3I41+u22CFWIJBxegCfSx2C
01LEhsTRxliKJ/pWdj2NHSE=
=iCEy
-----END PGP SIGNATURE-----

C:\>_
```

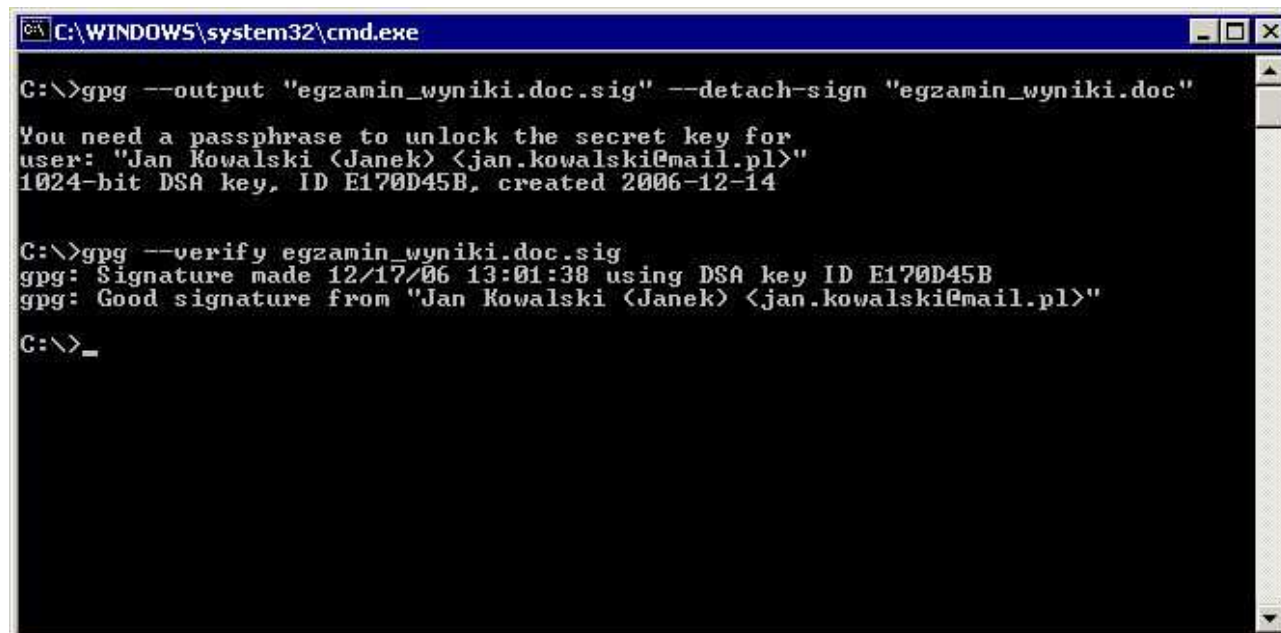
Rys. 10 Podpisywanie pliku binarnego. Podpis w formacie tekstowym w osobnym pliku.

Weryfikacja popisanego pliku jest również prosta. Jeżeli posiadamy klucz publiczny osoby podpisującej, to po wpisaniu w konsoli:

gpg --verify PODPIS (lub plik tekstowy z załączonym podpisem)

otrzymamy potwierdzenie autentyczności, jeśli plik nie uległ zmianie. Jeśli nie posiadamy w naszym zbiorze klucza publicznego osoby podpisującej, musimy go wcześniej zaimportować.

Wszystkie operacje podpisywania i weryfikacji można wykonać za pomocą menu kontekstowego eksploratora Windows (wystarczy wybrać odpowiednią opcję z menu kontekstowego pliku) lub aplikacji Kleopatra.



```
C:\WINDOWS\system32\cmd.exe

C:\>gpg --output "egzamin_wyniki.doc.sig" --detach-sign "egzamin_wyniki.doc"

You need a passphrase to unlock the secret key for
user: "Jan Kowalski <Janek> <jan.kowalski@email.pl>"
1024-bit DSA key, ID E170D45B, created 2006-12-14

C:\>gpg --verify egzamin_wyniki.doc.sig
gpg: Signature made 12/17/06 13:01:38 using DSA key ID E170D45B
gpg: Good signature from "Jan Kowalski <Janek> <jan.kowalski@email.pl>"

C:\>_
```

Rys. 11 Podpisywanie pliku binarnego. Weryfikacja podpisu.

Szyfrowanie i deszyfrowanie

OpenPGP zapewnia ochronę prywatności umożliwiając szyfrowanie wiadomości i danych kryptografią asymetryczną.

Plik zaszyfrujemy z konsoli w następujący sposób:

gpg --recipient ODBIORCA --output PLIK.gpg --encrypt PLIK

Nowo powstały plik binarny *PLIK.gpg* zawiera zaszyfrowane dane kluczem publicznym odbiorcy. Tylko on będzie mógł go zdeszyfrować swoim kluczem prywatnym.

Zaszyfrowany plik wynikowy może być również tekstowy, jeśli jego rozmiar nie jest zbyt wielki:

gpg --armor --recipient ODBIORCA --output PLIK.asc --encrypt PLIK

```
Konsola
C:\>gpg --recipient "Jan Nowak" --output autentyk.txt.gpg --encrypt autentyk.txt

C:\>gpg --armor --recipient "Jan Nowak" --output autentyk.txt.asc --encrypt autentyk.txt

C:\>type autentyk.txt.asc
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.4.5 (MingW32)

hQIOA5Deoqe j4xXoEaf8CotwySnrs6mUbxxtf73RUb6TqUTagWfJ8kHSeWcgszUd
uwurA6Ti7vvhliZFdJu8e5mf yEYKUdHlM4ScUcE7C9uZSSfWBh3Q3sWNm+yzRiRK
CUmo8DdfunnGgEcY3tDXUxfQeeI0yeG8EcT3E9jinYHGLfA0L67v7qb5EdC00LJx
HgaipC9SEky yBkwed6kAMfP7msCWHcH2pHYAAndS1cRuJmzFJg4L5UM7QY0KEXR3
RkRUXXZTDJGJUJhoLkEpy3AA7fYox7fMhD+qIn3Y72Ua6hA2900eKKI0FYrp/7X6v
H1ogfJb+biiZwMH55LfKRJmfOU7f+8sP8NED5BuNzQf/d/q57eW81DxWEUIPWL2J
Kh/5H12vsX4maZhuZxPuPjOGj21nxrkGrI8BKAtMiv4Axz3zmf0MjUepgNAUUiZw
00GkXZku5DUdF4G8BxzeQSMiXrAcnGIwWBhQ1/bXHN/RhpJ0JaxlanwttQn0BGU7
aHl3MJubqHdcu806CsQSDksusNEleOWHZZS6mDWeKiXJb2S1IMSSeG8TZEWH4y/
/iQwb8jzM6KzUuYUjD9keBItia3Hc8sqPH/cU5pxUGJTUZI870d0lh4plbB/c0AD
GxloRBSMgGZdZk927eUfHjnhwNiLK24D51SH1BJMpgHTVCizq9BcLPKkjuxqNNPN
CtJvAb7iwjkgSpnUIkBOJyDBCEZK4qJXneZUqi8tcCsJROQUBAvMoBnjLsSqUtD
z5RgdpQKs8U2cJxUvtIL3sj+30S5MCcUqPlXt8JL5d4MU2ca2UAXoczd86FwsM8I
w1H5nYcTPAYzaGE9BK0CgBxU
=KHsc
-----END PGP MESSAGE-----

C:\>_
```

Rys. 12 Szyfrowanie pliku. Postać tekstowa po zaszyfrowaniu.

Czasem chcemy przekazać szyfrowaną wiadomość więcej niż jednej osobie. Stwarza to pewien problem, gdyż dzielenie klucza prywatnego nie wchodzi w rachubę, a szyfrowanie dla każdego z osobna nieco kłopotliwe. GnuPG pozwala nam jednak zakodować wiadomość dla więcej niż jednego odbiorcy jednocześnie. Wtedy jeden z ich kluczy prywatnych wystarcza do rozszyfrowania, więc każdy może to zrobić indywidualnie. Składnia tej operacji różni się jedynie dodaniem większej ilości odbiorców `--recipient` :

```
gpg --armor --recipient ODBIORCA_1 --recipient ODBIORCA_2 --output PLIK.asc --encrypt PLIK
```

Teraz spróbujmy taki zaszyfrowany plik odkodować. Jeśli był on adresowany do nas, musimy posiadać odpowiedni klucz prywatny.

Poniższe polecenie :

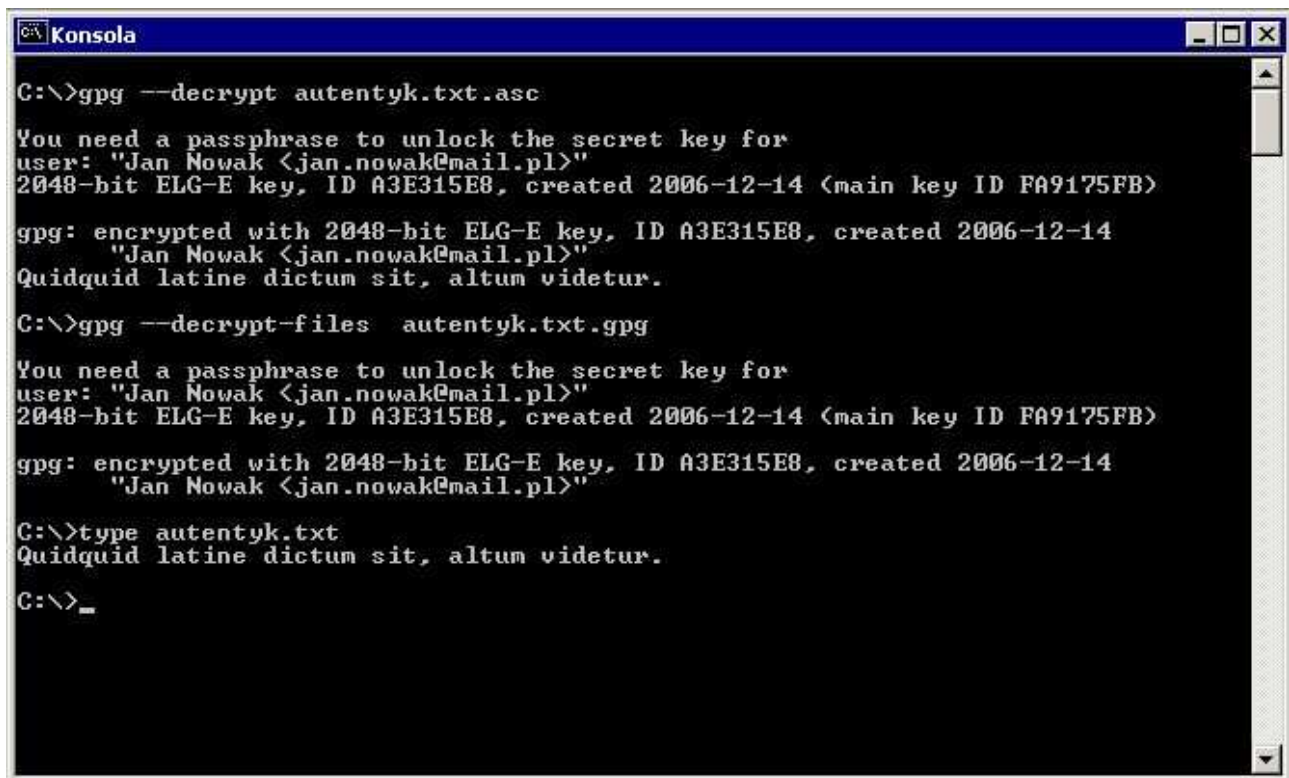
```
gpg --decrypt-files PLIK.gpg
```

spowoduje zdekodowanie zaszyfrowanego pliku. Po podaniu hasła do klucza plik rozkodowany plik zostanie umieszczony w bieżącym katalogu pod nazwą *PLIK*.

Jeśli szyfrowany plik był wiadomością tekstową, można użyć alternatywnie polecenia:

```
gpg --decrypt PLIK.asc
```

Nie wypakuje ono pliku z zaszyfrowanej postaci, lecz wyświetli zawartość na ekranie.



```
C:\>gpg --decrypt autentyk.txt.asc
You need a passphrase to unlock the secret key for
user: "Jan Nowak <jan.nowak@email.pl>"
2048-bit ELG-E key, ID A3E315E8, created 2006-12-14 (main key ID FA9175FB)

gpg: encrypted with 2048-bit ELG-E key, ID A3E315E8, created 2006-12-14
      "Jan Nowak <jan.nowak@email.pl>"
Quidquid latine dictum sit, altum videtur.

C:\>gpg --decrypt-files autentyk.txt.gpg
You need a passphrase to unlock the secret key for
user: "Jan Nowak <jan.nowak@email.pl>"
2048-bit ELG-E key, ID A3E315E8, created 2006-12-14 (main key ID FA9175FB)

gpg: encrypted with 2048-bit ELG-E key, ID A3E315E8, created 2006-12-14
      "Jan Nowak <jan.nowak@email.pl>"

C:\>type autentyk.txt
Quidquid latine dictum sit, altum videtur.

C:\>_
```

Rys. 13 Deszyfrowanie pliku.

Podobnie jak przy podpisywaniu tak i operacje szyfrowania/ deszyfrowania można przeprowadzić za pomocą menu kontekstowego pliku lub aplikacji Kleopatra.

Sieć zaufania (web of trust)

Posiadanie kluczy publicznych daje możliwość weryfikacji nadawcy przesyłki. Otrzymując podpisanego maila od profesora Tadeusza Lutego mamy pewność, że to właśnie On jest nadawcą przesyłki. Jest jednak pewien szczegół, który do tej pory pomijaliśmy. Załóżmy następujący scenariusz. Dowolna osoba generuje parę kluczy. Tworzy e-maila i w polu nadawca wpisuje “prof. Tadeusz Luty”, po czym podpisuje przesyłkę wygenerowanym kluczem prywatnym i wysyła go do nas. Udostępnia wygenerowany niedawno klucz publiczny na serwerze kluczy. My odbieramy tak spreparowanego maila. Nieświadomi niebezpieczeństwa pobieramy klucz publiczny z serwera, aby sprawdzić autentyczność nadawcy. Oczywiście weryfikacja przebiega pomyślnie, choć nie wiemy, że zostaliśmy oszukani.

Standard OpenPGP definiuje, a GnuPG implementuje, rozwiązanie tego problemu. Cała idea opiera się na podpisywaniu kluczy publicznych i przechodniości zaufania.

Budowanie sieci zaufania musi zacząć się od minimum dwóch osób, które znają identyfikatory swoich kluczy. Niech takimi osobami będą Kasia i Tomek. Załóżmy, że Roman usłyszał o możliwościach GPG i również pragnie, aby osoby do których wysyła maile mogły sprawdzić czy faktycznie pochodzą od nNiego. Oczywiście w tym celu komendą `gpg --gen-key` generuje klucz prywatny i publiczny. Jednak ma świadomość, że to nie wystarczy, aby odbiorcy jego poczty mogły być pewne jej autentyczności. W takim wypadku prosi Kasię, aby podpisała jego klucz publiczny swoim. Następnie wysyła maila do Tomka. Tomek, pobiera z serwera klucz Romka i widzi, że klucz jest podpisany przez Kasię. Tomek ufa Kasi, więc może także zaufać podpisowi Romka. Tym sposobem, jeśli Romek wyśle maila do dowolnej osoby, która ufa kluczowi Kasi zaufa także przesyłkom podpisanym pośrednio przez niego.

Teraz Tomek może złożyć podpis cyfrowy na kluczu Romka, dzięki czemu osoby, które ufają kluczowi Tomka będą także mogły zaufać kluczowi Romka.

Jak widać sieć zaufania jest zdecentralizowana i opiera się na relacjach między kluczami. Relacja

może być jednokierunkowa (osoba A podpisuje klucz osoby B) lub dwukierunkowa (osoby A i B wzajemnie podpisują swoje klucze). Każdy klucz zawiera tzw. sygnaturę. Sygnatura jest skrótem (np. MD5 lub SHA) klucza. Jeśli sygnatura pasuje do klucza weryfikacja wiadomości przebiega pozytywnie. Sygnatury tworzą ścieżki zaufania, dzięki którym możliwe staje się budowanie sieci zaufania. W poprzednim przykładzie Tomek zaufał kluczowi Romka, ponieważ po ścieżce zaufania zbudowanej przez Kasię, był pewny autentyczności klucza publicznego, który pobrał z serwera. Idealnie jest, kiedy każde podpisanie kluczy przebiega w sposób dwukierunkowy, dzięki czemu sieć zaufania może znacznie się rozszerzać.

Aby w GPG podpisać klucz należy przejść w tryb edycji.

```
C:\Program Files\GNU\GnuPG>gpg --edit-key pawel.pacana@pwr.wroc.pl
gpg (GnuPG) 1.4.5; Copyright (C) 2006 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.

pub 1024D/53BE5975  created: 2006-10-15  expires: never      usage: SC
sub 2048g/8AC6C876  created: 2006-10-15  expires: never      usage: E
[ unknown] (1). Paweł Pacana <pawel.pacana@pwr.wroc.pl>

Command> _
```

Przed podpisaniem klucza zaleca się sprawdzenie jego odcisku (fingerprint) i skontaktowanie się z właścicielem w celu jego weryfikacji.

```
Command> fpr
pub 1024D/53BE5975 2006-10-15 Paweł Pacana <pawel.pacana@pwr.wroc.pl>
Primary key fingerprint: B066 6420 BD4F 1F76 385C 70A8 3165 D168 53BE 5975

Command> _
```

Jeśli fingerprint jest prawidłowy można przystąpić do podpisania klucza publicznego danej osoby swoim kluczem prywatnym. W trakcie podpisywania zostaniemy poproszeni o wpisanie hasła klucza, którego używamy do podpisania.

```
Command> sign
pub 1024D/53BE5975  created: 2006-10-15  expires: never      usage: SC
trust: unknown      validity: unknown
Primary key fingerprint: B066 6420 BD4F 1F76 385C 70A8 3165 D168 53BE 5975
Paweł Pacana <pawel.pacana@pwr.wroc.pl>

Are you sure that you want to sign this key with your
key "Michał Lomnicki <michal.lomnicki@pwr.wroc.pl>" (2ED8BD74)

Really sign? (y/N) y

You need a passphrase to unlock the secret key for
user: "Michał Lomnicki <michal.lomnicki@pwr.wroc.pl>"
1024-bit DSA key, ID 2ED8BD74, created 2006-12-21

Command> _
```

Teraz można odesłać właścicielowi jego klucz wraz z naszym podpisem i poprosić o podpisanie naszego klucza.

Podpisywanie i szyfrowanie e-maili

Aby wykorzystać gpg z poziomu klienta pocztowego należy zainstalować odpowiednie wtyczki (są dostępne dla większości programów pocztowych).

Należy podkreślić, że szyfrowana jest treść maila i ewentualnie załączniki. Nagłówki wiadomości pozostają niezaszyfrowane (część z nich musi być dostępna dla serwerów poczty, część zmienia się podczas transportu), w związku z tym temat maila, który jest jednym z nagłówków, pozostanie niezaszyfrowany.

Domyślnie w mailach wykorzystywane jest tzw. PGP/inline. Informacje o kluczu użytym do zaszyfrowania przechowywane są w treści wiadomości. Stwarza to problemy z szyfrowaniem i podpisywaniem załączników oraz znaków spoza standardowej tablicy ASCII. W związku z tym powstało udoskonalenie w postaci PGP/MIME (PGP Multipurpose Internet Mail Extensions, opisane w RFC 2015 i RFC 3156). W mailach korzystających z PGP/MIME informacje o PGP zawarte są w nagłówkach, a nie w ciele wiadomości. Większość klientów e-mail obsługuje PGP/MIME, więc jeśli mamy pewność, że korespondujemy z osobą, która używa takiego klienta zalecane jest włączenie PGP/MIME. Popularnym programem pocztowym, która nie wspiera PGP/MIME jest Outlook.

Należy pamiętać, że zaszyfrowanie załącznika może przynieść zaskakujące efekty. Przykładowo jeśli w zaszyfrowanym załączniku znajduje się malware, przejdzie on niepostrzeżenie przez skaner antywirusowy znajdujący się na w systemie lub na bramach pocztowych.

Kolejnym problemem jest odczytywanie e-maili w webowych klientach poczty. Aby odszyfrować wiadomość należałoby umieścić klucz prywatny na serwerze WWW, tak aby skrypty klienta mogły odszyfrować e-maila. Zgodnie z zasadą głoszącą, że klucz prywatny powinien być dostępny tylko dla jego właściciela, trzymanie klucza na publicznie dostępnym serwerze jest niedopuszczalne. Oczywiście uniemożliwia to odszyfrowanie wiadomości z poziomu czytnika webowego.

Bibliografia

- OpenPGP Message Format. <http://www.ietf.org/rfc/rfc2440.txt>
- Dokumentacja GnuPG. <http://www.gnupg.org/>
- <http://pl.wikipedia.org/wiki/Kryptografia>
- SPKI Certificate Theory. <http://www.ietf.org/rfc/rfc2693.txt>
- Lista dyskusyjna Enigmail. <http://mozdev.org/mailman/listinfo/enigmail/>
- MIME Security with OpenPGP. <http://www.ietf.org/rfc/rfc3156.txt>
- <http://www.openpgp.org/>
- <http://www.philzimmermann.com>