Praktyczne wykorzystanie narzędzi szyfrujących

DO_PRZYGOTOWANIA

Samodzielnie należy przed tym laboratorium przygotować:

- przeczytać Gnu Privacy Guard Mini Howto [https://web.archive.org/web/20161208111121/http://www.virtualblueness.net/GPG-HOWTO/GPGMiniHowto.html] (fragment po polsku [https://web.archive.org/web/20161208111121/http://7thguard.net/archiwalne_pliki/gpg.html]))
- przeglądnąć The GNU Privacy Handbook [https://web.archive.org/web/20161208111121/http://www.gnupg.org/gph/en/manual.html]

WPROWADZENIE

Podstawowe pojęcia

- kryptografia
- kryptologia
- szyfr (ang. *cipher*)
- tekst jawny,
- kryptogram,
- klucz (ang. *key*)
- hash, skrót (ang. *hash, digest*)
- szyfrowanie bez klucza: ROT13 (ROT3 - Szyfr C.I.Cezara)
- szyfrowanie z kluczem symetrycznym, np.: Crypt, DES, 3DES, Idea, Blowfish
- szyfrowanie z kluczem asymetrycznym (publicznym), np: RSA, DSA, ElGamal
- algorytmy generowania skrótu, np: MD5, SHA
- podpis elektroniczny

Szyfrowanie z kluczem symetrycznym

Nadawca i odbiorca mają ten sam klucz.

Nadawca:

- przygotowuje tekst wiadomości,
- szyfruje całość wiadomości kluczem,
- wysyła szyfrogram do odbiorcy.

Odbiorca:

- rozszyfrowuje szyfrogram przy pom. klucza
- otrzymuje tekst wiadomości

N. i O. mają pary własnych kluczy (Pub/Prv). Wymieniają się publicznymi.

Nadawca:

- przygotowuje tekst wiadomości,
- szyfruje całość wiadomości: tekst i zaszyfrowany skrót, kluczem publicznym odbiorcy,
- wysyła szyfrogram do odbiorcy.

Odbiorca:

- rozszyfrowuje szyfrogram przy pomocy swojego klucza prywatnego,
- otrzymuje tekst wiadomości oraz jej zaszyfrowany skrót,

Narzędzie mcrypt

Przykład z mcrypt:

```
$ mcrypt --list
blowfish (56): ofb cfb nofb cbc ecb ncfb ctr
des (8): ofb cfb nofb cbc ecb ncfb ctr
blowfish-compat (56): ofb cfb nofb cbc ecb ncfb ctr
tripledes (24): ofb cfb nofb cbc ecb ncfb ctr
enigma (13): stream
```

Szyfrowanie mcrypt

```
$ mcrypt -a des moj_plik
File: moj_plik
Enter the passphrase (maximum of 512 characters)
Please use a combination of upper and lower case letters and numbers.
Enter passphrase:
Re-Enter passphrase:
File moj_plik was encrypted.
```

Deszyfrowanie mcrypt

```
$ mdecrypt moj_plik.nc
File: moj_plik.nc
Enter passphrase:
File moj_plik.nc was decrypted.
```

Wykorzystanie funkcji skrótu

```
$ md5sum moj_plik
02a5c225dab5aaf2801c896c22203ac6  moj_plik
$ md5sum /bin/bash
603492287ea2f26b9fb9266c961d5b0c  /bin/bash
$ du -h /bin/bash moj_plik
503k    /bin/bash
2.0k    moj_plik
```

Nadawca:

- przygotowuje tekst wiadomości,
- wylicza skrót wiadomości (np. MD5),
- szyfruje skrót przy pomocy swojego klucza prywatnego,
- szyfruje całość wiadomości: tekst i zaszyfrowany skrót, kluczem publicznym odbiorcy,
- wysyła szyfrogram do odbiorcy.

To jest wersja podpisu z szyfrowaniem wiadomości.

Odbiorca:

- rozszyfrowuje szyfrogram przy pomocy swojego klucza prywatnego,
- otrzymuje tekst wiadomości oraz jej zaszyfrowany skrót,
- wylicza skrót wiadomości,
- rozszyfrowuje skrót przy pomocy klucza publicznego nadawcy,
- porównuje wyliczony skrót z rozszyfrowanym,
- jeżeli są zgodne potwierdzone zostają tożsamość nadawcy i spójność przesłanych informacji.

Gnu Privacy Guard

Najważniejsze etapy używania programu:

- wygenerowanie kluczy
- zarządzanie kluczami: export swojego klucza, import czyjegoś, wyświetlanie kluczy, edycja
- szyfrowanie pliku
- deszyfrowanie pliku
- podpisywanie pliku
- weryfikacja podpisu
- narzędzia

Gnu Privacy Guard - klucz

```
$ gpg --gen-key
gpg (GnuPG) 1.0.6; Copyright (C) 2001 Free Software Foundation, Inc.

gpg: /home/gjn/.gnupg: directory created
gpg: /home/gjn/.gnupg/options: new options file created
gpg: you have to start GnuPG again, so it can read the new options file
$ gpg --gen-key
gpg (GnuPG) 1.0.6; Copyright (C) 2001 Free Software Foundation, Inc.

gpg: /home/gjn/.gnupg/secring.gpg: keyring created
gpg: /home/gjn/.gnupg/pubring.gpg: keyring created
Please select what kind of key you want:
   (1) DSA and ElGamal (default)
   (2) DSA (sign only)
   (4) ElGamal (sign and encrypt)
Your selection?
DSA keypair will have 1024 bits.
```

```
What keysize do you want? (1024)
Requested keysize is 1024 bits
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0)
Key does not expire at all
Is this correct (y/n)? y

You need a User-ID to identify your key; the software constructs the user id
from Real Name, Comment and Email Address in this form:
    "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Real name: Grzegorz J. Nalepa
Email address: gjn@agh.edu.pl
Comment: Akademia Gorniczo-Hutnicza
You selected this USER-ID:
    "Grzegorz J. Nalepa (Akademia Gorniczo-Hutnicza) <gjn@agh.edu.pl>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? o
You need a Passphrase to protect your secret key.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
+++++.+++++.+++++..+++++.+++++.++++++++++++++++++++++++++++++++++++++++++++++
public and secret key created and signed.
```

Zarzadzanie kluczami GPG

```
$ gpg -a --export gjn > klucz_publiczny_gjn.asc
$ file klucz_publiczny_gjn.asc
klucz_publiczny_gjn.asc: GPG key public ring

$ gpg --list-keys
/home/gjn/.gnupg/pubring.gpg
----------------------------
pub  1024D/51DDA662 2003-01-08 Grzegorz J. Nalepa (Akademia Gorniczo-Hutnicza) <gjn@agh.edu.pl>
sub  1024g/D9D30169 2003-01-08

$ gpg --import ~/Igor.asc
gpg: key 65DE877A: public key imported
gpg: Total number processed: 1
gpg:               imported: 1

$ gpg --list-keys
/home/gjn/.gnupg/pubring.gpg
----------------------------
pub  1024D/51DDA662 2003-01-08 Grzegorz J. Nalepa (Akademia Gorniczo-Hutnicza) <gjn@agh.edu.pl>
sub  1024g/D9D30169 2003-01-08
pub  1024D/65DE877A 2001-10-22 Igor Wojnicki <wojnicki@agh.edu.pl>
sub  1024g/1819CFA3 2001-10-22

$ gpg --edit-key Wojnicki
gpg (GnuPG) 1.0.6; Copyright (C) 2001 Free Software Foundation, Inc.

pub  1024D/65DE877A  created: 2001-10-22 expires: never      trust: -/q
sub  1024g/1819CFA3  created: 2001-10-22 expires: never
(1). Igor Wojnicki <wojnicki@agh.edu.pl>
```

```
with your key: "Grzegorz J. Nalepa (Akademia Gorniczo-Hutnicza) <gjn@agh.edu.pl>"
Really sign? y

You need a passphrase to unlock the secret key for
user: "Grzegorz J. Nalepa (Akademia Gorniczo-Hutnicza) <gjn@agh.edu.pl>"
1024-bit DSA key, ID 51DDA662, created 2003-01-08
Command> quit
Save changes? yes

$ gpg -a --export gjn
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.0.6 (GNU/Linux)

mQGiBDu3Bh4RBACLZSaCMKZsYc1XIxFC3WyViY3qREBdC5Wo9D77ppBbmIHlWG/8
```

Szyfrowanie z kluczem asymetrycznym (GPG)

```
$ gpg --encrypt --armor -o moja_wiadomosc.asc moja_wiadomosc
You did not specify a user ID. (you may use "-r")

Enter the user ID: Nalepa
$ head -5 moja_wiadomosc.asc
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.0.6 (GNU/Linux)
Comment: For info see http://www.gnupg.org

hQEOA7Ek7eXZ0wFpEAP9ErVtkzekylOUMOyf2d+tw17eVUd7w5OGK++AZ6IZRXLB
$ file moja_wiadomosc.asc
moja_wiadomosc.asc: PGP armored text message
```

Deszyfrowanie GPG

```
$ gpg --decrypt -o moja_wiadomosc moja_wiadomosc.asc

You need a passphrase to unlock the secret key for
user: "Grzegorz J. Nalepa (Akademia Gorniczo-Hutnicza) <gjn@agh.edu.pl>"
1024-bit ELG-E key, ID D9D30169, created 2003-01-08 (main key ID 51DDA662)

gpg: encrypted with 1024-bit ELG-E key, ID D9D30169, created 2003-01-08
      "Grzegorz J. Nalepa (Akademia Gorniczo-Hutnicza) <gjn@agh.edu.pl>"
```

Podpisywanie GPG

```
$ gpg --sign --armor -o moja_wiadomosc.sign moja_wiadomosc

You need a passphrase to unlock the secret key for
user: "Grzegorz J. Nalepa (Akademia Gorniczo-Hutnicza) <gjn@agh.edu.pl>"
1024-bit DSA key, ID 51DDA662, created 2003-01-08

$ head -5 moja_wiadomosc.sign
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.0.6 (GNU/Linux)
Comment: For info see http://www.gnupg.org

owGNlL1v1DAYxo9+CIjEcExIMFiKkBgCTlLdtYW7Y0BMwFDEwkLlOA4XiD9kO73
```

Podpis elektroniczny (GPG)

```
$ grep -A1 '^-----BEGIN'  moja_wiadomosc.asc
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1
--
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.0.6 (GNU/Linux)

$ grep --verify moja_wiadomosc.asc
gpg: Signature made Fri Nov 29 00:55:22 2003 CET using DSA key ID 51DDA662
gpg: Good signature from "Grzegorz J. Nalepa (Akademia Gorniczo-Hutnicza) <gjn@agh.edu.pl>"
```

```
$ gpg --verify patch-2.4.20.gz.sign patch-2.4.20.gz
gpg: Signature made Fri Nov 29 00:57:46 2002 CET using DSA key ID 517D0F0E
gpg: Good signature from "Linux Kernel Archives Verification Key <ftpadmin@kernel.org>"
```

Podpisywanie i szyfrowanie GPG

```
$ gpg -sear Wojnicki -o moja_wiadomosc_do_igora.asc moja_wiadomosc_do_igora

You need a passphrase to unlock the secret key for
user: "Grzegorz J. Nalepa (Akademia Gorniczo-Hutnicza) <gjn@agh.edu.pl>"
1024-bit DSA key, ID 51DDA662, created 2003-01-08
```

Uwierzytelnianie SSH

- serwer SSH ma parę kluczy (publiczny i prywatny),
- klient rozpoczyna transmisję szyfrowaną przy pomocy klucza publicznego serwera (algorytm asymetryczny, np. DSA/RSA),
- podejmowana jest próba uwierzytelnienia użytkownika, (jedną z wybranych metod: klucz, hasło, OTP, itp.), najczęściej:
- jeżeli to możliwe, przeprowadzane jest uwierzytelnienie przy pomocy klucza publicznego użytkownika,
- jeżeli nie jest możliwe uwierzytelnianie przy pomocy kluczy realizowane jest uwierzytelnianie poprzez hasło,
- w przypadku pomyślnego uwierzytelnienia reszta sesji jest szyfrowana przy pomocy algorytmu symetrycznego (3DES, Idea, Blowfish) z kluczem losowanym na nowo dla każdej sesji,
- transmisja może być opcjonalnie kompresowana,
- zamiast uruchomienia powłoki interaktywnej można uruchomić dowolny program.

Uwierzytelnianie przez klucz publiczny SSH

- Wymaga wygenerowania pary kluczy na maszynie z której się logujemy.
- Przeniesienia kluczy publicznych na maszyny na które się logujemy.
- Klucze są zabezpieczane passfrazami.
- Zawierają również informacje o koncie na którym zostały stworzone.
- Pozwala to na uwierzytelenienie: użytkownika i maszyny (miejsca) z którego się loguje.
- Uwierzytelnienie jest dwustopniowe: hasło (passfraza) i token (klucz).

wygenerowanie kluczy na maszynie z której się logujemy:

```
Your identification has been saved in /home/gjn/.ssh/id_dsa.
Your public key has been saved in /home/gjn/.ssh/id_dsa.pub.
The key fingerprint is:
b2:2e:01:08:31:21:43:04:87:65:dc:ea:1c:dc:07:95 gjn@enterprise
```

skopiowanie klucza publicznego na maszyny na które się logujemy:

```
enterprise$ scp .ssh/id_dsa.pub gjn@voyager:.ssh/id_dsa.pub-enterpise
gjn@voyager's password:
id_dsa.pub           100% |*****************************|   604       00:00
```

dopisanie klucza publicznego do listy kluczy autoryzowanych

```
enterprise$ ssh voyager 'cat .ssh/id_dsa.pub-enterpise >>
                         .ssh/authorized_keys'
gjn@voyager's password:
voyager$
```

logowanie przy pomocy klucza:

```
enterprise$ ssh voyager
Enter passphrase for key '/home/gjn/.ssh/id_dsa':
Linux voyager 2.4.20 #4 Sun Jan 5 20:32:43 CET 2003 i586 unknown
voyager$
```

przekazywanie autoryzacji przez SSH Agent

```
enterprise$ ssh-add .ssh/id_dsa
Enter passphrase for /home/gjn/.ssh/id_dsa:
Identity added: /home/gjn/.ssh/id_dsa (/home/gjn/.ssh/id_dsa)
Identity added: .ssh/id_dsa (.ssh/id_dsa)
enterprise$ ssh [-A] voyager
Linux enterprise 2.4.20 #4 Sun Jan 5 20:32:43 CET 2003 i586 unknown
$ voyager
```

ĆWICZENIA

1 GnuPG

Uwaga: ćwiczymy na studencie!

1. wygenerować swój klucz, `gpg --gen-key`, *NIE* należy zmieniać proponowanych wartości domyślnych (poza pierwszą algorytm (2) = `DSA and ElGamal`)
2. *NIE* należy podpisywać kluczy (`--sign`)
3. zaszyfrować i rozszyfrować wybrany plik dla siebie,
4. wymienić się kluczami z sąsiadką/em, za/rozszyfrować dla siebie pliki, (należy wyeksportować swój klucz publiczny, przekazać drugiej osobie, i zaimportować jej/jego wyeksportowany klucz publiczny):
    - *PRZED* przystąpieniem do ćwiczenia przechodzimy do katalogu */tmp*: `cd /tmp`
    - wszystkie pliki (klucze, pliki do zaszyfrowania) mają się znajdować w tym katalogu

- import klucza: `--import`

5. podpisać elektronicznie plik,
6. ew. podpisać elektronicznie plik i zaszyfrować dla drugiej osoby.

2 SSH

1. Prześledzić nawiązywanie połączenia z serwerem SSH (logować się dodając opcję -v (ssh -v [konto@]maszyna))
2. Wygenerować na maszynie A parę kluczy, przenieść publiczny na maszynę B, zalogować się z A na B przy pomocy klucza.
3. Co należy zrobić, aby w danej sesji (na A) przy kolejnym logowaniu na B nie musieć podawać passfrazy do klucza?

3 Hashe

1. Wyliczyć przy pomocy `md5sum` skrót wybranego pliku.
2. jak wyżej, tylko dla pliku o innej długości, porównać długość hasha.
3. Zmodyfikować wcześniej używany plik, np. `echo a » plik`, wyliczyć hash i porównać z wcześniejszym.
4. Wyliczyć hashe dla podanej grupy plików (przy pomocy jednego wywołania md5sum) i zapisać je do pliku *MD5SUMS*.

4 Mcrypt

Przy pomocy wybranego algorytmu symetrycznego za/rozszyfrować plik za pomocą mcrypt.

5 Narzędzia GPG

Pracę z GPG mogą wspomagać:

- GPA GNU Privacy Assistant [https://web.archive.org/web/20161208111121/http://www.gnupg.org/related_software/gpa] wspomaga zarządzanie kluczami i szyfrowanie plików
- EnigMail [https://web.archive.org/web/20161208111121/http://enigmail.mozdev.org/] integruja GPG z klientami pocztowymi Mozilla
- inne programy…http://www.gnupg.org/related_software [https://web.archive.org/web/20161208111121/http://www.gnupg.org/related_software]

Pakiety

```
gnupg gnupg-doc gpa mcrypt openssh-client openssh-server
```