

Sprawozdanie z modułu 1

Zadanie 1

Jesteś administratorem systemów w firmie Northwind Traders (nwtraders.msft).

Otrzymałeś prośbę o przetestowanie w serwerze Linux SRV1 działania wbudowanych implementacji algorytmów haszujących SHA1 oraz SHA256.

Twoim zadaniem jest więc:

- utworzenie w serwerze Linux pliku: `$ cd /home/user$ echo "To jest plik." > plik.txt` (Uwaga: "\$" na początku powyższej linii oznacza założenie, że jesteśmy zalogowani jako zwykły użytkownik "user")
 - wygenerowanie skrótu z wykorzystaniem SHA1 dla pliku "plik.txt": `$ cd /home/user$ sha1sum /home/user/plik.txt`
 - wygenerowanie skrótu z wykorzystaniem SHA256 dla pliku "plik.txt": `$ cd /home/user$ sha256sum /home/user/plik.txt` i porównać powstały skrót z wygenerowanym wcześniej z wykorzystaniem SHA1
 - zmodyfikować plik "plik.txt" zamieniając kropkę na wykrzyknik: `$ cd /home/user$ echo "To jest plik!" > plik.txt`
 - wygenerowanie skrótu z wykorzystaniem SHA1 oraz SHA256 dla pliku "plik.txt": `$ cd /home/user$ sha1sum /home/user/plik.txt$ sha256sum /home/user/plik.txt` i porównać powstałe skróty z wygenerowanymi wcześniej przed dokonaną modyfikacją pliku
 - zmodyfikować plik "plik.txt" zamieniając wykrzyknik spowrotem na kropkę: `$ cd /home/user$ echo "To jest plik." > plik.txt`
 - wygenerowanie skrótu z wykorzystaniem SHA1 oraz SHA256 dla pliku "plik.txt": `$ cd /home/user$ sha1sum /home/user/plik.txt$ sha256sum /home/user/plik.txt` i porównać powstałe skróty z wygenerowanymi we wcześniejszych krokach
-

```
user@SRV1: ~
user@SRV1:~$ pwd
/home/user
user@SRV1:~$ echo "To jest plik." > plik.txt
user@SRV1:~$ shasum plik.txt
be4dcdf9927b1498e9a8a6d4b74b51d146c3deb0  plik.txt
user@SRV1:~$ sha256sum plik.txt
99e37d2fa954eae073a9510e54983c66b8fcb38ac11b75dd08a44fc739838354  plik.txt
user@SRV1:~$ echo "To jest plik!" > plik.txt
user@SRV1:~$ shasum plik.txt
f86dee5ab7e941be34924870077e218fea448330  plik.txt
user@SRV1:~$ sha256sum plik.txt
9160607d3bcb8245da9af8fd2f2849a8e13e7d9b7368e9383141a34450066b5d  plik.txt
user@SRV1:~$ echo "To jest plik." > plik.txt
user@SRV1:~$ shasum plik.txt
be4dcdf9927b1498e9a8a6d4b74b51d146c3deb0  plik.txt
user@SRV1:~$ sha256sum plik.txt
99e37d2fa954eae073a9510e54983c66b8fcb38ac11b75dd08a44fc739838354  plik.txt
user@SRV1:~$
```

Pliki o takej samej zawartości tworzą takie same hashe, zmiana nawet jednego bitu zmienia kompletnie cały hash.

sha256 tworzy dłuższy hash od sha1.

Zadanie 2

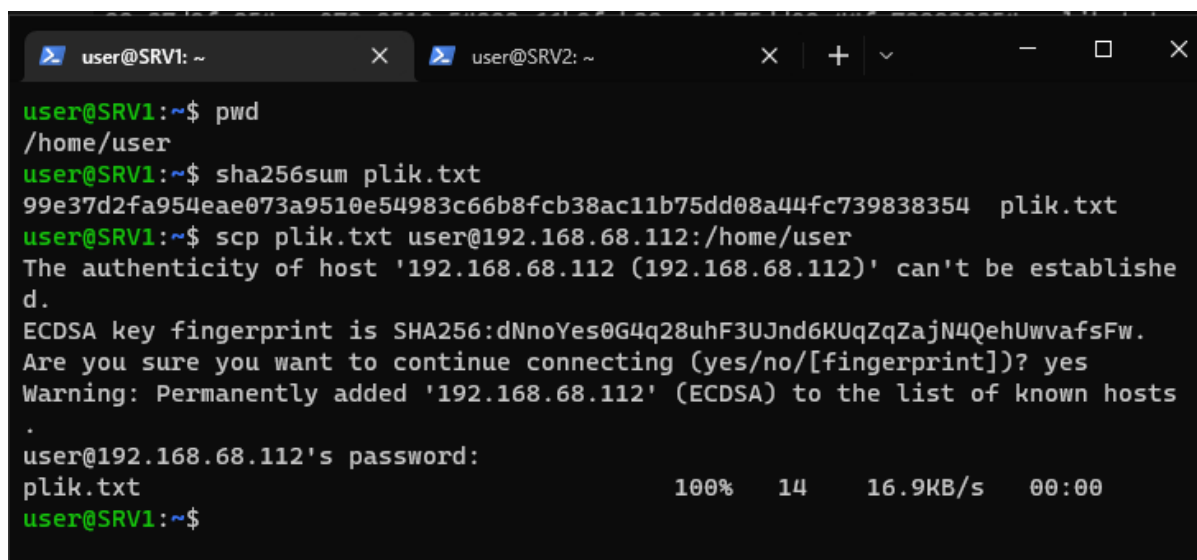
Jesteś administratorem systemów w firmie Northwind Traders (nwtraders.msft).

Otrzymałeś prośbę o przetestowanie funkcjonalności weryfikacji integralności danych, z wykorzystaniem wbudowanej implementacji algorytmu haszującego SHA256, przy kopiowaniu danych pomiędzy różnymi urządzeniami.

Twoim zadaniem jest więc:

- zalogować się w serwerze Linux SRV1 na użytkownika "user", i następnie wygenerować skrót z pliku "plik.txt" z wykorzystaniem algorytmu haszującego SHA256, a następnie przekopiować tenże plik do katalogu domowego użytkownika "user" w serwerze Linux SRV2: `$ cd /home/user$ sha256sum /home/user/plik.txt$ scp plik.txt user@172.16.0.2:/home/user`
- zalogować się w serwerze Linux SRV2 na użytkownika "user", i następnie wygenerować skrót z pliku "plik.txt" z wykorzystaniem algorytmu haszującego SHA256: `$ cd /home/user$ sha256sum /home/user/plik.txt` i porównać powstały skrót z wygenerowanym wcześniej w serwerze SRV1

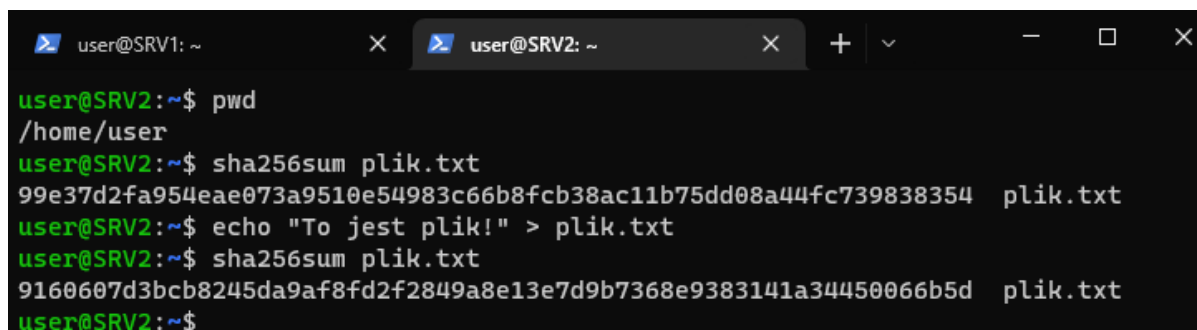
- zmodyfikować w serwerze Linux SRV2 plik "plik.txt" zamieniając kropkę na wykrzyknik: `$ cd /home/user$ echo "To jest plik!" > plik.txt`
- wygenerować w serwerze Linux SRV2 skrót z pliku "plik.txt" z wykorzystaniem algorytmu haszującego SHA256: `$ cd /home/user$ sha256sum /home/user/plik.txt` porównać powstały skrót z wygenerowanym wcześniej w serwerze SRV1



```

user@SRV1: ~$ pwd
/home/user
user@SRV1:~$ sha256sum plik.txt
99e37d2fa954eae073a9510e54983c66b8fcb38ac11b75dd08a44fc739838354 plik.txt
user@SRV1:~$ scp plik.txt user@192.168.68.112:/home/user
The authenticity of host '192.168.68.112 (192.168.68.112)' can't be established.
ECDSA key fingerprint is SHA256:dNnoYes0G4q28uhF3UJnd6KUqZqZajN4QehUwvafsFw.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.68.112' (ECDSA) to the list of known hosts
user@192.168.68.112's password:
plik.txt                                100% 14    16.9KB/s   00:00
user@SRV1:~$

```



```

user@SRV2: ~$ pwd
/home/user
user@SRV2:~$ sha256sum plik.txt
99e37d2fa954eae073a9510e54983c66b8fcb38ac11b75dd08a44fc739838354 plik.txt
user@SRV2:~$ echo "To jest plik!" > plik.txt
user@SRV2:~$ sha256sum plik.txt
9160607d3bcb8245da9af8fd2f2849a8e13e7d9b7368e9383141a34450066b5d plik.txt
user@SRV2:~$

```

Hash jest taki sam na obu maszynach. Zmiana treści pliku zmienia jego hash, w naszym wypadku wygenerowało taki sam hash jak w poprzednim zadaniu.

Zadanie 3

1. Znajdź w sieci internet 3 różne pliki do pobrania (na 3 różnych stronach internetowych), dla których podany został również odcisk palca/skrót/hasz, i pobierz te pliki.
2. Następnie wejdź na stronę: <http://www.hashemall.com/> i wykorzystaj ją do zweryfikowania odcisku palca/skrótu z pobranych plików.

Znalezienie strony z odciskami palca:

- <https://www.kali.org/get-kali/#kali-installer-images>
 - SHA256sum:
ae977f455924f0268fac437d66e643827089b6f8dc5d76324d6296eb11d997fd
 - SHA256sum z komputera:
ae977f455924f0268fac437d66e643827089b6f8dc5d76324d6296eb11d997fd

```
PS C:\Users\blumi> certutil -hashfile C:\Users\blumi\Downloads\kali-linux-2022.3-installer-amd64.iso SHA256
SHA256 hash of C:\Users\blumi\Downloads\kali-linux-2022.3-installer-amd64.iso:
ae977f455924f0268fac437d66e643827089b6f8dc5d76324d6296eb11d997fd
CertUtil: -hashfile command completed successfully.
```

- <https://parrotlinux.org/download/>
 - MD5sum: d1c4daef7c69db144f032ee6f0ee0026
 - MD5sum z komputera: d1c4daef7c69db144f032ee6f0ee0026

```
PS C:\Users\blumi> certutil -hashfile C:\Users\blumi\Downloads\Parrot-home-5.1_amd64.iso MD5
MD5 hash of C:\Users\blumi\Downloads\Parrot-home-5.1_amd64.iso:
d1c4daef7c69db144f032ee6f0ee0026
CertUtil: -hashfile command completed successfully.
```

- <https://www.python.org/downloads/release/python-3110/>
 - MD5sum: 4fe11b2b0bb0c744cf74aff537f7cd7f
 - MD5sum z komputera: 4fe11b2b0bb0c744cf74aff537f7cd7f

```
PS C:\Users\blumi> certutil -hashfile C:\Users\blumi\Downloads\python-3.11.0-amd64.exe MD5
MD5 hash of C:\Users\blumi\Downloads\python-3.11.0-amd64.exe:
4fe11b2b0bb0c744cf74aff537f7cd7f
CertUtil: -hashfile command completed successfully.
```

Zadanie 4

Jesteś administratorem systemów w firmie Northwind Traders (nwtraders.msft).

Otrzymałeś prośbę o przekopiowanie nowego hasła dla użytkownika jgula z serwera Linux SRV1 do serwera Linux SRV2.

Twoim zadaniem jest więc:

- zalogować się w serwerze Linux SRV1 oraz Linux SRV2 na użytkownika "root", i następnie utworzyć nowego użytkownika *jgula*:# *adduser jgula*przetestować również możliwość prawidłowego zalogowania się na utworzonego użytkownika w obydwu systemach
- w serwerze Linux SRV1 zmienić hasło dla użytkownika *jgula* (na dowolne inne):# *passwd jgula*przetestować również możliwość prawidłowego zalogowania się na użytkownika *jgula* w obydwu systemach po dokonanej modyfikacji
- wyedytować w serwerze Linux SRV2 plik */etc/shadow*, a następnie znaleźć linię z konfiguracją dla użytkownika "*jgula*", i całą linię przekopiować do identycznego pliku w serwerze Linux SRV2 (zamieniając adekwatną tam linię z konfiguracją dla użytkownika *jgula*)
- przetestować w serwerze Linux SRV2 możliwość prawidłowego zalogowania się na użytkownika *jgula* po dokonanej modyfikacji (ze zwróceniem uwagi na to czy możliwe jest zalogowanie się na hasło zmienione wcześniej w systemie Linux SRV1)

```

jgula@SRV1: /root
jgula@SRV2: /root

root@SRV1:~# adduser jgula
Dodawanie użytkownika "jgula"...
Dodawanie nowej grupy "jgula" (1001)...
Dodawanie nowego użytkownika "jgula" (1001) w grupie "jgula"...
Tworzenie katalogu domowego "/home/jgula"...
Kopiowanie plików z "/etc/skel" ...
Nowe hasło:
Proszę ponownie wpisać nowe hasło:
passwd: hasło zostało zmienione
Zmieniam informację o użytkowniku jgula
Wpisz nową wartość lub wciśnij ENTER by przyjąć wartość domyślną
    Imię i nazwisko []: Jacek Gula
    Numer pokoju []:
    Telefon do pracy []:
    Telefon domowy []:
    Inne []:
Czy informacja jest poprawna? [T/n] T
root@SRV1:~# su user
user@SRV1:/root$ su jgula
Hasło:
jgula@SRV1:/root$ |

```

```
jgula@SRV1: /root X jgula@SRV2: /root X + v
root@SRV2:~# adduser jgula
Dodawanie użytkownika "jgula"...
Dodawanie nowej grupy "jgula" (1001)...
Dodawanie nowego użytkownika "jgula" (1001) w grupie "jgula"...
Tworzenie katalogu domowego "/home/jgula"...
Kopiowanie plików z "/etc/skel" ...
Nowe hasło:
Proszę ponownie wpisać nowe hasło:
passwd: hasło zostało zmienione
Zmieniam informację o użytkowniku jgula
Wpisz nową wartość lub wciśnij ENTER by przyjąć wartość domyślną
    Imię i nazwisko []: Jacek Gula
    Numer pokoju []:
    Telefon do pracy []:
    Telefon domowy []:
    Inne []:
Czy informacja jest poprawna? [T/n] T
root@SRV2:~# su user
user@SRV2:/root$ su jgula
Hasło:
jgula@SRV2:/root$ |
```

```
jgula@SRV1:/root$ passwd jgula
Zmienianie hasła dla jgula.
Obecne hasło:
Nowe hasło:
Proszę ponownie wpisać nowe hasło:
passwd: hasło zostało zmienione
jgula@SRV1:/root$ |
```

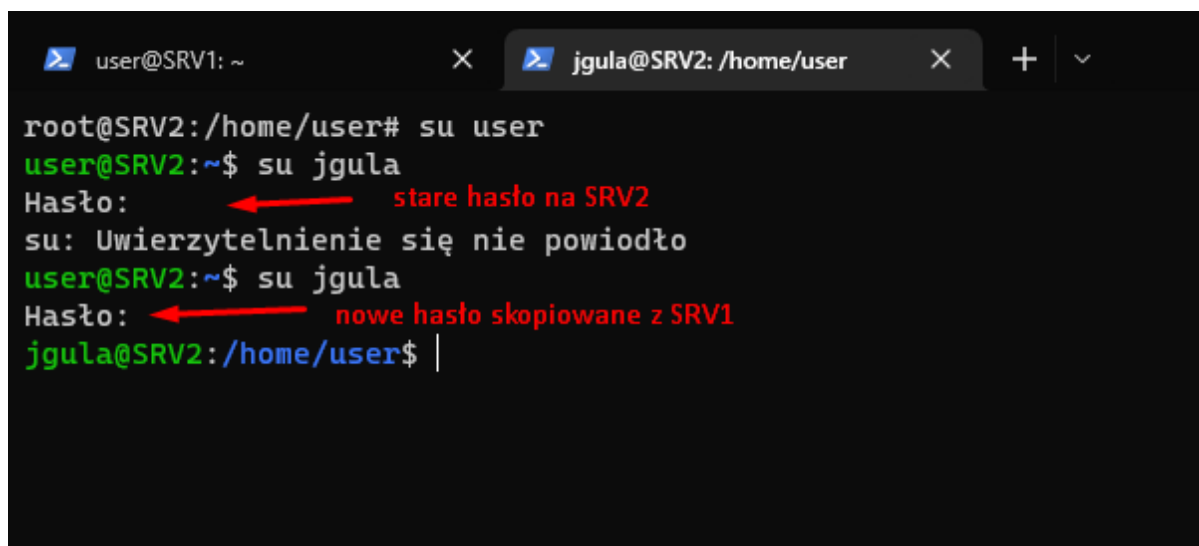
```
user@SRV1:/root$ su jgula
Hasło:
jgula@SRV1:/root$ |
```

```
GNU nano 5.4 /etc/shadow
root:$y$j9T$318VU7wsFM5DgKwkBwI.f1$mPPMT.IVvL4gC.p7AZDYNn5UfoS.DGyg4Lq/4ZoxWG1:18857:0:99999:7:::
daemon*:18857:0:99999:7:::
bin*:18857:0:99999:7:::
sys*:18857:0:99999:7:::
sync*:18857:0:99999:7:::
games*:18857:0:99999:7:::
man*:18857:0:99999:7:::
lp*:18857:0:99999:7:::
mail*:18857:0:99999:7:::
news*:18857:0:99999:7:::
uucp*:18857:0:99999:7:::
proxy*:18857:0:99999:7:::
www-data*:18857:0:99999:7:::
backup*:18857:0:99999:7:::
list*:18857:0:99999:7:::
irc*:18857:0:99999:7:::
gnats*:18857:0:99999:7:::
nobody*:18857:0:99999:7:::
_apt*:18857:0:99999:7:::
systemd-timesync*:18857:0:99999:7:::
systemd-network*:18857:0:99999:7:::
systemd-resolve*:18857:0:99999:7:::
messagebus*:18857:0:99999:7:::
sshd*:18857:0:99999:7:::
user:$y$j9T$xPpYg3Y1CdD9btREjAAmf.$1vF8Mo5oWtm5HbSpj75DQf.2DYGdyL6N7kFFOyQRb41:18857:0:99999:7:::
systemd-coredump:!:18857:0:99999:7:::
jgula:$y$j9T$2oyQL5CR/nVY0enjGmjhm/$XfqLPjEVjRu1iI8Khbk.e8/4aGYDSQky73UHPJRzQc4:19293:0:99999:7:::

^G Help      ^O Zapisz    ^W Wyszukaj  ^K Cut       ^T Execute   ^C Location  M-U Odwołaj
^X Wyjdź     ^R Wczyt.plik ^\ Zastąp   ^U Paste     ^J Wyjustuj  ^_ Do linii  M-E Odtwórz
```

```
GNU nano 5.4 /etc/shadow *
root:$y$j9T$318VU7wsFM5DgKwkBwI.f1$mPPMT.IVvL4gC.p7AZDYNn5UfoS.DGyg4Lq/4ZoxWG1:18857:0:99999:7:::
daemon*:18857:0:99999:7:::
bin*:18857:0:99999:7:::
sys*:18857:0:99999:7:::
sync*:18857:0:99999:7:::
games*:18857:0:99999:7:::
man*:18857:0:99999:7:::
lp*:18857:0:99999:7:::
mail*:18857:0:99999:7:::
news*:18857:0:99999:7:::
uucp*:18857:0:99999:7:::
proxy*:18857:0:99999:7:::
www-data*:18857:0:99999:7:::
backup*:18857:0:99999:7:::
list*:18857:0:99999:7:::
irc*:18857:0:99999:7:::
gnats*:18857:0:99999:7:::
nobody*:18857:0:99999:7:::
_apt*:18857:0:99999:7:::
systemd-timesync*:18857:0:99999:7:::
systemd-network*:18857:0:99999:7:::
systemd-resolve*:18857:0:99999:7:::
messagebus*:18857:0:99999:7:::
sshd*:18857:0:99999:7:::
user:$y$j9T$xPpYg3Y1CdD9btREjAAmf.$1vF8Mo5oWtm5HbSpj75DQf.2DYGdyL6N7kFFOyQRb41:18857:0:99999:7:::
systemd-coredump:!:18857:0:99999:7:::
jgula:$y$j9T$2oyQL5CR/nVY0enjGmjhm/$XfqLPjEVjRu1iI8Khbk.e8/4aGYDSQky73UHPJRzQc4:19293:0:99999:7:::

^G Help      ^O Zapisz    ^W Wyszukaj  ^K Cut       ^T Execute   ^C Location  M-U Odwołaj
^X Wyjdź     ^R Wczyt.plik ^\ Zastąp   ^U Paste     ^J Wyjustuj  ^_ Do linii  M-E Odtwórz
```



```
user@SRV1: ~
jgula@SRV2: /home/user

root@SRV2:/home/user# su user
user@SRV2:~$ su jgula
Hasło: ← stare hasło na SRV2
su: Uwierzytelnienie się nie powiodło
user@SRV2:~$ su jgula
Hasło: ← nowe hasło skopiowane z SRV1
jgula@SRV2:/home/user$ |
```

Zadanie 5

Jesteś administratorem systemów w firmie Northwind Traders (nwtraders.msft).

Otrzymałeś prośbę o zaszyfrowanie w serwerze Linux SRV1 z wykorzystaniem oprogramowania GPG oraz klucza symetrycznego AES256, pliku "klucze.txt" zawierającego istotne poufne informacje, tak aby zabezpieczyć go przed dostępem osób niepowołanych.

Twoim zadaniem jest więc:

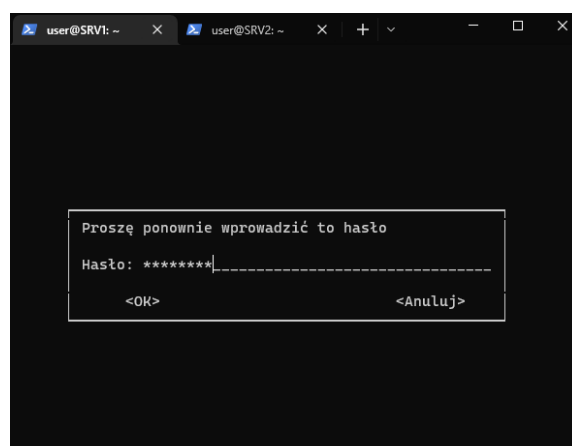
- zalogować się w serwerze Linux SRV1 na użytkownika "user", i następnie sprawdzić listę dostępnych mechanizmów szyfrowania: `$ gpg --version` (Uwaga: "\$" na początku powyższej linii oznacza założenie, że jesteśmy zalogowani jako zwykły użytkownik "user")
- utworzenie w serwerze Linux pliku: `$ cd /home/user$ echo "Moj plik" > klucze.txt`
- zaszyfrowanie w serwerze Linux pliku "klucze.txt" z wykorzystaniem GPG na bazie klucza symetrycznego AES256 tworzonego z wykorzystaniem hasła: `$ cd /home/user$ gpg --cipher-algo AES256 -c klucze.txt`
- dokonać próby w serwerze Linux wyedytowania pliku "klucze.txt" oraz "klucze.txt.gpg"
- bezpiecznie pozbyć się w serwerze Linux oryginału pliku "klucze.txt": `$ cd /home/user$ shred -n 35 -z -u klucze.txt`
- odszyfrować w serwerze Linux plik "klucze.txt" z wykorzystaniem GPG, a następnie usunąć sam zaszyfrowany plik: `$ cd /home/user$ gpg ~/klucze.txt.gpg$ rm ~/klucze.txt.gpg`

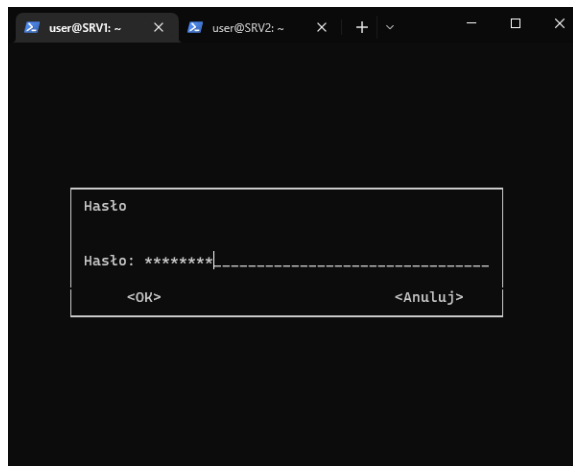

```
user@SRV1: ~ user@SRV2: ~
user@SRV1:~$ gpg --version
gpg (GnuPG) 2.2.27
libgcrypt 1.8.8
Copyright (C) 2021 Free Software Foundation, Inc.
License GNU GPL-3.0-or-later <https://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Home: /home/user/.gnupg
Obsługiwane algorytmy:
Asymetryczne: RSA, ELG, DSA, ECDH, ECDSA, EDDSA
Symetryczne: IDEA, 3DES, CAST5, BLOWFISH, AES, AES192, AES256,
              TWOFISH, CAMELLIA128, CAMELLIA192, CAMELLIA256
Skrótów: SHA1, RIPEMD160, SHA256, SHA384, SHA512, SHA224
Kompresji: Nieskompresowany, ZIP, ZLIB, BZIP2
user@SRV1:~$ |
```

```
user@SRV1:~$ pwd
/home/user
user@SRV1:~$ echo "Moj plik" > klucze.txt
user@SRV1:~$ ls
klucze.txt plik.txt
```

```
user@SRV1: ~ user@SRV2: ~
user@SRV1:~$ gpg --cipher-algo AES256 -c klucze.txt
user@SRV1:~$ ls
klucze.txt klucze.txt.gpg plik.txt
user@SRV1:~$ |
```





```
user@SRV1:~$ echo "XX" > klucze.txt
user@SRV1:~$ more klucze.txt
XX
user@SRV1:~$ nano klucze.txt.gpg
user@SRV1:~$ shred -n 35 -zu klucze.txt
user@SRV1:~$ ls
klucze.txt.gpg  plik.txt
```

```
user@SRV1: ~ user@SRV2: ~ + - □ ×
GNU nano 5.4 klucze.txt.gpg
^D ^C^B^N^i^H^A0\|^m^YzZ=0VR^K^W^)J1Ku<>
xxx

[ Zapisano 3 linie ]
^G Help ^O Zapisz ^W Wyszukaj ^K Cut ^T Execute
^X Wyjdź ^R Wczyt.pli ^\ Zastąp ^U Paste ^J Wyjustuj
```

```
user@SRV1:~$ gpg -d klucze.txt.gpg
gpg: dane zaszyfrowano za pomocą AES256.CFB
gpg: zaszyfrowane jednym hasłem
Moj plik
gpg: [don't know]: invalid packet (ctb=0d)
gpg: [don't know]: invalid packet (ctb=78)
user@SRV1:~$ ls
klucze.txt klucze.txt.gpg plik.txt
user@SRV1:~$ |
```



```
GNU nano 5.4 klucze.txt.gpg
^D      ^C^B^N^XASX^i^H^A^0^\\^|^m^YzZ=0VR^K^W^J^1Ku<)){{^B^N^>+^B^R^j>

[ Zapisano 2 linie ]
^G Help      ^O Zapisz     ^W Wyszukaj   ^K Cut        ^T Execute    ^C Location
^X Wyjdź     ^R Wczyt.plik ^\ Zastap     ^U Paste      ^J Wyjustuj   ^_ Do linii
```

```
user@SRV1:~$ nano klucze.txt.gpg
user@SRV1:~$ gpg -d klucze.txt.gpg
gpg: dane zaszyfrowano za pomocą AES256.CFB
gpg: [don't know]: indeterminate length for invalid packet type 12
```

```
user@SRV1:~$ rm klucze.txt.gpg
user@SRV1:~$ ls
plik.txt
```

Zadanie 6

Otrzymałeś prośbę o przetestowanie w serwerze Linux SRV1 wydajności wskazanych implementacji algorytmów haszujących, klucza symetrycznego, oraz klucza asymetrycznego.

Twoim zadaniem jest więc:

- zalogować się w serwerze Linux SRV1, a następnie wydać polecenie testujące wydajność wybranych implementacji algorytmów haszujących (a na końcu przeanalizować wyniki):`# openssl speed md5 sha1 sha256 sha512`
- zalogować się w serwerze Linux SRV1, a następnie wydać polecenie testujące wydajność wybranych implementacji klucza symetrycznego (a na końcu przeanalizować wyniki):`# openssl speed aes-128-cbc aes-192-cbc aes-256-cbc`

- zalogować się w serwerze Linux SRV1, a następnie wydać polecenie testujące wydajność wybranych implementacji klucza asymetrycznego (a na końcu przeanalizować wyniki):
openssl speed rsa ecdsa

```
user@SRV1:~$ openssl speed md5 sha1 sha256 sha512
Doing md5 for 3s on 16 size blocks: 7970896 md5's in 2.99s
Doing md5 for 3s on 64 size blocks: 6337885 md5's in 2.99s
Doing md5 for 3s on 256 size blocks: 3761369 md5's in 3.00s
Doing md5 for 3s on 1024 size blocks: 1442977 md5's in 2.99s
Doing md5 for 3s on 8192 size blocks: 213133 md5's in 3.00s
Doing md5 for 3s on 16384 size blocks: 107125 md5's in 2.99s
Doing sha1 for 3s on 16 size blocks: 7451375 sha1's in 2.99s
Doing sha1 for 3s on 64 size blocks: 5893345 sha1's in 2.99s
Doing sha1 for 3s on 256 size blocks: 4062074 sha1's in 2.99s
Doing sha1 for 3s on 1024 size blocks: 1767995 sha1's in 2.99s
Doing sha1 for 3s on 8192 size blocks: 281498 sha1's in 3.00s
Doing sha1 for 3s on 16384 size blocks: 143063 sha1's in 2.99s
Doing sha256 for 3s on 16 size blocks: 5589439 sha256's in 3.00s
Doing sha256 for 3s on 64 size blocks: 4007734 sha256's in 2.99s
Doing sha256 for 3s on 256 size blocks: 2286811 sha256's in 2.99s
Doing sha256 for 3s on 1024 size blocks: 828338 sha256's in 3.00s
Doing sha256 for 3s on 8192 size blocks: 120241 sha256's in 2.99s
Doing sha256 for 3s on 16384 size blocks: 60964 sha256's in 3.00s
Doing sha512 for 3s on 16 size blocks: 4802733 sha512's in 2.99s
Doing sha512 for 3s on 64 size blocks: 4819860 sha512's in 3.00s
Doing sha512 for 3s on 256 size blocks: 2599071 sha512's in 2.99s
Doing sha512 for 3s on 1024 size blocks: 1086235 sha512's in 2.99s
Doing sha512 for 3s on 8192 size blocks: 168634 sha512's in 3.00s
Doing sha512 for 3s on 16384 size blocks: 87131 sha512's in 2.99s
OpenSSL 1.1.1n 15 Mar 2022
built on: Fri Jun 24 20:22:19 2022 UTC
options:bn(64,64) rc4(16x,int) des(int) aes(partial) blowfish(ptr)
compiler: gcc -fPIC -pthread -m64 -Wa,--noexecstack -Wall -Wa,--noexecstack -g -O2 -ffile-prefix-map=/build/openssl-q
Y/ec/openssl-1.1.1n=. -fstack-protector-strong -Wformat -Werror=format-security -DOPENSSL_USE_NODELETE -DL_ENDIAN -DOP
ENSSL_PIC -DOPENSSL_CPUID_OBJ -DOPENSSL_IA32_SSE2 -DOPENSSL_BN_ASM_MONT -DOPENSSL_BN_ASM_MONT5 -DOPENSSL_BN_ASM_GF2m -
DSHA1_ASM -DSHA256_ASM -DSHA512_ASM -DKECCAK1600_ASM -DRC4_ASM -DMD5_ASM -DAESNI_ASM -DVPAES_ASM -DGHASH_ASM -DECP_NIS
TZ256_ASM -DX25519_ASM -DPOLY1305_ASM -DNDEBUG -Wdate-time -D_FORTIFY_SOURCE=2
The 'numbers' are in 1000s of bytes per second processed.
type      16 bytes      64 bytes      256 bytes      1024 bytes      8192 bytes      16384 bytes
md5        42653.62k      135660.41k      320970.15k      494183.43k      581995.18k      587002.01k
sha1       39873.58k      126145.18k      347789.61k      605493.94k      768677.21k      783927.82k
sha256     29810.34k      85784.27k      195793.85k      282739.37k      329436.21k      332944.73k
sha512     25700.24k      102823.68k      222529.16k      372008.24k      460483.24k      477442.91k
```

```
user@SRV1:~$ openssl speed aes-128-cbc aes-192-cbc aes-256-cbc
Doing aes-128 cbc for 3s on 16 size blocks: 18542606 aes-128 cbc's in 3.00s
Doing aes-128 cbc for 3s on 64 size blocks: 5967130 aes-128 cbc's in 2.99s
Doing aes-128 cbc for 3s on 256 size blocks: 1593069 aes-128 cbc's in 3.00s
Doing aes-128 cbc for 3s on 1024 size blocks: 405334 aes-128 cbc's in 2.99s
Doing aes-128 cbc for 3s on 8192 size blocks: 50879 aes-128 cbc's in 3.00s
Doing aes-128 cbc for 3s on 16384 size blocks: 25464 aes-128 cbc's in 2.99s
Doing aes-192 cbc for 3s on 16 size blocks: 16525944 aes-192 cbc's in 3.00s
Doing aes-192 cbc for 3s on 64 size blocks: 5158808 aes-192 cbc's in 2.99s
Doing aes-192 cbc for 3s on 256 size blocks: 1332078 aes-192 cbc's in 3.00s
Doing aes-192 cbc for 3s on 1024 size blocks: 345135 aes-192 cbc's in 2.99s
Doing aes-192 cbc for 3s on 8192 size blocks: 43399 aes-192 cbc's in 3.00s
Doing aes-192 cbc for 3s on 16384 size blocks: 21604 aes-192 cbc's in 2.99s
Doing aes-256 cbc for 3s on 16 size blocks: 14918482 aes-256 cbc's in 3.00s
Doing aes-256 cbc for 3s on 64 size blocks: 4544470 aes-256 cbc's in 2.99s
Doing aes-256 cbc for 3s on 256 size blocks: 1183176 aes-256 cbc's in 2.99s
Doing aes-256 cbc for 3s on 1024 size blocks: 285345 aes-256 cbc's in 3.00s
Doing aes-256 cbc for 3s on 8192 size blocks: 36317 aes-256 cbc's in 2.99s
Doing aes-256 cbc for 3s on 16384 size blocks: 18679 aes-256 cbc's in 3.00s
OpenSSL 1.1.1n 15 Mar 2022
built on: Fri Jun 24 20:22:19 2022 UTC
options:bn(64,64) rc4(16x,int) des(int) aes(partial) blowfish(ptr)
compiler: gcc -fPIC -pthread -m64 -Wa,--noexecstack -Wall -Wa,--noexecstack -g -O2 -ffile-prefix-map=/build/openssl-q
Y/ec/openssl-1.1.1n=. -fstack-protector-strong -Wformat -Werror=format-security -DOPENSSL_USE_NODELETE -DL_ENDIAN -DOP
ENSSL_PIC -DOPENSSL_CPUID_OBJ -DOPENSSL_IA32_SSE2 -DOPENSSL_BN_ASM_MONT -DOPENSSL_BN_ASM_MONT5 -DOPENSSL_BN_ASM_GF2m -
DSHA1_ASM -DSHA256_ASM -DSHA512_ASM -DKECCAK1600_ASM -DRC4_ASM -DMD5_ASM -DAESNI_ASM -DVPAES_ASM -DGHASH_ASM -DECP_NIS
TZ256_ASM -DX25519_ASM -DPOLY1305_ASM -DNDEBUG -Wdate-time -D_FORTIFY_SOURCE=2
The 'numbers' are in 1000s of bytes per second processed.
type      16 bytes      64 bytes      256 bytes      1024 bytes      8192 bytes      16384 bytes
aes-128 cbc 98893.90k      127724.52k      135941.89k      138816.73k      138933.59k      139532.50k
aes-192 cbc 88138.37k      110422.65k      113670.66k      118200.08k      118508.20k      118381.25k
aes-256 cbc 79565.24k      97272.94k      101302.03k      97397.76k      99501.29k      102012.25k
```

```
user@SRV1:~$ openssl speed rsa ecdsa
Doing 512 bits private rsa's for 10s: 54278 512 bits private RSA's in 9.97s
Doing 512 bits public rsa's for 10s: 750904 512 bits public RSA's in 9.99s
Doing 1024 bits private rsa's for 10s: 37028 1024 bits private RSA's in 9.97s
Doing 1024 bits public rsa's for 10s: 394855 1024 bits public RSA's in 9.99s
Doing 2048 bits private rsa's for 10s: 10758 2048 bits private RSA's in 9.98s
Doing 2048 bits public rsa's for 10s: 140158 2048 bits public RSA's in 9.98s
Doing 3072 bits private rsa's for 10s: 1556 3072 bits private RSA's in 9.99s
Doing 3072 bits public rsa's for 10s: 70256 3072 bits public RSA's in 9.98s
Doing 4096 bits private rsa's for 10s: 684 4096 bits private RSA's in 10.00s
Doing 4096 bits public rsa's for 10s: 41984 4096 bits public RSA's in 9.98s
Doing 7680 bits private rsa's for 10s: 76 7680 bits private RSA's in 10.01s
Doing 7680 bits public rsa's for 10s: 12409 7680 bits public RSA's in 9.98s
Doing 15360 bits private rsa's for 10s: 14 15360 bits private RSA's in 10.50s
Doing 15360 bits public rsa's for 10s: 3207 15360 bits public RSA's in 9.98s
Doing 160 bits sign ecDSA's for 10s: 8346 160 bits ECDSA signs in 9.95s
Doing 160 bits verify ecDSA's for 10s: 8198 160 bits ECDSA verify in 9.96s
Doing 192 bits sign ecDSA's for 10s: 6731 192 bits ECDSA signs in 9.98s
Doing 192 bits verify ecDSA's for 10s: 6993 192 bits ECDSA verify in 9.97s
Doing 224 bits sign ecDSA's for 10s: 45908 224 bits ECDSA signs in 9.96s
Doing 224 bits verify ecDSA's for 10s: 24373 224 bits ECDSA verify in 9.97s
Doing 256 bits sign ecDSA's for 10s: 137918 256 bits ECDSA signs in 9.91s
Doing 256 bits verify ecDSA's for 10s: 52863 256 bits ECDSA verify in 9.98s
Doing 384 bits sign ecDSA's for 10s: 1963 384 bits ECDSA signs in 9.96s
Doing 384 bits verify ecDSA's for 10s: 2391 384 bits ECDSA verify in 9.98s
Doing 521 bits sign ecDSA's for 10s: 8593 521 bits ECDSA signs in 9.99s
Doing 521 bits verify ecDSA's for 10s: 5157 521 bits ECDSA verify in 9.98s
Doing 163 bits sign ecDSA's for 10s: 8739 163 bits ECDSA signs in 9.95s
Doing 163 bits verify ecDSA's for 10s: 4448 163 bits ECDSA verify in 9.96s
Doing 233 bits sign ecDSA's for 10s: 6421 233 bits ECDSA signs in 9.97s
Doing 233 bits verify ecDSA's for 10s: 3295 233 bits ECDSA verify in 9.98s
Doing 283 bits sign ecDSA's for 10s: 3944 283 bits ECDSA signs in 9.97s
Doing 283 bits verify ecDSA's for 10s: 2027 283 bits ECDSA verify in 9.97s
Doing 409 bits sign ecDSA's for 10s: 2344 409 bits ECDSA signs in 9.98s
Doing 409 bits verify ecDSA's for 10s: 1195 409 bits ECDSA verify in 9.98s
Doing 571 bits sign ecDSA's for 10s: 1089 571 bits ECDSA signs in 9.97s
Doing 571 bits verify ecDSA's for 10s: 574 571 bits ECDSA verify in 9.97s
Doing 163 bits sign ecDSA's for 10s: 8409 163 bits ECDSA signs in 9.96s
Doing 163 bits verify ecDSA's for 10s: 4261 163 bits ECDSA verify in 9.96s
Doing 233 bits sign ecDSA's for 10s: 6401 233 bits ECDSA signs in 9.95s
Doing 233 bits verify ecDSA's for 10s: 3381 233 bits ECDSA verify in 9.97s
Doing 283 bits sign ecDSA's for 10s: 3998 283 bits ECDSA signs in 9.98s
Doing 283 bits verify ecDSA's for 10s: 2024 283 bits ECDSA verify in 9.98s
Doing 409 bits sign ecDSA's for 10s: 2337 409 bits ECDSA signs in 9.99s
Doing 409 bits verify ecDSA's for 10s: 1195 409 bits ECDSA verify in 9.99s
Doing 571 bits sign ecDSA's for 10s: 1112 571 bits ECDSA signs in 9.97s
Doing 571 bits verify ecDSA's for 10s: 568 571 bits ECDSA verify in 9.99s
Doing 256 bits sign ecDSA's for 10s: 4731 256 bits ECDSA signs in 9.97s
Doing 256 bits verify ecDSA's for 10s: 4817 256 bits ECDSA verify in 9.98s
Doing 256 bits sign ecDSA's for 10s: 4705 256 bits ECDSA signs in 9.96s
Doing 256 bits verify ecDSA's for 10s: 4836 256 bits ECDSA verify in 9.99s
```

```

Doing 384 bits sign ecDSA's for 10s: 2137 384 bits ECDSA signs in 9.99s
Doing 384 bits verify ecDSA's for 10s: 2413 384 bits ECDSA verify in 9.99s
Doing 384 bits sign ecDSA's for 10s: 2183 384 bits ECDSA signs in 9.99s
Doing 384 bits verify ecDSA's for 10s: 2516 384 bits ECDSA verify in 9.98s
Doing 512 bits sign ecDSA's for 10s: 1924 512 bits ECDSA signs in 9.98s
Doing 512 bits verify ecDSA's for 10s: 2083 512 bits ECDSA verify in 9.98s
Doing 512 bits sign ecDSA's for 10s: 1975 512 bits ECDSA signs in 9.98s
Doing 512 bits verify ecDSA's for 10s: 2237 512 bits ECDSA verify in 9.98s
OpenSSL 1.1.1n 15 Mar 2022
built on: Fri Jun 24 20:22:19 2022 UTC
options:bn(64,64) rc4(16x,int) des(int) aes(partial) blowfish(ptr)
compiler: gcc -fPIC -pthread -m64 -Wa,--noexecstack -Wall -Wa,--noexecstack -g -O2 -ffile-prefix-map=/build/openssl-qQ
Yec/openssl-1.1.1n=. -fstack-protector-strong -Wformat -Werror=format-security -DOPENSSL_USE_NODELETE -DL_ENDIAN -DOP
ENSSL_PIC -DOPENSSL_CPUID_OBJ -DOPENSSL_IA32_SSE2 -DOPENSSL_BN_ASM_MONT -DOPENSSL_BN_ASM_MONT5 -DOPENSSL_BN_ASM_GF2m -
DSHA1_ASM -DSHA256_ASM -DSHA512_ASM -DKECCAK1600_ASM -DRC4_ASM -DMD5_ASM -DAESNI_ASM -DVPAES_ASM -DGHASH_ASM -DECP_NIS
T256_ASM -DX25519_ASM -DPOLY1305_ASM -DNDEBUG -Wdate-time -D_FORTIFY_SOURCE=2

sign verify sign/s verify/s
rsa 512 bits 0.000184s 0.000013s 5444.1 75165.6
rsa 1024 bits 0.000269s 0.000025s 3713.9 39525.0
rsa 2048 bits 0.000928s 0.000071s 1078.0 14043.9
rsa 3072 bits 0.006420s 0.000142s 155.8 7039.7
rsa 4096 bits 0.014620s 0.000238s 68.4 4206.8
rsa 7680 bits 0.131711s 0.000804s 7.6 1243.4
rsa 15360 bits 0.750000s 0.003112s 1.3 321.3

sign verify sign/s verify/s
160 bits ecDSA (secp160r1) 0.0012s 0.0012s 838.8 823.1
192 bits ecDSA (nistp192) 0.0015s 0.0014s 674.4 701.4
224 bits ecDSA (nistp224) 0.0002s 0.0004s 4609.2 2444.6
256 bits ecDSA (nistp256) 0.0001s 0.0002s 13917.1 5296.9
384 bits ecDSA (nistp384) 0.0051s 0.0042s 197.1 239.6
521 bits ecDSA (nistp521) 0.0012s 0.0019s 860.2 516.7
163 bits ecDSA (nistk163) 0.0011s 0.0022s 878.3 446.6
233 bits ecDSA (nistk233) 0.0016s 0.0030s 644.0 330.2
283 bits ecDSA (nistk283) 0.0025s 0.0049s 395.6 203.3
409 bits ecDSA (nistk409) 0.0043s 0.0084s 234.9 119.7
571 bits ecDSA (nistk571) 0.0092s 0.0174s 109.2 57.6
163 bits ecDSA (nistb163) 0.0012s 0.0023s 844.3 427.8
233 bits ecDSA (nistb233) 0.0016s 0.0029s 643.3 339.1
283 bits ecDSA (nistb283) 0.0025s 0.0049s 400.6 202.8
409 bits ecDSA (nistb409) 0.0043s 0.0084s 233.9 119.6
571 bits ecDSA (nistb571) 0.0090s 0.0176s 111.5 56.9
256 bits ecDSA (brainpoolP256r1) 0.0021s 0.0021s 474.5 482.7
256 bits ecDSA (brainpoolP256t1) 0.0021s 0.0021s 472.4 484.1
384 bits ecDSA (brainpoolP384r1) 0.0047s 0.0041s 213.9 241.5
384 bits ecDSA (brainpoolP384t1) 0.0046s 0.0040s 218.5 252.1
512 bits ecDSA (brainpoolP512r1) 0.0052s 0.0048s 192.8 208.7
512 bits ecDSA (brainpoolP512t1) 0.0051s 0.0045s 197.9 224.1

```

Zadanie 7

Jesteś administratorem systemów w firmie Northwind Traders (nwtraders.msft).

Otrzymałeś prośbę, aby w serwerze Linux SRV1 oraz w serwerze Linux SRV2 wygenerować klucz asymetryczny na potrzeby szyfrowania plików oraz podpisu cyfrowego z wykorzystaniem oprogramowania PGP.

Twoim zadaniem jest więc:

- zalogować się w serwerze Linux SRV1 na użytkownika "user", i następnie wygenerować klucz asymetryczny na potrzeby szyfrowania plików oraz podpisu cyfrowego z wykorzystaniem oprogramowania PGP:

```
$ gpg --gen-key
```

podając dane:

- Proszę wybrać rodzaj klucza: (1) RSA i RSA (domyślne)
- Jakiej długości klucz wygenerować? (2048): 2048

- Okres ważności klucza ? (0): 0
- Imię i nazwisko: *Jacek Gula*
- Adres poczty elektronicznej: *kgula@nwtraders.msft*
- Komentarz: *{brak}*
- Zmienić (I)mię/nazwisko, (K)omentarz, adres (E)mail, przejść (D)alej, czy (W)yjść z programu?: D
- Musisz podać długie, skomplikowane hasło aby ochronić swój klucz tajny:
Zaq12wsx

Uwaga: aby zwiększyć liczbę entropii (gdy system wyrzuci informację o jej zbyt małej ilości) należy zainstalować oprogramowanie *rng-tools*: # apt update && apt install rng-tools -ywyedytować plik konfiguracyjny

/etc/default/rng-tools (lub /etc/default/rng-tools-debian, w zależności od wersji dystrybucji Debian), dodając

linię:HRNGDEVICE=/dev/urandomnastępnie uruchomić aplikację rng-tools:# systemctl restart rng-toolsa następnie obserwować ilość entropii na konsoli numer dwa, gdy będzie

wystarczająca, można wygenerować klucz na konsoli numer jeden

- zalogować się w serwerze Linux SRV2 na użytkownika "user", i następnie wygenerować klucz asymetryczny na potrzeby szyfrowania plików oraz podpisu cyfrowego z wykorzystaniem oprogramowania PGP:

\$ gpg --gen-key

podając dane:

- Proszę wybrać rodzaj klucza: (1) RSA i RSA (domyślne)
- Jakiej długości klucz wygenerować? (2048): 2048
- Okres ważności klucza ? (0): 0
- Imię i nazwisko: *Witold Sup*
- Adres poczty elektronicznej: *wsup@nwtraders.msft*
- Komentarz: *{brak}*
- Zmienić (I)mię/nazwisko, (K)omentarz, adres (E)mail, przejść (D)alej, czy (W)yjść z programu?: D
- Musisz podać długie, skomplikowane hasło aby ochronić swój klucz tajny:
Zaq12wsx

- zweryfikować w systemie Linux SRV1 oraz SRV2 listę posiadanych kluczy publicznych oraz prywatnych, wydając polecenie:

```
$ gpg --list-keys
```

```
$ gpg --list-secret-keys
```

```
user@SRV1:~$ gpg --gen-key
gpg (GnuPG) 2.2.27; Copyright (C) 2021 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Uwaga: pełną funkcjonalność generowania klucza można uzyskać przez „gpg --full-generate-key”.

GnuPG musi utworzyć identyfikator użytkownika do identyfikacji klucza.

Imię i nazwisko: Jacek Gula
Adres poczty elektronicznej: jgula@nwtraders.msft
Twój identyfikator użytkownika będzie wyglądał tak:
"Jacek Gula <jgula@nwtraders.msft>"

Zmienić (I)mię/nazwisko, adres (E)mail, przejść (D)alej,
czy (W)yjść z programu? D
Musimy wygenerować dużo losowych bajtów. Dobrym pomysłem aby pomóc komputerowi
podczas generowania liczb pierwszych jest wykonywanie w tym czasie innych
działań (pisanie na klawiaturze, poruszanie myszką, odwołanie się do dysków);
dzięki temu generator liczb losowych ma możliwość zebrania odpowiedniej ilości
entropii.
Musimy wygenerować dużo losowych bajtów. Dobrym pomysłem aby pomóc komputerowi
podczas generowania liczb pierwszych jest wykonywanie w tym czasie innych
działań (pisanie na klawiaturze, poruszanie myszką, odwołanie się do dysków);
dzięki temu generator liczb losowych ma możliwość zebrania odpowiedniej ilości
entropii.
gpg: /home/user/.gnupg/trustdb.gpg: baza zaufania utworzona
gpg: klucz A15BA4593040DE29 został oznaczony jako obdarzony absolutnym zaufaniem.
gpg: katalog „/home/user/.gnupg/openpgp-revocs.d” utworzony
gpg: certyfikat unieważnienia został zapisany jako „/home/user/.gnupg/openpgp-revocs.d/AD87EC5CAB29CA7D7927765AA15BA4
593040DE29.rev”
klucz publiczny i prywatny (tajny) zostały utworzone i podpisane.

pub  rsa3072 2022-10-28 [SC] [wygasa: 2024-10-27]
     AD87EC5CAB29CA7D7927765AA15BA4593040DE29
uid                               Jacek Gula <jgula@nwtraders.msft>
sub  rsa3072 2022-10-28 [E] [wygasa: 2024-10-27]
```

Proszę wprowadzić hasło do
zabezpieczenia swojego nowego klucza

Hasło: *****_____

<OK>

<Anuluj>

```

user@SRV2:~$ gpg --gen-key
gpg (GnuPG) 2.2.27; Copyright (C) 2021 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

gpg: katalog „/home/user/.gnupg” utworzony
gpg: keybox „/home/user/.gnupg/pubring.kbx” utworzony
Uwaga: pełną funkcjonalność generowania klucza można uzyskać przez „gpg --full-generate-key”.

GnuPG musi utworzyć identyfikator użytkownika do identyfikacji klucza.

Imię i nazwisko: Witold Sup
Adres poczty elektronicznej: wsup@nwtraders.msft
Twój identyfikator użytkownika będzie wyglądał tak:
    "Witold Sup <wsup@nwtraders.msft>"

Zmienić (I)mię/nazwisko, adres (E)mail, przejść (D)alej,
czy (W)yjść z programu? D
Musimy wygenerować dużo losowych bajtów. Dobrym pomysłem aby pomóc komputerowi
podczas generowania liczb pierwszych jest wykonywanie w tym czasie innych
działań (pisanie na klawiaturze, poruszanie myszką, odwołanie się do dysków);
dzięki temu generator liczb losowych ma możliwość zebrania odpowiedniej ilości
entropii.
Musimy wygenerować dużo losowych bajtów. Dobrym pomysłem aby pomóc komputerowi
podczas generowania liczb pierwszych jest wykonywanie w tym czasie innych
działań (pisanie na klawiaturze, poruszanie myszką, odwołanie się do dysków);
dzięki temu generator liczb losowych ma możliwość zebrania odpowiedniej ilości
entropii.
gpg: /home/user/.gnupg/trustdb.gpg: baza zaufania utworzona
gpg: klucz 198FB8BE2EEF0F1D został oznaczony jako obdarzony absolutnym zaufaniem.
gpg: katalog „/home/user/.gnupg/openpgp-revocs.d” utworzony
gpg: certyfikat unieważnienia został zapisany jako „/home/user/.gnupg/openpgp-revocs.d/0E5AD31BE1866F12A4625B9D198FB8
BE2EEF0F1D.rev”
klucz publiczny i prywatny (tajny) zostały utworzone i podpisane.

pub   rsa3072 2022-10-28 [SC] [wygasa: 2024-10-27]
       0E5AD31BE1866F12A4625B9D198FB8BE2EEF0F1D
uid    [ absolutne ] Witold Sup <wsup@nwtraders.msft>
sub    rsa3072 2022-10-28 [E] [wygasa: 2024-10-27]

```

```

user@SRV2:~$ gpg --list-keys
gpg: sprawdzanie bazy zaufania
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: poziom: 0 poprawnych: 1 podpisanych: 0 zaufanie: 0-,0q,0n,0m,0f,1u
gpg: następne sprawdzanie bazy odbędzie się 2024-10-27
/home/user/.gnupg/pubring.kbx
-----
pub   rsa3072 2022-10-28 [SC] [wygasa: 2024-10-27]
       0E5AD31BE1866F12A4625B9D198FB8BE2EEF0F1D
uid    [ absolutne ] Witold Sup <wsup@nwtraders.msft>
sub    rsa3072 2022-10-28 [E] [wygasa: 2024-10-27]

user@SRV2:~$ gpg --list-secret-keys
/home/user/.gnupg/pubring.kbx
-----
sec   rsa3072 2022-10-28 [SC] [wygasa: 2024-10-27]
       0E5AD31BE1866F12A4625B9D198FB8BE2EEF0F1D
uid    [ absolutne ] Witold Sup <wsup@nwtraders.msft>
ssb    rsa3072 2022-10-28 [E] [wygasa: 2024-10-27]

```

```

user@SRV1:~$ gpg --list-keys
gpg: sprawdzanie bazy zaufania
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: poziom: 0 poprawnych: 1 podpisanych: 0 zaufanie: 0-,0q,0n,0m,0f,1u
gpg: następane sprawdzanie bazy odbędzie się 2024-10-27
/home/user/.gnupg/pubring.kbx
-----
pub   rsa3072 2022-10-28 [SC] [wygasa: 2024-10-27]
      AD87EC5CAB29CA7D7927765AA15BA4593040DE29
uid   [ absolutne ] Jacek Gula <jgula@nwtraders.msft>
sub   rsa3072 2022-10-28 [E] [wygasa: 2024-10-27]

user@SRV1:~$ gpg --list-secret-keys
/home/user/.gnupg/pubring.kbx
-----
sec   rsa3072 2022-10-28 [SC] [wygasa: 2024-10-27]
      AD87EC5CAB29CA7D7927765AA15BA4593040DE29
uid   [ absolutne ] Jacek Gula <jgula@nwtraders.msft>
ssb   rsa3072 2022-10-28 [E] [wygasa: 2024-10-27]

```

Zadanie 8

Jesteś administratorem systemów w firmie Northwind Traders (nwtraders.msft).

Otrzymałeś prośbę, aby w serwerze Linux SRV1 wyeksportować klucz publiczny wystawiony dla jgula@nwtraders.msft na potrzeby szyfrowania plików oraz podpisu cyfrowego z wykorzystaniem oprogramowania PGP, i zaimportować go w serwerze Linux SRV2 dla konta użytkownika "user", jak również w serwerze Linux SRV2 wyeksportować klucz publiczny wystawiony dla wsup@nwtraders.msft na potrzeby szyfrowania plików oraz podpisu cyfrowego z wykorzystaniem oprogramowania PGP, i zaimportować go w serwerze Linux SRV1 dla konta użytkownika "user".

Twoim zadaniem jest więc:

- zalogować się w serwerze Linux SRV1 na użytkownika "user", i następnie wyeksportować klucz publiczny wystawiony dla jgula@nwtraders.msft, i następnie przekopiować go do katalogu domowego użytkownika "user" w serwerze Linux SRV2:
`$ gpg --output jgula.gpg --export jgula@nwtraders.msft`
`$ scp jgula.gpg user@172.16.0.2:/home/user`
- zalogować się w serwerze Linux SRV2 na użytkownika "user", i następnie wyeksportować klucz publiczny wystawiony dla wsup@nwtraders.msft, i następnie przekopiować go do katalogu domowego użytkownika "user" w serwerze Linux SRV1:

```
$ gpg --output wsup.gpg --export wsup@nwtraders.msft
$ scp wsup.gpg user@172.16.0.1:/home/user
```

- zalogować się w serwerze Linux SRV1 na użytkownika "user", i następnie zaimportować do bazy kluczy publicznych PGP, klucz publiczny wystawiony dla *wsup@nwtraders.msft*:

```
$ gpg --import wsup.gpg
zweryfikować zaimportowany klucz (czy zawiera prawidłowy
hasz/skrót/odcisk palca):
```

```
$ gpg --edit-key wsup@nwtraders.msft
```

```
Polecenie> fpr
```

```
Polecenie> sign
```

(powyższe polecenie należy wydać aby podpisać tenże klucz publiczny, potwierdzając jego autentyczność, w przeciwnym razie program, przy każdej próbie użycia tego klucza, będzie ostrzegał, że klucz może nie być wiarygodny)

```
Polecenie> quit
```

- zalogować się w serwerze Linux SRV2 na użytkownika "user", i następnie zaimportować do bazy kluczy publicznych PGP, klucz publiczny wystawiony dla *ygula@nwtraders.msft*:

```
$ gpg --import ygula.gpg
```

zweryfikować zaimportowany klucz (czy zawiera prawidłowy hasz/skrót/odcisk palca):

```
$ gpg --edit-key ygula@nwtraders.msft
```

Polecenie> fpr
Polecenie> sign
Polecenie> quit

- zweryfikować w systemie Linux SRV1 oraz SRV2 listę posiadanych kluczy, wydając polecenie:

```
$ gpg --list-keys
```

```
user@SRV1:~$ gpg -o ygula.gpg --export ygula@nwtraders.msft
user@SRV1:~$ scp ygula.gpg user@192.168.68.112:/home/user
user@192.168.68.112's password:
ygula.gpg
100% 1753      2.1MB/s   00:00
```

```

user@SRV2:~$ gpg -o wsup.gpg --export wsup@nwtraders.msft
user@SRV2:~$ scp wsup.gpg user@192.168.68.111:/home/user
The authenticity of host '192.168.68.111 (192.168.68.111)' ca
n't be established.
ECDSA key fingerprint is SHA256:dNnoYes0G4q28uhF3UJnd6KUqZqZa
jN4QehUwvafsFw.
Are you sure you want to continue connecting (yes/no/[fingerp
rint])? yes
Warning: Permanently added '192.168.68.111' (ECDSA) to the li
st of known hosts.
user@192.168.68.111's password:
wsup.gpg                               100% 1752      1.9MB/s   00:00

```

```

user@SRV1:~$ gpg --import wsup.gpg
gpg: klucz 198FB8BE2EEF0F1D: klucz publiczny ,,Witold Sup <wsup@nwtraders.msft>' wczytano do zbioru
gpg: Ogółem przetworzonych kluczy: 1
gpg:      dołączono do zbioru: 1
user@SRV1:~$ gpg --edit-key wsup@nwtraders.msft
gpg (GnuPG) 2.2.27; Copyright (C) 2021 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

pub  rsa3072/198FB8BE2EEF0F1D
     utworzono: 2022-10-28  wygasa: 2024-10-27  użycie: SC
     zaufanie: nieznany    poprawność: nieznany
sub  rsa3072/20FADAD643C46721
     utworzono: 2022-10-28  wygasa: 2024-10-27  użycie: E
[      nieznane      ] (1). Witold Sup <wsup@nwtraders.msft>

gpg> fpr
pub  rsa3072/198FB8BE2EEF0F1D 2022-10-28 Witold Sup <wsup@nwtraders.msft>
     Odcisk klucza głównego: 0E5A D31B E186 6F12 A462  5B9D 198F B8BE 2EEF 0F1D

gpg> sign

pub  rsa3072/198FB8BE2EEF0F1D
     utworzono: 2022-10-28  wygasa: 2024-10-27  użycie: SC
     zaufanie: nieznany    poprawność: nieznany
     Odcisk klucza głównego: 0E5A D31B E186 6F12 A462  5B9D 198F B8BE 2EEF 0F1D

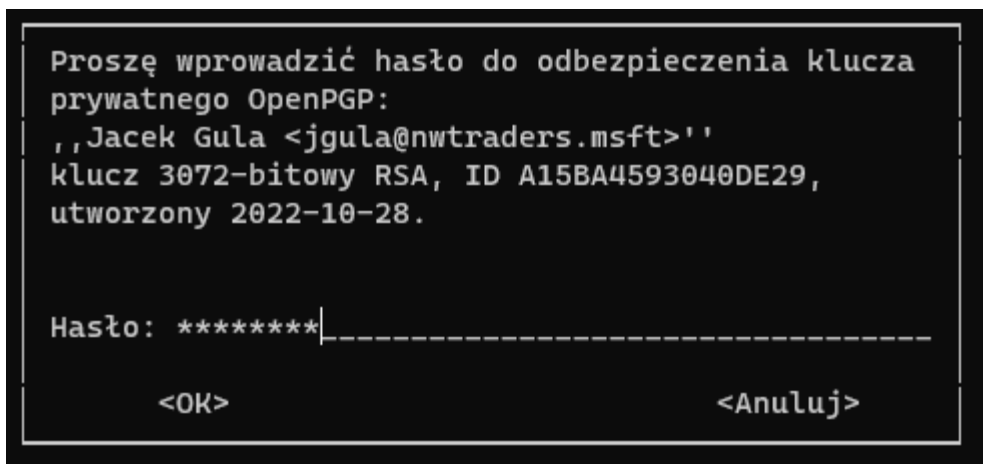
     Witold Sup <wsup@nwtraders.msft>

Ważność tego klucza wygasa 2024-10-27.
Czy jesteś naprawdę pewien, że chcesz podpisać ten klucz
swoim kluczem ,,Jacek Gula <jgula@nwtraders.msft>' (A15BA4593040DE29)

Czy na pewno podpisać? (t/N) t

gpg> quit
Zapisać zmiany? (t/N) t

```



```
user@SRV2:~$ gpg --import jgula.gpg
gpg: klucz A15BA4593040DE29: klucz publiczny ,,Jacek Gula <jgula@nwtraders.msft>' wczytano do zbioru
gpg: Ogółem przetworzonych kluczy: 1
gpg: dołączono do zbioru: 1
user@SRV2:~$ gpg --edit-key jgula@nwtraders.msft
gpg (GnuPG) 2.2.27; Copyright (C) 2021 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

pub  rsa3072/A15BA4593040DE29
   utworzono: 2022-10-28  wygasa: 2024-10-27  użycie: SC
   zaufanie: nieznany    poprawność: nieznany
sub  rsa3072/BE98A5587949EEDC
   utworzono: 2022-10-28  wygasa: 2024-10-27  użycie: E
   [ nieznane ] (1). Jacek Gula <jgula@nwtraders.msft>

gpg> fpr
pub  rsa3072/A15BA4593040DE29 2022-10-28 Jacek Gula <jgula@nwtraders.msft>
   Odcisk klucza głównego: AD87 EC5C AB29 CA7D 7927 765A A15B A459 3040 DE29

gpg> sign

pub  rsa3072/A15BA4593040DE29
   utworzono: 2022-10-28  wygasa: 2024-10-27  użycie: SC
   zaufanie: nieznany    poprawność: nieznany
   Odcisk klucza głównego: AD87 EC5C AB29 CA7D 7927 765A A15B A459 3040 DE29

       Jacek Gula <jgula@nwtraders.msft>

Ważność tego klucza wygasa 2024-10-27.
Czy jesteś naprawdę pewien, że chcesz podpisać ten klucz
swoim kluczem ,,Witold Sup <wsup@nwtraders.msft>' (198FB8BE2EEF0F1D)

Czy na pewno podpisać? (t/N) t

gpg> quit
Zapisać zmiany? (t/N) t
```

```

user@SRV1:~$ gpg --list-keys
gpg: sprawdzanie bazy zaufania
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: poziom: 0 poprawnych: 1 podpisanych: 1 zaufanie: 0-,0q,0n,0m,0f,1u
gpg: poziom: 1 poprawnych: 1 podpisanych: 0 zaufanie: 1-,0q,0n,0m,0f,0u
gpg: następane sprawdzanie bazy odbędzie się 2024-10-27
/home/user/.gnupg/pubring.kbx
-----
pub  rsa3072 2022-10-28 [SC] [wygasa: 2024-10-27]
    AD87EC5CAB29CA7D7927765AA15BA4593040DE29
uid  [ absolutne ] Jacek Gula <jgula@nwtraders.msft>
sub  rsa3072 2022-10-28 [E] [wygasa: 2024-10-27]

pub  rsa3072 2022-10-28 [SC] [wygasa: 2024-10-27]
    0E5AD31BE1866F12A4625B9D198FB8BE2EEF0F1D
uid  [ pełne ] Witold Sup <wsup@nwtraders.msft>
sub  rsa3072 2022-10-28 [E] [wygasa: 2024-10-27]

```

```

user@SRV2:~$ gpg --list-keys
gpg: sprawdzanie bazy zaufania
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: poziom: 0 poprawnych: 1 podpisanych: 1 zaufanie: 0-,0q,0n,0m,0f,1u
gpg: poziom: 1 poprawnych: 1 podpisanych: 0 zaufanie: 1-,0q,0n,0m,0f,0u
gpg: następane sprawdzanie bazy odbędzie się 2024-10-27
/home/user/.gnupg/pubring.kbx
-----
pub  rsa3072 2022-10-28 [SC] [wygasa: 2024-10-27]
    0E5AD31BE1866F12A4625B9D198FB8BE2EEF0F1D
uid  [ absolutne ] Witold Sup <wsup@nwtraders.msft>
sub  rsa3072 2022-10-28 [E] [wygasa: 2024-10-27]

pub  rsa3072 2022-10-28 [SC] [wygasa: 2024-10-27]
    AD87EC5CAB29CA7D7927765AA15BA4593040DE29
uid  [ pełne ] Jacek Gula <jgula@nwtraders.msft>
sub  rsa3072 2022-10-28 [E] [wygasa: 2024-10-27]

```

Zadanie 9

Jesteś administratorem systemów w firmie Northwind Traders (nwtraders.msft).

Otrzymałeś prośbę, aby przesłać w bezpieczny i szybki sposób plik "klucze.txt" z serwera Linux SRV1 do serwera Linux SRV2, co oznacza, że należy zaszyfrować w serwerze Linux SRV1 z wykorzystaniem oprogramowania PGP plik "klucze.txt", następnie przesłać go do serwera Linux SRV2, gdzie należy go odszyfrować.

Twoim zadaniem jest więc:

- zalogować się w serwerze Linux SRV1 na użytkownika "user", następnie zaszyfrować plik "klucze.txt" używając klucza publicznego *wsup@nwtraders.msft*, i przesłać zaszyfrowany plik na serwer Linux SRV2:

```
$ cd /home/user
```

```
$ gpg --output klucze.txt.gpg --encrypt --recipient wsup@nwtraders.msft  
klucze.txt
```

```
$ scp klucze.txt.gpg user@172.16.0.2:/home/user
```

- zalogować się w serwerze Linux SRV2 na użytkownika "user", następnie zweryfikować czy możliwe jest wyedytowanie pliku "*klucze.txt.gpg*"
- zalogować się w serwerze Linux SRV2 na użytkownika "user", następnie odszyfrować plik "*klucze.txt.gpg*":
\$ cd /home/user\$ gpg --output klucze.txt --decrypt klucze.txt.gpg
- zalogować się w serwerze Linux SRV2 na użytkownika "user", następnie zweryfikować czy możliwe jest wyedytowanie pliku "*klucze.txt*"

```
user@SRV1:~$ pwd  
/home/user  
user@SRV1:~$ gpg --output klucze.txt.gpg --encrypt --recipient wsup@nwtraders.msft klucze.txt  
user@SRV1:~$ scp klucze.txt.gpg user@192.168.68.112:/home/user  
user@192.168.68.112's password:  
klucze.txt.gpg 100% 479 719.1KB/s 00:00
```

```
user@SRV2:~$ ls  
jgula.gpg klucze.txt.gpg wsup.gpg  
user@SRV2:~$ nano klucze.txt.gpg
```



```
user@SRV2:~$ ls
jgula.gpg  klucze.txt  klucze.txt.gpg  wsup.gpg
user@SRV2:~$ more klucze.txt
Moj plik
```

Zadanie 10

Jesteś administratorem systemów w firmie Northwind Traders (nwtraders.msft).

Otrzymałeś prośbę, aby korzystając w serwerze Linux SRV1 z oprogramowania GPG u użytkownika "user" podpisać cyfrowo plik "wiadomosc.txt", przesłać go z serwera Linux SRV1 do serwera Linux SRV2 do katalogu domowego użytkownika "user" i tam zweryfikować podpis cyfrowy tegoż pliku.

Twoim zadaniem jest więc:

- zalogować się w serwerze Linux SRV1 na użytkownika "user", a następnie utworzyć plik "*wiadomosc.txt*":
\$ cd /home/user
\$ echo "Moj plik wiadomosc" > wiadomosc.txt
- zalogować się w serwerze Linux SRV1 na użytkownika "user", a następnie podpisać cyfrowo plik "*wiadomosc.txt*" i przesłać zaszyfrowany plik na serwer Linux SRV2:
\$ cd /home/user
\$ gpg --sign wiadomosc.txt
\$ ls\$ scp wiadomosc.txt.gpg user@172.16.0.2:/home/user
\$ rm wiadomosc.txt.gpg
- zalogować się w serwerze Linux SRV2 na użytkownika "user", następnie zweryfikować czy możliwe jest wyedytowanie pliku "*wiadomosc.txt.gpg*"
- zalogować się w serwerze Linux SRV2 na użytkownika "user", następnie zweryfikować podpis cyfrowy pliku "*wiadomosc.txt.gpg*", po czym go odszyfrować:
\$ cd /home/user
\$ gpg --verify wiadomosc.txt.gpg
\$ gpg --output wiadomosc.txt --decrypt wiadomosc.txt.gpg
- zalogować się w serwerze Linux SRV2 na użytkownika "user", następnie zweryfikować czy możliwe jest wyedytowanie pliku "*wiadomosc.txt*"
- zalogować się w serwerze Linux SRV2 na użytkownika "user", następnie usunąć plik "*wiadomosc.txt*":

```
$ rm /home/user/wiad*
```

```
user@SRV1:~$ pwd
/home/user
user@SRV1:~$ echo "Moj plik wiadomosc" > wiadomosc.txt
user@SRV1:~$ gpg --sign wiadomosc.txt
```

```
Proszę wprowadzić hasło do odbezpieczenia klucza prywatnego OpenPGP:
,,Jacek Gula <jgula@nwtraders.msft>''
klucz 3072-bitowy RSA, ID A15BA4593040DE29,
utworzony 2022-10-28.
```

```
Hasło: *****|-----
```

<OK>

<Anuluj>

```
user@SRV1:~$ ls
jgula.gpg wiadomosc.txt wiadomosc.txt.gpg wsup.gpg
user@SRV1:~$ scp wiadomosc.txt.gpg user@192.168.68.112:/home/user
user@192.168.68.112's password:
wiadomosc.txt.gpg                               100% 496   717.9KB/s   00:00
user@SRV1:~$ rm wiadomosc.txt.gpg
```



```

user@SRV2:~$ pwd
/home/user
user@SRV2:~$ ls
jgula.gpg wiadomosc.txt.gpg wsup.gpg
user@SRV2:~$ nano wiadomosc.txt.gpg
user@SRV2:~$ gpg --verify wiadomosc.txt.gpg
gpg: Podpisano w pon, 31 paź 2022, 17:38:28 CET
gpg:      przy użyciu klucza RSA AD87EC5CAB29CA7D7927765AA15BA4593040DE29
gpg: Poprawny podpis złożony przez „Jacek Gula <jgula@nwtraders.msft>” [pełne]
user@SRV2:~$ gpg --output wiadomosc.txt.gpg --decrypt wiadomosc.txt.gpg
Plik „wiadomosc.txt.gpg” istnieje. Nadpisać? (t/N) t
gpg: Podpisano w pon, 31 paź 2022, 17:38:28 CET
gpg:      przy użyciu klucza RSA AD87EC5CAB29CA7D7927765AA15BA4593040DE29
gpg: Poprawny podpis złożony przez „Jacek Gula <jgula@nwtraders.msft>” [pełne]

```

```

user@SRV2:~$ more wiadomosc.txt.gpg
Moj plik wiadomosc

```

```

GNU nano 5.4 wiadomosc.txt.gpg
xxxxMoj plik wiadomosc

[ Zapisano 1 linię ]
^G Help      ^O Zapisz    ^W Wyszukaj  ^K Cut       ^T Execute   ^C Location
^X Wyjdź     ^R Wczyt.plik ^_ Zastąp    ^U Paste     ^J Wyjustuj  ^_ Do linii

```

```
user@SRV2:~$ rm wiadomosc.txt.gpg
user@SRV2:~$ ls
jgula.gpg  wsup.gpg
user@SRV2:~$ |
```

Zadanie 11

Jesteś administratorem systemów w firmie Northwind Traders (nwtraders.msft).

Otrzymałeś prośbę, aby korzystając w serwerze Linux SRV1 z oprogramowania GPG u użytkownika "user" podpisać cyfrowo plik "wiadomosc.txt" metodą "clearsign", przesłać go z serwera Linux SRV1 do serwera Linux SRV2 do katalogu domowego użytkownika "user" i tam zweryfikować podpis cyfrowy tegoż pliku.

Twoim zadaniem jest więc:

- **zalogować się w serwerze Linux SRV1 na użytkownika "user", a następnie podpisać cyfrowo plik "*wiadomosc.txt*" i przesłać zaszyfrowany plik na serwer Linux SRV2:**
\$ cd /home/user\$ gpg --clearsign wiadomosc.txt
\$ ls
\$ scp wiadomosc.txt.asc user@172.16.0.2:/home/user
\$ rm wiadomosc.txt.asc
- **zalogować się w serwerze Linux SRV2 na użytkownika "user", następnie zweryfikować czy możliwe jest wyedytowanie pliku "*wiadomosc.txt.asc*"**
- **zalogować się w serwerze Linux SRV2 na użytkownika "user", następnie zweryfikować podpis cyfrowy pliku "*wiadomosc.txt.asc*", po czym go odszyfrować:**
\$ cd /home/user
\$ gpg --verify wiadomosc.txt.asc
\$ gpg --output wiadomosc.txt --decrypt wiadomosc.txt.asc
- **zalogować się w serwerze Linux SRV2 na użytkownika "user", następnie zweryfikować czy możliwe jest wyedytowanie pliku "*wiadomosc.txt*"**
- **zalogować się w serwerze Linux SRV2 na użytkownika "user", następnie usunąć plik "*wiadomosc.txt*":**
\$ rm /home/user/wiad*

```
user@SRV1:~$ pwd
/home/user
user@SRV1:~$ gpg --clearsign wiadomosc.txt
```

Proszę wprowadzić hasło do odbezpieczenia klucza prywatnego OpenPGP:
,,Jacek Gula <jgula@nwtraders.msft>''
klucz 3072-bitowy RSA, ID A15BA4593040DE29,
utworzony 2022-10-28.

Hasło: *****|-----

<OK>

<Anuluj>

```
user@SRV1:~$ ls
jgula.gpg wiadomosc.txt wiadomosc.txt.asc wsup.gpg
user@SRV1:~$ scp wiadomosc.txt.asc user@192.168.68.112:/home/user
user@192.168.68.112's password:
wiadomosc.txt.asc                               100% 727      1.2MB/s   00:00
user@SRV1:~$ rm wiadomosc.txt.asc
```

```
user@SRV1: ~ user@SRV2: ~
GNU nano 5.4 wiadomosc.txt.asc
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA512

Moj plik wiadomosc
-----BEGIN PGP SIGNATURE-----

iQGzBAEBCgAdFiEErYfsXKspyn15J3ZaoVukWTBA3ikFAMNgDZsACgkQoVukWTBA
3ik7Lwv+Joe0o0fUqo/d9Tww5NtCUt/t+NV/c6UqATqEQuuRSkFGyBKeGqPMYTKd
M3dPq/iz8Ju2DPxcU4W4zmweSFPEzFavDRoPtzyWbKfn2HGxzecpLR00vrWIVCoZ
uMGttOG108qdFAkAo4PgbAuxrrv/Oz+Fimm/uEhOuXkAe6rPlcE7+YCNak9bPTg+
Q7bZB5NF2QQGsII8btcaafai5b0elv4G/vdzt6sk92dkR/gXQhUgCZ5zNodg/ZfL
Vm0BYm+VMz5CeRUY8mgiAdRRV9Q1AP8rP2Ast5Tg7Nm/viXfLKMtB5mnWMqF5LJV
Pj+Dd+ruR3nJrIMJ12kMsTXNNfrvG9YMOH400G7LYGP502XfBIv1Qo6lrxjhSo0V
L4pFOu9Le51chBEk3tdXN8XNuStiMbFb//zEEz5KUHlisqgxFo0Uw0LWuxBVxDqy
c9hQT2mmZ428RB+VfkeELScok9rEQjdHmThPq/wn1U2TJNERl18D2WHJQJ7wuRxX
|
=bRT8
-----END PGP SIGNATURE-----

[ Zapisano 18 linii ]
^G Help      ^O Zapisz    ^W Wyszukaj  ^K Cut       ^T Execute   ^C Location
^X Wyjdź     ^R Wczyt.plik ^\ Zastap   ^U Paste     ^J Wyjustuj  ^_ Do linii
```

```
user@SRV2:~$ gpg --verify wiadomosc.txt.asc
gpg: Błąd sumy CRC; 1712E5 - 6D14FC
gpg: nie znaleziono podpisu
gpg: nie można sprawdzić podpisu.
Należy pamiętać o podawaniu pliku podpisu (.sig lub .asc) jako pierwszego
argumentu linii poleceń.
```



```
GNU nano 5.4                                wiadomosc.txt.asc
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA512

Moj plik wiadomosc
-----BEGIN PGP SIGNATURE-----

iQGzBAEBCgAdFiEErYfsXKspyn15J3ZaoVukWTBA3ikFAmNgDLAACgkQoVukWTBA
3inNVwwAj2xvDaVpLSWAGIbmqcitBmVxvpQAlS0aqfSdbq8WX1QFWQNJMXcY2gwS
eDZDR13Ny2LnvoS2IL69HumvFi+4Ff5cYNPMKA845c5y/f5m0C5J2+GUBVKhiRZT
PGzPgeJeJfLhFsoCv3b7qktpzymT42q3jQBfFwdWqBzNljte/n4rc2I+XH/6fY6e
/8CXUiFg7Q/wROhl4WuJbQgxookZdsJAbYLAHz2ZqjTc0bcBPF8LZaowyfRHG+gB
gIZABoGToDqH772Mcx0SgOoW8bn4HKWGemSM8049LMSAon20UwtmcRr2wMf3GXpC
1ZDnycaGqRVlwSdj1M+XIgkjYfkf0aDrdtRmnhdtNWy9XT67gfunEtIULWvuZDVq
/Ey8LjWAEaztb6AJafjqyKKdc3sNK0vHciAW3KltXTrWDE111NiQ0INSMoPVfHKd
+0Tob4hPxJnRzJ+JW5dxRijDP1n02iR+GV33cBhm5rR0iWboPR9RPXMpRYJ4iE1T
yyGHkUqP
=oF4a
-----END PGP SIGNATURE-----
```

```
user@SRV2:~$ gpg --verify wiadomosc.txt.asc
gpg: Podpisano w pon, 31 paź 2022, 19:05:04 CET
gpg:      przy użyciu klucza RSA AD87EC5CAB29CA7D7927765AA15BA4593040DE29
gpg: Poprawny podpis złożony przez ,,Jacek Gula <jgula@nwtraders.msft>' [pełne]
user@SRV2:~$ gpg -o wiadomosc.txt -d wiadomosc.txt.asc
gpg: Podpisano w pon, 31 paź 2022, 19:05:04 CET
gpg:      przy użyciu klucza RSA AD87EC5CAB29CA7D7927765AA15BA4593040DE29
gpg: Poprawny podpis złożony przez ,,Jacek Gula <jgula@nwtraders.msft>' [pełne]
user@SRV2:~$ more wiadomosc.txt
Moj plik wiadomosc
user@SRV2:~$ rm wiadomosc.txt*
```

Zadanie 12

Jesteś administratorem systemów w firmie Northwind Traders (nwtraders.msft).

Otrzymałeś prośbę, aby korzystając w serwerze Linux SRV1 z oprogramowania GPG u użytkownika "user" podpisać cyfrowo plik "wiadomosc.txt" metodą "detach", przesłać go z serwera Linux SRV1 do serwera Linux SRV2 do katalogu domowego użytkownika "user" i tam zweryfikować podpis cyfrowy tegoż pliku.

Twoim zadaniem jest więc:

- zalogować się w serwerze Linux SRV1 na użytkownika "user", a następnie podpisać cyfrowo plik "*wiadomosc.txt*" (wykorzystując mocniejszy algorytm haszujący SHA512, w porównaniu ze standardowo używanym SHA1) i przesłać zaszyfrowany plik na serwer Linux SRV2:

\$ cd /home/user

```
$ gpg --detach-sign --digest-algo SHA512 wiadomosc.txt
```

```
$ ls
```

```
$ scp wiadomosc.txt user@172.16.0.2:/home/user
```

```
$ scp wiadomosc.txt.sig user@172.16.0.2:/home/user
```

```
$ rm wiadomosc.txt.sig
```

- zalogować się w serwerze Linux SRV2 na użytkownika "user", następnie zweryfikować czy możliwe jest wyedytowanie pliku "*wiadomosc.txt*", "*wiadomosc.txt.sig*"
- zalogować się w serwerze Linux SRV2 na użytkownika "user", następnie zweryfikować podpis cyfrowy pliku "*wiadomosc.txt.sig*":
\$ cd /home/user
\$ gpg --verify wiadomosc.txt.sig
- zalogować się w serwerze Linux SRV2 na użytkownika "user", następnie wyedytować plik "*wiadomosc.txt*" i w dowolny sposób zmodyfikować jego zawartość
- zalogować się w serwerze Linux SRV2 na użytkownika "user", następnie zweryfikować podpis cyfrowy pliku "*wiadomosc.txt.sig*":
\$ cd /home/user
\$ gpg --verify wiadomosc.txt.sig (zwrócić uwagę na wskazanie informacji "*NIEPOPRAWNY podpis złożony przez*", wskazujący, że weryfikowany plik nie jest dokładnie tym plikiem jaki był oryginalnie podpisywany cyfrowo)
- zalogować się w serwerze Linux SRV2 na użytkownika "user", następnie usunąć plik "*wiadomosc.txt*":
\$ cd /home/user
\$ rm wiadomosc.txt
- zalogować się w serwerze Linux SRV2 na użytkownika "user", następnie zweryfikować podpis cyfrowy pliku "*wiadomosc.txt.sig*":
\$ cd /home/user
\$ gpg --verify wiadomosc.txt.sig (zwrócić uwagę na niemożność zweryfikowania poprawności podpisu, ze względu na brak pliku dla którego ten podpis miałby być weryfikowany)
- zalogować się w serwerze Linux SRV2 na użytkownika "user", następnie usunąć plik "*wiadomosc.txt.sig*":
\$ rm /home/user/wiad*

```

user@SRV1:~$ gpg --detach-sign --digest-algo SHA512 wiadomosc.txt
user@SRV1:~$ ls
jgula.gpg wiadomosc.txt wiadomosc.txt.sig wsup.gpg
user@SRV1:~$ scp wiadomosc.txt.sig user@192.168.68.112:/home/user
user@192.168.68.112's password:
wiadomosc.txt.sig                                100% 438   425.0KB/s   00:00
user@SRV1:~$ scp wiadomosc.txt user@192.168.68.112:/home/user
user@192.168.68.112's password:
wiadomosc.txt                                    100% 19    32.2KB/s   00:00
user@SRV1:~$ rm wiadomosc.txt.sig

```

```

user@SRV1: ~
GNU nano 5.4 wiadomosc.txt
Moj plik wiadomosc

[ Zapisano 1 linie ]
^G Help      ^O Zapisz    ^W Wyszukaj  ^K Cut       ^T Execute   ^C Location
^X Wyjdź     ^R Wczyt.plik ^\ Zastap    ^U Paste     ^J Wyjustuj  ^_ Do linii

```

```
user@SRV1: ~
user@SRV2: ~
GNU nano 5.4 wiadomosc.txt.sig
^A^D^A
^@]^V!^D\}\}y'vZ^[Y@)^E^Bc`^P^@
^P^[Y@){^K^N^Eh^Z^Q^UO&$(F^S^dC%DZM+^D(0* ^AV&X^Wmo}~e9
g/^E^ZR^S^F^I^A
/^__Q^T^K^C^Q^'.^_W)-I^Z^X^K^PG ^c^ B^U^&2j<^r7t8^a^2cw^H^U^H^RD^U/^~:9@>
```

[Zapisano 5 linii]

^G Help	^O Zapisz	^W Wyszukaj	^K Cut	^T Execute	^C Location
^X Wyjdź	^R Wczyt.plik	^_ Zastąp	^U Paste	^J Wyjustuj	^_ Do linii

```

user@SRV2:~$ ls
jgula.gpg wiadomosc.txt wiadomosc.txt.sig wsup.gpg
user@SRV2:~$ nano wiadomosc.txt
user@SRV2:~$ nano wiadomosc.txt.sig
user@SRV2:~$ gpg --verify wiadomosc.txt.sig
gpg: [don't know]: invalid packet (ctb=0a)
gpg: przyjęto obecność podpisanych danych w 'wiadomosc.txt'
gpg: Podpisano w pon, 31 paź 2022, 19:15:42 CET
gpg:      przy użyciu klucza RSA AD87EC5CAB29CA7D7927765AA15BA4593040DE29
gpg: Poprawny podpis złożony przez „Jacek Gula <jgula@nwtraders.msft>” [pełne]
user@SRV2:~$ nano wiadomosc.txt
user@SRV2:~$ more wiadomosc.txt
Moj plik wiadomosc ligma
user@SRV2:~$ gpg --verify wiadomosc.txt.sig
gpg: [don't know]: invalid packet (ctb=0a)
gpg: przyjęto obecność podpisanych danych w 'wiadomosc.txt'
gpg: Podpisano w pon, 31 paź 2022, 19:15:42 CET
gpg:      przy użyciu klucza RSA AD87EC5CAB29CA7D7927765AA15BA4593040DE29
gpg: NIEPOPRAWNY podpis złożony przez „Jacek Gula <jgula@nwtraders.msft>” [pełne]
user@SRV2:~$ rm wiadomosc.txt
user@SRV2:~$ gpg --verify wiadomosc.txt.sig
gpg: [don't know]: invalid packet (ctb=0a)
gpg: brak podpisanych danych
gpg: can't hash datafile: No data
user@SRV2:~$ rm wiad*
user@SRV2:~$ |

```

Zadanie 13

Jesteś administratorem systemów w firmie Northwind Traders (nwtraders.msft).

Otrzymałeś prośbę, aby usunąć w serwerze Linux SRV2 z bazy GPG u użytkownika "user" klucz publiczny jgula@nwtraders.msft.

Twoim zadaniem jest więc:

- **zalogować się w serwerze Linux SRV2 na użytkownika "user", a następnie usunąć z bazy GPG klucz publiczny jgula@nwtraders.msft:**
\$ gpg --delete-key jgula@nwtraders.msft
- **zweryfikować w systemie Linux SRV2 listę posiadanych kluczy, wydając polecenie:**
\$ gpg --list-keys

```

user@SRV2:~$ gpg --delete-key jgula@nwtraders.msft
gpg (GnuPG) 2.2.27; Copyright (C) 2021 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

pub  rsa3072/A15BA4593040DE29 2022-10-28 Jacek Gula <jgula@nwtraders.msft>

Usunąć ten klucz ze zbioru? (t/N) t
user@SRV2:~$ gpg --list-keys
/home/user/.gnupg/pubring.kbx
-----
pub  rsa3072 2022-10-28 [SC] [wygasa: 2024-10-27]
    0E5AD31BE1866F12A4625B9D198FB8BE2EEF0F1D
uid  [ absolutne ] Witold Sup <wsup@nwtraders.msft>
sub  rsa3072 2022-10-28 [E] [wygasa: 2024-10-27]

```

Zadanie 14

Jesteś administratorem systemów w firmie Northwind Traders (nwtraders.msft).

Otrzymałeś prośbę, aby usunąć w serwerze Linux SRV1 z bazy GPG u użytkownika "user" klucz publiczny *wsup@nwtraders.msft*.

Twoim zadaniem jest więc:

- zainstalować w serwerze Linux SRV1 serwer kluczy GPG (wykorzystując użytkownika o uprawnieniach administracyjnych do tegoż celu):
apt-get update
apt-get install sks
- zbudować bazę danych dla serwera SKS i nadać potrzebne uprawnienia:
sks build
chown -R debian-sks:debian-sks /var/lib/sks/DB
chmod g+s /var/lib/sks/DB (istnienie użytkownika i grupy można sprawdzić poleceniem:
grep debian-sks /etc/{passwd,group})
- wyedytować */etc/sks/sksconf* i odhaszować linię: *hkp_port: 11371* jak również w parametrze "hkp_address" dodać adres IP naszego serwera Linux, na którym mamy uruchomiony serwer SKS, np: *hkp_address: 127.0.0.1 ::1 172.16.0.1*
- uruchomić serwer SKS:
systemctl restart sks
- w systemie MS Windows uruchomić przeglądarkę internetową, i przetestować prawidłowe działanie serwera kluczy PGP otwierając stronę internetową: *http://172.16.0.1:11371/* (jeżeli serwer SKS działa, to powinna się pojawić strona internetowa)

- zalogować się w serwerze Linux SRV1 na użytkownika "user", a następnie wyeksportować klucz publiczny *kgula@nwtraders.msft* na serwer kluczy PGP:
\$ gpg --list-keys
\$ gpg --keyserver 172.16.0.1 --send-keys F6E6FBDC
- w serwerze Linux SRV do pliku */etc/hosts* dodać wpis nakierowujący nazwę "srv1" na adres IP serwera Linux SRV1, i zrestartować system Linux SRV2
- zalogować się w serwerze Linux SRV2 na użytkownika "user", a następnie wyszukać i zaimportować klucz publiczny *kgula@nwtraders.msft* z serwera kluczy PGP (pamiętając o podaniu prawidłowego numeru klucza publicznego, podobnie jak we wcześniejszym punkcie):
\$ gpg --keyserver 172.16.0.1 --search-keys kgula
\$ gpg --keyserver 172.16.0.1 --recv-keys F6E6FBDC
- zalogować się w serwerze Linux SRV2 na użytkownika "user", a następnie sprawdzić listę posiadanych kluczy publicznych w bazie PGP (powinien na nowo pojawić się wcześniej usunięty klucz publiczny Jacka Guli):
\$ gpg --list-keys
- w systemie MS Windows można także uruchomić przeglądarkę internetową, otwierając następnie stronę internetową: *http://172.16.0.1:11371/*, i w polu "Search string" wpisać "kgula" a następnie zatwierdzić naciskając klawisz "Enter" (powinna pojawić się informacja o kluczu publicznym Jacka Guli)

```
root@SRV1:~# sks build
root@SRV1:~# chown -R debian-sks:debian-sks /var/lib/sks/DB
root@SRV1:~# chmod g+s /var/lib/sks/DB
root@SRV1:~# grep debian-sks /etc/{passwd,group}
/etc/passwd:debian-sks:x:107:113:./var/lib/sks:/bin/bash
/etc/group:debian-sks:x:113:
```

```
GNU nano 5.4 /etc/sks/sksconf
# Set recon binding address
#recon_address: 0.0.0.0

# Set recon port number
#recon_port: 11370

# Set hkp binding address:
# listen on loopback where possible, and put a good reverse
# HTTP proxy on the public-facing addresses
# (see https://github.com/SKS-Keyserver/sks-keyserver/wiki/Peering)
hkp_address: 127.0.0.1 ::1 192.168.68.111

# Set hkp port number
hkp_port: 11371

# Have the HKP interface listen on port 80, as well as the hkp_port
#use_port_80:

# From address used in synchronization emails used to communicate with PKS
#from_addr: "PGP Key Server Administrator <pgp-public-keys@this.server.example.net>"

# Command used for sending mail (you can use -f option to specify the
# envelope sender address, if your MTA trusts the sks user)
#sendmail_cmd: /usr/lib/sendmail -t -oi

# Runs database statistics calculation on boot (time and cpu expensive)
#initial_stat:

# bdb's db_tune program suggests a pagesize of 65536 for [K]DB/key. In practice
# this caused page deadlocks. I found 8K (16) and 16K (32) to be better values
pagesize:      16
#
# The tuner recommended 4096 (8) for the pagesize for PTree/ptree. I have had
# very good results with 8196 (16)
ptree_pagesize: 16

[ Zapisano 43 linie ]
^G Help      ^O Zapisz    ^W Wyszukaj  ^K Cut      ^T Execute  ^C Location
^X Wyjdź     ^R Wczyt.plik ^\ Zastap   ^U Paste    ^J Wyjustuj ^_ Do linii
```

```
root@SRV1:~# systemctl restart sks
root@SRV1:~# |
```


SKS key server

Niezabezpieczona | 192.168.68.111:11371

SKS OpenPGP Key server

Extract a key

You can find a key by typing in some words that appear in the userid (name, email, etc.) of the key you're looking for, or by typing in the keyid in hex format ("0x...")

Search for a public key

String

0xDEADBEEF

Show PGP Fingerprints

☐

Show SKS full-key hashes

☐

Get regular index of matching keys

☐

Get verbose index of matching keys

☒

Retrieve ascii-armored keys

☐

Retrieve keys by full-key hash

☐

Reset

Search for a key

Submit a key

You can submit a key by simply pasting in the ASCII-armored version of your key and clicking on submit.

Reset

Submit this key

SKS is a new [OpenPGP](#) keyserver. The main innovation of SKS is that it includes a highly-efficient reconciliation algorithm for keeping the keyservers synchronized.

[SKS statistics](#)

```

user@SRV1:~$ gpg --list-keys
/home/user/.gnupg/pubring.kbx
-----
pub   rsa3072 2022-10-28 [SC] [wygasa: 2024-10-27]
      AD87EC5CAB29CA7D7927765AA15BA4593040DE29
uid   [ absolutne ] Jacek Gula <jgula@nwtraders.msft>
sub   rsa3072 2022-10-28 [E] [wygasa: 2024-10-27]

pub   rsa3072 2022-10-28 [SC] [wygasa: 2024-10-27]
      0E5AD31BE1866F12A4625B9D198FB8BE2EEF0F1D
uid   [ pełne ] Witold Sup <wsup@nwtraders.msft>
sub   rsa3072 2022-10-28 [E] [wygasa: 2024-10-27]

```

```

user@SRV1:~$ gpg --keyserver 192.168.68.111 --send-keys AD87EC5CAB29CA7D7927765AA15BA4593040DE29
gpg: wysyłanie klucza A15BA4593040DE29 na hkp://192.168.68.111
user@SRV1:~$

```

```

user@SRV1: ~
user@SRV2: ~
GNU nano 5.4 /etc/hosts
127.0.0.1    localhost
127.0.1.1    debian11
192.168.68.111 srv1

# The following lines are desirable for IPv6 capable hosts
::1          localhost ip6-localhost ip6-loopback
ff02::1      ip6-allnodes
ff02::2      ip6-allrouters

[ Zapisano 8 linii ]
^G Help      ^O Zapisz    ^W Wyszukaj  ^K Cut       ^T Execute   ^C Location
^X Wyjdź     ^R Wczyt.plik ^\ Zastąp    ^U Paste     ^J Wyjustuj  ^_ Do linii

```

```
user@debian11:~$ gpg --keyserver 192.168.68.111 --search-keys jgula
gpg: data source: http://192.168.68.111:11371
(1) Jacek Gula <jgula@nwtraders.msft>
    3072 bit RSA key A15BA4593040DE29, utworzono: 2022-10-28, wygasa: 2024-10-27
Keys 1-1 of 1 for "jgula". Wprowadź numer(y), N)astępny lub Q)uit > y
```

```
user@debian11:~$ gpg --keyserver 192.168.68.111 --recv-keys A15BA4593040DE29
gpg: klucz A15BA4593040DE29: klucz publiczny „Jacek Gula <jgula@nwtraders.msft>” wczytano do z
bioru
gpg: Ogółem przetworzonych kluczy: 1
gpg: dołączono do zbioru: 1
```

```
user@debian11:~$ gpg --list-keys
/home/user/.gnupg/pubring.kbx
-----
pub  rsa3072 2022-10-28 [SC] [wygasa: 2024-10-27]
    0E5AD31BE1866F12A4625B9D198FB8BE2EEF0F1D
uid  [ absolutne ] Witold Sup <wsup@nwtraders.msft>
sub  rsa3072 2022-10-28 [E] [wygasa: 2024-10-27]

pub  rsa3072 2022-10-28 [SC] [wygasa: 2024-10-27]
    AD87EC5CAB29CA7D7927765AA15BA4593040DE29
uid  [ pełne ] Jacek Gula <jgula@nwtraders.msft>
sub  rsa3072 2022-10-28 [E] [wygasa: 2024-10-27]
```

Search results for 'jgula'

×

+

←

→

↺

⚠ Niezabezpieczona | 192.168.68.111:11371/pks/lookup?search=jgula&

Information displayed on this website, including public keyblocks and anything associated with them, *is not cryptogra*
keyblocks using OpenPGP software on a secured device that you control to see verified information.

Search results for 'jgula'

Type	bits/keyID	cr. time	exp time	key expir
------	------------	----------	----------	-----------

pub	3072R/ 3040DE29	2022-10-28		
uid	Jacek Gula <jgula@nwtraders.msft>			
sig	sig3 3040DE29	2022-10-28	2024-10-27	[selfsig]
sub	3072R/7949EEDC	2022-10-28		
sig	sbind 3040DE29	2022-10-28	2024-10-27	[.]

Zadanie 15

Jesteś administratorem systemów w firmie Northwind Traders (nwtraders.msft).

Otrzymałeś prośbę, aby usunąć w serwerze Linux SRV1 z bazy GPG u użytkownika "user" klucz publiczny wsup@nwtraders.msft.

Twoim zadaniem jest więc:

- **zalogować się w serwerze Linux SRV1 na użytkownika "user", a następnie odwołać w ramach GPG certyfikat *jgula@nwtraders.msft*:**
\$ gpg -output jgula_revoke.gpg -gen-revoke jgula@nwtraders.msft
\$ gpg --edit-key jgula@nwtraders.msft
gpg> revkey
gpg> quit
 - **zalogować się w serwerze Linux SRV1 na użytkownika "user", a następnie wyeksportować klucz publiczny *jgula@nwtraders.msft* na serwer kluczy PGP:**
\$ gpg --keyserver 172.16.0.1 --send-keys F6E6FBDC
 - **w systemie MS Windows uruchomić przeglądarkę internetową, i przetestować unieważniony klucz publiczny *jgula@nwtraders.msft* w serwerze kluczy PGP otwierając stronę internetową:**
http://172.16.0.1:11371/* i przeglądając informacje o statusie klucza publicznego *jgula@nwtraders.msft
 - **zalogować się w serwerze Linux SRV2 na użytkownika "user", a następnie wyszukać i zaimportować klucz publiczny *jgula@nwtraders.msft* z serwera kluczy PGP::**
\$ gpg --keyserver 172.16.0.1 --search-keys jgula
\$ gpg --keyserver 172.16.0.1 --recv-keys F6E6FBDC
lub zalogować się w serwerze Linux SRV2 na użytkownika "user", a następnie importujemy przekopiowany "ręcznie" odwołany certyfikat:
\$ scp jgula_revoke.gpg user@172.16.0.2:/home/user
\$ gpg --import /home/user/jgula_revoke.gpg
 - **zalogować się w serwerze Linux SRV2 na użytkownika "user", a następnie zweryfikować, czy klucz publiczny *jgula@nwtraders.msft* został unieważniony:**
\$ gpg --lists-keys
-

```

user@SRV1:~$ gpg -o jgula_revoke.gpg --gen-revoke jgula@nwtraders.msft
sec  rsa3072/A15BA4593040DE29 2022-10-28 Jacek Gula <jgula@nwtraders.msft>

Stworzyć certyfikat unieważnienia tego klucza? (t/N) t
Proszę wybrać powód unieważnienia:
  0 = nie podano przyczyny
  1 = klucz został skompromitowany
  2 = klucz został zastąpiony
  3 = klucz nie jest już używany
  Q = Anuluj
(Prawdopodobnie chcesz tu wybrać 1)
Decyzja? 0
Wprowadź opis (nieobowiązkowy) i zakończ go pustą linią:
>
Powód unieważnienia: nie podano przyczyny
(nie podano)
Informacje poprawne? (t/N) t
wymuszono opakowanie ASCII wyniku.
Certyfikat unieważnienia został utworzony.

Należy przenieść go na nośnik który można bezpiecznie ukryć; jeśli źli ludzie
dostaną ten certyfikat w swoje ręce, mogą użyć go do uczynienia klucza
nieużytecznym.

Nieźłym pomysłem jest wydrukowanie certyfikatu unieważnienia i schowanie
wydruku w bezpiecznym miejscu, na wypadek gdyby nośnik z certyfikatem stał się
nieczytelny. Ale należy zachować ostrożność, systemy drukowania różnych
komputerów mogą zachować treść wydruku i udostępnić ją osobom nieupoważnionym.

```

```

Proszę wprowadzić hasło do odbezpieczenia klucza prywatnego OpenPGP:
,,Jacek Gula <jgula@nwtraders.msft>''
klucz 3072-bitowy RSA, ID A15BA4593040DE29,
utworzony 2022-10-28.

```

```

Hasło: *****|-----

```

```

<OK>

```

```

<Anuluj>

```

```

user@SRV1:~$ gpg --edit-key jgula@nwtraders.msft
gpg (GnuPG) 2.2.27; Copyright (C) 2021 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Dostępny jest klucz tajny.

sec  rsa3072/A15BA4593040DE29
    utworzono: 2022-10-28  wygasa: 2024-10-27  użycie: SC
    zaufanie: absolutne    poprawność: absolutne
ssb  rsa3072/BE98A5587949EEDC
    utworzono: 2022-10-28  wygasa: 2024-10-27  użycie: E
[ absolutne ] (1). Jacek Gula <jgula@nwtraders.msft>

gpg> revkey
Czy na pewno chcesz unieważnić cały klucz? (t/N) t
Proszę wybrać powód unieważnienia:
  0 = nie podano przyczyny
  1 = klucz został skompromitowany
  2 = klucz został zastąpiony
  3 = klucz nie jest już używany
  Q = Anuluj
Decyzja? 0
Wprowadź opis (nieobowiązkowy) i zakończ go pustą linią:
>
Powód unieważnienia: nie podano przyczyny
(nie podano)
Informacje poprawne? (t/N) t

Ten klucz został unieważniony 2022-10-31 przez klucz użytkownika RSA A15BA4593040DE29 Jacek Gula
<jgula@nwtraders.msft>
sec  rsa3072/A15BA4593040DE29
    utworzono: 2022-10-28  unieważniono: 2022-10-31  użycie: SC
    zaufanie: absolutne    poprawność: unieważniony
Ten klucz został unieważniony 2022-10-31 przez klucz użytkownika RSA A15BA4593040DE29 Jacek Gula
<jgula@nwtraders.msft>
ssb  rsa3072/BE98A5587949EEDC
    utworzono: 2022-10-28  unieważniono: 2022-10-31  użycie: E
[ unieważniony ] (1). Jacek Gula <jgula@nwtraders.msft>

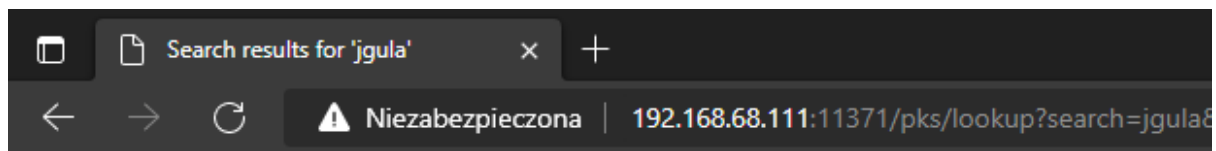
gpg> quit
Zapisać zmiany? (t/N) t

```

```

user@SRV1:~$ gpg --keyserver 192.168.68.111 --send-keys A15BA4593040DE29
gpg: wysyłanie klucza A15BA4593040DE29 na hkp://192.168.68.111

```



Search results for 'jgula'

Type bits/keyID cr. time exp time key expir

```
pub 3072R/3040DE29 2022-10-28
sig revok 3040DE29 2022-10-31 [selfsig]

uid Jacek Gula <jgula@nwtraders.msft>
sig sig3 3040DE29 2022-10-28 2024-10-27 [selfsig]

sub 3072R/7949EEDC 2022-10-28
sig sbind 3040DE29 2022-10-28 2024-10-27 [.]
```

```
user@SRV2:~$ gpg --keyserver 192.168.68.111 --search-keys jgula
gpg: data source: http://192.168.68.111:11371
(1) Jacek Gula <jgula@nwtraders.msft>
    3072 bit RSA key A15BA4593040DE29, utworzono: 2022-10-28, wygasa: 2024-10-27 (unieważniony)
Keys 1-1 of 1 for "jgula". Wprowadź numer(y), N)astępny lub Q)uit > q
gpg: error searching keyserver: Operation cancelled
gpg: szukanie w serwerze kluczy nie powiodło się: Operation cancelled
user@SRV2:~$ gpg --keyserver 192.168.68.111 --recv-keys A15BA4593040DE29
gpg: klucz A15BA4593040DE29: „Jacek Gula <jgula@nwtraders.msft>” dodany certyfikat unieważnienia
gpg: klucz A15BA4593040DE29: „Jacek Gula <jgula@nwtraders.msft>” 1 nowy podpis
gpg: Ogółem przetworzonych kluczy: 1
gpg: nowych podpisów: 1
user@SRV2:~$
```

```
user@SRV2:~$ gpg --list-keys
/home/user/.gnupg/pubring.kbx
-----
pub rsa3072 2022-10-28 [SC] [wygasa: 2024-10-27]
    0E5AD31BE1866F12A4625B9D198FB8BE2EEF0F1D
uid [ absolutne ] Witold Sup <wsup@nwtraders.msft>
sub rsa3072 2022-10-28 [E] [wygasa: 2024-10-27]

pub rsa3072 2022-10-28 [SC] [unieważniono: 2022-10-31]
    AD87EC5CAB29CA7D7927765AA15BA4593040DE29
uid [ unieważniony ] Jacek Gula <jgula@nwtraders.msft>
```

Zadanie 16

Jesteś administratorem systemów w firmie Northwind Traders (nwtraders.msft).

Otrzymałeś prośbę, aby usunąć w serwerze Linux SRV1 z bazy GPG u użytkownika "user" jego pełny klucz asymetryczny *jgula@nwtraders.msft*.

Twoim zadaniem jest więc:

- zalogować się w serwerze Linux SRV1 na użytkownika "user", a następnie usunąć z bazy GPG klucz prywatny *jgula@nwtraders.msft*:
`$ gpg --delete-secret-key jgula@nwtraders.msft`
- zalogować się w serwerze Linux SRV1 na użytkownika "user", a następnie usunąć z bazy GPG klucz publiczny *jgula@nwtraders.msft*:
`$ gpg --delete-key jgula@nwtraders.msft`

```
user@SRV1:~$ gpg --delete-secret-key jgula@nwtraders.msft
gpg (GnuPG) 2.2.27; Copyright (C) 2021 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

sec  rsa3072/A15BA4593040DE29 2022-10-28 Jacek Gula <jgula@nwtraders.msft>

Usunąć ten klucz ze zbioru? (t/N) t
To jest klucz tajny! - czy na pewno go usunąć? (t/N) t
```

```
Czy na pewno trwale usunąć klucz prywatny OpenPGP:
,,Jacek Gula <jgula@nwtraders.msft>''
klucz 3072-bitowy RSA, ID A15BA4593040DE29,
utworzony 2022-10-28.
?

<Usuń klucz> <Nie>
```

```
user@SRV1:~$ gpg --delete-key jgula@nwtraders.msft
gpg (GnuPG) 2.2.27; Copyright (C) 2021 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

pub  rsa3072/A15BA4593040DE29 2022-10-28 Jacek Gula <jgula@nwtraders.msft>

Usunąć ten klucz ze zbioru? (t/N) t
```