Politechnika Wrocławska, Informatyka Stosowana

ZASTOSOWANIA KRYPTOGRAFII

Cyberbezpieczeństwo, Laboratorium nr.6 - raport

Autor: Aleksander Stepaniuk
Nr. Indeksu: 272644

Zad 1. Zastosowanie kryptografii

Zadanie 1.1;

Przebieg zadań:

```
student@student-ubuntu:~/exchange/lab7$ openssl genrsa -out private_key.pem 2048
student@student-ubuntu:~/exchange/lab7$ ls
private_key.pem
```

```
student@student-ubuntu:~/exchange/lab7$ openssl rsa -in private_key.pem -outform PEM -pubout -out public_key.pem
writing RSA key
student@student-ubuntu:~/exchange/lab7$ ls
private_key.pem  public_key.pem
student@student-ubuntu:~/exchange/lab7$
```

```
student@student-ubuntu:~/exchange/lab7$ openssl dgst -sha256 -sign private_key.pem -out encrypted_hash.sha256 data.txt
student@student-ubuntu:~/exchange/lab7$ ls
data.txt  encrypted_hash.sha256  private_key.pem  public_key.pem
student@student-ubuntu:~/exchange/lab7$
```

```
student@student-ubuntu:~/exchange/lab7$ openssl dgst -sha256 -verify public_key.pem -signature encrypted_hash.sha256 da
ta.txt
Verified OK
student@student-ubuntu:~/exchange/lab7$
```

```
student@student-ubuntu:~/exchange/lab7$ gpg --full-generate-key
gpg (GnuPG) 2.4.4; Copyright (C) 2024 g10 Code GmbH
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
```

```
student@student-ubuntu:~/exchange/lab7$ gpg --list-keys
gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   2  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 2u
gpg: next trustdb check due at 2024-12-17
/home/student/.gnupg/pubring.kbx
--------------------------------
pub   ed25519 2024-11-17 [SC] [expires: 2024-12-17]
      90B60D3C6F094F7C4465E82027E6A25E7E129B91
uid           [ultimate] Aleksander
sub   cv25519 2024-11-17 [E] [expires: 2024-12-17]

pub   rsa2048 2024-11-17 [SC] [expires: 2024-12-17]
      06DC4A318211942DA703F0A92D689CFA475F7A02
uid           [ultimate] Aleksander
sub   rsa2048 2024-11-17 [E] [expires: 2024-12-17]
```

Zadanie 1.2;

Przebieg zadań:

```
student@student-ubuntu:~/exchange/lab7$ gpg --export --armour --output public_key_pgp2.pem Aleksander
student@student-ubuntu:~/exchange/lab7$ ls
public_key.pem  public_key_pgp2.pem  public_key_pgp.pem
student@student-ubuntu:~/exchange/lab7$ gpg --export --armour --output public_openpgp.asc Aleksander
student@student-ubuntu:~/exchange/lab7$ openssl rsa -in private_key.pem -pubout -out public_key.pem
writing RSA key
```

```
student@student-ubuntu:~/exchange/lab7$ ls
public_key.pem  public_pgp.asc
```

Zadanie 1.3;

```
┌──(stud㉿kali-vm)-[~/exchange/lab7]
└─$ ls
public_key.pem  public_key_pgp.pem

┌──(stud㉿kali-vm)-[~/exchange/lab7]
└─$
```

Zadanie 1.4;

```
┌──(stud㉿kali-vm)-[~/exchange/lab7]
└─$ gpg --import public_pgp.asc
gpg: key 27E6A25E7E129B91: "Aleksander" not changed
gpg: key 2D689CFA475F7A02: public key "Aleksander" imported
gpg: Total number processed: 2
gpg:               imported: 1
gpg:              unchanged: 1

┌──(stud㉿kali-vm)-[~/exchange/lab7]
└─$
```

```
┌──(stud㉿kali-vm)-[~/exchange/lab7]
└─$ gpg --fingerprint Aleksander
pub   ed25519 2024-11-17 [SC] [expires: 2024-12-17]
      90B6 0D3C 6F09 4F7C 4465  E820 27E6 A25E 7E12 9B91
uid           [ unknown] Aleksander
sub   cv25519 2024-11-17 [E] [expires: 2024-12-17]

pub   rsa2048 2024-11-17 [SC] [expires: 2024-12-17]
      06DC 4A31 8211 942D A703  F0A9 2D68 9CFA 475F 7A02
uid           [ unknown] Aleksander
sub   rsa2048 2024-11-17 [E] [expires: 2024-12-17]
```

```
┌──(stud㉿kali-vm)-[~/exchange/lab7]
└─$ gpg --sign-key Aleksander

pub  ed25519/27E6A25E7E129B91
     created: 2024-11-17  expires: 2024-12-17  usage: SC
     trust: unknown       validity: unknown
sub  cv25519/BD75FE34677041ED
     created: 2024-11-17  expires: 2024-12-17  usage: E
[ unknown] (1). Aleksander

gpg: no default secret key: No secret key

Key not changed so no update needed.
```

```
┌──(stud㉿kali-vm)-[~/exchange/lab7]
└─$ gpg --sign-key Aleksander

pub  ed25519/27E6A25E7E129B91
     created: 2024-11-17  expires: 2024-12-17  usage: SC
     trust: unknown       validity: unknown
sub  cv25519/BD75FE34677041ED
     created: 2024-11-17  expires: 2024-12-17  usage: E
[ unknown] (1). Aleksander


pub  ed25519/27E6A25E7E129B91
     created: 2024-11-17  expires: 2024-12-17  usage: SC
     trust: unknown       validity: unknown
 Primary key fingerprint: 90B6 0D3C 6F09 4F7C 4465  E820 27E6 A25E 7E12 9B91

     Aleksander

This key is due to expire on 2024-12-17.
Are you sure that you want to sign this key with your
key "Aleksander" (430CF873874EAA40)

Really sign? (y/N) y


┌──(stud㉿kali-vm)-[~/exchange/lab7]
└─$
```

```
┌──(stud㉿kali-vm)-[~/exchange/lab7]
└─$ gpg --fingerprint Aleksander
gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   1  signed:   1  trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: depth: 1  valid:   1  signed:   0  trust: 1-, 0q, 0n, 0m, 0f, 0u
gpg: next trustdb check due at 2024-12-17
pub   ed25519 2024-11-17 [SC] [expires: 2024-12-17]
      90B6 0D3C 6F09 4F7C 4465  E820 27E6 A25E 7E12 9B91
uid           [  full  ] Aleksander
sub   cv25519 2024-11-17 [E] [expires: 2024-12-17]

pub   rsa2048 2024-11-17 [SC] [expires: 2024-12-17]
      06DC 4A31 8211 942D A703  F0A9 2D68 9CFA 475F 7A02
uid           [ unknown] Aleksander
sub   rsa2048 2024-11-17 [E] [expires: 2024-12-17]

pub   rsa2048 2024-11-17 [SC] [expires: 2024-12-17]
      9800 3D10 ECE4 3F9C 9C30  26E3 430C F873 874E AA40
uid           [ultimate] Aleksander
sub   rsa2048 2024-11-17 [E] [expires: 2024-12-17]
```

Zadanie 1.5;

```
student@student-ubuntu:~/exchange/lab7$ echo "test" > data.txt
student@student-ubuntu:~/exchange/lab7$ ls
data.txt  public_key.pem  public_pgp.asc
```

```
student@student-ubuntu:~/exchange/lab7$ openssl dgst -sha256 -sign private_key.pem -out data.sig data.txt
student@student-ubuntu:~/exchange/lab7$ ls
data.sig  data.txt  encrypted_hash.sha256  private_key.pem  public_key.pem  public_pgp.asc
```

```
student@student-ubuntu:~/exchange/lab7$ gpg --armor --sign data.txt
student@student-ubuntu:~/exchange/lab7$ ls
data.sig  data.txt  data.txt.asc  encrypted_hash.sha256  private_key.pem  public_key.pem  public_pgp.asc
student@student-ubuntu:~/exchange/lab7$
```

```
student@student-ubuntu:~/exchange/lab7$ gpg --sign data.txt
student@student-ubuntu:~/exchange/lab7$ ls
data.sig  data.txt  data.txt.asc  data.txt.gpg  encrypted_hash.sha256  private_key.pem  public_key.pem  public_pgp.asc
student@student-ubuntu:~/exchange/lab7$
```

```
student@student-ubuntu:~/exchange/lab7$ gpg --sign -u Aleksander data.txt
File 'data.txt.gpg' exists. Overwrite? (y/N) y
student@student-ubuntu:~/exchange/lab7$
```

Zadanie 1.6;

```
┌──(stud㉿kali-vm)-[~/exchange/lab7]
└─$ openssl dgst -sha256 -verify public_key.pem -signature data.sig data.txt
Verified OK
```

```
┌──(stud㉿kali-vm)-[~/exchange/lab7]
└─$ gpg --verify data.txt.gpg
gpg: Signature made Sun Nov 17 16:43:21 2024 CET
gpg:                using EDDSA key 90B60D3C6F094F7C4465E82027E6A25E7E129B91
gpg: Good signature from "Aleksander" [full]
```

Zadanie 1.7;

```
┌──(stud㉿kali-vm)-[~/exchange/lab7]
└─$ echo "teraz plik jest inny wiec sie nie powinno zgadzac" > data.txt
```

```
┌──(stud㉿kali-vm)-[~/exchange/lab7]
└─$ openssl dgst -sha256 -verify public_key.pem -signature data.sig data.txt
Verification failure
4096842FF07F0000:error:02000068:rsa routines:ossl_rsa_verify:bad signature:..
/crypto/rsa/rsa_sign.c:426:
4096842FF07F0000:error:1C880004:Provider routines:rsa_verify:RSA lib:../provi
ders/implementations/signature/rsa_sig.c:785:
```

```
┌──(stud㉿kali-vm)-[~/exchange/lab7]
└─$ gpg --armor --detach-sign data.txt
File 'data.txt.asc' exists. Overwrite? (y/N) y

┌──(stud㉿kali-vm)-[~/exchange/lab7]
└─$ gpg --verify data.txt.asc data.txt
gpg: Signature made Sun Nov 17 17:00:37 2024 CET
gpg:                using RSA key 98003D10ECE43F9C9C3026E3430CF873874EAA40
gpg: Good signature from "Aleksander" [ultimate]

┌──(stud㉿kali-vm)-[~/exchange/lab7]
└─$ echo "nowa zawartosc pliku" > data.txt

┌──(stud㉿kali-vm)-[~/exchange/lab7]
└─$ gpg --verify data.txt.asc data.txt
gpg: Signature made Sun Nov 17 17:00:37 2024 CET
gpg:                using RSA key 98003D10ECE43F9C9C3026E3430CF873874EAA40
gpg: BAD signature from "Aleksander" [ultimate]

┌──(stud㉿kali-vm)-[~/exchange/lab7]
└─$
```

Dla gpg musimy zrobić detach-sign, inaczej zmiana oryginalnego dokumentu nic nie zmieni, bo nie będzie porównywał sygnatury z plikiem data.txt (sygnatura + zawartość pliku będzie zapisana razem w jednym pliku)

Zadanie 1.8;

```
student@student-ubuntu:~/exchange/lab7$ gpg --encrypt --recipient Aleksander --output secret_data.txt data.txt
student@student-ubuntu:~/exchange/lab7$ ls
data.sig  data.txt.asc  encrypted_hash.sha256  public_key.pem  secret_data.txt
```

```
student@student-ubuntu:~/exchange/lab7$ gpg --decrypt --output received_data.txt secret_data.txt
gpg: encrypted with cv25519 key, ID BD75FE34677041ED, created 2024-11-17
      "Aleksander"
student@student-ubuntu:~/exchange/lab7$ nano received_data.txt
```

```
  GNU nano 7.2                                          received_data.txt
nowa zawartosc pliku
```

```
  GNU nano 7.2                                          secret_data.txt *
◆^^C◆u◆4gpA◆^R^A^G@^W◆◆^Fr~W◆◆◆F^ST◆◆b◆◆◆◆|l◆h  ◆^V◆8◆6>0◆◆◆◆gGQ◆^\◆{◆O◆A◆◆6&Y◆◆◆◆W.1^\_^C>◆|◆◆9◆Б◆◆(◆|Q
◆n◆b^A  ^B^Px◆fo◆u◆^@◆I◆◆^T^X◆◆◆◆◆:◆cz}◆^D◆="Ydj뇨^SU^Y◆R◆Б3Vbi◆^S◆ ^P◆^NQ◆_^X9O^◆◆M\B◆+e%◆◆;^?^BZcO ◆B.◆◆>◆◆◆:t◆^\◆
```

```
student@student-ubuntu:~/exchange/lab7$ gpg --decrypt --output received_data.txt secret_data.txt
gpg: encrypted with cv25519 key, ID BD75FE34677041ED, created 2024-11-17
      "Aleksander"
gpg: gcry_cipher_checktag failed: Checksum error
gpg: [don't know]: invalid packet (ctb=00)
gpg: [don't know]: invalid packet (ctb=00)
student@student-ubuntu:~/exchange/lab7$
```

Pytanie 1.9;

Teoretycznie można wygenerować np. tylko klucz prywatny, ale nie ma to sensu, ponieważ klucz publiczny jest niezbędny do szyfrowania wiadomości oraz weryfikacji podpisów. Para kluczy działa tylko i wyłącznie razem przy korzystaniu z komunikacji asymetrycznej.

Pytanie 1.10;

Klucz GPG zawiera metadane, takie jak ID użytkownika, adres email, komentarze i jest zapisywany w formacie OpenPGP. Klucz PEM, jest ogólnym formatem do przechowywania kluczy w postaci tekstowej (base64) i nie posiada dodatkowych metadanych, ani funkcjonalności charakterystycznych dla OpenPGP.

Pytanie 1.11;

Wymiana kluczy prywatnych jest nieuzasadaniona, a do tego dosyć niebezpieczna, ponieważ klucz prywatny powinien pozostać tajny, aby zapewnić bezpieczeństwo procesu kryptograficznego. Udostępnienie takiego klucza narusza całą zasadę i sens kryptografii asymetrycznej.

Pytanie 1.12;

W zadaniu 1.7 podpis cyfrowy stał się niepoprawny po modyfikacji pliku. Wynika to z tego, że podpis związany jest sumą kontrolną z oryginalną treścią i zmiana nawet choćby jednego znaku powoduje, że sumy te nie będą do siebie pasować. Reguła ta nie sprawdza się dla plików, posiadających zapisaną sygnaturę razem z danymi w jednym pliku, bo wtedy oryginalny plik nie ma wpływu na nic, bo została wykonana kopia danych i zapisanie ich razem z podpisem w jednym pliku.

Pytanie 1.13;

Odcisk palca to inaczej skrót (hash) klucza publicznego, który umożliwia łatwą i jednoznaczną identyfikację klucza. Służy do weryfikacji, czy klucz nie został zmodyfikowany albo podmieniony.

Pytanie 1.14;

Nie udało się odszyfrować pliku po zmianie jednego znaku w zaszyfrowanej treści. Algorytmy asymetryczne takie jak RSA są odporne na modyfikacje danych, więc uszkodzony szyfrogram skutkuje błędem podczas procesu odszyfrowywania.

Pytanie 1.15;

Dla algorytmu ECC sygnatura jest krótsza niż dla RSA, co wynika z większej efektywności kluczy eliptycznych przy tej samej przyłożonej sile kryptograficznej. Sygnatury ECC są na ogół bardziej wydajne w kontekście zajmowanej przestrzeni i wykonywanych obliczeń.

Pytanie 1.16;

Teoretycznie klucz publiczny może podpisać wiadomość może podpisać wiadomość, ale jest to sprzeczne z założeniami kryptografii asymetrycznej. Wynikiem takiego działania byłoby to, że każdy kto posiada klucz publiczny (czyli każdy, bo klucz jest publiczny) mógłby tworzyć takie podpisy, co prowadziłoby do braku zaufania i możliwości podszywania się pod kogokolwiek. Klucze asymetryczne są jednak matematycznie powiązane, więc możemy technicznie zatrzymać klucz publiczny jako ten „tajny” a udostępnić klucz prywatny i system technicznie będzie działać.

Zad 2. Utwórz CA

```
#
#set_var EASYRSA_REQ_COUNTRY    "PL"
#set_var EASYRSA_REQ_PROVINCE   "Wroclaw"
#set_var EASYRSA_REQ_CITY       "Wroclaw"
#set_var EASYRSA_REQ_ORG        "PWR"
#set_var EASYRSA_REQ_EMAIL      "272644@student.pwr.edu.pl"
#set_var EASYRSA_REQ_OU         "PWR"
```

```
student@student-ubuntu:~/Downloads/easy-rsa-master/easyrsa3$ ./easyrsa init-pki
Using Easy-RSA 'vars' configuration:
* /home/student/Downloads/easy-rsa-master/easyrsa3/vars

Notice
------
'init-pki' complete; you may now create a CA or requests.

Your newly created PKI dir is:
* /home/student/Downloads/easy-rsa-master/easyrsa3/pki

Using Easy-RSA configuration:
* /home/student/Downloads/easy-rsa-master/easyrsa3/vars
student@student-ubuntu:~/Downloads/easy-rsa-master/easyrsa3$
```

```
Using Easy-RSA 'vars' configuration:
* /home/student/Downloads/easy-rsa-master/easyrsa3/vars
..++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*.+......+...+.......+...+......+...+.....+...+....
........+..+....+...+......+......................+.+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*...
.............+.............+.....+.+.........+..+.......+..+..+.....+....+..+.+...+...............+.........+.....+.
+.........+........+...+.......+..+......+.+.........+......+.........+...+....+.....+......+..........+............+.
..+.+..............+...........+..........+........+......+......+.......+....+.......+...+...+.....+.......+...+.
....+...+..........+...+.......+...+...+...+.......+.....+.+..+...............+..........+...+...........+...+..+......
..........+....+...+......+..........+.........+.............+..+.+.........+...+...+...+.........+........+...
.+.........+..+....+.........+..+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Terminal ...+...+......+.....+.+.........+..+...+.....+.+..+......+.....+...+.........+...+.......+++++++++++++++
+++++++++++++++++++++++++++++++++++++++++++++++++*....+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+*..+......+..+.......+...+..+.............+.......+.........+...+..+.+..+...+....+...+..+.+.......+.....+.......+..+....+..+
.+.....+.......+......+............+.....+.........+....+..+...........++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++++++
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Common Name (eg: your user, host, or server name) [Easy-RSA CA]:Aleksander

Notice
------
CA creation complete. Your new CA certificate is at:
* /home/student/Downloads/easy-rsa-master/easyrsa3/pki/ca.crt

Create an OpenVPN TLS-AUTH|TLS-CRYPT-V1 key now: See 'help gen-tls'

Build-ca completed successfully.
```

```
student@student-ubuntu:~/Downloads/easy-rsa-master/easyrsa3$ ./easyrsa gen-crl
Using Easy-RSA 'vars' configuration:
* /home/student/Downloads/easy-rsa-master/easyrsa3/vars
Using configuration from /home/student/Downloads/easy-rsa-master/easyrsa3/pki/cfada623/temp.0.1

Notice
------
An updated CRL DER copy has been created:
* /home/student/Downloads/easy-rsa-master/easyrsa3/pki/crl.der

An updated CRL has been created:
* /home/student/Downloads/easy-rsa-master/easyrsa3/pki/crl.pem
```

Zad 3. Wygeneruj certyfikaty serwera i klienta

```
student@student-ubuntu:~/Downloads/easy-rsa-master/easyrsa3$ ./easyrsa gen-req server nopass
Using Easy-RSA 'vars' configuration:
* /home/student/Downloads/easy-rsa-master/easyrsa3/vars
.....+..++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*...+......+......+....
```

```
Notice
------
Private-Key and Public-Certificate-Request files created.
Your files are:
* req: /home/student/Downloads/easy-rsa-master/easyrsa3/pki/reqs/server.req
* key: /home/student/Downloads/easy-rsa-master/easyrsa3/pki/private/server.key
```

```
Notice
------
Inline file created:
* /home/student/Downloads/easy-rsa-master/easyrsa3/pki/inline/private/server.inline


Notice
------
Certificate created at:
* /home/student/Downloads/easy-rsa-master/easyrsa3/pki/issued/server.crt

student@student-ubuntu:~/Downloads/easy-rsa-master/easyrsa3$
```

```
Notice
------
Private-Key and Public-Certificate-Request files created.
Your files are:
* req: /home/student/Downloads/easy-rsa-master/easyrsa3/pki/reqs/client1.req
* key: /home/student/Downloads/easy-rsa-master/easyrsa3/pki/private/client1.key

student@student-ubuntu:~/Downloads/easy-rsa-master/easyrsa3$
```

```
Notice
------
Inline file created:
* /home/student/Downloads/easy-rsa-master/easyrsa3/pki/inline/private/client1.inline


Notice
------
Certificate created at:
* /home/student/Downloads/easy-rsa-master/easyrsa3/pki/issued/client1.crt

student@student-ubuntu:~/Downloads/easy-rsa-master/easyrsa3$
```

```
DH parameters appear to be ok.

Notice
------

DH parameters of size 2048 created at:
* /home/student/Downloads/easy-rsa-master/easyrsa3/pki/dh.pem

student@student-ubuntu:~/Downloads/easy-rsa-master/easyrsa3$
```

```
 GNU nano 7.2     ta.key
#
# 2048 bit OpenVPN static key
#
-----BEGIN OpenVPN Static key V1-----
e51c62c9ca20e62284eeeff37cf16ffe
cdd0b8edf467c09a78fcf1eb637e74a6
f[illegible]16368433c5487cd7b44ffa2428f
[illegible]e87604b451dc0d3b2cb47518a3e
3dde8a3aede65a33b873d6d1dd8316b2
e97e1d25c4771252aed527c53d83be1f
9d35947a63bd9b1c70114fc38597d8a3
2e151fb8f1d221a62e9a71ec944f9de2
c7c6580cb6e245501bb7f4e2839fc3d7
649def68fdf57eb1ce4f7ee3879cd9ce
5b3e187b69eb67b1d983ba98e9ef4794
a54594697a84e0149a8c8c0ba0a2c077
0794e3d8b17dbabe35a9858e32567483
9a0c438224511c912ba3acc483748e09
aac0711f560f010f0bab9555b5087e59
d355b20116297a0574774c3ed506053d
-----END OpenVPN Static key V1-----
```

Help

Pytanie 3.1;

Algorytm podpisu w certyfikatach zazwyczaj jest określony w polu "Signature Algorithm" i zależy od konfiguracji. Często używane są algorytmy takie jak SHA256 wraz z RSA. W powyższym zadaniu używanym algorytmem podpisu jest *„sha256WithRSAEncryption”.*

Pytanie 3.2;

Algorytm klucza publicznego zależy od klucza użytego podczas generowania certyfikatu. Dla RSA jest to *rsaEncryption*, a dla ECC na przykład: *id-ecPublicKey*. W zadaniu nr. 2 jest to *rsaEncryption.*

Pytanie 3.3;

Easy RSA i większość współczesnych podobnych do niego narzędzi używa standardu X.509 w wersji 3 (X.509v3), co widać na samej górze certyfikatu (*Version: 3 (0x2)*)

Pytanie 3.4;

Z certyfikatu można odczytać między innymi: nazwę podmiotu (subject), nazwę wystawcy (Issuer), daty ważności, numer seryjny, rozszerzenia, długość oraz typ klucza publicznego