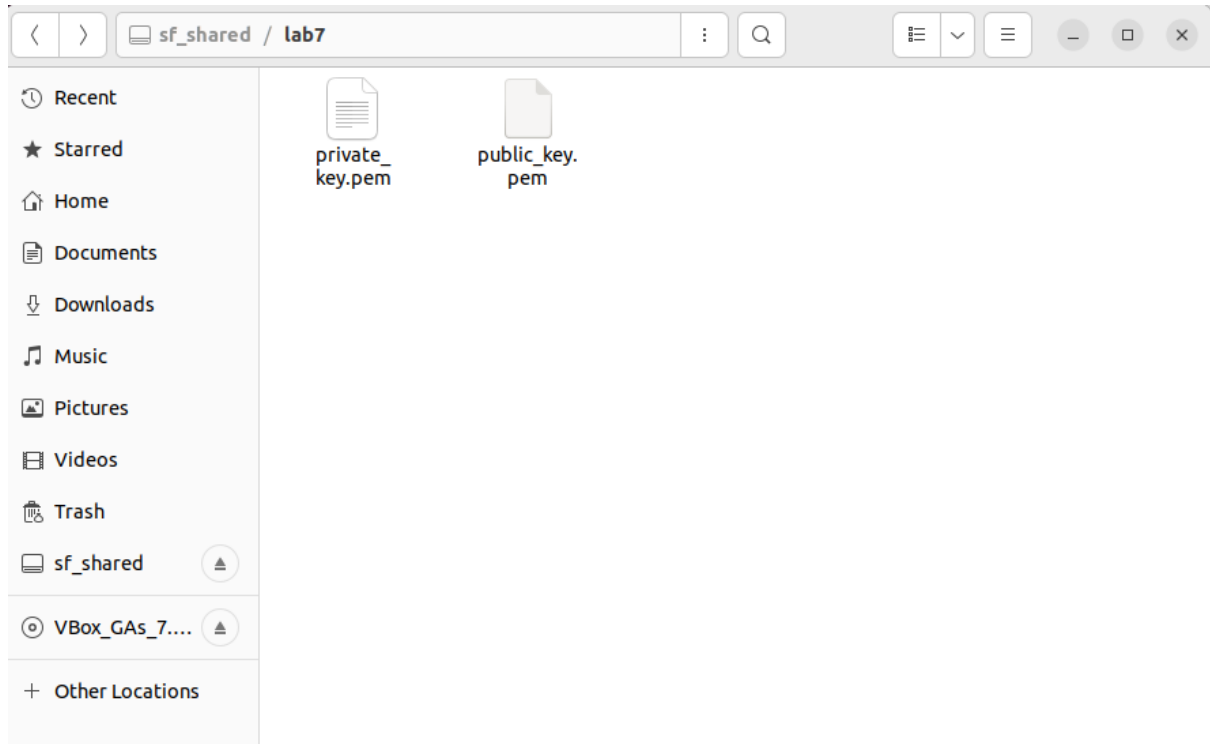


## Task 1.1



### RSA

```
GnuPG needs to construct a user ID to identify your key.

Real name: levi
Name must be at least 5 characters long
Real name: Volodymyr
Email address: v.shepel158@gmail.com
Comment: some comment
You selected this USER-ID:
    "Volodymyr (some comment) <v.shepel158@gmail.com>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? 0
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: key E7F4ECF05AA1FBFF marked as ultimately trusted
gpg: directory '/home/stud/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/home/stud/.gnupg/openpgp-revocs.d/53389A35B893767E829F3596E7F4ECF05AA1FBFF.rev'
public and secret key created and signed.

pub   rsa2048 2023-11-25 [SC]
       53389A35B893767E829F3596E7F4ECF05AA1FBFF
uid    Volodymyr (some comment) <v.shepel158@gmail.com>
sub    rsa2048 2023-11-25 [E]
```

## ECC

```
(13) Existing key
(14) Existing key from card
Your selection? 9
Please select which elliptic curve you want:
(1) Curve 25519
(3) NIST P-256
(4) NIST P-384
(5) NIST P-521
(6) Brainpool P-256
(7) Brainpool P-384
(8) Brainpool P-512
(9) secp256k1
Your selection? 9
Please specify how long the key should be valid.
  0 = key does not expire
<n> = key expires in n days
<n>w = key expires in n weeks
<n>m = key expires in n months
<n>y = key expires in n years
Key is valid for? (0)
Key does not expire at all
Is this correct? (y/N) y

GnuPG needs to construct a user ID to identify your key.

Real name: Volodymyr
Email address: v.shepel158@gmail.com
Comment: another comment
You selected this USER-ID:
  "Volodymyr (another comment) <v.shepel158@gmail.com>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? 0
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: key EB1D8703FBCAAE22 marked as ultimately trusted
gpg: revocation certificate stored as '/home/stud/.gnupg/openpgp-revocs.d/391C32FF3A952DEEBC28D76CEB1D8703FBCAAE22.rev'
public and secret key created and signed.

pub   secp256k1 2023-11-25 [SC]
      391C32FF3A952DEEBC28D76CEB1D8703FBCAAE22
uid           Volodymyr (another comment) <v.shepel158@gmail.com>
sub   secp256k1 2023-11-25 [E]
```

```
stud@ubuntu-srv:/media/sf_shared/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
/home/stud/.gnupg/pubring.kbx
-----
pub   rsa2048 2023-11-25 [SC]
      53389A35B893767E829F3596E7F4ECF05AA1FBFF
uid           [ultimate] Volodymyr (some comment) <v.shepel158@gmail.com>
sub   rsa2048 2023-11-25 [E]

pub   secp256k1 2023-11-25 [SC]
      391C32FF3A952DEEBC28D76CEB1D8703FBCAAE22
uid           [ultimate] Volodymyr (another comment) <v.shepel158@gmail.com>
sub   secp256k1 2023-11-25 [E]
```

## Task 1.2

```
stud@ubuntu-srv:/media/sf_shared/lab7$ ls -a
.  ..  private_key.pem  public_key.asc  public_key.pem
stud@ubuntu-srv:/media/sf_shared/lab7$
```

## Task 1.3

```
levi@ubuntu-srv:~$ cd /home/levi/lab7
levi@ubuntu-srv:~/lab7$ ls -a
.  ..  public_key.asc  public_key.pem
levi@ubuntu-srv:~/lab7$
```

## Task 1.4

```
stud@ubuntu-srv:/media/sf_shared/lab7$ gpg --fingerprint
/home/stud/.gnupg/pubring.kbx
-----
pub   rsa2048 2023-11-25 [SC]
      5338 9A35 B893 767E 829F  3596 E7F4 ECF0 5AA1 FBFF
uid           [ultimate] Volodymyr (some comment) <v.shepel158@gmail.com>
sub   rsa2048 2023-11-25 [E]

pub   secp256k1 2023-11-25 [SC]
      391C 32FF 3A95 2DEE BC28  D76C EB1D 8703 FBCA AE22
uid           [ultimate] Volodymyr (another comment) <v.shepel158@gmail.com>
sub   secp256k1 2023-11-25 [E]
```

```
levi@ubuntu-srv: ~/labka7
levi@ubuntu-srv:~/labka7$ gpg --import public_key.asc
gpg: /home/levi/.gnupg/trustdb.gpg: trustdb created
gpg: key E7F4ECF05AA1FBFF: public key "Volodymyr (some comment) <v.shepel158@gmail.com>" imported
gpg: key EB1D8703FBCAAE22: public key "Volodymyr (another comment) <v.shepel158@gmail.com>" imported
gpg: Total number processed: 2
gpg:         imported: 2
levi@ubuntu-srv:~/labka7$
```

```
stud@ubuntu-srv: /media/sf_shared/lab7
stud@ubuntu-srv:/media/sf_shared/lab7$ gpg --fingerpring
invalid option "--fingerpring"
stud@ubuntu-srv:/media/sf_shared/lab7$ gpg --fingerprint
/home/stud/.gnupg/pubring.kbx
-----
pub   rsa2048 2023-11-25 [SC]
      5338 9A35 B893 767E 829F  3596 E7F4 ECF0 5AA1 FBFF
uid   [ultimate] Volodymyr (some comment) <v.shepel158@gmail.com>
sub   rsa2048 2023-11-25 [E]

pub   secp256k1 2023-11-25 [SC]
      391C 32FF 3A95 2DEE BC28  D76C EB1D 8703 FBCA AE22
uid   [ultimate] Volodymyr (another comment) <v.shepel158@gmail.com>
sub   secp256k1 2023-11-25 [E]

stud@ubuntu-srv:/media/sf_shared/lab7$ █

levi@ubuntu-srv: ~/labka7
levi@ubuntu-srv:~/labka7$ gpg --fingerprint "Volodymyr (some comment)"
pub   rsa2048 2023-11-25 [SC]
      5338 9A35 B893 767E 829F  3596 E7F4 ECF0 5AA1 FBFF
uid   [ unknown] Volodymyr (some comment) <v.shepel158@gmail.com>
sub   rsa2048 2023-11-25 [E]

levi@ubuntu-srv:~/labka7$
```

## Task 1.5

```
stud@ubuntu-srv: /media/sf_shared/lab7
stud@ubuntu-srv:/media/sf_shared/lab7$ ls -a
.  ..  data.txt  private_key.pem  public_key.asc  public_key.pem
stud@ubuntu-srv:/media/sf_shared/lab7$ openssl dgst -sha256 -sign private_key.p
m -out signature.sha256 data.txt
stud@ubuntu-srv:/media/sf_shared/lab7$ openssl dgst -sha256 -sign private_key.p
m -out signature.sha256 data.txt
stud@ubuntu-srv:/media/sf_shared/lab7$ openssl dgst -sha256 -verify public_key.p
em -signature signature.sha256 data.txt
Verified OK
stud@ubuntu-srv:/media/sf_shared/lab7$
```

```
stud@ubuntu-srv:/media/sf_shared/lab7$ gpg --sign -u 391C32FF3A952DEEBC28D76CEB1
D8703FBCAAE22 data.txt
stud@ubuntu-srv:/media/sf_shared/lab7$ ls -a
.  data.txt  private_key.pem  public_key.pem
.. data.txt.gpg  public_key.asc  signature.sha256
stud@ubuntu-srv:/media/sf_shared/lab7$ gpg --verify data.txt.gpg
gpg: Signature made Sat 25 Nov 2023 04:26:20 PM UTC
gpg: using ECDSA key 391C32FF3A952DEEBC28D76CEB1D8703FBCAAE22
gpg: Good signature from "Volodymyr (another comment) <v.shepel158@gmail.com>" [
ultimate]
stud@ubuntu-srv:/media/sf_shared/lab7$
```

## Task 1.6

```
levi@ubuntu-srv:~/labka7$ openssl dgst -sha256 -verify public_key.pem -signature signature.sha256 data.txt
Verified OK
levi@ubuntu-srv:~/labka7$ gpg --verify data.txt.gpg
gpg: Signature made Sat 25 Nov 2023 04:26:20 PM UTC
gpg: using ECDSA key 391C32FF3A952DEEBC28D76CEB1D8703FBCAAE22
gpg: Good signature from "Volodymyr (another comment) <v.shepel158@gmail.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg: There is no indication that the signature belongs to the owner.
Primary key fingerprint: 391C 32FF 3A95 2DEE BC28 D76C EB1D 8703 FBCA AE22
levi@ubuntu-srv:~/labka7$
```

## Task 1.7

```
levi@ubuntu-srv: ~/labka7
levi@ubuntu-srv:~/labka7$ sudo echo "This is the new content I'm adding." >> data.txt
levi@ubuntu-srv:~/labka7$ openssl dgst -sha256 -verify public_key.pem -signature signature
.sha256 data.txt
Verification failure
4077F905DF7F0000:error:02000068:rsa routines:ossl_rsa_verify:bad signature:../crypto/rsa/rsa_sign.c:430:
4077F905DF7F0000:error:1C880004:Provider routines:rsa_verify:RSA lib:../providers/implementations/signature/rsa_sig.c:774:
levi@ubuntu-srv:~/labka7$ gpg --verify data.txt.gpg
gpg: Signature made Sat 25 Nov 2023 04:26:20 PM UTC
gpg: using ECDSA key 391C32FF3A952DEEBC28D76CEB1D8703FBCAAE22
gpg: Good signature from "Volodymyr (another comment) <v.shepel158@gmail.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg: There is no indication that the signature belongs to the owner.
Primary key fingerprint: 391C 32FF 3A95 2DEE BC28 D76C EB1D 8703 FBCA AE22
levi@ubuntu-srv:~/labka7$
```

## Task 1.8

One character has been changed

```
sf_shared / lab7
stud@ubuntu-srv: /media/sf_shared/lab7
stud@ubuntu-srv:/media/sf_shared/lab7$ gpg --encrypt --recipient 391C32FF3A952DEEBC28D76CEB1D8703FBCAAE22 --output encrypted_file.gpg data.txt
stud@ubuntu-srv:/media/sf_shared/lab7$ ls -a
. data.txt encrypted_file.gpg public_key.asc signature.sha256
.. data.txt.gpg private_key.pem public_key.pem
stud@ubuntu-srv:/media/sf_shared/lab7$ gpg --decrypt --output decrypted_data.txt encrypted_file.gpg
gpg: encrypted with 256-bit ECDH key, ID 2F76F619F53383D6, created 2023-11-25
"Volodymyr (another comment) <v.shepel158@gmail.com>"
stud@ubuntu-srv:/media/sf_shared/lab7$ ls -a
. data.txt decrypted_data.txt private_key.pem public_key.pem
.. data.txt.gpg encrypted_file.gpg public_key.asc signature.sha256
stud@ubuntu-srv:/media/sf_shared/lab7$ cat decrypted_data.txt
Some content of text file.
stud@ubuntu-srv:/media/sf_shared/lab7$
stud@ubuntu-srv:/media/sf_shared/lab7$ gpg --decrypt --output decrypted_data.txt encrypted_file.gpg
gpg: packet(1) with unknown version 75
```

## Task 1.9

In asymmetric cryptography, a key pair consists of a public key and a private key, serving distinct purposes. While the public key is openly shared, the private key is kept confidential.

It is technically feasible to generate only one part of the key pair—either the private or public key. However, this practice is generally discouraged as the security and functionality of asymmetric cryptography hinge on the intricate interdependence of both keys.

The effectiveness of asymmetric cryptography relies on the inherent relationship between the public and private keys. The public key is employed for data encryption or signature verification, while the private key is utilized for decryption or signing. If only one key is generated, the advantages of this paired relationship are forfeited.

Security is compromised when only the private or public key is generated, as it becomes easier for an adversary to deduce the missing key. Both keys are integral to maintaining the robust security of the system.

In standard use cases of asymmetric cryptography, such as public-key encryption, having both keys is essential. For instance, data encrypted with the public key can only be decrypted with the corresponding private key, and vice versa.

## Task 1.10

A GPG (GNU Privacy Guard) key typically consists of two components: the public key and the private key. These keys serve various purposes, including encryption, decryption, and digital signatures.

**Public Key:** The public key is commonly presented in ASCII-armoured format, often denoted by a **.asc** or **.gpg** file extension. This key is intended for public sharing and allows others to encrypt messages that only the possessor of the corresponding private key can decipher.

**Private Key:** Encrypted in ASCII-armoured format, the private key is kept confidential. Safeguarding the private key is crucial, as it is utilized to decrypt messages encrypted with the corresponding public key and to generate digital signatures.

Now, let's delve into the distinction between a PEM key and an OpenPGP key:

**PEM Key:** PEM (Privacy Enhanced Mail) serves as a prevalent encoding format for binary data, including cryptographic keys. PEM keys can adopt various formats like RSA or ECDSA and are often employed for diverse cryptographic purposes beyond PGP.

**OpenPGP Key:** OpenPGP establishes a standard defining formats for public and private keys used in PGP. OpenPGP keys are specifically designed for PGP applications like GPG. While the OpenPGP public key mirrors a PEM key by being ASCII-armoured for public sharing, OpenPGP private keys are



also ASCII-armoured and secured with a passphrase for enhanced security. Common file extensions for OpenPGP keys include **.asc** or **.gpg**.

### Task 1.11

Exchanging private keys is generally not justified and is strongly discouraged in the context of secure cryptographic practices. The security of asymmetric cryptography heavily relies on maintaining the confidentiality of private keys. The private key is intended to be known exclusively to its owner and should never be openly shared or exchanged.

Several reasons highlight why the exchange of private keys is not justified:

**Security Risk:** Sharing private keys poses a significant security risk. The strength of asymmetric cryptography is built on the premise that the private key remains secret. If the private key is shared, individuals with access to it can decrypt messages or forge digital signatures, jeopardizing the overall security of the system.

**Loss of Control:** Private keys are designed to be under the sole control of the key owner. Sharing them means surrendering control, potentially leading to unauthorized use and security breaches.

**Violation of Trust Model:** The trust model in asymmetric cryptography is constructed on the principle that private keys are kept confidential. Exchanging private keys contradicts this trust model and can undermine the fundamental basis of secure communication.

**Alternative Solutions:** Secure communication protocols provide established methods for exchanging public keys securely without the necessity to exchange private keys. Public keys can be freely shared, enabling others to encrypt messages, while private keys remain safeguarded.

### Task 1.12

In the realm of cryptography, a fingerprint serves as a succinct and distinct representation of a cryptographic key. It is a fixed-size character string generated through the application of a hash function to the public key. This fingerprint acts as a unique identifier for the key and is widely employed for verification and validation purposes.

Functioning as a concise identifier, the fingerprint allows users to verify the integrity and authenticity of a public key without the need to scrutinize the entire key. By sharing and comparing fingerprints, users can ensure that the received public key is both the intended and unaltered key.



For instance, in the context of PGP (Pretty Good Privacy) or GPG (GNU Privacy Guard), the fingerprint is a character sequence derived from the public key. When users exchange public keys, presenting or verifying fingerprints becomes a common practice to affirm the possession of the correct and unaltered cryptographic key. This verification process significantly bolsters security by mitigating the risk of utilizing compromised or falsified keys.

## Task 1.13

The integrity of digital signatures was scrutinized following the modification of the original file, data.txt.

### Using OpenSSL for verification

The verification process failed as anticipated. Modifying the file altered its hash, rendering the original signature invalid.

### GPG Verification on Another Machine/Account:

On the machine/account where the signature was created, modifying the content of the original file was repeated. Surprisingly, when verifying the signature using GPG:

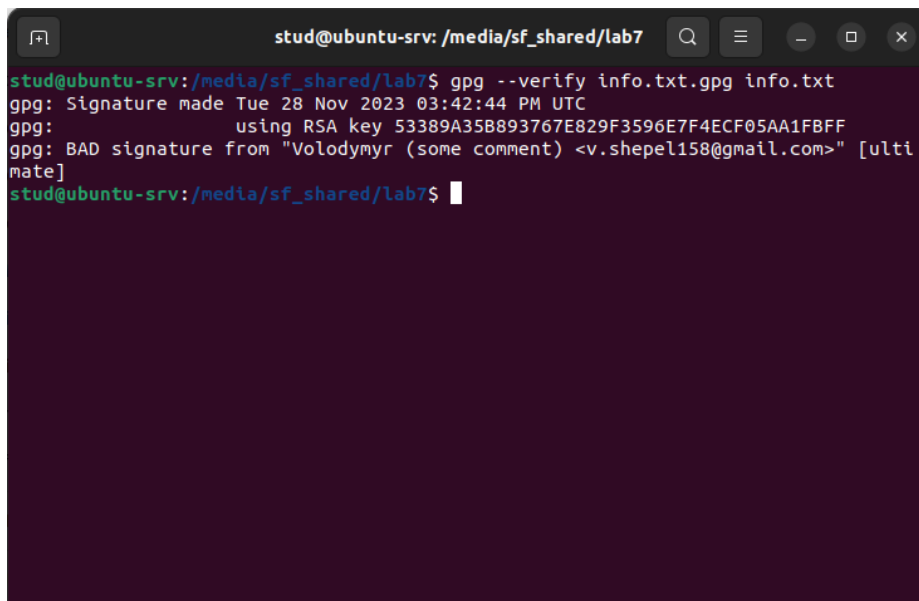
```
gpg --verify data.txt.gpg
```

The output indicated a "Good signature" even after the content modification:

```
gpg: Good signature from "Volodymyr (another comment) <v.shepel158@gmail.com>" [ultimate]
```

Explanation : while **gpg --sign** provides authentication of the source, it does not protect against subsequent changes to the document. For stronger integrity checks, especially in scenarios where data tampering is a concern, using **gpg --detach-sign** is recommended, which would fail verification if the file is modified after signing.

After modifying the content of the file(signed with gpg –detach sign)

A terminal window titled 'stud@ubuntu-srv: /media/sf\_shared/lab7' showing a gpg verification command and its output. The command is 'gpg --verify info.txt.gpg info.txt'. The output shows the signature was made on Tue 28 Nov 2023 03:42:44 PM UTC using RSA key 53389A35B893767E829F3596E7F4ECF05AA1FBFF, but it is a BAD signature from 'Volodymyr (some comment) <v.shepel158@gmail.com>' [ultimate].

```
stud@ubuntu-srv:/media/sf_shared/lab7$ gpg --verify info.txt.gpg info.txt
gpg: Signature made Tue 28 Nov 2023 03:42:44 PM UTC
gpg:                using RSA key 53389A35B893767E829F3596E7F4ECF05AA1FBFF
gpg: BAD signature from "Volodymyr (some comment) <v.shepel158@gmail.com>" [ultimate]
stud@ubuntu-srv:/media/sf_shared/lab7$
```

## Task 1.14

The decryption attempt yielded the following error message:

**gpg: packet (1) with unknown version 75**

This error suggests an issue with the GPG packet format or version during the decryption process. When a single character in the encrypted file is modified, the encrypted data's structure is disrupted. Encryption algorithms operate on data blocks, and even minor alterations can lead to inconsistencies in the entire file.

The observed error underscores the sensitivity of encrypted data to modifications. Encryption algorithms rely on specific formats and versions for successful decryption. Intentional changes to encrypted content should follow established procedures, such as decrypting the file, making modifications, and then encrypting it again.

To ensure successful decryption, it is crucial to maintain the integrity of the encrypted data. Any modifications, even minor ones, can result in decryption errors.

## Task 1.15

The distinction between the signature schemes employed by Elliptic Curve Cryptography (ECC) and Rivest-Shamir-Adleman (RSA) in OpenPGP lies chiefly in the underlying algorithms and the length of cryptographic keys. Here's a concise overview:

Algorithmic Variations:

**RSA:**

- RSA operates as an asymmetric cryptographic algorithm utilizing a public key for encryption and a private key for decryption.
- Its security hinges on the challenge of factoring the product of two large prime numbers.
- RSA signatures involve modular exponentiation with the private key.

**ECC:**

- ECC is another asymmetric cryptographic algorithm grounded in the mathematics of elliptic curves over finite fields.
- While offering equivalent security to traditional methods, ECC employs shorter key lengths, enhancing computational efficiency.
- ECC signatures involve operations on elliptic curve points and are computationally more efficient than RSA.

**Signature Format:**

The precise details of the signature format, although often standardized (e.g., in RFC 4880 for OpenPGP), differ for RSA and ECC, reflecting their distinct mathematical principles and algorithms.

**Efficiency:**

ECC signatures are generally acknowledged as more computationally efficient than RSA signatures for equivalent security levels, particularly advantageous in resource-constrained environments.

**Key Management:**

ECC keys, being shorter, may simplify key management. However, the selection between ECC and RSA depends on compatibility and the security requirements of the specific use case.

In summary, the primary distinctions between ECC and RSA signatures in OpenPGP relate to the underlying mathematical principles, key lengths, computational efficiency, and the specific algorithms used for signing and verification. The choice between ECC and RSA often involves a trade-off between security and computational efficiency.

## Task 1.15

Signing a message with a public key diverges from the customary norms in public-key cryptography. Typically, messages are signed using a private key and validated with the corresponding public key, with the private key held confidentially and the public key openly shared.

Engaging in the unconventional practice of signing messages with a public key can entail various consequences:

### **Security Implications:**

Using a public key for message signing may jeopardize the security of the cryptographic system. Public keys are designed for verification, and the corresponding private key is intended to remain confidential. If a public key is employed for signing, it might expose crucial information necessary for authenticating the messages.

### **Authentication Issues:**

Public-key cryptography relies on the principle that only the holder of the private key can produce a valid signature. Should messages be signed with a public key, it could introduce confusion into the authentication process. There is a risk of others mistakenly assuming that messages signed with a public key are authentic, despite public keys not being kept secret.

### **Violation of Cryptographic Principles:**

The established cryptographic practices advocate the use of private keys for signing and public keys for verification. Departing from this recognized framework may contravene the principles and assumptions upon which the security of public-key cryptography is founded.

## TASK 2

```
stud@ubuntu-srv:/media/sf_shared/lab7/EasyRSA-3.1.6$ ./easyrsa init-pki
```

## Notice

.....

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

```
Your newly created PKI dir is:
```

```
* /media/sf_shared/lab7/EasyRSA-3.1.6/pki
```

### Using Easy-RSA configuration:

```
* /media/sf_shared/lab7/EasyRSA-3.1.6/vars
```

**IMPORTANT:**

The preferred location for 'vars' is within the PKI folder.

```
To silence this message move your 'vars' file to your PKI
```

or declare your 'vars' file with option: `--vars=<FILE>`

stud@ubuntu-srv: /media/sf\_shared/lab7/EasyRSA-3.1.6

```

+++++
.....+.+......+......+......+......+......+......+......+.
.....+......+......+......+......+......+......+......+......+.
.....+......+......+++++
+++++*.....+......+......+......+......+......+++++
+++++*.....+......+......+......+......+......+.
+......+......+......+......+......+++++
+++++

```

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.

100 100 100 100 100

```
Common Name (eg: your user, host, or server name) [Easy-RSA CA]:levi
```

## Notice

CA creation complete. Your new CA certificate is at:

```
* /media/sf shared/lab7/EasyRSA-3.1.6/pki/ca.crt
```

```
stud@ubuntu-srv:/media/sf_shared/lab7/EasyRSA-3.1.6$
```











```
stud@ubuntu-srv:/media/sf_shared/lab7/EasyRSA-3.1.6$ openssl x509 -in pki/issued/client1.crt -text
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      77:90:f9:3b:ad:14:ce:fe:6f:23:92:61:f9:76:f6:76
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: CN = levi
    Validity
      Not Before: Nov 28 08:47:44 2023 GMT
      Not After : Mar  2 08:47:44 2026 GMT
    Subject: CN = bleach
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      Public-Key: (2048 bit)
      Modulus:
        00:9c:a0:28:28:dc:35:04:a3:bf:42:29:10:ea:e2:
        56:24:14:05:75:6c:fb:4d:ea:92:a5:64:e3:f7:aa:
        b4:a1:dd:60:58:c8:7b:50:dc:7c:f9:c0:5b:2d:6b:
        73:61:f8:c3:da:22:dd:86:d5:c0:b7:32:c0:63:49:
        7d:45:1c:50:95:4d:d9:9b:0a:05:30:d4:9e:e2:bf:
        31:93:ab:2b:c2:4b:e5:f0:9b:a4:ad:85:92:b8:5a:
        a7:e9:fd:d0:53:b0:44:33:2c:02:72:49:44:c7:bd:
        4e:80:60:ab:75:2b:bc:c3:2e:32:e5:54:9e:35:9a:
        3d:89:ae:ce:60:82:08:0d:9a:7e:47:ec:9c:1f:1e:
        ae:a8:c4:39:92:bb:4f:e5:3c:72:7e:51:e3:df:e2:
        26:08:5c:a2:47:dc:40:e6:13:59:32:cc:f8:bc:88:
        63:e0:45:e3:7e:30:f2:35:db:95:d4:95:3f:e0:67:
        ef:65:c6:99:16:6d:f5:3f:c3:da:0c:37:c0:8a:38:
        15:3a:20:76:ce:d8:f3:99:fb:e8:72:70:fe:8a:5f:
        8d:e5:07:70:c4:e5:1f:d5:1c:d9:03:11:25:79:34:
        70:9d:b2:f8:dc:3f:b4:da:84:2b:9c:bb:6d:2e:0c:
        01:13:80:63:56:60:0d:97:d0:f0:24:3e:f4:20:a8:
        ba:b9
      Exponent: 65537 (0x10001)
    X509v3 extensions:
      X509v3 Basic Constraints:
        CA:FALSE
      X509v3 Subject Key Identifier:
        3F:DB:1E:6B:82:E8:B7:9C:6D:D5:AF:3C:13:DE:B0:8D:13:99:8B:FA
      X509v3 Authority Key Identifier:
        keyid:40:8A:5F:13:CC:75:FA:3A:59:EA:61:C9:3C:08:DC:B9:5D:5E:B8:B7
        DirName:/CN=levi
        serial:71:CE:61:3E:12:7F:07:A6:DF:8C:DB:69:1C:0E:EB:00:38:80:B7:57
      X509v3 Extended Key Usage:
        TLS Web Client Authentication
      X509v3 Key Usage:
        Digital Signature
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
      81:4c:42:80:63:10:6f:4f:0a:84:3a:77:05:35:7f:62:60:6c:
```

Signature Algorithm: sha256WithRSAEncryption

Signature Value:

81:4c:42:89:63:19:6f:4f:9e:84:3a:77:05:35:7f:62:60:6c:  
09:ea:a4:87:3a:71:e3:8f:e9:50:4c:ad:c8:51:f1:14:f3:d0:  
77:0c:5c:73:49:43:cb:14:d8:b5:0e:57:f6:20:1f:83:7a:d1:  
c4:4d:10:29:f1:16:28:99:a9:a3:d2:6b:cd:89:5a:a9:aa:ce:  
78:a2:8d:4f:dc:47:88:ba:6a:8e:7a:76:63:9a:bd:b8:5e:88:  
6c:ff:74:d6:d6:cb:1c:3c:fe:40:5e:61:65:58:63:cd:a0:70:  
49:66:3c:c1:be:07:41:f3:b5:c5:5f:30:40:d1:fb:4e:8a:4b:  
ad:9f:6c:86:48:5f:73:6a:e3:22:e8:7a:a9:55:38:8b:ef:9d:  
8c:36:76:3d:88:57:31:f5:64:22:49:e4:3e:72:d3:dc:2b:74:  
2e:be:70:3e:32:86:33:f7:35:56:8b:80:ba:b9:31:6e:5a:2c:  
92:4e:1d:7e:11:fa:fe:0f:45:20:cf:2d:b4:fb:f3:2f:3e:74:  
ac:78:86:69:37:af:de:fb:9c:d1:30:57:7e:da:47:66:72:05:  
46:a1:df:b0:ad:f0:e1:53:4e:d9:54:f4:b2:68:e5:01:30:d6:  
bc:5b:36:20:53:28:c7:6b:f8:7f:a8:bb:c6:2b:4f:b5:e7:9f:  
93:40:10:3a

-----BEGIN CERTIFICATE-----

MIIDRjCCAi6gAwIBAgIQd5D5060Uzv5vI5Jh+Xb2djANBgkqhkiG9w0BAQsFADAP  
MQ0wCwYDVQQDDARsZXZpMB4XDTIzMTYyODc0NFoXDTI2MDMwMjA4NDc0NFow  
ETEPMA0GA1UEAwYGmXlYWN0MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKC  
AQEAnKAoKNw1BK0/QikQ6uJWJBQFdw77TeqSpWTj96q0od1gWMh7UNx8+cBbLwtz  
Yfjd2iLdhtXAtzLAY0l9RRxQlU3ZmwoFMNSe4r8xk6srwkv18JukrYWSuFqn6f3Q  
U7BEMyWCcklEx710gGCrdSu8wy4y5VSeNZo9ia70YIIIDZp+R+ycHx6uqMQ5krTP  
STxyflHj3+ImCFyiR9xA5hNZMsZ4vIhj4EXjffjDyNduV1JU/4GfvZcaZFm31P8Pa  
DDfAiJgVOiB2ztjzmfvocnD+il+N5QdwxOUf1RzZAXeLeTRwnbL43D+02oQrnLtt  
LgwBE4BjVmANl9DwJD70IKi6uQIDAQABo4GbMIGYMAkGA1UdEwQCMAAHQYDVRR00  
BBYEFD/bHmuC6LecbdwvPBPeSI0TmYv6MEoGA1UdIwRDMGAFECKXxPMdf06Weph  
yTwI3LldXri3oROkETAPMQ0wCwYDVQQDDARsZXZpghRxxzmE+En8Hpt+M22kcDusa  
OIC3VzATBgNVHSUEDDAKBggrBgEFBQcDAjALBgNVHQ8EBAMCB4AwDQYJKoZIhvcN  
AQELBQADggEBAIFMQoljGW9PnoQ6dwU1f2JgbAnqpIc6ceOP6VBMrchR8RTz0HcM  
XHNJQ8sU2LU0V/YgH4N60cRNECnxFiiZqaPSa82JWmqzniiJU/cR4i6ao56dm0a  
vbheiGz/dNbWyw8/kBeYWVYY82gcElmPMG+B0HztcVfMEDR+06KS62fbIZIX3Nq  
4yLoeqLVOIvvnYw2dj2IVzH1ZCJJ5D5y09wrdC6+cD4yhjP3NVaLgQ5MW5aLJJ0  
HX4R+v4PRSDPLbT78y8+dKx4hmk3r977nNEwV37aR2ZyBUah37Ct80FTTtLU9LJo  
5QEw1rxbnIBTKMdr+H+ou8YrT7Xnn5NAEDo=

-----END CERTIFICATE-----

stud@ubuntu-srv:/media/sf\_shared/lab7/EasyRSA-3.1.6\$

### TASK 3.1-3.4

**Signature Algorithm** : sha256WithRsaEncryption

**Public Key Algorithm**: rsaEncryption

**Version of the X.509**: v3

**Extra information which can be read from the certificate:**

- **Issuer:**
  - Issuer: CN = levi indicates the entity that issued the certificate (in this case, "levi").
- **Validity:**
  - Not Before: Nov 28 08:47:44 2023 GMT and Not After : Mar 2 08:47:44 2026 GMT specify the period during which the certificate is valid.
- **Serial Number:**
  - Serial Number: 77:90:f9:3b:ad:14:ce:fe:6f:23:92:61:f9:76:f6:76 is a unique identifier assigned by the CA to the certificate.
- **Subject:**
  - Subject: CN = bleach indicates the entity (often a server or client) to which the certificate is associated.
- **X509v3 extensions:**
  - Basic Constraints: Indicates whether the certificate is a CA certificate.
  - Subject Key Identifier and Authority Key Identifier: Provide identifiers for the subject and issuer public keys.
  - Extended Key Usage: Specifies the purpose for which the public key may be used.
  - Key Usage: Specifies the key usage restrictions.
- **Subject Public Key Info:**
  - Public Key Algorithm: rsaEncryption indicates the algorithm used for the public key.
  - Public-Key: (2048 bit) specifies the size of the public key (2048 bits).
  - Exponent: 65537 (0x10001) is the public exponent.
  - Modulus: ... is the actual public key modulus.