# Task 1.1

A screenshot of a computer

Description automatically generated

RSA

A screenshot of a computer program

Description automatically generated

ECC

A screenshot of a computer program

Description automatically generated

A computer screen shot of white text

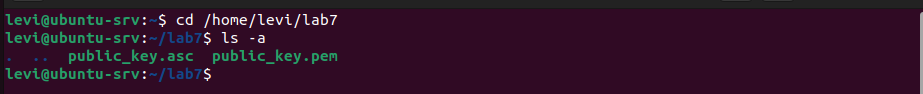
Description automatically generated

# Task 1.2

A computer screen shot of text

Description automatically generated

# Task 1.3



# Task 1.4

A computer screen shot of white text

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# Task 1.5

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

# Task 1.6

A computer screen with white text

Description automatically generated

# Task 1.7

A screenshot of a computer

Description automatically generated

# Task 1.8

One character has been changed

A screenshot of a computer screen

Description automatically generated



# 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 screenshot of a computer

Description automatically generated

# 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

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

# TASK 3

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A computer screen shot of a computer code

Description automatically generated

A screenshot of a computer code

Description automatically generated

## 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.