Lab 4

Encrypted data over TCP Socket

Note: Mac users can do WSL experiments on their own machines
and work with other classmates for Windows related activities or
If you have any VM installed try the same between Mac and the VM.
Can also try connecting from one machine to the other which are on the same subnet.

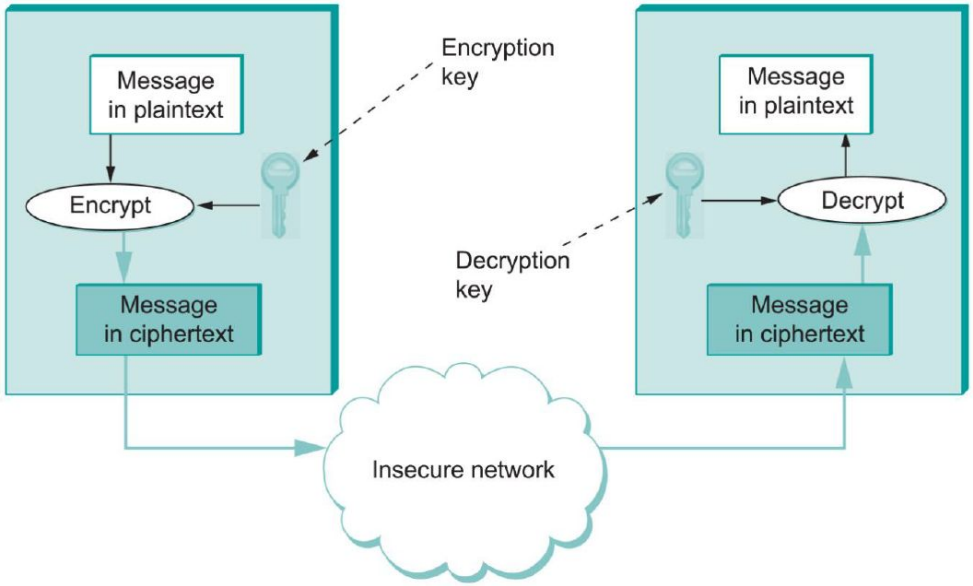
Mouli Sankaran

Lab 4: Focus

- **Symmetric Key Ciphers**
  - **OpenSSL** Introduction
  - Library Installations on WSL and Windows
  - **Exp 1**: Server in C - WSL – Client in Python – Win
- **Asymmetric Key Ciphers**
  - Explanation about the **pem** format
  - **Exp 2**: Server in C - WSL – Client in Python – Win

Symmetric Key Ciphers
(secret-key)

Secret-key Encryption and Decryption



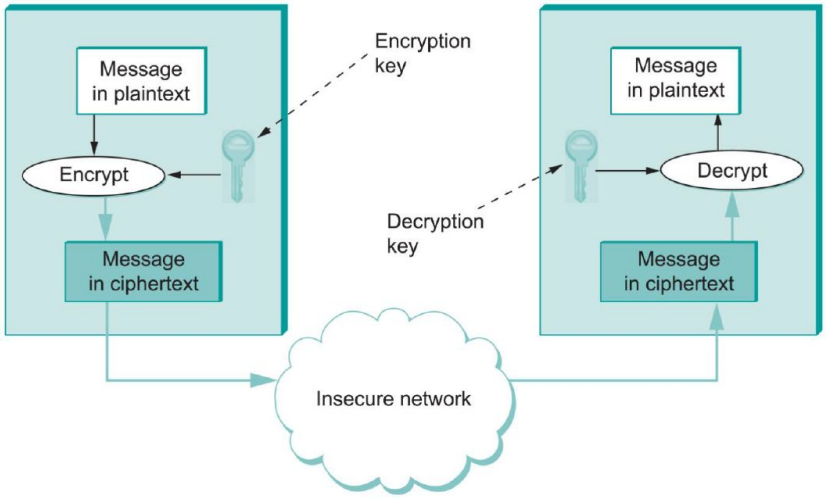
Plain text is the original, unencrypted information or data.

Cipher text is the result of applying an encryption algorithm to the plain text; it appears scrambled or unreadable without the appropriate decryption key.

- The **transformation** represented by an **encryption** function and its corresponding **decryption** function is called a **cipher**.
- Encryption and decryption functions have to be parameterized by a key and the functions are considered to be public knowledge—only the key needs to be secret.
- The **ciphertext** produced for a given **plaintext** message depends on both the **encryption function** and the **key**.

Secret-key Encryption and Decryption

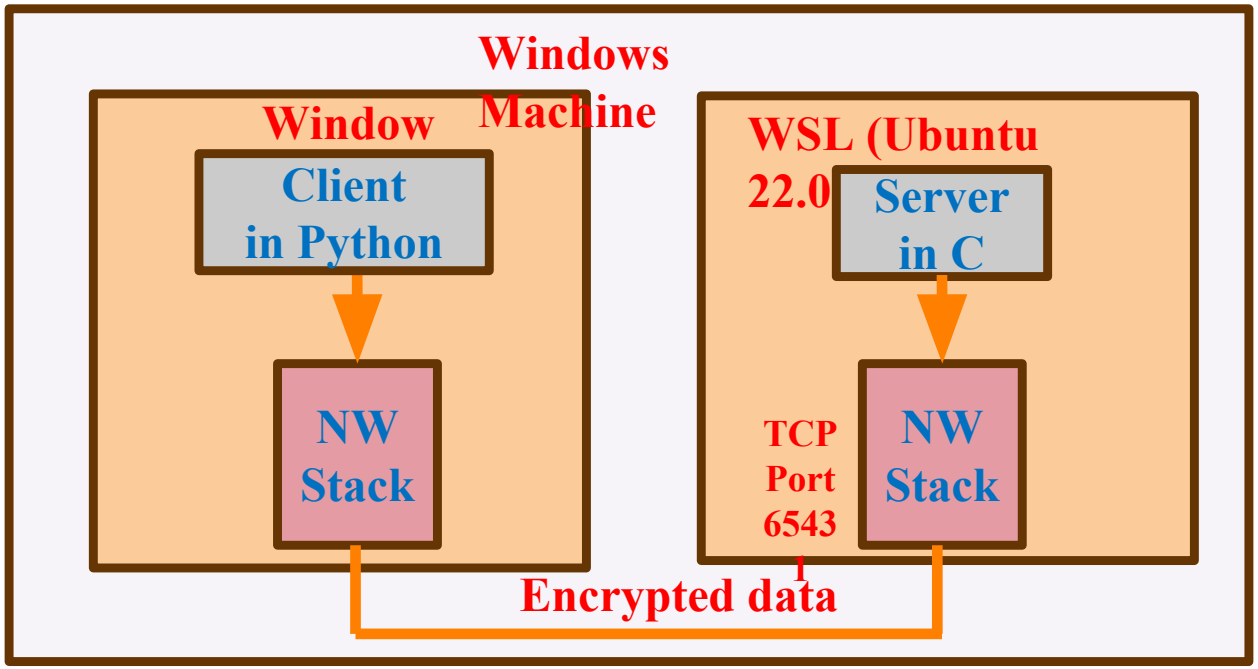
- Here, **both participants** in a communication **share** the **same key**.
- **Secret-key** ciphers are also known as **symmetric-key** ciphers.
- Advanced Encryption Standard (**AES**) standard issued by NIST.



- AES supports key lengths of 128, 192, or 256 bits, and the block length is 128 bits. AES permits fast implementations in both software and hardware.
- It does not require much memory, which makes it suitable for small mobile devices.
- It is used to securely encrypt data in fixed-size blocks (often 128 bits).
- AES has some mathematically proven security properties and, has not suffered from any significant successful attacks.

Exp 1 – WSL Server – Win Client (symmetric key)

Exp 1: System Diagram: Symmetric keys (code shared)



Server code:

// For AES-128, key and IV are 16 bytes.
unsigned char key[16] = "0123456789abcdef";
unsigned char iv[16] = "abcdef9876543210";

Client code:

# Same key and IV as in Server written in C
key = b"0123456789abcdef"
iv = b"abcdef9876543210"

IV: Initialization Vector

Open SSL: An Introduction Ref: OpenSSL

- **OpenSSL** is an open-source software library that implements the Secure Sockets Layer (**SSL**) and Transport Layer Security (**TLS**) protocols
  - OpenSSL enables secure communication between clients and servers.
  - It's used by many web servers, including those that use HTTPS.
  - It generates and manages certificates that verify the identity of websites and servers
  - It encrypts data using symmetric/asymmetric encryption algorithms
  - It implements basic cryptographic functions and provides various utility functions
- **SSL** was the first widely adopted protocol to secure Internet, developed by Netscape in mid-1990s.
  - Due to vulnerabilities and design issues, SSL has been largely phased out in favor of its successor TLS
- **TLS** offers stronger cryptographic algorithms and better security practices
  - TLD is used for securing web traffic (HTTPS), email, VoIP, and many other applications.

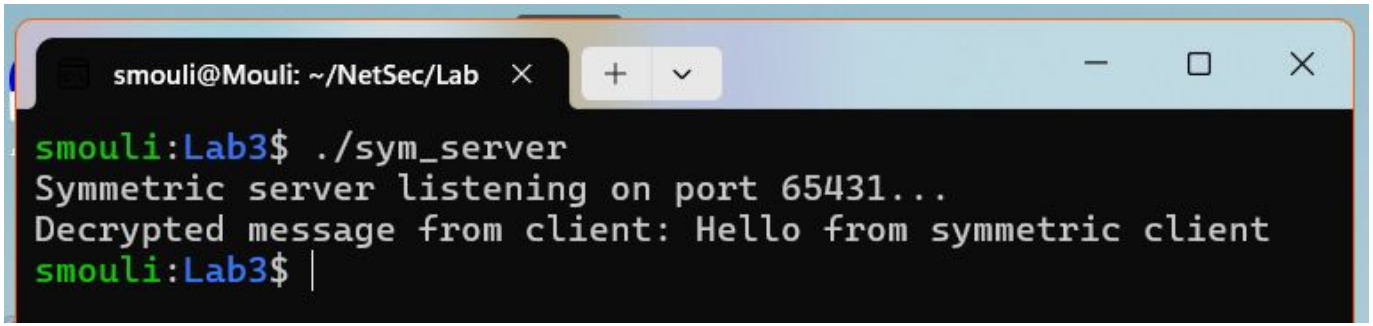
Pre-requisite: Library Installations

- **On WSL (Ubuntu):** using the Terminal:
  - For running the Server program (in C) on WSL do the following:
  - ***sudo apt-get update***
  -
  - **Building the server executable on WSL**
  - ***gcc Lab4_Exp1_Sym_Server_WSL.c -lssl -lcrypto -o sym_server -Wno-deprecated-declarations***
  - *-l* links dynamic (shared object) libraries libssl.so and libcrypto.so.
  - *-W* for suppressing deprecated declarations while compiling
- **On Windows:** using the Command Prompt:
  - For running the Clien program (in Python) on Windows do the following:
  - ***pip install cryptography***
  - **Compiling and running the client on Windows**
  - ***python Lab4_Exp1_Sym_Client_Win.py***

Exp 1: Execution and output

gcc Lab3_Exp1_Sym_Server_WSL.c -lssl -lcrypto -o sym_server

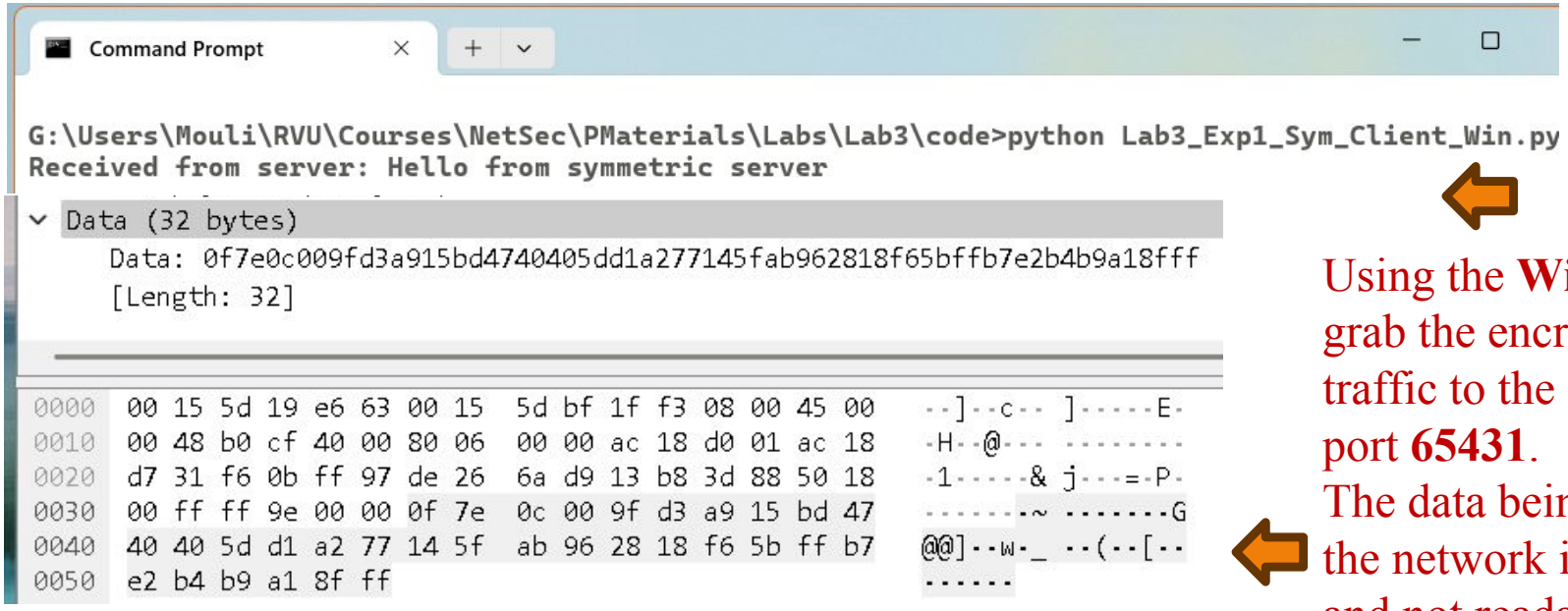
Build the server executable on WSL



Run the Server 1st on WSL

pip install cryptography

Install Python cryptography package on Windows



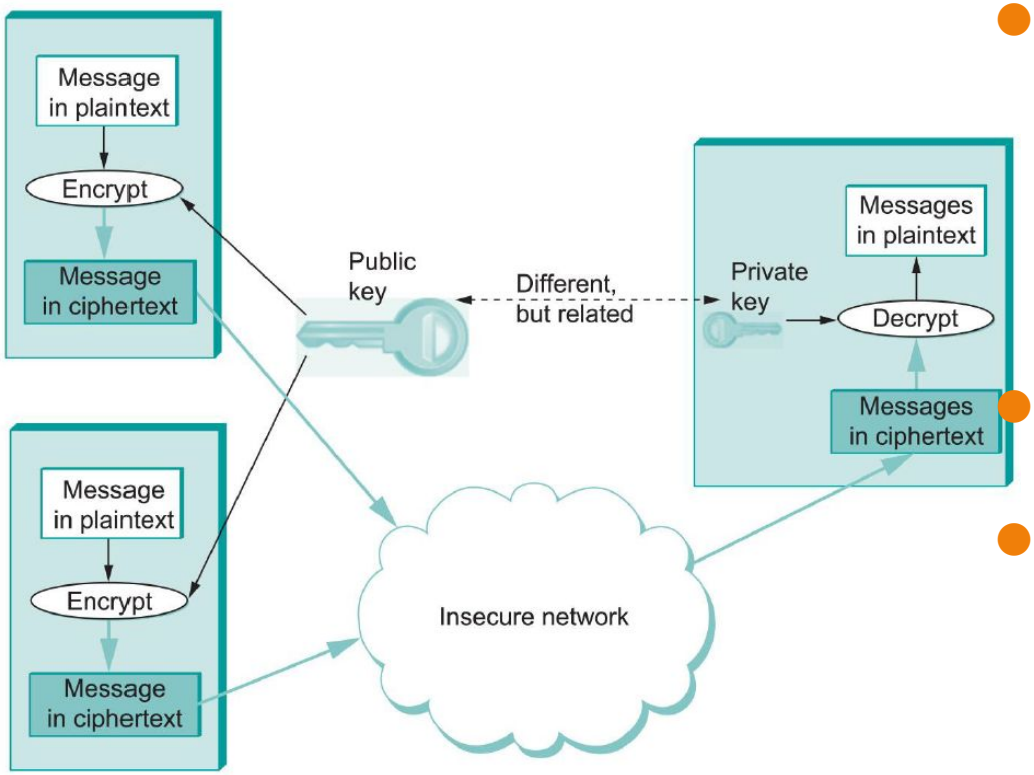
Run the client on Windows

Using the **Wireshark** grab the encrypted data traffic to the Server port **65431**.

The data being sent on the network is encrypted and not readable.

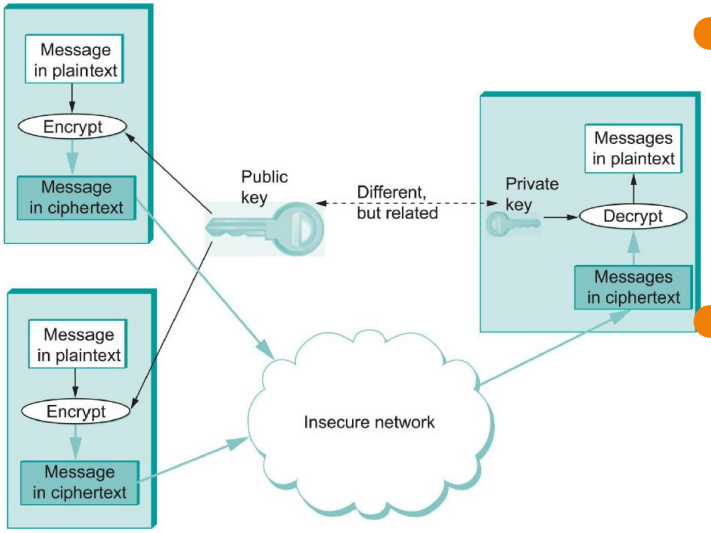
Asymmetric Key Ciphers
(private and public keys)

Public-key Encryption and Private-key Decryption



- Instead of a single key shared by two participants, a **public-key cipher** uses a **pair of related keys**, one for encryption and a different one for decryption.
- The pair of keys is "owned" by just one participant.
- The **owner** keeps the **decryption key secret** so that only the owner can decrypt messages; that key is called the **private key**.
- The **owner** makes the **encryption key public** so that anyone can **encrypt messages** for the owner; that key is called the **public key**.
- Obviously, for such a scheme to work, it must not be possible to deduce the private key from the public key.

Public-key Encryption and Decryption



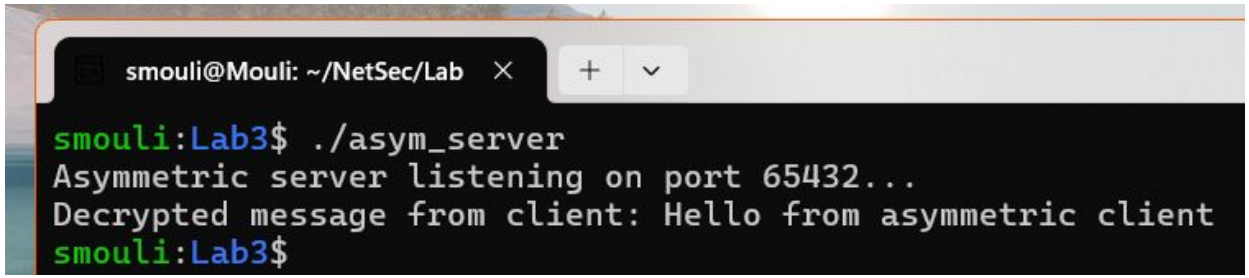
- Any participant can get the public key and send an encrypted message to the owner of the keys, and only the owner has the private key necessary to decrypt it.
- If we think of keys as defining a communication channel between participants, secret-key cipher provides a channel that is two-way between two participants.
- In secret-key, each participant holds the same (symmetric) key that either one can use to encrypt or decrypt messages in either direction.
- A public/private key pair, in contrast, provides a channel that is one-way and many-to-one: from everyone who has the public key to the unique owner of the private key.
- In public-key, for two-way confidentiality between two participants, each participant needs its own pair of keys, and each encrypts messages using the other's public key.

Exp 2: Generating pem files

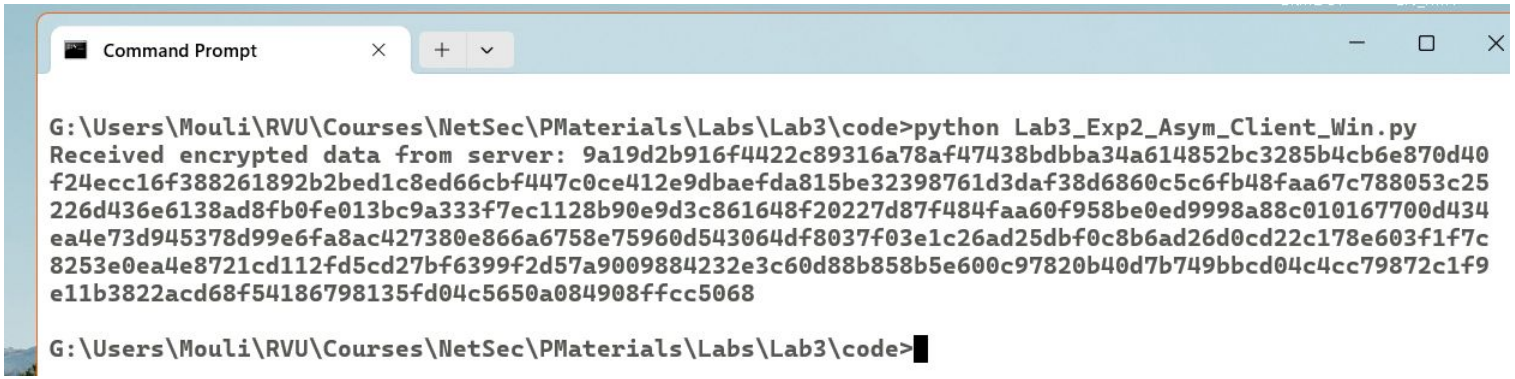
- **On WSL (Ubuntu):** using the Terminal:
  - Generate **private.pem** file on WSL, using the below command
  - ***openssl genpkey -algorithm RSA -out private.pem –pkeyopt rsa_keygen_bits:2048***
  - **genpkey utility is used to generate 2048 bits wide private key using RSA algorithm**
  - Generate **public.pem** file on WSL, with the below command by giving **private.pem** as input and ask openssl tool to extract its **public key**
  - ***openssl rsa -pubout -in private.pem -out public.pem***
- public.pem and private.pem files generated on WSL, need to be copied to Windows
- **Client** is to be run on **Windows**.
  - It uses the public.pem to encrypt the data before sending it to the server running on WSL
- **Server** is to be run on **WSL**.
  - It uses the private.pem file to decrypt the data encrypted using the public.pem

Exp 2: Running this program

- **Build and run the Server on WSL (Ubuntu):** using the Terminal:
  - **Building the server executable on WSL**
  - ***gcc Lab4_Exp2_Asym_Server_WSL.c -lssl –lcrypto -o asym_server -Wno-deprecated-declarations***
  - **For running the *./asym_server* keep private.pem in the same directory**
- **Run the Client on Windows**: using the command prompt
  - ***python Lab4_Exp2_Asym_Client_Win.py* keep public.pem in the same directory**



asym_server running on WSL:

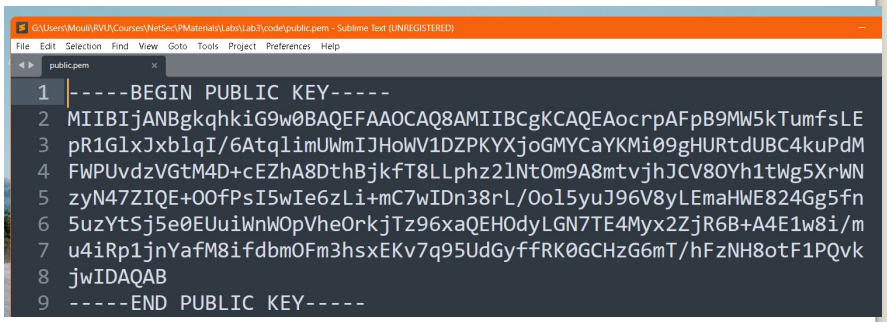


asym_client running on Windows:

pem format: Explained

- The **".pem"** extension stands for "**Privacy Enhanced Mail**."
- It denotes that the file's contents are encoded in the **PEM format**: a **Base64 encoding** of binary data
- **Base64** encoding is a method for converting binary data into an ASCII string format by representing it using 64 different readable characters
- Encoding involves taking the binary data, breaking it into groups of 3 bytes (24 bits), and then splitting those 24 bits into four 6-bit groups
- Each 6-bit group is then mapped to a specific character from a set of 64 readable characters:
  - Uppercase letter (26)
  - Lowercase letters (26)
  - Digits (10)
  - Two additional symbols, '+' and '/' (2)
  - In total 64 unique readable characters

Sample **public.pem** contents



Lab 4: Summary

- **Symmetric Key Ciphers**
  - **OpenSSL** Introduction
  - Library Installations on WSL and Windows
  - **Exp 1**: Server in C - WSL – Client in Python – Win
- **Asymmetric Key Ciphers**
  - Explanation about the **pem** format
  - **Exp 2**: Server in C - WSL – Client in Python – Win