פרטי צד שרת:

**The server will store customer data and files saved in RAM (Random Access Memory). Additionally, it will maintain an SQLite database that will include a table of user listings, the encryption keys sent to them, and a table of files received from them, along with whether the file passed successful verification against the client using checksum. It will also hold a local directory containing files received from clients. Data storage will be done through SQL tables in a file named db.defensive. This will allow, in the event of a server crash and recovery, retrieval of data on registered clients and stored files. Information about clients will be saved in a table named "clients." Table structure:**

* **ID (16 bytes, 128 bits): A unique identifier for each client.**
* **Name (255 characters): An ASCII string representing the username, including a null terminator.**
* **PublicKey (160 bytes): The client's public key.**
* **LastSeen: Date and time of the last request received from the client.**
* **AESKey (128 bits): AES key sent to the client.**

**Information about received files will be stored in a table named "files." Table structure:**

* **ID (16 bytes, 128 bits): A unique identifier for each client.**
* **FileName (255 characters): An ASCII string representing the file name as sent by the user, including a null terminator.**
* **FilePath (255 characters): An ASCII string representing the relative path and file name as stored on the server directory, including a null terminator.**
* **Verified (Boolean): Whether the checksum was successfully verified against the client.**

לוגיקה צד שרת :

**Read the port from the file info.port. (If the file does not exist, issue a warning and work with the default port 1357 to avoid crashing with a traceback if the file is unavailable.)**

**2. Check the database; if it already exists, load customer data registered in previous runs.**

**3. Wait for client requests in an infinite loop.**

**4. Upon receiving a request, decode the request according to the protocol:**

a. **Registration Request: If the requested username already exists, the server will return an error. Otherwise, the server will generate a new UUID for the user, store the data in memory and the database, and return a success response.**

b. **Client Public Key: The client's public key will be received and updated in the database. In response, the server will generate an AES key, encrypt it with the public key, and send it back to the client.**

c. **Message with an Encrypted File: The server will decrypt the encrypted file using the original AES key sent by the same client and calculate the CRC (which is the value obtained from the checksum operation). The calculation, on both the server and client sides, should be performed identically to the "cksum" command in Linux. For reference, code for students performing the calculation will be provided, which you can use.**

d. **The server will receive a success message from the client (verified CRC) or up to three retries with the file.**

תקשורת :  
  
**Public Key** The client generates a pair of RSA keys, public and private, and sends the public key to the server while saving the private key in a file named key.priv. (In subsequent connections, the same file will be reused, and new RSA keys will not be generated.) In response, the server should send an AES key that has been encrypted using the public key.

**Receiving the AES Key and Encrypting the File** After receiving the AES key, the client uses its private RSA key to decrypt it and obtain the AES key. In response, the client encrypts the file it needs to transfer using the AES key and sends the encrypted file to the server. Simultaneously, the client should calculate the CRC of the file so it can compare it to the CRC received from the server.

**Verification Using CRC** The server should receive the encrypted file from the client, decrypt it using the AES key, and also calculate the CRC. It then sends the CRC back to the client for verification.

**Communication Protocol** General:

* The protocol is binary and implemented over TCP.
* All numerical fields must have values greater than zero (unsigned) and are represented as little endian.
* This protocol supports requests to the server and responses to the client. Requests or responses can contain a "message."
* Messages can be exchanged between clients.

Remember! The protocol is strict and cannot be modified. As such, any server and client implementing the protocol can work with each other.

**Registration to the System**

1. Every client connecting for the first time registers with a service using a name (a string with a maximum length of 255 characters) and passes their public key.
2. The server returns a unique identifier generated for the client or an error if the name already exists in the database.

פרוטוקול:

**Protocol Details** **Requests** **Request Structure from the Client to the Server:** The server will decode the content (payload) according to the request code.

**Request to the Server**

* **Header (Header)**
  + **ID Client (16 bytes, 128 bits):** A unique identifier for each client.
  + **Version (variable bytes):** Client version number.
  + **Code (2 bytes):** Request code.
  + **Payload Size (4 bytes):** Size of the request's content.

**Content (Payload):** The payload is a variable content depending on the request.

Please note that these are the specifications for the structure of requests from the client to the server in the communication protocol. The payload content will vary depending on the specific request being made.

בקשות:

**Payload Content (payload)** The content varies depending on the request, and each request has a different structure.

**Request Code 1025 – Registration**

* **Name (255 bytes):** An ASCII string representing the username, including a null terminator. (terminated null)
* Note: The server will ignore the ID Client field.

**Request Code 1026 – Sending a Public Key**

* **Name (255 bytes):** An ASCII string representing the username, including a null terminator. (terminated null)
* **Key Public (160 bytes):** The client's public key.

**Request Code 1027 – Returning Login (If the client has registered previously)**

* **Name (255 bytes):** An ASCII string representing the username, including a null terminator. (terminated null)

**Request Code 1028 – Sending a File**

* **Size Content (4 bytes):** Size of the file (after encryption).
* **Name File (255 bytes):** Name of the sent file.
* **Content Message:** Variable content of the file, encrypted using a symmetric key.

**Request Code 1029 – Valid CRC**

* **Name File (255 bytes):** Name of the sent file.

**Request Code 1030 – Invalid CRC, Resend (Subsequent request will be 1028)**

* **Name File (255 bytes):** Name of the sent file.

**Request Code 1031 – Invalid CRC on Fourth Attempt, Finished**

* **Name File (255 bytes):** Name of the sent file.

These are the specifications for the payload content for various request codes in your communication protocol. Depending on the specific request being made, the payload will contain different fields and data.

תשובות מהשרת:

**Responses** **Server Response**

* **Header (Header)**
  + **Version (variable bytes):** Server version number.
  + **Code (2 bytes):** Response code.
  + **Payload Size (4 bytes):** Size of the response's content.

**Content (Payload):** The payload content varies depending on the response.

**Response Code 2100 – Registration Successful**

* **ID Client (16 bytes, 128 bits):** Unique identifier of the client.

**Response Code 2101 – Registration Failed**

**Response Code 2102 – Public Key Received, Sending Encrypted AES Key**

* **ID Client (16 bytes, 128 bits):** Unique identifier of the client.
* **Encrypted Symmetric Key:** Variable content of the encrypted AES key sent to the client.

**Response Code 2103 – File Received with Valid CRC**

* **ID Client (16 bytes, 128 bits):** Unique identifier of the sending client.
* **Size Content (4 bytes):** Size of the file (after encryption).
* **Name File (255 bytes):** Name of the sent file.
* **CRC (4 bytes):** Cksum value.

**Response Code 2104 – Acknowledgment of Message Received**

* **ID Client (16 bytes, 128 bits):** Unique identifier of the client.
* Note: This message can be received as a response to message 1029 or 1031 from the client.

**Response Code 2105 – Acknowledgment of Return Login Request, Sending Encrypted AES Key – Same Table as Code 2102**

* **ID Client (16 bytes, 128 bits):** Unique identifier of the client.
* **Encrypted Symmetric Key:** Variable content of the encrypted AES key sent to the client.

**Response Code 2106 – Rejected Return Login Request (Client not registered or invalid public key). In such a case, the client must re-register as a new client.**

* **ID Client (16 bytes, 128 bits):** Unique identifier of the client.

**Response Code 2107 – General Server Error Not Handled in Previous Cases (e.g., Disk Space Exhausted, General Database Error, etc.).**

These are the specifications for the content of responses to various response codes in your communication protocol. Depending on the specific response, the payload will contain different fields and data.

הצפנה:

**Symmetric Encryption** For symmetric encryption, use CBC-AES (Cipher Block Chaining with Advanced Encryption Standard).

* Key Length: 128 bits.
* It can be assumed that the Initialization Vector (IV) is always set to zero (the memory is filled with zeros).
* Note that using the same IV with the same key every time is not secure, but it suffices for the purpose of this exercise.

**Asymmetric Encryption** For asymmetric encryption, use RSA.

* Key Length: 1024 bits.