Network Security 1[a] --- TCP-like Layer (Simple Protocol)

Author: Ruofan Shen

For this assignment, I made a "restaurant-service like" protocol to simulate a customer-and-waiter situation. And below comes the detail of the protocol:

First of all, I should clarify that the server has a list for each dish in stock.

```
dict = {
            "Sandwiches": {"Bacon Avocado Chicken": 96,
                           "Buffalo Chicken Ranch": 23,
                           "Grilled Chicken": 57},
            "Salads & Soups": {...},
            ...
            "Desert": {...}
        }
```

- The client sends a request with the client's name to the server for the Menu
  - The client sends RequestMenu(name, tableNumber) to Server
- The server responds by sending a menu including an identifier for this exact request, with the same name, table number and a dictionary object representing the menu. e.g:

  ```
  dict = {
              "Sandwiches": ["Bacon Avocado Chicken", "Buffalo Chicken Ranch", "Grilled Chicken"],
              "Salads & Soups": [...],
              ...
              "Desert": [...]
          }
  ```

  - Therefore, the server will respond:

    ```
    Menu(ID, name, tableNumber, dict)
    ```

- The client checks if the name and table number are the same as the original customer name and the exact numbers. Therefore, here comes the two possibilities for the client-side response:
  - If the fields are the same as the original sent one, the client chooses the food from the responded menu and comfirms the quantity along with the same ID, name and the table number. That is, the respond message will look like:

  ```
  # 'Ordered' means the dishes that the customer has ordered
  ordered = {
      "Sandwiches": [("Bacon Avocado Chicken", 1), ("Buffalo Chicken Ranch", 2)],
      ...
      "Desert": [...]
  }
  Order(ID, name, tableNumber, ordered)
  ```

  - If the decrypted message doesn't match with the original one, then the client will respond an error message with the same ID:

  ```
  Error(ID, errorMessage)
  ```

- The server receives the Order message and check if every dish the client chose exists in the menu. So there will be two possibilities for

the response.

- If all the dishes exist in the menu and are still avaliable in the stock list, the server will send a "Start Cooking" message with ID, the client name and table number to client. Then the server reduces the corresponding item in stock.

```
Cooking(ID, name, tableNumber)
```

- If some of the dishes do not exist or has been sold out, the server will send a "Warning" message with ID, name, table number and the list of missing dishes to client

```
Warning(ID, name, tableNumber, List[missing dishes])
```

- If it turns out that the client didn't order anything, it will respond a "Nothong" message with with ID, name, table number:

```
Nothing(ID, name, tableNumber)
```

- If the client receives a Warning message, redos the processes after the server sent the menu.
- If the client receives a Nothing message, it does nothing.
- If the client receives a Cooking message, it will respond a Thanks message.

```
Thanks(ID, name, tableNumber)
```