Adversary model and GDPR

The parties who follow the protocol trust each other, it is therefore assumed any adversaries are passive and don't deviate from the protocol. But since the communication is over the internet there is a chance of a Dolev-Yao adversary. The active Dolev-Yao adversary is assumed to have full control over the network, being able to inject, modify, or intercept messages, but cannot break encryption. This means the Dolev-Yao adversary cannot decrypt encrypted messages or forge digital signatures without a corresponding key.

The hospital must ensure patient data according to GDPR. Showing patient data to other patients violates GDPR which instructs the protection of personal data. Furthermore, if the Hospital decided to host patient information in plain text it could further contribute to the identification of individuals, also violating GDPR (unencrypted plain text documents are easy to read by an adversary). Even if the patients' names were to be removed from the data there could be enough data to extract who the patient is. This could either be by cross referencing with outside sourced data, or looking at the unique combinations of attributes like age, gender, or zip codes.

Finally, if we were to use Federated Learning with Secure Aggregation, there could still be potential risks involved. Either the machine learning model could be compromised - and if the model leakage has information about the patients - that information could be extracted. There could also be algorithmic vulnerabilities which could be discovered, or if the dataset contains unique patterns (or is small enough), the adversary could gain further sensitive information from the aggregated data.

Protocol Design

To ensure security the protocol uses Secure Multi-Party Computation (SMPC) and Transport Layer Security (TLS). The protocol uses TLS to secure against potential Dolev-Yao adversaries to prevent them from decrypting encrypted messages. It also uses SMPC to ensure that individual data pieces are not revealed during computation, which allows for secure handling of patient data according to GDPR standards. This means neither the patients nor the hospital have access to each other's individual data.

These are the building blocks:

- **SMPC**
    - **n-out-of-n Additive Secret Sharing**: Each patient's data is split into shares, and the sum of these shares is computed without revealing individual data pieces. (i.e. no party can learn the secret without collaborating)
- **TLS**
    - **Confidentiality:** The data is encrypted using symmetric cryptography
    - **Integrity:** Through Message Authentication Codes (MAC) it verifies data integrity and protects it against modification
    - **Authentication:** Through digital signatures to authenticate communicating parties to prevent impersonation attacks

The hospital acts as the aggregator in the protocol. It receives shares of patient data, aggregates them, and computes the final result without accessing individual data pieces.

The hospital must:

- Establish a secure TLS connection with each patient to ensure confidentiality, integrity, and authentication
- Collect shares from patients using SMPC
- Perform the aggregation of shares
- Show the final aggregated result

Patients are responsible for generating and sharing their data securely.

Each patient must:

- Split their data into shares using n-out-of-n Additive Secret Sharing
- Establishes a secure TLS connection with the hospital
- Transmits their shares to the hospital over the secure TLS connection
- Ensure that their individual data pieces are not revealed during the process

The hospital and patients certificate is generated through a secure RSA algorithm.

Final

The program was written in GO using its built-in packages, like http, to serve TLS. This made the implementation a bit easier, as the patients and the hospital are set up as both server and client, where they communicate with each other through secure http-requests.

The patients generate shares of their data using n-out-of-n additive secret sharing. The shares are securely transmitted to the hospital using TLS.

The hospital aggregates the shares without accessing individual data pieces. The final aggregated result is computed.

Following the protocol, SMPC ensures that individual data pieces are not revealed during computation, and TLS prevents unauthorized interception, modification, impersonation, and ensures secure communication.

After running the program, an example output with all printed statements from the program is shown in the terminal. The hospital server port is 8080 and the three patient server ports are 8081, 8082, 8083. Here is an example of such an output where each transaction is shown.

```
2024/10/22 17:35:02 Creating hospital server at port 8080
2024/10/22 17:35:02 Patient 8081 has data value 57
2024/10/22 17:35:02 Patient 8081 registered with hospital
2024/10/22 17:35:02 Creating patient server at port 8081
2024/10/22 17:35:02 Patient 8083 has data value 34
2024/10/22 17:35:02 Patient 8083 registered with hospital
```

2024/10/22 17:35:02 Creating patient server at port 8083
2024/10/22 17:35:02 Patient 8082 has data value 90
2024/10/22 17:35:02 Patient 8082 registered with hospital
2024/10/22 17:35:02 Creating patient server at port 8082
2024/10/22 17:35:02 Hospital 8080 received POST from /patients
2024/10/22 17:35:02 Hospital 8080 registered new patient at port 8081
2024/10/22 17:35:02 Patient 8081 registered with hospital. Received response code 200 OK
2024/10/22 17:35:02 Hospital 8080 received POST from /patients
2024/10/22 17:35:02 Hospital 8080 registered new patient at port 8083
2024/10/22 17:35:02 Patient 8083 registered with hospital. Received response code 200 OK
2024/10/22 17:35:02 Hospital 8080 received POST from /patients
2024/10/22 17:35:02 Hospital 8080 registered new patient at port 8082
2024/10/22 17:35:02 Hospital 8080 Sending ports to patients
2024/10/22 17:35:02 Hospital 8080 sending ports [8083 8082] to 8081
2024/10/22 17:35:02 Patient 8081 received POST from /patients
2024/10/22 17:35:02 Patient 8081 received list of patients: [8083 8082]
2024/10/22 17:35:02 Hospital 8080 sent ports to 8081 with response code 200 OK
2024/10/22 17:35:02 Hospital 8080 sending ports [8081 8082] to 8083
2024/10/22 17:35:02 Patient 8083 received POST from /patients
2024/10/22 17:35:02 Patient 8083 received list of patients: [8081 8082]
2024/10/22 17:35:02 Patient 8082 received POST from /shares
2024/10/22 17:35:02 Patient 8083 received POST from /shares
2024/10/22 17:35:02 Hospital 8080 sent ports to 8083 with response code 200 OK
2024/10/22 17:35:02 Sent share to 8082 from 8081 Received response code: 200
2024/10/22 17:35:02 Hospital 8080 sending ports [8081 8083] to 8082
2024/10/22 17:35:02 Sent share to 8083 from 8081 Received response code: 200
2024/10/22 17:35:02 Patient 8082 received POST from /shares
2024/10/22 17:35:02 Patient 8081 received POST from /shares
2024/10/22 17:35:02 Sent share to 8082 from 8083 Received response code: 200
2024/10/22 17:35:02 Sent share to 8081 from 8083 Received response code: 200
2024/10/22 17:35:02 Patient 8082 received POST from /patients
2024/10/22 17:35:02 Patient 8082 received list of patients: [8081 8083]
2024/10/22 17:35:02 Computing patient 8082 aggregate share
2024/10/22 17:35:02 Patient 8082 aggregate share is 87
2024/10/22 17:35:02 Patient 8082 sending aggregate share 87 to hospital
2024/10/22 17:35:02 Patient 8081 received POST from /shares
2024/10/22 17:35:02 Computing patient 8081 aggregate share
2024/10/22 17:35:02 Patient 8083 received POST from /shares
2024/10/22 17:35:02 Patient 8081 aggregate share is 1
2024/10/22 17:35:02 Hospital 8080 received POST from /shares
2024/10/22 17:35:02 Computing patient 8083 aggregate share
2024/10/22 17:35:02 Patient 8083 aggregate share is 93
2024/10/22 17:35:02 Hospital 8080 received share 87 with a total of 1
2024/10/22 17:35:02 Patient 8081 sending aggregate share 1 to hospital
2024/10/22 17:35:02 Patient 8083 sending aggregate share 93 to hospital
2024/10/22 17:35:02 Sent aggregate share to hospital from 8082 with response code: 200
2024/10/22 17:35:02 Hospital 8080 received POST from /shares
2024/10/22 17:35:02 Hospital 8080 received share 1 with a total of 2
2024/10/22 17:35:02 Hospital 8080 received POST from /shares

2024/10/22 17:35:02 Sent aggregate share to hospital from 8081 with response code: 200
2024/10/22 17:35:02 Hospital 8080 sent ports to 8082 with response code 200 OK
2024/10/22 17:35:02 Sent share to 8081 from 8082 Received response code: 200
2024/10/22 17:35:02 Hospital 8080 received share 93 with a total of 3
2024/10/22 17:35:02 Patient 8082 registered with hospital. Received response code 200 OK
2024/10/22 17:35:02 Finished computing: Final value is 181
2024/10/22 17:35:02 Sent aggregate share to hospital from 8083 with response code: 200
2024/10/22 17:35:02 Sent share to 8083 from 8082 Received response code: 200