Handin 2

Part 1

Reflect on this scenario in the context of the GDPR: What are the potential issues in having the hospital store plaintext private data provided by patients even if they have consented to participate on the experiment and have their data processed?

When processing data, there are multiple principles you need to adhere to in order to comply with GDPR. In this scenario of the hospital I would like to reflect on two of these principles:

- Principle of data minimization
  - This principle states that *"…You should collect and process only as much data as absolutely necessary for the purposes specified…"*. Given that the ML-algorithm being trained needs the aggregated data of the participants, one could argue that the hospital does not need to collect the data of the individual participant. Storing the plaintext data of the participants on site could therefore mean that the hospital would store more data than absolutely needed, thus breaking this principle.
- Principle of integrity and confidentiality
  - As a datacontroller the hospital is required to handle data securely by implementing "appropriate technical and organizational measures". Storing the plaintext information on site will therefore result in the hospital having to implement further security procedures and training in order for them to guarantee that they would be able to keep the integrity and confidentiality of the data.

Would these issues be solved by removing the patients' names from their data before storing it?

These issues would not be solved by simply removing the names of the patients. Following GDPR personal data is defines as "Personal data is any information that relates to an individual who can be directly or indirectly identified."
One could assume that there are more direct identifiable data than names, such as email or even social security numbers. Even more health data it self is personal data, as this also can help to identify a individual.

What are the remaining risks in using Federated Learning with Secure Aggregation as suggested?

Even if we use Federated Learning with Secure Aggregation, the communication for running this protocol is still on the internet and open for vulnerabilities. The individual shares being send over the network could be vulnerable to a dolev-Yao attacker, who would be able to read, modify or inject shares.

Part 2

The advisarial model

The adversary model can be broken into two part:

- The participants are not trusted, but expected to follow the protocol (assumed passive adversaries)
- Because communication happens over the internet, vulnerabilities agains Dolev-Yao attackers are introduced.

My solution secures against this adversarial model as follows:

- It protects against Network attackers in the form of Dolev-Yao Attackers. -> This is because all communication in the solution happens over the TLS protocol which ensures integrity, confidentiality and authentication.
- It also holds secure against the passive adversaries, under the assumption of the honest majority. -> Following the nature of the Additive Secret Sharing implemented in the solution.

My solution

First I have to admit, my solution does not run. I have all building blocks, but I have a problem with my connections. This is my first time running distributed systems in python, and since Bernardo said that at the exam we will not be coding, I decided to spend the rest of the time answering this report, given I could see that I would not get done in time. -> I am hoping there is an opportunity to resubmit.

But feel free to look around still. Each client has its own file, but all logic is shared, it is only port-setup that differentiates them. To run the connections with TLC using the python SSL(outdated name) package, i have also created certificates for each participant (clients and server)

Still, let me describe my solution, as it is supposed to run.

We have a server node(the hospital) and the client nodes (the patients). They all communicate using TLS to protect agains network attackers. They all have the same understanding of R, which is the order of our Field F.

Each client node then picks a random element in F, this is their secret. They then follow the protocol for additive secret sharing as following “Lecture Notes: Secure Three Party Computation via Replicated Secret Sharing, by Bernardo David”. They select two random elements from F (2 shares), and subtract these from their secret, which leaves them with their own share.

Following this sentence: “Consider that all individual values held by patients are integers in a range [0,...,R] and that the aggregated value is the sum of all individual values, which is also assumed to be in the same range.” It should be noted that addition mod R is used as addition in F.

They then share one share with each of the other clients. Open receiving shares from other clients, the clients add this to their own kept share (addition mod R), which gives them each their local computation.

When each client have registered that they have received 2 shares (amount of participants - 1) they conclude the protocol, and sends their local computation to the server (hospital.) Here the hospital adds each of the received computations (addition mod R), for then finally to print the aggregated result. (“Teaching the LLM”).