# Mini Project 2

[rasni@itu.dk]

October 31, 2023

## 1 Introduction

This report outlines a secure solution for aggregating sensitive health data from multiple patients in a federated learning scenario. My solution ensures the privacy of individual data points while allowing for the training of a machine learning model by the hospital.

## 2 Adversarial Model and Security Requirements

### 2.1 Adversarial Model

In this scenario, i assume a semi-honest adversarial model where all parties follow the protocol but do not trust each other with access to the plaintext data. The hospital is trusted not to deviate from the established protocol, but the communication channels are insecure. An adversary could be:

- External attackers who eavesdrop on the communication to steal sensitive data.
- Internal attackers, such as patients or hospital staff, who attempt to learn about other patients' data.

### 2.2 Security Requirements

Our solution must satisfy the following security requirements:

- **Data confidentiality:** No patient's data should be revealed to other patients or unauthorized entities.
- **Data integrity:** The data must not be tampered with during transmission.
- **Data availability:** The hospital should be able to compute the aggregate value without interference.

## 3 Building Blocks for Proposed Solution

The proposed solution is built on the following primary technologies:

1. **Transport Layer Security (TLS):** To secure the communication channel against eavesdropping and tampering. Each participant (patient, server, hospital) uses their own TLS certificates. This verifies the identity of each party.
2. **Additive Secret Sharing:** Patients' data is split into multiple shares using an additive secret sharing scheme. These shares are then securely communicated, ensuring that only the aggregate value can be computed without revealing individual data points.
3. **Secret Share Reconstruction:** The server performs computations on the shares and returns reconstructed shares to the patients for further processing or finalization of the aggregation process.

## 4 Implementation and Instructions

### 4.1 Code Structure

The code base is structured into three main scripts:

- `tls_client.py`: Handles patient's local computations and interactions with the server and hospital.
- `tls_hospital.py`: Manages the hospital server, receiving and computing the aggregate data.
- `tls_server.py`: Acts as an intermediary to facilitate the secure multi-party computation.

### 4.2 Running the Code

To operate the solution, participants have the option to start the system using a single script or to manually launch each component in separate terminals. Follow the steps below according to your preferred method.

#### 4.2.1 Using the `run_all.py` Script

1. Ensure Python is installed on your system.
2. Install required dependencies, if any, using a package manager like `pip`.
3. Execute the `run_all.py` script to automatically initiate the server, hospital, and client scripts. It will ask for a value for each patient.

#### 4.2.2 Manual Execution in Separate Terminals

For a more hands-on approach, you can manually start each component of the system as follows:

1. Open a terminal and start the server by running `python tls_server.py`.
2. In a new terminal, start the hospital by running `python tls_hospital.py`.
3. For each client, open a separate terminal. You can start a client with no arguments by simply running `python tls_client.py`, which will prompt you to enter the name and value, or you can provide the arguments directly, like `python tls_client.py Alice 100`.

The scripts uses libraries like asyncio for asynchronous communication and SSL for securing the channels.

## 5 Solution Overview

The solution integrates the building blocks to achieve a secure aggregation:

1. Patients use `tls_client.py` to securely send data shares to the server.
2. `tls_server.py` collects the shares and performs the secure aggregation.
3. `tls_hospital.py` receives the securely aggregated result from each patient without learning individual values for then to compute the sum.

This approach uses TLS to lower the risk of data exposure on the network and employs additive secret sharing to ensure that no single entity, including the hospital, has access to the full data, protecting against internal threats.

## 6 Conclusion

I have designed and implemented a secure solution for aggregating sensitive patient data in a federated learning setting. The solution protects against both external and internal threats while enabling the hospital to build a machine learning model without ever accessing individual patient data in plaintext.