# Secret Sharing Across Smart Grid Domain

## Introduction

The goal of the implementation is to show the ability to privately share data from Smart Meters to a Utility Company through the use of Aggregators. The implementation shows a naïve example of a network for three Smart Meters connected to two Aggregators which are then connected to a singular Electrical Utility Company.

## Included Files

ElectricalUtility.py

Aggregator.py

SmartMeter.py

UtilityServer.py

AggServer.py

SmartMeterClient.py

## Usage

1. Set up a VM for the Electrical Utility Company
2. Set up a VM for 2 Aggregators
3. Set up a VM for 3 Smart Meters
4. Run the Electrical Utility Company
5. Run each of the VMs for the Aggregators with the corresponding command line arguments
   1. Aggregator 1: 8001 1
   2. Aggregator 2: 8002 2
6. Run each of the Smart Meters with corresponding command line arguments
   1. Smart Meter 1: 1
   2. Smart Meter 2: 2
   3. Smart Meter 3: 3

Note for set up the IP address in the code must reflect the IP addresses of the VMs that the code is being run in.

## Electrical Utility

The ElectricalUtility.py file contains the functionality of the Electrical Utility (EU) company. The company stores the number of Aggregators that it is connected to, the number of Smart Meters in the network, and the total reading from each of the Smart Meters for billing purposes. The EU holds an array that is the length of the number of Smart Meters in the network.

## Utility Server

The UtilityServer.py file sets up a TCP server to connect the Electrical Utility (EU) company to the Aggregators within the network. The utility company receives information from the Aggregator in the format [Number of Aggregators, Smart Meter ID, Value from Aggregator]. The incoming data is appropriately processed by being split on a delimiter. The value from the aggregator is then appended to the list of values that the EU holds at the appropriate location in the array based on the Smart Meter ID that is sent. This value is the total consumption from each Smart Meter over a time period.

## Aggregator

The Aggregator.py file contains the functionality of the Aggregators. Each Aggregator has its on ID, LaGrange function, and delta function multiplier. It also holds a list of the shares that have been sent from each smart meter, a list of the current totals of the shares for each smart meter, and a list named “sumofshares” that holds the current total value of the shares multiplied with the delta function multiplier.

The Aggregator class contains methods that apply mathematical functions to the data that it receives from the Smart Meters in order to encrypt and hide the specific data from the utility company.

## Aggregator Server

The AggServer.py file sets up a TCP client connection to the Electrical Utility (EU) company and a TCP server connection to each of the Smart Meters in the network. The AggServer sets the number of smart meters that the Aggregator will connect to. The Aggregator will then listen for connections from the Smart Meters. The Aggregator will then receive the ID of the Smart Meter it is communicating with and the share data from each Smart Meter. This data is then used to calculate a coefficient using LaGrange Interpolation to send the value to the EU.

## Smart Meter

The SmartMeter.py file contains information for each Smart Meter. It holds and ID, a utility reading called a secret, a degree, a polynomial, a list of coefficients for the polynomial, a list of its shares, and a list of the time it took to send each share. It contains other methods to create the polynomial and the secrets based on randomly generated data.

## Smart Meter Client

The SmartMeterClient.py file sets up a TCP connection to each of the Aggregators. The number of Aggregators is preset and then the connections are established. The Smart Meter then creates and sends the shares to each of the Aggregators based on their IDs.