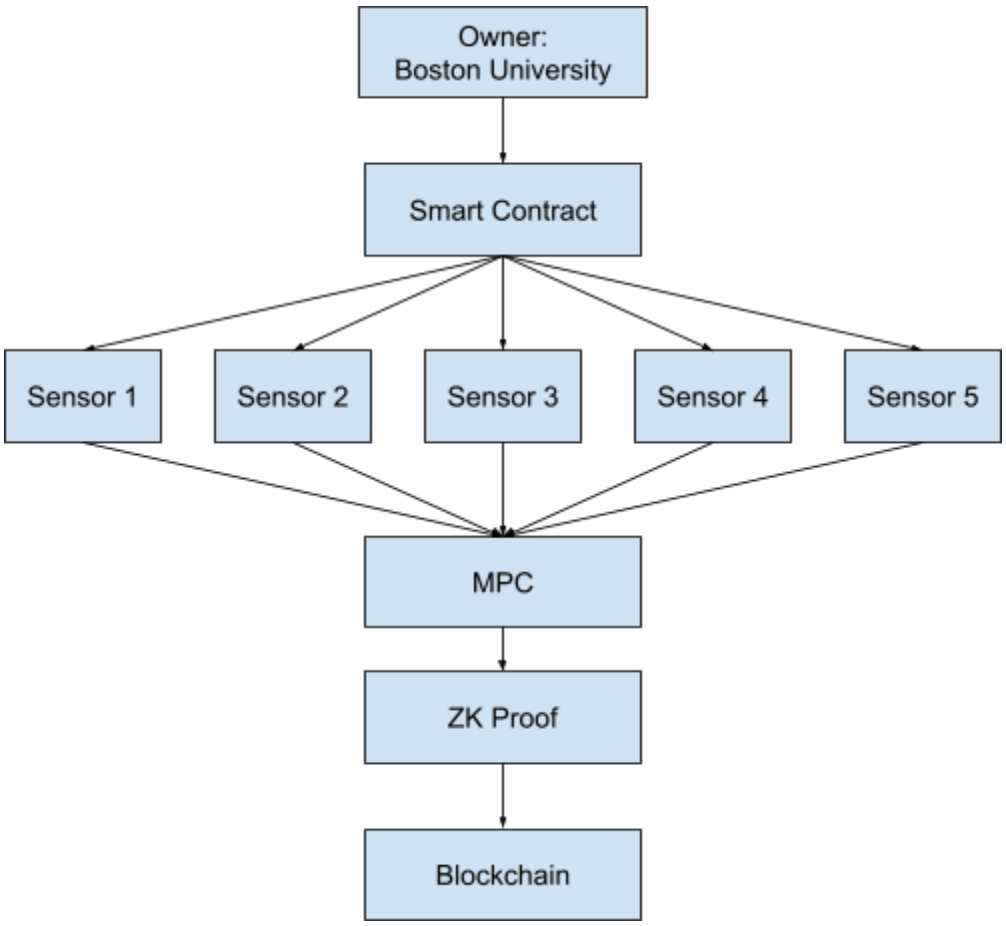
Report:

Our company is pleased to present our data collection sensors and software that capture and distribute crucial information about temperature, energy consumption, and trash collection throughout the CCDS building at Boston University.

Our sensor design and corresponding software makes use of multiple components that ensure accuracy and security of data collected.

Design:



- **Sensors:** Each sensor will collect data including energy consumption, temperature, and waste production. Each sensor will securely store its private key and other relevant data. Sensors will be collecting data locally every minute, and would aggregate data daily.
- **Data Aggregation:** Sensors will compute data such as daily totals using Multi-Party Computation (MPC). The MPC protocol will ensure that data from individual sensors will remain confidential.

- **Digital Signatures:** Sensors will locally and securely store its own private key, and will have a public key infrastructure (PKI) to securely communicate and verify the source of the data transmitted.
- **Smart Contract:** Sensors will be governed by a smart contract, which would serve as a governing mechanism for sensors and ensure that data collected is accurate and trustworthy.
- **Blockchain:** Daily totals/averages and their corresponding ZK proof will be posted to the blockchain. This data would be available publicly to anyone who is interested. However, all other intermediate data will be handled securely using the aforementioned components.
- **ZK Proof:** For each day, a zero knowledge proof will be generated using the sensor data and total calculated from MPC for each data type. It will prove (and thus convince the user) that the 5 numbers actually came from our sensors, and that they do in fact add up to the daily total, without giving any information about those 5 numbers to the user. For temperature data there is a different Circom circuit used to generate the proof, as for temperature instead of proving that the numbers add to a total we prove that the numbers average to a posted average. When the daily total is posted, the proof will be as well. The user can then use one of the two verifiers to verify any proof they want (one verifier for summed totals and one verifier for averaged totals).

Our code demonstrates the functionality of the sensors for calculating the amount of trash generated. Similarly, the code base will be adjusted to work concurrently to support energy and temperature data sharing.

The sensors are designed to capture data with high precision and accuracy. This includes the use of high-precision thermometers, highly-sensitive weight machines to measure weight of trash generated and highly sensitive current sensors to capture the best possible raw data.

To ensure that the data posted to the blockchain is accurate, the behavior of the sensors are governed by a smart contract, helping to maintain integrity in the system. It ensures that the sensors only function according to the standards set and record data accordingly. This includes provisions such as sensitive data remains private to sensors themselves helping maintain confidentiality, only sensors authenticated by the contract can access the relevant functions, participate in the MPC, and post their data. Additionally, we ensure that sensors only reveal their local totals once, and the owner only posts the daily total once and after a certain time of day.

The various parts of the code come together to achieve the goal of providing accurate and trustworthy data to the members of Boston University. Sensors collect data each minute, and then are aggregated using MPC. MPC protocol ensures that data remains confidential, as no single party/sensor has access to the total recorded values of another individual party/sensor. In our implementation, MPC is a good option for our sensor data system because each sensor can simply calculate shares and send it to other sensors, distributing computing workload and ensuring security of data.

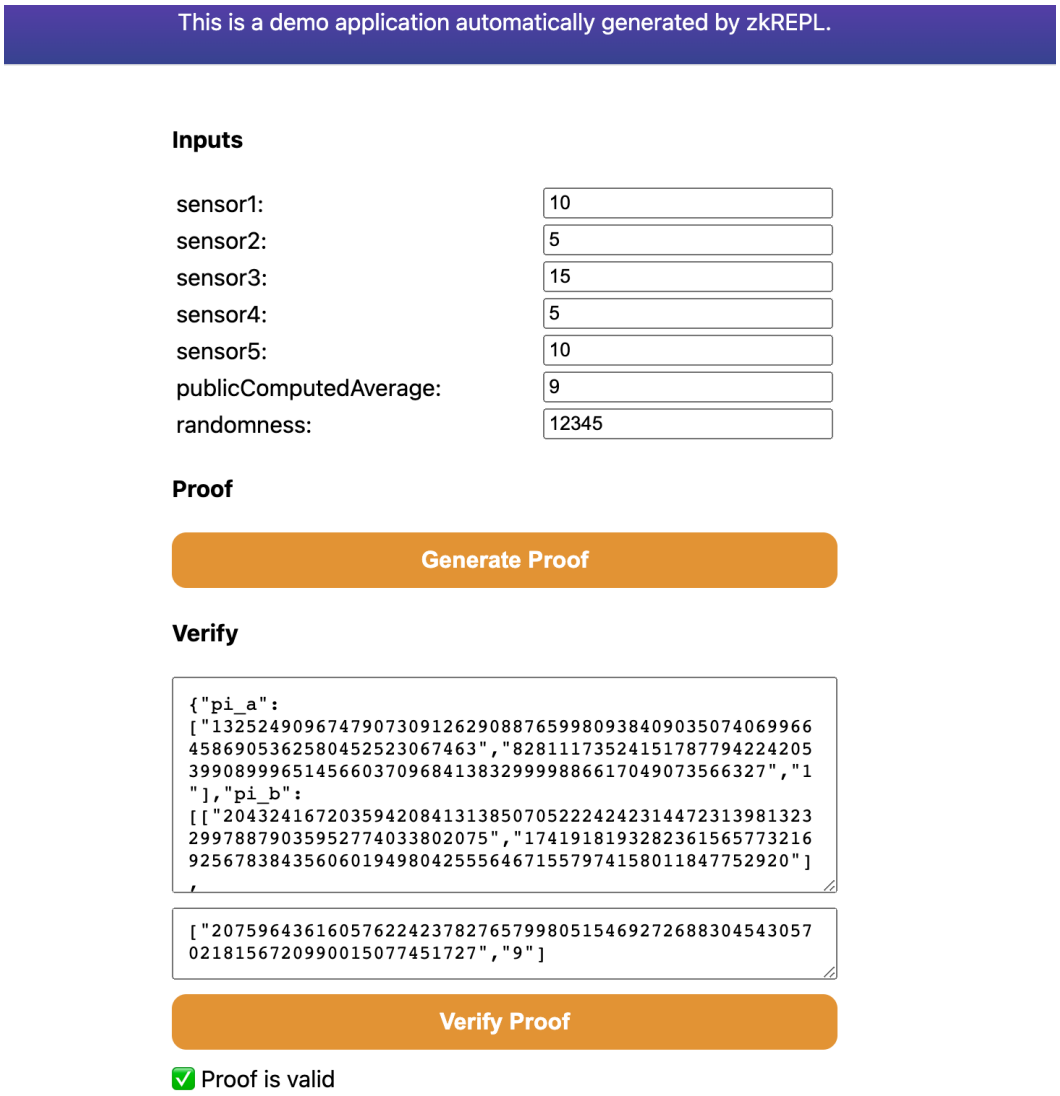
The use of digital signatures and the PKI help ensure that data transferred is authentic and that the sensor belongs to the network. Each sensor contains a public-private key pair for the purpose of this interface. It holds its own private key which signs the data it transmits. Additionally, other sensors can make use of the PKI to verify the authenticity of the data being sent by other sensors and authenticate that it is part of the network. Only authenticated sensors can access relevant functions, participate in the MPC, and post their data, helping ensure integrity of the system.

PKI Functionality: (Implemented within the Smart Contract)

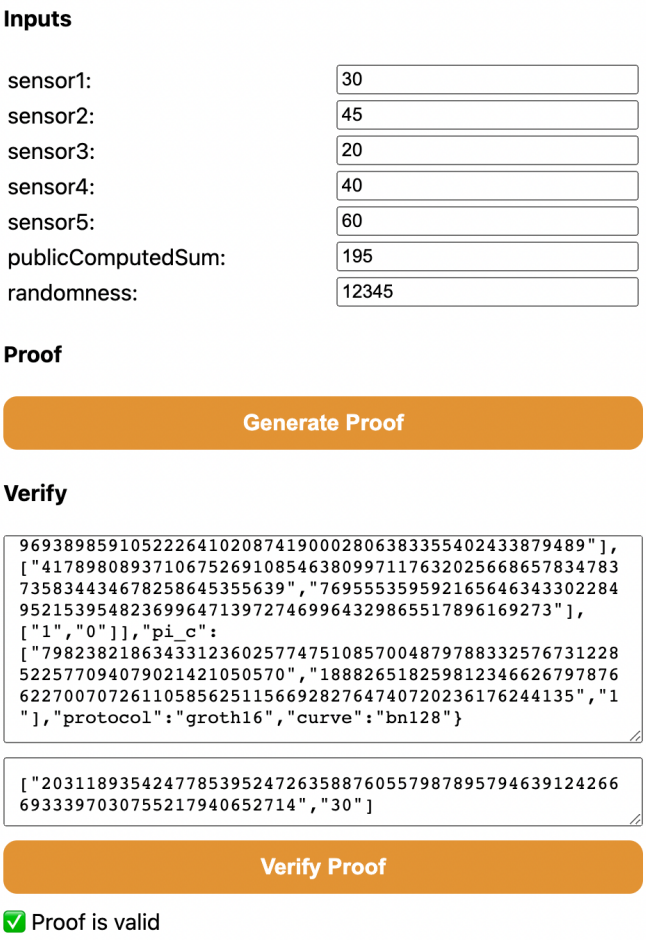
- Each sensor has its own private key with which it signs its data. A digital signature is created by encrypting the hash of the data being sent with the private key.
- Other sensors can make the use of public keys to verify the authenticity of data being transmitted by a sensor. If a signature is valid, the data is being sent by the correct sensor. If not, it would know that the data is tempered.
- Additionally, when joining the network, sensors need to share their public key so that they can be authenticated as part of the network.

Each day we also generate a ZK proof for the value the sensors collect associated with each data type using one of two Circom circuits as a base (one is for the energy consumption and food waste data and the other the average temperature). This proof will prove that the posted daily totals are the correct sum/average of the sensor data (implemented in the Circom circuits in our software portion) and that the sensor data did in fact come from our sensors (due to the limited time constraints we did not implement this feature into the circuits in the software portion of the project. In practice we would need to prove this - if we neglected to then a user would only be convinced that the poster of the proof knows 5 numbers that add/average to the total. They would not be convinced that these 5 numbers actually came from our sensors).

Below is an example of a generated proof/verifier for the average temperature data:



Below is an example of a generated proof/verifier for the food waste and energy consumption data:



When the daily totals are posted we will also post the generated proofs for each data type. A user could access these and run them through one of the two verifiers we have (one for each circuit) to verify its authenticity as a proof. After doing this, the user will be convinced that the daily total is correct and that the sensor data is legitimate.

We post the daily totals/average to the blockchain along with their respective ZK proof. The blockchain allows users a convenient, accessible, and secure way to access the collected data alongside proof of its correctness. The blockchain also preserves the record of posted data. If a user is to look back at a block for a piece of information that was previously posted, by nature of the blockchain they can be confident that the information has not been changed. They can then plug the proof into the verifier to check the correctness of the data and confidently use the data how they need without needing to worry about correctness and security.

Modifications to the code:
Some elements that were not fully implemented:

- Energy Data: For Energy Data, the overall procedure remains the same. However, for MPC, since the values are greater than 256, we would need to modify modulus to accommodate that. The corresponding code would also be similar in the smart contract, where energy totals would be posted along with the trash logging data.
- Temperature Data: Temperature as a total would not be beneficial to users. Therefore, the MPC would be done differently. We could still use the total, but then it would need to be divided by the number of sensors to come up with the average temperature in the building. Similar to Energy and Trash, the corresponding code will be added to the smart contract to allow for data to be posted to the blockchain with the trash, energy, and proofs.
- PKI: As mentioned previously in the report, a PKI interface will be used in our system. The PKI will be added to the smart contract to help ensure only authorized sensors take part in the network helping maintain integrity.
- UInt256: We were unable to find a more precise data type to use in solidity. From our understanding, floats and doubles do not exist in solidity. Therefore, in our code we assume that we are able to post data accurate to 2 decimal places.
- We did not implement the part of the ZK proof that would ensure that the data came from our sensors. In order to do this, we would need the PKI to be implemented so we could access the public keys for the sensors. As implemented the ZK proofs prove that the total/average is correct but in practice they would also prove that the data came from our sensors.

Future Addition- Outlier Detection:

Once a statistically significant amount of data has been collected we would intend to implement outlier detection. When a sensor sends data we could compare it to historical data of that type. If the data reported by the sensor falls outside of a predetermined range, for example above or below two standard deviations from the mean (this would be determined in consultation with the client to flag data in a way that best reflects their needs), we would flag the data. If, for example, a sensor's reported food waste data was flagged then we can check if the sensor is faulty. If it is not faulty, there may have been something of note that day that caused greater/less than the usual range of food waste. This would be implemented both to help ensure the accuracy of the raw data and better meet the clients needs by flagging outliers.

Summary of Individual Contribution:
Hemanshu: Business plan, Trash/Energy/Temperature Data Collection Functions, MPC Code, Smart Contract, PKI and their corresponding parts in the report.

Matthew: Business plan, blockchain considerations and corresponding parts of report, ZK proofs (circom code, generating proofs and verifiers with zkREPL, PKI considerations) and corresponding parts of report, outlier detection