Federated Calculations

Data Privacy and Security - CS528 Fall, 2021

Isaias Rivera

Dec 7, 2020

Contents

Team Description

The team is only myself. I am a third year undergraduate computer science student doing Co-Terminal at Illinois Tech. I value my online privacy greatly, which is why I took an interest in this class.

Application

The ultimate goal with this project was to have the capability of running calculations over an entire user base. This means the server, or whoever is running the calculation would not know any intermediate values of the computation. The only thing that should be revealed is the final computation to the entity requesting.

Dataset

Fitbit Fitness obtained from Kaggle includes personal fitness tracker data from 33 Fitbit users. The data includes things like daily calories burnt to hourly steps taken. Each user is separated with their own data, where no other object/entity is allowed directly access this data.

Privacy and Security Techniques

When a attribute is to be used for a calculation Laplacian noise is added to the value. Because calculations can use multiple attributes, it is also ensured that privacy parameter is split up between each attribute. Additionally, to ensure the entity requesting the calculation does not see any intermediary values, the calculations are performed on HE values as it is passed from user to user.

Lifecycle of a request

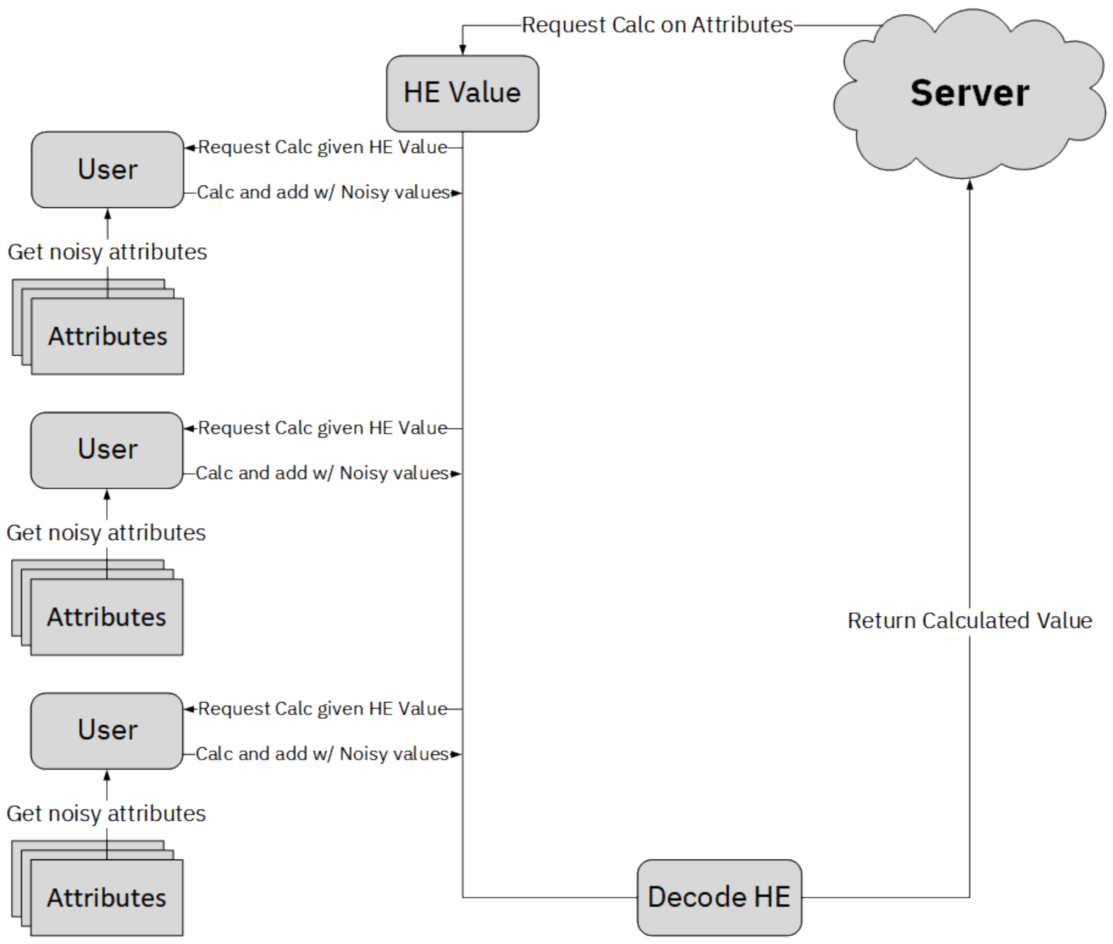
Request Calc on Attributes
HE Value
Server
User
Request Calc given HE Value
Calc and add w/ Noisy values
Get noisy attributes
Attributes
User
Request Calc given HE Value
Calc and add w/ Noisy values
Get noisy attributes
Attributes
Return Calculated Value
User
Request Calc given HE Value
Calc and add w/ Noisy values
Get noisy attributes
Attributes
Decode HE

Figure 1: The lifecycle of a server request to a user base

Fig.(1)

Implementation

This project was made using Python 3.10. Currently, communication between Server and User is done on the same instance, however, both objects do not access attributes directly from each other.

Example Dataset User

```
--------[ 8877689391 ]--------
  daily
    TotalSteps                  31
    TotalDistance               31
    TrackerDistance             31
    LoggedActivitiesDistance    31
    VeryActiveDistance          31
    ModeratelyActiveDistance    31
    LightActiveDistance         31
    SedentaryActiveDistance     31
    VeryActiveMinutes           31
    FairlyActiveMinutes         31
    LightlyActiveMinutes        31
    SedentaryMinutes            31
    Calories                    31
  heart
    Value    228841
  hourly
    Calories            735
    TotalIntensity      735
    AverageIntensity    735
    StepTotal           735
```

An example of a single User in the dataset. Each number represents the number of entries for that specific field

Example Calculation

```python
def main():
    """Main Function"""
    users: list[User] = dataset.get_dataset()  # Get all users as objects
    server = Server(users, epsilon=5)  # Initialize a server with privacy parameter eps

    # Example calculation given a list of attributes, determined upon request
    def test(pub: PubKey, acc: EncryptedNumber, usr: list[Number]) -> EncryptedNumber |
        calories = usr[0]
        mod_active_dist = usr[1]

        if mod_active_dist != 0:
            acc = acc + EncodedNumber.encode(pub, calories * mod_active_dist)
            return acc

        return False
```

```
# Run function test with the fields "daily calories" and "Daily moderately active
request = server.requestAction([FRAME.DAILY.CALORIES, FRAME.DAILY.MODERATELY_ACTIVE_
print(request.value / request.counter)  # Print the average, counter is how many us
```

Testing

Results

Summary

Improvements

In addition to a server being able to run calculations on a userbase, I also would want each user to be able to compare with each other without revealing each other's values. This might be accomplished with garbled circuits, however that might be too inefficient. Additionally, this could be expanded to federated learning, where each user could create a local model and be grouped with a server model.