LDP APPLICATIONS ON ENERGY CONSUMPTION DATA

Çisem Özden, Emircan Kocatürk, Talha Enes Güler, Zeynep Sıla Kaya

Abstract

In our project, we applied LDP protocols to three different energy consumption datasets, as we believe that sharing energy consumption data may lead to privacy concerns for households. There were studies of LDP applications on categorical data, but research on numerical data was more limited. For this reason, we aimed to make a benchmark project by designing and testing several LDP methods for numerical data. We divided our data into ranges to implement the LDP protocols used on categorical data. Our implementation includes Direct Encoding, Unary Encoding, Histogram Encoding, Binary Local Hashing, and Optimal Local Hashing. Lastly, based on the analyzed results, we developed a blended LDP protocol, taking into consideration the outputs of the seven LDP protocols that we have tested. There are tradeoffs between various implementation choices. We implemented eight different types of protocols in total and compared them in terms of Epsilon budget, error rate, and running time.

1. Methods

For distribution estimation, we implemented the GRR, RAPPOR, and OUE protocols that are already mentioned and discussed in the lecture. We adopted these protocols as a benchmark for the other LDP protocols. We chose BLH (Binary Local Hashing), OLH (Optimal Local Hashing), SHE (Summation with histogram encoding), and THE (Thresholding with histogram encoding) in addition to GRR, RAPPOR, and OUE.

One of the basic encoding types used in LDP protocols is direct encoding, which does not include the actual encoding process. It can be seen as the generalized version of the Randomized Response (RR) protocol, meaning it extends the binary response protocol to the case where the domain size of the possible responses is more than 2. The main purpose of this protocol is to convey the answers to the questions asked by the user with the probability of lying determined by Epsilon. The most familiar example of this is GRR. The basic logic of GRR is to equate the

probability of lying (q) to (1-p)/(d-1) where d denotes the domain size of the possible answers and p is the probability of telling the truth.

In order to implement this protocol and since we have numeric data, we divide the datasets into specific ranges. Through this operation, we created our dynamic domains which change from month to month depending on the maximum value in that dataset. For instance, in some dataset, if the range size is equal to 10 and the maximum energy consumption data is 2432, then this data set has a domain size of 243 in our estimation protocols and the 56 kW energy consumption amount falls into the 5th range. Estimation of the values are executed on modified domain values in every protocol. Consequently, range size is a decision variable because it affects the accuracy of protocols.

As the second type of encoding, we have used unary encoding. The primary function of this type of encoding is to convert each user's data set into a bit vector. First of all, during the encoding process, a bit vector consisting of 0s is created, then the corresponding index for which the user's data is correct is assigned to 1. Perturbation is performed according to the probabilities p and q which are determined by the epsilon value. At this point, the unary encoding can be performed as symmetric or optimized unary encoding. Here, the main difference between symmetric and optimized unary encoding comes from treating 1s and 0s in the bit vector differently. In symmetric unary encoding, regardless of whether the bit is 0 or 1, the bits remain as they are with probability p and are reversed with probability q.

Unlike RAPPOR, OUE does not treat 0s and 1s equally, meaning preserving and flipping probabilities differ for 0 and 1. Corresponding probabilities can be seen as follows:

$$\Pr[B'_\ell[i] = 1] = \begin{cases} \frac{1}{2} & \text{if } B_\ell[i] = 1 \\ \frac{1}{e^\varepsilon + 1} & \text{if } B_\ell[i] = 0 \end{cases}$$

Figure 1

$$\mathrm{Var}^*[\tilde{c}_{\mathrm{UE}}(i)] = \frac{nq(1-q)}{(p-q)^2} = \frac{nq(1-q)}{(\frac{e^\varepsilon q}{1-q+e^\varepsilon q} - q)^2}$$

Figure 2

The main purpose here is to reduce the equation given on the left and thus increase the utilization. You can see the p and q values minimizing this equation on the right.

$$q = \frac{1}{e^{\varepsilon} + 1} \text{ and } p = \frac{1}{2}$$

Figure 3

As the third type of encoding, we have implemented Histogram Encoding. For histogram encoding, the encoding part is very much the same as unary encoding. However, this time, rather than creating a bit vector, the corresponding histogram is created. All the bins are initialized as 0, except the one that is corresponding to the correct answer. The value corresponding to the index of the correct answer is initialized to 1. Furthermore, the perturbation part differs from the usual unary encoding. It is completely different from the two approaches we saw above, Laplace noise is added to each of the bins of the histogram. During this operation, the Laplace noise is determined from the distribution formed with 0 mean and 2/ε variance. Thresholding with histogram encoding is another protocol of histogram encoding.

$$\text{Support}_{\text{THE}}(B) = \{v \mid B[v] > \theta\}$$

Figure 4

This means that the value is supported by each noise count that is greater than θ. The user or the aggregator can both carry out this thresholding step. Since it doesn't get to the initial number, the privacy guarantee is unaffected. The protocol is pure when a Support function is provided via thresholding. The formulas for p and q are as follows:

$$p^* = 1 - F(\theta - 1);\ q^* = 1 - F(\theta),$$
$$\text{where } F(x) = \begin{cases} \frac{1}{2}e^{\frac{\varepsilon}{2}x}, & \text{if } x < 0 \\ 1 - \frac{1}{2}e^{-\frac{\varepsilon}{2}x}, & \text{if } x \geq 0 \end{cases}$$

Figure 5

The 6th and 7th protocols we suggest here differ from others in that hashing is used before the encoding process. Moreover, they do not utilize classical hashing protocols, rather they use local hashing to avoid possible collisions. BLH, in particular, uses hash functions that output a single bit, whereas OLH uses choices that are optimized from among hash functions. In BLH, each hash function in a hash function family hashes the input from the domain of input to a single bit, and then it is transmitted via randomized response. Here, perturbation is applied according to the following probabilities:

$$\Pr\left[b'=1\right]=\begin{cases} p=\frac{e^{\varepsilon}}{e^{\varepsilon}+1}, & \text{if } b=1 \\ q=\frac{1}{e^{\varepsilon}+1}, & \text{if } b=0 \end{cases}$$

Figure 6

OLH operates pretty much the same as BLH, but it is actually the generalized version of BLH. In OLH, each input value is hashed into a value in [g] where g is greater than or equal to 2. First, the input value is taken and hashed with a function selected among the hash functions in a randomized manner just like in BLH. Afterward, the value is obtained using modulo k, and a required value between 0 and k-1 is obtained. This value is then perturbed as in GRR and the output is transmitted to the user. Perturbation is applied according to the following probabilities:

$$\textbf{Perturbing. } \text{Perturb}(\langle H,x\rangle)=(\langle H,y\rangle), \text{ where}$$

$$\forall_{i\in[g]}\ \Pr[y=i]=\begin{cases} p=\frac{e^{\varepsilon}}{e^{\varepsilon}+g-1}, & \text{if } x=i \\ q=\frac{1}{e^{\varepsilon}+g-1}, & \text{if } x\neq i \end{cases}$$

Figure 7

Lastly, we added our own interpretation of the BLH and OLH implementations. We implemented it by combining the hashing and modulo parts. Our logic here is that if the values are hashed twice, the elements within the groups can be equally distributed. That's why we divided the values directly into groups in a randomized manner, distributing them equally.

In combined protocol, our aim was to minimize the error rates of LDPs for all Epsilon values. Upon implementing and testing the known protocols, we found that each protocol gave

different error performances when Epsilon ranges were changed. We consider that this variety of performances can be used in combination so that protocols process with higher accuracy while preserving privacy.

As part of this protocol, we separated epsilon values into three ranges: [0.0 - 1.0], [1.0 - 3.0], and [3.0 - 6.0]. For each epsilon range, we evaluated the error rate of the implemented seven protocols and selected the most accurate ones to create our results. For small values of epsilon, RAPPOR, OUE, and BLH return the lowest error rates with considerably high differences compared to others. For the medium range, RAPPOR, OUE, and BLH have strong contributions, while GRR, SHE, and THE have a smaller proportion. OUE works with high accuracy and RAPPOR has low error performance.

After these observations, we created the weight array such that [0.0,0.4,0.3,0.3,0.0,0.0,0,0] for low epsilon values which implies that each weight corresponds to relevant protocols. The weights were determined inversely proportional to error rates, which means that low error performance had a higher weight. Initially, we run the created protocols to get the results and multiply these results with the weight array, then the protocol returns weighted results. In the middle range, the weight array becomes [0.1,0.3,0.2,0.2,0.0,0.1,0,1], and these weights demonstrate that almost every protocol has a contribution to results with different powers. Lastly, epsilon values between 3.0 and 6.0 have weight arrays of [0.45,0.1,0.45,0.0,0.0,0.0,0.0].

2. Experimental Design

We designed our experiments carefully in order to evaluate the results and make accurate comparisons. Our London dataset contained consumption data for 1,5 years, while the other Solar and Non-Solar datasets covered 8 years. We decided to work with one year's data from each dataset in order to perform operations on them in parallel. For this reason, we made our calculations for 2008 in all three data sets. For each dataset, we ran the program for every month of the year we selected. We did not conduct experiments using different types of queries. According to the datasets we used, we can construct queries, such as how many people's consumption lies within the range [x,y], or what is the average consumption of people in X

months. In our experiments, no matter what the query type is, one can answer them by utilizing the distribution estimation of the LDP protocols given.

The LDP protocols that we used did not contain many parameters. The size of the range and the value of Epsilon were the only parameters that had an impact on implementation. Based on this, we decided that it would be appropriate to use different values for these parameters during the experiments. In the present study, the value of Epsilon was significant in the sense that it would result in fewer average error values when large. In contrast, the error is higher when Epsilon is small. Besides, some of the LDP protocols may yield better results, i.e., yield fewer error values, depending on the value of Epsilon. Therefore, using different Epsilon values when performing the experiments was significant. Since each protocol gives a different average error value, we attempted to include as many Epsilon values as possible to observe them more accurately. However, we didn't randomly choose them. We considered other similar works and included reasonable Epsilon values. Specifically, we ran our experiments with 0.1, 0.2, 0.5, 1.0, 2.0, 3.0, 4.0, 5.0, and 6.0 values of Epsilon.

The other parameter was range size. Typically, LDP protocols are used to perform estimations of frequencies based on categorical data. However, the datasets we worked on contained numerical data, rather than categorical data. To overcome this problem, we needed to split the distribution of the data into ranges. When splitting the whole distribution into ranges, we needed to consider the tradeoff between the time needed to implement the protocols and the maximum possible deviation from the real data. Here, the maximum possible deviation means how much we may deviate from the real data. For example, if the original amount of consumption is 153, the protocols may estimate the range of the data as [150, 160] when the range size is 10. However, the protocols may yield more precise estimates if the range size is 5, e.g. [150, 155]. Since it is not possible to estimate the exact correct consumption value by estimating ranges, range size was affecting the possibility of making better estimates in terms of closeness to the exact value. Although we cannot demonstrate the difference between the range estimation and the exact value of the data of the user, we thought it would be better to implement protocols for different values of range sizes in case it might have an effect on the average error values. Following this, we decided to conduct experiments with range sizes of 5, 10, and 20.

Lastly, in order to ensure the average error values that the protocols give, we ran the same experiments multiple times and took the average of them.

3. Results

Experiment results can be divided into mainly 4 parts which are the comparison of average error values of different protocol implementations under different values of epsilon data, the comparison of average error values of the implementations under different range values, overall runtime comparison of the protocols used, and the comparison of average error values depending on the different datasets.

In order to compare the performance of the protocols from a broad perspective, we tried to include as various graphics as possible. To compare the general performance of the LDP protocols in terms of average values under different epsilon values, one can look at the graphics in the figures numbered 8-21. The first 6 figures show the results of the implementations performed on the Non-Solar dataset. We included the results for all the ranges we used as a parameter. We added an extra figure to show the differences more clearly when the epsilon value is larger than 2. Figures 8 and 9 show that RAPPOR, OUE, and the combined protocol performed better for all epsilon values. OLH and GRR performed worse in general, especially for smaller epsilon values. Also, it can be seen from Figure 9 that OLH gives much better results for larger values of epsilon, which is almost the same as the results of the superiors which are OUE, RAPPOR, and combined. One particular observation is that the value of OLH didn't change for epsilon values less than 1. This is because the number of groups was determined as 2 when the range size is less than 1 in the implementation of OLH. Since the range size is the same for the epsilon values less than 1, the average value didn't change.

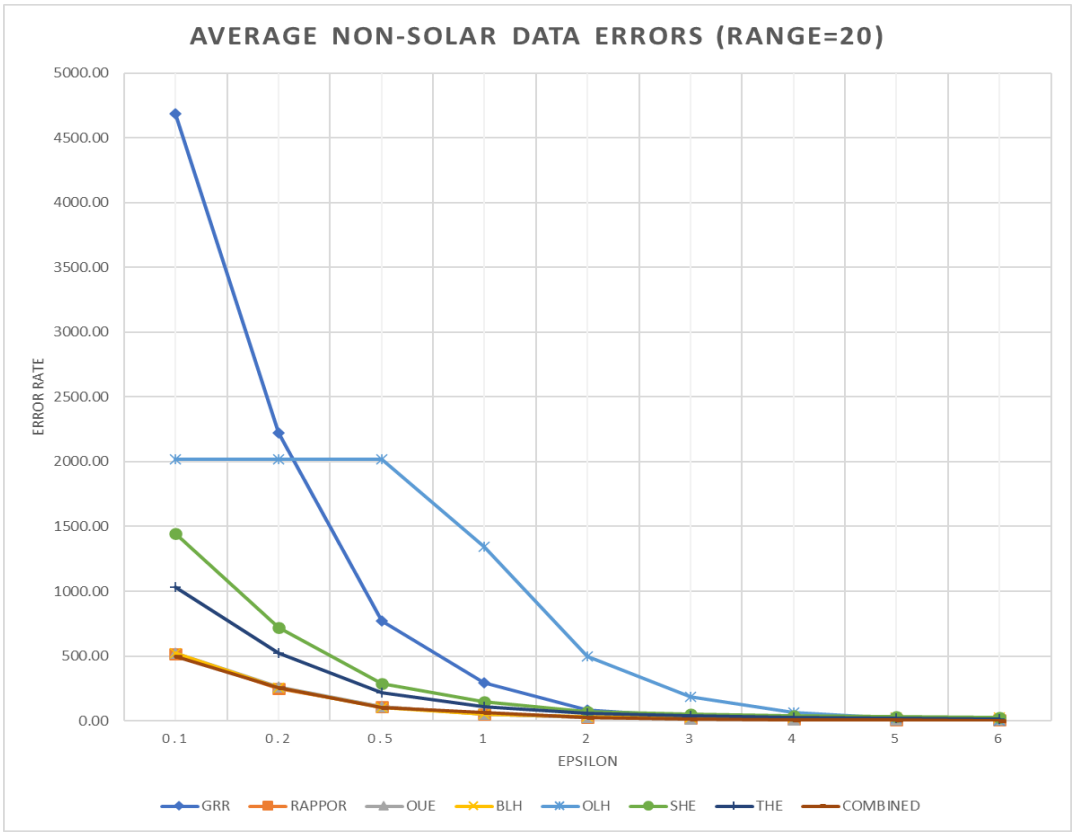

Figure 8

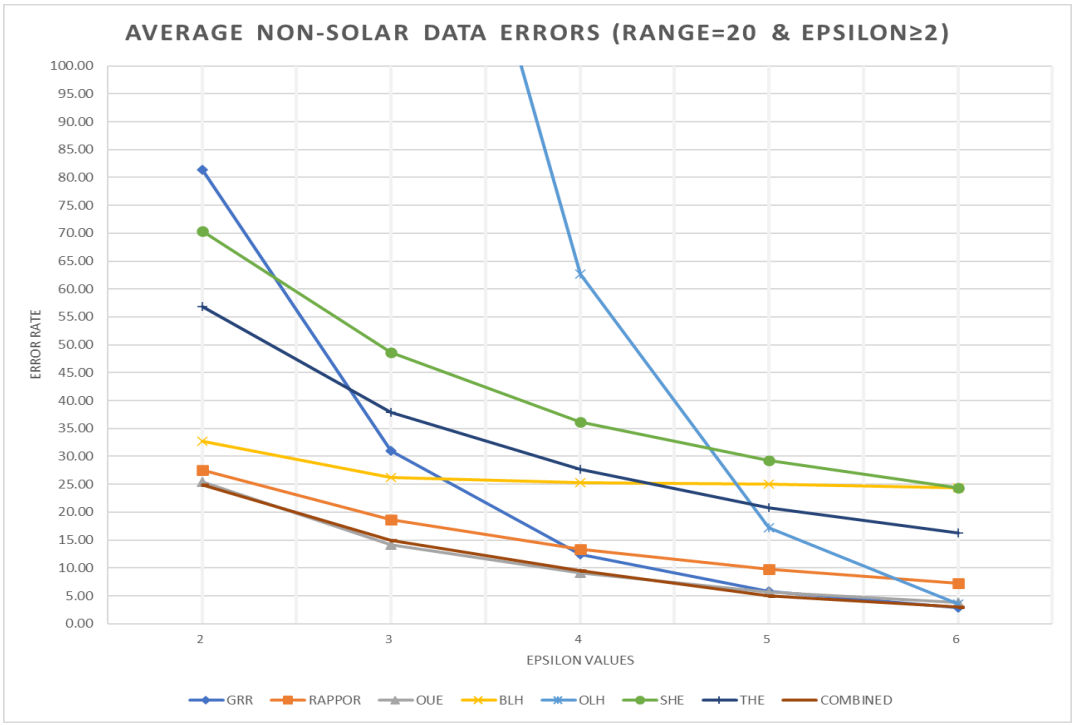

Figure 9

Implementations for other range choices show similar results. In Figures 10 and 11, we see the result of the same protocols when the range size is 10. There is not much difference in the relative behavior of the protocols in general. Again, we see that OLH, RAPPOR, and combined protocol are superior to others. One remarkable observation is the decrease in the average error values. Although their relative performances are the same, it is evident that the average error value decreased for all the protocols. This also supports our optimal choice of 10 as the range size. Moreover, in Figures 12 and 13, another similar result can be seen, this time using range size 5. Although it is not much different than the previous results, one observation here is that the average error value increased dramatically in the implementation of the GRR protocol. When the range size becomes 5, domain size increased by 100 percent compared to one with a range size of 10. Knowing that GRR protocol is highly dependent on domain size, it is not surprising that it yielded larger average error values.

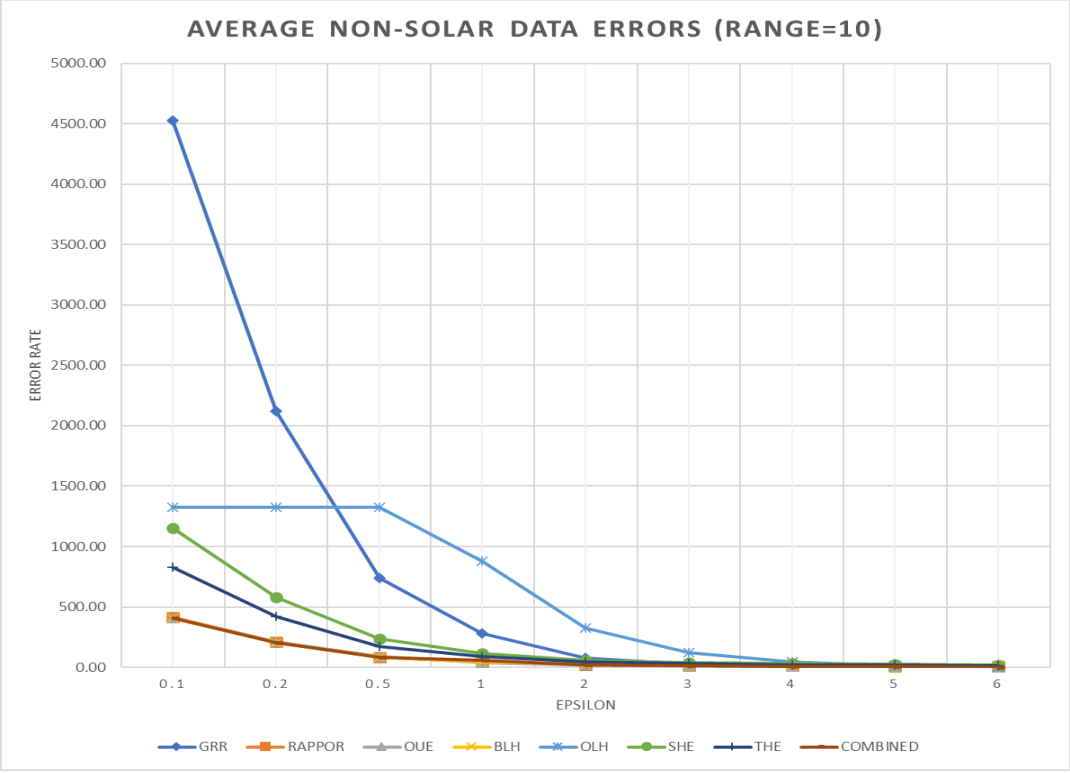

Figure 10

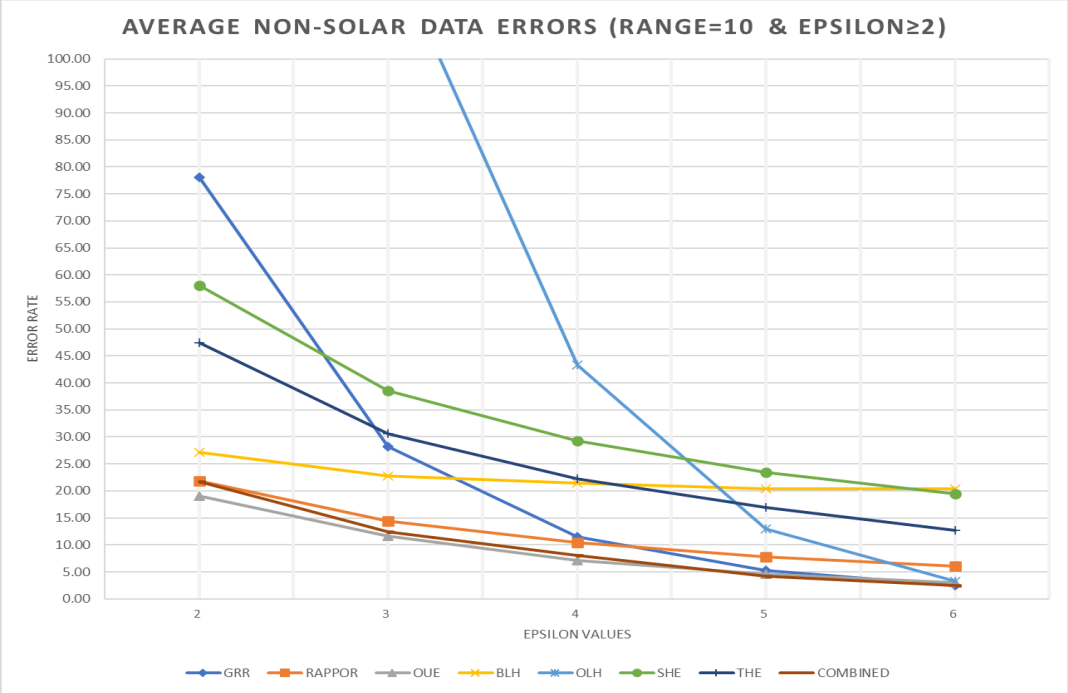

Figure 11

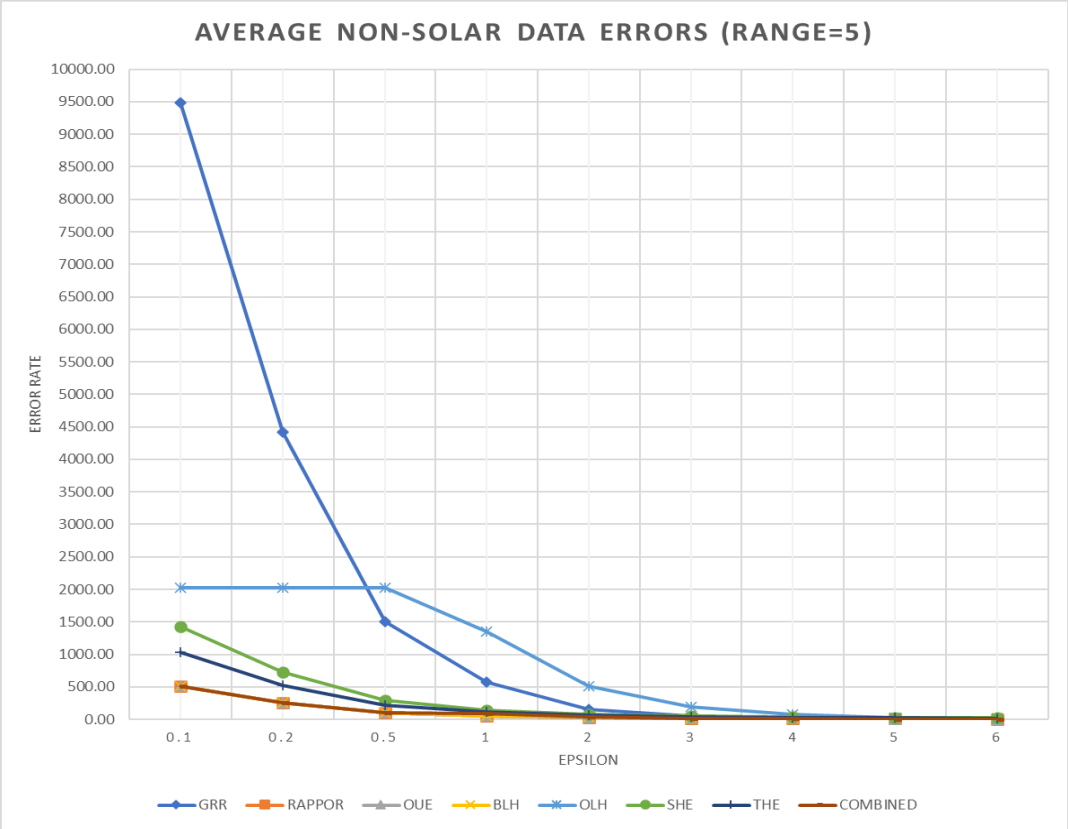

Figure 12

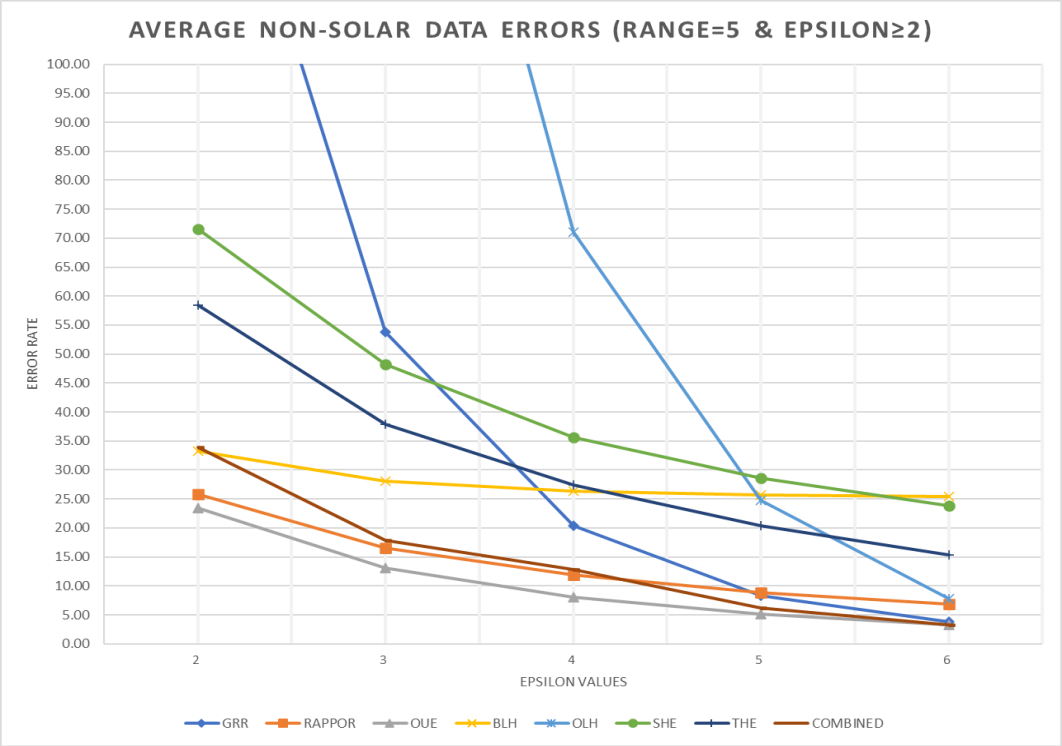

Figure 13

In addition to the comprehensive results on Non-Solar data, results from implementations performed on Solar data and London data are also shown in Figures 14, 15, 16, and 17. According to the results obtained using Non-Solar data, we discovered that the optimal value for range size was 10. Because of this reason, we demonstrated the results for other datasets when the range size is 10. Looking at Figures 14 and 15, one can see that average error values didn't change much compared to London data. Although there are slight differences in the magnitude of average errors, there is no difference in how 8 protocols performed at various Epsilon values.

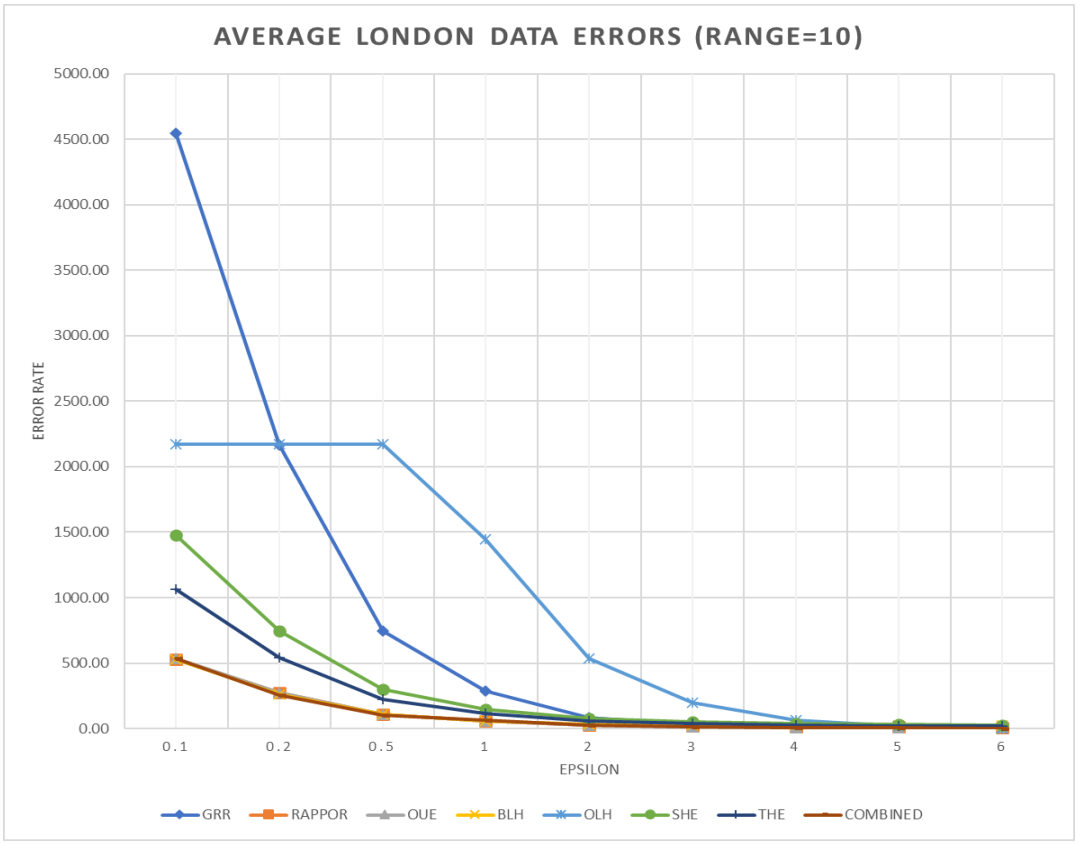

Figure 14

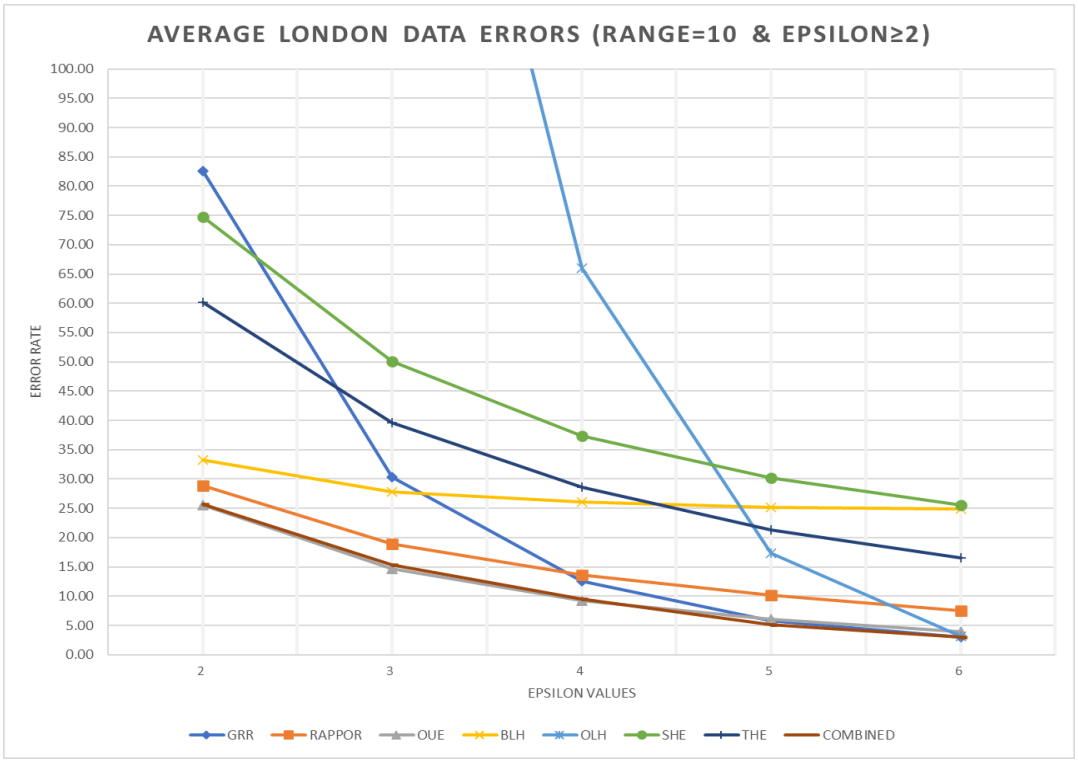

Figure 15

Besides, in Figures 16 and 17, we demonstrated the results obtained from performing protocols on Solar data. We didn't encounter major differences in terms of average error values at all. Again, as in the previous results, OLH, RAPPOR, and combined protocol have performed better for all Epsilon values, whereas OLH and GRR performed let to noticeably worse performance for smaller values of Epsilon compared to other protocols. It is not surprising that we had similar results for three different datasets since there were no major differences in terms of the profile of the data that may lead to a change in the implementation of the LDP protocols.

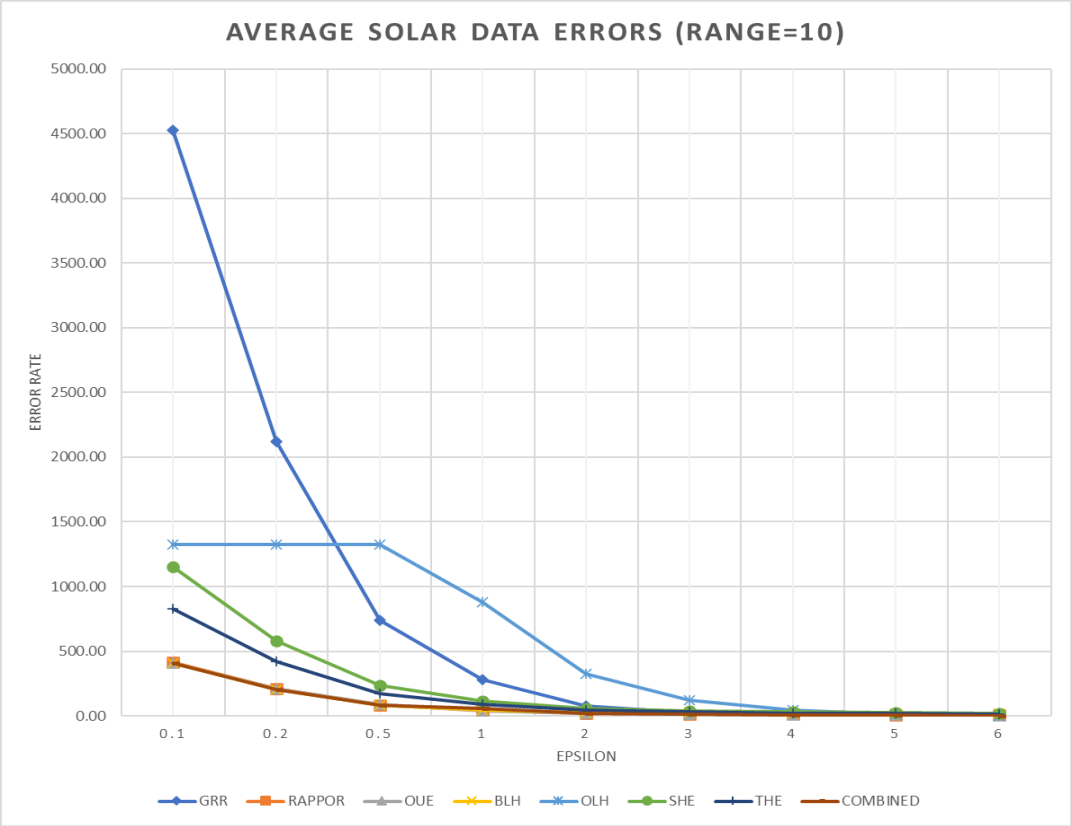

Figure 16

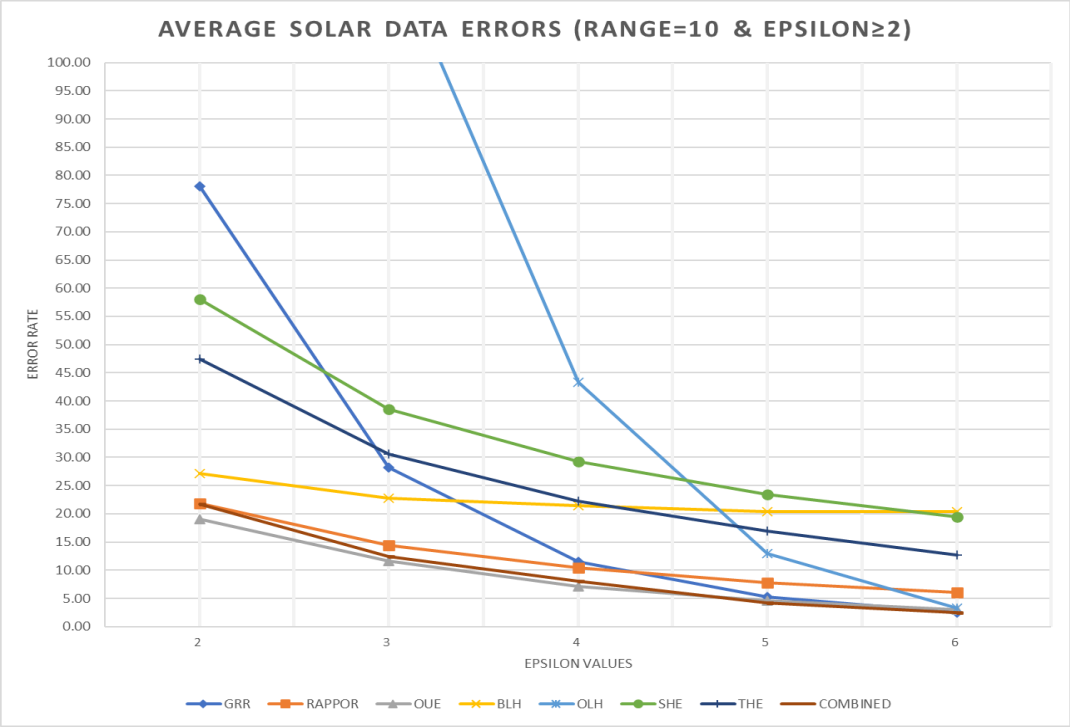

Figure 17

In addition, we compared the months of January and July of the London dataset which can be seen in Figures 18, 19, 20, and 21 to determine whether the error values of the protocols differ in the monthly data. The coldest month in London is January, and the hottest month is July. We thought there might be more unique values during the hottest and coldest months because of increased electricity use for heating and cooling. For this reason, we chose January and July to compare. However, when we compared the graphs, we could not see any difference in the error values of the protocols. In both months, we obtained close error values for all Epsilon values.

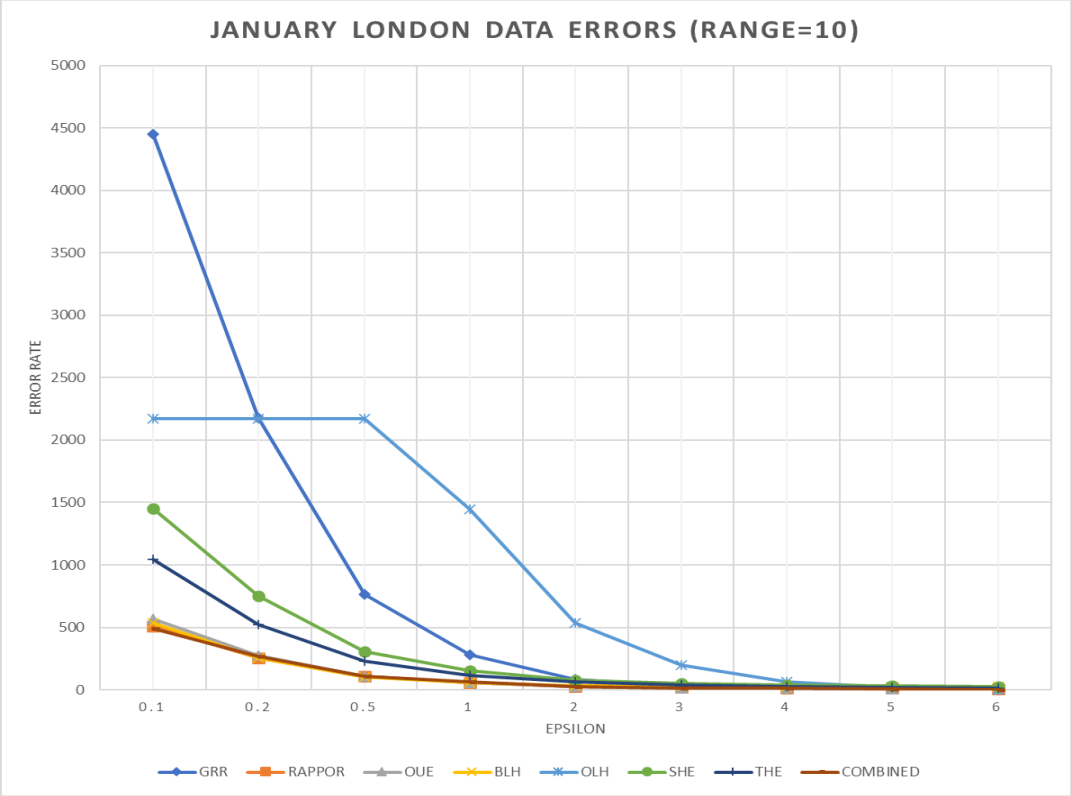

Figure 18

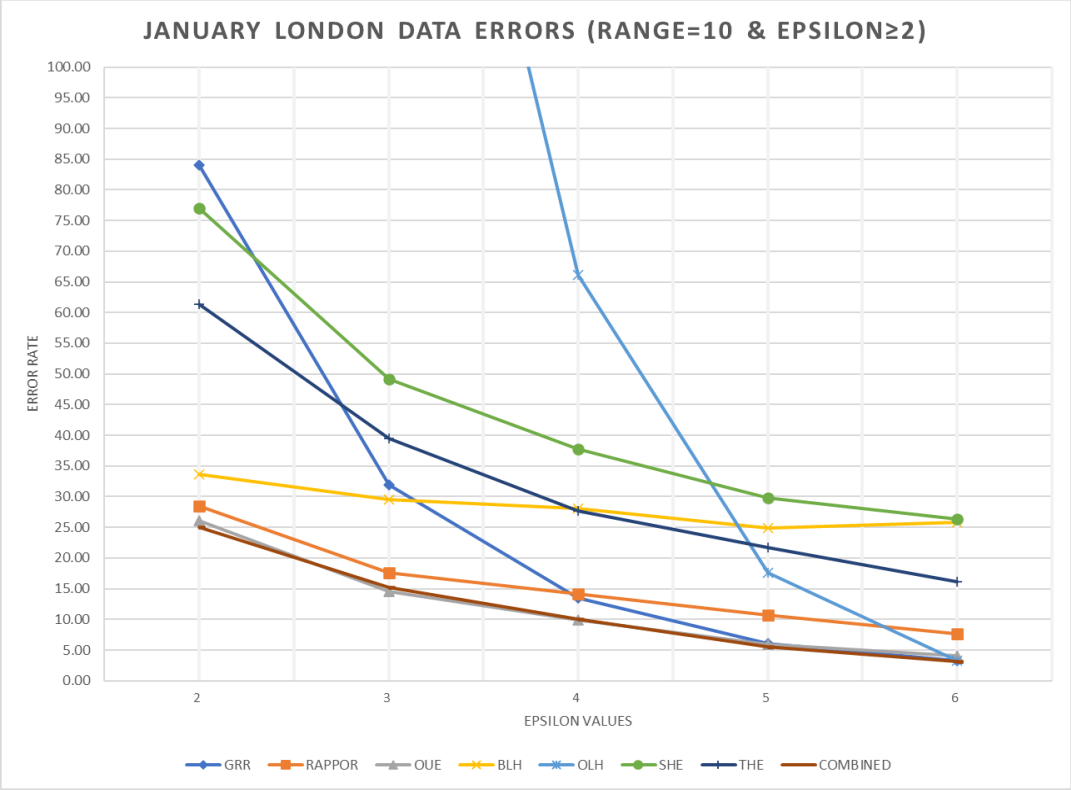

Figure 19

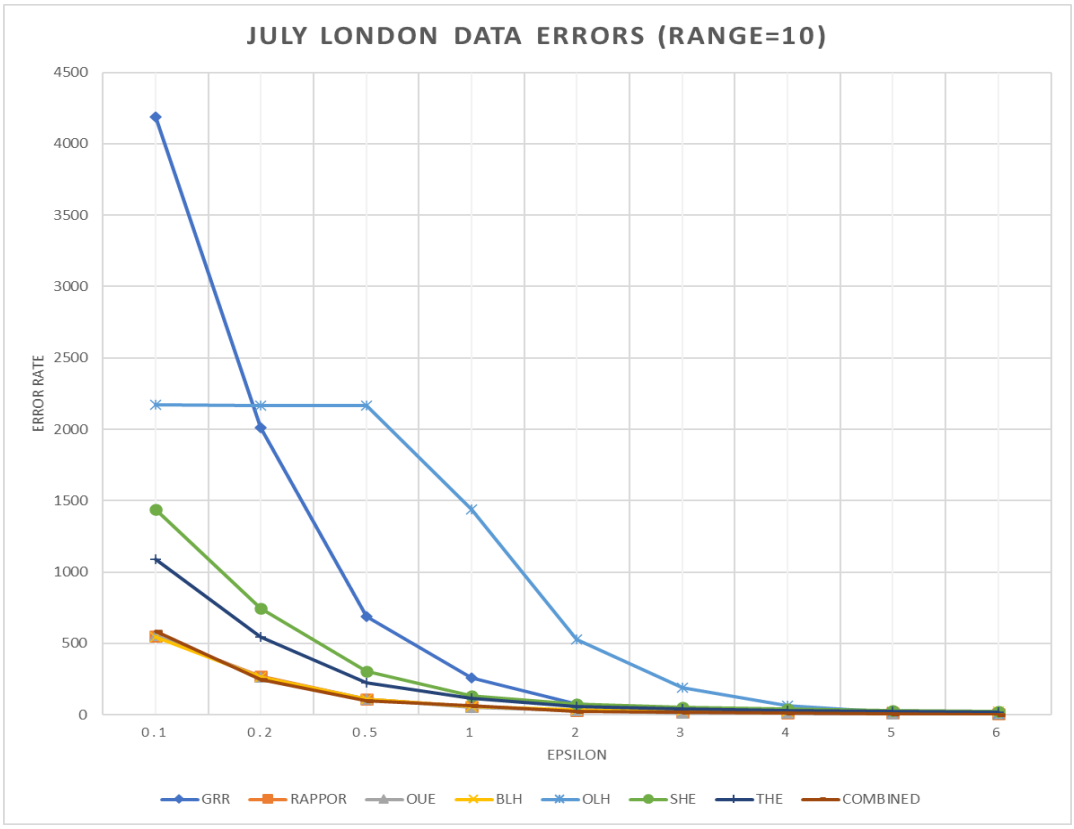

Figure 20

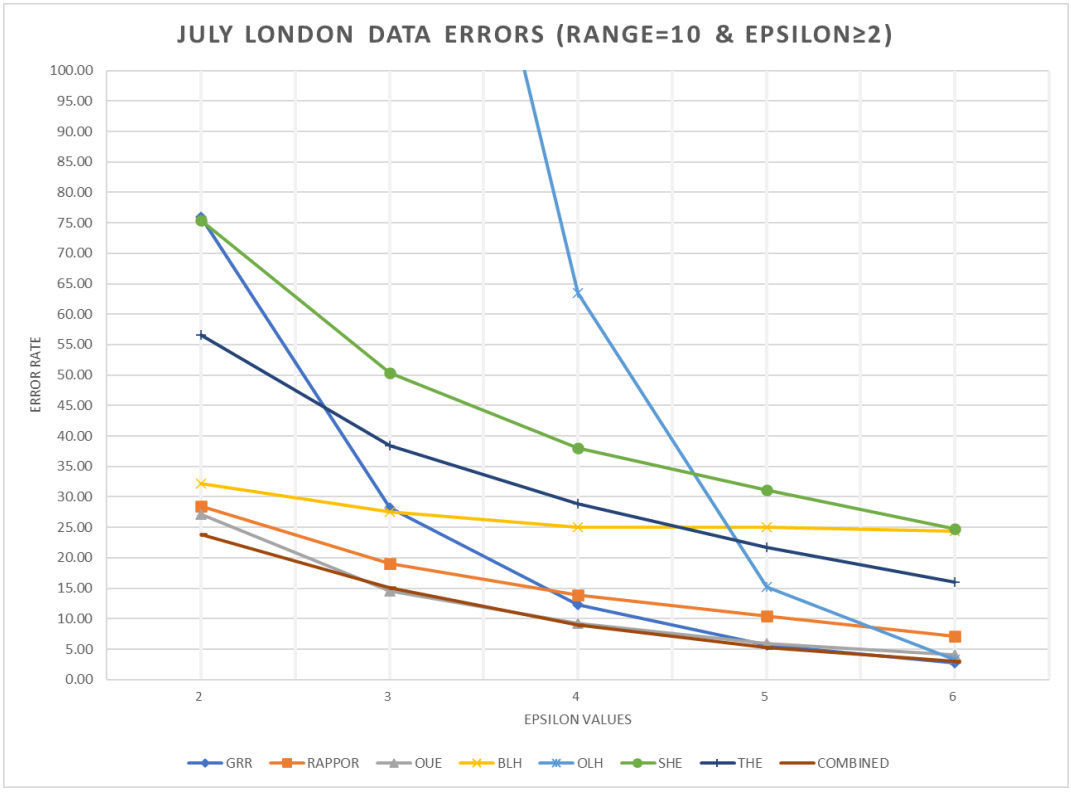

Figure 21

Furthermore, we wanted to examine the effect of the number of groups on the error by changing the range values. As you can see in Figures 22 and 23, in cases where we increase the Range value, our group number decreases accordingly. First of all, the error rate increases when the number of groups increases for GRR. For this reason, before examining the results, we predicted that our error rate would progress in the same way for the remaining protocols due to the increase in the number of groups, but we realized in the experiments we performed that this was not the case. We saw that the number of groups did not have a direct or inverse ratio on the error rate. Another thing that we realized as a result of the experiments is that there is an optimal range for our datasets. We saw that this value is 10 for our Non-Solar dataset.

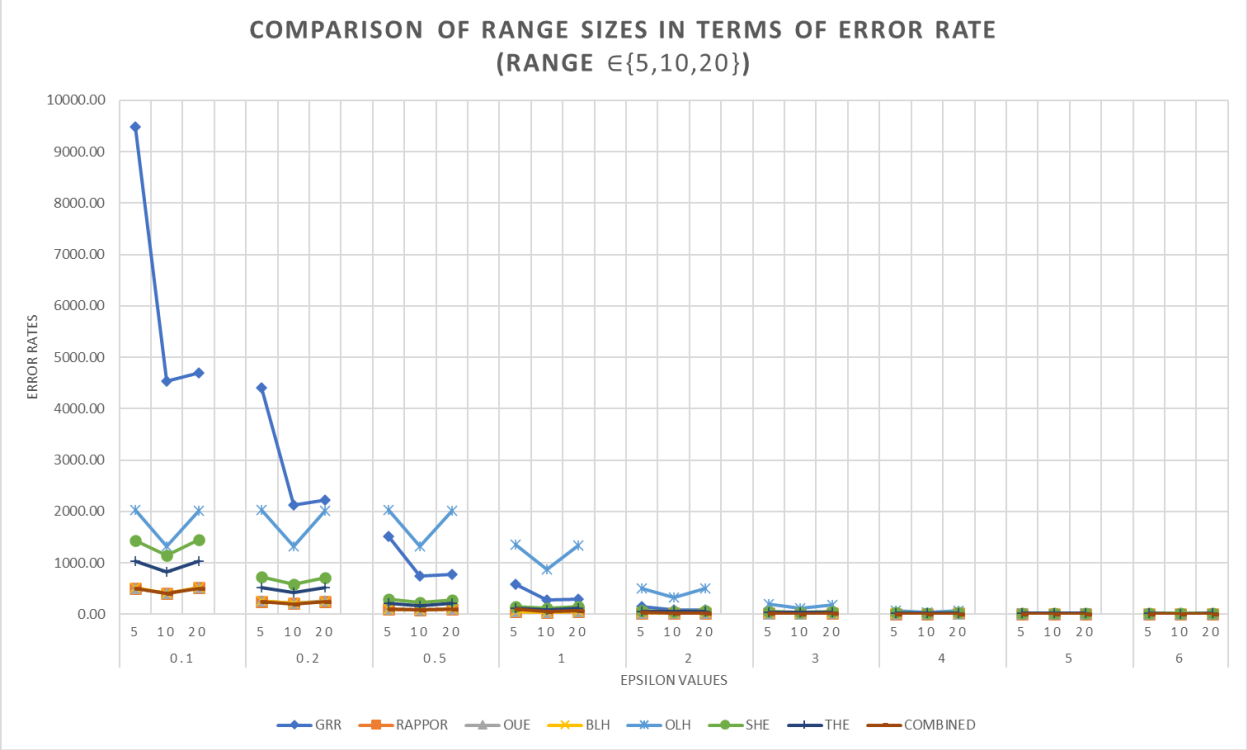

Figure 22

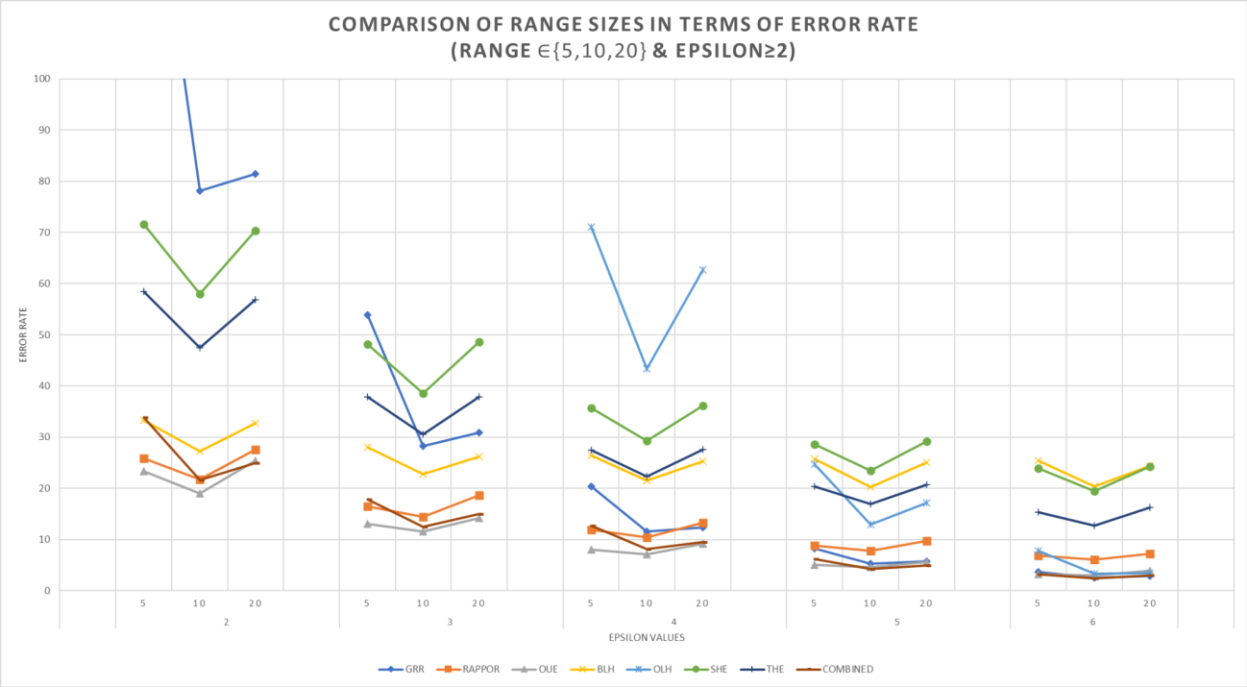

Figure 23

Finally, we compared the 8 LDP protocols we implemented in terms of runtime. The specifications of the computer we used while testing our protocols were 16 gb ram intel i7 gtx1650. Firstly, we tested the protocols with different Epsilon values on monthly data. Seeing that the Epsilon value did not affect the algorithm's running time, we ran each protocol for January 2007 of the London dataset and calculated the average running time for each protocol. As you can see in the Figure 24, the protocol with the shortest running time is the SHE protocol. The protocol with the next shortest running time is THE protocol. Due to the addition of a thresholding step to the SHE protocol, THE has a longer running time, but it still takes less time than other protocols. GRR is slightly efficient in terms of time, compared to OUE and RAPPOR however there are slight time differences between the GRR, OUE, and RAPPOR protocols. Finally, our blended protocol has the longest running time among the protocols. Although we achieved a low error rate for each different epsilon value, unfortunately, we could not be efficient in terms of running time.

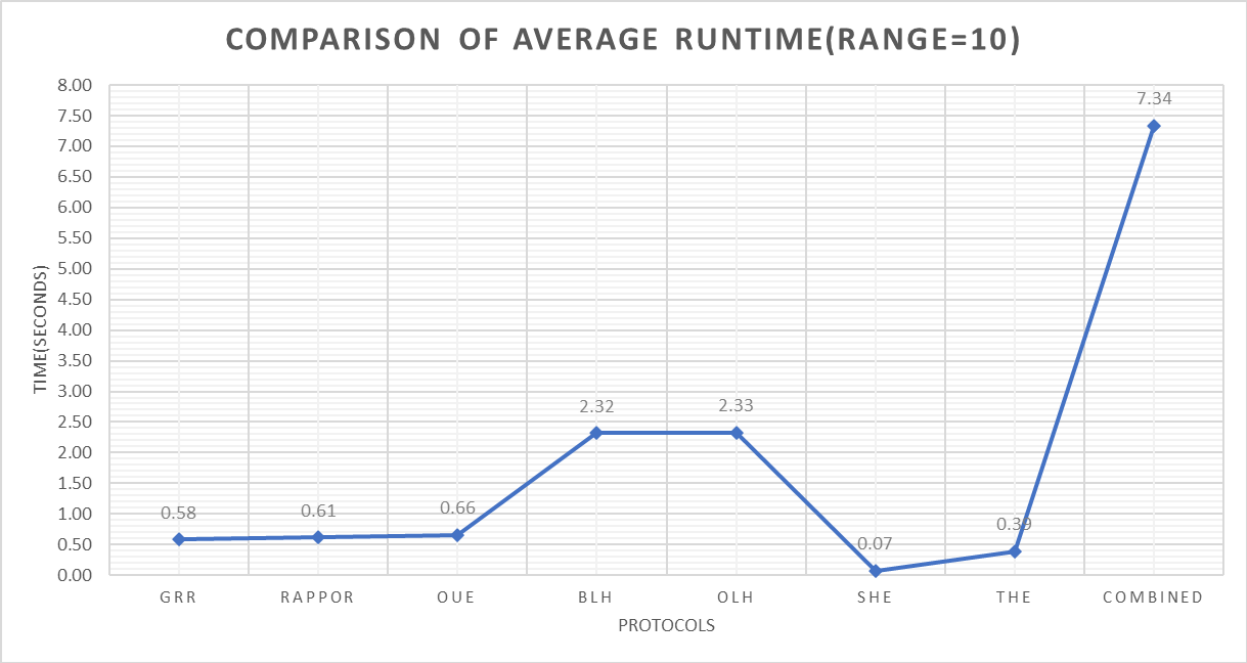

Figure 24

4. Conclusion

Overall, this benchmarking project provided a comprehensive comparison of various LDP protocols applied to different datasets. In total, we applied 7 different LDP protocols to 3 different numerical datasets in this study. After considering the various performances of these protocols, we developed a blended protocol that provides higher accuracy while maintaining privacy for the various Epsilon values. Although the error rates of the protocols varied according to Epsilon values, the protocol that gave the most accurate overall results was OUE. Furthermore, based on the datasets we have worked with, we determined that 10 is the ideal range size. We also observed that the SHE protocol was the fastest of all the protocols we implemented. In conclusion, this project compared 8 LDP protocols with different parameters and error rates and tried to find the most optimal one.

References

Wang, Tianhao & Blocki, Jeremiah & Li, Ninghui & Jha, Somesh. (2017). Optimizing Locally Differentially Private Protocols.