MDD 2022 Summer School

Data Security and Privacy for Outsourced Data in the Cloud

Lab Session

Tristan Allard (Univ Rennes, CNRS, IRISA) & Amr El Abbadi (UC Santa Barbara)
tristan.allard@irisa.fr

Prerequisites

- ⚠ You must be able to run `docker` images.
- ⚠ You must have a working connection to the web in order to perform the setup. Once the setup is done, it is possible to work offline (although being able to access documentations and forums might help).
- ⚠ You must have a basic knowledge of programming (e.g., loops, data structures, objects). The assignment is expected to be doable without any knowledge of `Python` if you can access online documentations and forums. Of course, having already programmed with `Python` might help.
- ⚠ You will feel more comfortable if you have a good `Python` IDE. But only a text editor is really necessary :)
- ⚠ After editing `RunMe.ipynb` in your IDE, reload it in Jupyter Lab (« File → Reload Notebook from disk »).
- ⚠ After editing `pinedrq.py` in your IDE, relaunch your Jupyter Notebook kernel and launch all the cells (« Run → Restart kernal and Run all cells »).
- ⚠ **We recommend you to perform all the changes to the code through your IDE. You will be able to update `RunMe.ipynb` through Jupyter Lab from your browser, but your changes might not be persistant due to the use of docker !**

1 Setup

1. Go to https://gitlab.inria.fr/tallard/mdd2022-public.
2. Clone the repository (with `HTTPS`) or download it and uncompress it.
3. Open a shell in the resulting local directory.
4. Run the following `docker` command :
```
docker run -ti --network host \
-e NB_UID=$UID -v $(pwd):/home/jovyan/work \
registry.gitlab.inria.fr/tallard/mdd2022-public:latest
```
5. Launch your favorite web browser and go to the URL that is prompted in the shell : http://127.0.0.1:8888/lab?token=<token>
6. Your browser is now displaying Jupyter Lab. You can open and execute the `RunMe.ipynb` notebook listed on the left in the `work` directory.
7. Launch your favorite IDE and open : `pinedrq.py` and `RunMe.ipynb`.

2 Introduction

During this lab session you will perform an experimental study of the tradeoff proposed by `PinedRQ`. You will work with a Python implementation of (static) `PinedRQ` for indexing the `adult` dataset, a dataset commonly used in privacy-preserving related works. But ...The implementation is incomplete! Encryption and perturbation have been simply skipped, and the query processing strategy is overly naive (it returns the full dataset at each query). Before performing your experimental study you will thus have to fix these issues. You are however lucky, the culprit wrote the corresponding `TODO`s at the places where the code needs to be completed...

Before going to the code, we will start with a short *pen and paper* study of the differentially private perturbation of the hierarchy of histograms of `PinedRQ`.

3 Pen and Paper Preliminary Study

3.1 Reminder

We assume in the following that PinedRQ indexes the `age` column. The hierarchy of histograms is a tree designed as follows :

— Each node is equivalent to a bin in a histogram. It consists in (1) a range over `age` (e.g., $[19, 32]$) and (2) the count of records falling within the range (e.g., 76).

— The range of a node is the union of its children's ranges.

— The range of each leaf corresponds to the smallest possible range (e.g., $[19, 32]$).

— The ranges of the nodes that are at the same level do **not** overlap.

— There are at least two levels, i.e., the leaf nodes and the root node.

— The structure of the hierarchy (i.e., number of leaves and degree of the tree) is fixed [1].

3.2 Questions

1. What is the sensitivity of the count of a single node ?
2. Identify the composition property that applies to the nodes of a single level. What is the sensitivity of a single level ?
3. Let ϵ_l denote the privacy budget dedicated to level l. What is the scale factor of the Laplace distribution that will be used for perturbing the count of each node at level l ?
4. Identify now the composition property applies to two distinct levels ? What is the impact on the management of the total privacy budget ϵ_{total} ?
5. Could we post-process a perturbed hierarchy of histograms in order to make the count of each parent node consistent with the counts of its children ?

4 Go To Code

4.1 Warm-up

6. Run all the cells of `RunMe.ipynb`.
7. Inspect the code and identify the parameters values and the goal of the example experiment that is performed.
8. Observe the graphs at the bottom and explain the reason why the recall is perfect and the precision extremely low.

1. Hence, it does not depend on data.

4.2 Encryption

9. Inspect `pinedrq.py` and spot the function in which the secret key and initialization vector must be generated, and the function in which the records must be encrypted.
10. Generate the secret key and the initialization vector (use a 16-bytes key length).
11. Encrypt the records based on the `AES` cipher in `AES.MODE_CBC` mode (use the API provided by the `pycryptodome` module). Do not forget to `pad(\cdot)` the sequence so that its size is a multiple of `AES.block_size`.
12. Spot the function in which the encrypted records must be decrypted. Be careful with following the exact reverse path followed by the encryption function.

4.3 Perturbation

13. Now spot the function in which the counts of the nodes must be perturbed.
14. Compute the sensitivity of a count in a hierarchy of histograms (go back to Section 3).
15. Assume that you distribute the privacy budget ϵ uniformly along the levels (i.e., all levels are given the same privacy budget). Compute the privacy budget ϵ_l per level l.
16. Perturb the count of each node by adding to it a random variable sampled from the Laplace distribution well parameterized (you can use the API provided by the `scipy` module).
17. You can truncate negative values to 0. Which differential privacy property guarantees the security of this operation ?

4.4 Query Processing

18. Spot the function in which the cloud processes a query.
19. Design and implement a query processing strategy that improves precision (at the cost of the recall). Warning : keep the naive query processing strategy somewhere. It will be useful as a baseline.

5 Stepping Back

20. Visualize the impact of your query code on the recall and precision obtained (simply run Step 3 and Step 4 in `RunMe.ipynb`).
21. What do you observe by comparing the graphs together ? How do you explain the variations ?
22. Same question but compared to the graphs obtained by the naive query processing strategy.

Acknowledgments

We warmly thank Louis Béziaud (Univ Rennes, CNRS, IRISA, UQÀM) and Matthieu Simonin for their valuable help on this lab session.