CS-523 Project 2
SecretStroll

You are an avid user of location-based applications such as Foursquare, Yelp, and Google Maps. These applications provide users with details and reviews of nearby Points of Interest (POIs) based on the users' location. You have installed and logged into Foursquare on your smartphone, and you always keep your location on to receive notifications about nearby places. However, you recently read an article on location tracking, [1] and it terrified you. Therefore, you decide that not only are you going to stop sharing your location all the time, but you are also going to build a better, more private, location-based service.

You would like your application (app) to work as follows: When a user opens the app, the app authenticates with the service provider. After authentication, the app sends the user's location to the service provider. The service provider returns nearby POIs. Figure 1 shows an overview of the application.

Because your app is privacy-preserving, you do not want to earn revenue by monetizing users' personal data. Instead, you plan to use a subscription model. To make the app more specialized and affordable, you decide to implement interest-based subscriptions. For example, Bob can get a monthly subscription for restaurants ($2.50), gyms ($1.99), and dojos ($1.50), for a total of $5.99. The user selects a subset of subscriptions when sending the location to the server so that only POIs of that type are retrieved.

As you learn in the CS-523 lectures, there are three main information leaks in this app which could become privacy risks:

1. Disclosing the sender of each query through authentication creates a time-stamped list of visited places for each user, and undermines their privacy. The adversary, in this case, is the service provider.

2. Disclosing user location and subscription data in each query enables the service provider to infer information about users' behaviour. The adversary, in this case, is the service provider.

3. Network traffic meta-data between the user and the service provider can reveal information about the query, and therefore about the users' whereabouts. The adversary, in this case, is someone who eavesdrops the network connection between the user and the service provider.

[1] `https://www.nytimes.com/interactive/2019/12/19/opinion/location-tracking-cell-phone.html`

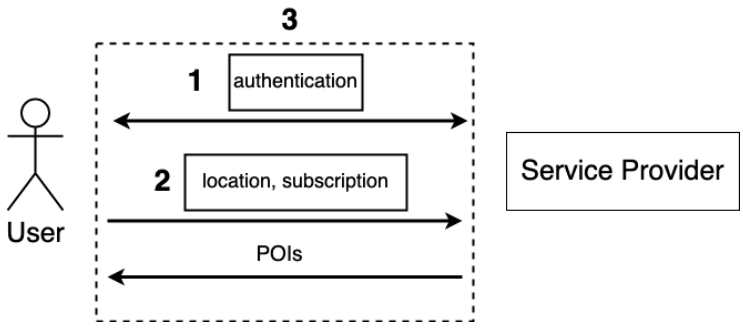

Figure 1: Overview of your location-based system. There are three information leaks to address. The numbers indicate where each leak occurs, and the project part corresponding to the leak.

We illustrate the points in which the leaks happen in Figure 1. This project is divided into three parts which tackle each of the information leaks mentioned above in each part.

Project Objectives

In this project, you have to do the following:

- **Part 1.** Design and implement an anonymous authentication mechanism using attribute-based credentials.
- **Part 2.** Conduct a privacy evaluation of a location-based service. Propose and evaluate a privacy defence.
- **Part 3.** Implement and evaluate a network-traffic fingerprinting attack when a user makes a location query.

Deliverables

By the due date specified on Moodle, you must submit:

- A PDF file with your IEEE-format two-column report of **at most 5 pages** (excluding references). The report template is on Moodle. The sections for each part contain the requirements for the report content.
- A `zip` or `tar.gz` code archive. The archive should contain three folders: `part1`, `part2`, `part3`. Each folder should contain all the code for the respective project parts. Code quality is *only* graded in Part 1, and we will not use code quality for grading in Parts 2 and 3. However, all code needs to be working, and the documentation for each part needs to include README files containing the instructions for running all the runnable artifacts in the respective folders.

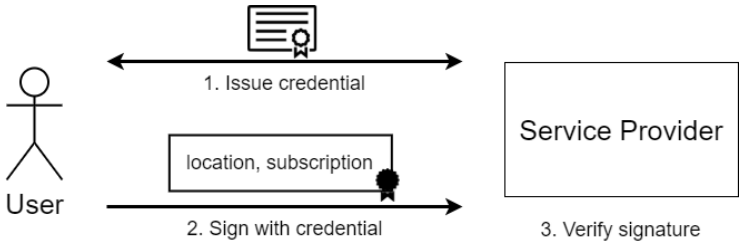

Figure 2: Using attribute-based credentials as an authentication mechanism.

Suggested Timeline

You have six weeks in total for this project. A suggested timeline:

- Weeks 1–2: Complete attribute-based credentials (Part 1), and write the corresponding report section.
- Weeks 3–4: Complete (de)anonymization of user trajectories (Part 2), and write the corresponding report section.
- Weeks 5–6: Complete cell fingerprinting (Part 3), and write the corresponding report section.

We **strongly** suggest that you finalize the code submission and the report for each part right away as soon as you finish working on it, as opposed to finalizing the archive and the report right before the deadline.

1 Attribute-Based Credentials

A common practice in location-based apps is that the app asks users to sign up. Users needs to be logged in to search, and search requests are linked to the user's account. In other words, the application creates a profile of visited places for each user. This is a privacy risk (Figure 1, box 1). SecretStroll does not require the user's identity to search for nearby POIs. An easy fix for this privacy risk is not asking for the user's identity in search! However, SecretStroll is not a free service and it requires proof of an active subscription. In Part 1 of the project, we use attribute-based credentials (ABCs) to address this problem.

When users sign up, the SecretStroll server issues a credential for them (step 1 in Figure 2). Unlike classic authentication systems, there is no "login" protocol in SecretStroll. Each user signs each search request with his/her credential to prove ownership of an active subscription (step 2 in Figure 2). The properties of ABCs ensure that different signatures are unlinkable. The SecretStroll server verifies signatures before processing their corresponding messages (step 3 in Figure 2).

You decide to use the Pointcheval and Sanders (PS) attribute-based credential scheme [1]. You have seen this scheme in the CS-523 lecture slides (ABC

lecture, week 4). The original paper gives a detailed description of the scheme. The paper [1] designs a randomizable signature in Section 4. Section 6.1 (Signing a committed message) is equivalent to issuing a credential, and Section 6.2 (Proving the knowledge of signature) explains how to prove possession of a credential.

You have to implement PS attribute-based credentials, integrate them into the SecretStroll system, test the ABC, and evaluate its performance as Part 1 of the project.

Warning: The PS paper [1] does not provide a step by step "how to build an ABC". This means that you have to put your zero-knowledge knowledge to use, and fill in the missing steps. We recommend you to read the paper early on and compare it with the lecture slides to make sure you understand the steps.

Hint: You need to prove the ownership of a valid credential when signing with a credential. What heuristic helps you in making a zero-knowledge proof non-interactive? Can this heuristic be used to make the non-interactive proof into signing a message?

A common ABC practice is to include a secret key in the credential as an attribute. You should follow this practice and include a secret key in the credential. (*Hint:* Users should not reveal their secret key.) Furthermore, assuming that two users A and B perform a search request and reveal the same set of subscriptions, given one of these two signatures SecretStroll should not be able to distinguish the credential used to produce the signatures or learn anything beyond the revealed attributes about the signer.

1.1 Code skeleton

To help you with the project, we have prepared a skeleton for SecretStroll which handles the communication. You need to integrate the credential with the skeleton in Part 1. Later on, Part 3 uses the finished system of Part 1 as a black-box. You are not allowed to change the internal parts of the skeleton, but following the structure in `credential.py` is optional and as long as your code is compatible with the skeleton you can change or discard any part of `credential.py`.

Notes on coding. The skeleton is documented and guides you towards what you have to do. We repeat some important notes and hints in this document, but you need to read the README for further instructions.

- You are only allowed to change the `your_code.py` and `credential.py` files in the skeleton, but you can create and edit new files at will.
- The skeleton provides a command-line interface to interact with the SecretStroll system (e.g., register, query, etc.). Use `--help` or check the README file for more information.
- The skeleton uses docker containers to enforce isolation between the client and server.

- We strongly recommend to use the `petrelic` cryptographic pairing library to implement PS credentials. You can find the petrelic project in `https://github.com/spring-epfl/petrelic/` and visit `https://petrelic.readthedocs.io` for documentation. This library is bundled in the provided docker container and virtual machines.
- The skeleton handles the communication without knowing the application layer logic. The skeleton treats all communications as a byte array. This means that you have to (de)serialize your objects. We provide `serialization.py` as a 'petrelic' extension of `jsonpickle` to help you with the serialization of cryptographic objects.
- We expect you to implement the zero-knowledge proofs yourself. You are not allowed to use external libraries to help you with the zero-knowledge proofs (e.g., the `zksk` library).
- Normally the server should serve requests through HTTPS. The skeleton reduces this requirement to having simple HTTP connection to make the deployment and operation of dockers easier.

1.2 Requirements

Your report should answer the following questions:

- How did you map PS scheme to the SecretStroll system. How did you design and reveal attributes? Does this approach guarantee that minimum information necessary for the operation is revealed in each request?
- How did you use the Fiat-Shamir heuristic to make SecretStroll's zero-knowledge proofs non-interactive?
- How did you test the system? How would you assess the effectiveness of your tests? (Performing the assessment is not necessary.)
- How did you evaluate the performance of the ABCs? You need to report a fine-grained evaluation of communication and computation cost in key generation, issuance, showing the credential, and verifying. Each reported number must include both the mean and standard error of the mean.

Hint: Does your test result depend on the hardware that you are running on? What about the way you set up your testing environment?

Hint: How many measurements do you need to compute the standard error?

The code that you deliver must:

- Work without any changes in the immutable part of the skeleton.
- Follow common coding practice like providing good documentation, having comments, having good variable names, etc.
- Include tests (use pytest format). You need to test at least two failure conditions in addition to a successful run.

2 (De)Anonymization of User Trajectories

Anonymous credentials clearly improve the privacy of your application, as you do not know which users are issuing which location queries. Unfortunately, despite all the complicated cryptographic machinery that you have built, there are other information leaks that are still open. As you can see in Figure 1, box 2, the service provider knows the user's location and their subscription in every query. Even though each query is anonymous in the application layer thanks to anonymous credentials, the service provider still observes another identifying piece of metadata: the source IP address.

To evaluate the remaining privacy risks before deploying the system to real users, you want to test it using simulated data. You have simulated the behaviour of two hundred users over twenty days in one geographic area. Given these simulation results, you wonder how bad is the privacy leakage due to the IP-level metadata, cleartext location and subscription information, and what can you do about it.

2.1 Data Description

You get two datasets as a result of your simulation: a dataset of queries `queries.csv`, and the POI database `pois.csv`.

Dataset of Queries The first dataset contains details of the location queries issued by simulated users. It is represented as a table where each row corresponds to a query which contains the following fields:

Column	Type	Description
IP address	`str`	Origin of the query
Latitude and longitude	`float`	Location queried by the user
Timestamp	`int`	Time when the query was issued
POI type filter	`str`	POI type the user requests (e.g., restaurants, cafes, museums, clubs, etc.)

For your convenience, the timestamp format is in **hours** from the beginning of the simulated data capture. The data capture begins on a midnight between a Sunday and a Monday.

POI Database The second dataset is your current database of POIs. It is represented as a table with the following fields:

Column	Type	Description
POI ID	`int`	ID of this POI
Cell ID	`int`	Grid cell ID (see below)
POI type	`int`	POI type (e.g., cafe, museum, club, etc.)
Latitude and longitude	`float`	Location of this POI

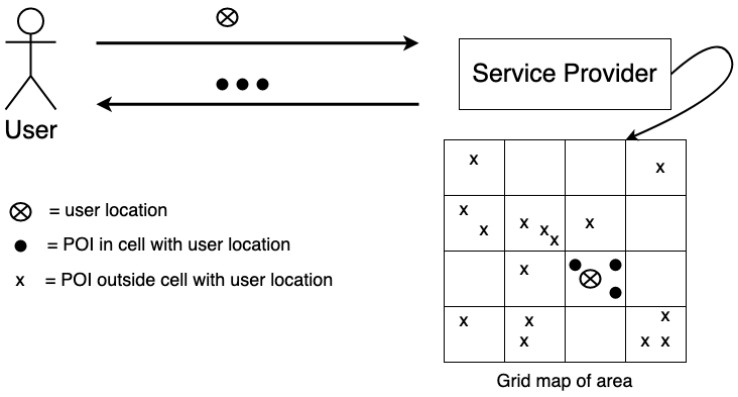

Figure 3: Obtaining POIs from user location. The provider divides the map of the area into a grid of cells, and responds with the POIs in the same cell as the user location.

2.2 Grid

Recall that in each query, a user asks the service for a list of POIs of a given type (the one they are subscribed to) in the vicinity of their current geographic location.

For efficiency reasons, in your current implementation the notion of "vicinity" is implemented as follows. You have divided the simulated area into a rectangular grid. Each cell of the grid has the same dimensions. Upon querying the service, the user receives all the POIs of the requested type *in the grid cell containing their location.* This process is illustrated in Figure 3.

We provide a file called `grid.py` containing the concrete parameters of the grid. For your convenience, the file also has a function `location_to_cell_id` that, given an input location, returns a unique identifier ("cell ID") of the grid cell which contains that location.

2.3 Privacy Evaluation

Your first task is to evaluate the privacy leakage of the service. Can you breach the privacy of simulated users in the dataset? Can you figure out where some users live, work, or what are their interests? What other information can you infer? You should assume that in this dataset *each IP address corresponds to a different person.*

Requirements

Your evaluation has to include these points:

- Specification of your assumptions and adversary models. Explain why do you think these assumptions and models are useful for the privacy analysis.

- Definition of the privacy attacks within the stated adversary models. Describe how you are mounting these attacks.
- Demonstration of the attacks, possibly with plots or tables. Discuss their severity and the cost to mount them.

It is convenient to use a Jupyter notebook for this analysis. You cannot, however, substitute the report with the notebook; even if you include it in your submitted code, you still need to write the report.

2.4 Defence

In the previous task, you have found attacks that can breach users' privacy. You still want the service to exist and be privacy-preserving, so you have to think about privacy defences for the users. A solution would be to have users homomorphically encrypt their queries and process them in the encrypted domain, but in the previous project you have learned this entails a heavy overhead. Therefore, you choose to implement a client-side defence, such as those seen in CS-523. In this task, you need to propose and evaluate one such defence. For the purpose of this task your defence **cannot** rely on users changing their IP address.

Requirements

The report has to include this information:

- Definition and description of the defence that you are proposing. Provide intuitions for why it should work: what kind of attacks it prevents.
- Experimental evaluation of the defence in terms of privacy. State your definition of privacy, and evaluate how does your defence impact this notion of privacy. Use both available datasets and the grid specification in `grid.py` for your evaluation. Remember that in a privacy evaluation you have to consider a **strategic adversary**. *Hint:* Your definition of privacy has to be quantifiable.
- Experimental evaluation of the defence in terms of utility of the service. State your definition of utility. What is the utility loss associated with using the defence? Use both available datasets and the grid specification in `grid.py` for your evaluation.
- Brief discussion of your defence. What are its privacy-utility trade-offs?

You do not need to find the best possible defence in terms of privacy-utility trade-off. What is important is that your write-up includes the points above, demonstrates your good understanding of adversarial modeling and privacy, and your ability to assess the level of privacy a defence can provide.

Non-graded Questions

We encourage you to think about these questions about your defence: What attacks it cannot prevent and do you think that is acceptable? When does it fail? What if your initial assumptions are broken? You don't need to include the answers in your report.

3 Cell Fingerprinting via Network Traffic Analysis

In the previous part, you saw that the users' IP addresses were collected, and this helped the service provider to infer some information about the user activity. As a defence, you decide to randomize user's IP addresses, so that it is harder to link a single user to an IP address. In order to effectively randomize the IP addresses, you set up the user and the service provider to communicate using Tor. Because the connection is encrypted, someone eavesdropping the network connection between the user and the service provider cannot observe the user's location queries. However, after attending CS-523, you are concerned about the possibility that an eavesdropper could use network traffic meta-data to infer the content of the queries. You want to evaluate whether such an attack is possible on your system.

In this part, your goal is to perform a network-traffic analysis attack. You have to identify the cell ID that a user is querying for, solely by analyzing the network traffic.

3.1 Task

The skeleton provided to you has a `client.py` script. You can supply a cell ID value as input to this script. The skeleton README contains instructions on how to run the script. The script simulates the behaviour of the user's app. To simplify this part, we choose to simulate that the application queries for a specific cell in the grid instead of a location (description of the grid in Section 2.2). The grid in this case consists of 100 cells in total (i.e., the cell ID value is in the range 1-100). Note that, in reality, the number of cells will be much higher. The provider returns the list of POIs in the specified cell, and data about each POI in the list. The system is shown in Figure 4. There is Tor communication between the user and the service provider. You eavesdrop the communication between the user and the entry guard in the Tor network.

You need to develop a method to identify the cell ID queried by the application from the network traffic that occurs when the query is run. (Hint: we expect you to use machine learning, and build a classifier). Note that multiple cell ID queries for the same cell might return slightly different responses every time, due to the dynamic nature of the POI information.

While you are free to study the responses returned by the server, you cannot use the response content to identify the cell, only use

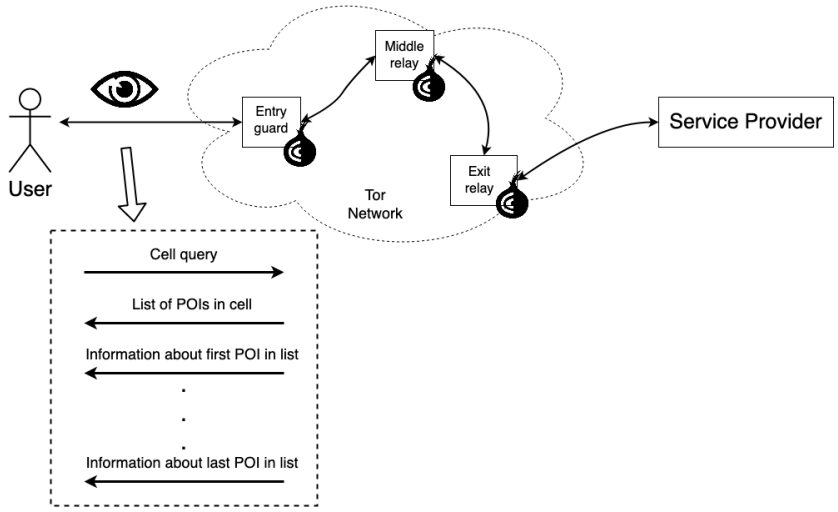

Figure 4: Scenario for Cell Fingerprinting. The app and the service provider communicate via Tor. The app sends a cell query. The provider responds with the list of POIs corresponding to the query, and data for each POI in the list.

network traffic.

3.2 Requirements

Your report should include:

- Description of the process you followed to train your classifier.
- An evaluation of your classifier's performance (evaluation should be done using 10-fold cross validation), using standard performance metrics. A discussion on how well your classifier performed and what factors could influence the performance. Note that the description of the process you followed is more important than the actual performance values of the classifier.
- A short discussion on possible countermeasures to prevent this fingerprinting.

References

[1] David Pointcheval and Olivier Sanders. Short Randomizable Signatures. *IACR Cryptology ePrint Archive*, 2015. `https://eprint.iacr.org/2015/525`.