CS-523 SecretStroll Report

Wicky Simon, Nunes Silva Daniel Filipe

***Abstract*—This project consists of three parts, all related to a central topic : Privacy in a location based service. The first part makes use of Attributes based credentials (ABC) and zero-knowledge proof to anonymously query a server for nearby Points of Interests (PoIs), while proving the ownership of a subscription. The second part we evaluate the loss of privacy induced by IP-level metadata. The last part of this project is dedicated to trying to infer location of query author, based solely on metadata.**

I. Introduction

The aim of the first part of this project is to build a privacy-friendly location based service, SecretStroll, which allows users to query a database about nearby PoIs while remaining anonymous. Since this service is not free, every query must provide the proof of an active subscription. Each PoIs is part of a category such as *Restaurant*, *Cinema*, *Train station*, etc. Each of these category are part of a subscription and a user can have many of them, which naturally leads to ABC. A user joining the service first register himself with the category he paid for and then sign each of his query anonymously, revealing only the categories he want to get PoIs from.

II. Attribute-based credential

The base of the ABC scheme for this work is described in Section 6 of [1]. This scheme has been used as follows :

- The messages mentionned in [1] are attributes, representing categories of PoIs. The server creates a public key large enough to encode all of these attributes.
- When a user wants to register, she commits to attributes she paid for, that will remain hidden. She uses a zero-knowledge proof (ZKP) to prove that she did so correctly. The server checks the proof, and sign this commitment that she will then use as credential.
- When a user makes a query, she choose the attributes (i.e the categories of PoIs) that she want. She then makes use of her credential to prove she has a subscription for the queried attributes, using a ZKP. She also uses this credential to sign a message representing her current location using the Fiat-Shamir heuristic described in [2] and detailed later.

In order to make the ZKP non-interactive, we use the Fiat-Shamir heuristic. The challenge that the prover would send is replaced by a cryptographic hash of all known values. During the registration, the challenge is composed of the committed attributes, the commitment used for the ZKP and the server's public. To sign the message for a query, it is simply added to the other values constituting the challenge.

This overall approach leaks only the number of attributes chosen, but it is necessery for the server to correctly compute the ZKP. Otherwise, hidden attributes cannot be inferred and every credential is unique.

A. Test

The system is tested with six different tests :

1) *test_generate_ca()* : The server's certificate generation is tested with a simple structure test, to see if all keys have the right lenghth.
2) *test_credentials_difference()* : Two requests are created with the same input parameters, a registration is perforemd with both of them and two credentials are created. To preserve anonymity, the requests and the built credential should be different.
   Two credentials created from the same request should also be different.
3) *test_tampered_credential()* : A request with invalid credential should be rejected. In this test, a valid credential is tampered to be invalid.
4) *test_correct_credential()* : A request with a correct credential should be accepted.
5) *test_correct_credential_no_attributes()* : A request revealing no attributes should be valid.
6) *test_wrong_revealed_attr()* : A request revealing unobtained attributes should be invalid.

The effectiveness of these tests could be assessed using a standard metric such as branch coverage or coverage. However, these metrics are not complete and will not discover every bug present in the code.

B. Evaluation

Evaluate your ABC: report communication and computation stats (mean and standard deviation). Report statistic on key generation, issuance, signing, and verification.

The system is benchmarked in two parts, an offline part, which evaluate pure computationnal time and an online part which evaluate the communication time.

1) Offline benchmark: The offline benchmark tests four features :

- Key generation
- Credential issuance
- Message signing
- Signature verification

The result are shown in figure 1 and reported in table I

2) Online benchmark: The onlin benchmark tests two features :

- Registration
- PoIs request

Note : This benchmark has requirements. Follow the instructions at the beginning of *benchmark_net.py*.

	Rounds	Mean	Std. deviation
Key generation	1000	27.30 ms	4.30 ms
Credential issuance	1000	41.86 ms	8.24 ms
Message signing	1000	44.34 ms	10.32 ms
Signature verification	1000	44.31 ms	9.52 ms

TABLE I: Result of the offline benchmark,

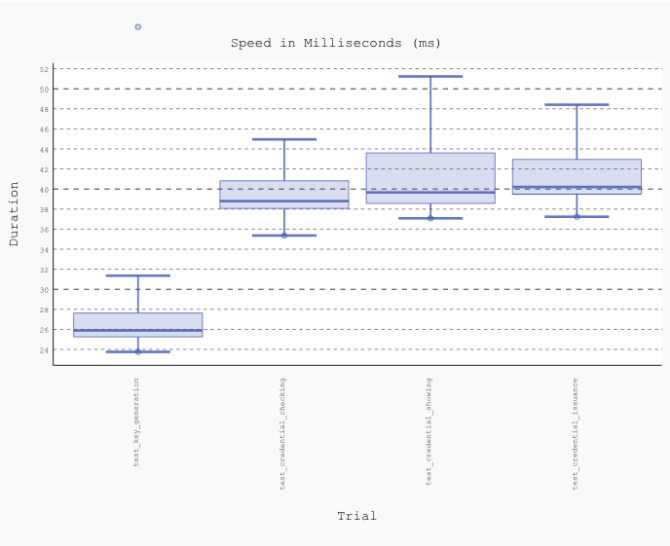

Fig. 1: Histogram of the offline benchmark

The result are shown in figure 2 and reported in table II

III. (De)Anonymization of User Trajectories

A. Privacy Evaluation

We evaluate the privacy risks using simulated data of two hundred users who made use of the application hundred times each in average over twenty days. We assume that no mechanism to hide any kind of data is used, i.e. data is sent in cleartext inside standard IP packets. Any malicious adversary could hack the application servers or sniff the network between a user and the server in order to retrieve similar datasets which include IP addresses, locations, query types, timestamps and responses. According to [3], the IP address or a set or IP addresses are relevant attributes because they can be linked to a given user since they are persistent for a certain duration. Moreover, combining IP addresses with location and time data makes sense since users often keep their habits that they may share with very few people [4], i.e. they may work during the day at their office, be back home in the evening and do an activity at some specific location. Therefore, we deduce that this implementation leak senstive informations about the users. As a proof of concept for breaching users privacy, we try to infer work and home locations of users as well as habits in their activities.

	Rounds	Mean	Std. deviation
Registration	1000	53.58 ms	1.93 ms
PoIs request	1000	90.07 ms	1.90 ms

TABLE II: Result of the online benchmark

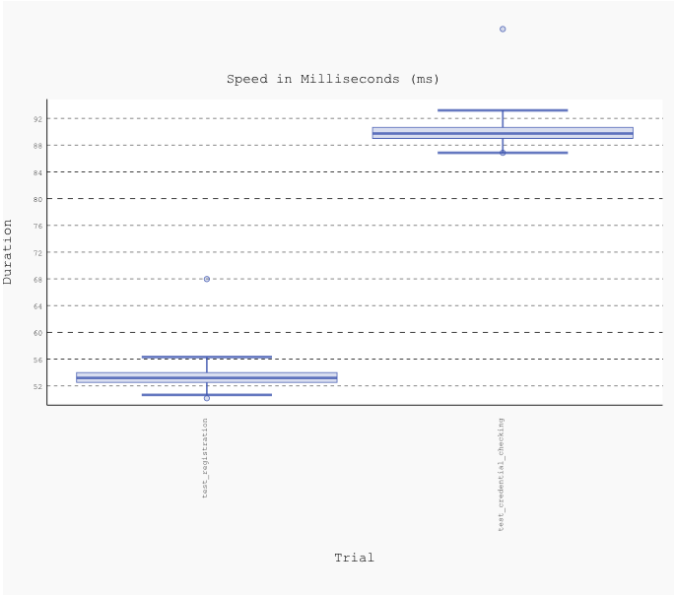

Fig. 2: Histogram of the online benchmark

We build our attacks in a constructive way, starting by learning general tendencies about the users, then, we focus on one specific user in order to learn how far we can go with breaching breach its privacy.

As stated before, home and work are sensitive locations because they allow to identify users. In order to learn about them, we go through all users queries and check if users queries locations correspond to home types locations such as *appartment_block* or *villa* as well as work types locations such as *office*, *laboratory* or *company*. It turns out that each user has one location of each from where it does a majority of its queries which allows us to draw a map linking IP addresses to their corresponding home and work location. Moreover 3 confirms our concern about users anonymity once their home and work locations would leak in the given setup. The logarithmic scale clearifies the orders of magnitude between people who share or not these POIs showing that the majority of the users have a tuple home and work location that they share with very few people as presented by [4].

Then, we analyze when users make use of the app from their home and work locations. On the one hand, we observe that they use it regularly on a daily basis from home. On the other hand, we can see that they query the service from their work location only during weekdays, which is compatible with not working on the weekend. Moreover, we consider when the app is used during the day at work in 4. This shows that users likely spend their day at work while being at home at the end of the day. This short analysis gives a great understanding of users habits even though the presented results seem trivial.

Provide a privacy analysis of the dataset. You should explicitly state your assumptions, adversary models, methods, and findings.

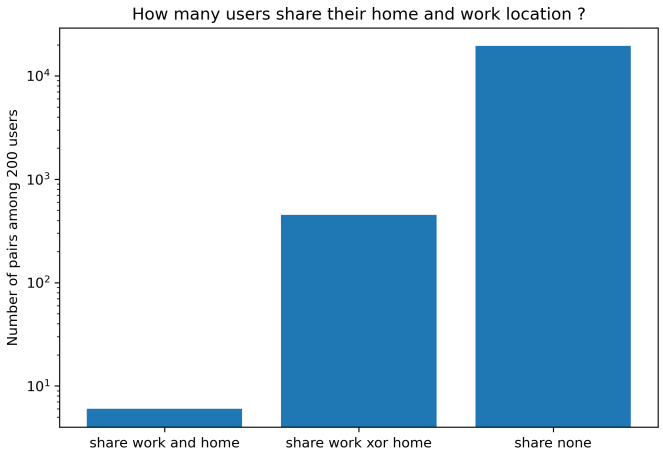

Fig. 3: Comparison of how people share their main POIs

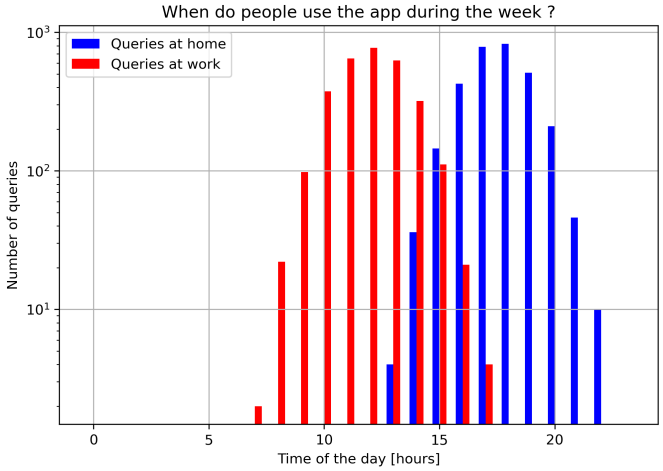

Fig. 4: User habits

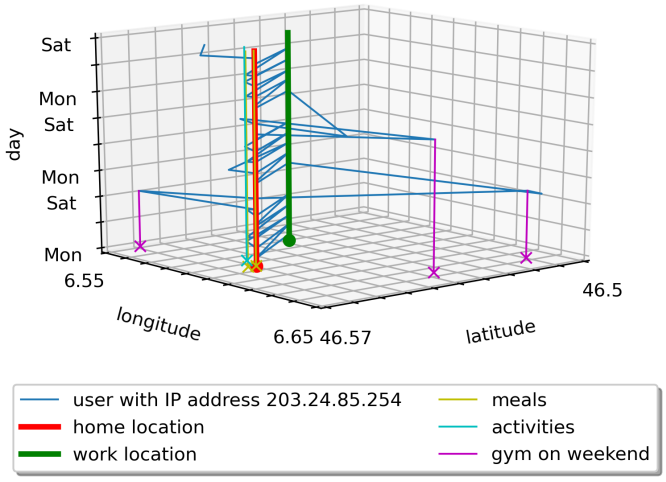

Fig. 5: Monitoring of a randomly chosen user using collecting data

B. *Defences*

Propose a defence that users of the service could deploy to protect their privacy. You should state your assumptions, adversary models, and provide an experimental evaluation of your defences using the datasets and the grid specification. You should also discuss the privacy-utility trade-offs of your defence.

IV. Cell Fingerprinting via Network Traffic Analysis

A. *Implementation details*

Provide a description of your implementation here. You should provide details on your data collection methods, feature extraction, and classifier training.

B. *Evaluation*

Provide an evaluation of your classifier here – the metrics after 10-fold cross validation.

C. *Discussion and Countermeasures*

Comment on your findings here. How well did your classifier perform? What factors could influence its performance? Are there countermeasures against this kind of attack?

References

[1] D. Pointcheval and O. Sanders, "Short randomizable signatures," Cryptology ePrint Archive, Report 2015/525, 2015, https://eprint.iacr.org/2015/525.

[2] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," vol. 263, 03 1999.

[3] V. Mishra, P. Laperdrix, A. Vastel, W. Rudametkin, R. Rouvoy, and M. Lopatka, "Don't count me out: On the relevance of ip addresses in the tracking ecosystem," *HAL*, 01 2020.

[4] P. Golle and K. Partridge, *On the Anonymity of Home/Work Location Pairs*, 2009.