

CS-523 SecretStroll Report

Rassene M'Sadaa, Luna Ralet, Leila Sidjanski

Abstract—Please report your design, implementation details, and findings of the second project in this report.

You can add references if necessary [?].

Remember to add a paragraph describing the contributions of each team member.

THE REPORT SHOULD NOT EXCEED 5 PAGES.

I. INTRODUCTION

The goal of this project is to design and evaluate a privacy-preserving location-based service that protects users against identity, location, and behavior tracking. The system enables users to query nearby Points of Interest (POIs) without revealing their identity or sensitive preferences.

To achieve this, we implement anonymous authentication using Pointcheval-Sanders attribute-based credentials, allowing users to prove subscription rights without exposing their identity. We then analyze how metadata such as IP addresses, timestamps, and POI types can still compromise user privacy through behavioral inference. Finally, we investigate the potential for an adversary to identify user locations by analyzing encrypted network traffic patterns, and we train classifiers to evaluate the feasibility of such fingerprinting attacks.

This project demonstrates the practical integration of cryptographic techniques with privacy risk analysis and machine learning to explore both the strengths and limitations of privacy protections in real-world location services.

II. ATTRIBUTE-BASED CREDENTIAL

We mapped the Pointcheval-Sanders (PS) attribute-based credential scheme to the SecretStroll system by clearly distinguishing between user-controlled and issuer-controlled attributes in the credential issuance process. Specifically, we adopted Option 5 provided in the project instructions. In this setup:

- The user holds one private attribute, their **secret key**, which they never disclose.
- The issuer holds two attributes: the **valid subscriptions** chosen by the user and their unique **username**.

Our rationale for selecting this configuration was driven by the need to minimize information leakage while ensuring system integrity and correctness:

- 1) **Secret Key Attribute:** By including a secret key as a user-controlled hidden attribute, we guarantee that credentials can only be used by their rightful owner, as verification relies implicitly on this hidden attribute. This ensures security by preventing credential misuse.
- 2) **Valid Subscriptions and Username Attributes:** These issuer-controlled attributes allow precise subscription management and uniquely link each credential to a specific user at issuance time without revealing user identity

during the showing protocol. The username ensures uniqueness and protects against credential sharing, while valid subscriptions enforce accurate service provision.

In each request to the service provider, we reveal only the subscriptions relevant to that specific query. This selective disclosure aligns with the PS scheme's objective to minimize information exposure. The secret key remains undisclosed at all times, maintaining user anonymity. Our approach ensures that:

- The minimum information necessary (subscriptions relevant to the query) is revealed during each interaction.
- User anonymity is preserved through the hidden secret key.
- Credential integrity is maintained, ensuring the credential is uniquely tied to the user and their subscribed services.

To make SecretStroll's zero-knowledge proofs non-interactive, we employed the Fiat-Shamir heuristic, converting the interactive proofs into non-interactive ones suitable for asynchronous verification. Specifically, we computed a cryptographic hash over the transcript elements of the interactive proof (such as commitments and public parameters), incorporating these values into the hash input to generate the challenge scalar deterministically. This approach effectively eliminates the need for verifier interaction, transforming what would typically require a multi-step, interactive proof into a single-step, verifiable proof without compromising security or privacy.

A. Test

To ensure the reliability and correctness of our implementation, we developed comprehensive unit and integration tests for both `credential.py` and `stroll.py`. These tests exercise each method individually and verify their integration through complete protocol paths.

The `credential.py` tests cover key generation, signature creation, verification processes, issuance requests, and disclosure proofs. Each method has a dedicated unit test validating correct behavior. Additionally, we implemented multiple negative test cases, including scenarios with incorrect messages, tampered signatures, and mismatched public-private key pairs, ensuring robustness against common failure modes and malicious manipulations.

Similarly, the `stroll.py` tests validate the end-to-end flow encompassing CA generation, registration protocol, request signing, and server verification. We implemented clear success paths verifying typical use-cases, and multiple failure paths, such as attempts to reveal unowned attributes, signature tampering, and mismatched declarations versus actual revealed attributes.

To assess the effectiveness of our tests, we considered comprehensive coverage of possible edge cases, unusual behaviors, and deliberate tampering attempts that could mimic real-world attacks. By explicitly testing for these scenarios, our approach aims to ensure the resilience and security of the implemented system.

B. Evaluation

We evaluated the performance of our Attribute-Based Credential (ABC) implementation through comprehensive benchmarking, measuring both computation time and communication overhead across the critical operations of our system. Each operation was executed 200 times to ensure statistical significance, and we calculated both the mean and standard error of the mean (SEM) for these measurements.

The table below summarizes the results:

Operation	Time (ms) \pm SEM	Size (bytes) \pm SEM
Credential Authority Generation	3.03 \pm 0.02	2126 \pm 0
Client Registration Preparation	1.79 \pm 0.01	743 \pm 0
Server Registration Processing	2.58 \pm 0.02	698 \pm 0
Client Credential Finalization	1.49 \pm 0.01	800 \pm 0
Request Signing (Showing)	11.39 \pm 0.05	1768 \pm 0
Signature Verification	11.03 \pm 0.04	-

TABLE I: Performance evaluation results of the ABC operations

These results clearly illustrate the computational and communication efficiency of each ABC operation. Credential issuance and signing (showing) are the most computationally demanding operations, whereas credential authority setup and registration phases incur relatively moderate computational costs. The absence of communication overhead for the verification step is due to the operation being entirely local, relying only on previously received data.

Overall, our evaluation demonstrates that our ABC implementation provides an efficient balance between performance and security, effectively meeting the practical requirements of privacy-preserving credential management systems.

III. (DE)ANONYMIZATION OF USER TRAJECTORIES

A. Privacy Evaluation

Assumptions and adversarial model: As stated in the assignment, we assume that each ip address correspond to a unique user. We also assume the following conditions hold:

- Users issue multiple requests to the service provider over time, allowing behavioral patterns to be established.
- The system does not apply strong anonymization techniques beyond omitting names

These assumptions correspond to how real-world apps operate: even if they remove names, they often store metadata such as query location, time, and device/IP. This metadata is retained for analytics or optimization but can also be exploited for privacy attacks.

The adversary (the service provider) aims to extract private or identifying information about the users, even in the absence

of explicit identifiers. Having access to the anonymized data, the adversary has access to :

- The number of users
- The number of queries per user
- The time of each query
- The location of each query
- The poi type of each query

The adversary can use this information to identify users based on their query patterns, even if they are anonymized. More specifically, the adversary can use the information to infer the following:

1) *Home/work location of the user*: The goal of this attack is for the adversary to identify the home or work location of the user base on their query patterns.

Assumption: The assumption done for this attack is that the user will more likely be at home/work when making query request on certain hour of the day : 9am-5pm for work, 10pm-6am for home.

Adversarial Strategy: Since the timestamps format in the database are in hours since the beginning of the survey, we first had to convert the timestamps to a more human readable format. For this we first did modulo 24 on the timestamps, and parallely we did modulo 168 to get the weekd day (and separate the week-end from the week days). They then count the most frequent coordinates within each time window and infer these as the user's primary home and work locations. This strategy leverages regular patterns in user behavior and assumes users tend to be stationary during key time periods, enabling high-confidence spatial inference.

Cost and complexity: This attack requires no special computation and relies only on filtering and counting by timestamp.

Severity: This attack poses a severe privacy risk. Home and work locations are highly sensitive. This can lead to the identification of the user. Moreover, if an adversary obtains a user's home location, they could physically visit or target the user at their actual home address. This risk goes beyond profiling and can lead to stalking, harassment, or other physical security threats.

Demonstration: The table 1 shows that, for most users, distinct locations can be consistently identified for both home and work, often with high query counts. This demonstrates that an adversary can accurately pinpoint where users live and work, with minimal effort, simply by observing their query patterns over time.

2) *The user's interests*: The goal of this attack is to identify the user's interests and preference based on their query patterns.

Assumption: This attack is based on the assumption that users will more likely query for certain types of POIs based on their interests. For example, a user who frequently queries for restaurants is likely to be interested in food, while a user who queries for gyms is likely to be interested in fitness.

Adversarial Strategy: The adversary aggregates all queries by user (using the unique ip address), then counts the number of times each POI type is queried. The POI types are then

	ip address of user	Home Coords	Home Count	Work Coords	Work Count
0	34.101.177.245	(46.54988036021286, 6.60944855546026)	3	(46.53294222140508, 6.59119086010503)	30
1	244.190.169.115	(46.54718692793925, 6.648468571463564)	7	(46.540782352688166, 6.5911895877586055)	30
2	60.235.63.253	(46.562443656835285, 6.64557145225506)	8	(46.54637740125457, 6.75535054770723)	30
3	60.109.165.215	(46.53356684429047, 6.581943720121332)	8	(46.5684471723514, 6.612007421146351)	30
4	159.118.124.69	(46.54425116348349, 6.5858751166104685)	2	(46.52941519324925, 6.583668262303588)	30
5	203.24.85.254	(46.558108868577, 6.59892712376769)	8	(46.5359190517015, 6.57548739371415)	30
6	82.230.180.56	(46.5059605841714, 6.64222386125065)	8	(46.52578117179067, 6.600312404721805)	30
7	58.9.101.51	(46.56853325936272, 6.60127105009158)	7	(46.54849046584162, 6.615589389036846)	30
8	118.135.166.110	(46.50716497818096, 6.61165138355449)	6	(46.5200043699363, 6.61385858630442)	30
9	146.71.112.211	(46.52915570266951, 6.627355926244209)	6	(46.5359190517015, 6.57548739371415)	28

Fig. 1: Inferred home and work locations (for 10 users), identified from their most frequent query coordinates during nighttime (home) and daytime (work) hours.

ranked by frequency to infer the user’s dominant preferences. This can reveal hobbies, routines, and potentially sensitive characteristics.

Cost and complexity: This attack is extremely simple to mount, requiring only a group-by and count operation over the query logs. It can be done in seconds and scales linearly with the number of users.

Severity: The privacy risk is that the adversary can infer the user’s interests and preference based on the query patterns and can use this information to target ads or content to the user, which can lead in discrimination. It can also expose sensitive information about the user’s life, such as their health status, religious practices and beliefs and more.

Demonstration: Table 2 presents, for a selection of users, the POI type that was most frequently queried. By counting queries per POI category, an adversary can infer each user’s dominant interests or preferences—such as a strong preference for restaurants, gyms, or specific leisure activities. This enables the construction of user profiles even when the user’s real-world identity is unknown, revealing behavioral patterns and potential preferences that could be exploited for targeted advertising or other forms of profiling.

	ip_address	poi_type_query	count
0	0.98.248.97	cafeteria	29
6	10.229.150.53	dojo	21
10	100.255.65.73	cafeteria	31
17	101.193.212.180	restaurant	35
21	103.107.27.105	restaurant	37
24	103.75.228.13	restaurant	36
26	104.149.206.168	cafeteria	35
30	105.148.239.144	cafeteria	37
35	106.44.169.163	cafeteria	36
39	107.201.148.122	cafeteria	33

Fig. 2: Inferred primary interest for each user, based on their most frequently queried POI type.

3) The user’s habits and behavioral patterns: The goal of this attack is to identify the user’s temporal activity habits to understand when they are most active. **Assumption:** Users tend to issue queries at times that reflect their daily routines. For example, a user who regularly queries in the late evening may be a night owl, while another who frequently queries in the early morning may follow a more traditional work schedule.

Adversarial Strategy: The method for this attack is similar to the one used for the home/work location attack and consist in counting the hours of the day when the user made the most queries.

Cost and complexity: This attack is extremely simple to mount, requiring only counting operation over the query logs. It can be done in seconds and scales linearly with the number of users.

Severity: This attack reveals a user’s behavioral rhythm (e.g., sleep cycle, daily routine). When combined with spatial data, it can help infer occupation (e.g., night-shift worker), family situation (e.g., early-morning school runs), or lifestyle habits (e.g., nightlife activity). These inferences can compromise anonymity or be exploited for profiling and targeted interventions.

Demonstration: Table 3 lists, for 10 users, the hour of the day when they are most active in sending queries. This reveals daily habits or routines, allowing an adversary to infer, for example, if a user is typically active in the evening, late at night, or during work hours.

	ip_address	most_active_query_hour
0	34.101.177.245	18.0
1	244.190.169.115	21.0
2	60.235.63.253	19.0
3	60.109.165.215	20.0
4	159.118.124.69	18.0
5	203.24.85.254	19.0
6	82.230.180.56	13.0
7	58.9.101.51	18.0
8	118.135.166.110	13.0
9	146.71.112.211	20.0

Fig. 3: Result of the user’s pattern attack - most active hours for users

B. Defences

To mitigate one of the privacy risks identified in the previous section, we propose a location obfuscation defence at the client’s side. In this scheme, before submitting a query, the user adds a small amount of random noise to their true location

coordinates, so that each reported location slightly differs from the user's actual position.

This approach makes it harder for an adversary to accurately infer sensitive locations such as home and work, with high precision. The amount of noise can be chosen so that the utility of the service remains acceptable to the user.

This defence specifically targets the home/work inference attack. After obfuscation, queries originating from home or work are no longer always aligned, making clustering-based inference methods less accurate. As a result, the adversary must contend with greater uncertainty and decreased accuracy when attempting to infer a user's home or work location.

Experimental evaluation: To demonstrate the effectiveness of our defence, we applied Gaussian noise to the latitude and longitude coordinates in the `queries.csv`. Specifically, for each query, we added a random noise drawn from a normal distribution, $\sigma = 0.002$ degrees which correspond to approximately 200 meters, to both the latitude and longitude values. The user's true location is therefore slightly randomized before each query is sent to the service provider. By repeating the home/work inference attack on the obfuscated dataset, we can assess how the added noise impacts the adversary's ability to accurately infer the sensitive locations. As shown in Table 4, the adversary can no longer identify clusters of repeated locations for each user, users have only a single query per inferred home or work location. This shows that location obfuscation can greatly reduce the attacker's abilities and the accuracy of this specific attack.

Privacy-utility trade-offs: Location obfuscation tries to significantly reduce the adversary's success rate in inferring home and work locations, while still preserving most of the service's utility for users. However, there is a trade-off between privacy and utility:

- If the added noise is too large, the return POIs may no longer be relevant or conveniently located for the user, degrading the user experience.
- If the noise is too small, the obfuscation may not be enough to ensure privacy, as the adversary can still accurately cluster the obfuscated points and infer the sensitive locations.

Choosing the correct noise level thus depends on the desired balance between privacy and usability for the application and its users.

	ip address of user	Home Coords	Home Count	Work Coords	Work Count
0	34.101.177.245	(46.5528116577507, 6.810736381566818)	1	(46.53393564971104, 6.58843582068574)	1
1	244.190.169.115	(46.5514925020605, 6.647199506707446)	1	(46.5400992936011, 6.50242947909843)	1
2	60.235.63.253	(46.56258843383715, 6.645264069094091)	1	(46.5474049715639, 6.57823821115227)	1
3	60.109.165.215	(46.53275903330104, 6.681312652631285)	1	(46.56804278704633, 6.610620954472917)	1
4	159.18.174.68	(46.54217667117485, 6.588824307263083)	1	(46.52913043427921, 6.508985808679854)	1
5	203.24.85.254	(46.55797470870406, 6.600366279138692)	1	(46.535441169076414, 6.576682626502216)	1
6	82.230.180.56	(46.568685142859455, 6.645807118023598)	1	(46.523530193696075, 6.601680572679357)	1
7	58.9.101.51	(46.56915499813574, 6.80643232402444)	1	(46.5487910636454, 6.611803446910020)	1
8	118.135.166.111	(46.50516887073966, 6.61251810930838)	1	(46.52017779887275, 6.617155411519831)	1
9	146.71.112.211	(46.531303133522094, 6.624241107556799)	1	(46.54024557461676, 6.574634213713738)	1

Fig. 4: Result of the home/work inference attack after location obfuscation

IV. CELL FINGERPRINTING VIA NETWORK TRAFFIC ANALYSIS

A. Implementation details

First, to complete the data collection, we decided to collect 20 network traces for each of the 100 cells, giving us a total of 2000 traces to use to train and test the classifier. We automated the process using a Bash script (`collect_traces.sh`). During data collection, we quickly noticed that Tor would occasionally become overloaded or crash, which can be explained by the large number of rapid, successive queries. The typical error messages we obtained indicated that the SOCKS proxy could not be reached or that requested addresses could not be assigned. To resolve this issue and improve the reliability of our data collection, we modified our script to restart Tor after the queries for a single cell were finished. We also added a small delay before continuing with the next cell, to be sure that Tor fully reinitialized before going further. We used `tcpdump` to capture network traffic trace for each query. It enables recording of all outgoing Tor traffic, allowing us to collect essential metadata such as packet sizes, counts, and timings. For each query the process resulted in `.pcap` files ready for feature extraction.

For each captured `.pcap` file, we extracted a set of traffic features using Pyshark. The script iterates over every packet in a trace, collecting both packet size and its timestamp. From these raw values, we compute the following features for each traces, allowing us to capture subtle differences between network behavior:

- Total number of packets
- Total bytes transferred
- Average packet size and Standard deviation of packet sizes
- Minimum and maximum packet sizes
- Duration of the trace (from first to last packet)
- Mean and standard deviation of inter-packet times
- Minimum and maximum inter-packet times
- Distribution of packet sizes
- Sizes of the first three packets

These features summarize the observable metadata of each query and will be used to train and test our classifier.

B. Evaluation

Initially, we trained the classifier on a basic (small) feature set, which was total number of packets, total bytes, average packet size, standard deviation and duration. With these features alone, the mean accuracy ranged between 50% and 60%. Those results indicate us that the classifier could identify the queried cell better than with random guessing, but could be improved. To improve performance, we expanded our feature set by modifying our `extract_features`. We progressively added features until reaching the set described above. After incorporating these features, the classifier's mean accuracy improved to approximately 65% as shown below:

Accuracy for this fold : 0.2600

Accuracy for this fold : 0.6750

```

Accuracy for this fold : 0.6950
Accuracy for this fold : 0.7200
Accuracy for this fold : 0.6734
Accuracy for this fold : 0.6482
Accuracy for this fold : 0.7538
Accuracy for this fold : 0.7035
Accuracy for this fold : 0.7035
Accuracy for this fold : 0.6683

```

Mean accuracy over 10 folds: 0.6501

We observed that the first fold in our 10-fold cross-validation consistently resulted in a lower accuracy (around 26%) while the other folds achieved higher accuracy.

Factors that influenced the classifier performance included mainly the richness of the set of features, the degree of similarity between the server responses for different cells, and occasional trace anomalies. Despite these different parameters, the overall mean accuracy shows the effectiveness of traffic analysis even on encrypted and anonymized network traffic.

C. Discussion and Countermeasures

Our results highlight that even with Tor anonymization and encryption, observable traffic metadata, can leak sensitive information. With a standard machine learning classifier we achieved a mean accuracy of 65% which demonstrates that a significant amount of location can be inferred from traffic analysis.

Our intuition tells us that the classifier performed best for cells with very distinctive server responses (high or low number of POIs), while it is harder to distinguish cells with similar traffic features.

To mitigate this kind of attack, several countermeasures could be applied :

- 1) We could make all server responses the same size, (in bytes and in number of packets), regardless of the actual number of POIs in a cell. This make it difficult for an adversary to infer which cell was queried based on traffic analysis. This could be achieved by sending dummy data to fill shorter responses or by splitting large responses into multiple smaller packets.
- 2) We could combining multiple queries into a single network session, it adds randomness and uncertainty for the attacker, as each traffic trace will reflects multiple cell queries. However, it is not always practical because the application would need to collect multiple queries before sending them over the network (unless combined with dummy queries).

In conclusion, we demonstrate that traffic analysis methods can compromise user privacy with an adversary eavesdropping the connection between the user and the service provider. Effective countermeasures should focus on hiding the queries metadata.

- Rassene M'sadaa: Part 1
- Leila Sidjanski: Part 2
- Luna Ralet: Part 3

REFERENCES

V. CONTRIBUTION:

Each member took care of an entire part: