CS-523 SecretStroll Report

Michael Gysel, Valentim Romão

***Abstract*—The design of our system consists of two main components, client and server, in a many-to-one relationship. The client role is to query the server for suggestions based on their subscriptions and location. The server role is to respond to these queries with relevant points of interest. In order to provide privacy for clients regarding their subscriptions, we use attribute-based credentials so users can prove they are allowed to receive certain suggestions without needing to identify themselves [1]. However, this system is still vulnerable to some attacks, such as location inference based on the locations sent by the user, the responses, or the IP address [2]. Another possible attack is cell fingerprinting, where the effectiveness depends on the model used, the amount of network traces, and the features extracted.**

I. Introduction

The aim of this project is to develop a privacy-preserving location-based service that enables users to learn nearby points of interest based on their location and subscriptions. To implement this application, we considered three main privacy threats: protecting user identity during authentication, protecting user location which is used to retrieve the nearby points of interest, and protecting against cell fingerprinting attacks based on the network traffic exchanged between the user and SecretStroll.

Taking these privacy threats into consideration, we split the system into three parts where each part addresses one of the privacy threats mentioned above. In the first part, we protect user identity by implementing attribute-based credentials, which allow for user subscription authentication without revealing user identity. In the second part, we conducted a privacy evaluation to determine location inference threats, suggest ways to protect user locations, and assess the privacy and utility such methods provide. In the third section, we perform a cell fingerprinting attack by collecting network traces, extracting features from those traces, and inferring users' cell locations.

II. Attribute-based credential

The mapping of the system to the attribute based credential was made in three steps. Firstly, we develop three functions for the Pointcheval-Sanders signature scheme which were responsible for generating a pair of keys, signing messages, and verifying signatures. In the second step, we developed the issuance protocol, where a user commits values they want to include in the credential and use a zero-knowledge proof to prove the commitment was computed correctly. After verifying proof validity, the issuer chooses attributes and generates a signature to send to the user, which the user then uses to generate the final signature, which is used as their credential (the signature and the attributes). In the final step, we implemented the showing protocol, where the user shows the credential to the verifier, the verifier checks the validity of the credential, and the user gains authorization. It is important to note that the user only discloses those attributes essential for the operation.

For the SecretStroll system, the list of attributes consists of the username as well as all available subscriptions in the service. The user's only attribute would be their secret key, which is always kept hidden in every request they make. The issuer's attributes consist of the username and the subscriptions for each point of interest; moreover, the issuer controls these attributes to prevent the user from obtaining free subscriptions. As a result, the user obtains a credential with their username, the subscriptions they registered, and their secret key. When using this credential, the user would only need to disclose the subscription to the specific points of interest they are querying SecretStroll for, minimising the information they reveal. For example, if the user wants to discover nearby restaurants, they would disclose their restaurants subscription while omitting their username, secret key, and other possessed subscriptions. In this scenario there would always be at least two hidden attributes in each request, the user's username and secret key.

The communication between the client and server is secured using public key cryptography. The server generates a key pair and sends the public key to the client. The client then encrypts messages with this public key and sends the encrypted messages to the server, which only the server can decrypt.

The communication costs between the client and server are reduced by using the Fiat-Shamir hueristic when creating the zero-knowledge proofs. This allows the client to generate their own challenge to avoid the challenge being generated and sent from the server. This challenge consists of the hash of multiple public values and the commitments generated.

For the user commitment proof, the client would compute the challenge using the server's public key, the commitment with secret values, and the commitment with random values. The only difference for the server is that it uses the recomputed commitment with random values.

$$com = commitment\_secret\_values$$
$$R = commitment\_random\_values$$
$$challenge = Hash(server\_public\_key \| com \| R)$$

For the disclosure proof, the client computes the challenge using the server's public key, the signature, the right side of the proof expression with random values, and the message m. The server computes the challenge with the same values, but to recompute the right side of the proof expression with random values it needs to first compute the left side, which consists of disclosed values. Next, the server proceeds to use

Computation	Subscriptions	Mean (μs)	Standard Deviation (μs)
Key Generation	1	1.69	1.74
	5	5.81	5.80
	10	8.06	2.34
Issuance	1	9.24	10.42
	5	13.49	9.13
	10	19.22	5.96
Showing	1	10.27	13.80
	5	10.94	3.57
	10	12.76	5.12
Verification	1	17.18	8.58
	5	33.86	16.56
	10	53.59	26.56

TABLE I: Computation Cost of Key Generation, Issuance, Signing, Verification

the values given by the client, so that it obtains the same value for the expression as the client generating the same challenge.

$$R = right\_side\_disclosure\_proof\_with\_random\_values$$
$$challenge = Hash(server\_public\_key \| signature \| R \| m)$$

A. Test

How did you test the system? You need to test the correct path and at least two failure paths.

Testing was conducted on the signature scheme, issuance protocol, and showing protocol and then on various user scenarios.

To test the signature scheme, the following scenarios were tested: correct key generation, signing, and verification; incorrect message length used to sign; incorrect public key used for verification; and incorrect message used for verification. To test the issuance protocol and showing protocols, the following scenarios were tested: correct credential issuance, correct credential verification, incorrect public key used for verification, and incorrect message used for verification.

B. Evaluation

Key generation, issuance, signing, and verification were evaluated in terms of both computation and communication costs.

To evaluate computation costs, 1,000 trials each were conducted for key generation, issuance, signing, and verification. Because the computation costs of key generation are impacted by the number of attributes, 1, 5, and 10 attributes were evaluated, which would be typical of user scenarios. The mean and standard deviation of these trials were then calculated. As can be seen in Table 1, the computation cost of key generation is the lowest while the cost of verification is significantly higher than the other protocols. Furthermore, the number of subscriptions significantly impacts the cost of key generation, issuance, and showing protocols. Furthermore, the standard deviation of each protocol varies, with no clear trends relating to the number of subscriptions.

Command	Number of POIs	Mean (μs)	Standard Deviation (μs)
get-pk	N/A	6.09	1.54
register	1	17.81	4.26
	3	18.14	19.35
loc	1	143.74	43.13
	3	138.56	40.86

TABLE II: Communication Cost of get-pk, register, loc

To evaluate communication costs, the client get-pk, register, and loc commands were tested with 1,000 trials. Typical of user scenarios, 1 and 3 points of interest were measured for the register and loc commands, but not for the get-pk command as it has no bearing on the client obtaining the server's public key. As can be seen in Table 2, the communication costs increase in the following order: get-pk, register, and loc. Furthermore, the costs of retrieving information about POIs through the loc command is significantly more costly than any other user action. Notably, the number of POIs a user selects has little bearing on the communication overhead. Lastly, as is expected, the communication costs of the system, which require computation of the key generation, issuance, signing, and verification, are larger than the computation costs.

III. (De)Anonymization of User Trajectories

A. Privacy Evaluation

The goal of SecretStroll is to provide anti-surveillance privacy, where the user's location data is protected without the need to trust the provider, infrastructure, or any strategic adversary. Specifically, the system should protect user location privacy against the following adversaries: SecretStroll and any strategic adversaries such as individuals, companies, and governments who have access to all network traffic and any data stored by SecretStroll. These strong assumptions in both adversaries and adversarial knowledge were chosen because location-based information can be used by a variety of adversaries to infer information about users. Furthermore, recent history suggests that data leaks are frequent and governments monitor network traffic. Thus, in order to truly preserve user privacy, these assumptions are necessary. In terms of privacy properties, the system should provide confidentiality so adversaries cannot learn user requests from network data and anonymity so adversaries cannot link location data to specific users.

Both adversaries share similar attack strategies: collect considerable amounts of user queries or network traffic and then attempt to correlate a user's ip address with their query location. We assume the system uses HTTP, so an adversary monitoring traffic could see packets in plaintext. First, the adversary would collect this network traffic. After collecting this data, the adversaries would require nearby points of interest (POIs) as auxiliary information. This task is easy for SecretStroll as it already has this information; however, external attackers would need to obtain query responses from SecretStroll or manually search for possible POIs. Finally, the

adversary would compute the probability of a user visiting a given POI based on the percentage of occurrences of that POI in the list of POIs returned by the SecretStroll server. Based on the most frequently returned POIs, an attacker could potentially infer user routines and very significantly, the user's home and work addresses. This inference would be made even stronger if the attacker used as complementary information the timestamps of each query, since most people are at their home address between 20:00 and 8:00 and work address between 8:00 and 17:00. The adversaries could also perform subscription inference attacks; however, because http is used, the attacker could obtain a user's subscriptions by simply looking at their requests. Thus, performing a subscription inference attack would be trivial and was not conducted during the privacy evaluation.

In the following section, we perform these attacks with and without the proposed defenses and discuss the resulting privacy and utility tradeoffs.

B. Defenses

In order to protect user location privacy, we suggest adding noise to the location of each user request. This defense consists of generating a perturbed location within a radius R of the user's actual location, and then using this perturbed location when querying SecretStroll. Using this technique, an attacker would only have access to the perturbed user location, which would reduce the likelihood of an attacker inferring the user's actual location, particularly when the user makes numerous requests from their home and work locations. This is due to the fact that perturbed queries would show different user locations, even when making the same request from the same location. This approach would also reduce the likelihood of an adversary correlating network traffic between groups of users, thereby determining social connections. This is due to the fact that when groups of users make requests from one location, the perturbed queries would show different locations for each user in the group. The effectiveness of this countermeasure depend on the value of R used to perturb the user locations. We also considered K-anonymity to protect user location privacy. K-anonymity was not chosen as it would result in a significant loss of utility, requiring K-1 other users to also send a request. Furthermore, K-anonymity has not been shown to increase inference error, per Professor Reza Shokri's lecture on location privacy.

In the context of SecretStroll, we define privacy as the user not having their request associated to them, which would result in a linkage between their identity and their location. To assess the level of privacy the SecretStroll system provides, we performed 100 perturbed and unperturbed queries each for 200 different IP addresses and recorded the POI recommendations. We then measured the percent similarity when comparing the five most common POI recommendations for each IP address for the perturbed and unperturbed queries. We chose the five most commonly returned POIs as these would most likely correspond to a user's home or work address. The resulting similarity between perturbed and unperturbed queries was 29.7%, suggesting the perturbations provide additional privacy to users, particularly in regards to their most sensitive locations.

For the SecretStroll system, we define utility as the ability of the application to provide the user with POIs within 0.25km of their location. To assess the level of utility, the distance between the user's query location and their perturbed query location was measured. Again, this assessment was performed with 100 perturbed queries each for 200 different IP addresses. The mean and max distances of the 20,000 perturbed queries was 0.14km and 0.25km, respectively. While the Secretstroll perturbation provides additional privacy, it also degrades utility; however, because the perturbed queries are within 0.25km of the user's location, the utility was deemed sufficient.

To perform both evaluations, we used a script (compute_perturbed.py) that simulates the behavior of our defense strategy. It receives a list of several query locations as input and returns a list of perturbed query locations, where each location is modified by adding a factor of 0.002 * R1 to the latitude and 0.002 * R2 to the longitude, where R1 and R2 are random float numbers between -1 and 1.

Our defense strategy provides additional user location privacy because the users' exact location would not be used, resulting in an increased inference error for the attacker. In addition, this defense would require the attacker to use more computationally-intensive strategies, such as Markov Chains, in order to have a reasonable probability of inferring the user's location. However, this defense decreases the system's utility, since the nearby POIs are computed from the user's perturbed location as opposed to their actual location. Thus, this implies a less accurate response. To conclude, this defense has its advantages in terms of privacy and its disadvantages in terms of utility, and the radius R can be adjusted to balance the two. Based on our privacy and utility metrics, increasing R would provide greater privacy and lower utility, while decreasing R would provide lower privacy and greater utility.

IV. Cell Fingerprinting via Network Traffic Analysis

A. Implementation details

To train our classifier, we collected significant amounts of metadata for queries from each cell.

To automate the data collection process, we created a script that uses *tcpdump* to generate 100 grid commands for each cell. First, we setup the system with server initiation and the client retrieving their public key and registering. Next, we move to the "traffic_collection" folder and start the script. In the beginning of each script iteration, tcpdump is run to monitor the metadata of the packets exchanged between the client and TOR's entry node, from the client perspective. Thirdly, the script calls the grid command using the cell id we are analysing as its argument. When the grid command completes, the script stops tcpdump and stores the traffic exchanged in a pcap file.

For the feature extraction, we parsed each of the pcap files, ignoring the ones that were incomplete due to network errors,

and from there we were able to extract the features that we considered to be effective in determining the cell that was queried. The features extracted, which were modeled from [2], related to the number of total packets, incoming and outgoing packets, packet ordering statistics, packet inter-arrival time statistics, concentration of incoming and outgoing packets, concentration of incoming and outgoing packets within the first 30 and last 30 total packets, number of packets per second, and packet length, which were seen in the Slides from Lecture 6 [3]. Finally, we used the Random Forest Classifier using 10-fold cross validation to predict cell id based on the features extracted.

B. Evaluation

The Random Forest Classifier predicted cell id with an 18% accuracy. The most significant features in terms of prediction accuracy were the number of packets, packets per second, packet inter-arrival time statistics, and packet lengths.

C. Discussion and Countermeasures

The factors that could influence the performance are related to the features we chose. Therefore, the performance could change, if the server handling of requests was dynamic, leading to different outcomes in each execution, decreasing the frequency of patterns, and resulting in a higher randomness for the values of the features. The best countermeasures would obfuscate the features that were most important in the Random Forest Classifier. The use of Tor would mask the packet lengths as each Tor packet is an equivalent size. Secondly, the use of dummy traffic would add noise to the number of packets, packets per second, and packet inter-arrival time statistics. Thus, this would mask the adversary's ability to produce these statistics accurately, thereby reducing prediction accuracy.

REFERENCES

[1] Carmela Troncoso, and Wouter Lueks. Slides from lecture 2 - anonymous authentication. [Online]. Available: https://moodle.epfl.ch/mod/resource/view.php?id=1195985

[2] Reza Shokri. Slides from lecture 3 - location privacy. [Online]. Available: https://moodle.epfl.ch/mod/resource/view.php?id=1197156

[3] George Danezis. Slides from lecture 6 - anonymous communicationsfile. [Online]. Available: https://moodle.epfl.ch/mod/resource/view.php?id=1201347