Prox

October 29, 2017

1 General Overview

The goal of this application is to be able to share your location with friends if and only if you are in ”close proximity” with each other while keeping locational privacy. Specifically, no central server should ever see your location, your friends that are far away cannot see your location, and anybody who is not your friend cannot see your location.

Any information sent to any server is considered **entirely public**; meaning, all communication can be made on an insecure connection. Each exchange of information between server and clients needs to be encrypted in a very specific way.

2 Primitives

One large public prime P is determined at the inception of the project.

2.1 A user

A user’s local storage contains the following:

- Two large primes p and q.
- A public key e relatively prime with $(p-1)(q-1)$.
- A private key d such that $ed \equiv 1 \mod (p-1)(q-1)$.
- A list of public keys we have initiated either a friendship request or a friendship with.

For each user, the server stores the following, and lets it be available to anyone:

- The product of the user’s large primes, pq. This is the user’s unique identifier.
- The user’s public key e.

- A queue of all messages that were sent to that user. This is stored as a list $(c, p'q', 2^{d'} \mod p'q', pq, M_1^{d'} \mod p'q', M_2^{d'} \mod p'q', ...)$, where c is the message type, d' is the private key of the sender, $p'q'$ is the unique identifier of the sender, and M_i are the numbers in the message.

The server clears all messages after they are alive for some short amount of time.

2.2 Making friends

A requester initiates Diffie-Hellman exchange with the friend by sending a message of type 0. If the friend responds with a message of type 1, which includes the second part of the exchange, then they share a secret. See “Sending messages”.

2.3 Location

We apply the Gall-Peters projection on the longitude from the central meridian λ and the latitude ϕ, in degrees:

$$x := [R\lambda],$$

$$y := [2R \sin \phi],$$

where [] denotes the floor function and R is an integer related to the radius of Earth. We then round x and y down to the nearest integer, and assign each location a number

$$L(x, y) := x \bmod 360R + 360R(y \bmod 2R).$$

2.4 Matching point test

Consider two parties A and B with secrets l' and l, respectively. The following procedure will allow A to know the truthness of the statement $l' = l$ without either gaining insight on the values of l', l. Additionally, any party without knowledge of l', l given all communications cannot recover the truthness of the statement $l' = l$.

1. A requests the location of B, choosing a secret k' relatively prime to $P - 1$ and sending $l'^{k'} \pmod P$.

2. B chooses a secret k relatively prime to $P - 1$ and sends $l^k \pmod P$.

3. A sends $l^{k'k}$.

4. B calculates $l'^{k'k} \pmod P$ and compares it with $l^{k'k} \pmod P$.

5. If they are the same and B wants to say so, B responds with l.

2.5 Sending messages

Upon receipt of a message $(c, p'q', 2^{d'} \mod p'q', pq, ...)$, a user first queries the server database for the e' associated to $p'q'$, then evaluates $2^{d'e'}$ to confirm that it is 2.

- 0: Friend request from $p'q'$ to pq. The sender generates random numbers g, k' relatively prime to $P-1$. The message is

$$(0, p'q', 2^{d'} \mod p'q', pq, g \mod P, g^{k'} \mod P).$$

- 1: Friend confirmation of request from $p'q'$ to pq. The sender (of the confirmation) generates a random number k relatively prime to $P-1$. The message is

$$(1, pq, 2^{d} \mod pq, p'q', g^{k} \mod P).$$

  The friends now have a shared secret $s := g^{k'k} \mod P$.

- 2: Location request from $p'q'$ to pq: Step 1 of the matching point test. The sender $p'q'$ generates a random number k' relatively prime with $P-1$ and calculates $l' = sL' \pmod P$, where L' is his location. The message is

$$(2, p'q', 2^{d'} \mod p'q', pq, (l')^{k'} \mod P).$$

- 3: Location response of the matching point test: Step 2 of the matching point test. The sender (of the response) pq generates a random number k relatively prime with $P-1$ and calculates $l = sL \pmod P$, where L is his location. The message is

$$(3, pq, 2^{d} \mod pq, p'q', l^{k} \mod P).$$

- 4: Step 3 of the matching point test. The message is

$$(4, p'q', 2^{d'} \mod p'q', pq, l^{k'k} \mod P).$$

- 5: Step 5 of the matching point test. The message is

$$(5, pq, 2^{d} \mod pq, p'q', l \mod P).$$

3 Things to do not described here

- At the beginning of a friendship, the friends need to agree on a private key for encrypting all future messages. A user would then decrypt all of their messages by applying each of these keys to the first element in all messages, and seeing which ones ended up as identifiers for friends.
- We have to keep track of how long messages are alive for.