Hermes
Building a practical multi-device SPARTA

Surendra Jammishetti
CSE 108C
sjammish@ucsc.edu

I. Introduction

With the threat of a global adversary looming over online communication, its becoming a more pressing concern to have secure, relaible, messaging services. We came up with E2E encryption to protect the contents of our messages, but its not enough to protect against a global adversary. E2E encryption doesnt hide the metadata of our conversations, as the adversary can still reconstruct who is talking to who, and when. As former NSA Chief Gen. Michael Hayden said, "The U.S. government kills people based on metadata" [1], necissating the need for metadata-private systems.

Groove [2], an existing system, uses mix-nets and public providers to offer a metadata-private solution, but has many pitfalls. It is an synchronous system, requiring and limited to one message per round. Additionally, the latencies are in the order of epoch times, with a really complex architecture due to the underlying mixnets to route messages.

The SPARTA [3] construction offers a metadata-private anonymous communication system, and for the first part of my project it'll detail the implementation of SPARTA-LL. Then, taking inspiration from Groove, I wanted to add multi-device functionality to SPARTA. Additionally I've been able to get my SPARTA implementation running inside an AWS Nitro Enclave!

II. Base Sparta

For the core implementation of SPARTA, I followed the pseudocode provided in the paper [3]. As for my language of Some of the core things I had the liberty of implementing myself were

1) Oblivious Select

   I ended up making a function that would take in a conditional, and two integers, where it would execute in constant time without branching and would return the first integer if the condition was true, and the second integer if it was false. I used oblivious select in send / fetch implementation, using it to select between two pointers obliviously.

2) Oblivious Multi-Queue

The oblivious multi-queue is baked into the send and fetch operations, as their pseudocode constructs this multi-queue by reading the queue location from the user store and then en/de queueing when necessary.

3) Oblivious Map

The oblivious map used for the userstore is a custom construction thats not as efficient as state of the art. It boasts a time complexity of $O(N \log(N))$, and operates on the following pseudocode.

A. *Facebook PathORAM discussion*

III. Multi-Device Extension

IV. Experiments and Results

V. Conclusion

References

[1] L. Ferran, "Ex-NSA Chief: "We Kill People Based on Metadata." [Online]. Available: https://abcnews.go.com/blogs/headlines/2014/05/ex-nsa-chief-we-kill-people-based-on-metadata

[2] L. Barman, M. Kol, D. Lazar, Y. Gilad, and N. Zeldovich, "Groove: Flexible Metadata-Private Messaging," in *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, Carlsbad, CA: USENIX Association, Jul. 2022, pp. 735–750. [Online]. Available: https://www.usenix.org/conference/osdi22/presentation/barman

[3] K. Fredrickson, I. Demertzis, J. P. Hughes, and D. D. Long, "Sparta: Practical Anonymity with Long-Term Resistance to Traffic Analysis." 2024.