Hermes
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!

A. Threat Model

II. Base Sparta

For the core implementation of SPARTA, I followed the pseudocode provided in the paper [3]. Since an oram is required for all of SPARTA's internal data structures, I used Facebook's implementation.

A. Facebook Oram discussion

The facebook PathORAM implementation requires two things to be cryptographically secure; oram clients are running in a secure enclave architecture, that also provides memory encryption as they do not encrypt on write themselves. Additionally they implement the oblivious position

map and stash from Oblix [4]. Like Oblix, its a pure rust implementation, and is recursive in nature. It recurses down until the size of the oram is small enough to fit inside of a Linear Time ORAM, which maintains oblivousness by reading/writing over each memory location per access. I'm unsure why they chose to have a constant time ORAM ($O(N)$) for the base case instead of a PathORAM ($O(\log(N))$). It would be interesting to benchmark the difference between these two base cases.

B. Personal Implementations

I implemented the other data structures and operations myself, using Facebooks PathORAM as the underlying oram data structure.

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 $O(N \log(N))$, as it iterates through every element in the allocated oram. It's akin to the LinearTime Oram in the Facebook oram library.

C. Technical Details

The core of the sparta server is a GRPC server that binds to a virtual socket inside of an AWS Nitro Enclave. As the enclave implementation was only done in the past week, there are a few holes that need to be cleaned up for it to be a deployable instance of SPARTA.

1) TLS

   Amazon has support for enclave traffic to be encrypted via TLS from inside the enclave, ensuring that any client and SPARTA have a secure channel for communication. It would require to move the core logic and state out of the GRPC server and instead wrap it with

a HTTP server, and then proxy the traffic via NGINX as described here. Currently there is a custom, insecure, GRPC proxy, called Trojan that wraps around the SPARTA vsock and enables the exchange of traffic between SPARTA and clients.

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.

[4] P. Mishra, R. Poddar, J. Chen, A. Chiesa, and R. A. Popa, "Oblix: An Efficient Oblivious Search Index," in *2018 IEEE Symposium on Security and Privacy (SP)*, 2018, pp. 279–296. doi: 10.1109/SP.2018.00045.