Hermes: Building a practical multi-device SPARTA

Surendra Jammishetti
sjammish@ucsc.edu

I. Problem Statement

Initially, I wanted to try to implement Groove, [1], but after technical / scope limitations, I pivoted towards Sparta, [2]. While working on the implementation for Sparta, ideas from Groove started to influence the path I wanted to take my work. Groove has a rigorous explanation of multi-device support, which the existing work for Sparta lacked. The goal for my project is to embed the high-level design of Sparta with practical messaging semantics, like multi-device support, authentication, and dialing.

II. Implementation

1) Intel-SGX

   Unfortunately, I have an AMD laptop and have no access to an Intel machine. Therefore, using the simulator is impossible, and after talking with Apostolos, he said it was all right to forego the usage of SGX. Porting the current implementation to be SGX compatible is not impossible / wouldn't need an entirely new codebase, I would need access to an SGX machine and would move the security-relevant code into an enclave using Fortanix's rust SGX development platform, and proxy in the requests from a userspace program into the enclave.

2) Sparta-LL

   I have a rudimentary version of Sparta-LL compiling and working, using Facebooks' oram library. The user store is still a non-oblivious data structure (a simple KV map) instead of an OMAP due to my previous difficulties understanding how OMAP's work. In the meantime, it is a non-oblivious data structure (A simple KV map). However, the message store is fully functional, along with the oblivious selection. Other than the user store's incomplete OMAP implementation, it works perfectly.

3) Client implementation

   I've completed a barebones implementation of a client sending and receiving messages, you can check it out in `/athens`.

A. *Papers read*

a) *Messaging:*

1) Groove, [1]
    - Multi-device, provider model, mix-nets, high latency / overhead.
2) Sparta, [2]
    - Decided to implement Sparta-LL. Low-overhead, low latency, simple.

b) *OMAP's:*

The following papers I read to understand how to create an efficient OMAP, hoping for more pseudocode / simpler explanations, but this was before the lecture on Oblivious Data Structures, so I'm much more confident now. One interesting thing to note is EnigMAP's criticism of Oblix, showing that they didn't adhere to instruction level obliviousness, which as a concept is very interesting. Reading their critique made me think about whether it's possible to have a compiler transform all memory accesses to be oblivious, and do the same with branches, making the user-side implementation trivial.

1) Oblivious Data Structures, [3]
2) EnigMAP, [4]
3) Oblix, [5]

III. TO-DO

- [ ] implement the OMAP for the user store
    - https://francismurillo.github.io/2019-07-31-Understanding-Rust-Through-AVL-Trees/
    - maybe just do a trivial avl implementation and slap in oram to access nodes?
        - $O(\log^3(N))$ time complexity
- [ ] have a simple client side impl, preferably on pc / cli
- [ ] simple mobile client impl
- [ ] proxy / provider model to facilitate multi-device support
    - This would be the anonymizing layer that Kyle talked about in his lecture.
- [ ] Nice documentation for everything
- [ ] End 2 End encryption

References

[1] 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

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

[3] X. S. Wang *et al.*, “Oblivious Data Structures,” in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, in CCS '14. Scottsdale, Arizona, USA: Association for Computing Machinery, 2014, pp. 215–226. doi: 10.1145/2660267.2660314.

[4] A. Tinoco, S. Gao, and E. Shi, “Enigmap : External-Memory Oblivious Map for Secure Enclaves.” [Online]. Available: https://eprint.iacr.org/2022/1083

[5] 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.