TODO

https://github.com/fortanix/rust-sgx/issues/401

MVP

- [x] implement the OMAP for the user store
  - https://francismurillo.github.io/2019-07-31-Understanding-Rust-Through-AVL-Trees/
  - [1]
  - [2]
  - ~~maybe just do a trivial avl implementation and slap in oram to access nodes that way?~~
    - $O(\cancel{\log^3}(N))$ ~~time complexity~~
  - nahhh just do the dumb version, linked list user store, just compare each one to make this version oblivious, need to traverse the entire list each time.
    - $O(N \log(N))$
- [x] benchmark throughput of the server
- [x] user server implementation to create new user
- [x] figure out how adding a user works (OMAP semantics)
- [x] In the omap, make `get_data` and `update_data` have a similar trace.
- [ ] have a simple client side impl, preferably on pc / cli
- [ ] Nice documentation for everything
- [ ] End 2 End encryption
- [ ] Discuss the implementation of the Facebook ORAM library
- [ ] performance discrepancy possibly due to hardware im running on
- [ ] apache teaclave http server example with tls
- [ ] aws nitro enclaves https://dev.to/bendecoste/running-an-http-server-with-aws-nitro-enclaves-elo
  - https://docs.aws.amazon.com/enclaves/latest/user/getting-started.html
  - https://github.com/aws/aws-nitro-enclaves-samples/tree/main/vsock_sample/rs
  - `nitro-cli run-enclave --eif-path sparta.eif --cpu-count 2 --memory 4096 --debug-mode --enclave-cid 16`

Post MVP

- [ ] multi-device support via support of proxy
  - inspired by how groovy had the provider system
  - to make sparta-ll into a provider based system, its fairly cheap to have a small embedded device to act as a proxy for each user its not super unfeasable
- [ ] Add TLS
  - this would also require me to host it on a server
- [ ] client side implementation with sqlite?
- [ ] ~~figure out how to get this building on fortranix sgx~~
  - ~~If I dont reach this, could sell this as not feasable in such a short amount of time but looking forward to do it in the future.~~

EC2 INSTRUCTIONS

1. launch ec2 instance

```
  aws ec2 run-instances \
--image-id ami-04acda42f3629e02b \
--count 1 \
--instance-type m5.xlarge \
--key-name 'hello_world keypair' \
```

```
--enclave-options 'Enabled=true' \
--block-device-mappings 'DeviceName=/dev/sda1,Ebs={VolumeSize=64,VolumeType=gp3}'
```

2. add ssh security group TODO: describe how to do this later

3. ssh onto ec2 instance

4. `sudo yum install git -y`

`+ yum install make glibc-devel gcc patch`

`+ sudo amazon-linux-extras install aws-nitro-enclaves-cli -y`

https://docs.aws.amazon.com/enclaves/latest/user/nitro-enclave-cli-install.html
- install the nitro cli

Project Structure

Hermes
- proxy? or could be the name for the entire thing

Athens
- Client Cli
- Tauri mobile app

Sparta
- Sparta LL implementation

Sator
- tester utiliy to help with seeding database

Things to note

- my shit is 5-6x slower bru
  - facebook oram library is around 3-5 ms per access, main cause of slowdown, could be better with a different oram implementation

Questions for Kyle

1. Does sparta support users with multiple devices?
2. What sort of E2E encryption scheme can be added onto sparta?
3. How does authentication work with oblivious systems?

Qucklinks

oram library:
- https://github.com/facebook/oram?tab=readme-ov-file
  - only secure inside of an enclave with memory encryption

enclave framework:
- https://github.com/fortanix/rust-sgx

intel-sgx?
- https://github.com/intel/linux-sgx-driver
- i dont have the hardware

Kyle Notes

encryption isnt sufficient to protect messaging “with enough metadata you dont really need content” - NSA

theoretically sparta can have multiple layers an anonymyzing layer could be used to aggregate your devices and then pull them that way.

Bibliography

[1] 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.

[2] 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.