TODO for the thang

main

- [ ] implement the OMAP for the user store
- [ ] have a simple client side impl, preferably on pc / cli
- [ ] Nice documentation for everything
- [ ] figure out how adding a user works (OMAP semantics)

expansion

- [ ] client side implementation with sqlite?
- [ ] figure out how to get this building on fortranix sgx

sgx feature

when enabled, compiles for sgx using enclave runner otherwise just runs inside

could sell it as like ok i dont have the hardware for this yet so im kinda cringing

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

https://francismurillo.github.io/2019-07-31-Understanding-Rust-Through-AVL-Trees/

Brainstorming questions

How do i do this?

how does a user sign up? is that even such a thing?

how do we make sure recipients are secure? like what would be an ideal end to end protection scheme?

what would need to be inside an enclave and what would not have to be inside of an enclave?

Vague plan

Hermes

- Lets goooo
- cinema

Athens

- Client Cli
- Tauri mobile app

Sparta

- Sparta LL implementation

sparta paper notes

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

ideas?
- add group messaging to sparta-ll
  - internal omap to store mappings between group id's and recipients?
  - and then you clone to those recipients?

```
send(r: Recipient, m: Message; US:UserStore, MS:MessageStore):

  nexttail <- U(0,2^l - 1) // are these sampling for random values?
  rand <- U(0, 2^l - 1)    // ++

  (head, tail) <- US.update(r, (head, nexttail))
  MS.access(write, rand, (r, tail, nexttail, m))

fetch(r, k; US, MS)
  (first, last) <- US.update(r, (last, last))
  x = first, M = {}
  while |M| < k do
    if x != last then
      (r, curr, next, m) <- MS.access(read, x, null)
      x = next
    else
      (-,-,-,m) <- MS.access(read, dummy, null)
    end if
    M = M union {m}
  end while
  return M
```

step 1 complete a solid implementation of sparta-ll

expansion: implement a secure method for group messaging

how does encryption here work? because im assuming the reciever and the message isnt in plain text, wouldnt I have to work on that too?

problems: I dont have the hardware to run enclaves, pretty sure simulation wont work? intex sgx drivers say they require intel hardware, which i dont have