Table of Contents

01

Table of Contents

04

Background research, meeting prep, design process

17/11/21

08:55–9:20

1. Found an article about the IoT statistics. Have to ask Kevin if it's an appropriate article, because I have only seen research papers like the ones I am reading to have cited similar research papers.
2. Wrote a paragraph of the background research section using a couple of sources Kevin linked.

17/11/21

Design brainstorming (Chen paper)

05

11/18/2021

Ref -

attachment:

11/18/2021

Design brainstorming

Chen paper

Symmetric-key encryption
Yao's garbled circuits

Analysis of current TAPs
- types of sensitive information
- operations on trigger data
- execution model of trigger-action systems
- example rule

Design considerations for providing data confidentiality in trigger-action systems
- threat model and functionality goals
  - security goals
  - security non-goals
  - functionality goals
- Design space exploration
  - computation at the edges
  - secure hardware???
  - homomorphic encryption of the trigger data
  - Secure multi-party computation
  - Secret sharing based SMC

Design of encrypted TAP
- decentralized trust model
- Rule setup
- Circuit garbling
  - crypto details
  - encoding blob
  - garbled circuit
  - encrypted decoding blob
  - encoded user constants
- Rule execution
- Rule execution
  - cryptographic details
- Rationale for Novel TC Protocol & Security Analysis
- Supporting TAP-specific Operations with Yao's Garbled Circuits
  - Input and Output representations
  - Converting DFAs into circuit
  - Supported functions

Evaluation of eTAP
- Performance of basic operations
- Performance of Running Complete Rules
  - Throughput
- Large-scale Evaluation
  - Computation overhead on TC
  - Storage overhead
  - Latency and throughput

Related Work
- Cryptographic techniques for secure computation

Discussion and Limitations
- security against metadata leakage
- Integrating with existing TAPs
- Rule semantics
- Circuit id synchronization
- Loss of the trusted client (TC)
- Circuit usage feedback

done 11/18/20

06

Design brainstorming (Chen paper) continuation

11/18/202

Attachment:

Ruf

11/18/202

trusted generator of garbled circuits → allows to use semi-honest implementations of SFE ???

↳ user's phone periodically generates and transmits circuits to the untrusted TAP

↳ trigger service garbles sensitive data and calls TAP

↳ executes the circuit and contacts the action service

garbled

security checks → executes

treat trigger and action services as semi-honest

computes 93.4% of all rules published on Zapier
100% of 500 most used IFTTT rules.

eTAP also provides mutual secrecy between the trigger and action services
↳ but not for TAP

they have a huge limitation that phone has to transmit 61.7 MB of data per day

their code is available on https://github.com/EsoMadSec/etap

Ruf 11/18/2021

Mentor meeting 11/18/2021

11/18/2021

Preliminary timeline:

Week 1:

Background finish what I can
High-level design overview

Week 2:

Design details:
required crypto protocols
protocol flow → UML sequence diagram

Week 3:

Security Analysis
Specify privacy and functionality goals
Start security proofs

Week 4:

Implementation details: software

---

outputs – public key crypto?

oblivious transfer???

- TO Background:

  TAP- books, components, how they work today

- DBLP to use for citations

Thuy 11/18/2021

Choosing software language

11/22/2021

Read an article on differences between Go and Rust, two languages my mentor mentioned coding on.

Decided to use Rust because:

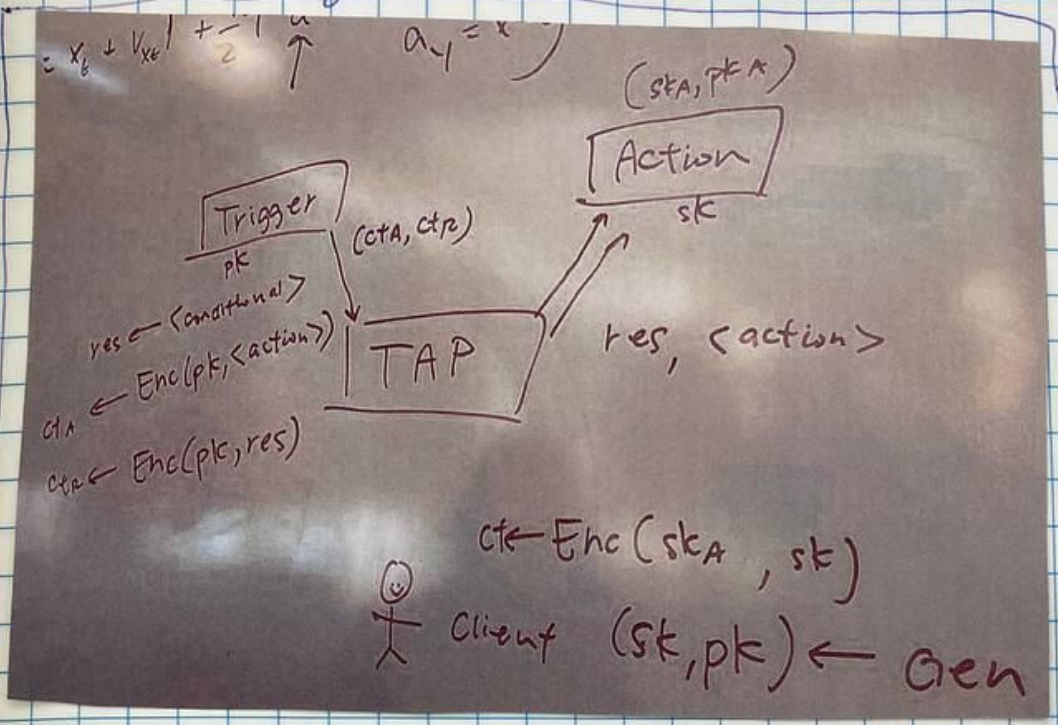
- very fast (if you code really well, you can even reach almost full potential of the hardware
- many more features than in Go
- extensive support system w. guides, videos, users...
- if needed concurrency (like in Go), can be done manually
- ensures high safety, the compiler pings every possible minute bug (which is what we need)
  → Rust gives us the ability to say that we own a specific piece of data; it's not possible for anything else to claim ownership, so we know nothing else will be able to modify it.
- efficient code = you don't write much to do much
- Rust is more suitable for research, and Go is a better go-to for companies and bigger teams
- Go's garbage collection is questionable, and we don't need unnecessary unclarity
- People say they hugely learned Go in two days so if sth goes very badly, we can switch in no time

*(1) Q: do we take advantage of parallel computations?

Ruf 11/22/2021

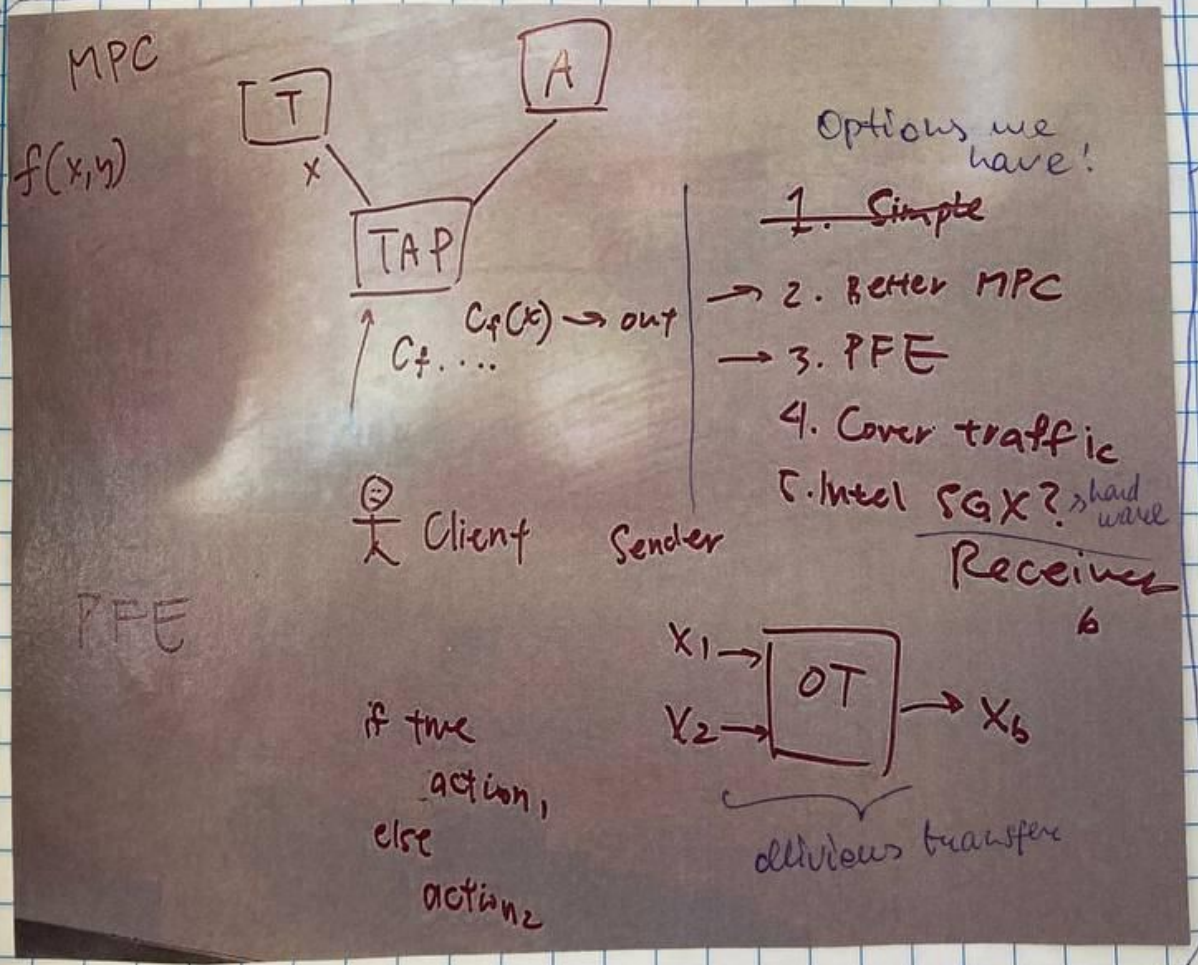
Mentor meeting 11/23/2021

11/23/2021



→ MPC (2)

1 → let the TI do all the work (computation) ✗



Mentor meeting 11/23/2021 (continuation) 11/23/2021

1. Simple is a bit dangerous. I think because Chen et al. paper warned against it.
2 or 3 is a probable way to go
4 is an add-on
5 is an add-on

They are not mutually exclusive except 2 and 3

*Q (01) page 08: crypto uses mostly sequential flow, so concurrency does not really matters.

TO-DO's:

- ☐ point out two papers Kevin gave me access to
- ☐ Read on PFE
- ☐ cite ifttt for services
- ☐ fix bibtex for two articles I've added to the paper
- ☐ continue background and finish everything I know by Wednesday's meeting (?) or this week for sure
- ☐ keep learning Rust and crypto course
- ☐ Read articles in the lit section

Keiji 11/23/2021

Mentor Meeting 12/01/2021 12/01

User sets the polling interval in IFTTT?
investigate!

PFE - how slow is low?
→ a lot worse than MPC

public circuit for MPC

PFE: one party has a private circuit and another one its inputs
hide gates like XOR, and, or, etc.
nothing goes into the trigger service

Figures in LaTeX - learn to do!!!

if $f_1(x, c_1)$ then $(f_2(x, c_2)) \rightarrow$ action
↓ email "confidential" — MPC here

$f_2(x, c_2) \rightarrow$ „text the email to my number"
c_2 - number

TAP: shouldn't learn x, c_1, c_2, and (results?)
TS: shouldn't learn c_1, c_2
AS: x, c_1 c_2?

argue with the datasets!!!

People are mostly executing the function the API is capable of doing

c_1 is not sensitive to leak to TS because we already know it?

but a lot of performance? saves a lot of computations

Mentor Meeting 12/01 (continuation) 12/01

PFE might be too much in conjunction with MPC

Storage/Costs

TAP: circuits for f_2 by client, triggers and actions
- for each service, set of f_1's and f_2's
- when it polls, send c_1 → ciphertext (MPC)

TS: all inputs
computes $f(x, c_1)$ with c_1

AS:
gets the outputs of $f_2(x, c_2)$
participate in MPC (not expensive)

Where do we access the API?

What are the metrics we care about?

- Client: computation, communication to TAP
  the data transmitted
  we want to get lower
  (test through 500 runs and average it)
- TS: computation and communication to TAP
  computation of $f_1(x, c_1)$ in plaintext vs
  generating GC input for x
  communication decreases, sends encryption of a bit
- AS: decrypt $f_2(x, c_2)$

- Motivate that C_1 does not need to be hidden ↓ find a real-world example

Wed 12/01

Mentor meeting 12/08 12/08

TLS - as a communication channel
Think about API usage details

$\mathcal{C}$ - security parameter for security proofs
$1/2^{28}$ is secure today
computational security ↓ find them from testing

use real - ideal paradigm for security proofs

- semi - honest security type

Prof - 12/08

Mentor Meeting 22/22 12/22

trigger computes. p
only compute f in TAP. that involves

overview of our threat models
semi-honest
TAP is malicious

threat model
↓ then
real-ideal paradigm

oblivious transfer → we don't use it here
sender receiver
x_0, x_1 b
sender learns nothing x_b
receiver learns nothing

MPC
Yaobler Evaluator
x y
$f(x,y)$
Encodes the inputs to f of x and y

look at MPC at the original eTAP and
think about optimizations

in Yao's - you have to send a lot at once
I use one because our messages

Next: MPC libraries EMP Toolkit Docker
message formats between parties Rust
TCP protocol

Rust front-end? Mobile app? → IFTTT one
12/22 Rust 12/22

Mentor meeting 12/29 12/29

figure 1 → start from eTAP paper - start thinking of it on paper

do the web server in Rust

store garbled circuit

on a TAP

TCP listener on the AS

somewhere

opt the shell

code w. a library

---

mobile app

what examples do we want to use on our testing?

↳ figure 9 in the eTAP paper

two data sets:

eTAP paper rules + (their + our)

sensitive rules, what are their limitations???

Ref:

12/29

Coding a web-Server in Rust: Single-Threaded

01/12/22

So, I need to learn to code in Rust, a competely new language to me. There is a book on doc.rust-lang.org about all the things I need to know to work with it on a high level.

My mentor, Kevin, sent me a chapter on coding a multithreaded web-server that uses HTTP and TCP/IP protocols for me to practice implementing them for the future, when we actually build the network connections in our THP.

Today, I woke up early and did some work on the first step of coding a single-threaded server.

finished code

main.rs M | 404.html | hello.html

Users > veronikakitsul > Desktop > TAP-research > rust > server > src > main.rs

```rust
use std::io::prelude::*;
use std::net::TcpListener;
use std::net::TcpStream;
use std::fs;

fn main() {
    let listener = TcpListener::bind("127.0.0.1:7878").unwrap();

    for stream in listener.incoming() {
        let stream = stream.unwrap();

        handle_connection(stream);
    }
}

fn handle_connection(mut stream:TcpStream) {
    let mut buffer = [0; 1024];
    stream.read(&mut buffer).unwrap();

    let get = b"GET / HTTP/1.1\r\n";

    let (status_line, filename) = if buffer.starts_with(get) {
        ("HTTP/1.1 200 OK", "hello.html")
    } else {
        ("HTTP/1.1 404 NOT FOUND", "404.html")
    };

    let contents = fs::read_to_string("hello.html").unwrap();

    let response = format!(
        "{}\r\nContent-Length: {}\r\n\r\n{}",
        status_line,
        contents.len(),
        contents
    );

    stream.write(response.as_bytes()).unwrap();
    stream.flush().unwrap();
}
```

01/12/22 pm

Mentor meeting 01/12/2022

$p(x_t, C_t) = 1$, then send $f(x_t, C_a)$ to a.

for some rules: you can have this much privacy

for others: this much

depending on the type

an anonymous token

↳ log in

↓

log off

submit token → TAP knows I'm authorized

the TAP gives away a token but it doesn't know what it gave away

recount the data -?

proxy ⇒ or use two sequential ones?

mixed networks

↳ mix net

how to bring security into Android apps?

check if I installed a lib with brew?

↓ google

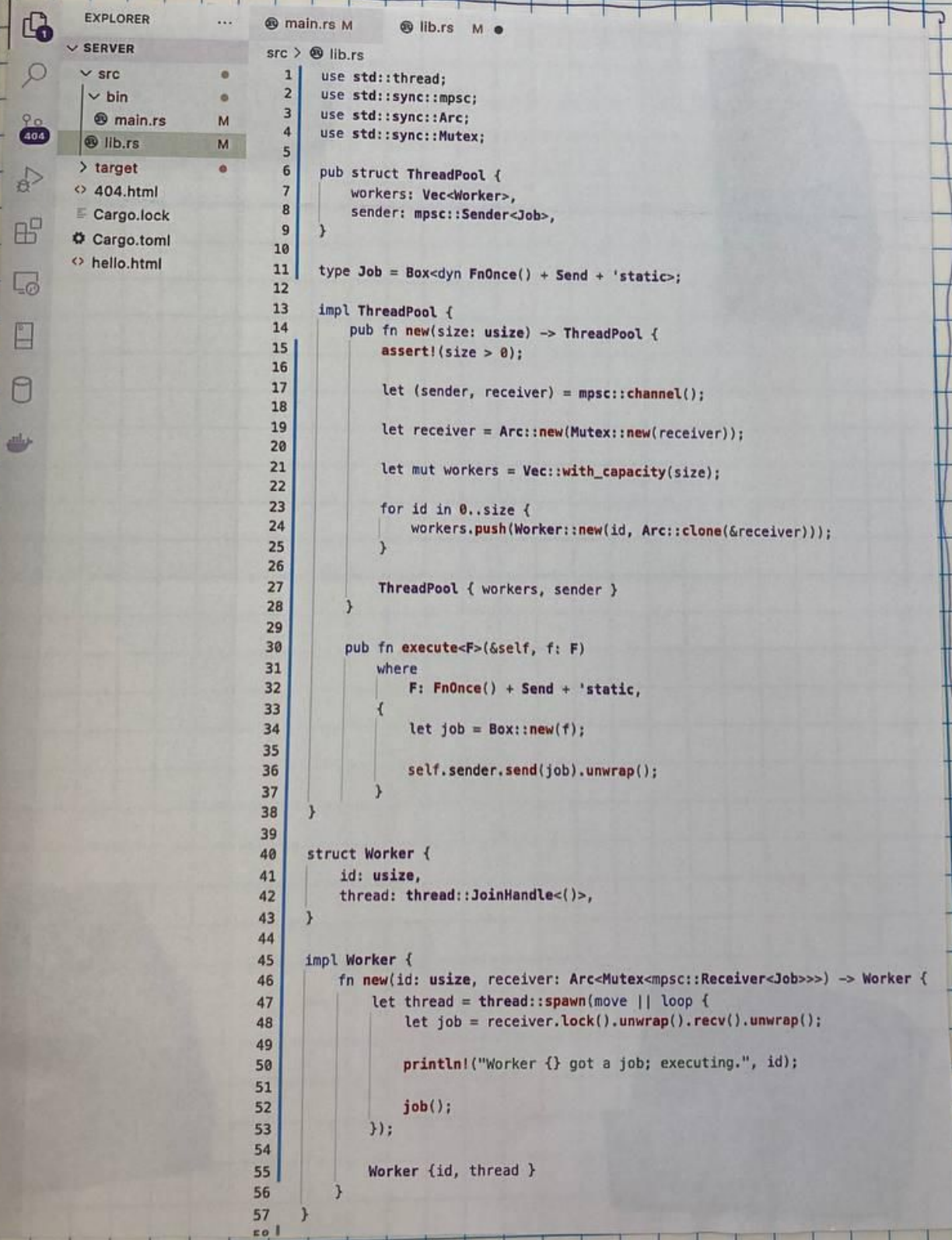
google about routers

01/12

MS

Turning a Single-Threaded server into a Multithreaded 01/17/22

Goals: using Rust online textbook, finish the second part of server making




```rust
use std::thread;
use std::sync::mpsc;
use std::sync::Arc;
use std::sync::Mutex;

pub struct ThreadPool {
    workers: Vec<Worker>,
    sender: mpsc::Sender<Job>,
}

type Job = Box<dyn FnOnce() + Send + 'static>;

impl ThreadPool {
    pub fn new(size: usize) -> ThreadPool {
        assert!(size > 0);

        let (sender, receiver) = mpsc::channel();

        let receiver = Arc::new(Mutex::new(receiver));

        let mut workers = Vec::with_capacity(size);

        for id in 0..size {
            workers.push(Worker::new(id, Arc::clone(&receiver)));
        }

        ThreadPool { workers, sender }
    }

    pub fn execute<F>(&self, f: F)
        where
            F: FnOnce() + Send + 'static,
        {
            let job = Box::new(f);

            self.sender.send(job).unwrap();
        }
}

struct Worker {
    id: usize,
    thread: thread::JoinHandle<()>,
}

impl Worker {
    fn new(id: usize, receiver: Arc<Mutex<mpsc::Receiver<Job>>>) -> Worker {
        let thread = thread::spawn(move || loop {
            let job = receiver.lock().unwrap().recv().unwrap();

            println!("Worker {} got a job; executing.", id);

            job();
        });

        Worker {id, thread }
    }
}
```

finished code

Mufy 01/17/22

Mentor meeting 01/19/2022 → Rust 01/19

anonymous sending : OAth tokens

code for a TCP server that will listen to messages in the loop:

different message types

struct → marshalling dat
{
type
...
}

↓ stuffing a struct into a network packet
packages about it

send message
↓ create message

crypto packages → Rust
encrypt ↔ decrypt

Benchmarking package → Kevin will send it

All the steps!
→ design on paper + guarantees
→ run MPC lib
→ trigger service
→ TAP : let specify the rules
→ action service
→ connect them all
→ client : crypto code

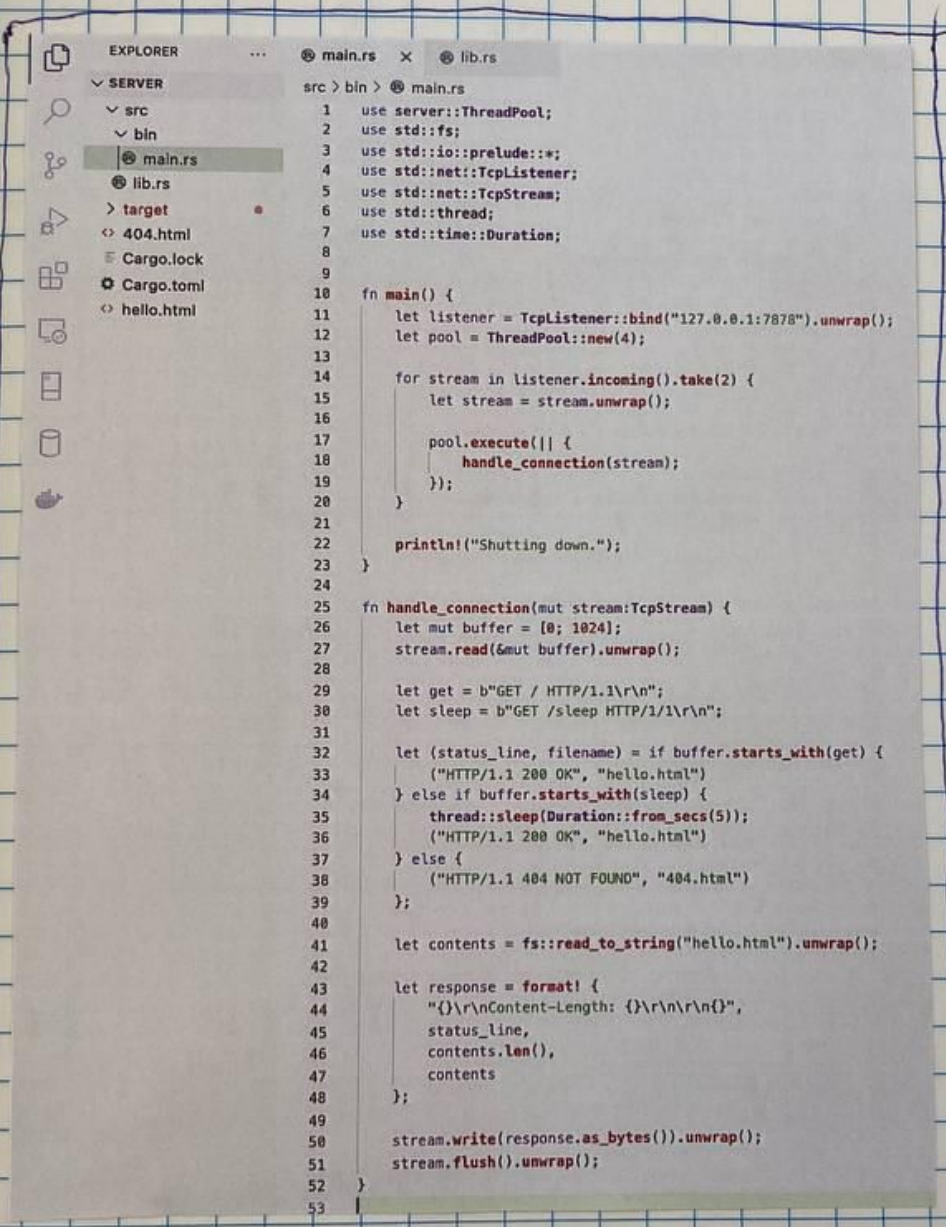
wait for more resources from Kevin

Tue 01/19

Rust Web-Server: Graceful shutdown and cleanup

01/23/22

Today, I'm finishing up coding a web-server through Rust textbook. See code attached on the next pages:

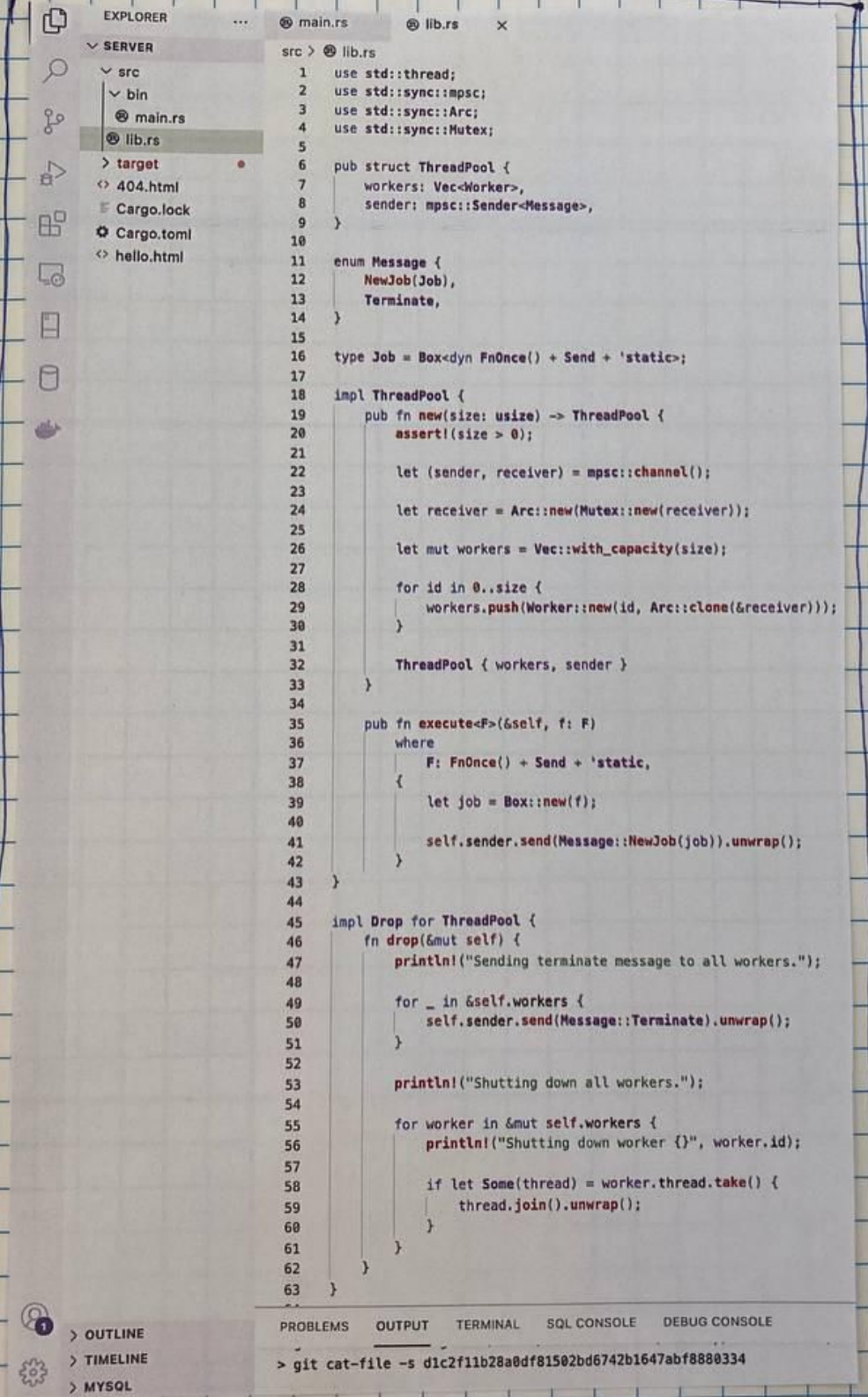


main.rs file with main connections

01/23/2022
Barj

Rust Web-Processor: Graceful Shutdown and Cleanup (cont.) 01/23/22

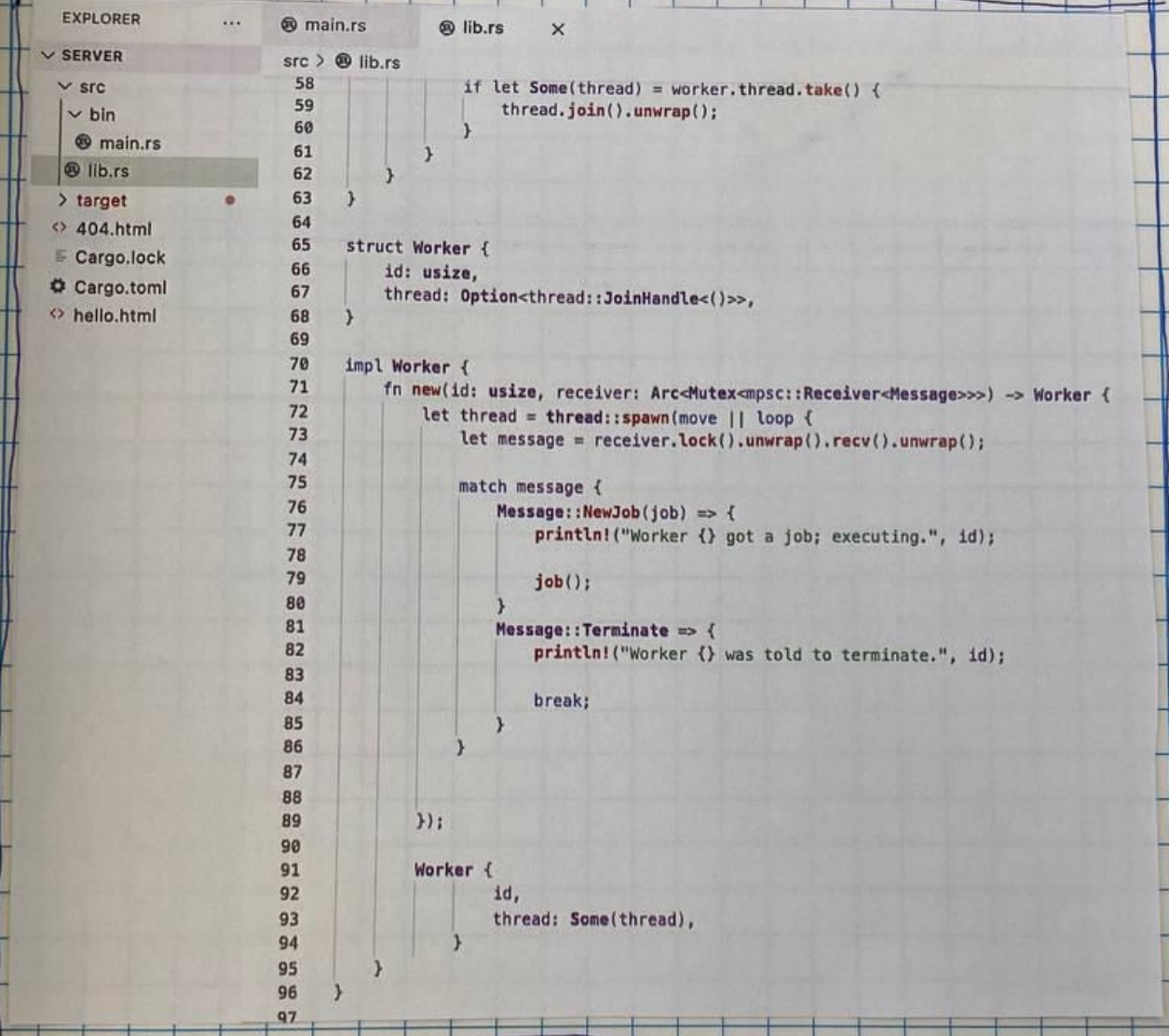
Part 1 of lib.rs file with constructor



01/23/22

Rust Web-Server: Graceful Shutdown and Cleanup cont. 01/23/2022

Part 2 of lib.rs file with constructor:




```rust
                if let Some(thread) = worker.thread.take() {
                    thread.join().unwrap();
                }
            }
        }
    }

    struct Worker {
        id: usize,
        thread: Option<thread::JoinHandle<()>>,
    }

    impl Worker {
        fn new(id: usize, receiver: Arc<Mutex<mpsc::Receiver<Message>>>) -> Worker {
            let thread = thread::spawn(move || loop {
                let message = receiver.lock().unwrap().recv().unwrap();

                match message {
                    Message::NewJob(job) => {
                        println!("Worker {} got a job; executing.", id);

                        job();
                    }
                    Message::Terminate => {
                        println!("Worker {} was told to terminate.", id);

                        break;
                    }
                }


            });

            Worker {
                    id,
                    thread: Some(thread),
                }
        }
    }
```

01/23/2022

My-

Rust crates and other resources: benchmarking 01/24/22

Today, I'm looking at all of the resources kevin sent me to get familiar with.

1. Benchmarking - run program to assess a performance by running tests for it.
I'm going through tutorial on
bheisler.github.io/criterion.rs/book/

Q: what is the standard benchmark harness ???

Procedure:
1) create a simple fibonacci program:

main.rs src 1, U main.rs ~/.../benchmark-criterion-pract/... U

Users > veronikakitsul > Desktop > TAP-research > rust > benchmark-criterion

```rust
fn main() {
    fibonacci(50);
}

fn fibonacci(n: u64) -> u64 {
    match n {
        0 => 1,
        1 => 1,
        n => fibonacci(n-1) + fibonacci (n-2),
    }
}
```

2) Add dependency to Cargo.toml:

```
[dev-dependencies]
criterion = "0.3"

[[bench]]
name = "my_benchmark"
harness = false
```

3) run cargo bench with the code in "project-name"/benches/my_benchmark.rs
✓ fails !!! → emailed kevin, can't find how to fix

01/24/22 finish

Rust - Crypto Crate

01/24/22

I will try to work with Rust „crypto" crate today.

to install: add „rust-crypto = "^0.2"
to [dependencies] in Cargo.toml
and extern crate crypto;
to root

Okay but I don't understand how to use it. And I can't find any sources on how to do it either, so I will ask Kevin on wednesday.

01/24/22

Mentor meeting 01/26/2022 : Rust networking + crypto 01/26

how long does it take to do specific encryptions: hybrid, public-key, secret-key :?

Run benchmarks?

I can do mobile app skeleton.

Running the MPC lib

↳ Docker image creation

My next to-dos:

- run different encryptions
- create their benchmarks
- keep learning Rust

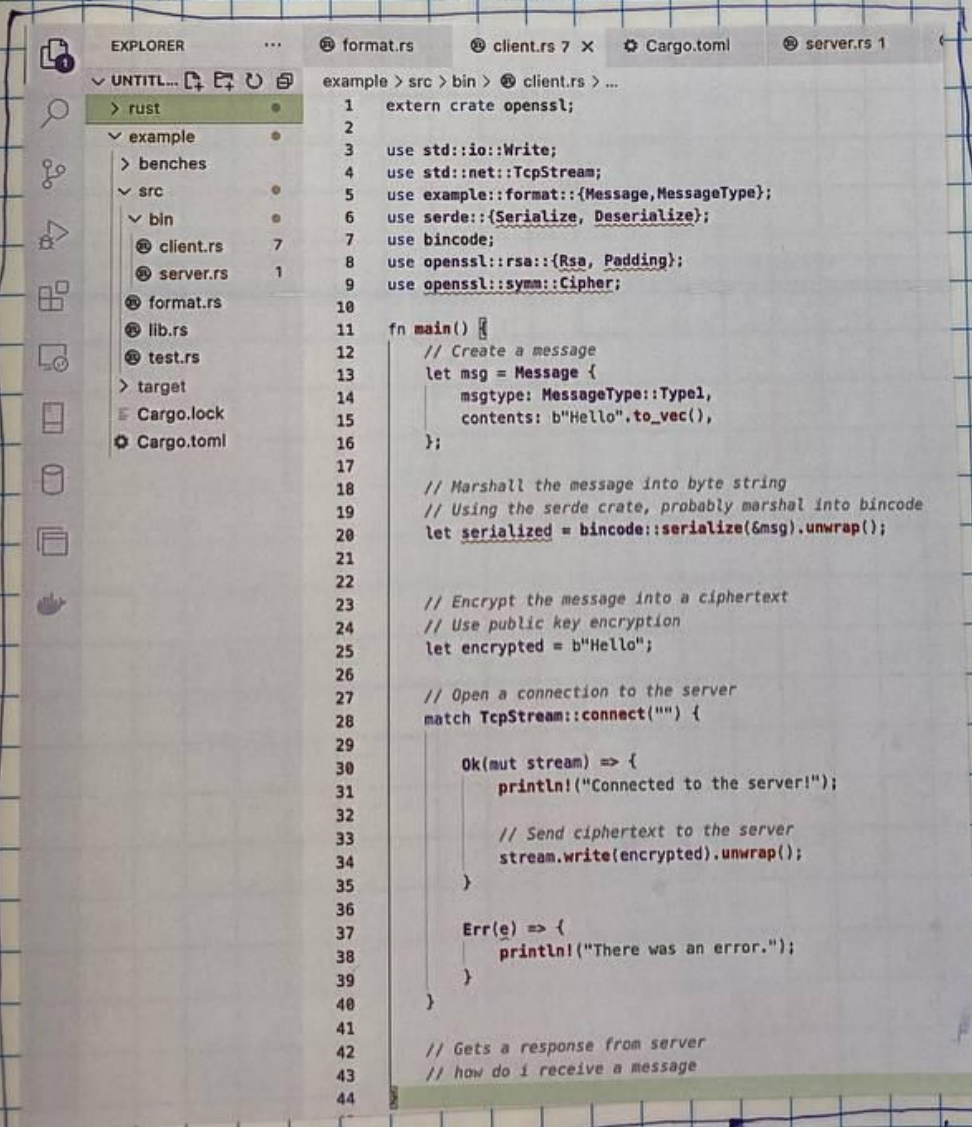
Rust

01/26

Creating a simple test server: marshal, encrypt-decrypt 01/31

Today, I need to learn to marshal (serialize) data to uniformly transmit it. Because I will transmit a struct data via server, I need to make sure it will be read properly.
To do that, I use serde crate.

1) Do a lot of research on the web to find some fitting examples.
2) Realized I have to marshal struct into bytes.
3) See code attached:



client.rs

Code is running with no mistakes, albeit it is not useful yet.

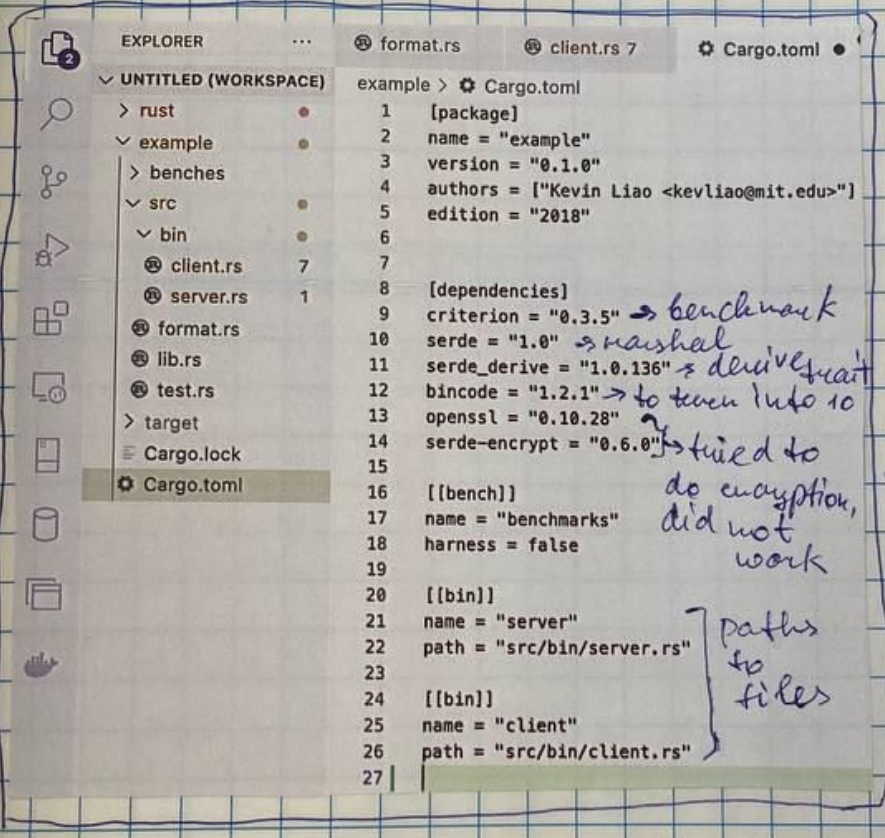
Bye

01/31/2022

Cargo.toml configurations to make two files connect 01/31

Cargo.toml file shows all configurations, packages versions, settings for imported crates, and paths to files to run them properly.

From previous page on marshaling, these are the configurations I ended up with:



Buff- 01/31/2022

Mentor Meeting 02/02 : HPKE, FFI, meetings logistics, MPC 02/02

- try to do hpke encryption
- Mobile code and server → all good because it's through network
- Rust and C++ → foreign function interface
- Generate secure keys ? → hpke lib ? or maybe another library
- Meetings in-person : Mon, Wed, Fri would be best for Kevin
- MPC library setup via docker :
  1) git clone emp repo
  2) docker build in the folder
  3) docker run to run the image
  4) execute tests with ./run ./bin/test_bit

  ↳ more useful examples to follow from Kevin.
- Veronika : background and introduction complete more.

bcf 02/02/2022

HPKE hybrid encryption

02/03

Today, I will try to do hpke hybrid encryption that I will need to do for my TAP.
There's no decent documentation of it, so I'm looking for examples on the web.
Found 2 examples: one on github and one on docs.rs
However, none of them fits perfectly for what I'm trying to do, so I'm combining them and looking at compiler feedback to fix mistakes.
As of now, I have so many mistakes, but I'll try to fix them as I read more on it and hpke.
The code I ended up today with:

EXPLORER
EXAMPLE
- benches
- src
  - bin
    - client.rs 8, M
    - server.rs 1, M
  - format.rs
  - lib.rs
  - test.rs
- target
- Cargo.lock M
- Cargo.toml M

client.rs 8, M | Cargo.toml M | server.rs 1, M

src > bin > client.rs > ...

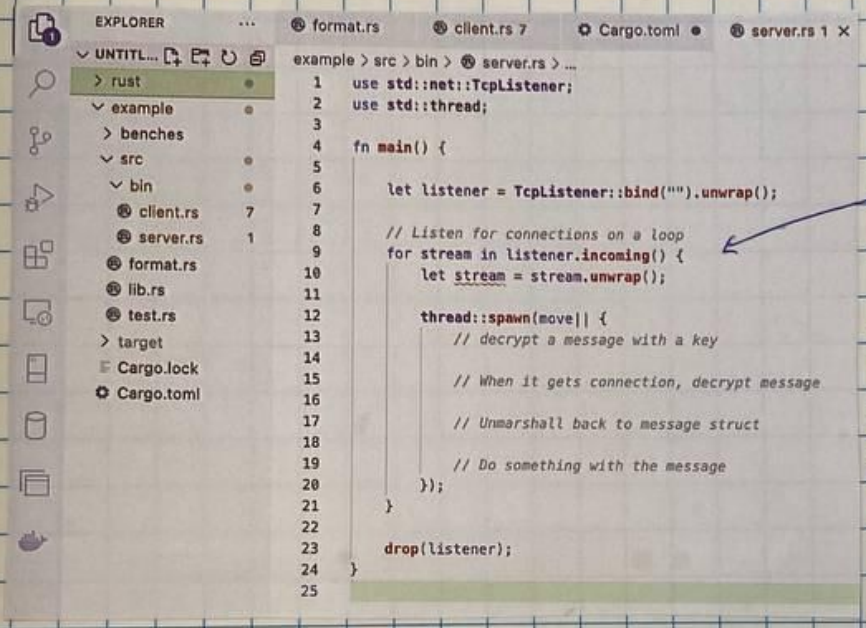
```rust
use std::io::Write;
use std::net::TcpStream;
use example::format::{Message,MessageType};
use serde::{Serialize, Deserialize};
use bincode;
use rand::prelude::*;
use hpke::{
    aead::{AeadTag, ChaCha20Poly1305},
    kdf::HkdfSha384,
    kem::X25519HkdfSha256,
    Deserializable, Kem as KemTrait, OpModeR, OpModeS, Serializable,
};


fn main() {
    // Create a message
    let msg = Message {
        msgtype: MessageType::Type1,
        contents: b"Hello".to_vec(),
    };

    // Marshall the message into bincode with Serde
    let serialized = bincode::serialize(&msg).unwrap();


    // Encrypt the message into a ciphertext by hybrid encryption
    type Kem = X25519HkdfSha256;
    type Aead = ChaCha20Poly1305;
    type Kdf = HkdfSha384;
    let mut csprng = StdRng::from_entropy();
    let info_str = b"first message to the sever";
    let server_pk = b"aefejojTajvodha3795u9";

    let (encapsulated_key, mut encryption_context) = hpke::setup_sender::<Aead, Kdf, Kem, _>
                                                    (&OpModeS::Base, &server_pk, info_str,
                                                    &mut csprng).expect("invalid server pubkey!");


    let aad = b"First encrypted message";
    let mut ciphertext = serialized.to_vec();
    let tag = encryption_context.seal_in_place_detached(&mut ciphertext, aad).expect("encryption failed!");

    // Open a connection to the server
    match TcpStream::connect("") {

        Ok(mut stream) => {
            println!("Connected to the server!");

            // Send ciphertext to the server
            stream.write(&ciphertext).unwrap();
        }

        Err(e) => {
            println!("There was an error.");
        }
    }
```

Ruff 02/03

Server.rs → listening for connection on the loop 02/03

Writing up server.rs looks like more work than client.rs. I need to listen to connections on the loop as the first step, which is what I'm planning to do today.

The code that did not give mistakes at build time:



need to build thread pool

server.rs

Buy. 02/03/2022

HPKE encryption -> successful

02/13

Today I continue working on encrypting a message with HPKE crate.

I was also to do a successful encryption by following github.com/rozbb/rust-hpke/blob/master/examples/client_server.rs and fixing mistakes with compile time.

Here's the code from today:

client.rs 8, M | Cargo.toml M | server.rs 1, M

src > bin > client.rs > ...

```rust
use std::io::Write;
use std::net::TcpStream;
use example::format::{Message,MessageType};
use bincode;
use rand::prelude::*;
use hpke::{
    aead::{AeadTag, ChaCha20Poly1305},
    kdf::HkdfSha384,
    kem::X25519HkdfSha256,
    Deserializable, Kem as KemTrait, OpModeR, OpModeS, Serializable,
};

type Kem = X25519HkdfSha256;
type Aead = ChaCha20Poly1305;
type Kdf = HkdfSha384;

const INFO_STR: &[u8] = b"example session";

// initialize the server with a key pair
fn server_init() -> (<Kem as KemTrait>::PrivateKey, <Kem as KemTrait>::PublicKey) {
    let mut csprng = StdRng::from_entropy();
    Kem::gen_keypair(&mut csprng)
}

fn encrypt_msg(
    msg: &[u8],
    aad: &[u8],
    server_pk: &<Kem as KemTrait>::PublicKey,) -> (<Kem as KemTrait>::EncappedKey, Vec<u8>, AeadTag<Aead>) {
        let mut csprng = StdRng::from_entropy();

        // encapsulate a key and use the resulting shared secret to encrypt a message
        // encrypt with AEAD context
        let (encapsulated_key, mut sender_ctx) = hpke::setup_sender::<Aead, Kdf, Kem, _>
                                                 (&OpModeS::Base, &server_pk, INFO_STR,
                                                 &mut csprng).expect("invalid server pubkey!");
        // seal in place will encrypt the plaintext in place if success
        let mut msg_copy = msg.to_vec();
        let tag = sender_ctx.seal_in_place_detached(&mut msg_copy, aad).expect("encryption failed!");

        let ciphertext = msg_copy;
        println!("ciphertext: {:?}", ciphertext);
        // return
        (encapsulated_key, ciphertext, tag)
    }


fn main() {
    // set up the server
    let (server_privkey, server_pubkey) = server_init();

    // Create a message
    let msg = Message {
        msgtype: MessageType::Type1,
        contents: b"Hello".to_vec(),
    };

    let aad = b"First encrypted message";

    // Marshall the message into bincode
    let serialized: Vec<u8> = bincode::serialize(&msg).unwrap();
    // looks good
    println!("serialized: {:?}",serialized);

    let (encapsulated_key, ciphertext, tag) = encrypt_msg(&serialized, aad, &server_pubkey);
    let encapsulated_key_bytes = encapsulated_key.to_bytes();
    let tag_bytes = tag.to_bytes();
```

Tue
02/13

Introduction section, Title

02/14

Today, I am planning on working with direct paper writing.
For the fair that is coming up soon I need to turn in a coherent paper, so I will work on the Introduction part now.
Another important thing I need to devise is paper's name and TAPs name.
For paper, one idea is: "IFTTT alternative: trigger-Action platform based on Multi-Party Computation Garbled Circuits and ——"

One of the most important things for my background is to find a compelling example showing the TAP usage. One interesting one could be "Post new emails received in Gmail to Facebook page" from real Zapier applets.

Also found more papers about market analysis of other TAPs and their security flows.

I feel like my Introduction is mostly done by now, just needs editing. Also will look at Lauren's and Brian's comments.

Bauj

Mentor Meeting 02/16 : paper writing, tokens, Ubuntu 02/16

Run Ubuntu VM → Virtual box

Use TOR for installing a rule → but not store anything else

Recipe-specific tokens - transfer tokens,
XTokens

Paper writing :
edit introduction
write background
write design

write by 02/26 → the slides

paper title: used Acronimify to create one:

ACACIA: A Practical Privacy-Preserving
Trigger-Action Platform using
Multi-Party Computation

Buy 02/16

server.rs → server TCP connection, decryption 02/17

Today, I'm trying to build a TCP server to connect a client to it.
I also need to create a decryption function to decrypt a marshaled message.
Here's what I was able to accomplish today:
There are some mistakes, but I know how to fix most of them, just don't ~~kno~~ have the time!

example > src > bin > server.rs > ...

```rust
use std::net::{TcpListener, TcpStream, Shutdown};
use std::thread;
use std::io::{Read, Write};
use hpke::{
    aead::{AeadTag, ChaCha20Poly1305},
    kdf::HkdfSha384,
    kem::X25519HkdfSha256,
    Deserializable, Kem as KemTrait, OpModeR, OpModeS, Serializable,
};

fn decrypt_msg(
    server_sk_bytes: &[u8],
    encapped_key_bytes: &[u8],
    ciphertext: &[u8],
    aad: &[u8],
    tag_bytes: &[u8],
) -> Vec<u8>
{

    let server_sk = <Kem as KemTrait>::PrivateKey::from_bytes(server_sk_bytes).
                expect("could not deserealize server privkey");
    let tag = AeadTag::<Aead>::from_bytes(tag_bytes).expect("could not deserialize AEAD tag");
    let encapped_key = <Kem as KemTrait>::EncappedKey::from_bytes(encapped_key_bytes).
                expect("could not deserialize the encapsulated pubkey");

    // decapsulate and derive the shared secret to create AEAD context
    let mut receiver_ctx = hpke::setup_receiver::<Aead, Kdf, Kem>
            (&OpModeR::Base, &server_sk, &encapped_key, INFO_STR).expect("failed to set up receiver!");

    let mut plaintext = ciphertext.to_vec();
    receiver_ctx.open_in_place_detached(&mut plaintext, aad, &tag).expect("invalid ciphertext");

    plaintext
}

fn handle_client(mut stream: TcpStream) {
    let mut data = [0 as u8; 2000];
    while match stream.read(&mut data) {
        Ok(size) => {
            decrypt_msg(data);
            true
        }
        Err(e) => {
            println!("An error occured, terminating connection with {}", stream.peer_addr().unwrap());
            stream.shutdown(Shutdown::Both).unwrap();
            false
        }
    } {}
}

fn main() {
    let listener = TcpListener::bind("127.0.0.1").unwrap();

    //pass key

    // Listen for connections on a loop
    for stream in listener.incoming() {
        match stream {
            Ok(stream) => {
                println!("New connection: {}", stream.peer_addr().unwrap());
                thread::spawn(move || {

                    handle_client(stream)

                    // decrypt a message with a key

                    // When it gets connection, decrypt message

                    // Unmarshall back to message struct

                    // Do something with the message

                });
            }
            Err(e) => {
                println!("Error: {}", e)
            }
        }
        drop(listener);
    }
}
```

02/17 Rdy

02/17 Rdy

Report: writing out most parts: design, background 02/21

Today, I met with Kevin to write out the report parts because it's due shortly.

I finished the Background section's subsections on Trigger-Action Systems, Market Analysis by the paper Rhema has sent me, and on Multi-Party Computation.

I think I need to write more on the garbled circuits part.

Kevin worked more closely with the overview and goals, design, and evaluation.

I have also drafted an abstract.

I will also need to add credits to Ms. Lohwater and STEM Fellowship, why we chose Rust, add on Related Work, and edit other parts as well

I also think that I need more extensive literature review to add credibility.

02/21 MY