Attachment 1

Project task schedule

1.1 Winter semester

1st-4th week	Exploring ZK ecosystem and finding suitable ZK theme
5th-8th week	Researching ZK-EVM, ZK-Rollups, Stealth Addresses
9th-11th week	Researching Stealth Addresses with ZKP scheme
12th-13th week	Writing Analysis and Solution design

1.2 Summer semester

1st-2nd week	Learning CIRCOM
3rd-4th week	Implementing ZKP with CIRCOM
5th-7th week	Implementing smart contracts in Solidity
8th-9th week	Implementing browser experimentation wallet
10th-11th week	Solution testing
12th-13th week	Revision of final document

The beginning of this work started in different direction. First application of ZKPs in the ZK-EVM or in ZK-Rollups was being researched and considered.

However, after discussion with supervisor, these themes were considered too advanced and a more suitable theme for Stealth Addresses with ZKP was chosen. The remainder of the winter semester was split between researching how a stealth address scheme with ZKPs can be designed and implemented, and then the solution design was proposed.

Start of the summer semester was more focused on learning CIRCOM for implementing the a circuit needed to generate and verify ZKPs. Then the main part of the semester was spent implementing the ZKP circuit, implementing set of smart contracts and finally a proof of concept browser wallet supporting implemented scheme. Last four weeks were spent on testing, fixing bugs and revising this document.

Attachment 2

Setup guide

The following guide was done in a Ubuntu 24.04 docker image. To run this setup in different environment, install necessary requirements listed bellow and follow rest of the guide (all commands in the guide are ran under root user, consider using `sudo` if not you are not running them under root).

One remark, this guide shows setup for the whole project, including compiling circuits, contracts and interacting with deployed ones. You can skip the circuits and contracts and go directly to the 2.5 after the project initialization to interact with already deployed contracts.

2.1 Requirements

1. **Linux with at least kernel version 6** (other versions may or may not work).
2. **Git**
3. **Makefile**

4. **Node version 20.10.0 or higher**
5. **NPM version 10.2.3 or higher**
6. **Rust version 1.77.0 or higher** - Installation guide
7. **Circom version 2.1.18 or higher** - Installation guide
8. **SnarkJS version 0.7.3 or higher** - Same link as Circom installation guide, bottom of the page
9. **Foundry version 0.2.0 or higher** - Installation guide

2.2 Initialize project

Start the Ubuntu 24.04 docker image with (or any other preferred way of staring a docker image):

```
docker run -it -p 4173:4173 ubuntu:24.04
```

Firstly, to setup environment run these inside the running container:

```
apt update && apt upgrade -y
apt install git make unzip curl wget -y
```

To install Node, NVM is used, as it is easiest way to manage Node versions:

```
curl -o-
https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.7/install.sh
| bash
export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"  # This loads
nvm
```

```
[ -s "$NVM_DIR/bash_completion" ] && \.
"$NVM_DIR/bash_completion" # This loads nvm bash_completion
nvm install 20.10.0
node --version # output 20.10.0
npm --version # output 10.2.3
```

Install Rust with Rustup:

```
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
source root/.cargo/env
rustup --version # output 1.27.0
rustc -V # output 1.77.0
cargo -V # output 1.77.0
```

Install Circom, there are two options, either build it from source, or download the Linux binary. To build from source, please refer to this Circom installation guide. In this setup, the binary will be downloaded

```
wget
https://github.com/iden3/circom/releases/latest/download/circom-linux-amd64
chmod 777 circom-linux-amd64
mv circom-linux-amd64 /usr/local/bin/circom
circom --version # output 2.1.18
```

Install SnarkJS, this tool can be downloaded via npm as a global package:

```
npm install -g snarkjs
snarkjs # long output...
```

Install Foundry:

```
curl -L https://foundry.paradigm.xyz | bash
source /root/.bashrc
foundryup
forge --version # output 0.2.0
```

Download the project, either via git:

```
git clone --recurse-submodules
https://github.com/Nesquiko/ZK-in-blockchain-Bachelor-thesis.git
```

Or unzip (if you are running this setup inside docker, see docker cp command for copying the zip from host machine into the container) submitted `BP_LukasCastven.zip` file.

```
mkdir bp
mv BP_LukasCastven.zip bp
cd bp
unzip BP_LukasCastven.zip
```

2.3 Compile circuits

To compile circuits navigate from project root to `circuits` and run:

```
make prover
```

The command should end with these lines:

```
[INFO]  snarkJS: ZKey Ok!
snarkjs zkey export verificationkey
./ownership_final.zkey ./verification_key.json
[INFO]  snarkJS: EXPORT VERIFICATION KEY STARTED
[INFO]  snarkJS: > Detected protocol: groth16
```

```
[INFO]  snarkJS: EXPORT VERIFICATION KEY FINISHED
rm ownership_0000.zkey ownership_0001.zkey
cp ./build/ownership_js/ownership.wasm ../stealth-wallet-app/public
cp ./ownership_final.zkey ../stealth-wallet-app/public
```

2.4 Compile smart contracts

To compile smart contracts navigate from project root to `stealth-wallet` and run:

```
forge compile
```

As these contracts are already deployed on Sepolia, you don't have to deploy them, but if you want, then create a `.env` which looks like this:

```
SEPOLIA_RPC_URL=<YOUR-SEPOLIA-RPC-URL>
PRIVATE_KEY=<YOUR-PRIVATE-KEY>
ETHERSCAN_API_KEY=<YOUR-ETHERSCAN-API-KEY>
```

And then run this commnad:

```
make deploy-sepolia
```

2.5 Run web browser wallet

To run the wallet, first navigate to `stealth-wallet-app` and create a `.env` file which looks like this (these private keys are random ones, they may contain some funds on some mainnet, in this project they were used only as a testing ones and already have some Ether on Sepolia):

```
VITE_SEPOLIA_RPC=<YOUR-SEPOLIA-RPC-URL>
```

```
VITE_ALICE_PK=0xff56fc4f1ee05fca64b57dfa70cd3362af082024e2cb10e6507bb7fa0781887d
VITE_BOB_PK=0x91a03d17e4436b2bafabbdd84335c3086c313e5c24122804ce4de94957502981
VITE_BOB_PK_2=0xb91317c163be14ee7a2d39208e813a81eb335a34536329309340f4da821840dc
```

Then run these commands:

```
npm i
npm run build
npm run serve
```

Alice's sender part can be accessed on http://localhost:4173/alice, and Bob's receiver part can be accessed here http://localhost:4173/bob.

Then just copy Bob's primary address, paste it into Alice's search and send some Ether. The average time for this process to be done is around 3 blocks, because the RPC url can sometimes put the transactions in next block. But it should not take more than one minute. After you get a confirmation popup on Alice's part, you can refresh Bob's tracked stealth addresses.

Attachment 3

Contents of the digital medium

Registration number of the thesis in the information system: FIIT-16768-116160

Contents of the digital medium (ZIP archive):

Folder	Contents
/circuits	Circom circuits
/circomlib	Circom library
/docs	Latex documentation source codes
/bachelor	Bachelor thesis latex source code
/iitsrc	Latex source code for IITSRC paper
/iitsrc-poster	PDF of a IITSRC poster
/stealth-wallet	Foundry project
/broadcast	Information about deployed contracts
/lib	Foundry and OpenZeppelin libraries

```
    /script                 Smart contracts for deploying
    /src                    Stealth addresses smart contracts
    /test                   Foundry test
/stealth-wallet-app     Web browser stealt wallet
    /public                 Static files
    /script                 Scripts for setup
    /src                    Typescript source files
    /test                   Unit tests
```

Name of the submitted archive: BP_LukasCastven.zip.