Introduction

We have learned about the classic secret sharing scheme due to Shamir [1] . We also learned about the bilinear pairing-based threshold signature scheme due to Boldyreva [2], which is the threshold variant of the pairing-based signature scheme due to Boneh, Lynn, and Sacham [3]. Here on, we will refer to it as the BLS threshold signature scheme.

In this homework, you will implement the BLS threshold signature in a modular way using the `python` programming language. We recommend using `python` version 3.6 or higher.

NOTE: In this homework, you will only implement only the cryptographic operations of the BLS threshold signature scheme. You are **not required** to do any network programming to finish this homework.

Setup Instructions

This library depends on `numpy` and `py_ecc`. You can install it with

`pip install -r requirements.txt`

You can also manually install the dependencies.

The `finitefield` module is taken from `https://github.com/initc3/babySNARK` You can look at the original repository for examples on how to use it.

How to use py_ecc: https://github.com/ethereum/py_ecc

You can test your setup by running

```
python main.py
```

Code structure:

```
-- finitefields/
-- bls.py
-- bls_ths.py
-- requirements.txt
-- shamir.py
-- main.py
-- utils.py
```

The `finitefields` folder consists of files that implements the field operations that will help you add, subtract, multiply and compute multiplicative inverses of the field elements.

The `bls.py` implements the `bls` signature (non-threshold variant) scheme. You can use this as a reference to implement the threshold variant of the protocol.

The `main.py` is the entry point of this library. We are currently using the `main.py` to test the implementation of the `bls` signature scheme.

The `utils.py` import and defines some basic functionalities which will be useful to implement the

Homework Requirements

You will be changing the `shamir.py` and `bls_ths.py` files. Specifically, you will implement the following functions. Refer to the specific file for more details on the requirements.

In `shamir.py` you will implement:

```
-- gen_share(...)
-- interpolate_at_0(...)
-- interpolate_at_j(...)
-- interpolate_at_g0(...)
-- interpolate_at_gj(...)
```

In `bls_ths.py` you will implement:

```
-- generate_bls_ths_keys(...)
-- partial_sign(...)
-- aggregate_signature(...)
-- verify(...)
```

Throughout this assignment we will use `bls12381` pairing friendly elliptic curve and `sha256` hash function. You can read more (optional) about the `bls12381` curve here https://hackmd.io/@benjaminion/bls12-381

References

[1] Shamir, Adi. "How to share a secret." Communications of the ACM 22.11 (1979): 612-613.

[2] Boldyreva, A. (2003, January). Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In International Workshop on Public Key Cryptography (pp. 31-46). Springer, Berlin, Heidelberg.

[3] Boneh, Dan, Ben Lynn, and Hovav Shacham. "Short signatures from the Weil pairing." International conference on the theory and application of cryptology and information security. Springer, Berlin, Heidelberg, 2001.