Universität Basel
Department of Mathematics
and Computer Science **Foundations of Distributed Systems** Fall 2024 **Exercise sheet 6**

Lecturers
Prof. Dr. Florina Ciorba. (mailto:florina.ciorba@unibas.ch) Office 06.007
Prof. Dr. Heiko Schuldt (mailto:heiko.schuldt@unibas.ch) Office 06.005
Prof. Dr. Christian Tschudin (mailto:christian.tschudin@unibas.ch) Office 06.002
Prof. Dr. Isabel Wagner (mailto:isabel.wagner@unibas.ch) Office 05.009

Assistant
Shiva Parsarad (mailto:shiva.parsarad@stud.unibas.ch)

Release: 18 November 2024
Deadline: 2 December 2024, 23:55 (ADAM)

Requirements

In this exercise, you will work with **homomorphic encryption**, a state-of-the-art mechanism for computations on encrypted data. You will implement a homomorphic cryptosystem and implement two applications using three different homomorphic cryptosystems.
Requirements common to all tasks:

(a) Your implementations must be in Python 3.
(b) The only import statements you may use are:

```
import numpy as np
import concrete.numpy as cnp
import random
import phe as paillier
```

(c) The phe library can be installed with `pip install phe`
(d) The Concrete Numpy library can be installed with `pip install concrete-numpy`
(e) Create a brief manual that explains how to run your code and interpret the output. If you do not provide this manual, you will not get any points.

The solutions for this exercise must be delivered in a zip or tar file containing:

- All source codes containing the solutions.
- A single PDF file containing the written answers and the manual.

1 The ElGamal cryptosystem (8 points)

Write code to:

- Implement key generation, encryption, and decryption for the ElGamal cryptosystem.
- Implement multiplication of ElGamal ciphertexts.
- Write test cases to show the correctness of your implementation.

More information about the ElGamal cryptosystem is available online:
https://en.wikipedia.org/wiki/ElGamal_encryption

2 Application of partially homomorphic encryption: Paillier (4 points)

Use the implementation of the partially homomorphic Pailler encryption scheme from the `phe` library to implement a shopping cart system such that:

- The client sets up homomorphic encryption (generates keys)
- The client populates the shopping cart and encrypts the quantities for each item
- For simplicity, we assume that the shopping cart consists of a list of tuples with the price and quantity for items, i.e.,

```
cart = [
  # (price, quantity)
  (2000, 1),
  (120, 5),
  (1999, 3),
]
```

- The server computes the encrypted total cost of the shopping cart and returns the value
- The client decrypts the total cost

Write code to:

- Implement the class Client and three methods: `generate_paillier_keypair`, `encrypt_cart`, and `decrypt_total`
- Implement the class Server and the method `compute_encrypted_total`
- Implement the interaction between client and server as described above

Explain:

- Which operations does the Pailler encryption scheme support?
- Which data types does it operate on?

3 Application of partially homomorphic encryption: ElGamal (4 points)

Change the shopping cart scenario to work with the ElGamal cryptosystem.
Write code to:

- Implement the classes `Client_EG` and `Server_EG` as well as the interaction between client and server similar to Question 2.

Explain:

- Which changes did you have to make?
- What do your changes mean for the client (which advantages and/or disadvantages does the client have)?
- *Hint:* Which operations can the server perform? Which operations would be left for the client? Which operands can be encrypted, in comparison to the Paillier implementation?

UNI
BASEL

4 Fully homomorphic encryption (4 points)

Write code to:

- Use the Concrete Numpy library (`https://pypi.org/project/concrete-numpy/`) to compute the mean over a list of six encrypted integers.
- Implement a second version that gives you a mean with a precision of two decimal digits.
- Provide test cases to show the correctness of your implementation

Explain:

- How can you obtain the mean of a list with seven encrypted integers?
- What are the limitations of your implementations?