Distributed Systems **Spring-Summer 2017**

Lab **Simon Razniewski/Florian Hofer**

21. RPI Lab III: Distributed MD5 Cracker

MD5 (= Message Digest algorithm 5) is an algorithm that computes for a given string (e.g., a file, a part of a file, or a password) a 128-bit hash value.

The goal of this lab is to start implementing Assignment 4, a distributed MD5 cracker. Given a MD5 hash value your software should be able to find an (numeric) String with that hash value by using a simple brute-force approach. The software should be able to take advantage of multiple, networked computers to distribute the compute load.
MD5 hashes can be computed as follows:

```
import java.security.*;
..
MessageDigest md = MessageDigest.getInstance(”MD5”);
byte[] bytesOfString = yourString.getBytes(”UTF−8”);
byte[] theHash = md.digest(bytesOfString);
```

To obtain the MD5 hashes to be cracked, your system should contact the lab teacher’s machine using the following RMI interface:

```
package server;

import java.rmi.Remote;

public interface ServerCommInterface extends Remote {

        public void register(String teamName, ClientCommInterface cc) throws Exception;

        public void submitSolution(String name, String sol) throws Exception;

}
```

With the method *register*, your system would register to the server, providing your teamname and an object reference that the server can use to give you tasks. With the method *submitSolution*, you can submit solutions to the server.
The ClientCommInterface must implement a single method with which the server can give you tasks:

```
package client;

import java.rmi.Remote;

public interface ClientCommInterface extends Remote {

        void publishProblem(byte[] hash, int problemsize) throws Exception;

}
```

For simplicity, passwords will be strings of numbers, e.g. “53071” or “1234567”. The parameter *problemsize* will tell you the maximal integer that the password may represent. Only one machine of your team can connect to the server.

Tasks

1. Copy the interfaces from the description.
2. Implement the interfaces.
3. Connect to the lab teachers machine using the *register* primitive in the *ClientCommInterface*.
4. Try to solve the problems that the server provides you upon connection and submit your solution to the server.