Highly Dependable Systems

Sistemas de Elevada Confiabilidade

2016-2017

Dependable Password Manager
Stage 2

Goals

The goal of this part of the project is to extend your implementation from stage 1 to tolerate the possibility that the server side of the infrastructure may suffer from a successful attack.
More precisely, we will keep the same specification for the dependable password manager from the previous stage, but change the dependability guarantees of the implementation of the server by making it Byzantine fault tolerant. To this end, you will need to replace the server with a set of *N* replicas, which collectively implement a Byzantine replication protocol. The size *N* of this set is chosen by you in a way that depends on a value *f*, which is a system parameter representing the maximum number of Byzantine faults, i.e., attacks that can succeed in the system while preserving the correctness of the password manager.

Design and implementation requirements

The basic change to the existing design in this stage of the project is to replace the client-server communication between the library and the server with a replication protocol between the library (acting as a client of that protocol) and a set of server replicas. The specification of that protocol is that the outcome of the reads and writes (i.e., register, put, and get operations) must obey the semantics of a (N,N) Byzantine atomic register. This means that each user can concurrently access the Dependable Password Manager using multiple devices.

Students can assume that all the devices of a given user have a copy of the keystore used to store the cryptographic material of the user.

Implementation hints

To help in your design and implementation task, we suggest that you break up this task into a series of steps, and thoroughly test each step before moving to the next one. Here is a suggested sequence of steps that you can follow:

- Step 1: Replicate the server, without implementing any scheme to synchronize the state of the various replicas. This will involve starting N server replicas instead of one, and replacing the client to server communication with a loop that repeats each communication step N times.

- Step 2: As for the next step, students are advised to implement first the (1,N) Regular Byzantine in Section 4.7 of the course book.
- Step 3: Next, consider a transformation from (1,N) Regular register to a (1,N) Atomic register.
- Step 4: Finally, use a transformation from (1,N) Atomic register to (N,N) Atomic register.
- Step 5: Implement the new set of dependability tests.

Submission

Submission will be done through Fénix. The submission shall include:

- a self-contained zip archive containing the source code of the project and any additional libraries required for its compilation and execution. The archive shall also include a set of tests for the new functionality, and dependability tests showing how the system tolerates or breaks under specific types of faults or attacks. A README file explaining how to run the demos/tests is mandatory.
- a concise report of up to 4,000 characters addressing:
    - explanation of the new features;
    - explanation of the new dependability guarantees that are provided.

The deadline is **April 28, 2017 at 17:00**. More instructions on the submission will be posted in the course page.