# Highly Dependable Systems

# Sistemas de Elevada Confiabilidade

# 2015/16

Project 3: Block server with intrusion-tolerant replication

**Goals**

The goal of this part of the project is to extend your implementation from stage 2 (or alternatively, stage 1, if it is more convenient) in order to tolerate the possibility of the server side of the infrastructure suffering a successful attack.

More precisely, we will keep the same specification for the file system from the previous stage, but change the dependability guarantees of the implementation of the block server that supports the file system, by making the block server Byzantine fault tolerant. To this end, you will need to replace the block server with a set of *N* block 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 file system.

**Design 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 block 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., gets and puts) to both types of objects must obey the semantics of a (1,N) Byzantine regular register, as specified in Section 4.7 of the course book.

As an optimization, you must also adapt the protocol from the book for the specific case of content-hash blocks, to take advantage of the fact that these blocks are immutable and self-verifying. This optimization must allow you to reduce the number of replicas required to store content-hash blocks (i.e., user a smaller N for content hash blocks than the value of N for public key blocks). To understand what are the possible values for N (number of replicas) and Q (size of the quorum) for content-hash blocks, we suggest that you draw a diagram with replicas and different quorums, to understand what are the required intersection properties. You are as usual free to consult with your instructor, who can help you with this optimization.

**Implementation hints**

To simplify your implementation, you are encouraged to turn off the authentication based on the Cartão de Cidadão, and employ instead the simpler primitives based on a

Crypto library used in Stage 1. This is will, in particular, simplify the automation of the tests, since you do not have to insert PINs. If convenient, you can entirely revert back to the Stage 1 code.

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 the algorithm from the book. This will involve starting N server replicas instead of a single one, and replacing the client to server communication with a loop that repeats each communication step N times.

Step 2: Implement the appropriate replication protocol (from the book).

Step 3: Implement the protocol optimizations that reduce the number of replicas required for content-hash blocks.

Step 4: Implement the new set of dependability tests.

**Submission**

Submission will be done through fenix. 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, including dependability tests to demonstrate that the system works correctly despite faults.
- a concise report of up to 4,000 characters addressing:
    - brief outline of the stage 3 design
    - explanation of any modifications to the protocols presented in the book, and the rationale behind those modifications
    - careful explanation of the dependability guarantees provided, namely in which way and under which conditions can the final implementation of the file system not work as desired.

The deadline is May 13 at 17:00. More instructions on the submission will be posted in the course page.