Highly Dependable Systems

Sistemas de Elevada Confiabilidade
2019-2020

Project – Stage 1
Dependable Public Announcement Server

The emergence of fake news and the need for trusted sources of information requires an information system where relevant public information and facts can be posted, tracked and verified. The goal of the project is to design and implement a Dependable Public Announcement Server (DPAS) with the following characteristics:

- Each user of the system has an Announcement Board where only she/he can post announcements
- Once an announcement is posted in the Announcement Board it cannot be removed from the system. Besides, users should be held accountable for the announcements they posted
- Users can read all the announcements of other users and obtain their chronological order
- When posting announcements, users can refer to previous announcements posted by them or by other users. This is fundamental to ensure information provenance
- There is a special “General Board” where all users are allowed to post announcements. Posts in this board should remain accountable
- The system should be resilient to attackers that aim to tamper with the integrity of the announcements

The system must have a client-server architecture where each client and server has a Public key pair. For simplicity, students can assume that there is a Public Key Infrastructure in place, that performs the distribution of keys among all participants before the start of the system. The server should support concurrent clients. Furthermore, in this stage students should assume that there is a single server. This restriction will be lifted in stage 2 by requiring multiple servers.
The client API has the following specification:

- `register(PublicKey key, …)`

Specification: register the user and associated public key in the system before first use. In particular, it should make the necessary initializations to enable the first use of the DPAS.

- `post(PublicKey key, char[255] message, Announcement[] a,…)`

Specification: post an announcement of up to 255 characters to the DPAS. Optionally the announcement can refer to previous announcements.

- `postGeneral(PublicKey key, char[255] message, Announcement[] a,…)`

Specification: post an announcement of up to 255 characters in the General Board. Optionally the announcement can refer to previous announcements.

- `read(PublicKey key, int number,…)`

Specification: obtain the most recent *number* announcements post by the user with associated key. If *number* is set to 0 all announcements should be returned.

- `readGeneral(int number,…)`

Specification: obtain the most recent *number* announcements on the General Board. If *number* is set to 0 all announcements should be returned.

Design requirements

The design of the HDS DPAS system consists of two main parts: a library that is linked with the application and provides the API specified above, and the server that is responsible for keeping the state of the system. The library is a client of the server, and its main responsibility is to translate application calls into requests to the server. Anomalous inputs shall be detected by the server, which should generate appropriate exceptions and return them to the client-side.

In this stage, the following assumptions are done:

- The client library is trusted.
- The server is honest.
- The attacker can drop, reject, manipulate and duplicate messages.

Students will have to analyze the potential threats to the system such as man-in-the-middle or replay attacks, and design application level protection mechanisms to cope with them. There are several approaches to address these issues, so it is up to students to propose a design and justify why it is adequate.

Note that even though the server is honest, it can crash and later recover. The implementation of the HDS DPAS system should guarantee no loss (or corruption) of its internal state in the presence of crash faults. Besides the system should operate under the assumption that the communication

channels are not secured, in particular solutions relying on secure channel technologies such as TLS are not allowed.

Each user is equipped with a Citizen Card ("*Cartão do Cidadão*"), which they should use to satisfy the dependability requirements above.

Implementation requirements

The project must be implemented in Java using the Java Crypto API for the cryptographic functions. We do not prescribe any type of communication technology to interface between the client and the server. In particular, students are free to choose between using sockets, a remote object interface, remote procedure calls, or a SOAP-based web service.

Implementation Steps

To facilitate the design and implementation of the HDS DPAS, students are encouraged to break up the project into a series of steps, and thoroughly test each step before moving to the next one. Having an automated build and testing process (e.g.: JUnit) will help students progress faster. Here is a suggested sequence of steps:

- Step 1: Simple server implementation without dependability and security guarantees. Design, implement, and test the server with a trivial test client with the interface above that ignores the crypto parameters (signatures, public keys, etc.)
- Step 2: Develop the client library and complete the server – Implement the client library and finalize the server supporting the specified crypto operations.
- Step 3: Dependability and security – Extend the system to support the dependability and security guarantees specified above.
- Step 4: Implement the usage of the Citizen Card by the users. For testing purposes, we recommend that the implementation supports operating with or without the Citizen Card.

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 demo applications/tests that demonstrate the mechanisms integrated in the project to tackle security and dependability threats (e.g., detection of attempts to tamper with the

data). A README file explaining how to run the demos/tests is mandatory.

- a concise report of up to 4,000 characters addressing:
    - explanation and justification of the design, including an explicit analysis of the possible threats and corresponding protection mechanisms.
    - explanation of the integrity guarantees provided by the system
    - explanation of other types of dependability guarantees provided

The deadline is **April 3 2020 at 17:00**. More instructions on the submission will be posted in the course page.