INSTITUTO SUPERIOR TÉCNICO

HIGHLY DEPENDABLE SYSTEMS

SISTEMAS DE ELEVADA CONFIABILIDADE

MEIC

Project 1: File server with integrity guarantees
Report

Group: 9

2015/2016

Ricardo Filipe Fonseca Silva
Rui Filipe do Rosário Galão Ribeiro
João Daniel Jorge Machado

1 Introduction

For the first stage of this course's project, we were asked to develop a file server that conformed to a certain set of security guarantees.

In this report, we will go over the design chosen for the FS, as well as the integrity guarantees it offers, plus any other dependability attributes it might offer at the time of delivery.

2 File System Design

We chose to split the File System into two distinct parts.

On one side, we have a client that implements an interface provided by the file system, and uses its carefully crafted methods to read data from any of the available files on the system, and to write data to file belonging to him. While a client is free to read any file, he may only write to his own.

A file, and by proxy, a client, are both identified by a unique ID (the hash of the public key of the client who can write to the file), and is split into separate blocks. All the security information required for the cryptographic operations is stored safely in a Public Key data block, while the actual contents of the file are stored into multiple Content data blocks.

On the other side of the FS, we have a server that handles the more specific methods for writing and reading data blocks. Here is where most of the cryptographic checks are conducted.

We've chosen this separation to make sure that the client only handles the writing and reading of raw data in byte format, while the server handles the more sensitive operations, requiring the validation of signatures and integrity checks.

3 Integrity Guarantees

The file system assures data integrity by restricting access to the files, via the use of a secure interface. A client may only write to the file that belongs to him and that matches his ID, since the interface does not allow him to specify the ID of the file to write to. However, should an attempt be made by a client to write to a file that does not belong to him, then an ID mismatch will be triggered. The data is also signed, using the client's public key, so that its integrity may be checked on the side of the server.

On the server side, the signed data is verified, and only if the verification succeeds is the data then stored by the server. A signature violation is triggered should the validation fail. The data of the file is then split into various content blocks, who are stored on the FS. The client's signature and public key are also stored on a Public Key block, along with the hash information of the content blocks belonging to that client's file, and then also stored on the FS.

While no particularly aggressive encryption mechanisms are used when storing the file contents, it is assumed that no intruder is able to freely access them by bypassing the server's interface. Regardless, should an attacker be able to maliciously alter the data of a file, the contents of said file are checked when a read operation is issued by a client, and an exception will be triggered should the hashes of the content blocks not match the hashes that were stored on the public key block during the write operation.

4 Other Dependability Guarantees

The implemented File Server also provides Safety guarantees. Should any security operation fail, the server will simply deny the request, without compromising the safety of the client, or the client's system.

Assuming a real world scenario where the server would be set up and made available for public use, the FS also assures a high level of availability and reliability in its current implementation. Any operations that violate the security guards will merely be denied, and will not prevent the server to stop receiving requests from other clients.