MIECT: Security 2014-15

Project: SafeBox: Cryptographic File Sharing

Presentation: Nov 21 or 22
Deadline: Nov 10 (1st sub-project), Dec 19 (final project)

Changelog

- v1.0 - Initial Version.

Introduction

The goal of this project is to develop a remote, secure repository (SafeBox) for exchanging files among a set of well-identified people. Anyone can create a personal repository in SafeBox (Personal Box, PBox), where secure files may be deposited by others. A PBox is a metaphor: it is formed by all the files a person was given access to within SafeBox.

1 File protection

Within SafeBox, file protection will consist on content encryption, for preventing its observations by unauthorized persons, and integrity control, for preventing files from being tampered. Furthermore, SafeBox servers are not to be fully trusted, they should be assumed as correct but curious. Therefore, they should not be given, at any time, the means required to have access to the original contents of the protected files.

2 File sharing

Protected files stored by a SafeBox server are always accessible to their creators (the ones that have send them to SafeBox), and these can give access to the files to others (i.e., make them part of others' PBoxes). A PBox owner can share any files placed in his/her PBox with other PBoxes. Each file on each PBox must state who have (i) created or updated its contents and (ii) given access to its contents.

File sharing should reduce to a minimum the redundancy of information. If a protected file is shared by N PBoxes, we should not have N copies of the entire file, but rather N records that provide the required information to have access to the original file contents. For this purpose it is recommended to use hybrid encryption strategies (encryption of file contents with random file keys and encryption of file keys with the public keys of the PBoxes that have access to the related files).

Once a file is shared by N PBoxes, any of their owners can change the file contents and automatically share the modification with the other $N-1$ PBoxes.

3 PBox ownership

A PBox owner must prove its ownership when initiating a session with SafeBox. This ownership proof is to be performed using a password or Citizen's Card, in the beginning of the session, and with a stream of cryptographic authenticators, during the session (an authenticator per request within the session). The initial proof should be managed through a PAM (Pluggable Authentication Modules), which should handle the possibility to use a password or a Citizen Card.

The identity of the owner of a PBox must be bound to an identity provided by a Citizen Card. In other words, each PBox must be identified, for SafeBox users, with the owner's identity attributes provided by his/her Citizen Card (namely, the person's name). For simplicity, aliases or ordinal numbers may be used to refer PBoxes in commands given to SafeBox.

4 SafeBox client-server interaction

SafeBox is a service that should be implemented by a server. The interaction with this server can use any available technology. For instance, the interaction can be supported by independent UDP messages, by data frames transmitted through a TCP stream, by remote object invocations (e.g. using Java RMI), by application-layer requests encapsulated in HTTP requests, by Web Services, etc.

A SafeBox client should be an application, or a set of application, not necessarily with a graphical interface (console applications are fine!), that interacts with SafeBox in order to allow the following high-level operations (at least):

- Create a PBox to a person.
- Initiate or terminate a session with my PBox.
- List the existing PBoxes.

- Add a protected file to my PBox.
- List my PBox files.
- Get the original file contents of a protected file in my PBox.
- List the PBoxes sharing a file of my PBox.
- Share a file in my PBox with other PBoxes.
- Delete a file from my PBox.

Other low-level operations, used by these high-level operations, may be implemented.

5 SafeBox storage

A SafeBox server can use any appropriated technology to manage storage of the protected files and the meta-information required to manage them properly. However, an elegant solution must use a database (e.g. SQLite) to store all meta-information and an ordinary file system to store the protected files.

6 Project execution

This project is to be developed in two phases, or sub-projects, which are to be evaluated independently. The first sub-project concerns everything except the usage of Citizen Cards and the authentication of PBox owners. These two aspects should be considered only in the second sub-project.

In the 6th practical class (week from 20/Oct to 24/Oct) each group should make a slide-based public presentation of their proposed implementation strategy for the first sub-project. About 4-5 slides should be enough for this presentation. Groups may as well reuse strategies proposed by others, but in this case they should give credit to those strategies in a related work section. This presentation will be considered for the final grading of the project.

After this presentation, the groups can keep their initial strategy and improve it, or change radically their strategy in the rest of the project. In any case, these facts should be referred in the final project report, where the initial strategy should be presented, together with the improvements learned upon listening to all the other presentations (or the comments of the professors).

Students are free to use any technology for implementing a prototype of their system.

We strongly recommend students to start the project design and development at once, even without knowing much about cryptography, because the foundations of the client-server interaction are critical and can be developed without such know-how.

Delivery Instructions

You should deliver all code produced and a report before the deadline. That is, 23.59 of the delivery date. The delivery dates are November 10, 2014 (Monday) and December 19, 2014 (Friday).

In order to deliver the project you should create a project in the CodeUA[1] platform. The project should be named after the course name (`security2014`), the practical class name (e.g. `p2`) and the group number in the practical class (e.g. `g5`), with the following complete format for the examples given before: `security2014-p2g5`). Please commit to this format to simplify the evaluation of the projects! Each project should be given access to all the course professors (André Zúquete, João Paulo Barraca and Paulo Salvador Ferreira) for enabling them to download and evaluate them.

Each CodeUA project should have a `git` or `svn` repository, and folders for each sub-project of this project (`proj1` for the first sub-project, `proj2` for the second sub-project). The repository can be used for members of the same group to synchronise work. After the deadline, and unless otherwise requested by students, the content of the repository will be considered for grading.

The report should address all the studies performed, all the decisions taken, all the functionalities implemented and all known problems and deficiencies. Grading will be focused in all these topics, and not only on the code produced!

Using materials, code snippets, or any other content from external sources without proper reference (e.g. Wikipedia, colleagues), will imply that the submission will not be considered for grading.

7 Grading

Grading will take in consideration the capabilities of the software delivered, as well as the elegance of both the design and actual implementation.

8 Bonus Points

Bonus points will be awarded if the solution implemented supports extra features with security interest, conceived by the students. For instance, a

[1] `http://code.ua.pt`

security feature could be an access control mechanism to manage read and write rights given to PBoxes other than the one that creates (and shares) a protected file.