CS 1653: Applied Cryptography and Network Security
Fall 2022

Term Project, Phase 3

**Assigned:** Wed, Oct 12 **Due:** Thu, Nov 03 11:59 PM

---

# 1 Background

At this point in the semester, your group has developed a fully-functional file sharing system; congratulations! This system should currently be capable of storing and retrieving files from a collection of distributed servers, but should *not* yet offer any protection from clients or servers that behave maliciously. In this phase of the project, we will begin to harden your system against a variety of common security threats. Later sections of this assignment detail a threat model that describes the types of assumptions that you are permitted to make regarding the behavior and intentions of each class of principals in the system. A list of *specific* classes of threats for which your system must provide protections is then described in detail.

Your deliverables for this phase of the project will include (i) a writeup describing your proposed protections for each type of threat, as well as a description of why these protections are sufficient, and (ii) a set of modified file sharing applications that implement each of your protections. Your group's grade for this project will be based on the correctness of the protections described in your preliminary writeup, the accuracy with which your implementation embodies these protections, and a live demonstration of your system.

# 2 Trust Model

In this phase of the project, we are going to focus on implementing a subset of the security features that will be required of our trustworthy file sharing service. Prior to describing the specific threats for which you must provide protections, we now characterize the behavior of the four classes of principals that may be present in our system:

- **Group Server** The group server is entirely trustworthy. In this phase of the project, this means that the group server will only issue tokens to *properly authenticated* clients and will properly enforce the constraints on group creation, deletion, and management specified in the previous phase of the project.
- **File Servers** In this phase of the project, you may assume that *properly authenticated* file servers are entirely trustworthy. In particular, you do not need to worry about a properly authenticated file server corrupting files, leaking files to unauthorized users, or stealing user tokens.

- **Clients** We will assume that clients are not trustworthy. Specifically, clients may attempt to obtain tokens that belong to other users and/or modify the tokens issued to them by the group server to acquire additional permissions.

- **Other Principals** You should assume that *all* communications in the system are monitored by a *passive adversary*. A passive adversary is able to watch all communication channels in an attempt to learn information, but cannot alter or disrupt any communications in the system.

## 3 Threats to Protect Against

Given the above trust model, we must now consider certain classes of threats that were not addressed in the last phase of the project. In particular, your group must develop defenses against the following classes of threats in this phase of the project:

**T1 Unauthorized Token Issuance** Due to the fact that clients are untrusted, we must protect against the threat of illegitimate clients requesting tokens from the group server. Your implementation must ensure that all clients are authenticated in a secure manner prior to issuing them tokens. That is, we want to ensure that Alice cannot request and receive Bob's security token from the group server.

**T2 Token Modification/Forgery** Users are expected to attempt to modify their tokens to increase their access rights, and to attempt to create forged tokens. Your implementation of the `UserToken` interface must be extended to allow file servers (or anyone else) to determine whether a token is in fact valid. Specifically, it must be possible for a third-party to verify that a token was in fact issued by a trusted group server and was *not* modified after issuance.

**T3 Unauthorized File Servers** The above trust model assumes that properly authenticated file servers are guaranteed to behave as expected. In order for this guarantee to mean anything, your implementation must ensure that if a user attempts to contact some server, $s$, then they actually connect to $s$ and not some other server $s'$.

Note that any user may run a file server. As such, the group server *can not* be required to know about all file servers. Your mechanism for enabling users to authenticate file servers should require communication between only the user and the file server, and possibly client-side application configuration changes. **Hint:** You may wish to look into how SSH allows users to authenticate servers.

**T4: Information Leakage via Passive Monitoring** Since our trust model assumes the existence of passive attackers (e.g., nosy administrators), you must ensure that all communications between your client and server applications are hidden from outside observers. This will ensure that file contents remain private, and that tokens cannot be stolen in transit.

# 4 What Do I Need to Do?

This phase of the project has two deliverables. The first deliverable is a semi-formal writeup describing the protection mechanisms that your group proposes to implement, and the second is your actual implementation. We now describe both aspects of the project in greater detail.

## 4.1 Mechanism Description

The first deliverable for this phase of the project will be a writeup (3–5 pages) describing the cryptographic mechanisms and protocols that you will implement to address each of the threats identified in Section 3 of this assignment. This writeup should begin with an introductory paragraph or two that broadly surveys the types of cryptographic techniques that your group has decided to use to address threats T1–T4. You should then have one section for each threat, with each section containing the following information:

- Begin by describing the threat treated in this section. This may include describing examples of the threat being exploited by an adversary, a short discussion of why this threat is problematic and needs to be addressed, and/or diagrams showing how the threat might manifest in your group's current (insecure) implementation.
- Next, provide a short description of the mechanism that you chose to implement to protect against this threat. For interactive protocols, include diagrams explaining the messages exchanged between participating principals. (We have seen a few examples of protocol diagrams in class, and will see many more in Lecture 11.) Be sure to explain any cryptographic choices that your group makes: What types of algorithms, modes of operation, and/or key lengths did you choose? **Why?** If shared keys are needed, how are they exchanged? Recall that security is not absolute nor are any tools appropriate for all situations; every component of your design should be intentional and justified relative to the given threat model.
- Lastly, provide a short argument addressing why your proposed mechanism sufficiently addresses this particular threat. This argument should address the correctness of your approach, as well as its overall security. For example, if your mechanism involves a key agreement or key exchange protocol, you should argue that both parties agree on the same key (correctness) and that no other party can figure out the key (security). You do not need a formal proof, but you should convince me that an attacker can no longer exploit each threat.

After completing one section for each threat, conclude with a paragraph or two discussing the interplay between your proposed mechanisms, and commenting on the design process that your group followed, including any extra credit that you did. Did you discuss other ideas that didn't pan out before settling on the above-documented approach? Did you end up designing a really interesting protocol suite that addresses multiple threats at once? Use this space to show off your hard work!

**Discussion of your design:** The types of security mechanisms that you will develop during this phase of the project are notoriously finicky, and easy to build incorrectly. The goal of this writeup is to focus your group on properly designing and evaluating your mechanisms *before* you spend time implementing them. A well-executed writeup will give your group a concrete reference point to discuss, debate, analyze, and (finally!) implement. To encourage such discussion, *10% of your grade for this phase of the project is based upon approval of your writeup in a meeting with the instructor before the deadline* (the earlier, the better!). I will arrange dedicated office hours slots for this task soon. This may include "flipping" lectures to asynchronous videos and using our lecture time to discuss progress on the project. As always, watch Discord for announcements.

Your discussions can also occur during regular office hours. Have your writeup pushed to GitHub **in advance** of this meeting to facilitate the conversation. I will not discuss a scheme that is described only out loud or that lacks the details necessary to evaluate it. Note that, if there are flaws in your approaches, we will not design new approaches in our meeting. This means you may need to meet multiple times, so plan ahead!

### 4.2 Implementation Requirements

In order to properly address the Unauthorized Token Issuance threat (T1) described above, you will need to modify the `getToken` method described in `GroupClientInterface.java`. Since every group may choose to address this threat in a different way, we will not specify a new method signature—feel free to modify this method however is needed. We strongly recommend that you leverage the expertise developed in Homework HW1 and use the BouncyCastle cryptography API to incorporate any cryptographic functionality that you may need. As before, this project will be graded using the CS Linux Cluster. You are responsible for ensuring that your code runs correctly on these machines well before the due date.

## 5 Extra Credit

As in the last phase of the project, you have the opportunity to earn up to 8% extra credit. Should you happen to complete the required portions of the project early, consider adding in extra functionality in exchange for a few extra points (and a more interesting project). Any extra features that you add may qualify, so brainstorm as a group and see what you come up with! If you opt to do any extra credit, include a brief description of it in the discussion section of your writeup.

(Unless a previously-implemented extra credit feature requires substantial updates within the new threat model, you cannot use the same feature for extra credit in multiple phases.)

## 6 What (and how) do I submit?

Your grade for this project will be based upon your technical writeup (50%), approval of your design with the instructor (10%), a demonstration and assessment of the code that your team produces (35%), and scheduling a demo with the TA prior to the deadline (5%).

An initial writeup skeleton is available to you via the following GitHub repository:

`https://github.com/2231-cs1653/cs1653-project-phase3-writeup`

You should copy the provided HTML file into the documentation directory of your main project repository without changing its name (`doc/phase3-writeup.htm`). Modify this file only within the denoted areas.

Within your project repository (your existing `cs1653-project-*` repository from phase 2), you should include the following files and directories.

- `src/` In this directory, include all of your source code that is needed to compile your project. You may create subdirectories (packages) within this folder if you'd like to better organize the code. Please do not commit any JAR or class files, as we will be rebuilding your code before your demo (and it is common version-control etiquette not to commit files that can be re-derived from the included source). Also, please do not commit any publicly available libraries (e.g., Apache Commons, BouncyCastle) that you make use of—include instructions for acquiring those libraries in `doc/compile.txt` (see below).

  Note that we will be grading your assignment using the CS Linux Cluster as with Phase 2, so you must test your code in that environment prior to the submission deadline!

- `doc/` In this directory, include all documentation for your project, including *at least* the files named below.

  - `doc/compile.txt` In this text file, please write detailed instructions for compiling your code, including direct links to any publicly available libraries that you used in your project. If you opt to use a Makefile or other script to build your code, include instructions here on how to run your build script.
  - `doc/usage.txt` In this text file, include explicit instructions on how to use your system. In particular, this should explain how to start your group server and file server, as well as how to start your client application(s). For each operation supported by your client application(s), provide a short statement of how to invoke this operation.
  - `doc/phase3-writeup.htm` Copy the provided HTML writeup skeleton and edit it where noted, as discussed above.

As mentioned above, your repository's commit log will serve (in part) to ensure each individual is contributing to the group project. In addition, *each student in your group* should send an email to `bill@cs.pitt.edu` that indicates their assessment of each group member's contribution to this phase of the project.

Your project is due at the precise date and time stated above. We will clone your repository immediately after the due date, so you will be graded on whatever changes have been committed **and pushed** to your repository's main branch by this time. No changes made after this point will be considered in your demo or in grading your project. Make sure you understand the submission process well in advance!