CS 1653: Applied Cryptography and Network Security
Spring 2016

Term Project, Phase 3

Assigned: Tuesday, October 3 **Due:** Tuesday, October 31 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 brief 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 short 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, it is highly recommended to include diagrams explaining the messages exchanged between participating principals. (See the notes from Lecture 11 for example diagrams.) 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?
- 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).

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!

Important Note: 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 a discussion of your writeup in-person with both the TA and Professor Lee. To receive full credit, Veronica must pre-approve your writeup by 10/20, and Professor Lee must approve your writeup no later than 10/27. Keep in mind that multiple discussions with each of us may be needed!.*

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 machines in Sennott Square 6110. Please ensure that your code runs correctly under Linux on these machines before the due date.

5 Extra Credit

As in the last phase of the project, you again have the opportunity to earn up to 5% 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, be sure to include a brief description of it in the discussion section of your writeup.

6 What (and how) do I submit?

Your grade for this project will be based upon your technical writeup (45%), your discussion of this writeup with the instructor (10%), a demonstration and assessment of the code that your team produces (35%), scheduling a demo with the TA prior to submission (5%), and properly submitting your assignment (5%).

Your project will be submitted via GitHub, using your existing group repository. Please ensure that this is (still) a private repository, and is (still) shared with Prof. Lee (GitHub ID: `ajl-pitt`) and the TA (GitHub ID: `potato84`). Within your existing project repository, 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. Please do not commit any JAR or class files, as we will be rebuilding your code before we evaluate it (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.md` (see below).
- `doc/` In this directory, include all documentation for your project, including *at least* the files named below.

- `doc/compile.md` In this Markdown 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.md` In this Markdown 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/p3_extra_credit.md` *(optional)* If you have gone beyond the requirements of the project as described in Section 5, please explain what you have done in this Markdown file.

- `reports/phase3-writeup.md` This will be where the Markdown version of the report described above will go. Veronica and Professor Lee will read directly from your repository during your team's discussions regarding your proposed mechanisms for this phase of the project.

Preparing for submission. When your repository is ready for submission, commit and push your repository back to GitHub. Then, create a git *tag* called `phase_3`:

```
git tag -a phase_3 -m "Phase 3 submission"
```

Then, push this tag to GitHub:

```
git push origin phase_3
```

In git, a tag is simply human-readable shorthand for an (otherwise unreadable) commit hash. The first command, above, links your commit hash to the string `phase_3`, while the second command syncs that tag to GitHub so that others can use it. After this tag is available, anyone with access to your repository can examine your Phase 3 submission by typing `git checkout phase_3` from inside a local clone of your repository.

Submission. Your project is due at 11:59 PM on Tuesday, October 31. To timestamp your submission, run the command `git log -n 1` and email the TA (veronica@cs.pitt.edu) the commit hash for your submission **prior to the due date**. After cloning your repository, we will grade the version of your code that corresponds to this commit hash.

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 `adamlee@cs.pitt.edu` that indicates his or her assessment of each group member's contribution to this phase of the project (e.g., *Bob did 40% of the work, and Mary did 60% of the work*).