CS 1653: Applied Cryptography and Network Security

Fall 2023

Term Project, Phase 4

Assigned: Mon, Nov 27 **Due:** Sun, Dec 10 11:59 PM

---

1 Background

In this phase of the project, you will further extend your distributed systems to protect against a few more classes of security vulnerabilities. You will be provided with a threat model describing the types of assumptions that you will be able to make regarding the principals in the system, and will be given a list of specific classes of threats for which your system must provide protections.

Your deliverables for this phase of the project will again 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 applications that implement each of your protections. As before, your group's grade 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 another set of the *security features* required of our trustworthy distributed system. 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:

- **Authentication Server** The authentication server is entirely trustworthy. In this phase of the project, this means that the authentication server will only issue tokens to *properly authenticated* clients and will properly enforce the constraints on group creation, deletion, and management specified in previous phases of the project. The authentication server is *not* assumed to share secrets with the resource servers in the system.
- **Resource Servers** In this phase of the project, resource servers will be assumed to be largely untrusted. In particular, resource servers might leak private resources to unauthorized users or attempt to steal 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 authentication server to acquire additional permissions.

- **Other Principals** You should assume that *all* communications in the system might be intercepted by a *active attacker* that can insert, reorder, replay, or modify messages.

3 Threats to Protect Against

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

T5 Message Reorder, Replay, or Modification After connecting to a properly authenticated resource server or authentication server, the messages sent between the user and the server might be reordered, saved for later replay, or otherwise modified by an active attacker. You must provide users and servers with a means of detecting message tampering, reordering, or replay. Upon detecting one of these exceptional conditions, it is permissible to terminate the client/server connection.

T6 Private Resource Leakage Since resource servers are untrusted, resources may be leaked from the server to unauthorized principals. You must develop a mechanism for ensuring that resources leaked from the server are only readable by users with the necessary access. As in previous phases of the project, we stress that the authentication server cannot be expected to know about all resource servers to which its users may wish to connect. Further, your proposed mechanism *must* ensure that some level of security is maintained as accesses change (i.e., are granted and revoked).

T7 Token Theft A resource server may "steal" the token used by one of its clients and attempt to pass it off to another user. You must develop a mechanism for ensuring that any stolen tokens are usable only on the server at which the theft took place (and are thus effectively useless, as this rogue resource server could simply allow access without checking the token).

Note that you must also address *all* threats from the previous phase of the project. That is, your new functionality should not invalidate previous requirements.

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.

4.1 Mechanism Description

As in Phase 3, the first deliverable for this phase of the project will be a writeup (approximately 3–5 page) 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 T5–T7.

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 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. Be sure to explain any cryptographic choices that your group makes: What types of algorithms, modes of operation, and/or key lengths did you chose? **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. 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!

Finally, spend about one paragraph convincing me that your modified protocols still address the threats T1–T4 described in Phase 3 of the project. Full credit for Phase 4 requires that all Phase 3 threats are still protected against.

Approval of your design: As in the last phase of the project, **10% of your grade for this phase of the project is based upon obtaining the instructor's approval of your design by Dec 04** (the earlier, the better!). You can attach your draft writeup to the `#p4` Discord channel for instructor review. This channel is also a good place to discuss other groups' designs.

Please note that your instructor will not attempt to evaluate a scheme that is described only informally or that lacks the necessary details. If there are flaws in your approaches, you will need to adjust your design and resubmit. This means you may need to submit multiple times, so plan ahead!

4.2 Implementation Requirements

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.

You can (and should) use library functions that implement cryptographic primitives, but you should not use any prebuilt protocols or other complex constructions. In particular, avoid external implementations of SSL/TLS/HTTPS, SSH, Kerberos, OTR, IPsec, OAuth, etc. You may use block ciphers, stream ciphers, public key encryption/signing, hash functions, MACs, Diffie-Hellman, secure pseudorandom number generators, and key derivation functions (such as bcrypt). If you're unsure about whether you can use a particular library function, ask on `#p4` on Discord.

If your design includes any out-of-band key exchange, be sure to design your system to make this feasible, and justify why it would be realistic in practice (or what should happen instead, if relevant). To ensure a smooth demo, you also need to plan how keys will be exchanged when demonstrating your functionality to your TA. Test run each server and client from a different folder, and make sure that key exchange is being done intentionally and consistently.

5 What (and how) do I submit?

Your grade for this project will be based upon your technical writeup (50%), approval of your design by the instructor on Discord (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%).

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

- `desc/` In this directory, we provided this project description. No changes are necessary.
- `doc/` In this directory, include all documentation for your project. Include an updated user's manual and technician's guide (separately or combined) for your system based on what you developed in Phase 2. All documentation should be in PDF or HTML format.

  In addition, include `doc/phase4-writeup.htm` or `doc/phase4-writeup.pdf`, your writeup that explains your mechanism design as described in Section 4.1.
- `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 compiled code (e.g., JAR or class files). 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 that you make use of—include instructions (or, even better, a script) for acquiring those libraries.

> Note that we will be evaluating your submission using the CS Linux Cluster as with previous phases, so you must test your code in that environment prior to the submission deadline!

Each individual's contribution to the group's work will be judged in part by the version control logs. In addition, *each student in your group* will be asked to evaluate the group's performance, including how well everyone is working together and supporting one another.

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 your repository is created and you understanding the submission process well in advance!