Kryptose: Sprint Report Alpha

Personnel: Jonathan Shi (js2845), Antonio Marcedone (am2623), Alexander Guziel (asg252), Jeff Tian (yt336)

Activity Breakdown

Jonathan:

I wrote the thread-based architecture for the server (after a bit of fumbling around partially writing an event-based architecture that turned out to be a bad idea), as well as handling how everything is written to file. I researched proper security practices around using Java serialization of objects and implemented the validation of deserialized objects. I implemented non-user-specific logging and exception handling. I'm gonna guess 40 hours. These activities were a good use of time, since they mostly contributed directly to the functionality of our project or the security guarantees.

Antonio:

My main task in the first sprint was to choose appropriate cryptographic primitives for the various tasks, and to make an interface for the java cryptographic libraries that the rest of the team could use transparently. This involved simulating a fake certification authority and issuing a certificate for the server, designing and setting up methods for communication between the client and the server (we used TLS1.2 with specific ciphersuites), and encryption of the data structure containing the credentials of the user. I also implemented xml files to store configuration settings for both the client and the server that are read each time the applications are started.

This is my first Java project, so I started before everyone else by reading a Java book to understand the basics of the language (this part is not counted in the estimate below). The JSSE libraries (and certificate generation) were not very straightforward to use for me, I estimate around 45/50 hours overall. I think it was a good use of time, considering it is my first java application. I will probably be more efficient in the future.

Alexander:

My first task was to decide on the response and request types which are the objects used for communication between client and server. I implemented the handling of requests server side and what responses to send back. This also logs the user's actions.

I also worked on handling of responses in the client REPL to handle all responses, even unexpected ones, and improved the client so unexpected input would be handled gracefully rather than with an exception. I made the client logic match up with the server logic. I also fixed various bugs in the client and server.

I probably spent about 30 hours coding. These activities were a good use of time since a lot of bugs were fixed and things were made rigorous by working across both client and server logic. Working with the client also helping me learn about Model-View-Controller framework which took some time but was helpful personally.

Jeff:

I wrote the client side read-eval-print-loop for handling user input, which involved setting up the model view controller framework on the client side and writing all of the logic that allows it to translate client commands to interactions with the server, and translating back the server responses in a meaningful way for the user to access their passwords.

More specifically, this involved setting up the flow sequence for the steps the user has to take upon starting up the app, which involves logging in, retrieving the passwords from the server, and waiting for user to enter a command. I invented the different commands the user could make, and implemented the code for them within the context of the model view and controller, including retrieving passwords from the server, pushing them to the server, creating a new password, modifying an old password, deleting a password, and displaying passwords to the user, and logging out. The implementation of these commands also involved the setting up of internal structures and classes to store and access associated data such as username, derived from master password, and the set of stored passwords in the password file, and also serialization procedure of items that needed to be sent to the server.

I spent around 45 hours total on coding. I completed all of the items delegated to me, and successfully debugged the errors within a reasonable amount of time, nothing stumped me too hard. It took roughly the amount of time I expected it to, only that I made a few more iterations than necessary before finalizing the Model-View-Controller framework, in future sprints I might do a little more code structure planning than I did this time.

Productivity Analysis

We finished all of our planned functionality, except for maybe the ability for a user to inspect logs related to their account. We also have a bit more functionality than we planned, in that we handle multiple users. We spent more time than expected with debugging and integration, and we've learned that meetings is probably not the right time to do our first integration tests. Writing the design document took more time than expected because we don't know how to write I guess. Nothing much took less time than expected.

Our meetings are less productive than ideal because we get sidetracked by minor issues and team debugging. We can fix this by having integration testing before meetings.

We’ve been unreliable in attending scheduled meetings. Several of our project team members do not wake up. We will move the scheduled meeting time to the afternoon.