COSI 107a Lab 6: Implementing TinySSL

Revision 1

COSI 107a

April 2023

1 TinySSL Assignment

Using the TinySSL specification, you will write a client implementation of TinySSL to be turned in using a Programming Assignment on Gradescope. The exact format and assignment for Gradescope will be posted soon; there is enough here for you to get started on your implementation.

Your implementation will be in Python. You will need to implement methods that can send each of the client messages in the protocol, process the responses from the server, and then exchange "application data" over your encrypted communications channel.

2 Supporting code and documents

Some helpful code and documents are in the `lab-06` subdirectory of the class git repository `https://github.com/treese/cosi-107.git`.

At the moment, the following directories are present:

`docs` Contains the TinySSL protocol specification and this assignment file in PDF form. The source form of the documents, in Emacs Org mode (a text markup format similar to Markdown) is included as well.

`sample-code` Contains sample code fragments that may be helpful to your implementation. For now, it has demo code on how to simple networking in Python to write client and server programs. The assignment focus is on writing the client, but in doing that it is often helpful to write your own server code for both testing your code and understanding the protocol better. You can adapt the parts you need into your own implementation. Additional code example for things like how to use the Python cryptography libraries will be added here soon.

`tests` Contains some test simple code in Python `unittest` form. The tests are intended to help you check your code to make sure your implementation is on the right track. They are not exhaustive in testing your code. You might want to use the `unittest` approach to write your own test code for your methods.

3 Implementation approach

To implement a new network protocol, it helps to go in small steps so that you are not trying to debug a dozen different possible bugs. As noted, you might find it helpful to implement the server side yourself as you go, but that is not necessary for the assignment.

One approach would be like this:

1. Read over the protocol to get a general idea of what's going on, and sketch out how you want to approach each part.
2. Implement and test the clear-text record-layer protocol, using the simple test suite as a starting point for testing. By then, the test server may be running for you to test against as well.
3. Implement the "security escapes" in the record-layer protocol for debugging.
4. Add the `CLIENT-HELLO` message, which goes in plain text.
5. Add processing the server response to `CLIENT-HELLO`, which includes encrypted data (details on using the encryption library to come)
6. Continue to work through the rest of the protocol.

4 Details of the simple test suite

The existing TinySSL tests assume a class `TinySSLClient` in the file `tinyssl_client` in the `lab-06` directory. The current tests assume a class constructor with no required arguments. The implemented tests are for methods specified as follows:

`make_record(data)` Returns a TinySSL record-layer message as a byte array with a header. The test calls it with `data` and checks to make sure the returned message is correctly formatted.

`decode_record(msgdata)` Given an input record-layer message, extracts and returns the application-level data contained in `msgdata`.

Your final program does not have to implement the interfaces in this way. You can also adapt the tests to a different set of methods as well.

Additional methods may be added here as we go. Also, you will get a chance to test against a running server, available soon.