Applied Cryptography – Project assignment
October 18, 2017

The goal of the semester project is to use the cryptographic building blocks that we learned during the lectures in practice. We believe that a learning-by-doing approach leads to deeper understanding of the subject.

To make it fun and somewhat less of an effort, the semester project can be carried out in groups of 2-3 students. This also allows students to gain experience in team work and collaborative development of software. So, please, team up with your buddies, and be prepared for the challenge!

The theme of the semester project is the development of a secure, multi-party, on-line chat application. The project has two phases: (1) in the first phase, teams have to come up with a design and submit a sufficiently detailed description of their envisioned cryptographic protocols for verification by the professors, and (2) in the second phase, after receiving some feedback on the design, teams have to implement their (corrected) designs using a real crypto library.

We prepared and made available a sample chat application (source code in Python for the chat server and chat clients), which will implement the basic communication primitives (sending and receiving chat messages) and some management functions (logging into the chat server, creating a new conversation, listing ongoing conversations, and joining an active conversation). The sample application will be fully functional, but it won't contain any cryptographic protection of chat messages. The task will be to extend this application with security features (setting up cryptographic keys among the chat participants, and applying encryption, integrity protection, and sequence numbering mechanisms on chat messages). Extending the sample application allows teams to focus on the crypto part, rather than spending time on the chat primitives themsleves. Nevertheless, if you want to start from scratch (e.g., because you want to implement your project in another programming language, or for any other reason), then we will not stop you doing that!

In the first (design) phase of the project, you do not really need the sample application. You can think of it as a standard client-server application, and you can assume that primitives are available for managing conversations and sending textual messages. It is important, however, that all communications should be mediated by the server, so chat clients cannot talk directly to each other. Teams should design protocols for setting up keys among the chat participants, such that the server should not have access to those keys (i.e., we assume that the server is an "honest but curious" party, which we trust for managing user accounts and chat conversations, but we don't trust it for accessing the content of those conversations - pretty much like the trust we should have in a cloud service provider in real life). Teams should also design protection mechanisms for chat messages to prevent eavesdropping and to detect replay and modification of messages, as well as injection of fake messages by the server. Designs should follow the design principles that we have learned in the classroom and in homework. Teams should produce a document that contains the description of their design in sufficient details, and submit that document **by November 6, 2017 (midnight)**.

In the second (implementation) phase, teams will implement their design (after feedback from professors and potential corrections). Those relying on the sample chat application provided should use the PyCrypto crypto library. The sample application implements an API to send and recieve strings to and from the server within the context of an ongoing conversation managed by the server. The server will forward a received string to all other participants of the conversation. Your implementation should use these primitives to carry your protocol messages. This means that your

protocol messages should be encoded by the sender chat participant as a string, and parsed and processed by the receiving chat participants. Depending on the message type, receiving chat participants should then either respond with an appropriate protocol message, or display the decoded chat messages to the user. Implementations should keep track of protocol states and handle errors. Teams should complete and submit their work (source files in a zip file) **by December 6, 2017 (midnight)**. All teams will run a demo in the classroom on December 11, 2017. Professors will look into the source code to check if teams carefully implemented their designs.

Design document outline

- Overview of system architecture (client-server setup, managing chat rooms, envisioned operation of the chat service as a whole)
- Attacker model (assumptions made about external attackers and the server as a dishonest insider)
- Security requirements identified based on the attacker model
- Description of the key establsihment protocol for chat participants (message sequence figures, message format specification, description of how messages are processed and protocol states are handled)
- Description of the secure channel protocol for chat rooms (message formats, message processing rules, handling protocol states, cryptographic primitives used and their parameter sizes, key derivation)
- Optional: Plan to integrate your design into the sample chat application provided

Hints for the design phase

- Do not over-complicate things; try to make your protocols as simple as you can, because complex protocols are more likely to have flaws and they are more difficult to implement.
- Try to use cryptographic primitives that are supported by the crypto library that you will use in your implementation. For instance, the Diffie-Hellman protocol is not supported in PyCrypto, and it is not trivial to implement it correctly with the existing primitives.
- You can re-use stuff that you have already implemented in homework assignments (e.g., CCM mode or other secure channel implementations).
- Use Piazza for discussion with other groups, and in particular to share technical difficulties that you encounter; others may have already solved the very same problem, and can help you!

Project presentation outline

- Brief overview of your design
- Brief overview of your impementation, and some key features that you are proud of
- Demo (including normal operation and some potential attacks)
- Main difficulties that you encountered during the project
- Main lessons that you learned, and some advice you would give to students of the next semester