CS 6349- Network Security – Fall 2017 Programming Project (Initial Draft)

In this project, you will design and implement a secure Internet file transfer application/protocol. Your application will include a client and a server. Your program will include several security requirements as outlined below. We will evaluate your programs based on their conformance to these requirements.

Supported Functionality

- Client should be able to upload files to the server in a secure fashion. Client also downloads files from server in a secure fashion.
- When the file is uploaded, or downloaded, it should be intact, i.e. it should retain its features. For instance, if it is executable, it should be able to run, or if it is an image, the image must be same as the original file.
- Client only needs to authenticate the server. The server need not authenticate client.
- The only feature that is allowed for securing communication is keyed hash, e.g. SHA-1.

Security Requirements

The application will include the following security requirements:

1. **Authentication**: Client authenticates the server using server's certificate signed by the CA. The TA will act as a CA and will create a certificate for the server and give out the server certificate and CA's public key to you to use in your project (will be done soon).
2. **Confidentiality**: The messages exchanged between client and server will be protected from exposure to others that are not authorized to read what is being communicated. The only security primitive that is available to build a confidential communication mechanism is a keyed hash, e.g. SHA-1. Part of the project is designing a communication protocol that is secure against well-known attacks on confidentiality.
3. **Integrity**: The possible message alteration in transit should not go undetected by the communicating parties. Again, you can only use a keyed hash mechanism to achieve this.

Project Details

- This is a two-person team project. Please let us know if you have difficulties in finding a partner for your project.
- You will need to complete a project design report and discuss it with the TA. The design report should be submitted by October 6, 2017, but earlier is the better so that you can meet the TA and get feedback about your design. The design report will include a summary of the functional capabilities, security features, threat model, and attacks considered when designing the system.

- At the end of the semester, in addition to your program, you will submit a revised project report discussing the above components as they are finally included in the resulting application.

Technical Details

- For simplicity, you can assume that all the file transfers happen in the current directory of the server. That is there is no need to implement file system structure and directory commands (e.g. cd, ls, …)
- You will need to use multiple keys to provide both integrity and confidentiality of messages; they can be derived from a session key using a known pattern.
- You are not supposed to use any secure transport layer service implementations. The only cryptographic function that is available to you is SHA-1. However, for authenticating server using a certificate, you may use relevant crypto libraries (e.g. to verify the certificate).
- You are supposed to build the secure transport layer services by yourselves at the application layer. If you find a library that guarantees authentication and privacy, and gives you an API to call it, then you are essentially using an existing transport layer service and this is NOT acceptable. Again, verifying server certificate is an exception to this.

What to submit

- **First Milestone**. A design document describing your protocol implementation should be submitted by October 6, 2017. It should in general contain how your implementation: 1) performs server authentication using certificate, 2) allows client and server to establish a session 3) provides integrity and confidentiality in file transfer and any other assumptions you have made.
- **Last Milestone**. Source code including *makefiles* and a readme file detailing how to execute your program. Revised (final) project report.