ASSIGNMENT III

Date of evaluation: 16.03.2015, 1.00 pm – 3.0 pm

Build your own simplified version of SSL, called *mySSL*. You can use one of C, C++, or Java for this assignment. Use client server sockets to create a TCP connection. Your client - server programs must do the following:

Handshake Phase (use the SSL handshake)

• The client and the server authenticate each other using certificates. You need to create the certificates and include them in the mySSL messages.

• The client also informs the server what data encryption and integrity protection scheme to use (there is no negotiation). Pick your favorite integrity protection and encryption algorithms.

• The client and server also send encrypted nonces to each other. These nonces are then *XORed* to create a master secret.

• Compute a hash of all messages exchanged at both the client and server and exchange these hashes. Use keyed SHA-1 for computing the hash. The client appends the string CLIENT for computing its keyed hash and the server appends the string SERVER for computing its keyed hash. Verify the keyed hashes at the client and the server.

• Generate four keys (two each for encryption, authentication, in each direction of the communication between the client and the server) using this master secret. Pick your own key generation function (should be a function of the master secret).

Data Phase

• Use the SSL record format and securely transfer a file, at least 50 Kbytes long, from the server to client.

• Decrypt the file at the client and do a *diff* of the original and the decrypted file to ensure that the secure file transfer was successful.

Use *openssl* or any other security library of your choice in any form convenient to you to generate certificates and to extract public keys from certificates and also for keyed hash computation, encryption, and data integrity protection.

Include print commands in your code to show:

1. a failed verification of keyed hashes (possibly due to corruption or changes in one of the handshake messages), and 2. a successful client-server mutual authentication, key establishment, and secure data transfer.