CMPSC443 - Fall 2018
Assignment 2 - Needham-Schroeder Variant Protocol Specification
Professor McDaniel - Due November 9th, 2018

1 Overview

In this assignment you will implement the client side of a key distribution protocol which is an adaptation of the Needham-Schroeder protocol discussed in class. This will require you to implement the network protocol, parse packets, build a protocol state machine and use encryption libraries.

2 Protocol Definition

The protocol consists of three parties, Alice (A), Bob (B) and the Key Distribution Center Server (S). You are to initiate a network connection to the server and execute the protocol as described below. Note that structures and definitions for all of the fields and structures are defined in `cmpsc443_ns_proto.h`.

N_i	Nonce (random number, 64 bit)
A	Identity of Alice (16 byte, zero filled)
B	Identity of Bob (16 byte, zero filled)
K_A, K_B	Keys of Alice and bob respectively
K_{AB}	Session key between Alice and Bob
D	Data sent between Alice and Bob

Msg #	Message type	Direction	Message contents
1	Ticket request	$A \rightarrow S$	N_1, A, B
2	Ticket response	$S \rightarrow A$	$\{N_1, B, K_{AB}, \{K_{AB}, A\}_{K_B}\}_{K_A}$
3	Service request	$A \rightarrow B$	$A, B, \{K_{AB}, A\}_{K_B}, \{N_2\}_{K_{AB}}$
4	Service response	$B \rightarrow A$	$\{N_2 - 1, N_3\}_{K_{AB}}$
5	Service acknowledgement	$A \rightarrow B$	$\{N_3 - 1\}_{K_{AB}}$
6	Data request	$B \rightarrow A$	$\{D\}_{K_AB}$
7	Data response	$A \rightarrow B$	$\{D \oplus$ 1011 0110 $\}_{K_AB}$
8	Service finish[1]	$B \rightarrow A$	

Other information

1. The entire protocol is performed over a single connection. That means that you only need to create a single socket connection and send messages over to to both S and B. The server port is defined as `NS_SERVER_PROTOCOL_PORT` in `cmpsc443_ns_proto.h`.

2. Each message contains a fixed header of four bytes followed by a variable length payload. The first two bytes are the length of the payload in network byte order (number of bytes). The second two bytes are the message type in network byte order as defined int he `message_type_tv` type in the `cmpsc443_ns_proto.h` file.

3. We are using 128-BIT AES encryption. You will use the encryption functions defined in the `gcrypt.h` file. The gcrypt library is documented in:

   https://www.gnupg.org/software/libgcrypt/index.html

4. Each encrypted block within a message is preceded by 128-bit block of random text (known as a initialization vector) and two byte length (in network byte order). You should the following encryption parameters (see documentation for details): `GCRY_CIPHER_AES, GCRY_CIPHER_MODE_CBC`

5. Alice and Bob identity and passwords are hard coded in the cmpsc443_ns_proto.h include file. They are `NS_ALICE_IDENTITY`, `NS_BOB_IDENTITY`, `NS_ALICE_PASSWORD` and `NS_BOB_PASSWORD`, respectively.

6. You are given several helper functions. Two that are absolutely necessary are `createNonce()` function (which createa a N_x value) and `makeKeyFromPassword()` function which creates an AES key from a password. See the `cmpsc443_ns_util.h` and `cmpsc443_ns_util.c` file for documentation how to use them.

7. There a number of other utility functions that are provided to you that will significantly simplify your code. They are listed in `cmpsc311_log.h`, cmpsc311_network.h and `cmpsc311_util.h`. See documentation in the files for details.

8. One particular utility `logMessage()`, is going to be very useful. Basically this is like a printf function but creates logging that can be controlled in various ways. To use it, just call

   ```
   logMessage(<level>, <pattern>, <parameters>)
   ```

   where, level is `LOG_INFO_LEVEL` (indicating that this informational), pattern is a printf style patter, and paramters are the parameter to the printf. For example, if you wanted to print out a character string name NAME and unsigned int VALUE, you would call

   ```
   logMessage(LOG_INFO_LEVEL, ``Name:  %s, value=%u'', NAME, VALUE)
   ```

   Also see the function `logBufferMessage()` which prints out the hexidecmal byte values of the buffer.