CS6500 - Network Security
Assignment 2: KDC-based key establishment

Instructor: Krishna M. Sivalingam
Assigned on: *Feb. 10, 2020*
Due: *Feb. 23, 2020, 11PM*; 15% Penalty – *Feb. 24, 2020, 11PM*; 30% Penalty – *Feb. 25, 2020, 11PM*
The assignment can be done individually or as a team of at most 2 members.
One student can belong to only one team
Updated on: February 19, 2020

The objective of this assignment is to implement the KDC-based key establishment (exchange) for use with symmetric key encryption algorithms.

There will be two programs: (a) Key Distribution Center (KDC), and (b) Client (C). At start of execution, open at least 3 Terminal windows, one each for the KDC and two clients. In each terminal, invoke the corresponding program as described later. All communications between the KDC and the clients will be via TCP sockets.

The KDC's activities include: (i) storing clients' master keys in a file (that uses encryption/hash such that the master keys are not publicly readable); (ii) generating a symmetric key for use between two clients (A and B) for secure data communication; either A or B can request the secret key for their communication.

The data encryption algorithm to be used for client-client and KDC-client communication is: AES-128-CBC, i.e. 128-bit key with CBC mode.

1 KDC

The KDC is invoked as follows:

```
./kdc -p portid -o outfilename -f pwdfile
```

The **kdc** process will listen on port number (*portid*) specified in the command line; the diagnostic output messages (that describe kdc's activities) will be stored in the file named, *outfilename*; the client's details will be stored in the password file (*pwdfile*), specified in the command line.

For example:

```
./kdc -p 12345 -o out.txt -f passwd.txt
```

The KDC's activities are:

1. A message from a client is received with the following format:

   ```
   | 301| IPAddress| ClientPortNum| ClientMasterKey| ClientName|
   ```

   In the above message:

- the first field (“301”) is of type integer; Here, the code 301 indicates that this is a registration message from the client to the KDC.
- the second field is an IPv4 address, stored as a dotted string of length 16 bytes (e.g. “10.4.5.11”);
- the third field is a TCP port number, stored as a string of length 8 bytes (e.g. “35678”);
- the fourth field is the passphrase, a string of length 12 bytes; This string is converted into a 128-bit key and stored in an encrypted manner (with base64 encoding) in the password file maintained by the KDC.
- the last field is the client’s name, a string of length 12 bytes (only lowercase letters from ’a’ – ’z’ are permitted), e.g. “alice”.

The KDC will store the client’s details in the specified password file, as shown in the following example:

```
:alice:10.4.5.11:35678:ABCDEFabcdef123456789=:
:babu:10.4.6.13:45561:abcdef12345ABCDEF6789=:
```

If the username specified above already exists, the KDC overwrites the password file with the latest information.

Upon successful retrieval of the message and storage in the password file, the KDC will send a message to the client in the following format:

```
| 302| ClientName|
```

When the client receives this message with code “302”, it processes the message but takes no further action.

2. A message from a client is received with the following format:

```
| 305| E_KA[ IDA|| IDB|| Nonce1] | IDA |
```

This message is sent from client (IDA) to KDC, requesting a secret key for communication with client (IDB). Assume that an entry for IDB is present in the password file. The KDC sends a response as follows:

Updated on Feb. 18, 2020 to match what was explained in class:

```
| 306| E_KA[ Ks || IDA || IDB || Nonce1 || IPAddrB || PortNoB ||
   E_KB[ Ks || IDA || IDB || Nonce1 || IPAddrA || PortNoA]  ]  |
```

The KDC generates *Ks* as a 8-byte random string from the character set, $\{'A'-'Z','a'-'z','0'-'9'\}$. The message also includes Client B’s IP Address and Port Number that it is listening to.

The client receives this message and retrieves the key, *Ks*, for communication with client IDB, as explained in the class, by decrypting uses its shared master key with the KDC. It then sends a message to the specified IP address and Port number of Client B.

```
| 309| E_KB[ Ks || IDA || IDB || Nonce1 || IPAddrA || PortNoA] || IDA |
```

2 Client

A client will act either as a sender or a receiver. If it is a sender, the client program is invoked as follows:

```
./client -n myname -m S -o othername  -i inputfile -a kdcip -p kdcport
```

Here, *myname* denotes the client's (sender's) name; *othername* denotes the receiver's name; *inputfile* contains the contents that have to be encrypted and sent to the receiver; *kdcip* indicates the KDC's IP address; *kdcport* indicates the KDC's port number.

If it is a receiver, the client program is invoked as follows:

```
./client -n myname -m R -s outenc -o outfile -a kdcip   -p kdcport
```

Here, *myname* denotes the client's (receiver's) name; *outenc* contains the contents received by the receiver; *outfile* contains the decrypted contents and received by the receiver; *kdcip* indicates the KDC's IP address; *kdcport* indicates the KDC's port number.

The client's activities are:

- Initiate the registration with the KDC, as explained above.
- After completion of initialization step, the client will sleep for 15 seconds, to ensure that all other clients are registered.
- If the client is a sender, it will initiate key request protocol with the server; once it obtains the secret key (Ks), it will follow the protocol described in class for client-to-client communication and encrypted data transfer.
- If the client is a receiver, it will wait to receive sender's messages on its TCP port. The encrypted data file sent by the sender will be received and stored in the receiver's output file (specified in the command line).
- The specific message formats are to be designed by you.

3 Sample Run

3.1 KDC Terminal Window

Assume that the KDC is running on a system with IP address of `10.21.22.23`.

```
Prompt> script kdcscript
System outputs: Script started, output file is kdcscript
Prompt> ./kdc -p 12345 -o out.txt -f pwd.txt
...
Starts TPC server to listen on port 12345
Wait for messages from clients..
Type Ctrl-C to finally quit
Prompt> exit
System outputs: Script done, output file is kdcscript
Prompt> cat kdcscript
```

3.2 Sender Client Terminal Window

```
P> script sendscript
...
P> ./client -n alice -m S -o bob  -i in.txt -a 10.21.22.23 -p 12345
...
Contacts KDC and registers..
Sleeps for 15 seconds ..
Contacts KDC for secret key ....
Quits after sending the file to bob.
P> exit
System outputs: Script done, output file is sendscript
P> cat sendscript
```

3.3 Receiver Client Terminal Window

```
Prompt> script recvscript
...
Prompt> ./client -n bob -m R -s outenc.txt -o out.txt -a 10.21.22.23 -p 12345
...
Contacts KDC and registers..
Sleeps for 15 seconds ..
Contacts KDC for secret key ....
Quits after recving the file from alice.
Prompt> exit
System outputs: Script done, output file is recvscript
Prompt> cat recvscript
```

4 What to Submit on IITM Moodle

A single tar.gz file with name ROLLNO-Lab2.tgz (OR) ROLLNO1-ROLLNO2-Lab2.tgz containing:

- Source files and Makefile
- A README File that explains how to compile and run the programs; whether your programs works correctly or whether there are any known bugs/errors in your program.
- Using the "script" Linux command, record the session from the Terminal window for KDC, sender and receiver; thus, there will be three typescript files, named as above.

Your system may be tested with the KDC, Sender and Receiver running on different machines (or) virtual machines on a given system.

For 2-member teams, upload the tar-file to ROLLNO1's account; also, each member should upload an INDIVIDUAL a statement describing both team members contributions to the assignment. The final grade may be different for the two team members, based on these statements and the instructor's assessment.

5 Grading

- KDC implementation: 40 points
- Client implementation: 40 points
- Generation of typescript: 10 points
- Viva Voce Session (to answer questions about the program, demo. of running code, etc.): 10 points

6 Policies

- This is an INDIVIDUAL / 2-member TEAM assignment. Please refer to first day handout (on Moodle) regarding penalties for any form of academic dishonesty, plagiarism, etc. There should be no downloaded code. Teams should not communicate/share any code with each other.
- Plagiarism Checking Software will be used.