SIL Network & System Security; Assignment No. 3, due THU April 26, 2017

Listed below, you will find brief description of 4 projects, numbered 0 through 3. In groups of 2, you are required to pick one (see algorithm to pick a project), complete that project and submit a report (with a working system) on or before THU April 26. The outcome will be evaluated by me.

The algorithm to pick a project:

A. you are required to pick project numbered 0, 1, 2, or 3 as determined by k = A1+A2 mod 4, where

   A1 = last_4_digits_of_entry_no_of_first_student, and A2 = last_4_digits_of_entry_no_of_second_student.

B. If A1 = 3, and you did a similar project on "key distribution centre" as part of Programming Exercise no. 1, then re-do the step 1, but this time with mod 3.

The submission will consist of three parts:

1. a 2 to 4 page document describing the system you have designed,
2. the code as a separate file, and
3. 5 to 8 slides that you will use to present your work.

Project no. 0: Implementation of Kerberos

You are required to (a) build an AS, a TGS, an application server (indeed a web server), and (b) build one or more clients that wish to connect to the application server using the services of the AS and TGS (Please see slides on Kerberos)

To do so, you will need to:

- ensure that AS and TGS have the required information concerning the server(s) and the clients,
- AS, TGS, the servers and clients are able to generate/interpret tickets and thus provide services to clients as when sought.

Once a session key has been established between a client and the web application server, the client can download the home page of the web server.

Project no. 1: Certification Authority (CA)

You are required to (a) build a CA, and (b) build clients that wish to send messages suitably encrypted with public key of receiver, but only after they know the other client's public key in a secure manner. There are two ways for client A to know the public key of another client, B: (a) get it from CA (which is rarely done), or receive a "certificate" from B itself. (Pl. refer Slide 81 of Lecture 6). We will presently limit the fields in the "certificate" to the following:

$CERT_A = ENC_{PRX} (ID_A, KU_A, T_A, \sout{DUR_A, INFO_{CA}})$

where

- PR_X is private key of certification authority (PU_X is public key of certification authority)
- ID_A is user ID,
- KU_A is public key of A,
- T_A is time of issuance of certificate.

To do so, you will need to:

- ensure that clients already (somehow) know the public key of the certification authority,
- CA has the public keys of all the clients, and corresponding to which the clients themselves have their corresponding private keys with themselves,
- messages between CA and clients are encrypted using RSA algorithm (with fixed parameters (n, p, q)) and using CA's private key,
- messages sent/received between clients (once they have each other client's public key) are encrypted using RSA algorithm using the same parameters (n, p, q).
- find a way to generate and encode "current time".

Once the public keys have become known to the clients, the clients should encrypt and send “hello xxx” message using their private keys.

Project no. 2: Diffie-Hellman key generation and exchange

You are required to build clients A and B that wish to send messages from B to A (only B to A) but encrypted using Elgamal Cryptosystem, but only after having exchanged messages that finally result in computation of one-time keys using the Diffie Hellman protocol. (Pl. refer Slides 18 to 22 of Lecture 8 Part 1.)

The Elgamal Cryptosystem uses the parameters (q, a). These are known to each other using “other means”.

To ensure that man-in-the-middle attack is not possible, the two clients A and B send initial messages to each other by adding to each message a MAC (or an “integrity check”) that itself is based on HMAC algorithm and uses a shared “secret”.

To do so, you will need to:

- ensure that clients already (somehow) know the shared “secret” to be used with HMAC,
- assume that Diffie-Hellman parameters (q, a) are already fixed, and known to them,
- (once they know how to generate or compute one-time passwords) messages sent from B to A are encrypted using Elgamal cryptosystem with parameters, (q, a).

Once the one-time keys have been computed (or can be computed on the fly), B should encrypt and send messages “hello 1”, “hello 2”, “hello 3”.

Project no. 3: Public Key Distribution Authority (PKDA)

You are required to (a) build a PKDA, and (b) build clients that wish to send messages suitably encrypted with public key of receiver but of course only after they know each other’s public key in a secure manner. (Pl. refer Slide 73 of Lecture 6 on Public-key Cryptography.)

To do so, you will need to:

- ensure that clients already (somehow) know the public key of the PKDA,
- PKDA has the public keys of all the clients, and corresponding to which the clients themselves have their corresponding private keys with themselves,
- messages between PKDA and clients are encrypted using RSA algorithm (with fixed parameters (n, p, q)) and using PKDA’s private key,
- messages sent/received between clients are encrypted using each other’s public keys, and
- find a way to generate and encode “current time”, as also nonces.

Once the public keys have become known to the clients, the clients should encrypt and send “hello xxx” messages using receiver’s public keys.