CSE 350/550 Network Security; Assignment No. 3, due Tuesday Nov 10, 2020

Listed below, you will find brief description of 2 projects, numbered 0 through 1. In groups of 2, you are required to pick one (see algorithm below), complete that project and submit a report (with a working system) on or before Tuesday Nov 10, 2020 midnight.

The algorithm to pick a project: you are required to pick project numbered 0, 1 as determined by k = A1+A2 mod 2, where

A1 = last_4_digits_of_roll_no_of_first_student, and

A2 = last_4_digits_of_roll_no_of_second_student.

The submission will consist of three parts:

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

Project no. 0: Public-key Certification Authority (CA)

You are required to

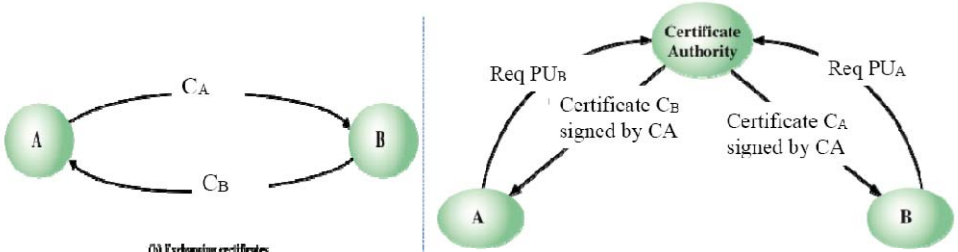
a. build a public-key CA, that responds to requests from clients for their own public-key certificates of that of other clients,

b. build 2 clients that:
   - exchange messages with each other in a confidential manner, or encrypted with public key of receiver, but only after they know the other client's public key in a secure manner, and
   - send requests to the CA for their own public-key certificates or that of other clients.

There are two ways for client A to know the public key of another client, B:

a. Receive a "certificate" from B itself, or

b. Get it from CA – this is the scheme we shall follow.



C_A
A
B
C_B
Certificate Authority
Req PU_B
Req PU_A
Certificate C_B signed by CA
Certificate C_A signed by CA
A
B

(b) Exchanging certificates

(a) Obtaining certificates from CA

We will presently limit the fields in the "certificate" to the following:

$CERT_A = ENC_{PR\text{-}CA}\ (ID_A, PU_A, T_A, DUR_A, ID_{CA})$

where

- PR-CA is private key of certification authority (PU-CA is public key of certification authority)
- ID_A is user ID, ID_{CA} is the ID of the CA,
- PU_A is public key of A,
- T_A is time of issuance of certificate, and DUR_A is the duration for which the certificate is valid.

To do so, you will need to:

- Decide that you will use method (b) above to obtain each other's public key,
- Assume:
  1. that clients already know their [private-key, public-key], but do not have their own certificates or that of others,

2. that clients already (somehow) know the public key of the certification authority,
3. that CA has the public keys of all the clients.

- Decide that messages from CA to clients are encrypted using RSA algorithm and CA's private key,
- Encrypted messages are sent/received between clients once they have each other client's public key, and finally
- Find a way to generate and encode "current time".

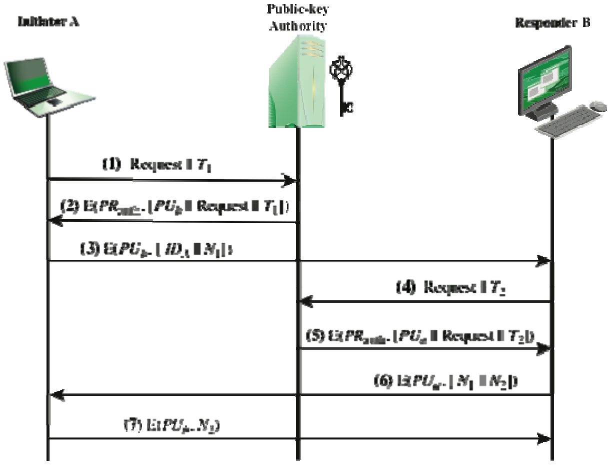
As a test, use the above to determine each other's public key, and then ensure client A can send 3 messages to B, viz. Hello 1, Hello 2, and Hello3. Client B in turn responds with ACK 1, ACK 2, and ACK 3 to messages received from A.

Project no. 1: Public Key Distribution Authority (PKDA)

You are required to:

a. build a PKDA, and
b. build 2 clients that confidentially send messages suitably encrypted with public key of receiver, but only after they know the other client's public key in a secure manner.
c. build 2 clients that:
    - exchange messages with each other in a confidential manner, or encrypted with public key of receiver, but only after they know the other client's public key in a secure manner, and
    - send requests to the PKDA for public-keys of other clients.

Specifically use the scheme described below (do add the ID of the client making the request, and that of client for whom the public-key is sought).


Initiator A
Public-key Authority
Responder B
(1) Request || T_1
(2) $E(PR_{auth}, [PU_b \| Request \| T_1])$
(3) $E(PU_b, [ID_A \| N_1])$
(4) Request || T_2
(5) $E(PR_{auth}, [PU_a \| Request \| T_2])$
(6) $E(PU_a, [N_1 \| N_2])$
(7) $E(PU_b, N_2)$

To do so, you will need to:

- Assume:
    a. that clients already know their [private-key, public-key], but do not have the public-keys of other clients,
    b. that clients already (somehow) know the public key of the distribution authority, PKDA,
    c. that PKDA has the public keys of all the clients,
- Messages between PKDA and clients are encrypted using RSA algorithm and PKDA's private key,
- Encrypted messages are sent/received between clients once they have each other's public key, and finally
- Find a way to generate and encode "current time" and "nonces".

As a test, use the above to determine each other's public key, and then ensure client A can send 3 messages to B, viz. Hi 1, Hi 2, and Hi 3. Client B in turn responds with Got-it 1, Got-it 2, etc. to messages received from A.