CSE 350/550 Network Security; Assignment No. 3, due Sunday, April 2, 2023

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 Sunday, April 2, 2023 midnight.

The algorithm to pick a project: pick project numbered 0 or 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.

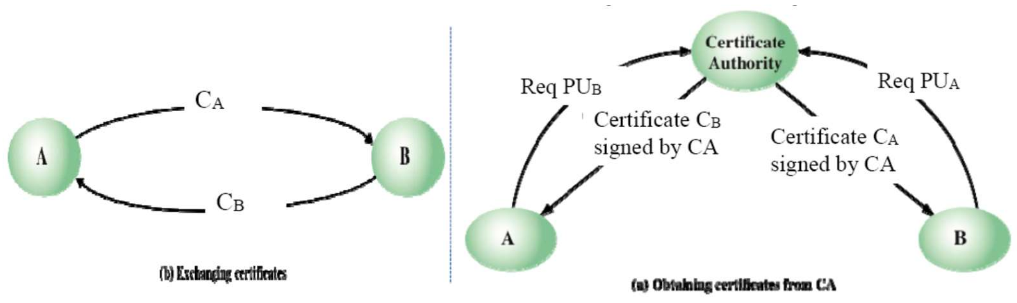
Project no. 0: RSA-based Public-key Certification Authority (CA)

You are required to

a. build a public-key CA, that responds to requests from clients that seek their own public-key certificates OR that of other clients, and
b. build 2 clients that:
   - send requests to the CA for their own public-key certificates OR that of other clients, and
   - exchange messages with each other in a confidential manner, or 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. Receive a "certificate" from B itself, or
b. **Get it from CA – this is the scheme we shall follow.**



C_A
C_B
A
B
(b) Exchanging certificates
Certificate Authority
Req PU_B
Req PU_A
Certificate C_B signed by CA
Certificate C_A signed by CA
A
B
(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 (you decide the format for each of these):

- PR-CA is private key of certification authority (PU-CA is public key of certification authority)
- ID_A is user ID of A, 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 use method (b) above to obtain each other's public key:

- Assume:
  1. that clients already (somehow) know their own [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
- Find a way to generate and encode "current time", and "duration".

Above you need to think hard as to who generates the pair of keys, viz. [private-key, public-key], and how do the CA and/or client get to know it.

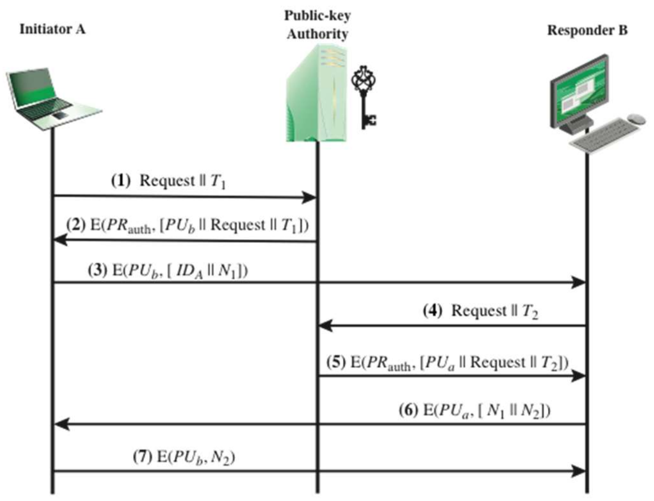
As a test, use the above to determine each other's public key, and then ensure client A can send 3 messages to B, viz. Hello1, Hello2, and Hello3. Client B in turn responds with ACK1, ACK2, and ACK3 to messages received from A.

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

You are required to:

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

Specifically, use the scheme described (below) for a client to request public-key of another client (or for PKDA to respond). Since not all parameters are shown below, add parameters such as IDs, time of day, duration, nonces, etc. to the messages between clients and between client and PKDA.


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 (somehow) know the public key of the distribution authority, PKDA,
   b. that clients already know their own [private-key, public-key], but do not have the public-keys of other clients,
   c. that PKDA has the public keys of all the clients,
- Messages *from* PKDA to 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".

Above you need to think hard as to who generates the pairs of keys, viz. [private-key, public-key], and how do the PKDA and/or client get to know it.

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. Hi1, Hi2, and Hi3. Client B in turn responds with Got-it1, Got-it2, etc. to messages received from A.