5 problems over 5 pages. 60 points. *Closed book. Closed notes. No calculator or electronic device.*

1. [10 points]

Here is a protocol that uses Diffie-Hellman with public parameters g and p (known to all). The goal is for client A to get data from server B. Attacker can read and write the channel.

```
Protocol(A, B) {
   chan ← [];                      // channel
   K ← random();                   // A-B key
   startSystem(Attacker());
   startSystem(Client(A,B,K));  // client A
   startSystem(Server(B,A,K));  // server B
}
```

```
Attacker() {
   α; // initially has A, B, g, p, all programs
   // functions executable by attacker
   function rChan {α ← chan;}     // read chan
   function wChan(x) {chan ← x;}  // write chan
}
```

```
Client(A, B, K) {  // atomicity points: 1
   t ← startThread(client());
   return;

   function client() {
      nL ← random();
      tL ← g^nL mod p;
      tx([A,B,1,tL]);
   1: msg ← rx([B,A,1,.,.]);
      tR ← dec(K,msg[3]);
      kDH ← tR^nL mod p;
      data ← dec(kDH,msg[4])[0];
   }
}
```

```
Server(B, A, K) {  // atomicity points: 1,2
   t ← startThread(server());
   return;

   function server() {
   1: msg ← rx([A,B,1,.]);
      tR ← msg[3];
      nL ← random();
      tL ← g^nL mod p;
      kDH ← tR^nL mod p;
      tx([B,A,1,enc(K,tL),enc(kDH,['HELLO'])]);
   }
}
```

Can the attacker obtain K by dictionary attack, assuming K is a weak key?

- If no, explain why the attacker cannot get values needed for a dictionary attack.
- If yes, give an evolution where the attacker obtains the values for a dictionary attack, and give the dictionary attack itself.

2. [20 points] Here is a protocol based on Needham-Schroeder with KDC `Z`, client `A`, server `B`, and attacker. Attacker can read and write the channel.

```
Protocol(Z, A, B) { // kdc, client, server
   chan ← [];
   hst ← [];            // connect history
   kAZ ← random();      // A-Z key
   kBZ ← random();      // B-Z key
   startSystem(Attacker());
   startSystem(Kdc(Z,A,B,kAZ,kBZ));
   startSystem(Client(A,Z,B,kAZ));
   startSystem(Server(B,Z,A,kBZ));
}
```

```
Client(A, Z, B, kAZ) {  // atomicity points: 1, 2
   t ← startThread(client());
   return;

   function client() {
      while (true) {
         tx([A,Z,B]);
    1:   msg ← rx([Z,A,.]);
         z ← dec(kAZ,msg[2]);
         if (z.size = 3) and z[0] = [B]) {
            kAB ← z[1];
            tkt ← z[2];
            nL ← random();
            tx([A,B,tkt,enc(kAB,nL)]);
    2:      msg ← rx([B,A,.]);
            z ← dec(kAB,msg[2]);
            if (z.size = 2 and z[1] = nL−1) {
               nR ← z[0];
               hst.append([A,kAB]);
               tx([A,B,enc(kAB,nR−1)]);
            }
         }
      }
   }
}
```

```
Kdc(Z, A, B, kAZ, kBZ) {
   // atomicity points: 1
   while (true) {
  1:  msg ← rx([A,Z,B]);
      kAB ← random();
      tkt ← enc(kBZ,[kAB,A]);
      tx([Z,A,enc(kAZ,[B,kAB,tkt])]);
   }
}
```

```
Server(B, Z, A, kBZ) {  // atomicity points: 1,2
   t ← startThread(server());
   return;

   function server() {
      while (true) {
    1:   msg ← rx([A,B,.,.]);
         tkt ← msg[2];
         z ← dec(kBZ,tkt);
         if (z.size = 2 and z[1] = A) {
            kAB ← z[0];
            nR ← dec(kAB,msg[3]);
            nL ← random();
            tx([B,A,enc(kAB,[nL,nR−1])]);
    2:      msg ← rx([A,B,.]);
            if (msg[2] = enc(kAB,nL−1))
               hst.append([B,kAB]);
            }
         }
      }
   }
}
```

```
Attacker() {
   α; // initially has A, B, Z, all programs

   // functions executable by attacker
   function rChan {α ← chan;}     // read chan

   function wChan(x) {chan ← x;}  // write chan

}
```

For each part, answer yes or no. If yes, come up with an argument. If no, come up with a counter-example evolution.

Part a. Does *Inv* A_1 hold, where

A_1 : `((j in hst.keys) and j > 0 and hst[j] = [B,p])` $\Rightarrow$ `hst[j-1] = [A,p]`

Part b. Does *Inv* A_2 hold, where

A_2 : `((j,k in hst.keys) and j` $\neq$ `k and hst[j][0] = hst[k][0])` $\Rightarrow$ `hst[j][1]` $\neq$ `hst[k][1]`

// Neither `A` nor `B` use the same session key more than once.

3. [10 points] Company xLtd has principals X, A_1, A_2, $\cdots$, where X issues certificates for the A_i's, and is their trust anchor. Company yLtd has principals Y, B_1, B_2, $\cdots$, where Y issues certificates for the B_i's, and is their trust anchor. One day, xLtd acquires yLtd. You are to obtain a new PKI for the new xLtd. Parts a and b are independent.

Part a

Modify the old PKIs to obtain a new PKI in which X is the sole trust anchor for all A_i's and B_i's; minimize the number of new certificates.

Give the certificate chain that A_1 needs to get the public key of B_1 in the new PKI.

Give the certificate chain that B_1 needs to get the public key of A_1 in the new PKI.

Part b

Modify the old PKIs to obtain a new PKI in which X is the sole trust anchor for all A_i's, and Y is the sole trust anchor for all B_i's; minimize the number of new certificates.

Give the certificate chain that A_1 needs to get the public key of B_1 in the new PKI.

Give the certificate chain that B_1 needs to get the public key of A_1 in the new PKI.

4. [10 points] Below, "structure of an IP packet" means its headers (IP, TCP, etc, up to the application payload) and, for each header, the values of its fields (addresses, ports, SPIs, next protocol id, etc).

Part a

Client A and server B interact over TCP/IP. The client is at TCP port p and IP address F. The server listens at TCP port q and IP address G.

There are two nodes, J and K, on the IP path between F and G. An IP packet from F to G goes first to J then to K. In the other direction, an IP packet from G to F goes first to K then to J.

Give the structure of an IP packet from A to B (i.e., containing payload of A) at the following points:

- between n_A and J;
- between J and K;
- between K and n_B.

Part b

The configuration in part a is changed as follows.

First, A and B use SSL (over TCP).

Second, F and G operate IPsec-ESP in transport mode providing both encryption and authentication. SPI of 11 is used for both directions.

Third, J and K operate IPsec-ESP in tunnel mode providing both encryption and authentication. SPI of 22 is used for both directions.

Give the structure of an IP packet from A to B (i.e., containing payload of A) at the following points:

- between n_A and J;
- between J and K;
- between K and n_B.

5. [10 points] User A logs in for a session in a Kerberos realm. The user shares a password-derived key, K, with the KDC. After login, the user has a session key S and a ticket-granting ticket TGT.

Now an application B (at say another node) wants to talk (as a client) to the user shell (as a server).

Give the messages exchaged between A, B and the KDC in order for B to talk to A.