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.

```plaintext
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
}
```

Client(A, B, K) { // atomicity points: 1
    t ← startThread(client());
    return;
    function client() {
        nL ← random();
        tl ← g^nL mod p;
        tx([A, B, 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.

\[
\text{Protocol}(Z, A, B) \quad \text{// kdc, client, server} \\
\begin{align*}
\text{chan} &\leftarrow \emptyset; \\
\text{hst} &\leftarrow \emptyset; \\
\text{kAZ} &\leftarrow \text{random}(); \\
\text{kBZ} &\leftarrow \text{random}(); \\
\text{startSystem}(\text{Attacker}()); \\
\text{startSystem}(\text{Kdc}(Z, A, B, \text{kAZ}, \text{kBZ})); \\
\text{startSystem}(\text{Client}(A, Z, B, \text{kAZ})); \\
\text{startSystem}(\text{Server}(B, Z, A, \text{kBZ}));
\end{align*}
\]

\[
\text{Client}(A, Z, B, \text{kAZ}) \quad \text{// atomicity points: 1, 2} \\
t \leftarrow \text{startThread(\text{client}());} \\
\text{return;}
\]

\[
\begin{align*}
\text{function} \quad \text{client}() \quad \{ \\
\quad \text{while (true) \{} \\
\quad \quad \text{tx}([A,Z,B]); \\
\quad 1: \quad \text{msg} \leftarrow \text{rx}([Z,A,.]); \\
\quad \quad \quad z \leftarrow \text{dec(kAZ, msg[2]);} \\
\quad \quad \quad \text{if (z.size = 3 and z[0] = } [B]) \{ \\
\quad \quad \quad \quad \text{kAB} \leftarrow z[1]; \\
\quad \quad \quad \quad \text{tkt} \leftarrow z[2]; \\
\quad \quad \quad \quad \text{nL} \leftarrow \text{random}(); \\
\quad \quad \quad \quad \text{tx}([A,B,\text{tkt}, \text{enc(kAB,nL)}]); \\
\quad \quad 2: \quad \text{msg} \leftarrow \text{rx}([B,A,.]); \\
\quad \quad \quad z \leftarrow \text{dec(kAB, msg[2]);} \\
\quad \quad \quad \text{if (z.size = 2 and z[1] = nL-1) \{} \\
\quad \quad \quad \quad \text{nR} \leftarrow z[0]; \\
\quad \quad \quad \quad \text{hst.append([A,kAB]);} \\
\quad \quad \quad \quad \text{tx}([A,B,\text{enc(kAB,nR-1)}]); \\
\quad \quad \} \\
\quad \} \\
\}
\]
\]

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

\[
\text{Server}(B, Z, A, \text{kBZ}) \quad \text{// atomicity points: 1,2} \\
t \leftarrow \text{startThread(\text{server}());} \\
\text{return;}
\]

\[
\begin{align*}
\text{function} \quad \text{server}() \quad \{ \\
\quad \text{while (true) \{} \\
\quad \quad 1: \quad \text{msg} \leftarrow \text{rx}([A,B,...]); \\
\quad \quad \quad \text{tkt} \leftarrow \text{msg[2]}; \\
\quad \quad \quad \text{z} \leftarrow \text{dec(kBZ, tkt);} \\
\quad \quad \quad \text{if (z.size = 2 and z[1] = A) \{} \\
\quad \quad \quad \quad \text{kAB} \leftarrow z[0]; \\
\quad \quad \quad \quad \text{nR} \leftarrow \text{dec(kAB, msg[3]);} \\
\quad \quad \quad \quad \text{nl} \leftarrow \text{random}(); \\
\quad \quad \quad \quad \text{tx}([B,A,\text{enc(kAB,[nL,nR-1])}]); \\
\quad \quad 2: \quad \text{msg} \leftarrow \text{rx}([A,B,.]); \\
\quad \quad \quad \text{if (msg[2] = enc(kAB, nL-1)) \{} \\
\quad \quad \quad \quad \text{hst.append([B,kAB]);} \\
\quad \quad \} \\
\quad \}
\]
\]

\[
\text{Attacker}() \quad \{ \\
\quad \alpha; \quad \text{// initially has A, B, Z, all programs} \\
\quad \text{// functions executable by attacker} \\
\quad \text{function} \quad \text{rChan} \quad \{ \alpha \leftarrow \text{chan}; \} \quad \text{// read chan} \\
\quad \text{function} \quad \text{wChan(x)} \quad \{ \text{chan} \leftarrow x; \} \quad \text{// write chan} \\
\}
\]

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

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

**Part b.** Does Inv \( A_2 \) hold, where

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

// Neither A nor B use the same session key more than once.
3. [10 points] Company xLtd has principals $X, A_1, A_2, \ldots$, where $X$ issues certificates for the $A_i$’s, and is their trust anchor. Company yLtd has principals $Y, B_1, B_2, \ldots$, 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 exchanged between $A$, $B$ and the KDC in order for $B$ to talk to $A$. 