Solution to 1 [20 points]

Solution to part a [10 points]

We prove it.

The predicates below are from homework 5A. G_1 is the conjunction of A_1 and A_2 (5A part 1). G_1 is the conjunction of B_1 and B_2 (5A part 2). C_2–C_5 are as in 5A part 3.

G_1: ((y in α.*inpts*(A.mKey)) or (y in α.*inpts*(A.mKey))) $\Rightarrow$ (y *seu* A.mKey)

G_2: ((y in α.*inpts*(B.mKey)) or (y in α.*inpts*(B.mKey))) $\Rightarrow$ (y *seu* B.mKey)

C_2: ((y in α.*inpts*(A.mKey) or chan.*inpts*(A.mKey))
and y = enc(A.mKey,p) and p.size = 4)
$\Rightarrow$ ((p[2] *ncf* α) and (p[3] *seu* Z.mKeyB) and p[3] = enc(Z.mKeyB,[p[2],A]))

C_3: ((y in α.*inpts*(B.mKey) or chan.*inpts*(A.mKey))
and y = enc(B.mKey,q) and q.size = 2)
$\Rightarrow$ (q[0] *ncf* α)

C_4: ((A.kAB defined) and (y in α.*inpts*(A.kAB)))
$\Rightarrow$ ((y *seu* A.kAB) or (y *seu* A.mKey) or (y *seu* B.mKey))

C_5: ((B.kAB defined) and (y in α.*inpts*(B.kAB)))
$\Rightarrow$ ((y *seu* B.kAB) or (y *seu* A.mKey) or (y *seu* B.mKey))

Solution to part b [10 points]

We prove it.

The predicates below are from homework 7A (with the same labels).

D_1: (A at 2) $\Rightarrow$ ([.,A.kAB] not in hst)

D_2: ((B at 2) and (enc(B.kAB,B.n3−1) in chan/α))
$\Rightarrow$ (([B,B.kAB] not in hst) and ([A,B.kAB] in hst))

D_3: ((A at 1) and (enc(A.mKey,[A.n1,B,k,tkt]) in chan/α)
$\Rightarrow$ ([.,k] not in hst)

D_5: (A.mKey *ncf* α) and (B.mKey *ncf* α) // G_1, G_2
and ((A.kAB defined) $\Rightarrow$ (A.kAB *ncf* α)) // implied by C_4
and ((B.kAB defined) $\Rightarrow$ (B.kAB *ncf* α)) // implied by C_5

F_0: ((i in hst.keys, i $\neq$0) and hst[i] = [B,k]) $\Rightarrow$ hst[i-1] = [A,k]

F_1: ((B at 2) and (enc(B.kAB,B.n3−1) in chan/α))
$\Rightarrow$ hst.last = [A,B.kAB]

F_2: ((A at 2) and (enc(A.kAB,[A.n2−1,.]) in chan/α))
$\Rightarrow$ (enc(.,B.n3−1) not in chan/α)

Solution to 2 [20 points]

We disprove both assertions.

Counter-example evolution (from homework 7B)

1. Initial step: `A` send `[A,Z,B,n1]`.
2. `Z.1` step: receive `[A,Z,B,n1]`, send `[Z,A,enc(kAZ,[n1,B,k,tkt)]` with `tkt` = `enc(kBZ,[k,A])`.
3. `A.1` step: receive message in step 2.
   After: `A.kAB = k`
4. Attacker `getPwdA`: add `kAZ` to α, set `A.mKey` and `Z.mKeyA` to random value.
   After: `kAZ` and `[Z,A,enc(kAZ,[n1,B,k,tkt])]` are in α. From these attacker gets `[n1,B,k,tkt]`, from which it gets `k`.

   Part a predicate does not hold in this state

4. Attacker: send message `[A,B,tkt,enc(k,9)]`. `B.1`: receive above message, send message `[B,A,enc(k,[9,n3])`.
   Attacker: receive above message, send message `[A,B,enc(k,n3−1)`.
   `B.2`: receive above message, add `[B,k]` to `hst`.

   Part b predicate does not hold in this state

Solution to 3 [10 points]

Solution to part a [2 points]

Attacker cannot obtain W by offline-dictionary attack because the only quantities that are encrypted using W, i.e., enc(W,cB) and enc(W,nB+1), are themselves encrypted using the Diffie-Hellman key, which is a strong key.

Solution to part b [8 points]

Attacker can obtain W by offline-dictionary.

First, it does the classic man-in-the-middle attack, from which it obtains enc(W,cB+1) and enc(W,cB). It can then do an offline-dictionary attack on these two quantities.

Details of the man-in-the-middle attack [4 points]:

- It intercepts A's initial message, say [A,B,1,tA].
- It generates a DH random number nZ and sets tZ to g^{nZ} mod p. It sends [A,B,1,tZ]. It constructs the DH key shared with A, i.e., kAZ $\leftarrow$ tA^{nZ} mod p.
- It intercepts B's response message, say [B,A,tB,enc(kBZ,enc(W,cB))].
- It constructs the DH key shared with B, i.e., kBZ $\leftarrow$ tB^{nZ} mod p. It decrypts the last field of the message using kBZ and then encrypts it with kAZ. It sends [B,A,tZ,enc(kAZ,enc(W,cB))]. It now has enc(W,cB).
- It intercepts A's response message, say [A,B,enc(kAZ,enc(W,cB+1))].
- It decrypts the last field of the message using kAZ, thereby obtaining enc(W,cB+1).
- It now has enc(W,cB+1).

Details of the dictionary attack [4 points]:

- Let p = enc(W,cB+1) and q = enc(W,cB).

  For candidate password, obtain candidate key cW and check for dec(cW,p) = dec(cW,q)+1 until match.

Solution to 4 [10 points]

Part a: `A` authenticates `B` when its ssl receives `enc(K,[another keyed hash of handshake])`.
`B` authenticates `A` at end of authentication handshake involving `W`.

Part b: No. Messages between client `A` and its ssl are not encrypted.

Part c: A tcp message in the data exchange phase would have:
ip/tcp header: not encrypted.
ssl header: encrypted.
ssl payload: encrypted.

Part c: No. Because the data between ssl and tcp are encrypted.