1. What are the tradeoffs between using nonces and using timestamps in protocols? What are some situations where each would be appropriate?

2. Imagine the following proposal, designed to allow a party A to authenticate to a party B.

   Let K_x be the public key of agent x and K_x^{-1} be the corresponding private key. N_x represents a nonce. Before the protocol, B believes that K_A is A's public key.

   $$\begin{aligned} A &\rightarrow B : \{A, N_1\}_{K_B} \\ B &\rightarrow A : \{N_1, N_2\}_{K_A} \\ A &\rightarrow B : \{N_2\}_{K_B} \end{aligned}$$

   Suppose this protocol was used to authenticate users to UNIX servers. Each users would only have to deal with one private key and the public key of each server they need to talk to. Each server would just store a copy of each authorized user's public key.

   Unfortunately, this scheme is really broken. Show how a malicious server can abuse the protocol and suggest a fix that would prevent this attack.

3. You've been hired by a company to consult on the security of their new product. The product is a fancy distributed chat room, similar in some ways to IRC (for more info, see `www.irchelp.org`, but note that this system is not intended to be compatible with IRC). They claim they want their chat product to be "secure". As you know, there are many different ways this product could be attacked. They claim to want to support both "public" chat rooms (where anybody in the world can log in) and "private" chat rooms (where some administrator can restrict access).

   (a) What cryptographic primitives would be appropriate to use for the chat service? Given that there are many, many chat servers all talking to each other, with different users logged into each chat server, how might one chat server prove who you are to another chat server? How can you prevent a user from being spoofed? Identify all the *trust relationships* in the system, that is, point out the places where the system can't *prove* some important fact conclusively but instead is relying on somebody else to tell it facts.

   (b) What software engineering precautions should be taken in the design and implementation of this system? Are there any special issues that arise from the design of the chat system? Read the IRC FAQ. Are there any features you'd cut out or change to improve security?