1. (Question 9.10.2) Consider the following authentication protocol based on a cryptographic hash \( H \). The server stores \( z = H(w) \), where \( w \) is the user’s password. To log on, the user enters their password \( w \). The server sends a random challenge \( r \), and the client responds with \( h = H(H(w), r) \). The server accepts the user iff \( h = H(z, r) \). Does this give an example of an authentication scheme that is not based on public-key cryptography, yet is secure against both eavesdropping and server compromise?

2. (Question 11.9.5) Consider the following 2-round authentication protocol where the server does not have to maintain any state. The client and server share a symmetric key \( k \). To authenticate, the server sends a random challenge \( r \) and the client responds with \( \langle r, F_k(r) \rangle \), where \( F \) is a block cipher. Upon receiving the message \( \langle \hat{r}, \hat{y} \rangle \), the server accepts iff \( \hat{y} = F_k(r) \). Is this protocol secure?

3. (Question 11.9.6) Consider the following variant of the protocol from the previous question. The server sends \( \langle r, F_{k'}(r) \rangle \), where \( r \) is again chosen at random and \( k' \) represents a secret key known only to the server. The client responds with \( \langle r, F_{k'}(r), F_k(r) \rangle \). Upon receiving the message \( \langle r, x, y \rangle \), the server accepts iff (1) \( x = F_{k'}(r) \) and (2) \( y = F_k(r) \). Is this protocol secure?

4. (Question 11.9.9) The expanded Needham-Schroeder protocol (discussed in class and also Protocol 11-19 in the book) can be shortened to 6 rounds, without compromising security, by removing the final message. Why is this true? Specifically, why does the resulting 6-round protocol convince Bob he is talking to Alice?

5. (Question 12.5.15) Consider the following protocol, where the client begins holding password \( w \) and the server holds \( g^w \mod p \). (Here, \( g \) and \( p \) are Diffie-Hellman parameters.)

   (a) The client sends \( g^a \mod p \) for random \( a \).
   (b) The server sends \( g^b \mod p \) for random \( b \), along with a random challenge \( r_1 \).
   (c) The client computes \( K = H(g^{ab} \mod p, g^{wb} \mod p) \), where \( H \) is a cryptographic hash function. The client then sends \( \langle F_K(r_1), r_2 \rangle \) to the server, where \( F \) is a block cipher.
(d) The server also computes $K$, and accepts the incoming message $\langle y, r_2 \rangle$ from the client iff $y \equiv F_K(r_1)$. If so, the server responds with $F_K(r_2)$.

(e) The client accepts the server’s response $x$ iff $x \equiv F_K(r_2)$.

How do the client and server each compute $K$? Show also how an adversary impersonating the server can perform an off-line dictionary attack on the password.