Question 1  *Password Hashing*  
(10 min)

When storing a password $p$ for user $u$, a website randomly generates a string (called a salt) $s$, and saves the tuple $(u, s, r = H(p||s))$, where $H$ is a cryptographic hash function.

(a) Say user $u$ tries to log in submitting a password $p'$ (which may or may not be the same as $p$). How does the site check if the user should be allowed to log in?

(b) Why use a hash function $H$ rather than just store $(u, p)$? Isn’t that just so much simpler? Keep it simple stupid, right?

(c) What is the purpose of the salt $s$?

(d) Suppose the site has three candidate hash function to choose from, $H_1$, $H_2$, and $H_3$. They each satisfy the following properties as displayed in the table.

<table>
<thead>
<tr>
<th>Functions</th>
<th>One-Way</th>
<th>Second Pre-Image Resistant</th>
<th>Collision Resistant</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H_1$</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>$H_2$</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>$H_3$</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Which of these hash functions are suitable choices for the website’s password hashing scheme, in that they provide a significant gain in security over just storing passwords?
Question 2  PKI and TLS  
(25 min)

Your web browser’s traffic is secured when you use HTTPS, which relies on the TLS protocol. In this problem, we will construct the basic framework of TLS (though the actual TLS protocol is more complex and detailed).

Our browser \( B \) wants to exchange messages with Apache web server \( A \). The Apache server \( A \) has an RSA key pair \((PK, SK)\). Whenever a browser \( B \) initiates communication with \( A \), \( A \) first sends a certificate essentially saying “I’m server \( A \) and my public key is \( PK \)”.

(a) How can \( B \) be sure that the certificate it supposedly got is really from \( A \)? Describe what is done by the browser to authenticate the certificate. Whom is the browser trusting?

(b) Why is it okay for \( B \) to get the certificate directly from \( A \), rather than querying a certificate authority (CA)? What are some advantages of getting the certificate directly?

(c) Now \( B \) has authenticated the certificate, so it knows \( A \)'s public key. RSA is slower than symmetric-key crypto, so ideally we want to use AES instead of RSA to encrypt messages. \( B \) generates a random AES key \( k \). Describe a simple protocol such that \( A \) and only \( A \) obtains the true value of \( k \), and both \( B \) and \( A \) know they each have the same \( k \).

(d) Oh noes! The server admin has made a terrible mistake and accidentally leaked \( SK \), the RSA secret key. Evil Eve, who happens to be between \( B \) and \( A \) in the communication path, has gotten her hands on a copy of \( SK \). What sort of badness could she do from that point onward?

(e) Eve, always a data squirrel, has also been recording the encrypted traffic between \( B \) and \( A \) even before the secret key leak. Now that she has the secret key, she’s glad she did all that work to save the old traffic data. How is she able to access the encrypted data?

(f) How might our protocol be changed to protect against this attack on recorded old traffic?

(g) What should \( A \) do now that its secret key has been leaked?
Question 3  Crypto Protocol Errors  (15 min)

Alice (A) and Bob (B) want a secure communication channel between them, and will rely on a trusted server (S). Assume the following notation:

- A and B are the identities of Alice and Bob respectively.
- $K_{AS}$ is a symmetric key only known to A and S.
- $K_{BS}$ is a symmetric key only known to B and S.
- $N_A$ and $N_B$ are random nonces generated by A and B, respectively.
- $K_{AB}$ is a symmetric key generated for this communication session between A and B.

They use the following protocol (known as the Needham-Schroeder protocol):

- $A \rightarrow S : A, B, N_A$
- $S \rightarrow A : Enc_{K_{AS}}(N_A, K_{AB}, B, Enc_{K_{BS}}(K_{AB}, A))$
- $A \rightarrow B : Enc_{K_{BS}}(K_{AB}, A)$
- $B \rightarrow A : Enc_{K_{AB}}(N_B)$
- $A \rightarrow B : Enc_{K_{AB}}(N_B - 1)$

(a) Eve has been monitoring the Needham-Schroeder protocol messages exchanged between Alice and Bob. One day Eve uncovers an old session key (an old $K_{AB}$ value) from a previous protocol exchange. How can Eve leverage this old key?

(b) How do you fix the protocol vulnerability?