# Suggested Solutions to Homework Assignment #3

(prepared by Willy Chang)

Exercise problems from [Stallings 6E, intl.]:

**11.2** **a**. For clarity, we use overbars for complementation. We have:

$$E(\overline{M_i}, \overline{H_{i-1}}) = \overline{E(M_i, H_{i-1})} \oplus \overline{H_{i-1}} = E(M_i, H_{i-1}) \oplus H_{i-1}$$

Therefore, the hash function of message $M$ with initial value $I$ is the same as the hash function for message $N$ with initial value $\overline{I}$ for any given $I$, where

$$M = M_1||M_2||...||M_n; \quad N = \overline{M_1}||M_2||...||M_n$$

**b**. The same line of reasoning applies with the $M$s and $H$s reversed in the derivation.

**11.5** The opponent has the two-block message $B1$, $B2$ and its hash $RSAH(B1, B2)$. The following attack will work. Choose an arbitrary $C1$ and choose $C2$ such that:

$$C2 = RSA(C1) \oplus RSA(B1) \oplus B2$$

then

$$\begin{aligned} RSA(C1) \oplus C2 &= RSA(C1) \oplus RSA(C1) \oplus RSA(B1) \oplus B2 \\ &= RSA(B1) \oplus B2 \end{aligned}$$

so

$$\begin{aligned} RSAH(C1, C2) &= RSA[RSA(C1) \oplus C2] = RSA[RSA(B1) \oplus B2] \\ &= RSAH(B1, B2) \end{aligned}$$

**12.1** No. If internal error control is used, error propagation in the deciphering operation introduces too many errors for the error control code to correct.

**12.9** **a**. The following matrix shows the message for each received 2-bit word.

| | Word | | | |
|---|---|---|---|---|
| Key | 00 | 01 | 10 | 11 |
| 1 | 0 | 1 | - | - |
| 2 | 1 | - | 0 | - |
| 3 | - | 0 | - | 1 |
| 4 | - | - | 1 | 0 |

**b**. The probability that some one can successfully impersonate Alice is 0.5 because only two of the four words are possible as transmitted word under the joint secret key.

**c**. An opponent Eve who tries to replace a transmitted message by another one will know that only two keys can possibly have been used, but she doesn't know which one. So, the probability of a successful substitution is also 0.5.

**14.1** **a**. A sends a connection request to B, with an event marker or nonce ($Na$) encrypted with the key that A shares with the KDC. If B is prepared to accept the connection, it sends a request to the KDC for a session key, including A's encrypted nonce plus a nonce generated by B ($Nb$) and encrypted with the key that B shares with the KDC. The KDC returns two encrypted blocks to B. One block is intended for B and includes the session key, A's identifier, and B's nonce. A similar block is prepared for A and passed from the KDC to B and then to A. A and B have now securely obtained the session key and, because of the nonces, are assured that the other is authentic.

**b**. The proposed scheme appears to provide the same degree of security as that of Figure 14.3. One advantage of the proposed scheme is that, in the event that B rejects a connection, the overhead of an interaction with the KDC is avoided.

**14.2** (a) sending to the server the source name A, the destination name Z (his own), and $E(K_a, R)$, as if A wanted to send him the same message encrypted under the same key $R$ as A did it with B

(b) The server will respond by sending $E(K_z, R)$ to A and Z will intercept that

(c) because Z knows his key $K_z$, he can decrypt $E(K_z, R)$, thus getting his hands on $R$ that can be used to decrypt $E(R, M)$ and obtain $M$.

**14.6** **a**. A believes that she shares $K'_{AB}$ with B since her nonce came back in message 2 encrypted with a key known only to B (and A).
B believes that he shares $K'_{AB}$ with A since $N_A$ was encrypted with $K'_{AB}$, which could only be retrieved from message 2 by someone who knows $K'_{AB}$ (and this is known only by A and B).
A believes that $K'_{AB}$ is fresh since it is included in message 2 together with $N_A$ (and hence message 2 must have been constructed after message 1 was sent).
B believes (indeed, knows) that $K'_{AB}$ is fresh since he chose it himself.

**b**. We consider the following interleaved runs of the protocol:

$$
\begin{aligned}
&1. \quad A \rightarrow C(B) : A, N_A \\
&1'. \quad C(B) \rightarrow A : B, N_A \\
&2'. \quad A \rightarrow C(B) : E(K_{AB}, [N_A, K'_{AB}]) \\
&2. \quad C(B) \rightarrow A : E(K_{AB}, [N_A, K'_{AB}]) \\
&3. \quad A \rightarrow C(B) : E(K'_{AB}, N_A)
\end{aligned}
$$

C cannot encrypt A's nonce, so he needs to get help with message 2. He therefore starts a new run with A, letting A do the encryption and reflecting the reply back. A will accept the unprimed protocol run and believe that B is present.

**c**. To prevent the attack, we need to be more explicit in the messages, e.g. by changing message 2 to include the sender and receiver (inthis order), i.e. to be $E(K_{AB}, [A, B, N_A, K'_{AB}])$.

**15.3** **a**. An unintentionally postdated message (message with a clock time that is in the future with respect to the recipient's clock) that requests a key is sent by a client. An adversary blocks this request message from reaching the KDC. The client gets no response and thinks that an omission or performance failure has occurred. Later, when the client is off-line, the adversary replays the suppressed message from the same workstation (with the same network address) and establishes a secure connection in the client's name.

**b**. An unintentionally postdated message that requests a stock purchase could be suppressed and replayed later, resulting in a stock purchase when the stock price had already changed significantly.

**15.4** All three really serve the same purpose. The difference is in the vulnerability.
In **Usage 1**, an attacker could breach security by inflating $N_a$ and withholding an answer from B for future replay attack, a form of suppress-replay attack.
The attacker could attempt to predict a plausible reply in **Usage 2**, but this will not succeed if the nonces are random. In both Usage 1 and 2, the messages work in either direction. That is, if N is sent in either direction, the response is E[K, N].
In **Usage 3**, the message is encrypted in both directions; the purpose of function f is to assure that messages 1 and 2 are not identical.
Thus, Usage 3 is more secure.

**15.8** **a**. This is a means of authenticating A to B.
$R_1$ serves as a challenge, and only A is able to encrypt $R_1$ so that it can be decrypted with A's public key.

**b**. Someone (e.g., C) can use this mechanism to get A to sign a message. Then, C will present this signature to D along with the message, claiming it was sent by A. This is a problem if A uses its public/private key for both authentication, signatures, etc.