Assignment 2

SENG2250

Sam Dolbel – 3130069

1. Perfect Forward Secrecy

Yes, this KEA protocol provides perfect forward secrecy.

For each communication between the two users, each user provides a **random number** – user *A* selects number r_A and user *B* selects r_B. This number is used only once in a single exchange, then discarded.

This adds a layer of security against any potential attackers, as even if they acquire the long-term key, they won't know any of the random number pairs required to decrypt the past exchanges.

2. Hash Chain

a.

Yes, the authentication protocol provides replay attack resistance.

To ensure the freshness of the message, a **nonce** is added in the transmission by the user. Nonces are used specifically to prevent replay attacks. A malicious attacker attempting to replay the session would need to provide a fresh nonce – attempting to use the same nonce will end in failure.

b.

The protocol does not provide complete forward security. It is deterministic can be brute-forced, assuming the adversary captures all previous messages. If an adversary retrieves key k_i, they know that $k_i = h(k_{i-1})$. Using brute-force methods they can find the value of k_{i-1} where $h(k_{i-1}) = k_i$

Depending on the size of the hash this may be impractical – for example, a 64-bit hash has approximately $1.8 * 10^{19}$ different outputs. However, using a birthday attack this may be reduced to around $5.38 * 10^9$, or 5.38 billion, which is a far more practical number.

To provide forward security, the user can include a **random number** *r* in each message, such that $E(k_i; H_{i-1}, N_u, i, r_i)$ which is multiplied by the hashed value. In other words, session key $H_i = h(H_{i-1} * r_i)$. This will make the true value of H_{i-1} impossible to determine. However, it will also no longer be deterministic.

3. Two-Factor Authentication Protocol Analysis

a.

The protocol achieves two-factor user authentication using the smart-card and password.

b.

The simplest part of the security analysis is the password. The user inputs a password which is then hashed. The hashed password is XOR'ed with the smart-card value B, where B = h(pwd) XOR h(x || ID_c). If the password is correct, the result Z = h(x || ID_c). The advantage of this is that the password is authenticated **before** the message is sent to the server. This removes the risk of any network security attacks when validating the password, assuming the original password was securely sent.

Messages between the user and server are encrypted with hash functions, including hash functions within different hash functions. They provide **forward secrecy**, blocking precedent keys from being compromised.

When the user sends a login request, they produce a random number u. Assuming the request was successful, the server side returns a message containing their own random number v. These nonces are protection against **replay attacks**. An important part of this protection is that the server and client authenticate each other – the nonces produced by both sides must be authenticated by the other side.

The client authentication procedure on the server side renders the protocol functionally immune to any sort of brute-force attack. Depending on factors like message size, an average brute-force attack requires billions of attempts – usually orders of magnitude more. This protocol shuts down all authentication attempts after the 3rd fail, requiring the attacker to prove their identity to the administrator.

When we consider the possibility that one of the authentication factors is compromised, this means one of two things: the attacker knows the password, or the attacker has the smart card.
If an attacker knows the password, they are incapable of even sending a login request without the smart card. Their attack attempt will have been thwarted before it could even start.
If they have the smart card, they will be required to brute-force the hash function to learn the password. As discussed earlier, they have a maximum of 3 attempts to brute-force the password before they are locked out.

4. Multilevel Security

a.

The BLP model is “read-down, write-up”, meaning that subjects cannot read from objects of a higher security label and cannot write to objects of a lower security label.

	Readable Objects	Writable Objects
A	O2, O4	--
B	O4	O3
C	--	O1, O2, O3, O4
D	O4	O1

b.

The Biba model is “read-up, write-down”, meaning that subjects cannot read from objects of a lower security label and cannot write to objects of a higher security label.

	Readable Objects	Writable Objects
A	--	O2, O3, O4
B	O1	O3
C	O3	O4
D	O1, O2, O4	--