Solutions, exercises, set 3

Computer Security

Exercise 3.1
a) No read-up and no write-down.
b) Process A can be temporarily downgraded to a lower security level and write information to an object at that level. After writing, the security level of process A can be restored to $f_S(A)$. The information can then be read by process B.

Exercise 3.2
a) Eve records some encrypted and/or authenticated communication between Alice and Bob and uses it to later fool one or the other of them. For example, if Alice tells Bob to buy 500 Volvo stocks and Eve later replays the conversation with Bob, he will buy more stocks than he should and Alice will get the blame. Note that Eve doesn't have to know *what* she is replaying, she just has to do it and hope for some (disastrous) result.
b) Alice and Bob could make sure that each conversation is unique (not the information in it, of course, but the session setup, encryption, etc.). Some tools would include nonces or timestamps.
c) When Alice contacts the server she uses a nonce. The server responds with a newly created session key. The fact that it is new can be verified since the nonce is included in the response. Furthermore, when contacting Bob with the session key data, Alice includes a time stamp which he can compare against his own clock.

Exercise 3.3

Let's understand the scheme: Alice and Bob have a shared key, S. Alice creates a key pair and sends $X = E_S(PK)$ where PK is her public key. Bob knows S so he can find PK. He chooses a session key SK and sends $Y = E_S(E_{PK}(SK))$ to Alice. Since she knows both S and her private key, she can find SK. Eve can only observe X and Y. (Some observations: Bob needs to trust that Alice never uses an old key pair while, similarly, Alice needs to trust that Bob knows how to generate good session keys.)

The modified scheme is as follows: Alice generates a symmetric key AES and sends $X' = E_S(AES)$ and Bob responds with $Y' = E_S(E_{AES}(SK))$. Eve observes X' and Y'.

Doing a dictionary attack, Eve tries out some possible values for S. She calculates $AES' = D_S(X')$ and $SK' = D_S(D_{AES'}(Y'))$. She uses SK' to decrypt some session data. Since we assume some redundancy in the data sent between Alice and Bob (for example, they might be communicating in a human language), Eve will realize when she has made a correct guess of S and she now has the session key SK'.

Would the same attack apply to the original protocol? Well, Eve could guess S and calculate $PK' = D_S(X)$ but she'll have problems calculating $D_S(D_{PK'}(Y))$ since the public key is of no use for *decrypting*!

Exercise 3.4
a) The man-in-the-middle needs to know p and g, but this is a fair assumption. Alice and Bob can either a) use a predetermined parameter set[1] which has to be assumed public by Kerckhoff's principle, or b) start their conversation by negotiating p and g. This negotiation can be eavesdropped.
b) We'll use $p = 11$ and $g = 2$. a, b, x, z used below will be picked randomly. Alice chooses $a = 4$ and sends $y_a = g^a \bmod p = 5$. She thinks that she sends it to Bob, but Eve intercepts it, picks $z = 3$ and sends $y_z = g^z \bmod p = 8$ to him. Bob cheerfully picks $b = 7$ and sends $y_b = g^b \bmod p = 7$ to Eve, thinking she is Alice. Eve picks $x = 9$ and sends $y_x = g^x \bmod p = 6$ to Alice.

Alice will compute $A = y_x^a \bmod p = 9$ and Bob will find $B = y_z^b \bmod p = 2$. Clearly, they do not share keys. However, Eve calculates $A' = y_a^x \bmod p = 9$ and $B' = y_b^z \bmod p = 2$. She will now be able to relay Alice's and Bob's "secret" communications while recording everything!

We see that Alice chooses a while Bob chooses b, just as intended, but that Eve selects x and z. Alice will have access to a and A and Bob to b and B. Eve will have access to x, z, A and B.

Exercise 3.5
An answer might include the following:

- The TCSEC was a US standard while the Common Criteria is an international one (however, only EAL1 through EAL4 assurances are recognized in all countries; higher levels of assurance have to be accepted in each country that the product targets).
- Evaluation according to TCSEC was free of charge, while a Common Criteria evaluation can be quite costly.
- TCSEC does not separate functionality and assurance, while Common Criteria somewhat separates this by using Protection Profiles.
- TCSEC primarily focuses on operating systems while Common Criteria can be used to evaluates several other types of security products.

Exercise 3.6 Someone, say Alice, that has read sensitive information at Ericsson can potentially transfer this information by writing it to objects in Swedbank. Since there is no conflict of interest (we assume) between Ericsson and Swedbank, this should not be a problem. However, another person, say Bob, working at swedbank can read this sensitive information. Bob might later is hired by Nokia. He can then potentially transfer the information that he read at Swedbank that is in Ericssons data set, to Nokia. The *-property avoids this indirect information flow.

[1] In SSH, there are many variants of Diffie-Hellman, but one option that all clients have to support is `Diffie-Hellman-group1-sha1` which uses a quite scary $p = 2^{1024} - 2^{960} - 1 + 2^{64}\lfloor 2^{894}\pi + 129093 \rfloor$ and a somewhat simpler $g = 2$.

Exercise 3.7
a) The extra restriction says that if we have access to several files, and this access sometimes includes append and sometimes includes read, then the security level of all files that we can append to must dominate the security level of all files we have read access to.
b) Let us first consider only the read access. Since you will always be able to read files with security levels that are dominated by your maximum level, f_S, there is no motivation to use f_C.

If we instead consider append access, then this restriction in the ss-property is motivated since you will never be allowed to read objects with security level that dominates objects that you can append to. However, the result would actually be a little more restrictive than needed. Consider the case where we have 4 classifications. For somplicity we call them 1, 2, 3, 4 where 4 is the highest security level (system high) and 1 is the lowest (system low). Assume that user Alice has $f_S(Alice) = 4$ and that she downgrades her current level to $f_C(Alice) = 1$. In this case, reading an object with $f_O(o) = 2$ while at the same time appending to an object with $f_O(o) = 3$ would not risk any information flow downwards. This is also not violating the original security properties. However, with the proposed property this situation would be forbidden.

Exercise 3.8

Since the Orange book does not separate functionality and assurance it is possible to determine both functionality and assurance for OS A. At least to a certain extent. It has e.g., discretionary access control implemented for individual users and it implements an audit function. However, we do not know if it has mandatory access control. It is possible that it implements some functionality for higher evaluation classes but not all. For OS B we know the assurance level, but we do not have any information about the functionality of the product. To know this we additionally have to look at the Protection Profile(s) that it is evaluated against.

Exercise 3.9
a) Sending $eK_{ab}(n_b)$ would not authenticate A to B since this is the exact same message as A received in the previous message. Sending $eK_{ab}(n_b - 1)$ clearly shows that A has knowledge of the key K_{ab}.
b) Since the authenticator in the previous message, $eK_{ab}(A, T'_a)$ included also the identity A, B will not be able to create $eK_{ab}(T'_a)$ without knowing K_{ab}. The messages in Kerberos V5 has been constructed such that this is true. However, in Kerberos V4 the last message was actually $eK_{ab}(T'_a + 1)$, similar to the message in Needham-Schroeder.

Exercise 3.10
a) Without preauthentication an attacker can send the first message, claiming to be some client, and get an answer encrypted with the key (password) shared by KAS and the client (user). This answer can then be used to attack the password, using e.g., a dictionary attack. With preauthentication, the KAS would not respond if the preauthentication fails. An attacker would not be able to get answers from the KAS in order to check a password. However, if the attacker has the ability to listen to the network traffic, it is still possible

to eavesdrop encrypted packages and make a dictionary attack on these. This of course assumes that the attacker knows some of the plaintext used in the encryption. Some information is the same as in the first client message which can also be eavesdropped, so this should not be considered a problem.
b) If preauthentication is not used the client is authenticated by the fact that he can successfully decrypt the answer from the KAS. Since this answer contains a key to be used in subsequent messages, and this is encrypted with a key derived from the password, the client can not successfully continue the protocol without knowing the password.