SOEN 321

Prob. 1 Alice and Bob share a symmetric key k. Alice sends Bob a message stating, “I owe you $100", and also sends a message authentication code (MAC) on this message computed using the key k. Assuming the MAC algorithm is secure, can Bob go to his bank and prove to the bank teller that Alice does indeed owe him $100 by showing M and MAC(M) to the teller?

Remark (Just in case if you do not know what a MAC scheme is): A message authentication code (MAC) is a short piece of information used to authenticate a message. In other words, to confirm that the message came from the stated sender (its authenticity) and has not been changed in transit (its integrity). One way to build a MAC algorithm is through the use of a keyed hash function where it accepts as input a secret key and an arbitrary-length message to be authenticated, and outputs a MAC. The MAC value protects both the message's data integrity as well as its authenticity, by allowing verifiers (who also possess the secret key) to detect any changes to the message content.

Ans. No. Bob cannot do so because the MAC key has to be known by both Alice and Bob. Thus Bob could have produced this message M and MAC(M) by himself. In other words, unlike digital signature, a MAC scheme does not provide non-repudiation; it only ensure message integrity and authentication

Prob. 2
A security company has determined that, even using a bunch of fast computers, brute- force attacks can break at most a 50-bit key in a reasonable amount of time. So, they decide to design an encryption algorithm that will encrypt 64-bit messages under a 64-bit key. They already have a good block cipher E that can encrypt a 32-bit message under a 32-bit key and that is resilient against all attacks except brute-force attacks. So, for the 64-bit encryption algorithm, they simply split the message M in two parts M_1 and M_2 and the key K in two parts K_1 and K_2. The ciphertext C is then computed as $E_{K1}(M_1)||E_{K2}(M_2)$. Assume we know that C is the encryption of the 64-bit message M (using the 64-bit encryption algorithm, under the unknown key K). Show how to recover the 64-bit key K in a reasonable amount of time.

Ans. First, use brute force to recover K_1. Let $C=C_1||C_2$. Then we know that $C_1 = E_{K1}(M_1)$, and we know $M_1;C_1$, so try all possibilities for K_1 and see which one is consistent with this equation. Next, use brute force to recover K2, by a similar method. This requires $2^{32} + 2^{32} = 2^{33}$ trial decryptions in total, which is easily feasible.

Prob. 3
Alive runs a large website that allows users to log in and share images. When a new user sets up their account, the website hashes their password with SHA256 and stores the hash in a database. When a user logs in, the website hashes the supplied password with SHA256 and compares it to the stored hash. Alice figures that with this scheme, if anyone hacks into your database they will only see hashes and won't learn your users' passwords. Out of curiosity, Alice does a Google search on several hashes in the database and is alarmed to find that, for a few of them, the Google search results reveal the corresponding password. She comes to you for help.

a. What mistake did Alice make in how he stored passwords?
b. What is the consequence of this mistake? In other words, what is the risk that it introduces

and how many of Alice's users could be affected? Does it affect only users whose password hashes are available in Google search, or does it go beyond that?

c. How should Alice store passwords? More specifically, if a user's password is w, what should Alice store in the database record for that user?

Ans a. Alice didn't use a salt, which makes her scheme vulnerable to off-line dictionary attacks
Ans b. If the database is leaked (e.g., server compromise), the attacker can mount offline password guessing attacks. Such an attacker might be able to recover many of the users' passwords and not just those whose password hashes are listed in Google search.
Ans c. s; F(w; s) where s is a random salt chosen independently for each
user and where F is a slow cryptographic hash, e.g., SHA256 iterated many times, (F(x) = H(H(… (x) …)) where H is SHA256).

Prob 4. Alice would like to send a message M to Bob with confidentiality and integrity. Alice and Bob share symmetric keys k_1; k_2. Bob's public key is KB; we assume that Alice knows KB. Below, MAC is a secure message authentication code and H is a secure cryptographic hash. Consider the following two schemes:
S1: Alice sends $E_{k1}(M)$; $MAC_{k2}(M)$ to Bob
S2: Alice sends $E_{k1}(M)$;H(M) to Bob
Here $E_{k1}()$ is a secure stream cipher.
(a) Which scheme is better for confidentiality? Why?
(b) Which scheme is better for integrity? Why?

Ans a. S1 is better. S2 lets the attacker test a guess at M (given a guess g, compute H(g); if H(g) H(M), then he can conclude his guess was correct). Therefore, if there are only a few possibilities for M (maybe it is a 4-digit PIN), S2 allows the attacker to find M.

Ans b. S1 is better. With S2, if the attacker knows M, he can modify the ciphertext to turn it into a valid encryption of M_0. In particular, since E is a stream cipher, he can flip bits in $E_{k1}(M)$ to change it to an encryption of $E_{k1}(M_0)$, and then replace H(M) with $H(M_0)$. This attack is no possible against S1, since the MAC is secure and the attacker doesn't know the MAC key k2, so the attacker won't be able to predict $MAC_{k2}(M_0)$.