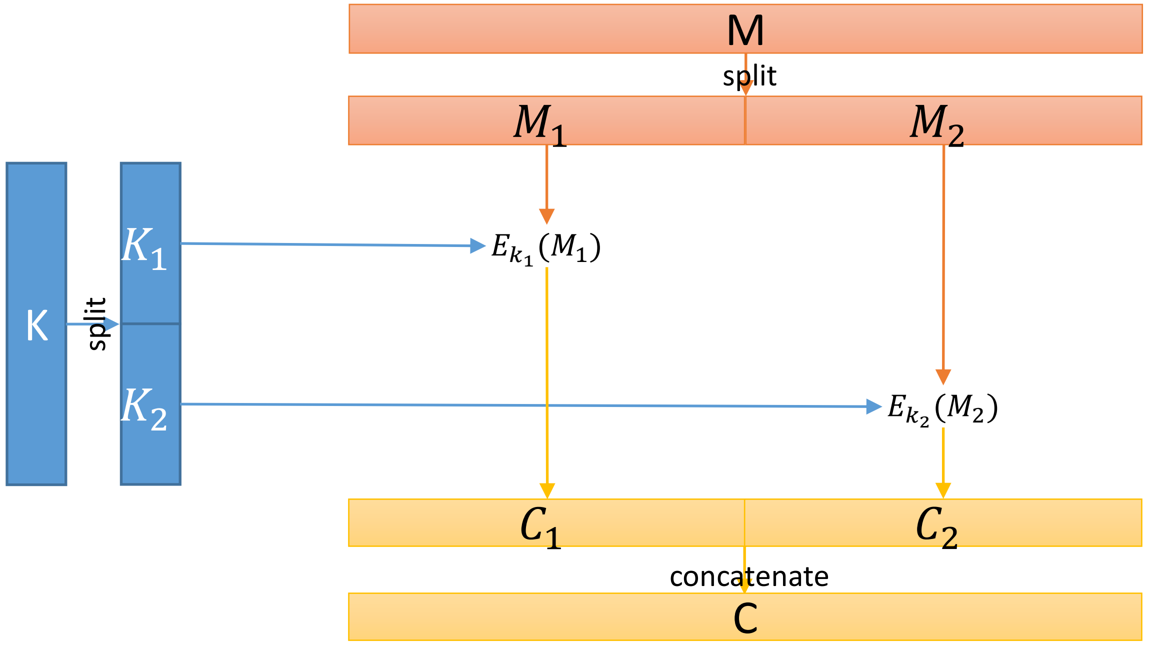
Tutorial 8

SOEN-321

Exercise 6 – Problem 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 M1 and M2 and the key K in two parts K1 and K2. The ciphertext C is then computed as EK1(M1)||EK2(M2). 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



Answer:

First, use brute force to recover K1. Let C=C1||C2. Then we know that C1 = EK1(M1), and we know M1;C1, so try all possibilities for K1 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

Exercise 6 – Problem 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?

Answer: Alice didn't use a salt, which makes her scheme vulnerable to off-line dictionary attacks

Exercise 6 – Problem 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.

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?

Answer: 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.

Exercise 6 – Problem 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.

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?

Answer: 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).