CS5430 Homework 2: Shared-key Cryptography

General Instructions. You may work alone or with one other person from our class on this assignment. If you do work with somebody then form a group on CMS and submit a single set of solutions.

***Note**: You are strongly urged to work with a partner. Don't just split the work. You will learn more and the assignment will be easier to finish if you both work together on all problems. We will help you find a partner, if needed. In the past, the average grade given to a pair working together has been significantly higher than the average grade given to individuals working alone.*

Due: Sept 26, 2021 11:59pm. No late assignments will be accepted.

Submit your solution using CMS. Prepare your solution as .pdf, as follows:

- Use 10 point or larger font.
- Submit each problem (as a separate file) into the correct CMS submission box for that problem.

Assume that the threat is a Dolev-Yao attacker who is also able to perform type-attacks.

Problem 1: xor-Encryption. For bit strings b and c, the bitwise exclusive-or operation $\oplus$ is defined as follows for n-bit strings:

$$b \oplus c = d \quad \text{if and only if} \quad \text{for } 1 \leq i \leq n\text{: } d[i] = (b[i] + c[i]) \bmod 2$$

where x[i] indicates the i^{th} bit of bitstring x.

Given an n-bit message m and an n-bit secret key k, we define the following encryption (E) and decryption (D) cryptographic operations. The notation k-F(x) is used to indicate the evaluation of a function F that takes an n-bit key k and an n-bit bitstring x as arguments.

k-E(m): $m \oplus k$
k-D(c): $c \oplus k$

(a) Show that k-D(k-E(m)) = m holds for all m and, therefore, D reverses any obfuscation that E introduces.

(b) An encryption function implements *perfect secrecy* if and only if ciphertext reveals nothing about plaintext. We can prove that an encryption function exhibits perfect secrecy by showing: for any ciphertext c (with length n) and any plaintext p (with length n) there exists some key k that, when used to encrypt p, produces c. That is, we are showing that any plaintext could be responsible for a given ciphertext (so the ciphertext is keeping the plaintext secret).

Show a function keygen(p,c) to generate the key for xor-Encryption of any plaintext p and ciphertext c. Show that your function suffices for proving perfect secrecy of k-E(p) or prove that no such function exists.

(c) xor-Encryption is defined above only for n-bit bit strings. We can also define xor-Encryption for an r-tuple of bit strings. It is the result of individually encrypting each of the fields in the r-tuple and separating them by commas. For example,

k-E("X , Y , Z"): "k-E(X) , k-E(Y) , k-E(Z)"

Here is a proposed protocol in terms of xor-Encryption that A could use to establish that A shares a key k_{AB} with B, despite the possibility of reflection attacks.

1. A → B: A, B, k_{AB} -E("A, B, r") for a fresh nonce r
2. B → A: A, B, k_{AB} -E("B, A, r")

But it is not secure. (i) Show an attack. (ii) What is the property that admits this attack with xor-Encryption but is not feasible with ordinary shared-key encryption.

(d) Here is a proposed protocol for A to communicate a secret message m to B. Principal A is the only one who knows bit string k_A; principal B is the only one who knows bit string k_B. Each of m, k_A, and k_B is n bits long.

1. A → B: k_A-E(m)
2. B → A: k_B-E(m1) where m1 = k_A-E(m) received in message 1
3. A → B: k_A-E(m2) where m2 = k_B-E(m1) received in message 2

(i) How can B recover m from the message received in step 3. Justify your answer by showing any calculations B performs with results from k_B-D(·) and k_B-E(·) invocations.

(ii) Is it possible for an eavesdropper to learn m, too? Explain why not, or give an attack (and give any calculations needed to prove that it works).

Problem 2. KDC protocol. The following protocol is intended to enable A and B to agree on a shared symmetric key K_{AB} for communicating with each other in a secure fashion. Assume that each principal P has been provisioned with a key K_P that is known only to P and to Key Distribution Center KDC.

1. A → KDC: A, B, r $\{ K_{AB}\} K_A$ where r is a fresh nonce and K_{AB} is a fresh key.
2. KDC → A: A, B $\{ r, K_{AB}\} K_B$
3. A → B: A, B $\{r, K_{AB}\} K_B$
4. B → A: $\{ r \} K_{AB}$
5. A → B: $\{ r+1\} K_{AB}$

(i) Upon receipt of message 1, how does the KDC determine which key to use to decrypt: "$\{X , K_{AB}\} K_A$"

(ii) Would there be any advantage to including r in the encrypted blob sent as part of step 1, so that the blob becomes:

"$\{r, K_{AB}\} K_A$"

If you believe there is an advantage, explain what kinds of attacks might become more difficult; if you believe there is no advantage, explain how attackers can circumvent the change.

(iii) The protocol as written above can be attacked. Give an attack that illustrates the vulnerability.

(iv) Can the attack in (iii) be prevented by replacing adding either "A" or "B" to the encrypted blob? If so, give a uniform substitution for X in the revised protocol below.

1. A → KDC: A, B, r $\{X , K_{AB}\} K_A$ where r is a fresh nonce and K_{AB} is a fresh key.
2. KDC → A: A, B $\{X , r, K_{AB}\} K_B$
3. A → B: A, B $\{X , r, K_{AB}\} K_B$
4. B → A: $\{ r \} K_{AB}$
5. A → B: $\{ r+1\} K_{AB}$