CS 161 Computer Security
Fall 2018 Nick Weaver HW 2

Due on 2018-09-28, at 11:59 pm

1 Problem 1: True or False

Solution
A. True
B. False
C. False
D. True
E. False
F. False
G. False
H. True
I. False

2 Finding Common Patients

(a) In the collaboration scheme described above, how should Bear Hospital compute x_i^* (as a function of x_i)? How should Tree Hospital compute y_i^* (as a function of y_i)? Specifically, your solution should define a function F that Bear Hospital will use to transform x_i into x_i^*, and if relevant, a function G that Tree Hospital will use to transform y_i into y_i^*.

Solution $E_k(x)$ =AES-CBC$_k(x)$ where IV is a fixed constant
$H(x)$ is the SHA-256 hash
$F_k(x) = E_k(H(x))$
The Bear Hospital could compute $y_i^* = F_k(y_i)$ where x_i and y_i are the names of the patients.

(b) Explain why requirement (1) is met by your solution

Solution Since Bear Hospital and Tree Hospital are doing equivalent operations, and all the blocks are deterministic, the outputs are determined if $x_i = y_j$, then $x_i^* = y_j^*$

(c) Explain why requirement (2) is met by your solution.

Solution Because SHA-256 is a pretty good hash and is collision resistant. This provides security. Also, because AES block ciphers are invertible (needed for decrypability), they cannot have collisions either.

(d) Explain why requirement (3) is met by your solution.

Solution Because we encrypt the hash of the plaintext, we can provide confidentiality. Not having an IV makes sure that we always get the same ciphertext if we encrypt 2 times. It becomes computationally extremely expensive to brute force this, and since Eve does not have the key, she cannot really attack the hash function.

3 Security Principles

Solution
A. 'x06jx0300kx8971x10x9ccj3¡x0b47x92x15x86Q]x1b1C1x81rNx0b9o68j$_{\mathbf{x}15*\mathbf{x}8ce1S5k\mathbf{x}1a4\mathbf{x}1d\mathbf{x}18\mathbf{x}99093r\mathbf{x}96\mathbf{x}}$

Solution C. Not it is not possible to find K, because we will never be able to see the private key. Even if somehow by extreme luck we found the message, we cannot reverse the encryption to reveal the key, in this way the key is safe unless the key is in the message, which is unlikely.

4 Why do RSA signatures need a hash?

(a) With this simplified RSA scheme, how can Bob verify whether S is a valid signature on message M? In other words, what equation should he check, to confirm whether M, S was validly signed by Alice? You dont need to justify your answer; just show the equation.

Solution Bob can check using essentially the same function, without the hash: $S^3 = M \mod n$

(b) Explain how Mallory can find M, S such that S will be a valid signature on M.

Solution Mallory can pick some S that is between 0 and n-1, M is then just S^3

(c) Help Mallory out: show how she can compute such an S prime

Solution We can figure out by what factor we need to multiply S by to make S' by taking 64 and finding the eth root of it, which is 4 in this case as n is 3. Then we divide S by 4 before exponentiating it. This way, when Sis cubed, we get S' being $S/64 = M$ this becomes $S = 64M$. So we can get Bob to pay through his nose in this fashion.

(d) Are your attacks in parts (b) and (c) possible against the real RSA signature scheme (the one that includes the cryptographic hash function)? Why or why not?

Solution No, partially because that would be hilarious if it was possible, but because the cryptographic hashing basically does not allow us to see M at all, so for example in part B, we would not be able to pick and S so that Bob can verify it as Alice, since the hash function is secure. This makes Bob's decryption process significantly less transparent, and doesn't allow us to easily imitate Alice. Also in C. we would not be able to manipulate M as easily because the hash function is not able to be operated upon like an integer, so there is no direct relationship between S and M, so manipulating S does not guarantee we can manipulate M.

5 New Block Cipher Mode

(a) Given $(C_0, C_1, ..., C_n)$ and the key k, explain how to recover the original message (P1, . . . , Pn).

Solution Since XOR is the inverse of itself, we XOR both sides to get $P_i = E_k(C_{i-1}) \oplus P_i$

(b) Is NBC encryption parallelizable? How about decryption? Provide a short justification for each.

Solution
Encryption: Not parallelizable because this function is similar to Cipher Feedback. Because we need the previous encrypted cipher for subsequent encryptions, this must be done serially.
Decryption: Parallelizable because like cipher feedback, be already know all the ciphertexts, and so we can feed them into decryption all at the same time and extract the plaintexts in parallel. Hex is size j + 1 (for null byte at end) and tmp is size j.

(c) As we saw in discussion, CBC mode is vulnerable to a chosen plaintext attack when the IV which will be used to encrypt the message is known in advance. Is NBC vulnerable to the same issue?

Solution No, because we encrypt the IV, even if IV is leaked, an attacker would be unable to find $E_k(IV)$

(d) Say that Alice means to send the message $(P_1, ..., P_n)$ to Bob using NBC mode. By accident, Alice typos and encrypts $(P_1 1, ..., P_n)$ instead (i.e., she accidentally flips the last bit of the first block).
True or False: after Bob decrypts the resulting ciphertext, every block after the first is incorrect. Explain your answer

Solution False, because we are just encrypting it like this. Since this is not cipher block chain, every block after the flip will not necessarily be incorrect. In this case we will just have encrypted this extra bit flip, and we will get back this flip when we decrypt, because we encrypt before XOR.

(e) Alice encrypts the message $(P_1, ..., P_5)$. Unfortunately, the block C_3 of the ciphertext is lost in transmission, so that Bob recieves $(C_0, C_1, C_2, C_4, _C 5)$. Assuming that Bob knows that he is missing C_3, which blocks of the original plaintext can Bob recover?

Solution Bob can recover P_1, P_2 and P_5, because he only needs the previous ciphertext block in order to decrypt the current one.