Pham Hoang Long Do
NetID: 2995060

Midterm

CSE 3400
Ghada Almashaqbeh

Problem 1

Problem 1.1

- This is **not a secure** PRF for the following reason:
  1. Since the attacker has the access to the oracle, he can submit as many queries as he wants.
  2. The attacker submits a queries and gets back the ciphertext $F_k(x)$ which has the length of $3n$ bits.
  3. He can divide the ciphertext into 3 blocks. Each block has the length n bits.
  4. The first block is $y_1 \oplus 0^n$. He can uses this block to xor with 0^n again to retrieve the first part of the key. He knows n since he is the one who generates the message.
  5. Same thing for the second block $y_2 \oplus 1^n$. He use this block to xor with 1^n again to retrieve the second part of the key. Again, he knows how long his message is.
  6. Lastly, he uses the last block $y_3 \oplus x$ to xor with his own message to retrieve the third part of the key. He knows his original message so it should not be the problem.
  7. After retrieve y_1, y_2, y_3 he can concatenate them together to get y which is the output of $G(k)$.
  8. Since G is deterministic and we keep reusing key k whenver we evaluate a new input. Therefore, y is always deterministic.
  9. The success chance for this attacker to tell which one is a PRF and which one is truly random is 1, and the probability to regconize which one is truly random is $\frac{1}{3n}$

$$\epsilon_{A,F}^{PRF}(n) = 1 - \frac{1}{3n}$$

- $1 - \frac{1}{3n}$ is non-negligible as n sufficient large.

Problem 1.2

- This is **a secure PRF** in a sense that:
  1. We use t as a seed to a PRG, and we always generate new seeds everytime we invoke the security scheme. Therefore, the output of function $G(t)$ is **not deterministic**, pseudo random and indistinguishable from a truly random function.
  2. $F_k(x)$ is a PRF.
  3. We have a concatenation between two pseudo random output functions. Therefore, the result is pseudo random as well. Hencer, this is a secure PRF.

Problem 1.3

- This is **a secure PRF** in a sense that:
  1. This encryption scheme only has two options for shifting. Either shift to the right by 1 or not shift at all.
  2. If the $lsb(x) = 0$ which means that we not shift at all. Then the out put of $F_k^{'}(x)$ is the same as the output of $F_k(x)$

3. If the $lsb(x) = 1$ which means that we shift the output of a PRF to the right by 1. Then the result still a PRF. The Attacker doesn't has any advantage to know the key or gain any information about the output of $F_k(x)$. Therefore, It's a secure PRF

Problem 2

Problem 2.1

$E_k(m) = (r, r', t, F_k(G(r \oplus t)) \oplus m_1, F_k(r') \oplus m_2)$. Let assign $c_1 = F_k(G(r \oplus t)) \oplus m_1$ and $c_2 = F_k(r') \oplus m_2$. We have the decryption algorithm as follow:

$$D_k(c) = (r, r', t, c_1 \oplus F_k(G(r \oplus t)), c_2 \oplus F_k(r')) = (r, r', t, m_1, m_2) \text{ where } m = m_1||m_2$$

- This is **a secure** encryption scheme in a sense that.
  1. The attacker can submit two queries(m_0 and m_1) of his choice that sastify $0, 1^{2n}$
  2. Challenger picks a bit b at random and encrypt m_b and return the attacker the ciphertext $c^* = ENC_k(m_b)$
  3. The attacker receives $c^* = c_0^*, c_1^*, c_2^*, c_3^*, c_4^*$
  4. The attacker can compute $c_0^* \oplus c_2^*$ together and put it into function G since G is public. He can put it to F. However, he won't know if his own calculated result will match the oracle result or not since he doesn't know the key. Same argument goes for c_4^*.
  5. He can't submit same queries again since r and r' are random and freshly generated for each query. So, this encryption scheme **deterministic**
- Therefore this encryption scheme does not give out the key and randomized. Therefore, it's secure.

Problem 2.2

$E_k(m) = (r, m \oplus F_k(m \oplus r))$. The decryption algorithm is:

$$D_k(r, c) = (r, c \oplus F_k(r \oplus c) \oplus r)$$

- This is **a secure** encryption scheme in a sense that:

1. The attacker can submit two queries(m_0 and m_1) of his choice that sastify $0, 1^n$
2. Challenger picks a bit b at random and encrypt m_b and return the attacker the ciphertext $c^* = ENC_k(m_b)$
3. The attacker receives $c^* = c_0^*, c_1^*$
4. He can use c_0 to xor with c_1 or whatever he wants. However, the function: $m \oplus F_k(m \oplus r)$ is a randomized function. Even if the attacker try to submit the same query over again. The output will always different.
5. This encryption scheme does not give out the key in it algorithm.

- Therefore this is a secure IDN-CPA.

Problem 2.3

- Yes. Bob's claim is **correct** since at the worst case scenario when the CPA attacker tries to squeeze advantages as much as he can (i.e he can wait until bob used his key 255th times and then ask for the challenge). Now there is only 1 bit left for the key, so the attacker still has a significant chance of guessing the correct key which is 50%. He doesn't improve his guessing chance by any percent since he always has 50% to guess $b*$ which is either 0 or 1 since the beginning (when Bob just started using the key).

Problem 3

Problem 1

Problem 1.a

- This is **not a secure** MAC in a sense that:
  1. an attacker can ask the oracle the tag for $m = m_0 \parallel m_1$ and receive the tag $y||z$ back in which $y = P_{k_0}(m_0)$ and $z = F_{k_1}(m_1)$
  2. He, then, divides the tag from the oracle into two blocks of n bits.
  3. Now, he swap his original message along with its tag. $m^{'} = m_1 \parallel m_0$ and submit a new tag $tag^{'} = z \parallel y$. This is a valid fogery since $m^{'} \neq m$ and $tag^{'}$ is a valid tag.
  4. The attacker will have a probability of 1 in winning this game, so the advantage is $1 - \frac{1}{2^{2n}}$ which is non-negligible.

Problem 1.b

- This is **a secure** MAC since the attacker can not compute $F_{k_0}(m)$ on his own without the key k_0. Same thing for $G(F_{k_1}(m))$, without the key he can't compute the result on his own. The two outputs of these function are pseudo random. Therefore, the attacker's advantage is negligible.

Problem 2

1. Mal can sit on the channel for a few days and observe the cipher pattern that being sent over (which is fixed). Then, Mal can change the recipient of the file or swap file to another doctor. In this case, Mal happens to be a **CTO** attacker when he can only have access to cipher text.
2. Yes. The new approach will change my answer since we are now using EtA which is proven to secure agaisnt CCA attacker, and MAL is just a CTO attacker which is a weaker type of attack. Therefore, using EtA will be secure agaisnt Mal.However, Mal still has the ability to relay the message which will create an overhead for the hospital.
3. This will impact security since the hospital will not know for sure who is the one that sent the message or if the message is tampered yet since Mal can still drop the ciphertexts were produced by the hashfunction. In addition, the message is **not long enough** and the Domains(doctor's name and files) are **too short** for the hash function work efficiently.
4. If Alice used the counter, it's like timestamped her messages which can help agaisnt relay attacks. The hospital can verify that the message was sent recently and is not a delayed or

replayed message from a previous communication session.

Problem 4

1. True
2. False
3. True
4. False
5. False