Practice Problem Set 1 Solutions

March 19, 2017

Exercise 1. (Encryption: From Midterm Spring 2015) Let f be a pseudorandom function (PRF) taking key k and input x and producing output $f_k(x)$.
The key k, input x and output $f_k(x)$ all have bit length n.

The following is a **CPA-secure** symmetric encryption scheme:
To encrypt n-bit message m under key k, select fresh random n-bit string r and output

$$r||(f_k(r) \oplus m)$$

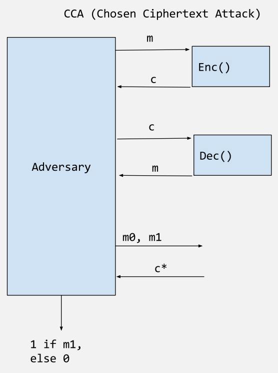
(The symbol $\oplus$ is the bitwise XOR; recall that $a \oplus a \oplus b = b$.)
(The symbol $||$ denotes concatenation.)

1. **(2 points).** Write down the decryption algorithm.

Solution: First separate out r, we'll call the cipher text without r, c. Compute $f_k(r)$, xor this with c to compute $m = f_k(r) \oplus c$.

2. **(2 points).** Write down the definition of **CCA-secure symmetric encryption**.

Solution: Adversary is given $m = Dec(c)$ and $c = Enc(m)$ oracles. Adversary constructs two messages m_0, m_1 such that $m_0 \neq m_1$ and is given one cipher text c^* that corresponds to one of those messages. Adversary wins if he can distinguish which message corresponds to which c^* with greater than $\frac{1}{2}$ probability. He cannot query $Dec()$ for c^*.



3. **(4 points).** This encryption scheme is **CPA-secure**.
   Prove that this scheme is **NOT CCA-secure**.

   Solution: Adversary chooses m_0, m_1 and gets back c^*, then flips the last bit of c^* to construct $c^{*'}$. He then queries the decryption oracle to get $m = Dec(c^{*'})$. If m is m_0 with the last bit flipped, he guesses m_0. If m is m_1 with the last bit flipped, he guesses m_1. Notice that he wins because this attack never queried the decryption oracle for c^*.

Exercise 2. Does it suffice to use CPA-secure encryption in the following scenario? Why or why not?

A user A wants to send his password to a server B, and suppose A and B have a shared symmetric key k. The password is encrypted under key k.

If the password is correct, then A receives the message "OK" encrypted under k from B, and then is allowed to interact with the server B, downloading webpages and sending and receiving other information.

Otherwise, then A receives the message "Fail" encrypted under k from B, and then communication stops.

Solution: No it does not. The adversary can choose ciphertexts and then learn if they correspond to valid plaintexts or not. Notice that the adversary is CHOOSING ciphertexts. Thus, we need CCA security (chosen ciphertext attack security) in this case.

Exercise 3. (MACs & Encryption schemes.)

Let f be a pseudorandom function (PRF). f takes in a key of length n and an input of length $2n$ and produces an output of length n (*i.e.,* it is length shrinking). In the question below, the symbol $||$ means concatenation and the symbol $\oplus$ is a bit-wise XOR and the symbol $|m| = n$ means the bitstring m has length n bit.

- Prove that the following "MAC" for messages of length $4n$ is an insecure MAC.

  The shared key is a random bitstring $k \in \{0,1\}^n$. To authenticate a message $m1||m2$ where $|m1| = |m2| = 2n$, compute the tag

$$f_k(m1)||f_k(f_k(m2)||0^n)$$

  Solution: Ask the MAC oracle for the mac of $m_1||m_2$ to get $t_1 = MAC(m_1||m_2)$ and for the MAC of $m_3||m_4$ to get $t_2 = MAC(m_3||m_4)$. Take the first n bits of t_1 and the last n bits of t_2 combine them to make a new tag t^*. Let $m^* = m_1||m_4$. Output m^*, t^*. t^* will pass verification on message $m_1||m_4$ so, you have broken the MAC security definition, because you have found a valid MAC on a message that has never been queried to the MAC oracle.

- The following is a CPA-secure encryption scheme. The shared key is a random bitstring $k \in \{0,1\}^n$. To encrypt a message m of length n bits, choose a random $2n$-bit string r and output the ciphertext

  $$r||(f_k(r) \oplus m)$$

  Now suppose we slightly modify the encryption scheme above, as follows. The shared key is a random bitstring $k \in \{0,1\}^n$. To encrypt a message m of length $2n$ bits, choose a random n-bit string r and output the ciphertext.

  $$r||(f_k(m) \oplus r)$$

  Explain why this is not an encryption scheme.

  Solution: There is no way to decrypt the cipher text, $f_k(m)$ is a PRF output that can only be computed if both parties know k and m. Since the receiving party does not know m, it is impossible to decrypt.

- Now we slightly modify the encryption scheme again. We use a collision resistant hash function H that maps $2n$-bit string to an n bit strings. The key is a random bitstring $k \in 0, 1^n$.

  To encrypt a message m of length n, choose a random $2n$-bit string r and output the ciphertext

  $$r||(H(r) \oplus m \oplus k)$$

  Prove that this is not a CPA secure encryption scheme.

  Solution: The adversary can compute $H(r)$ easily since r is included in the cipher text. Since $a \oplus a \oplus b = b$, the adversary can remove $H(r)$ from the cipher text by XOR it with $H(r)$ to obtain the value $c = m \oplus k$. You may remember this as the one-time-pad, with the key k reused for every encryption. You are only supposed to use the key to a one-time pad once. (hence the name!) to break it generate a plaintext m such that $m = \{0\}^n$, and query the encryption oracle to get a ciphertext $r||H(r) \oplus k \oplus m$. Now pull out the nonce r from the ciphertext and XOR the remaining part with $H(r)$, and you get $k \oplus m$ but since m is all zeros this is just k. You now know k and can decrypt all future messages. Ha.

Exercise 4. In Lab2 (the minilab), you were asked to prove that AES in CTR mode cannot satisfy the definition of CCA2-secure encryption. In other words, present an algorithm for an Adversary that wins the CCA2 security game described in the lab handout.

Now, suppose that you used an *authenticated* version of AES in CTR mode. That is, you secret keys are (k_1, k_2), and to encypt a message m you first take $c = Enc_{k_1}(m)$, where Enc is encryption using AES in CTR mode, and then you take $t = MAC_{k_2}(c)$ where MAC is a secure MAC algorithm. You then output the ciphertext (c, t).

- Write down the decryption algorithm.

Solution:

  - Check that the MAC verifies ($\mathsf{Ver}(k_2, t, c) = 1$).
  - If the MAC verifies, decrypt the message; return $m = \mathsf{Dec}(k_1, c)$.
  - If the MAC does not verify, return $\perp$.

- Explain why the attack you presented in Lab2 (*i.e.*, the algorithm for an Adversary that wins the CCA2 security game) no longer works.

Solution: The attack from lab2 consisted of asking the decryption oracle for a decryption of a ciphertext related to the challenge ciphertext. However, now, as soon as you try to derive a related ciphertext, the decryption oracle will return $\perp$ on that ciphertext, because in order to get anything other than $\perp$, you must make the MAC verify, breaking MAC security. Thus, you can learn nothing from the decryption oracle, because it will just give $\perp$ all the time!