Quiz4

Department: 資工 14 Student ID: 110550091 Name: 吳承瑀

1. Data compression is often used in data storage and transmission. Suppose you want to use data compression in conjunction with encryption. Does it make more sense to:
    a) Compress then encrypt.
    b) The order does not matter – either one is fine.
    c) The order does not matter – neither one will compress the data.
    d) Encrypt then compress.

    The answer is (a). Most compression techniques take advantage of the reoccurrence of certain letters or words to reduce data size. For example, Huffman Coding is one way to reduce data size by having the most occurred letter the shortest length of code. If we encrypt the data first, then the data will become more of a random distribution of letters in which loses the crucial criteria of data compression. Thus, the order does matter. If we compress then encrypt, we can achieve both effective data compression and encryption.

2. Let G: $\{0, 1\}^n$ be a secure PRG. Which of the following is a secure PRG (there is more than one correct answer):
    a) G'(k) = G(k) || 0 (Here || denotes concatenation)
    b) G'(k) = G(k) || G(k) (Here || denotes concatenation)
    c) G'(k) = G(0)
    d) G'(k) = G(k $\oplus$ 1^1)
    e) G'(k) = G(k) $\oplus$ 1^n
    f) G'(k) = reverse(G(k))

    (a) Is incorrect because concatenating a zero at the end makes it distinguishable from a random seed of 1s and 0s. Not only does the chance of 0s become slightly higher, always having a zero at the end makes it slightly noticeable as well.
    (b) Is incorrect because the reoccurrence of a completely same sequence doesn't seem random.
    (c) Is incorrect because the variable *k* is supposed to be a random seed. Always using a 0 as the variable value will not yield random results.
    (d) (e) (f) are correct because the result does not change length, nor does it change the randomness of the data.

3. Let G: K → $\{0, 1\}^n$ be a secure PRG. Define $G'(k_1, k_2) = G(k_1)\text{^}G(k_2)$ where ^ is the bit-wise AND function. Consider the following statistical test A on $\{0, 1\}^n$. A(x) outputs LSB(x), the last significant bit of x. What is $Adv_{PRG}[A, G']$? You may assume that LSB[G(k)] is 0 for exactly half the seeds k in K.

   The answer is 0.25. Since the chance of $LSB[G(K_1)]$ be a one is 0.5 and so does $LSB[G(K_2)]$, this means that after the bit-wise AND function, the chance of $LSB[G'(K_1, K_2)]$ becomes 0.5*0.5 = 0.25.

4. Let E, D be a one-time semantically secure cipher with key space $K = \{0, 1\}^l$. A bank wishes to split a decryption key $\in \{0, 1\}^l$ into two pieces p_1 and p_2 so that both are needed for decryption. The piece p_1 can be given to one executive and p_2 to another so that both must contribute their pieces for decryption to proceed. The bank generates random k_1 in $\{0, 1\}^l$ and sets $k' \leftarrow k \oplus k_1$. Note that $k1 \oplus k1' = k$. The bank can give k_1 to one executive and k_1' to another. Both must be present for decryption to proceed since, by itself, each piece contains no information about the secret key k (note that each piece is a one-time pad encryption of k). Now, suppose the bank wants to split k into three pieces p_1, p_2, p_3 so that any two of the pieces enable decryption using k. This ensures that even if one executive is out sick, decryption can still succeed. To do so the bank generates two random pairs (k_1, k_1') and (k_2, k_2') as in the previous paragraph so that $k_1 \oplus k_1' = k_2 \oplus k_2' = k$. How should the bank assign pieces so that any two pieces enable decryption using k, but no single piece can decrypt?
   a) $p_1 = (k_1, k_2)$, $p_2 = (k'_1)$, $p_3 = (k'_2)$
   b) $p_1 = (k_1, k_2)$, $p_2 = (k_2, k'_2)$, $p_3 = (k'_2)$
   c) $p_1 = (k_1, k_2)$, $p_2 = (k'_1, k_2)$, $p_3 = (k'_2)$
   d) $p_1 = (k_1, k_2)$, $p_2 = (k'_1, k'_2)$, $p_3 = (k'_2)$
   e) $p_1 = (k_1, k_2)$, $p_2 = (k_1, k_2)$, $p_3 = (k'_2)$

   Answer is (c) because $k = k_1 \oplus k_1'$ when p_1 and p_2 pairs, $k = k_1 \oplus k_1'$when p_1 and p_3 pairs, and $k = k_1 \oplus k_1'$ when p_2 and p_3 pairs. (a) has no key when p_2 and p_3 pairs. (b) has a key with p_2 itself. (d) has no key when p_2 and p_3 pairs. (e) has no key when p_1 and p_2 pairs.