COL759 Assignment 1

Anshik Sahu (2021CS10577)

August 2023

Part A

1: Perfect Two-time Security

(a)

Let the Adversary be defined as follows:

- A gives (m0,m0) and (m1,m2) to the Challenger and receives (ct0,ct1), where m1 and m2 are distinct.
- If ct0 equals ct1, it guesses 0 else it guesses from 0,1 uniformly.

If the Encryption is not randomized,

- if b=0, ct0=Enc(k,m0)=ct1. Hence b'=0.
- if b=1, ct0 and ct1 are distinct due to the correctness of the encryption. hence b'=0 or 1 with equal probability.

Thus, for a non randomized encryption,

$$Pr_k[A \text{ winning } \epsilon\text{-perfect two-time security game}] = \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} = \frac{3}{4}$$

Hence, a non randomized encryption is not perfect two-time secure.
For a randomized encryption, led the randomness be denoted by r0 and r1.

- if b=0,
$$Pr_k[r_1 = r_2] = \frac{1}{2^{l_3}}$$
Thus,
$$Pr_k[A \text{ guesses } 0 \mid b = 0] = \frac{1}{2^{l_3}} + \frac{1}{2}\left(1 - \frac{1}{2^{l_3}}\right) = \frac{1}{2} + \frac{1}{2^{l_3+1}}$$
- if b=1,
$$Pr_k[A \text{ guesses } 1 \mid b = 1] = \frac{1}{2}$$

Hence,

$$Pr_k[A \text{ winning } \epsilon\text{-perfect two-time security game}] = \frac{1}{2} \cdot \left(\frac{1}{2} + \frac{1}{2^{l_3+1}}\right) + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2} + \frac{1}{2^{l_3+2}}$$

Thus, even for a randomized encryption, perfect two-time security is not possible with symmetric key.

(b)

We can define the following encryption scheme:
(Here $k\epsilon\{0,1\}^n$, $r \leftarrow \{0,1\}^n$, $m\epsilon\{0,1\}^n$)

- Encryption(k,m):
  - Choose a random r.
  - Calculate $ct = m \oplus H_k(r \oplus k)$.
  - Output (r,ct).
- Decryption(k,r,ct):
  - Calculate $m = ct \oplus H_k(r \oplus k)$.
  - Output m.

To prove the security of this encryption we can create the following hybrid worlds for the ϵ-perfect two-time security game:

- World 0:
  - Challenger chooses $r1 \mid\mid r2 \leftarrow \{0,1\}^{2n}$.
  - Challenger sends $(r1, r2, m_{00} \oplus H_k(r1 \oplus k), m_{01} \oplus H_k(r2 \oplus k))$.
  - A outputs b'.
  $$Pr[A \text{ outputs } 0] = p$$
- Hybrid World 1a:
  - Challenger chooses $r1 \mid\mid r2 \mid\mid r3 \mid\mid r4 \leftarrow \{0,1\}^{4n}$.
  - Challenger sends $(r1, r2, m_{00} \oplus H_k(r3), m_{01} \oplus H_k(r4))$.
  - A outputs b'.
  $$Pr[A \text{ outputs } 0] = p1a$$
- Hybrid World 2a:
  - Challenger chooses $r1 \mid\mid r2 \mid\mid r3 \mid\mid r4 \leftarrow \{0,1\}^{4n}$.
  - Challenger sends $(r1, r2, m_{00} \oplus r3, m_{01} \oplus r4)$.
  - A outputs b'.
  $$Pr[A \text{ outputs } 0] = p2a$$
- Hybrid World r:
  - Challenger chooses $r1 \mid\mid r2 \mid\mid r3 \mid\mid r4 \leftarrow \{0,1\}^{4n}$.
  - Challenger sends $(r1, r2, r3, r4)$.
  - A outputs b'.
  $$Pr[A \text{ outputs } 0] = pr$$
- Hybrid World 2b:
  - Challenger chooses $r1 \mid\mid r2 \mid\mid r3 \mid\mid r4 \leftarrow \{0,1\}^{4n}$.
  - Challenger sends $(r1, r2, m_{10} \oplus r3, m_{11} \oplus r4)$.
  - A outputs b'.
  $$Pr[A \text{ outputs } 0] = p2b$$
- Hybrid World 1b:
  - Challenger chooses $r1 \mid\mid r2 \mid\mid r3 \mid\mid r4 \leftarrow \{0,1\}^{4n}$.

  - Challenger sends $(r1, r2, m_{10} \oplus H_k(r3), m_{11} \oplus H_k(r4))$.
  - A outputs b'.

$$Pr[A \text{ outputs } 0] = p1b$$

- World 1:
  - Challenger chooses $r1 \mid\mid r2 \leftarrow \{0,1\}^{2n}$.
  - Challenger sends $(r1, r2, m_{10} \oplus H_k(r1 \oplus k), m_{11} \oplus H_k(r2 \oplus k))$.
  - A outputs b'.

$$Pr[A \text{ outputs } 0] = p'$$

As H is pairwise independent and k and r are unrelated, $\mid p - p1a \mid$ and $\mid p' - p1b \mid$ are negligible. As H is pairwise independent and r1 is not related to r3 and r2 is not related to r4, $\mid p1a - p2a \mid$ and $\mid p1b - p2b \mid$ are negligible. As r3 and r4 are randomly chosen p2a=pr. Similarly, p2b=pr. Hence, we can conclude that $\mid p - p' \mid$ is negligible. Therefore the encryption is two-time secure against probablistic-polynomial time attacks.

2: Secure/Insecure PRGs and PRFs

(a)

i. Theorem 1: For a secure PRG, if b denotes the ith bit

$$| Pr[b = 1] - \frac{1}{2} | \text{ is negligible for all i}$$

Proof: If the theorem is not true,

$$\exists \ i \ \ s.t. \ \ Pr[b = 1] = \frac{1}{2} + \alpha \text{ , where } \alpha \text{ is non-negligible}$$

Thus, we can define an adversary for the PRG game which guesses 1 when the ith bit is 1. As this adversary has non-negligible advantage α, PRG cannot be secure. Hence by contradiction, the Theorem is correct. Using Theorem 1, for the ith bit b of G'_n

$$Pr[b = 1] = \frac{1}{4} + \epsilon \text{ where } \epsilon \text{ is negligible}$$

Hence, using the inverse of Theorem 1, G' is not PRG secure. An attacker A can choose any index i and guess random if ith bit is 1 else guess 0. Advantage of A is,

$$Pr[A \text{ guess } 0 \mid \text{ not random }] - Pr[A \text{ guess } 0 \mid \text{ random }] = \frac{1}{4} - \epsilon$$

As the attacker has non negligible advantage, it can break G'.

ii. We can prove that G' is secure by creating the following hybrid worlds:

- World 0:
  - Challenger chooses $s1 \mid\mid s2 \leftarrow \{0,1\}^{2n}$.
  - Sends $G(s1)\hat{}G(S2)$.
  - A outputs b'.

$$Pr[A \text{ outputs } 0] = p$$

- Hybrid World 1:
  - Challenger chooses $s1 \leftarrow \{0,1\}^n$ and $r \leftarrow \{0,1\}^{3n}$.
  - Sends $G(s1)\hat{}r$.
  - A outputs b'.

$$Pr[A \text{ outputs } 0] = p1$$

- Hybrid World 2:
  - Challenger chooses $r0 \leftarrow \{0,1\}^{3n}$ and $r1 \leftarrow \{0,1\}^{3n}$.
  - Sends $r0\hat{}r1$.
  - A outputs b'.

$$Pr[A \text{ outputs } 0] = p2$$

- World 1:
  - Challenger chooses $r \leftarrow \{0,1\}^{3n}$.
  - Sends r.
  - A outputs b'.

$$Pr[A \text{ outputs } 0] = p'$$

As G is secure, $\mid p - p1 \mid$ and $\mid p1 - p2 \mid$ are negligible. Thus, $\mid p - p2 \mid$ is also negligible. As the XOR of two random numbers id also random, Hybrid World 2 = World 1, $p2 = p'$. Therefore, $\mid p - p' \mid$ is negligible. As World 0 and World 1 denote the cases in which the challenger chooses b=0 and b=1 respectively in the PRG game for G', the advantage of any adversary for G' is negligible. Hence G' is PRG secure.

(b)

i. We can define the following adversary for the PRF game:

- Adversary gives input (x0,x0) to the challenger and receives cipher text (ct0,ct1).
- If ct0 equals ct1 it guesses 0 else guesses 1.

Thus,

$$Pr_k[Awins] = 1 - \frac{1}{2^{n+1}}$$

As the adversary has almost complete advantage, F' is not secure.

ii. We can prove that F' is secure by the following contradiction. Assume F' is not secure, thus there exists an adversary A which breaks the security of F' in polynomial time. Using A, we can construct the following reduction B:
For each query,

- Select a random input x from $\{0,1\}^n$ and give it to the challenger.
- Let ct be the value received from the challenger. Calculate $ct' = ct \oplus x$ and pass (x,ct') to A.
- Let b be the output from A. Output your answer as b.

By construction,

$$Pr_k[\text{B wins for } G] = Pr_k[\text{A wins for } G']$$

As A breaks the security of G',

$$Pr_k[\text{B wins for } G] - \frac{1}{2} \text{ is non negligible}$$

Hence, B breaks the security of G. This leads to a contradiction as G is PRG secure.
Therefore, G' is PRG secure.

3: PRG Security does not imply Related-Key-PRG Security

(a)

We can describe a new function $G' = \{G'_{n+1} : \{0,1\}^{n+1} \rightarrow \{0,1\}^{3n}\}_{n \epsilon N} \cap \{G'_1 : \{0,1\} \rightarrow \{0,1\}^2\}$ as follows:

$$G'_{n+1}(s||b) = \begin{cases} G_n(s) \oplus \{0\}^{3n} & \text{if } b = 0 \\ G_n(s) \oplus \{1\}^{3n} & \text{if } b = 1 \end{cases}$$

$$G'_1(0) = 01 \text{ and } G'_1(1) = 10$$

Note: the cases for G'_1 are not considered further as n is too small

(b)

We can define the following adversary A which uses key related attack against G':

- It queries the challenger three times and receives the cipher texts ct0, ct1 and ct2 respectively.
- If $ct0 \oplus ct1 = \{1\}^l$ or $ct1 \oplus ct2 = \{1\}^l$, where l is length of cipher, it outputs 0 else 1.

To calculate the winning probability of A, we consider the following cases:

- If b=0,
$$Pr[A \text{ guesses } 0] = 1$$
- If b=1,
$$Pr[A \text{ guesses } 1] = 1 - \frac{1}{2^{2l}}$$

Thus,

$$Pr[A \text{ wins}] = 1 - \frac{1}{2^{2l+1}}$$

As G' is length expanding, $l > n$. Hence, $\frac{1}{2^{2l+1}}$ is negligible.
Therefore,

$$Pr[A \text{ wins}] \approx 1$$

(c)

We can prove that there exists a reduction B which breaks the security of G' by creating the following hybrid worlds:

- World 0:
  - Challenger chooses $s \leftarrow \{0,1\}^{n+1}$.
  - Sends $G'(s)$.
  - A outputs b'.
$$Pr[A \text{ outputs } 0] = p$$
- World a:
  - Challenger chooses $s \leftarrow \{0,1\}^n$.
  - Sends $G(s) \oplus \{0\}^{3n}$.
  - A outputs b'.
$$Pr[A \text{ outputs } 0] = p_a$$
- World b:
  - Challenger chooses $s \leftarrow \{0,1\}^n$.

  - Sends $G(s) \oplus \{1\}^{3n}$.
  - A outputs b'.

$$Pr[A \text{ outputs } 0] = p_a$$

- Hybrid World a:
  - Challenger chooses $r \leftarrow \{0,1\}^{3n}$.
  - Sends $r \oplus \{0\}^{3n}$.
  - A outputs b'.

$$Pr[A \text{ outputs } 0] = p_{ha}$$

- Hybrid World b:
  - Challenger chooses $r \leftarrow \{0,1\}^{3n}$.
  - Sends $r \oplus \{1\}^{3n}$.
  - A outputs b'.

$$Pr[A \text{ outputs } 0] = p_{hb}$$

- World 1:
  - Challenger chooses $r \leftarrow \{0,1\}^{3n}$.
  - Sends r.
  - A outputs b'.

$$Pr[A \text{ outputs } 0] = p'$$

By construction,

$$p = \frac{1}{2}(p_a + p_b) \text{ and } p' = p_{ha} = p_{hb}$$

As A breaks the security of G' $\mid p - p' \mid$ is non negligible. Therefore either $\mid p_a - p_{ha} \mid$ or $\mid p_b - p_{hb} \mid$ is non-negligible.
If $\mid p_a - p_{ha} \mid$ is non-negligible, we can define B for PRG security game as:

- Queries and receives ct.
- Calculates and sends $ct \oplus \{0\}^{3n}$ to A.
- A outputs b'.

As A can differentiate between World a and Hybrid World a with non-negligible probability, B has non-negligible winning probability.
If $\mid p_a - p_{ha} \mid$ is non-negligible, we can define B for PRG security game as:

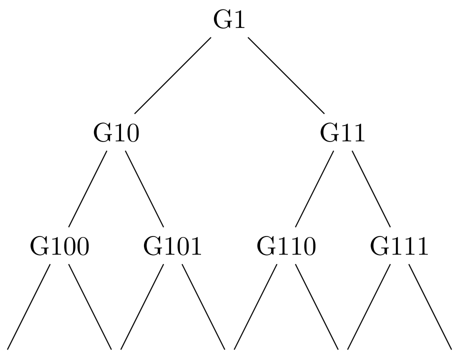
- Queries and receives ct.
- Calculates and sends $ct \oplus \{1\}^{3n}$ to A.
- A outputs b'.

As A can differentiate between World b and Hybrid World b with non-negligible probability, B has non-negligible winning probability. In either case, there exists a reduction B which breaks the PRG security of G. Hence Proved.

4: Constructing PRFs from PRGs

(a)

F can be represented as the following tree.



The height of the tree is log(n).
To prove the security of F we can create the following hybrid worlds:

- World 0:
  - Challenger chooses $s \leftarrow \{0,1\}^n$.
  - Sends $G'_{log(n)}(\underbrace{G'_{log(n)-1}(\dots G'_1(s))\dots)}_{\text{log(n) times}}$
  - A outputs b'.

  $$Pr[A \text{ outputs } 0] = p$$

  Here,

  $$G'_i(s1||s2) = \begin{cases} G(s1) & \text{if } x[i] = 0 \\ G(s2) & \text{if } x[i] = 1 \end{cases}$$

- Hybrid World i for $0 \leq i \leq log(n)$:
  - Challenger chooses $r \leftarrow \{0,1\}^n$.
  - Sends $G'_{log(n)}(\underbrace{G'_{log(n)-1}(\dots G'_1(r))\dots)}_{\text{i times}}$
  - A outputs b'.

  $$Pr[A \text{ outputs } 0] = p_{hi}$$

- World 1:
  - Challenger chooses $r \leftarrow \{0,1\}^n$.
  - Sends r.
  - A outputs b'.

  $$Pr[A \text{ outputs } 0] = p'$$

Here, Hybrid world 0 is same as World log(n) and Hybrid World 0 is same as World 1. As the difference between two consecutive hybrid worlds is only of one random and one pseudo random number generated by G, if there exists an adversary which can differentiate between two consecutive hybrid worlds, we can construct a reduction which breaks the security of G. Thus F is secure.

(b)

From the above proof, the probability of differentiating between World 0 and World 1 is at most log(n)*(maximum probability of differentiating between two consecutive hybrid worlds). Thus, if the input space is extended to $\{0,1\}^n$, the number of hybrid worlds becomes n and this value becomes n* (maximum probability of differentiating between two consecutive hybrid worlds) which is still negligible. Hence we can say that the proof still works and F is secure.

(c)

We can define the following adversary A to break the given PRF construction:

- The adversary queries two times, first for $x \leftarrow \{0,1\}^n$ and then for $0||x$.
- It receives two ciphers $ct00||ct01$ and $ct10||ct11$ respectively.
- if $ct10||ct11 = G_n(ct00)$ then it guesses 0 else guesses 1.

The winning probability of this adversary can be calculated by considering the following cases:

- the challenger chooses b=0,
$$Pr[A \text{ guesses } 0] = 1$$
- the challenger chooses b=1,
$$Pr[A \text{ guesses } 1] = 1 - \frac{1}{2^n}$$

Thus,
$$Pr[A \text{ wins}] = \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot (1 - \frac{1}{2^n}) = 1 - \frac{1}{2^{n+1}}$$
Therefore, the adversary A has non negligible advantage.

(d)

We can modify the given construction to the following:

PRG → PRF: variable length inputs

The PRF evaluation using key $k \in \{0,1\}^n$, on input $x \in \{0,1\}^*$ is defined as follows:

- Set $x = k||x$.
- Let $s = k$, and let l denote the length of x. For $i = 1$ to l, do the following:
  - Compute $(s_0, s_1) = G^n(s)$, where s_0 and s_1 are both n-bit strings.
  - Set $s = s_{x[i]}$.
- Output s.

The intuition behind this being secure is that the adversary cannot use attacks using the length of input like the previous attack as the last n operations are unknown which creates exponential possibilities of difference between two inputs in which one is prefix of other.

Part B B.1

1: CRIME Attack

```
from encrypt import encrypt
def attack():
    found=''
    d={}
    for i in range(24):
        for ch in range(97,123):
            e=encrypt(found+chr(ch)+" "*(23-i))
            d[ch]=len(e)
        c=min(d.keys(),key= lambda x: d[x])
        found+=chr(c)
    return found
```

The solution takes advantage of the following facts:

- If cmsg contains a substring of the cookie, then the compressed string is shorter.
- The length of the 'cookie' is fixed to be 24 bytes.

The attack function tries to recover the secret by iteratively guessing characters of the secret message from left to right. It begins with an empty found string and iterates over each position of the secret string (a total of 24 positions). For each position, the code iterates through all lowercase English letters (from 'a' to 'z') and generates a string by concatenating the characters guessed so far (found) with the current guessed character, followed by a number of spaces to make the total length of the string 24 characters. t then encrypts each constructed string and measures the length of the encrypted output. The length of the encrypted data is used as an indicator of the length of the compressed data, as compression reduces repeated characters. Among all the guesses for the current position, the code selects the character that results in the shortest encrypted length. This choice is made assuming that the character with the shortest encrypted length is more likely to be correct. The guessed character with the minimum encrypted length is appended to the found string. This process continues for all positions in the secret string. After iterating through all positions, the code accumulates the guessed characters into the found string, effectively revealing the secret message character by character.

2: Attack on 2DES encryption

```
from des import encrypt, decrypt, key_gen
def attack(message, ciphertext):
    txt1={}
    txt2={}
    for i in range(2**20):
        key=key_gen(i)
        e=encrypt(key,message)
        try:
            return (key,txt1[e])
        except:
            txt2[e]=key
        d=decrypt(key,ciphertext)
        try:
            return (txt2[d],key)
        except:
            txt1[d]=key
    return None
```

The solution takes advantage of the following facts:

- Functions used in both steps are the same(DES).
- The key-space is not so large and creating a database for it is feasible.

The code iterates over complete key-space using the key_gen function. For each key, the code encrypts the provided message using that key. The encrypted result (e) is used as a key to index the txt1 dictionary, storing the corresponding encryption key for that encrypted result. The code also attempts to decrypt the provided ciphertext using each key. The decrypted result (d) is used as a key to index the txt2 dictionary, storing the corresponding decryption key for that decrypted result. If a match is found between an encrypted result (e) and a previously stored decrypted result (d) in the txt1 dictionary, it implies that the message was encrypted using the current key and the key for the decrypted result respectively. If a match is found between a decrypted result (d) and a previously stored encrypted result (e) in the txt2 dictionary, it implies that the message was encrypted using the key for the encrypted result and the current key respectively. The key insight here is that the code leverages the fact that encryption and decryption are inverse operations. If the same intermediate result is obtained from encryption and decryption operations, it indicates a match between the keys used for 2DES.