Quiz 3

Problem 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:

- [x] Compress then encrypt.
- [x] Encrypt then compress.
- [x] The order does not matter – either one is fine.
- [ ] The order does not matter – neither one will compress the data.

Explanation

The answer depends on the specific scenario, but generally compressing before encrypting can offer better performance and security benefits.

- Compress the encrypt: Lower the redundancy of plaintext. This order can reduce the size and entropy of the data, making it easier to encrypt and harder to crack.
- Encrypt then compress: This order can avoid compression attacks or preserving encryption metadata

Problem 2

Let $G : \{0,1\}^s \to G : \{0,1\}^n$ be a secure PRG. Which of the following is a secure PRG (there is more than one correct answer):

- [ ] $G'(k) = G(k)||G(k)$: The output is simply two identical copies of $G(k)$ concatenated. This predictability and pattern break the indistinguishability property.
- [x] $G'(k) = G(k \oplus 1^s)$: Since G is a secure PRG, and XORing k with 1^s simply flips all bits in k, the modified key is still uniformly distributed if k is uniformly distributed.
- [ ] $G'(k) = G(0)$: Using a constant input for G results in a constant output, which is easily distinguishable from a random string.

- [ ] $G'(k) = G(k)||0$: Appending a known pattern (in this case, a single 0) to the output of a secure PRG might seem minor, but it introduces a predictable element to the output, potentially compromising its security.
- [x] $G'(k1, k2) = G(k1)||G(k2)$: If both k_1 and k_2 are chosen independently and uniformly at random, then the output of $G(k_1)||G(k_2)$ is also pseudorandom and indistinguishable from truly random output.
- [x] $G'(k) = \mathrm{reverse}(G(k))$: Reversing the output of a secure PRG does not affect its randomness. The sequence of bits is still pseudorandom, and no predictable pattern is introduced by simply reversing the order of bits. Therefore, this remains a secure PRG.
- [x] $G'(k) = rotaion_n(G(k))$: Similar to the reversal operation, rotating the bits of a secure PRG's output does not compromise its security. The bits remain pseudorandom, and the operation does not introduce any predictability or distinguishability from truly random sequences.

Probelm 3

Let (E, D) be a (one-time) semantically secure cipher with key space $K = \{0,1\}^\ell$. A bank wishes to split a decryption key $k \in \{0,1\}^\ell$ 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 $\{0,1\}^\ell$ and sets $k_1' \leftarrow k \oplus k_1$. Note that $k_1 \oplus k_1' = k$. 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?

- [ ] $p_1 = (k_1, k_2),\ p_2 = (k_1, k_2),\ p_3 = (k_2')$
- [ ] $p_1 = (k_1, k_2),\ p_2 = (k_1', k_2'),\ p_3 = (k_2')$

- [x] $p_1 = (k_1, k_2),\ p_2 = (k_1', k_2),\ p_3 = (k_2')$
- [ ] $p_1 = (k_1, k_2),\ p_2 = (k_2, k_2'),\ p_3 = (k_2')$
- [ ] $p_1 = (k_1, k_2),\ p_2 = (k_1'),\ p_3 = (k_2')$

Explanation

- Combinations 1: $p_1 + p_2$ cannot decrypt any.
- Combinations 2: $p_2 + p_3$ cannot decrypt any.
- Combinations 4: p_2 can decrypt by itself.
- Combinations 5: $p_2 + p_3$ cannot decrypt any.

Problem 4

Let $M = C = K = \{0, 1, 2, \ldots, 255\}$ and consider the following cipher defined over (K, M, C):

$$E(k, m) = m + k \mod 256;\ D(k, c) = c - k \mod 256$$

Does this cipher have perfect secrecy?

- [ ] No, there is a simple attack on this cipher.
- [x] Yes
- [ ] No, only the One Time Pad has perfect secrecy.

Explanation

Problem 5

Let (E, D) be a (one-time) semantically secure cipher where the message and ciphertext space is $\{0, 1\}^n$. Which of the following encryption schemes are (one-time) semantically secure?

- [ ] $E'(k, m) = E(0^n, m)$: Using a constant key for all encryptions violates the principle of semantic security, as it doesn't protect against chosen-plaintext attacks where the attacker can learn about the encryption function by analyzing encryptions of plaintexts they choose.
- [x] $E'((k, k'), m) = E(k, m) \parallel E(k', m)$: The encryption of m with k and then again with k' doesn't provide additional information about m
- [ ] $E'(k, m) = E(k, m) \parallel \mathrm{LSB}(m)$: This is not semantically secure because appending the least significant bit (LSB) of m to the ciphertext directly leaks information about the plaintext.

- [x] $E'(k,m) = 0 \parallel E(k,m)$: Prepending a 0 to the ciphertext does not provide any additional information about the plaintext, as the security of $E(k,m)$ is not compromised by this addition.
- [ ] $E'(k,m) = E(k,m) \parallel k$: Apending the key to the ciphertext completely undermines the security of the encryption, making it trivial for an attacker to decrypt the ciphertext.
- [x] $E'(k,m) = \text{reverse}(E(k,m))$: Reversing the ciphertext does not reveal any additional information about the plaintext.
- [x] $E'(k,m) = rotation_n(E(k,m))$: This operation, like reversing, does not inherently leak any information about the plaintext if the original encryption scheme E is semantically secure.

Problem 6

Suppose you are told that the one time pad encryption of the message "attack at dawn" is `6c73d5240a948c86981bc29481d` (the plaintext letters are encoded as 8-bit ASCII and the given ciphertext is written in hex). What would be the one time pad encryption of the message "attack at dusk" under the same OTP key?

```
m1 = 'attack at dawn'.encode(encoding='ascii')
m2 = 'defend at noon'.encode(encoding='ascii')
c1 = bytes.fromhex('6c73d5240a948c86981bc294814d')
c2 = bytes(i ^ j for i, j in zip(m2, bytes(i ^ j for i, j in
zip(m1, c1))))
c2_b = ''.join(format(byte, '08b') for byte in c2)
ic2_b = int(c2_b, 2)
print(hex(ic2_b)[2:])
```

```
ANSWER: 6962c720079b8c86981bc89a994d
```

Problem 7

The movie industry wants to protect digital content distributed on DVD's. We develop a variant of a method used to protect Blu-ray disks called AACS.

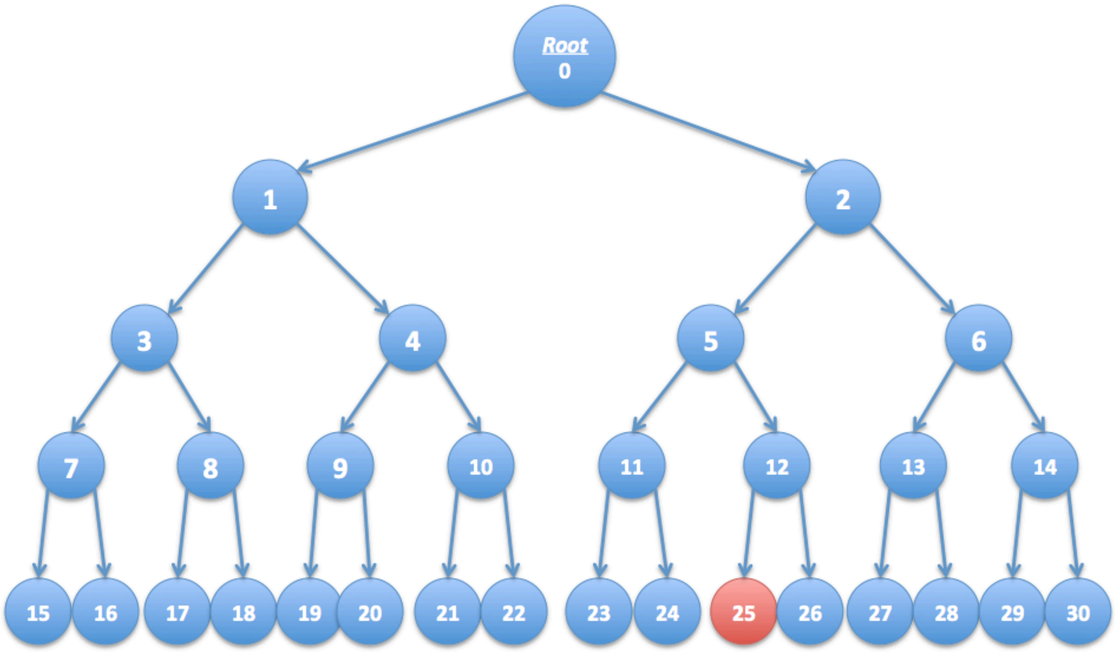
Suppose there are at most a total of n DVD players in the world (e.g. n=2^{32}). We view these n players as the leaves of a binary tree of height log2n. Each node in this binary tree contains an AES key k^i. These keys are kept secret from consumers and are fixed for all time. At manufacturing time each DVD player is assigned a serial number $i \in [0, n-1]$. Consider the set of nodes Si along the path from the root to leaf number i in the binary tree. The manufacturer of the DVD player embeds in player number i the keys associated with the nodes in the set S_i. A DVD movie m is encrypted as

$E(k_{root}, k) || E(k, m)$

where k is a random AES key called a content-key and kroot is the key associated with the root of the tree. Since all DVD players have the key kroot all players can decrypt the movie m. We refer to $E(k_{root}, k)$ as the header and E(k,m) as the body. In what follows the DVD header may contain multiple ciphertexts where each ciphertext is the encryption of the content-key k under some key k_i in the binary tree.

Suppose the keys embedded in DVD player number r are exposed by hackers and published on the Internet. In this problem we show that when the movie industry distributes a new DVD movie, they can encrypt the contents of the DVD using a slightly larger header (containing about $\log_2 n$ keys) so that all DVD players, except for player number r, can decrypt the movie. In effect, the movie industry disables player number r without affecting other players.

As shown below, consider a tree with n=16 leaves. Suppose the leaf node labeled 25 corresponds to an exposed DVD player key. Check the set of keys below under which to encrypt the key k so that every player other than player 25 can decrypt the DVD. Only four keys are needed.



- [ ] 21
- [ ] 17
- [ ] 5
- [x] 26
- [x] 6
- [x] 1
- [x] 11
- [ ] 24

Explanation

The path from root to 25 is : {0, 2, 5, 12, 25}, so the other child of each node is {1, 6, 11, 26}.

Extra Credit

Algorithm Differences

- **SHA-256** is part of the SHA-2 family of hashing algorithms, producing a 256-bit hash value. It operates on 32-bit words.
- **SHA-512/256** is a SHA-512 variant also in the SHA-2 family, but it operates on 64-bit words and produces a 512-bit hash value, truncated

to 256 bits. The initialization vectors differ from SHA-512, ensuring distinct outputs.

Security Properties

- **Resistance to Collisions**: Both algorithms aim for collision resistance. SHA-512/256 theoretically offers better resistance due to its larger internal state and truncation, which helps mitigate certain attacks like length extension.
- **Pre-image and Second Pre-image Resistance**: Designed to be secure against finding a pre-image or a second pre-image, making both algorithms robust against such attacks.

Performance

- **SHA-256** is generally faster on 32-bit hardware due to its optimization for 32-bit operations.
- **SHA-512/256** may perform better on 64-bit platforms, utilizing 64-bit operations more efficiently.

Which One Is Better?

- The choice depends on application requirements, including hardware architecture, performance needs, and security considerations.
- **For 64-bit Hardware**: SHA-512/256 might be preferable for its performance advantages.
- **Security**: Theoretically, SHA-512/256 has slight advantages due to its handling of internal state and resistance to specific attacks.
- **Overall**: The decision should be based on matching the algorithm to the specific context and needs of the application. SHA-256 remains widely used and trusted, while SHA-512/256 offers benefits in certain scenarios, particularly on 64-bit systems.