Homework 2 - HW46

Ari Gunnar Kristjónsson
Mikael Máni Eyfeld Clarke
Kristófer Birgir Hjörleifsson

Exercise 2.1

1.

Message	MerkleTree(m_i, s)	$\#H$
$m_1,\ \lvert m_1\rvert = 1$	$H(s\|m_1)$	1
$m_2,\ \lvert m_2\rvert = 4$	$H(s\|m_2)$	1
$m_3,\ \lvert m_3\rvert = 5$	$H(s\|m_3)$	1
$m_4,\ \lvert m_4\rvert = 8$	$H(s\|m_4)$	1
$m_5,\ \lvert m_5\rvert = 16$	$H\big(s\|(H(s\|m'[1;8])\|H(s\|m'[9;16]))\big)$	3

Explanation:

- Messages shorter than or equal to one block ($\lambda = 8$) are simply hashed once.
- For m_5, we have two 8-bit blocks, both hashed, then their hashes combined and hashed again, giving 3 total hash calls.

2. Merkle proof

Instead of sending the entire message, participant A can send only:

- the message block m_i, and
- the *authentication path*, i.e. all sibling hashes along the path from m_i to the root.

The authentication path has length $\log_2(t)$ where t is the number of blocks, and each hash is λ bits long. So the proof size is roughly:

$$\lambda(1 + \log_2(t)).$$

This is much smaller than sending all $t \cdot \lambda$ bits.

Example: $m = m_1\|m_2\|m_3\|m_4$ To prove m_4, A sends m_4 along with Hash(m_3) and Hash$(m_1\|m_2)$.

```
           Root
          /    \
Hash(m1|m2)  Hash(m3|m4)
               /        \
          Hash(m3)       Hash(m4)
```

Verification by B:

1. Compute $h_4 = H(s\|m_4)$.
2. Combine h_4 with Hash(m_3) to get Hash$(m_3\|m_4)$.

3. Combine this with $\text{Hash}(m_1 \| m_2)$ to get the root.
4. Accept if the root equals the known root hash.

3. Implementation note

We used a recursive approach with Shake-256 returning 256 bits. The message is split into fixed-size blocks, padded if necessary to make an even number of blocks. Each pair of blocks is concatenated, hashed, and forms a parent node. This continues until one hash remains—the Merkle root.

Exercise 2.2

1.

Let $c = f(m)^e \bmod N$ be a ciphertext from ENC$'$, and define $c' = 2c \bmod N$. Then:

$(c')^d \bmod N = (2c)^d \bmod N = (2^d \bmod N)(c^d \bmod N) \bmod N = (2^d \bmod N)f(m) \bmod N.$

So DEC$'$ outputs $\perp$ when this result is not within the valid λ-bit range. This does not break correctness for honest ciphertexts; it only shows that modified ciphertexts can decrypt to $\perp$.

2.

For any plaintext $m \in \{0,1\}^\lambda$, we do:

$$k \leftarrow \text{SKE.Keygen}() \qquad c_1 \leftarrow \text{RSA.Enc}'(pk, k), \qquad c_2 \leftarrow \text{SKE.Enc}(k, m).$$

Then the ciphertext is $c = (c_1, c_2)$.

Decryption does:

$$k' \leftarrow \text{RSA.Dec}'(sk, c_1), \qquad m' \leftarrow \text{SKE.Dec}(k', c_2).$$

Since $\text{RSA.Dec}'(\text{RSA.Enc}'(k)) = k$ and $\text{SKE.Dec}(k, \text{SKE.Enc}(k, m)) = m$, the scheme always decrypts correctly. Thus, HE is a **correct** public key encryption scheme.

3.

We have the challenge ciphertext

$$(c_1, c_2) = (\text{RSA.Enc}'(pk, k),\ m_b \oplus k),$$

where $b \in \{0,1\}$. The attacker can make a modified query:

$$(c_1, c_2 \oplus (m_0 \oplus m_1)).$$

When decrypted, we get:

$$(m_b \oplus k) \oplus (m_0 \oplus m_1) \oplus k = m_b \oplus (m_0 \oplus m_1) = \begin{cases} m_1 & \text{if } b = 0, \\ m_0 & \text{if } b = 1. \end{cases}$$

So the attacker can identify b perfectly with one oracle query. Therefore, this HE scheme is **not IND-CCA secure**.

4.

To make the scheme secure, we replace RSA with an IND-CCA secure PKE (for example, RSA-OAEP) and the symmetric scheme with an IND-CCA secure AEAD (authenticated encryption).

The new encryption works as:

$$k \xleftarrow{\$} \{0,1\}^\lambda, \quad c_1 = \text{PKE.Enc}(pk, k), \quad c_2 = \text{SKE.Enc}(k, m).$$

In the security proof:

- **Game 0 $\to$ 1:** Replace k by a random k^*. By IND-CPA of PKE, indistinguishable.
- **Game 1 $\to$ 2:** Replace $c_2 = \text{SKE.Enc}(k^*, m_b)$ by $\text{SKE.Enc}(k^*, m_{1-b})$. By IND-CPA of SKE, also indistinguishable.

Now the ciphertext distribution is independent of b, so the scheme is IND-CPA secure. If we use AEAD and IND-CCA secure PKE, the overall scheme becomes IND-CCA secure.