CS587 Assignement 2

Sri Teja Kale - G01501801

April 22, 2025

Exercise 1: Secure MAC [20 points]

Question: Suppose MAC is a secure MAC algorithm. Define a new algorithm:

$$MAC'(k, m) = MAC(k, m) \parallel MAC(k, m).$$

Prove that MAC' is also a secure MAC algorithm via a security reduction. Follow these steps:

1. Verification. Clearly define Verify'(k, m, t) for MAC'. Show under what condition (m, t) is considered valid.
2. Contrapositive statement. State precisely how breaking MAC' implies breaking MAC.
3. Security Reduction. Construct an explicit reduction that uses an adversary A against MAC' to build an adversary B against MAC.
4. Probability Analysis. Prove that if A succeeds with probability ε, then B succeeds with probability at least ε (or close to it). Conclude that MAC' is secure if MAC is secure.

Answer:

1. Verification

Let's clearly define the verification algorithm for MAC':

$$\text{Verify}(k, m, t):$$

1. Parse $t = (t_s \parallel t_z)$ where $|t_s| = |t_a|$ is the output length of MAC.
2. Check if $t_s = t_a$.
3. Verify if $t_1 = MAC(k, m)$.
4. Return 1 (valid) if both conditions are true, otherwise return 0 (invalid).

Thus, a message-tag pair (m, t) is valid if and only if t can be parsed as $(t_s \parallel t_z)$ where $t_s = t_e = MAC(k, m)$.

2. Contrapositive Statement

"If MAC' is not secure, then MAC is not secure." More precisely, if there is an adversary A that can break the security of MAC' (i.e., can forge valid tags for MAC) with non-negligible probability ε, then there is an adversary B that breaks the security of MAC with non-negligible probability at least ε.

3. Security Reduction

We will construct an adversary B against MAC using an adversary A against MAC':

1. B has access to the MAC$(k, \cdot)$ oracle for an unknown key k.
2. B simulates the MAC' oracle for A:
   - When A queries a message m, B queries its MAC oracle to get $t = MAC(k, m)$ and returns $t_s \parallel t_z$ to A as MAC'(k, m).
3. Eventually, A outputs a forgery (m^*, t^*) for MAC', where m^* was not previously queried.
4. B parses $t^* = (t_1 \parallel t_2)$ and outputs (m^*, t_1) as its forgery for MAC.

4. Probability Analysis

Let's analyze the success probability of B: If A forges MAC' with probability ε, then:

- The probability $t_1 = MAC(k, m)$ is still ε (since $t_1 = t_2$ is required).

Therefore, B forges a valid MAC(k, m) with at least probability ε. If adversary A successfully forges a valid MAC' tag with probability ε, then adversary B will successfully forge a valid MAC tag with the same probability ε. This proves that if MAC is secure, then MAC' is also secure.

Exercise 2: Insecure Hash Functions [30 points]

Question 1

Let $H : \{0,1\}^* \to \{0,1\}^n$ be any function satisfying:

$$H(x \oplus w) = H(x) \oplus H(w) \text{ for all } x, w.$$

Prove that such an H is not collision-resistant. (Hint: given any m, think of $m = x \oplus w$ and show how to find distinct $m_1 \neq m_2$ such that $H(m_1) = H(m_2)$.)

Answer

Given Hash function is defined as $H : \{0,1\}^* \to \{0,1\}^n$ with the property:

$$H(x \oplus w) = H(x) \oplus H(w).$$

To show that H is not collision-resistant, we need to find two different inputs that produce the same hash output.

Let $m_1 = 0^n$ (a string of n zeros) and $m_2 = 0^{2n}$ (a string of $2n$ zeros). These messages are clearly different since they have different lengths.

Now, let's calculate their hash values:

$$H(m_1) = H(0^n) = H(x \oplus x) = H(x) \oplus H(x) = 0^n.$$

Similarly:

$$H(m_2) = H(0^{2n}) = H(w \oplus w) = H(w) \oplus H(w) = 0^n.$$

Thus, $H(m_1) = H(m_2)$ even though $m_1 \neq m_2$. Therefore, H is not collision-resistant.

Question 2

Let H_s be a collision-resistant hash function, and define:

$$H_a(x_1 \parallel x_2) = H_s(x_1) \oplus H_s(x_2),$$

where $|x_1| = |x_2|$. Show that H_a is not collision-resistant by demonstrating a collision-finding strategy.

Answer

Given Hash function is defined as $H_a(x_1 \parallel x_2) = H_s(x_1) \oplus H_s(x_2)$ and x_1, x_2 are bitstrings of equal length.

To show that H_a is not collision-resistant, we need to find two different inputs that produce the same hash output.

Let us assume two bitstrings a and b of same length and $a \neq b$.

Define:

$$m_1 = a \parallel b, \quad m_2 = b \parallel a.$$

Since a and b are not equal, $m_1 \neq m_2$.

Now compute the hash:

$$H_a(m_1) = H_s(a) \oplus H_s(b), \quad H_a(m_2) = H_s(b) \oplus H_s(a).$$

Because XOR is commutative, we have:

$$H_a(m_1) = H_a(m_2),$$

So:

$$H_a(m_1) = H_a(m_2) \quad \text{where} \quad m_1 \neq m_2.$$

Thus, H_a is not collision-resistant.

Exercise 3: Secure Hash Functions [25 points]

Question: Let $H_b(x) = H_1(x) \parallel H_2(x) \parallel H_3(x)$ where H_1, H_2, and H_3 are different hash functions, and only one of them is collision-resistant. Show that $H_b(x)$ is a collision-resistant function (via a reduction).

Answer: Let us assume that H_1 is a collision-resistant hash function, while the other two are not. We want to prove that $H_b(x) = H_1(x) \parallel H_2(x) \parallel H_3(x)$ is also a collision-resistant function. We will prove this via a reduction: if an adversary could break $H_b(x)$, then they could also break $H_1(x)$, which contradicts our assumption.

Let us assume there is an algorithm A that can find collisions in H_1. This means A can find two different inputs x and y where $x \neq y$ such that:

$$H_1(x) = H_1(y).$$

Then:

$$H_2(x) = H_2(y) \quad \text{and} \quad H_3(x) = H_3(y).$$

Looking at the first part, we can see that $H_1(x) = H_1(y)$ where $x \neq y$.

This means algorithm A has found a collision for H_1, but this contradicts our assumption that H_1 is collision-resistant.

Therefore, no efficient algorithm can find collisions for $H_b(x)$ if H_1 is collision-resistant.

This proves that $H_b(x)$ is collision-resistant if at least one of its component hash functions is collision-resistant.

Exercise 4: Based on Katz-Lindell Exercise 7.14. Complementarity Property of DES [25 points]

Part (a): Recall that the DES block cipher operates on 64-bit blocks and uses a 56-bit key (plus 8 parity bits). A notable property of DES is that if you encrypt a plaintext x under a key k to get a ciphertext y, then encrypting the bitwise complement of x (denoted x) under the bitwise complement of k (k) results in the bitwise complement of y (y). Formally:

$$DES_k(x) = y \implies DES_k(x) = y.$$

Prove that this complementarity property holds for every key k and input x. In your proof, you can rely on how DES is defined at the bitwise level and show that negating all input bits (including the key bits) negates all output bits.

Answer: The DES complementation property states that: If $DES_k(x) = y$, then $DES_k(x) = y$ where x, k, and y represent the bitwise complements of x, k, and y respectively.

Now, let us examine each stage of the DES encryption process to prove this.

1. Initial Permutation (IP)

The initial permutation simply reorders bits without modifying their values. Therefore:

- If we apply IP to x, we get $IP(x)$.
- If we apply IP to x, we get $IP(x) = IP(x)$.

This gives us the complemented input blocks L_0 and R_0 after initial permutation.

2. Round Structure Analysis

Let's analyze a single round, then extend to all rounds: After the initial permutation, the 64-bit block is split into left (L_0) and right (R_0) halves of 32 bits each. In each round i:

- $L_i = R_{i-1}$.
- $R_i = L_{i-1} \oplus E(R_{i-1}, k_i)$,

where F is the round function and k_i is the subkey for round i.

3. The Round Function E

Let's trace what happens in the F function when using complemented inputs:

- **Expansion Permutation (E):**

$$E(R_{i-1}) = \text{some 48-bit value}, \quad E(R_{i-1}) = E(R_{i-1}).$$

  The expansion just reorders and duplicates bits, so complemented input produces complemented output.

- **Key XOR:** If we XOR $E(R_{i-1})$ with subkey k_i, we get some value C. Using the XOR property: $a \oplus b = a \oplus b$, we have:

$$E(R_{i-1}) \oplus k_i = E(R_{i-1}) \oplus k_i = C.$$

  The output after XOR with the complemented key is the same as the original value, i.e., when input and key are not complemented.

- **S-boxes:** Since the input to the S-boxes is the same in both cases (C), the output will be the same (let's call it D).

- **Permutation (P):** Since P just reorders bits:

$$P(D) \text{ in normal case}, \quad P(D) \text{ in complemented case}.$$

  Again, the same output from both.

4. Round Output

For the normal case:

$$R_i = L_{i-1} \oplus E(R_{i-1}, k_i),$$

For the complemented case:

$$R_i = L_{i-1} \oplus E(R_{i-1}, k_i).$$

Since $F(R_{i-1}, k_i) = F(R_{i-1}, k_i)$, we have:

$$R_i = L_{i-1} \oplus F(R_{i-1}, k_i),$$

and using the XOR property:

$$a \oplus b = (a \oplus b),$$

we get:

$$R_i = (L_{i-1} \oplus F(R_{i-1}, k_i)).$$

And we already know that $L_i = R_{i-1}$.

5. Final Swap

Since P just reorders bits, after 16 rounds, L_{16} and R_{16} are swapped:

$$(R_{16}, L_{16}) \quad \text{with complemented inputs, we get:} \quad (R_{16}, L_{16}).$$

6. Final Permutation

Since the final permutation (IP^*) also just reorders bits:

$$IP^*(x) = IP^*(x),$$

which proves that the complementarity property holds for every key k and input x.

Exercise 4: Part B

Part (b): Because a single DES key is only 56 bits (making it vulnerable to exhaustive key search by modern standards), the Triple-DES (3DES) construction was introduced to extend the effective key length. One common version is two-key 3DES:

$$3DES_{k_1,k_2}(x) = DES_{k_1}(DES_{k_2}^{-1}(DES_{k_1}(x))),$$

where $DES_{k_2}^{-1}$ is simply DES decryption using key k_2. (Another variant, three-key 3DES, uses distinct keys k_1, k_2, k_3.) Discuss whether the single-DES complementarity property $DES_k(x) = y \implies DES_k(x) = y$ implies a similar property for 3DES (with either two or three keys). In other words:

$$3DES_{k_1,k_2}(x) = y \implies 3DES_{k_1,k_2}(x) = y.$$

If it does not fully carry over, explain why. Does the complementarity property of single DES represent a serious weakness in 3DES when 3DES is used as a pseudorandom permutation? Justify your answer.

Answer:

Triple DES and Complementation Property

For two-key 3DES, we have:

$$3DES_{k_1,k_2}(x) = DES_{k_1}(DES_{k_2}^{-1}(DES_{k_1}(x))).$$

We need to determine if DES's complementation property (where $DES_k(x) = y \implies DES_k(x) = y$) carries over to 3DES. Let me work through the steps of two-key 3DES with complemented inputs and keys:

1. **Step 1: First DES encryption:**
   - Normal: $DES_{k_1}(x) = y_1$.
   - Complemented: $DES_{k_1}(x) = y_1$ (by DES complementation property).
2. **Step 2: Middle DES decryption:**
   - Normal: $DES_{k_2}^{-1}(y_1) = z$.
   - Complemented: $DES_{k_2}^{-1}(y_1) = z$ (complementation works for decryption too).
3. **Step 3: Final DES encryption:**
   - Normal: $DES_{k_1}(z) = y$.
   - Complemented: $DES_{k_1}(z) = y$ (by DES complementation property).

This shows that if $3DES_{k_1,k_2}(x) = y$, then $3DES_{k_1,k_2}(x) = y$. The same relationship holds for three-key 3DES through similar analysis.

So yes, the complementation property does carry over to Triple DES, for both two-key and three-key variants. Even though this property does effectively reduce the keyspace (for two-key 3DES, it reduces it from 2^{112} to 2^{56}), it does create a technical distinguisher between 3DES and a truly random permutation:

- A true PRP should not have predictable relationships between outputs.
- In 3DES, for a specific value x, if you know the encryption of x, you can predict the encryption of x without knowing the keys.

That said, this doesn't constitute a serious weakness for 3DES as a PRP because:

- The key space is only halved (so the reduction is negligible in this stage).
- Better attacks already exist (meet-in-the-middle and related key attacks reduce the security of two-key 3DES to approximately 2^{80}).
- This property does not help recover keys, nor break the cipher in practice.

In summary, while this complementation property violates the theoretical definition of a PRP, it does little to skeptically impact 3DES in practice compared to other known weaknesses.

Exercise 5: Insecure MACs [20 points]

1

Let F be a fixed-length PRF. Show that the following MAC scheme for messages m of length ℓn (where $m = m_1 \parallel \cdots \parallel m_\ell$ and each m_i is of size n-bits) is not secure:

$$t = F_k(m_1) \oplus F_k(m_2) \oplus \cdots \oplus F_k(m_\ell).$$

Answer

We are asked to show that the MAC scheme $t = F_k(m_1) \oplus F_k(m_2) \oplus \cdots \oplus F_k(m_\ell)$ is not secure.

To show this is not secure, let's describe an adversary that can break the scheme.

Adversary A:

1. A queries the MAC oracle with message $m = m_1 \parallel m_2$ and receives tag $t_1 = F_k(m_1) \oplus F_k(m_2)$.
2. A queries the MAC oracle with message $m' = m_2 \parallel m_3$ and receives tag $t_2 = F_k(m_2) \oplus F_k(m_3)$.
3. A computes $t^* = t_1 \oplus t_2 = [F_k(m_1) \oplus F_k(m_2)] \oplus [F_k(m_2) \oplus F_k(m_3)]$.
4. A outputs the forgery $(m^*, t^*) = (m_1 \parallel m_3, t^*)$.

This is a valid forgery because:

$$V_{\text{verify}}(m^*, t^*) = 1 \quad \text{since} \quad t^* = F_k(m_1) \oplus F_k(m_3),$$

which is the correct MAC for $m^* = m_1 \parallel m_3$. Thus, the adversary succeeds with probability 1, which violates the definition of EUF-CMA security.

2

Show that the following MAC scheme for messages m of length $2n$ (where $m = m_1 \parallel m_2$ and each m_1, m_2 is of size n-bits) is not secure:

$$t = F_k(m_1) \parallel F_k(m_2).$$

Answer

We are asked to show that the MAC scheme $t = F_k(m_1) \parallel F_k(m_2)$ is not secure.

To show this is not secure, let's describe an adversary that can break the scheme.

Adversary A:

1. A queries the MAC oracle with message $m = m_1 \parallel m_2$ and receives tag $t = F_k(m_1) \parallel F_k(m_2)$.

2. A queries the MAC oracle with message $m' = m_2 \parallel m_3$ and receives tag $t' = F_k(m_2) \parallel F_k(m_3)$.

3. A constructs the forgery tag $t^* = F_k(m_1) \parallel F_k(m_3)$.

4. A outputs the forgery $(m^*, t^*) = (m_1 \parallel m_3, t^*)$.

This is a valid forgery because:

$$V_{\text{verify}}(m^*, t^*) = 1 \quad \text{since} \quad t^* = F_k(m_1) \parallel F_k(m_3),$$

which is the correct MAC for $m^* = m_1 \parallel m_3$. Thus, the adversary succeeds with probability 1, which violates the definition of EUF-CMA security.