CS6111 (2025): Assignment 3

Instructor: John Augustine

Due: Oct. 26, 2025 (11 PM)

- Use LaTeX to typeset your answer. You must submit the PDF file.
- You can use theorems/lemmas from the textbook (Katz and Lindell, Edition 3 only) without proving them. However, you should mention the theorem/lemma number from the textbook and paraphrase the statement of the theorem/lemma to make your solution self-contained.
- The deadline is Oct. 26, 2025 (11 PM).
- Please follow the academic honesty policy for assignments. If you are stuck and need a hint, please post a question on Moodle.
- The assignment is 24 marks. It will be later scaled down to 6 marks.

1. (0 marks) Reading assignment: Katz and Lindell Ed. 3: §6.5, §6.6, §9.1, §9.3 (starred sections can be skipped).

Solution: The given sections have been read and understood to the best of my knowledge.

2. (5 marks) Explain the importance of padding and encoding message length in hash function constructions. Show with a concrete counterexample how omitting message length encoding can result in collisions even if the compression function is collision-resistant.

Solution: Assume that the compression function is a fixed length encoding of size n'. Then we need to extend the message of arbitrary length with padding so that it attains a multiple of n'.

If it is appended with padding, then there has to be a proper padding structure. 0 padding results in collisions, take for example,

$$m = 10 \quad and \quad m' = 100$$

And if the message length is 4 for example, appending 0s gives,

$$m = 1000 \quad and \quad m' = 1000$$

which results in a collision. Even though the original messages are different, the padded messages are identical, leading to a collision:

$$H(m) = H(m').$$

Consider a setup similar ot Merkle-Damgard without length encoding, here we first pad 1 to the message followed by 0s. Assume that the block size is 4. Consider 2 messages,

$$m = 100 \quad and \quad m' = 1001$$

After padding, we have

$$m = 1001 \quad and \quad m' = 1001$$

Hence, Even though the original messages are different, the padded messages are identical, leading to a collision $H(m) = H(m')$. Without length encoding, the hash function cannot distinguish between messages of different lengths that otherwise align in blocks. If we add the length encoding here, then the messages have distince blocks as shown below

$$m = 10010011 \quad and \quad m' = 10010100$$

Since, the compression function is collision resistant, the above messages collide only with negligible probability. Therefore, length encoding and padding are important.

3. (3 marks) Fix $H : \{0,1\}^n \to \{0,1\}^{2n}$, and define the keyed function $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^{2n}$ by

$$F_k(x) = H(k \oplus x).$$

Show that an attacker given oracle access to $F_k(\cdot)$ can recover the n-bit key k with constant probability in time approximately $2^{n/2}$.

Solution: If the adversary is given access to the oracle $F_k(.)$,

$$F_k(x) = H(k \oplus x)$$

Let us perform the following Birthday attack to recover the n-bit key k, lets query the oracle H(.) N times.

$$(H(y_1), H(y_2), H(y_3), \ldots, H(y_N)) \textit{ are the outputs from the N queries}$$

Now query the oracle $F_k(.)$ N times,

$$(F_k(x_1), F_k(x_2), \ldots, F_k(x_N)) \textit{ are the outputs from the N queries}$$

If there is a collision, i.e, for some (i, j), the key can be retrieved as follows

$$F_k(x_i) = H(y_j) \implies k \oplus x_i = y_j$$

$$k = x_i \oplus y_j$$

Let us now compute N to get a constant probability of collision, consider the balls and bins probem (birthday problem generalization), we have m balls and n bins,

$$Pr[collision] = 1 - Pr[No\ Collision] = 1 - (1 - \frac{1}{n})(1 - \frac{2}{n}) \ldots (1 - \frac{m-1}{n})$$

Using approximation, $1 - x \approx e^{-x}$,

$$Pr[collision] \approx 1 - e^{\frac{1+2+3+\cdots+(m-1)}{n}} \approx 1 - e^{\frac{m(m-1)}{2n}} \approx p$$

$$\implies m \approx \sqrt{2n \ln(\frac{1}{1-p})}$$

In this problem, the number of bins is the range space of the hash function H,

$$m = \sqrt{2 * 2^n \ln(\frac{1}{1-p})} = O(2^{\frac{n}{2}})$$

Hence, in time approximately $O(2^{\frac{n}{2}})$ with a constant probability p, we can retrieve the k using this attack.

Note: It is not necessary that if for some (i, j) there is a collision, then the inputs should match and $k = x_i \oplus y_j$. However, we can perform $O(2^{\frac{n}{2}})$ more queries and observe if we get the same k by this attack. This confirms that the value of key k retrieved is right.

4. (3 marks) Given a hash function (Gen, h) that is collision resistant, prove or disprove whether the following hash functions (Gen, H) are also collision resistant:

   (a) $H(x) = h(x \mid 1)$ (bitwise OR)

Solution: Given, (Gen, h) is collision resistant, this implies that, for all probabilistic polynomial time adversaries,n

$$Pr[Hash\_coll_{\mathcal{A},h}(n) = 1] \leq negl(n)$$

The given hash function, $H(x)$ is not collision resistant.

Let us take two inputs,

$$x = 101 \quad and \quad x' = 100$$

Then,

$$H(x) = h(101|1) = h(101|001) = h(101)$$
$$H(x') = h(100|1) = h(100|001) = h(101)$$

Therefore, when the inputs x and x' differ only in the last bit, they result in a collision,

$$H(x) = H(x')$$

Hence, the probability of collision can be computed as follows, if the inputs x and x' are of length l

$$\#\ of\ pairs\ with\ collision = \frac{2^l}{2} = 2^{l-1}$$

$$\#\ of\ pairs = \binom{2^l}{2}$$

Hence,

$$Pr[Hash\_coll_{\mathcal{A},\mathcal{H}}(n) = 1] = \sum_{l=1}^{\infty} \frac{1}{2^l - 1} \geq \sum_{l=1}^{\infty} \frac{1}{2^l}$$

By Reimann integral form,

$$\implies Pr[Hash\_coll_{\mathcal{A},\mathcal{H}}(n) = 1] \geq \int_1^{\infty} \frac{1}{2^l} dl = \frac{1}{2 \ln 2}$$

Since the success probability is not negligible, the hash function $H(x)$ is not collision resistant.

(b) $H(x) = h(x \| \text{LSB}(x))$ (concatenation, where LSB denotes the least significant bit)

Solution: The given hash function $H(x)$ is collision resistant.

Let us take two inputs x and x', such that, $x \neq x'$. Then there are two cases,

$$LSB(x) = LSB(x') \quad and \quad LSB(x) \neq LSB(x')$$

However, since $x \neq x'$, in both the cases,

$$x||LSB(x) \neq x'||LSB(x')$$

Therefore,

$$Pr[Hash\_coll_{\mathcal{A},\mathcal{H}}(n) = 1] = Pr[Hash\_coll_{\mathcal{A},h}(n) = 1] \leq negl(n)$$

Since the probability of collision is negligible, H is a collision resistant hash function.

5. (5 marks) Consider the modification of the Merkle–Damgård transform where the initial chain value, i.e. z_0, is the length L of the input string x instead of a fixed IV. Given that (Gen, h) is collision resistant, state and prove whether (Gen, H) is collision resistant or not.

Solution: Yes, the (Gen, H) is collision resistant.
Consider that we hash two messages m, m' using the Merkle–Damgård transform. There are two cases,

- **Case 1:** $|m| = |m'|$
  This is similar to the normal MD transform with the same IV, $m \neq m'$ and $L = L'$. Let ,
  $$I_i = z_{i-1}||m_i \quad and \quad I_B = z_B$$
  Let N be the largest index at which, $I_N \neq I'_N$, since $m \neq m'$, such an N definitely exists. Now, by contradiction assume that,
  $$z_B = z'_B$$
  $$\implies I_{N+1} = I'_{N+1}$$
  This means that, I_N and I'_N collide under h. Since, h is a collision resistant hash functions,
  $$Pr[Hash\_coll_{\mathcal{A},\mathcal{H}}(n) = 1] = Pr[Hash\_coll_{\mathcal{A},h}(n) = 1] \leq negl(n)$$
- **Case 2:** $|m| \neq |m'|$
  Since, $z_0 \neq z'_0$, hence the first compression outputs differ. Similarly, $x_B \neq x'_B$

(the last block encodes the length of the messages which differ here). Therefore, for collision in the final outputs many intermediate outputs ($z_i's$) need to collide. But since h is a collision resistant hash function,

$$Pr[Hash\_coll_{\mathcal{A},\mathcal{H}}(n) = 1] = Pr[Hash\_coll_{\mathcal{A},h}(n) = 1] \leq negl(n)$$

Thus , (Gen, H) is a collision resistant hash function.

6. (4 marks) Describe a man-in-the-middle attack on the Diffie–Hellman protocol where the adversary shares a key k_A with Alice and a (different) key k_B with Bob, and Alice and Bob cannot detect that anything is wrong. Show why your attack works.

Solution: The Diffie-Hellman protocol is defined as follows,

- Alice runs $\mathcal{G}(1^n)$ to obtain G, g, q
- Alice samples $x \in Z_q$ at random and computes $h_A = g^q$. Alice sends (G, g, q, h_A) to Bob
- Bob samples $y \in Z_q$ independently at random upon receiving. Bob computes $h_B = g^y$, sends it to Alice. Bob outputs key $k_B = h_A^y$
- Alice outputs key $k_A = h_B^x$ upon receiving.

However the Diffie-Hellman protocol is immune to the man-in-the-middle attack thus resulting in an insecure protocol without Alice and Bob realizing.
Consider an Eaves-dropper monitoring the channel between Alice and Bob. Since we assume that logarithm is a hard problem, Eaves-dropper cannot compute x and y by looking at the channel.

- **Attack intended for Bob :** $Alice \rightarrow Bob$
  Eav sees the message sent by Alice, and blocks it and instead sends, (G, g, q, h_{eav_A}) to Bob. Where,
  $$h_{eav_A} = g^{eav_x}$$
  Now Bob outputs key,
  $$k_B = h_{eav_A}^y = g^{y*eav_x}$$
- **Attack intended for Alice :**$Bob \rightarrow Alice$ Eav sees the message sent by Bob, and blocks it and instead sends, (G, g, q, h_{eav_B}) to Bob. Where,
  $$h_{eav_B} = g^{eav_y}$$
  Now Alice outputs key,
  $$k_A = h_{eav_B}^x = g^{x*eav_y}$$

After Alice and Bob has shared their keys, eav can now decrypt every message sent across the channel because Eav can compute the keys as follows,

- **Alice**: Since, Eav looked at (G, g, q, h_A) sent by Alice and knows eav_y, Eav computes k_A as follows,

$$k_A = h_A^{eav_y} = (g^x)^{eav_y} = g^{x*eav_y}$$

- **Bob**: Since, Eav looked at h_B sent by Bob and knows eav_x, Eav computes k_B as follows,

$$k_B = h_B^{eav_x} = (g^y)^{eav_x} = g^{y*eav_x}$$

Hence, Eav can decrypt and every message sent across the channel , modify the message re-crypt it with the right key and send it across the channel. Even a key confirmation protocol (message derived from the key) for Alice and Bob, can look legitimate as Eav can modify it to make it seem correct. Therefore, there is no way Alice and Bob realize the presence of an eaves dropper.

7. (4 marks) Define a MAC for arbitrary-length messages by

$$\mathrm{MAC}_k(m) = H(k\|m)$$

where H is a collision-resistant hash function.

Show that this is not a secure MAC when H is constructed via the Merkle–Damgård transform. Assume $k \in \{0,1\}^n$.

Solution:

Let us assume that we have access to the oracle of the compression function f(.) that is iteratively called in a MD transform. To attack the MAC algorithm, query the MAC oracle with a message m,

$$t = MAC_k(m) = H(k||m)$$

But since, we know that H is constructed via the Merkle-Damgard transform, we can use the tag t obtained from the previous query of MAC and query f(.) locally with any non-empty message m^*.

$$Let, \;\; pad(m) = 10\ldots0L_m$$

where, L_m denotes the length of the message m in binary.

The attacker can query f(.) locally now with message m^*, with the appropriate padding blockwise

$$t' = f(t, m^*, pad(m^*))$$

Since, we know the key k is n bits long and the length of message m itself, attacker can compute

$$pad(k||m)$$

Therefore, the new tag t' is a valid tag corresponds to the message given by,

$$m||pad(k||m)||m^*$$

Therefore, the output $(m||pad(k||m)||m^*, t')$ is a valid attack to the $MAC_k(.)$ oracle.

Hence, the $MAC_k(.)$ oracle is not secure.

Acknowledgment

I would like to thank Shri Prathaa M for their useful discussion on solving the problems.