CSE 40622 Cryptography
Writing Assignment 06

Name & netID: Walker Bagley (wbagley)

1. Suppose there are two hash functions H_1, H_2.
   - H_1: known to be collision resistant.
   - H_2: known to be second pre-image resistant.

1.1. (10 pts) Is H_1 always second pre-image resistant as well? Is H_1 always pre-image resistant as well?

Answer:

- Is H_1 second pre-image resistant?

  Yes. By the proof below, collision resistance implies second pre-image resistance. We know that H_1 is collision resistant, so it must be second pre-image resistant.

  Proof. By contrapositive, if we prove that a hash function that is not second pre-image resistant is not collision resistant, then we know that collision resistance implies second pre-image resistance. Assume H is not second pre-image resistant, so there exists an algorithm A which when given the security parameter and x can find some x' such that $x' \neq x$ and $H(x') = H(x)$. Select some x from the message space, and run A to find x'. Then we have a pair of messages x, x' where $x \neq x'$ and $H(x) = H(x')$, so we have broken collision resistance. Thus, collision resistance implies second pre-image resistance. $\square$

- Is H_1 pre-image resistant?

  Yes. Since collision resistance implies second pre-image resistance as shown above, H_1 must be second pre-image resistant. Further, via the proof in (1.2), we know that second pre-image resistance implies pre-image resistance. So, H_1 must also be pre-image resistant.

1.2. (10 pts) Is H_2 always collision resistant as well? Is H_2 always pre-image resistant as well?

Answer:

- Is H_2 collision resistant?

  Not necessarily. As second pre-image resistance requires the intractability of finding some x' where $x' \neq x$ and $H(x') = H(x)$ given any arbitrary x and the security parameter, we cannot infer collision resistance. Some hash function H could easily be second pre-image resistant but have an easy to find pair x, x' with identical hash digests. Due to the arbitrariness of x in second pre-image resistance, we cannot infer collision resistance as it does not need to apply to any arbitrary x in the message space.

- Is H_2 pre-image resistant?

  Yes. Second pre-image resistance implies pre-image resistance by the proof below. We know H_2 is second pre-image resistant, so it must be pre-image resistant.

  Proof. Suppose we have a hash function H which is second pre-image resistant. Towards a contradiction, assume H is not pre-image resistant, so there exists an algorithm A that when given the security parameter and a hash digest $H(x)$ for an unknown x, can find any x' such that $H(x') = H(x)$. Then we can use A to break second pre-image resistance by computing $H(x)$ for some x and running A on $H(x)$ to find some x' where $H(x') = H(x)$. If the message space is infinite it is incredibly likely that $x' \neq x$ as there must be many pre-images for a given hash digest. Then we have broken second pre-image resistance and have reached a contradiction. Thus, second pre-image resistance implies pre-image resistance. $\square$

2. Suppose we have a simple insecure hash function $H_{\mathsf{insecure}}(\texttt{input})$ whose digest is a 8-bit binary string. The algorithm of H_{insecure} is described below.

   (1.) Segment `input` into 8-bit segments.

   (2.) Assign the first segment to the internal state `int_state`.

   (3.) Compute XOR between `int_state` and the next segment, and overwrite `int_state` with the outcome.

   (4.) Repeat (3.) until all segments are XORed with `int_state`.

   (5.) Return `int_state` as the digest, i.e., $H_{\mathsf{insecure}}(\texttt{input})$.

   2.1. (10 pts) Compute the digest of H_{insecure} when the input is "110011001100110011001100".
   Answer:

$$\begin{aligned}
110011001100110011001100 &= 11001100\ 11001100\ 11001100 \\
\text{int_state} &= 11001100 \\
&= 11001100 \oplus 11001100 = 00000000 \\
&= 00000000 \oplus 11001100 = 11001100 \\
H_{\text{insecure}} &= 11001100
\end{aligned}$$

   2.2. (10 pts) Suppose we know $H_{\mathsf{insecure}}(m) =$"11111111" where m is an 256-bit message, but we do not know m.
   Compute $H_{\mathsf{insecure}}(m||\text{"11001100"})$ where $||$ denotes the string concatenation.
   Answer:

$$\begin{aligned}
H_{\text{insecure}}(m) &= 11111111 \\
H_{\text{insecure}}(m||\text{"}11001100\text{"}) &= H_{\text{insecure}}(m) \oplus 11001100 \\
&= 11111111 \oplus 11001100 = 00110011
\end{aligned}$$