Jiping Zhang
CSE 207B: Applied Cryptography Due:

Homework 4

0. To solve this problem, I modified the mal1 class by adding a hook before calling _process_chunk so that this class can record what chunks are put into this function.

   When attacking, we should guess all possible key lengths, and for each guess, use length extension attack to generate new url, and see if we can get something meaningful from server by using this url, and if we do get something meaningful, that means we guessed the length right.

   And, for a certain guessed key length, first use the modified mal1 class to generate mal1 hash value and get all the chunks produced, then append all the padding and length represented in bytes of the last chunk to the url(note, here if a byte can't be decoded by ascii or is an invalid char of url, we should use %0xbyte), finally, append the parameter string we want to url and generate hew hash value according to the last hash value , the parameter string we appended , padding and total length.

1. (a) lets prove its converse proposition :

   if H is not collision secure,

   that is, we can find m_1 and m_2 such that $H(m_1) = H(m_2)$ in reasonable time

   then , $(H_1(m_1), H_2(m_1)) = (H_1(m_2), H_2(m_2))$

   then , we have $H_1(m_1) = H_1(m_2)$ and $H_2(m_1) = H_2(m_2)$

   so if H is not collision secure, then neither H_1 nor H_2 is collision secure

   (b) if we can find m_1, m_2 such that $H_2(m_1) = H_2(m_2)$

   then $H'(m_2) = H_1(H_2(m_2)) = H_1(H_2(m_1)) = H'(m_1)$

2. Enc_1 is not safe, because attacker H_1 don't have a key. Attacker can compute $H_1(m)$ for all m and construct a table to map hash results of H_1 to possible original messages, and guess what original message m is every time attacker gets a $(e_k, h_1(m))$

   Enc_2 does not provide authentication, because attacker can substitute e_k to e'_k and append $H_2(e'_k)$ to it to generate a valid ciphertext. And the victim will receive some garbage if attacker randomly generates e'_k