HMAC using Collision resistant hash function

Sarthak Mahajan, 2018111005

March 7, 2022

1 Code structure:

- This folder contains hmac.py and run.py code files
- hmac.py contains the code to create the output of a HMAC.
- run.py is used to take inputs and return outputs of the HMAC

2 Instructions to run the code:

- Run the following command in the folder where this document is contained: python3 run.py
- There will be prompts asking you to input the relevant information serially, and will return the output.

3 Explanation of Functions in run.py:

3.1 run():

- It asks for inputs of key, key length in binary, the message(all in decimal format)
- Calculates q: the order of the group for DLP ,g: the generator of this group, and h: a random element in the group
- prints the HMAC tag of the message

4 Explanation of Functions in hmac.py:

4.1 HMAC(m, k, key_len,q,g,h):

- q,g,h are in decimal format.They are needed to be passes to the fixed_hash function. p is the largest prime possible in n bits, where n= length of the key, g ,h are random numbers in the range(1,q-1); note that they are

fixed for the program in terms of q(so that same seed gives same value for different iterations of the program), but they could be any random number in the given range.

- all inputs in decimal format, m is the message, k is the key, key_len is length of the key in binary format.
- The returned value is the HMAC of message m in binary format, hashed with key k of size _len in binary format.
- We apply fixed length collision resistant hash function h on $k \oplus ip(ipisactuallyipad)$ and an initialization vector iv, and the output of this is sent as an initialization vector to the Merkle Transform of the message m.
- Similarly, We apply fixed length collision resistant hash function h on $k \oplus op(opadisactuallyipad)$ and the same iv as before, and the output of this is and the result of the merkle transform is given to another h. The output of this is the returned value.
- Uses the pad function,merkle and fixed_hash functions