Assignment 5

Due Date: December 1st, 2024

1. Message Extension attacks on hashing algorithms:

Hashing mechanisms based on Merkle-Damgard-algorithm are all vulnerable to message extension attacks also known as length extension attacks. Popular hashing algorithms that rely on Merkle-Damgard-algorithm are MD4, MD5, SHA1 and SHA2.

Merkle-Damgard-algorithms operate on blocks of fixed size. If a message is longer than the block size, the message is split into blocks of equal size. However, it's possible that the message length is not exactly divisible by the block size. That is, one has to pad up the last block up, so that it fits the block size. This is called padding.

The algorithm stores the hash of each message block in a group of internal state registers. The state of internal registers becomes the starting point of operation for calculating the hash of next block. This cyclic operation continues until the lastmessage block is operated on. The internal state of registers after computation of hash for the last block becomes the hash value of the entire input message.

The goal of this assignment is to use the code provided in [1] to perform message extension attack. The source code provided in [1] implements a wide variety of hashing algorithms such as MD4, MD5, SHA1, SHA-256 and SHA-512 (SHA-256 and SHA-512 belong to SHA2 family of algorithms). You can use any of the listed algorithms to perform message extension attacks.

Problem Statement:

The author in [1] provides a ***demo code*** (hash_extension_1.c) to execute hash extension attack on MD5 and for the following specific scenario:

secret= “secret”
data= “data”
append=”append”

By studying the example code in hash_extension_1.c, you will have to perform message extension attack on the following scenario.

secret= “Hello, World!. This is a demonstration of message extension attacks. *Put your roll number in place of the text in italics.*”

Example:

Hello, World!. This is a demonstration of message extension attacks. SE21UCSE001

data= “My full name is *your full name here*”

Example: “My full name is N. Raghu Kisore.”

append=”*Your roll no will go here* ”.

Example:

append=” SE21UCSE001”

BONUS:

I will give you bonus points if you can demonstrate message extension attacks using ***hashlib*** python library. The bonus will be, I will push to the next higher letter grade than what you would normally get.

Output:

- Submit the modified version of ***hash_extension_1.c*** (will now work on the parameters defined in the problem statement above instead of the parameters used in the demo code).
- Your program should print **H(secret||data)** and **H(secret||data||append).**
- Remember that **H(secret||data||append)** will have to be calculated pretending that you don’t know the ***secret***.
- Submit ***hash_extension_1.c*** to your onedrive folder.

References:

1. https://github.com/iagox86/hash_extender
2. https://blog.skullsecurity.org/2012/everything-you-need-to-know-about-hash-length-extension-attacks
3. hashlib — Secure hashes and message digests — Python 3.13.0 documentation
4. I have uploaded relevant reading material into the OneDrive folder Assignment5 in the course OneDrive folder.

Other Useful Info:

- Download the zip file from [1] and unzip it.
- The zip folder contains make file use it to install the binaries.
- Use the command ***make all*** to install the binaries. Execute make all command from inside the unzipped folder.
- Read through the material in [1] and [2] to understand the working details of the code.
- The ***clue*** to executing the message extension attack is in fine tuning the following for loop in the file ***hash_extension_1.c***

```
for (i = 0; i < 16; i++) {
    printf("%02x", buffer[i]);
  }
```