Assignment # 4

Homework

Homework problems are a preparation for the quizzes. They are *not* graded. Please use the `piazza` forum to post questions you have on these problems.

- 11.2, 11.5, 11.6, 11.8, 12.2, 12.4, 12.5

Project

Note: For submissions on `mywpi`: Please submit a single pdf file containing your results. Please submit source code as a separate file, but make sure to have it listed in the pdf as well.

1. 12.3

2. Block ciphers can be turned into hash functions in a very simple way (cf. Section 11.3.2 of the book for a detailed explanation). One such construction is the *Matyas-Meyer-Oseas* construction defined as $H_i = \mathsf{Enc}_{H_{i-1}}(m_i) \oplus m_i$, where m_i is the i-th block of the message, H_i is the internal state of the hash function and the output $h(m) = H_l$ is the last state of the hash function. This construction is considered secure for an appropriate block cipher.

   A slight modification of the *Matyas-Meyer-Oseas* mode results in the following construction:
$$H_i = \mathsf{Enc}_{H_{i-1}}(m_i)$$

   (a) Draw the block diagram for both constructions.

   (b) Show why the modified construction is not secure by using the decryption function of the block cipher to obtain a (second) preimage. Assume H_0 is all-zeros.

   (c) Given the above method for finding preimages, describe how to find collisions.

3. The goal of this problem is to implement CBC-MAC yourself using a preexisting implementation of AES.

   (a) Your implementation should accept inputs of arbitrary length in bytes. Make sure to provide a single-1 padding (a single 1 followed by zeroes) combined with length strengthening where the length in bits is appended as a 64-bit number.

   (b) Implement the raw CBC-MAC using AES. This block takes the raw message and the key as input; it should call the padding function to create the input to the raw CBC-MAC; finally it should output the last ciphertext output without further processing. Make sure that inputs, outputs and intermediate states process on binary byte values. A template for this will be provided.

4. For this problem you will recover passwords from a hashed password file. The passwords are stored in the format used e.g. for `/etc/shadow` on linux systems. The second line is the format if the hash function is applied many (# of round) times consecutively.

   ```
   $[hash_algorithm_id]$[hash_salt]$[hash_data]
   $[hash_algorithm_id]$rounds=[# of rounds]$[hash_salt]$[hash_data]
   ```

   The password file is `passwords.txt`. Luckily you are aware that all passwords were chosen from an allowed list of passwords, which are provided in `dictionary.txt`.

   - Write a dictionary-attack that recovers all passwords of the password file using the provided dictionary `dictionary.txt`. Please use the provided template `pw_cracker_template`.
   - Note that the file contains three passwords following three different password policies. Document how long the run times of your recovery are for each of the different password policies. Explain your observations by commenting on the security of the three policies.

Good Luck and Have Fun!