MSU Denver, M&CS CS 3750: Computer and Network Security, Fall 2019 Dr. Weiying Zhu

Homework 5 (Total Points: 100), Due Date*: 12:00pm (noon) 10/14/2019, Cutoff Date*: 12:00pm 10/16/2019
***Late penalty will apply for past-due late submission **Submission will NOT be accepted after the cutoff deadline**
Submission: handwritten hardcopy at the beginning at the class (Email to wzhu1@msudenver.edu muse be used for late submission and the submission time is the moment when the email arrives at the instructor's inbox. Any copy in a student's mailbox cannot be used as a proof of submission.)

Part I. Review Questions for reading the textbook (No submission; however, the relevant contents will be included in Exams as True/False, Multiple-Choices, and/or Filling-in-the-Blanks questions for up to 20% of the total points.)

- **Textbook, Page 67, Review Questions**
  2.8 What are the principal ingredients of a public- key cryptosystem?
  2.10 What is the difference between a private key and a secret key?
  2.11 What is a digital signature?
  2.12 What is a public- key certificate?
  2.13 How can public- key encryption be used to distribute a secret key?

Part II. Problems (Submission Required Please!)

Problem A (based on 2/E Textbook, Page 104, 3.5) A phonetic password generator picks two segments randomly for each six- letter password. The form of each segment is CVC (consonant, vowel, consonant), where V = <a, e, i, o, u> and C = $\overline{V}$
a. What is the total password population?
b. What is the probability of an adversary guessing a password correctly (use scientific annotation and keep 3 significant digits)?

Problem B (based on Textbook, Page 104, 3.6) Assume that passwords are limited to the use of the 95 printable ASCII characters and that all passwords are 10 characters in length. Assume a password cracker with an encryption rate of 6.4 million encryptions per second. How long will it take to test exhaustively all possible passwords on a UNIX system **(unit: years)**?
(Hint: To keep it simple, just assumed that (1) the salt is either not used or for one given hashed password with the given salt; (2) one encryption means run the hash routine once.)

Problem C (**modified from Textbook, Page 104, 3.8 and 3.9**) It was stated that the inclusion of the salt in the UNIX password scheme increases the difficulty of guessing by a factor of 4096. Let's use a case to explain why it is asserted that the salt increases security although the salt is stored in plaintext in the same entry as the corresponding ciphertext password. Let's assume that the adversary has obtained a password file containing 4000 entries, each using a unique 12-bit salt.

a. In order to crack all the 4000 hashed passwords in the password file on a UNIX system, how many times does the hash routine, i.e., crypt(3), need to run in the worst case?
b. If the salt is NOT used, how many times does the hash routine, i.e., crypt(3), need to run in the worst case in order to crack all the 4000 hashed passwords?
c. If 24-bit salt is used for this case, does the hash routine need to run more times in the worst case to crack all the 4000 hashed passwords? If so, how many more?

Problem D. Consider a Bloom filter implemented in a proactive password checker. Four independent hash functions are used. There are 5 million words in the dictionary. In order to have a 0.008 probability of rejecting a password not in the dictionary,

a) What is the size of the hash table in bits? (Please use the formula to calculate it instead of reading from the graph.)
b) What is the size of the dictionary in bytes? (Assuming that 8 bytes are used to store each word)
c) What is the factor of compression achieved?

Problem E. **Select** the appropriate events in a simple challenge-response protocol from the following list and **list** them in the order of occurrence.

____

a) The host authenticates the user if the quantities match, or rejects the user otherwise.
b) The host generates a random number *r*, and returns *r* to the user.
c) The user responds the quantity *f*(*r, h*(P*))* back to the host, where *r* is the nonce, P *is the user's* password.
d) The host generates a random number *r*, and returns *r* and the specification of two functions, *h*() and *f*(), to the user.
e) The host compares the incoming *h*(P) to *h*(P(U)), where *h*(P(U))is the locally stored hash value of user U's password.
f) The user responds the hash value *h*(*r,* P*)* back to the host, where *r* is the nonce, P *is the user's* password.
g) The host compares the incoming *h*(*r*, P) to the calculated *h*(*r*, P(U)), where P(U) is the locally stored user U's password.
h) The user transmits his or her identity, U, to the remote host.

(more on next page!!!)

i) The host compares the incoming $f(r, h(\mathrm{P}))$ to the calculated $f(r, h(\mathrm{P(U)}))$, where $h(\mathrm{P(U)})$is the locally stored hash value of user U's password.

j) The user transmits his or her password, P, to the remote host.

k) The user transmits the hash value of his or her password, $h(\mathrm{P})$, to the remote host.