Homework 3

DUE: November 11, 2015 at 1:30pm

Name__________________________ CSE Login__________________________

Instructions: Follow instructions carefully, failure to do so may result in points being deducted. Clearly label each problem and submit the answers in order. Be sure to show sufficient work to justify your answer(s). If you are asked to prove something, you must give as formal, rigorous, and complete proof as possible. For programming problems, you need to provide a write-up to demonstrate your solutions and programs.

The CSE academic dishonesty policy is in effect (see http://www.cse.unl.edu/undergrads/academic_integrity.php).

Hand in **any electronic copies of artifacts** (code, input/output files, readme files, etc.) through **webhandin**.

Question	Points (Bonus)	Scores
1	40+(20)	
2	20	
3	40	
Total	100+(20)	

1. [40+(20) points] Implement one of the following Hash functions in the high-level programming language of your choice: MD4, MD5, SHA-1, SHA-256, or SHA-3[1]. Make your implementation callable via the command line by providing a file name (the contents of which will be hashed). The resulting digest should be output to the standard output. Note: you may use online tools or other implementations to debug and test your implementation. You may reference the code of other implementations to help you debug your code as long as the referenced code is in a different language than you are implementing it in.

Bonus Points: You will get 20 bonus points if you implement two Hash functions: one in set {MD4, MD5}, the other one in set {SHA-1, SHA-256, SHA-3}.

2. [20 points] In this exercise, you'll use a library to generate rainbow tables for various hash functions for dictionary-based passwords. Use the standard american dictionary file on CSE (located in the file /usr/share/dict/american which contains 305,089 words). Then, generate rainbow tables for MD4, MD5, SHA-1, SHA-256, and SHA3 (Keccak-256 version), PBKDF2 (using the salt, shoop, r = 1000). Your rainbow table files should have the format: hash:password one per line. Finally, they should be presorted lexicographically using the hash for easy look-up via binary search. You may/should collect the resulting files into one compressed zip file for ease of handing them in.

3. [40 points] We've provided several files containing usernames and hashed passwords using MD4, MD5, SHA1, SHA-256, SHA3 (Keccak 256 version), and PBKDF2 (salt: cse477, 10,000 rounds). You will break as many of these passwords as you can using any means and methods you have at your disposal. You could use your rainbow tables, brute-force strategies, online tools, or dedicated password breaking software such as John the Ripper. Break as many of the passwords as you can and document how each one was broken (what techniques or tools broke each password and how fast).

[1] There are slight differences between Keccak and the final SHA-3 standard that is under consideration. You may choose to implement one of the following variants: Keccak-256, Keccak-512, SHA-3:256 or SHA- 3:512

Additional Resources

You may find the following additional resources useful:

- John The Ripper
  - http://www.openwall.com/john/
  - http://pentestmonkey.net/cheat-sheet/john-the-ripper-hash-formats
- MD4 (RFC 1320): http://tools.ietf.org/html/rfc1320
- MD5 (RFC 1321): https://www.ietf.org/rfc/rfc1321.txt
- SHA-1:
  - RFC 3174 http://www.faqs.org/rfcs/rfc3174.html
- SHA-2:
  - NIST Overview: http://csrc.nist.gov/groups/STM/cavp/documents/shs/sha256-384-512.pdf
  - Psuedocode: http://en.wikipedia.org/wiki/SHA-1
- SHA-3:
  - Home Page: http://keccak.noekeon.org/;
  - Reference Paper: http://keccak.noekeon.org/Keccak-reference-3.0.pdf;
  - Implementation Overview: http://keccak.noekeon.org/Keccak-implementation-3.2.pdf
  - Test Vectors: http://keccak.noekeon.org/Keccak-reference-3.0-files.zip
  - Online Calculator: http://www.fishtrap.co.uk/online-sha3/
- Test Vectors for SHA1, 2, 3: http://www.di-mgt.com.au/sha_testvectors.html
- PBKDF2: http://csrc.nist.gov/publications/nistpubs/800-132/nist-sp800-132.pdf