Ryan Stillings

Cryptography and Network Security I

Exam 1 Makeup

1. **[30 points] Hash Functions**: Refer to the example in textbook pg 333-334 (in 7th addition) and implement a “collision attack” to the following wikipedia text: “More efficient attacks are possible by employing cryptanalysis to specific hash functions. When a collision attack is discovered and is found to be faster than a birthday attack, a hash function is often denounced as "broken". The NIST hash function competition was largely induced by published collision attacks against two very commonly used hash functions, MD5 and SHA-1. The collision attacks against MD5 have improved so much that, as of 2007, it takes just a few seconds on a regular computer. Hash collisions created this way are usually constant length and largely unstructured, so cannot directly be applied to attack widespread document formats or protocols.”

   In order to implement a collision attack, I started with an original message and a malicious message. To generate variations of the original message, I replaced space characters in the message with either space, space backspace space, space space backspace, or space space backspace backspace space depending on the iteration number. I then applied the same operation to the malicious message, iterating until I had located a combination with a matching hashes. For the sake of speed I limited hashes to 32-bit md5, however larger bit numbers are fully supported, just a bit slow (Python function execution speed is limiting). Finally, I printed the results to the console, with the raw strings outputted (ordinarily the strings output identically, so I needed to print the raw strings to show the additional space and backspace [\x08] chars). For the example, I used legitimate message = “The quick brown fox jumps over the lazy dog.” And fraudulent message = “The scheming purple fox jumps onto the frightened dog.”, though any messages may be supplied.

   Console Output:

```
>please enter legitimate message (leave blank for default):
>legitimate message detected as 'The quick brown fox jumps over the lazy dog.'
>please enter fraudulent message (leave blank for default):
>fraudulent message detected as 'The scheming purple fox jumps onto the
frightened dog.'
>generating x' variations of legitimate message...
>finished generating 65536 variations of original message in time = 0 seconds
>generating and comparing y' variations of fraudulent message...
>found y' number 14500 = 'The scheming  \x08\x08 purple  \x08fox jumps
\x08onto  \x08the \x08 frightened dog.' with hash 3cc950c9 matching
>x' number 20427 = 'The \x08 quick brown  \x08\x08 fox  \x08\x08 jumps
\x08\x08 over the  \x08lazy  \x08\x08 dog.' with hash 3cc950c9 in time = 18
seconds
```

   For code solution, see hash_functions.py

2. **[30points] Elliptic Curves and ECC**: Solve problems 10-12, 10-13, 10-14, 10-15 from the text book.

   10-12

   10-13

   10-14

   10-15

3. **[40points] Primality and Factorization**: Consider the following integers: 31531; 520482; 485827; 15485863.

   3.a. Implement and check with Miller-Rabin algorithm if they are prime or not

   31531 is prime

   520482 is not prime

   485827 is prime

   3.b. Implement Pollard-Rho method and factor them if they are not prime.

   520482 is not prime; factoring it yields 3

   For code solution, see primality_factorization.py