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**
   Consider the Elliptic curve $E_{11}(1,6)$; that is, the curve is defined by $y^2 = x^3 + x + 6$ with a modulus of $p = 11$. Determine all of the points in $E_{11}(1,6)$. Hint: Start by calculating the right-hand side of the equation for all values of x

| x | $x^3$+x+6 mod 11 | q |
|---|---|---|
| 1 | 8 | N/A |
| 2 | 5 | (4,7) |
| 3 | 3 | (5,6) |
| 4 | 8 | N/A |
| 5 | 4 | (2,9) |
| 6 | 8 | N/A |
| 7 | 4 | (2,9) |
| 8 | 9 | (3,8) |
| 9 | 7 | N/A |
| 10 | 4 | (2,9) |

Points = (2,4), (2,7), (3,5), (3,6), (5,2), (5,9), (7,2), (7,9), (8,3), (8,8), (10,2), (10,9)

**10-13**

What are the negatives of the following elliptic curve points over $Z_{17}$? P = (5,8); Q = (3,0); R = (0,6).

| | |
|---|---|
| -P | (5,9) |
| -Q | (3,0) |
| -R | (0,11) |

**10-14**

For $E_{11}(1,6)$, consider the point G = (2,7). Compute the multiples of G from 2G through 13G.

| G | m | Result |
|---|---|---|
| 2 | $\frac{3*2^2+1}{2*7}=8$ | (5,2) |
| 3 | $\frac{2-7}{5-2}=2$ | (8,3) |
| 4 | ..3 | (10,2) |
| 5 | ..9 | (3,6) |
| 6 | ..10 | (7,9) |
| 7 | ..7 | (7,2) |
| 8 | ..10 | (3,5) |
| 9 | ..9 | (10,9) |
| 10 | ..3 | (8,8) |
| 11 | ..2 | (5,9) |
| 12 | ..8 | (2,4) |
| 13 | ..8 | (5,2) |

**10-15**

This problem performs elliptic curve encryption/decryption using the scheme outlined in Section 10.4. The cryptosystem parameters are $E_{11}(1,6)$ and G = (2,7). B's private key is $n_B$ = 7.

a. Find B's public key $P_b$

   $P_B = n_B*G$ = (7,2)

b. A wishes to encrypt the message $P_m$ = (10,9) and chooses the random value k = 3. Determine the ciphertext $C_m$.

   $C_m = kG, P_m + k*P_B$ = (8,3),(10,2)

c. Show the calculation by which B recovers $P_m$ from $C_m$.

   $P_m = C_2 - n_B * C_1$ = (10,9)

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

15485863 is prime

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

31531 has no factors

520482 has a factor of 3

485827 has no factors

15485863 has no factors

For code solution, see primality_factorization.py