Cryptography & Network Security I - Fall 2018

Make-Up Exam Written Answers

Charles Schmitter

October 29, 2018

Q1.
For this question, the collision attack was implemented in *question1.py.* In short, this implementation demonstrates the possibility of an attacker knowing the patterns of a hash function. In particular, the hash function that I implemented builds a hash value based on every fourth character in a string. Once an attacker finds this pattern, he/she can manipulate data to abuse hash collisions.

For example, in a message passing example, User A may wish to communicate to User B to meet up somewhere at a specified time. An adversary, User C, can intercept this message through a man-in-the-middle attack, and manipulate this message to have the same hash as the original message, but with different message content. For example, in my sample implementation, the messages 'meet5pm!' and 'dontcome' will produce a hash collision. Therefore, unsuspecting User B will assume that the imposter message came from User A since the hash value is correct.

This pattern of manipulating data around a known pattern of a hash function exists in commonly used hash functions such as MD5 and SHA-1. These patterns can even be faster to perform than a birthday attack. In the case of MD5, there exists the following pattern:
If $MD5(A) = MD5(B)$, then $MD5(A + C) = MD5(B + C)$ where + represents concatenation, and A, B, and C are inputs to the MD5 hash function. Therefore, if one can alter the first section of the original message to his/her liking, and then alter it to produce the same hash value as the original, then one can simply add the rest of the original message since it is a common suffix to both the original and the altered message.

Q2.

10.12

X	output	Y
0	6	none
1	8	none
2	5	4, 7
3	3	5, 6
4	8	none
5	4	2, 9
6	8	none
7	4	2, 9
8	9	3, 8
9	7	none
10	4	2, 9

Therefore, the points are: $(2,4); (2,7); (3,5); (3,6); (5,2); (5,9); (7,2); (7,9); (8,3); (8,8); (10,2); (10,9);$

10.13
To find the negative of a point $P(X_P, Y_P)$, we can use the equation: $-P = (X_P, -Y_P)$. Once we have negated $-Y_P$, we use modulo 17 since we are in Z_17. Thus, we can map each given point to its negative:
$P(5,8) \rightarrow -P(5,9)$
$Q(3,0) \rightarrow -Q(3,0)$
$R(0,6) \rightarrow -R(0,11)$

10.14
Given $G(2,7)$, we first find $2G = (10,0)$. When we attempt to compute $3G = 2G + G$, we find that $3G$ is equivalent to $(2,7)$. Therefore, we can deduce that for any kG where k is odd, $kG = (2,7)$, and for any kG where k is even, $kG = (10,0)$. So:
$1G = (2,7)$
$2G = (10,0)$
$3G = (2,7)$
$4G = (10,0)$
$5G = (2,7)$
$6G = (10,0)$
$7G = (2,7)$
$8G = (10,0)$
$9G = (2,7)$
$10G = (10,0)$
$11G = (2,7)$
$12G = (10,0)$
$13G = (2,7)$

10.15
We first compute B's public key with $P_B = n_b * G$. Here, we notice we have $7(2,7)$ in the same field as the previous problem. Thus, we can deduce that $7(2,7)$ is equivalent to $(2,7)$. We can then compute the ciphertext with $C_M = \{kG, P_M + kP_B\}$. We then find $C_M = \{(2,7),(8,8)\}$. For (c) of this problem, we must now compute the decryption of C_M. We therefore must compute $(8,8) - 7(2,7)$, where the 7 is the private key of B. This subtraction is equivalent to $(8,8) + (2,4)$. We can do a simple ECC point addition, and our result is $(10,9)$, thus revealing the original plaintext as given by the problem.

Q3.

For this question, the Miller-Rabin and Pollard-Rho algorithms were implemented (found in *question3.py*). The answers found with this script are written below.

(a) Of the four numbers given:
The prime numbers are: $31531; 485827; 15485863$
The composite number is: 520482

(b) The composite number 520482 has factors: $2; 3; 223; 389$