Zelong Li

24569650

Discussion 102

TA: Qi Zhong

CS161 Homework 2

1.

a.

500-1 mod 10007

10007 = (500) 20 + 7

500 = (7) 71 + 3

7 = (3) 2 + 1

Therefore, we have the following:

1 = 7 – 3 2

= 7 – 2 (500 – 7 71)

= 7 143 – 2 500

= (10007 – 500 20) 143 – 2 500

= 143 10007 – 2862 500

Finally, we have:

500-1 (-2862) mod 10007

Thus, 500-1 7145 mod 10007

b.

From Euler-Fermat theorem, we know:

500(10007-1) 1 mod 10007

50010006 1 mod 10007

Therefore, we have:

50050010005 1 mod 10007

And 50010005 is the multiplicative inverse of 500 in modulo 10007

5001 500 mod 10007

5002 9832 mod 10007

5004 98322 mod 10007 604 mod 10007

5008 6042 mod 10007 4564 mod 10007

50016 45642 mod 10007 5529 mod 10007

50032 55292 mod 10007 8463 mod 10007

50064 84632 mod 10007 2270 mod 10007

500128 22702 mod 10007 9302 mod 10007

500256 93022 mod 10007 6682 mod 10007

500512 66822 mod 10007 7897 mod 10007

5001024 78972 mod 10007 8992 mod 10007

5002048 89922 mod 10007 9511 mod 10007

5004096 95112 mod 10007 5848 mod 10007

5008192 58482 mod 10007 5185 mod 10007

50010005 500(1+4+16+256+512+1024+8192) mod 10007 500\*604\*5529\*6682\*7897\*8992\*5185 mod 10007 7145 mod 10007

Thus, 500-1 7145 mod 10007 (19 multiplication used).

2.

a.

According to this protocol, A sends m’ = mr mod p to B and B sends m’’ = (m’)t mod p = mrt mod p to A; then, A sends m’’’ = (m’’)s mod p = (mrt)s mod p = mrst mod p to B. When B receives the message, the message he gets is mrst mod p. Since B has the value of u, he can verify the message with u. When B verifies the message, he needs to compute (m’’’)u mod p = mrstu mod p.

We already know:

rs = 1 (mod p-1) and tu = 1(mod p - 1);

Thus, we let rs = k\*(p-1) + 1 and tu = l\*(p-1) + 1 for k, l ­­p

Therefore, mrs = mk\*(p-1)+1mod p = mk\*(p-1) m mod p = (mp-1)k m mod p;

According to Fermat-Euler Theorem, mp-1 1 mod p.

Then, we have (mp-1)k m mod p = m mod p and therefore mrs = m mod p;

Similarly, mtu= m mod p.

Thus, mrstu = (mrs)tu mod p = mtu mod p = m mod p.

Also, if another guy other than A or B captures the encrypted message, he would get mr mod p, mrt mod p, or mrts mod p. If he gets mr mod p, without knowing r or s, he would not be able to recover m. If he gets mrt mod p, without knowing rt or su, he would not be able to recover m. If he gets mrts mod p, without knowing rts or u, he would not be able to recover m. Also, since compute discrete logarithm is hard, third party would not be able to compute s from these messages. Thus, the protocol’s encryption and decryption work.

Because of that, B can recover the message that A sends to him and this protocol works.

b.

Suppose the man in the middle is called Mallory (M). When A wants to send B a message, she sends mr mod p to B. However, M intercepts and captures the message mr mod p. M also picks a random x ­­p ­and sets y such that xy = 1(mod p-1). Then M forges a message n and sends nx mod p to B. Since M is man in the middle, B can’t know the message he receives gets forged. Then, B tries to nxt mod p to A. But M still captures the message and sends mrx mod to A instead. After A receives the message, she then sends mrxs mod p = (mrs)x mod p back. This time, M again gets the message and he can compute mrsxy mod p to recover A’s message (according to part a). At the same time, M can send nxty mod p to B. B then recovers the message by nxytu mod p and therefore recovers the forged message n.

c.

When, A sends m’ = mr mod p to B, the eavesdropper gets the message m’. And then, B sends m’’ = (m’)t mod p to A, the eavesdropper again gets the message m’’. Then when A sends m’’’ = (m’’)s mod p back to B, the eavesdropper captures the message m’’’. If this eavesdropper can compute discrete logarithm, he could compute s from m’’’ = (m’’)s mod p and m’’. Now, after the eavesdropper gets s, he could simply compute (m’)s mod p = mrs mod p = m and therefore recovers the message (same reasoning as part a). Thus, if an eavesdropper can compute discrete logarithm, this protocol no longer works.

3.

a.

h’() is not pre-image resistant. When 0 2n, we get a h’(x). In this situation, if we ignore the left-padded zeros, we get the binary representation of x and therefore we get x. Thus, h’() is not pre-image resistant.

b.

h’() is second pre-image resistant. Given x, we can calculate h’(x). If 0 2n, we get a binary representation of x left-padded with zeros. In this case, in order to have h’(y) = h’(x), y and x have to have the same binary representation and therefore y = x. If 2n h’(x) we get is just 1 concatenating with h(x). In this situation, h(x) is second pre-image resistant, so we still can’t find y x such that h(x) = h(y) to make h’(y) = h’(x). In general, h’() is second pre-image resistant.

c.

h’() is collision resistant. Without loss of generality, assume 0 2n and 2n In this case, h’(x) outputs an n bits string starting with 0. However, h’(y) outputs an n bits string starting with 1. It’s obvious that h’(x) h’(y). If 0 2n and 0 2n with x , since x and y are not equal, they have different binary representation. Therefore, h’(x) h’(y). If 2n and 2n since h() is collision resistant, h(x) h(y) and therefore h’(x) h’(y). Thus, in general, h’() is collision resistant.

d.

Second pre-image resistance does not imply pre-image resistance. In this question, our hash function is a good example, because it’s second pre-image resistant but not pre-image resistant. In a general case, if h() is second pre-image resistant, which means given x, we can’t find y x such that h(x) = h(y). But even if so, if the hash function’s output is unique with unique input, it is possible to get the input with its output. Another example is h(x) = x.

e.

Collision resistance does not imply pre-image resistance. Again, the hash function in this question is an example that is collision resistant but not pre-image resistant. Collision resistance just makes sure that unequal inputs would not have the same output value. But even if so, if the hash function’s output is unique with unique input, it is possible to get the input with its output. Another example is h(x) = x.