Homework 7 Chapter 6 Questions

Mudit Vats
mpvats@syr.edu
6/2/2020

Table of Contents

Overview

This lab report presents observations and explanations for the exercises described in Professor Weissman's HW7 assignment.

6.1

As we have seen in this chapter, public-key cryptography can be used for encryption and key exchange. Furthermore, it has some properties (such as nonrepudiation) which are not offered by secret key cryptography.

So why do we still use symmetric cryptography in current applications?

The primary reason is that **symmetric is significantly faster** computation-wise as compared to asymmetric public-key cryptography. We learned this in class and it was even reiterated in our last 6/2 lecture. So, we exchange or derive the symmetric key via asymmetric cryptography and then do all of the bulk data or traffic encryption/decryption with symmetric.

6.2

In this problem, we want to compare the computational performance of symmetric and asymmetric algorithms. Assume a fast public-key library such as OpenSSL [132] that can decrypt data at a rate of 100 Kbit/sec using the RSA algorithm on a modern PC. On the same machine, AES can decrypt at a rate of 17 Mbit/sec. Assume we want to decrypt a movie stored on a DVD. The movie requires 1 GByte of storage.

How long does decryption take with either algorithm?

Given:

- Data Size = 1 GByte = 8000 Mbit
- RSA, Asymmetric, 100 Kbit/sec = .1 Mbit/sec
- AES, Symmetric, 17 Mbit/sec

Decryption:

- **RSA**: 8000 Mbit / (.1 Mbit / sec) => **80000.000 seconds**
- **AES**: 8000 Mbit / (17 Mbit / sec) => **470.588 seconds**

Clearly, AES, which is a symmetric encryption / decryption algorithm is much faster.

6.3

Assume a (small) company with 120 employees. A new security policy demands encrypted message exchange with a symmetric cipher. How many keys are required, if you are to ensure a secret communication for every possible pair of communicating parties?

When we use a symmetric cipher, the same key is used for encryption and decryption. In this case, each employee would need a shared key for that employee and all other employees.

Per our class book (Understanding Cryptography by Christof Par and Jan Pelzl, page 151), the number of keys for each pair of n users is:

- n * (n – 1) / 2

So, for n=120 employees, we would have:

- 120 * (120 – 1) / 2 => **7140 keys**

6.5

Using the basic form of Euclid's algorithm, compute the greatest common divisor

of

1. 7469 and 2464

2. 2689 and 4001

For this problem use only a pocket calculator. Show every iteration step of Euclid's algorithm, i.e., don't write just the answer, which is only a number. Also, for every gcd, provide the chain of gcd computations, i.e.,

gcd(r0, r1) = gcd(r1, r2) = · · · .

7469 and 2464

- gcd(r0, r1) = gcd(r0 mod r1, r1), swap lower/higher with lower on right.
- gcd(7469, 2464) = gcd(7469 mod 2464, 2464)
- gcd(2464, 77) = gcd(2464 mod 77, 77)
- gcd(77, 0) = 77

2689 and 4001

- gcd(r0, r1) = gcd(r0 mod r1, r1), swap lower/higher with lower on right.
- gcd(4001, 2689) = gcd(4001 mod 2689, 2689)
- gcd(2689, 1312) = gcd(2689 mod 1312, 1312)
- gcd(1312, 65) = gcd(1312 mod 65, 65)
- gcd(65, 12) = gcd(65 mod 12, 12)
- gcd(12, 5) = gcd(12 mod 5, 5)
- gcd(5, 2) = gcd(5 mod 2, 2)
- gcd(2, 1) = gcd(2 mod 1, 1)
- gcd(1, 0) = 1

6.6

Using the extended Euclidean algorithm, compute the greatest common divisor and the parameters s, t of

1. 198 and 243
2. 1819 and 3587

For every problem check if sr0+t r1 = gcd(r0, r1) is actually fulfilled. The rules are the same as above: use a pocket calculator and show what happens in every iteration step.

198 and 243

gcd(243, 198) = s*243 + t*198

Euclidean	Extended Euclidean
r0=243 r1=198 243 = a * 198 + r2 243 = 1 * 198 + 45; r2=45	 r2=45= [s]243 + [t]198 r2=45= [1]243 + [-1]198
198 = a * 45 + r3 243 = 5 * 45 + 18; r3=18	r3=18= [s]198 + [t]45 r3=18= [1]198 + [-4]45 Substituting r2, then expressing in original r0 and r1 -- r3=18= 1*198 + -4* (1*243 -1*198) r3=18= 1*198 - 4*243 + 4*198 r3=18= -4*243 + 5*198 r3=18= [-4]243 + [5]198
45 = a * 18 + r4 45 = 2 * 18 + 9; r4=9	r4=9= [s]45 + [t]18 r4=9= [1]45 + [-2]18 Substituting r2 and r3, then expressing in original r0 and r1 -- r4=9= 1*45 + -2*18 r4=9= 1*([1]243 + [-1]198) + -2*([-4]243 + [5]198) r4=9= 1*243 + -1*198 + 8*243 + -10*198 r4=9= [9]243 + [-11]198
18 = a * 9 + r5 18 = 2 * 9 + 0; r5=0	r5=0= [s]18 + [t]9 r5=0= [1]18 + [-2]9 Substituting r3 and r4, then expressing in original r0 and r1 -- r5=0= [1]([-4]243 + [5]198) + [-2]([9]243 + [-11]198) r5=0= (-4*243 + 5*198) + (-18*243 + 22*198)

	r5=0= [22]243 + [-27]198 s=9 t=-11 gcd(243, 198)=9

1819 and 3587

gcd(3587, 1819) = s*3587 + t*1879

Euclidean	Extended Euclidean
r0=3587 r1=1819 3587= a * 1819+ r2 3587= 1 * 1819+ 1768; r2=1768	 r2=1768= [s] 3587+ [t] 1819 r2=1768= [1] 3587+ [-1] 1819
1819 = a * 1768 + r3 1819 = 1 * 1768 + 51; r3=51	r3=51= [s]1819 + [t]1768 r3=51= [1]1819 + [-1]1768 Substituting r2, then expressing in original r0 and r1 -- r3=51= [1]1819 + [-1]1768 r3=51= 1*1819 + -1*([1] 3587+ [-1] 1819) r3=51= [-1] 3587 + [2]1819
1768 = a * 51 + r4 1768 = 34 * 51 + 34; r4=34	r4=34= [s]1768 + [t]51 r4=34= [1]1768 + [-34]51 Substituting r2 and r3, then expressing in original r0 and r1 -- r4=34= [1]([1] 3587+ [-1] 1819) + [-34]([-1] 3587 + [2]1819) r4=34= [35]3587+ [-69]1819
51 = a * 34 + r5 51 = 1 * 34 + 17; r5=17	r5=17= [s]51 + [t]34 r5=17= [1]51 + [-1]34 Substituting r3 and r4, then expressing in original r0 and r1 -- r5=17= [1]51 + [-1]34

34 = a * 17 + r6 34 = 2 * 17 + 0; r6=0	r5=17= [1]([-1] 3587 + [2]1819) + [-1]([35]3587+ [-69]1819) r5=17= ([-1] 3587 + [2]1819) + ([-35]3587+ [69]1819) r5=17= ([-36]3587 + [71]1819) s=-36 t=71 gcd(3587, 1819)=17

6.7

With the Euclidean algorithm we finally have an efficient algorithm for finding the multiplicative inverse in Zm that is much better than exhaustive search. Find the inverses in Zm of the following elements a modulo m:

1. a = 7, m = 26 (affine cipher)

2. a = 19, m = 999

Note that the inverses must again be elements in Zm and that you can easily verify your answers.

a = 7, m = 26 (affine cipher)

gcd(7, 26) = s*26 + t*7

Euclidean	Extended Euclidean
r0=26 r1=7 26= a * 7 + r2 26= 3 * 7 + 5; r2=5 7 = a * 5 + r3 7 = 1 * 5 + 2; r3=2	 r2=5= [s]26 + [t]7 r2=5= [1]26 + [-3]7 r3=2= [s]7 + [t]5 r3=2= [1]7 + [-1]5

	Substituting r2, then expressing in original r0 and r1 -- r3=2= [1]7 + [-1]([1]26 + [-3]7) r3=2= [-1]26 + [4]7
5 = a * 2 + r4 5 = 2 * 2 + 1; r4=1	r4=1= [s]5+ [t]2 r4=1= [1]5+ [-2]2 Substituting r2 and r3, then expressing in original r0 and r1 -- r4=1= [1]5+ [-2]2 r4=1= [1]([1]26 + [-3]7)+ [-2]([-1]26 + [4]7) r4=1= [3]26 + [-11]7
2 = a * 1 + r5 2 = 2 * 1 + 0; r5=0	
	s=26 t=-11 gcd(7, 26)=1 The modular multiplicative inverse is t, which is -11 = 15 (since -11 + 26 => 15). So.... 7^{-1}=-11 mod 26 => 15

a = 19, m = 999

gcd(19, 999) = s*999 + t*19

Euclidean	Extended Euclidean
r0=999 r1=19 999= a * 19 + r2 999= 52 * 19 + 11; r2=11	r2=11= [s]999 + [t]19 r2=11= [1]999 + [-52]19
19 = a * 11 + r3 19 = 1 * 11 + 8; r3=8	r3=8= [s]19 + [t]11 r3=8= [1]19 + [-1]11 Substituting r2, then expressing in original r0 and r1 -- r3=8= [1]19 + [-1]11 r3=8= [1]19 + [-1]([1]999 + [-52]19)

11 = a * 8 + r4 11 = 1 * 8 + 3; r4=3	r3=8= [-1]999 + [53]19 r4=3= [s]11 + [t]8 r4=3= [1]11 + [-1]8 Substituting r2 and r3, then expressing in original r0 and r1 -- r4=3= [1]11 + [-1]8 r4=3= [1]([1]999 + [-52]19) + [-1]([-1]999 + [53]19) r4=3= [2]999 + [-105]19
8 = a * 3 + r5 8 = 2 * 3 + 2; r5=2	r5=2= [s]8 + [t]3 r5=2= [1]8 + [-2]3 Substituting r3 and r4, then expressing in original r0 and r1 -- r5=2= [1]8 + [-2]3 r5=2= [1]([-1]999 + [53]19) + [-2]([2]999 + [-105]19) r5=2= [-5]999 + [263]19
3 = a * 2 + r6 3 = 1 * 2 + 1; r6=1	r6=1= [s]3 + [t]2 r6=1= [1]3 + [-1]2 Substituting r4 and r5, then expressing in original r0 and r1 -- r6=1= [1]3 + [-1]2 r6=1= [1]([2]999 + [-105]19) + [-1]([-5]999 + [263]19) r6=1= [7]999 + [-368]19)
2 = a * 1 + r7 3 = 3 * 1 + 0; r7=0	s=7 t=-368 gcd(19,999)=1 The modular multiplicative inverse is t, which is -368 = 631 (since -368 + 999 => 631). So…. 19^{-1}=-368 mod 999 => 631