# CMPE-552 Database and File Security

# Problem Session 04.11.2019

Cryptography, both symmetric and asymmetric, widely uses number theory, in particular, relative primality, modular arithmetic, and multiplicative inverses. That’s why we begin with them.

1. Primality, Greatest Common Divisor (GCD), Euclidean Algorithm

Prime number is one having no other factors except one and itself, e.g. N=7 is a prime number

Relatively prime are such two numbers that have no common factors except one

Greatest Common Divisor (GCD) is the maximal common factor for two numbers

For example, gcd(12,15)=3

GCD can be obtained by factoring the numbers and comparing them

For example, 12=2x2x3, 15=3x5, hence, gcd(12,15)=3

Euclidean algorithm provides straightforward method of finding gcd without necessity of finding factors

EUCLID(a,b)

1. A:=a; B:=b
2. if B=0 return A=gcd(a,b)
3. R=A mod B
4. A:=B
5. B:=R
6. goto 2

The algorithm has the following progression:

A1=B1xQ1+R1

A2=B2xQ2+R2

A3=B3xQ3+R3

*To find gcd(1970,1066)*

*1970=1x1066+904 gcd(1066,904)*

*1066=1x904+162 gcd(904,162)*

*904=5x162+94 gcd(162,94)*

*162=1x94+68 gcd(94,68)*

*94=1x68+26 gcd(68,26)*

*68=2x26+16 gcd(26,16)*

*26=1x16+10 gcd(16,10)*

*16=1x10+6 gcd(10,6)*

*10=1x6+4 gcd(6,4)*

*6=1x4+2 gcd(4,2)*

*4=2x2+0 gcd(2,0)*

*Therefore, gcd(1970,1066)=2*

Given any positive integer n and any integer a, if we divide a by n, we get an integer quotient q and an integer remainder r that obey the following relationship:

a=qn+r 

where  is the largest integer less than or equal to x.



The remainder r is often referred to as a residue. Let Zn ={0,1,..,n-1}.



In general, an integer has a multiplicative inverse in Zn, if that integer is relatively prime to n. Table 4.1c shows that the integers 1, 3, 5, and 7 have a multiplicative inverse, but 2, 4, and 6 do not.



1. Multiplicative inverse, Extended Euclid



If gcd(m,b)=1, then b has a multiplicative inverse modulo m. That is, for positive integer b<m, there exists a b-1<m such that b b-1=1 mod m. Euclid’s algorithm can be extended so that, in addition to finding gcd(m,b), if the gcd is 1, the algorithm returns the multiplicative inverse of b.

EXTENDED EUCLID(m,b)

1. (A1,A2,A3):=(1,0,m); (B1,B2,B3):=(0,1,b);
2. if B3=0 return A3=gcd(m,b); no inverse
3. if B3=1 return B3 = gcd(m,b); B2= b-1 mod m
4. Q=
5. (T1,T2,T3):=(A1-QB1, A2-QB2, A3-QB3)
6. (A1,A2,A3):= (B1,B2,B3)
7. (B1,B2,B3):= (T1,T2,T3)
8. goto 2

Throughout the computation, the following relationships hold:

mT1+bT2=T3 mA1+bA2=A3 mB1+bB2=B3

To see that algorithm correctly returns gcd(m,b), note that if we equate A and B in Euclid’s algorithm with A3 and B3 in the extended Euclid’s algorithm, then the treatment of the two variables is identical. Note also that if gcd(m,b)=1, then on the final step we would have B3=0 and A3 =1. Therefore, on the preceding step, B3=1. But if B3=1, then we can say the following:

mB1+bB2=B3

mB1+bB2=1

bB2=1-mB1

bB21 mod m

Hence, B2 is the multiplicative inverse of b.

Table 4.4 is an example of the execution of the algorithm. It shows that gcd(550,1759)=1 and that the multiplicative inverse of 550 is 355; that is, 550x3551 mod 1759.



1. Simple symmetric ciphers

Caesar Cipher

c=Ek(p)=(p+k) mod 26

Each plaintext letter p of the English alphabet is encrypted by letter c using the secret key k

Decryption is made as follows

p=Dk(c)=(c-k) mod 26

Key has 25 different values, and the plaintext can be easily revealed by a brute-force attack.

For example

*Plain: meet me after the toga party*

*Cipher: PHHW PH DIWHU WKH WRJD SDUWB*

Transformation is made using the following mapping:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | *0* | *1* | *2* | *3* | *4* | *5* | *6* | *7* | *8* | *9* | *10* | *11* | *12* | *13* | *14* | *15* | *16* | *17* | *18* | *19* | *20* | *21* | *22* | *23* | *24* | *25* |
| *Plain:* | *a* | *b* | *c* | *d* | *e* | *f* | *g* | *h* | *i* | *j* | *k* | *l* | *m* | *n* | *o* | *p* | *q* | *r* | *s* | *t* | *u* | *v* | *w* | *x* | *y* | *z* |
| *Cipher:* | *D* | *E* | *F* | *G* | *H* | *I* | *J* | *K* | *L* | *M* | *N* | *O* | *P* | *Q* | *R* | *S* | *T* | *U* | *V* | *W* | *X* | *Y* | *Z* | *A* | *B* | *C* |

If instead, the “cipher” line can be any permutation of the 26 alphabetic characters, then there are 26! (greater than 4x10^26) possible keys. Such an approach is called monoalphabetic substitution cipher. Such a cipher can be attacked by use of information about frequencies of appearance of particular alphabetic symbols in texts.

1. Polyalphabetic ciphers – Vigenere cipher

$$c\_{i}=p\_{i}+k\_{i} mod 26$$

$$p\_{i}=c\_{i}-k\_{i} mod 26$$

where key $k=(k\_{1}..k\_{n})$. Plaintext=”message”, key=”secret” = (18,4, 2, 17,4, 19), ciphertext?

C1=p1+k1=12+18=30 mod 26=4=’E’

C2=p2+k2=4+4=8 mod 26 =8 =’I’

C3=p3+k3=18+2=20 mod 26 = 20 = ‘U’

C4=p4+k4=18+17=35 mod 26 = 9 =’J’

C5=p5+k5=0+4=4 mod 26 = 4 = ‘E’

C6=p6+k6=6+19=25 mod 26 = ‘Z’

C7=p7+k7=4+18=22 mod 26 =’W’

Ciphertext=”EIUJEZW”

p1=c1-k1=4-18=-14 mod 26=12=’m’

p2=c2-k2=8-4=4 mod 26 =4 =’e’

p3=c3-k3=20-2=18 mod 26 = 18 = ‘s’

p4=c4-k4=9-17=-8 mod 26 = 18 =’s’

p5=c5-k5=4-4=0 mod 26 = 0 = ‘a’

p6=c6-k6=25-19=6 mod 26=6 = ‘g’

p7=c7-k7=22-18=4 mod 26=4 =’e’

1. Multiplicative cipher

$$c\_{i}=k∙p\_{i} mod 26$$

$$p\_{i}=k^{-1}∙c\_{i} mod 26$$

P=12, k=11, c=11\*12 mod 26 = 132 mod 26 = 2

A=(1,0,26), B=(0,1,11)

$$q=\left⌊\frac{26}{11}\right⌋=2$$

T=A-qB=(1-2\*0,0-2\*1,26-2\*11)=(1,-2,4)

A=(0,1,11), B=(1,-2,4)

$$q=\left⌊\frac{11}{4}\right⌋=2$$

T=A-qB=(0-2\*1,1-2\*(-2),11-2\*4)=(-2,5,3)

A=(1,-2,4), B=(-2,5,3)

$$q=\left⌊\frac{4}{3}\right⌋=1$$

T=A-qB=(1-1\*(-2),-2-1\*5,4-1\*3)=(3,-7,1)

$$k^{-1}=-7 mod 26=19$$

Check it:

$$k^{-1}∙k=19∙11 mod 26=209 mod 26=8∙26+1 mod 26=1$$

$$p=k^{-1}∙c mod 26=19∙2 mod 26=38 mod 26=12$$

1. Types of attacks? What are passive attacks? What are active attacks?
2. How does virtual memory support memory protection? How is provided separation of processes?
3. What is access control? What types of access control do you know? What is user-oriented access control? What is data oriented access control? What is access matrix? What is access control list? What is capability list (ticket)? How these are related to each other?
4. What technique is used for intrusion?
5. Why “salt” is kept in clear in Unix password scheme?
6. What are password selection strategies and what are their deficiencies?
7. How intrusion can be detected?
8. What is audit? What information is usually kept in audit file records?
9. What is malicious software? What are the main virus types (Parasitic, Memory-resident, Boot sector, Stealth, Polymorphic, Macro)? What are the life steps of virus?
10. What is multilevel security? What are no read-up and no write-down rules? What is the use of the reference monitor? How can multilevel security system thwart Trojan horse attack?
11. What are the main sources of security vulnerabilities of software?
12. What does it mean “data consistency”?
13. Explain primary key constraint. How it can be checked?
14. Explain foreign key referential integrity constraint. How it can be checked?
15. Explain mapping cardinality constraints for binary relations. How mapping cardinality constraints can be checked?