**Nova Southeastern University**

**College of Computing and Engineering**

**ISEC 620 Applied Cryptography**

**Fall 2020**

**(August 17 – December 6, 2020)**

**Final Examination**

**Due Date: December 1, 2020 (Firm)**

Instructor: Dr. Junping Sun

**Name: ERIC WEBB NSU-ID:N01927543 Score:**

1. (30 points)
2. Encrypt the message “meet me at the usual place at ten rather than eight oclock” using the Hill Cipher with the key .

Show your calculations and the result.

To Encrypt (A=0, B=1, …through ... Y=24, and Z=25)

Since this encryption is based off pairs and the string length is odd, some padding needed to be chosen for the plaintext.

In this example we added padding to make the string even by appending a character “**a**” to the string, but any character can be chosen.

New clear text string = “**meet me at the usual place at ten rather than eight oclocka**”.

Starting with “**me**” in “**me**et me at the usual place at ten rather than eight oclocka”.

= mod 26 = =

Then with “**et**” in “me**et** me at the usual place at ten rather than eight oclocka”.

= mod 26 = =

Then with “**me**” in “meet **me** at the usual place at ten rather than eight oclocka”.

, this is shown in the first step using “**me**”.

Then with “**at**” in “meet me **at** the usual place at ten rather than eight oclocka”.

= mod 26 = =

Then with “**th**” in “meet me at **th**e usual place at ten rather than eight oclocka”.

= mod 26 = =

Then with “**eu**” in “meet me at th**e** **u**sual place at ten rather than eight oclocka”.

= mod 26 = =

Then with “**su**” in “meet me at the u**su**al place at ten rather than eight oclocka”.

= mod 26 = =

Then with “**al**” in “meet me at the usu**al** place at ten rather than eight oclocka”.

= mod 26 = =

Then with “**pl**” in “meet me at the usual **pl**ace at ten rather than eight oclocka”.

= mod 26 = =

Then with “**ac**” in “meet me at the usual pl**ac**e at ten rather than eight oclocka”.

= =

Then with “**ea**” in “meet me at the usual plac**e** **a**t ten rather than eight oclocka”.

= mod 26 = =

Then with “**tt**” in “meet me at the usual place a**t** **t**en rather than eight oclocka”.

= mod 26 = =

Then with “**en**” in “meet me at the usual place at t**en** rather than eight oclocka”.

= mod 26 = =

Then with “**ra**” in “meet me at the usual place at ten **ra**ther than eight oclocka”.

= mod 26 = =

Then with “**th**” in “meet me at the usual place at ten ra**th**er than eight oclocka”.

= , this is shown in the first step using “**th**”.

Then with “**er**” in “meet me at the usual place at ten rath**er** than eight oclocka”.

= mod 26 = =

Then with “**th**” in “meet me at the usual place at ten rather **th**an eight oclocka”.

= , this is shown in the first step using “**th**”.

Then with “**an**” in “meet me at the usual place at ten rather th**an** eight oclocka”.

= mod 26 = =

Then with “**ei**” in “meet me at the usual place at ten rather than **ei**ght oclocka”.

= mod 26 = =

Then with “**gh**” in “meet me at the usual place at ten rather than ei**gh**t oclocka”.

= mod 26 = =

Then with “**to**” in “meet me at the usual place at ten rather than eigh**t o**clocka”.

= mod 26 = =

Then with “**cl**” in “meet me at the usual place at ten rather than eight o**cl**ocka”.

= mod 26 = =

Then with “**oc**” in “meet me at the usual place at ten rather than eight ocl**oc**ka”.

= mod 26 = =

Then with “**ka**” in “meet me at the usual place at ten rather than eight ocloc**ka**”.

= mod 26 = =

Putting that all together the encrypted message for “meet me at the usual place at ten rather than eight oclocka” is the encrypted message is “**ukixukydromeiwszvwiokunukhxhroajroanqyebtlkjegmy**” Using the Hill Cypher.

1. Show the calculations for corresponding decryption of the ciphertext to recover the original plaintext.

To decrypt a matrix inversion must be performed to find the determinate of the encryption matrix.

det = ad -bc, so that det = 43.

43 mod 26 = 17.

(17-1)mod26=23

Because (17\*23)mod26 = 1 we can now calculate the inverse of the key matrix.

-1mod 26

=1/9mod 26

=9-1 mod 26

=23 mod 26

= mod 26

=

**The inverse key of**

Now that we have the inverse key we can decrypt with the same means before for the string “**ukixukydromeiwszvwiokunukhxhroajroanqyebtlkjegmy**” and .

Starting with “**uk**” in “**uk**ixukydromeiwszvwiokunukhxhroajroanqyebtlkjegmy”.

= mod 26 = =

Then with “ix” in “uk**ix**ukydromeiwszvwiokunukhxhroajroanqyebtlkjegmy”.

= =

Then with “**uk**” in “ukix**uk**ydromeiwszvwiokunukhxhroajroanqyebtlkjegmy”.

= from previous logic.

Then with “**yd**” in “ukixuk**yd**romeiwszvwiokunukhxhroajroanqyebtlkjegmy”.

mod 26 = =

Then with “**ro**” in “ukixukyd**ro**meiwszvwiokunukhxhroajroanqyebtlkjegmy”.

= =

Then with “ **me**” in “ukixukydro**me**iwszvwiokunukhxhroajroanqyebtlkjegmy”.

mod 26 = =

Then with “**iw**” in “ukixukydrome**iw**szvwiokunukhxhroajroanqyebtlkjegmy”.

mod 26 = =

Then with “**sz**” in “ukixukydromeiw**sz**vwiokunukhxhroajroanqyebtlkjegmy”.

mod 26 = =

Then with “**vw**” in “ukixukydromeiwsz**vw**iokunukhxhroajroanqyebtlkjegmy”.

mod 26 = =

Then with “**io**” in “ukixukydromeiwszvw**io**kunukhxhroajroanqyebtlkjegmy”.

mod 26 = =

Then with “**ku**” in “ukixukydromeiwszvwio**ku**nukhxhroajroanqyebtlkjegmy”.

mod 26 = =

Then with “**nu**” in “ukixukydromeiwszvwioku**nu**khxhroajroanqyebtlkjegmy”.

mod 26 = =

Then with “**kh**” in “ukixukydromeiwszvwiokunu**kh**xhroajroanqyebtlkjegmy”.

mod 26 = =

Then with “**xh**” in “ukixukydromeiwszvwiokunukh**xh**roajroanqyebtlkjegmy”.

mod 26 = =

Then with “**ro**” in “ukixukydromeiwszvwiokunukhxh**ro**ajroanqyebtlkjegmy”.

mod 26 = =

Then with “**aj**” in “ukixukydromeiwszvwiokunukhxhro**aj**roanqyebtlkjegmy”.

mod 26 = =

Then with “**ro**” in “ukixukydromeiwszvwiokunukhxhroaj**ro**anqyebtlkjegmy”.

= from previous logic.

Then with “**an**” in “ukixukydromeiwszvwiokunukhxhroajro**an**qyebtlkjegmy”.

mod 26 = =

Then with “**qy**” in “ukixukydromeiwszvwiokunukhxhroajroan**qy**ebtlkjegmy”.

mod 26 = =

Then with “**eb**” in “ukixukydromeiwszvwiokunukhxhroajroanqy**eb**tlkjegmy”.

mod 26 = =

Then with “**tl**” in “ukixukydromeiwszvwiokunukhxhroajroanqyeb**tl**kjegmy”.

mod 26 = =

Then with “**kj**” in “ukixukydromeiwszvwiokunukhxhroajroanqyebtl**kj**egmy”.

mod 26 = =

Then with “**eg**” in “ukixukydromeiwszvwiokunukhxhroajroanqyebtlkj**eg**my”.

mod 26 = =

Then with “**my**” in “ukixukydromeiwszvwiokunukhxhroajroanqyebtlkjeg**my**”.

mod 26 = =

Putting that all together the encrypted message “**ukixukydromeiwszvwiokunukhxhroajroanqyebtlkjegmy”** decrypts to **“meet me at the usual place at ten rather than eight oclocka”** in plaintext.

From this point you can chose to remove the padding of the appended “a” character at the end or not.

B. Determine the values of *φ*(27), *φ*(49) and *φ*(440), where *φ*(*n*) is the Euler’s Totient Function.

(15points)

*φ*(27)

Step 1: *φ*(27) = (*φ*(3) \* *φ*(9)) since *φ*(3) is prime *φ*(3)=2.

*Step 2: φ*(9) = (*φ*(3) \* *φ*(3)), from same logic above both instances of *φ*(3) = 2.

***Final answer is (2\*2\*2) = 8***

*φ*(49)

*Step 1: φ*(49) = (*φ*(7))\* *φ*(7)), since 7 is prime both *φ*(7) = 6

***Final answer is (6\*6) = 36***

*φ*(440)

*Step 1: φ*(440) = (*φ*(5))\* *φ*(88)), since 5 is prime *φ*(5) = 4

*Step 2: φ*(88) = (*φ*(11))\* *φ*(8)), since 11 is prime *φ*(11) = 10

*Step 3: φ*(8) = (*φ*(2))\* (*φ*(4))).

***Final answer is (4\*10\*4) = 160***

C Find 3201 mod 11; and 2341 mod 341

(20 points)

3201 mod 11:

By using Fermat’s theorem, we know that (ap-1=1mod p) where p is a prime number and a is a positive number and is not divisible by p.

Therefore

310=1 mod 11.

So that , 3201= (((310)20)\* 3) = **3 mod 11.**

2341 mod 341:

By Using Fermat’s theorem, we know that (ap-1=1mod p).

Therefore,

2340=1mod341

So, 2340 \* 2=2\*1mod341

So that, 2341 = **2 mod 341.**

D. Determine the multiplicative inverse of *x*3 + *x* + 1 in GF(24) with *m*(*x*) = *x*4 + *x* + 1.

(10 points)

Start with x4 + x + 1 = x (x3+x+1) + (x2+1)

=> x3+x+1 = x(x2+1)+1

=> x2+1 = x4+x+1+(x(x3+x+1))

=>1 = x3+x+1= x(x4+x+1+(x(x3+x+1))

=>1= (x2+1)(x3+x+1)+(x(x4+x+1))

=> (x3+x+1)= (x2+1)

**The multiplicative inverse is** **(x2+1).**

**This can be proven because**

(x3+x+1)\*(x2+1)= x5+x2+x+1.

So that x5+x2+x+1 mod x4+x+1= 1.

Since it is equal to 1, then x2+1 is the multiplicative inverse.

E. Develop a table similar to Table 5.5 on page 148 of the 7th edition textbook (Table 4.9 on page 121 of the 6th edition textbook) for GF(28), with *m*(*x*) = *x*8 + *x*4 + *x*3 + *x*2 + 1 (from 0 to g14)

(15 points)

|  |  |  |  |
| --- | --- | --- | --- |
| Power Representation | Polynomial | Binary | Hex |
| 0 | 0 | 00000000 | 0 |
| G0 | G0 | 00000001 | 1 |
| G1 | G1 | 00000010 | 2 |
| G2 | G2 | 00000100 | 4 |
| G3 | G3 | 00001000 | 8 |
| G4 | G4 | 00010000 | 10 |
| G5 | G5 | 00100000 | 20 |
| G6 | G6 | 01000000 | 40 |
| G7 | G7 | 10000000 | 80 |
| G8 | g4+g3+g2+1 | 00011101 | 1D |
| G9 | g5+g4+g3+g | 00111010 | 3A |
| G10 | g6+g5+g4+g2 | 01110100 | 74 |
| G11 | g7+g6+g5+g3 | 11101000 | E8 |
| G12 | g7+g6+g3+g2+1 | 11001101 | CD |
| G13 | g7+g2+g+1 | 10000111 | 87 |
| G14 | g4+g+1 | 00010011 | 13 |

G8 = g4+g3+g2+1

G9 = g5+g4+g3+g

G10 = g6+g5+g4+g2

G11= g7+g6+g5+g3

G12= g7+g6+g3+g2+1

G13= g7+g2+g+1

G14= g4+g+1

F The Miller-Rabin test can determine if a number is not prime but cannot determine if a number is prime. How can such an algorithm be used to test for primality? (10 points)

The Miller-Rabin test is similar to the Fermat primality test because it is based on the principle that if X2=Y2modN, BUT X!=+/- Y mod N, then N is composite.

Step 1: The algorithm takes N-1 and sets it equal to 2R\*D where D is an odd number and N is the number to be tested.

Step 2: Choose a number A in the range 2 through N-2.

Step 3: IF X0 = AD modN = +/- 1, then N can be prime.

Step 4: IF Xi = Xi -1modN = 1, then N is composite. IF it is -1, N is Prime.

Step 5: Repeat for all possibilities if neither -1 nor +1 appears for Xi then N is composite.

This means the Miller-Rabin test determines if a number is prime by exhausting efforts to see if it is NOT composite and if it has any characteristics of a prime number. If all possibilities are NOT composite, do not display any characteristics that make them composite, and simultaneously adhere to characteristics of being a prime number, then the number is PROBABLY prime.

G. Given *x* ≡ 2 (mod 3), *x* ≡ 2 (mod 7), and *x* ≡ 3 (mod 5), please solve the *x* by using Chinese Remainder Theorem. (15 points)

Let

X1 = 2 + 3a

X2= 2 +7b

X3= 3 +5c

Substituting X in X1 of X2,

2+3a = 2 mod 7 => 3a = 0mod 7,

Thus A=7B for some integer B.

Now let’s substitute A into X1.

X = 2 +3(7B) => 2 + 21B.

Now substituting X for X1 into X3.

2+21B= 3 mod 5 => B=1+5C.

Now returning to our earlier equation of X=2 +21B now equals X=2+ 21(1+5C)

=> **x= 23 + 105C**

**So that X = (23,128,233,343, and so on….)**

H. Given *p* = 17; *q* = 31; *e* = 7; *C* = 128, please calculate the *d* value for private key and recover the original plain text message *M*. (Need to show the details of the calculation in details) (15 points)

Let n=p\*q,

So that n= 17\*31=527.

*f*(n)= (p-1)\*(q-1)= 16\*30 =480.

Now compute d = e-1 mod f(n) by using backward substitution of GCD algorithm:

480 = 7 \* 68 + 4

7 = 4 \* 1 + 3

4 = 3 \* 1 + 1

3 = 1 \* 3 + 0

So now,

1 = 4 – 3

= 4 – (7 – 4)

= 4 – (7 – (480 – 7\*68))

= 4 – (7 – 480 + 7\*68)

= 480 – 7\*68 – 7 + 480 – 7\*68

= 480\*2 – 7\*137

Now d=e-1mod f(n) => d = -137mod480 = 343.

So that **d=343.**

**Public Key = 7,527**

**Private Key =343,527**

**Plaintext =2**

Plaintext P =(C)^d mod n

So that 128^(343) mod 527=> **2**

I. User A and B use the Diffie-Hellman key exchange technique with a common prime *q* = 71 and a primitive root *α* = 7. (15 points)

1. If user A has a private key *XA* = 5, what is *A*’s public key *YA*?
2. If user B has a private key *XB* = 12, what is *B*’s public key *YB*?
3. What is the shared security key?

Let YA= aYA mod q

So that 7^5mod71 = 51

**YA=51**

Let YB= aYB mod q

So that 7^12mod71 = 4

**YB=4**

Let Share key= YBXA mod q

So that 4^5mod71 = 30

**YB=30**

J. Using the extended Euclidean algorithm, find the multiplicative inverses of

1. 13 mod 2436 (10 points)

Step 1: Euclidean algorithm=>2436 = 187\*13+5

=>13=2\*5+3

=>5=1\*3+2

=>3=1\*2+1

Step 2: write the GCD as a multiple of (13,2436) =1

=> 1=3-1.2

=> 1= 3-1.(5-1.3) = 2.3-1.5

=> 1=2.(13-2.5)-1.51= 2.13-4.5-1.51=2.13-5.5

=> 1=2.13-5.(2436-187.13)=1=2.13-5.2436+935.13=1=937.13-5.2436

=>13(937)-2436(5) => 1

**Since GCD(13,2436) = 1,**

**the multiplicative inverse of 13 mod 2436 = 937**

1. 144 mod 233 (10 points)

Step 1: Euclidean algorithm=>233 = 1\*144+89

=>13=2\*5+3

=>5=1\*3+2

=>3=1\*2+1

Step 2: write the GCD as a multiple of (144,233) =1

=>1=3-1.2

=>1= 3-1\*(5-1.3) = 2.3-1.5

=>1=2\*8-3\*5

=>1=2\*8-3\*(13-1\*8)

=>1=5\*8-3\*13

=>1=5\*(21-1\*13)-3\*13

=>1=5\*21-8\*13

=>1=13\*21-8\*34

=>1=13\*(55-1\*34)-8\*34

=>1=13\*55-21\*34

=>1=13\*55-21\*(89-1\*55)

=>1=34\*55-21\*89

=>1=34\*(144-1\*89)-21\*89

=>1=34\*144-55\*89

=>1=34\*144-55\*(233-1\*144)

=>89\*144-55\*233=1

**Since GCD(144,233) = 1,**

**the multiplicative inverse of 144mod233 = 89**

K. Draw a matrix similar to Table 1.4 on page 15 of the 7th edition textbook (Table 1.4 on page 21 of the 6th edition textbook) that shows the relationship between security mechanisms and attacks. (20 points)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Attacks | | | | | |
| Mechanisms | Release of Message | Traffic Analysis | Masquerade | Replay | Modification of Message | Denial of Service |
| Encipherment | 🗹 |  |  |  |  |  |
| Digital Signature |  |  | 🗹 | 🗹 | 🗹 |  |
| Access Control | 🗹 | 🗹 | 🗹 | 🗹 |  | 🗹 |
| Data Integrity |  |  |  | 🗹 | 🗹 |  |
| Authentication Exchange | 🗹 |  | 🗹 | 🗹 |  | 🗹 |
| Traffic Padding |  | 🗹 |  |  |  |  |
| Routing Control | 🗹 | 🗹 |  |  |  | 🗹 |
| Notarization |  |  | 🗹 | 🗹 | 🗹 |  |