Information Security Technologies COMP607
Assignment 1 (20%)

Instructions:

1. Type (preferable) or write your answers neatly on A4 paper. The assignment must be done in English. It must be your own work. Show all workings. Do not copy material from anywhere without appropriate referencing of the source as it will be penalised.
2. As far as possible, take screenshots of your work and paste into your submission to show evidence of your work.
3. Begin each question on a fresh page, if possible.
4. Submit your answers (in pdf format), files, etc. in *https://canvas.aut.ac.nz/Assignments.*
5. Marks will be deducted for untidy work.
6. Date due: refer to Canvas

Notes: Some of the questions require you do the tasks using Linux commands. If you don't have Linux, you can use the server at *scopius.aut.ac.nz*. To do this, open a Windows Powershell and at the $ prompt, type `ssh username@scopius.aut.ac.nz.` Your username is your AUT login name and password is your day of birth, e.g. *01apr*

To copy the file from your directory in the Linux server to your local PC, you can use the WinSCP application.

Where required, to access files in *https://scopius.aut.ac.nz,* username/password: *student/student*

You may also need to use a scientific calculator capable of doing modulo math. You can use the genius math tool in the scopius server by typing at the $ prompt, `genius`

Questions/Tasks:

1. (a) The following cipher text is obtained using a rail-fence method. Using brute force, determine the key and the plaintext message in English? (2 marks)

AAEHDSGNMBTTAOHTODESTRNOAIOIEGB

In applying the Brute Force, trying to examine different choices of rails, starts from 2 and increase by 1 until receiving a meaningful plaintext result.

Attempt 1: 2 rails

A		A		E		H		D		S		G		N		M		B		T		T		A		O		H		T
	O		D		E		S		T		R		N		O		A		I		O		I		E		G		B	

The plaintext in this attempt: AOADEEHSDTSRGNNOMABITOTIAEOGHBT. This is a completely meaningless message, so 2 rails are not the correct choice.

Attempt 2: 3 rails

A				A				E				H				D				S				G				N		
	M		B		T		T		A		O		H		T		O		D		E		S		T		R		N	
		O				A				I				O				I				E				G				B

The plaintext in this attempt: AMOBATATEAIOHHOTDOIDSEESGTGRNNB. This is also a completely meaningless message, so 3 rails are not the correct choice.

Attempt 3: 4 rails

A						A						E						H						D						S
	G				N		M				B		T				T		A				O		H				T	
		O		D				E		S				T		R				B		O				A		I		
			O						I						E						G						B			

The plaintext in this attempt: AGOODNAMEISBETTERTHANGOODHABITS. This is meaningful in English, so this attempt is successful, and 4 rails is the correct choice.

Plaintext: A GOOD NAME IS BETTER THAN GOOD HABITS

(b) Encrypt the following plaintext using the Vignere cipher method, using modulo 26 addition (preferable) or the Vignere table. (3 marks)

KNOWLEDGE IS NEVER WASTED

Key: *EQUATORIAL*

Plaintext: KNOWLEDGE IS NEVER WASTED

` Ciphertext: ODIWESUOE TW DYVXF NISEIT

(c). Using a text editor, create a textfile called *secretLetter.txt* containing the text

IF YOU ARE STANDING STRAIGHT, DON'T WORRY IF YOUR SHADOW IS CROOKED.

```
Last login: Thu Feb  1 21:28:57 2024 from 1.53.27.4
kdr8943@Scopius:~$ touch secretLetter.txt
kdr8943@Scopius:~$
```

```
kdr8943@Scopius:~$ touch secretLetter.txt
kdr8943@Scopius:~$ cat > secretLetter.txt
IF YOU ARE STANDING STRAIGHT, DON'T WORRY IF YOUR SHADOW IS CROOKED.
kdr8943@Scopius:~$ type secretLetter.txt
-bash: type: secretLetter.txt: not found
kdr8943@Scopius:~$ cat secretLetter.txt
IF YOU ARE STANDING STRAIGHT, DON'T WORRY IF YOUR SHADOW IS CROOKED.
kdr8943@Scopius:~$
```

Encrypt this file using the openssl toolset with your family name as the encryption key, using each of the following methods:

(i) AES 192 bit key, ECB mode, name the encrypted file s*ecretLetter_aes_ecb.enc*
My encryption key: phu1404

Running the openssl with option about algorithm used (AES 192, mode ECB), along with information about input file and output decrypted file. Provide the encryption key with -k flag.

Test the encrypted file by attempting to decode it with the encryption key and save the decrypted text in the test file secretLetter1.txt. Examining the contents of the test file confirmed that the original file had been successfully encrypted.

```
kdr8943@Scopius:~$ openssl enc -aes-192-ecb -in secretLetter.txt -out secretLetter_aes_ecb.enc -k phu1
404
kdr8943@Scopius:~$ openssl enc -d -aes-192-ecb -in secretLetter_aes_ecb.enc -out secretLetter1.txt -k
phu1404
kdr8943@Scopius:~$ cat secretLetter1.txt
IF YOU ARE STANDING STRAIGHT, DON'T WORRY IF YOUR SHADOW IS CROOKED.
kdr8943@Scopius:~$
```

(ii) DES OFB mode, name the encrypted file *secretLetter_des_ofb.enc* (5 marks)

Running the openssl with option about algorithm used (DES, mode OFB), along with information about input file and output decrypted file. Provide the encryption key with -k flag.

Test the encrypted file by trying to decrypt the file with encryption key and store the decrypted content inside test file secretLetter2.txt. Examining the content of the test file verified that the original file has been successfully encrypted.

```
kdr8943@Scopius:~$ openssl enc -des-ofb -in secretLetter.txt -out secretLetter_des_ofb.enc -k phu1404
kdr8943@Scopius:~$ openssl enc -d -des-ofb -in secretLettet_des_ofb.enc -out secretLetter2.txt -k phu1
404
secretLettet_des_ofb.enc: No such file or directory
3082036916:error:02001002:system library:fopen:No such file or directory:bss_file.c:398:fopen('secretL
ettet_des_ofb.enc','r')
3082036916:error:20074002:BIO routines:FILE_CTRL:system lib:bss_file.c:400:
kdr8943@Scopius:~$ openssl enc -d -des-ofb -in secretLetter_des_ofb.enc -out secretLetter2.txt -k phu1
404
kdr8943@Scopius:~$ cat secretLetter2.txt
IF YOU ARE STANDING STRAIGHT, DON'T WORRY IF YOUR SHADOW IS CROOKED.
kdr8943@Scopius:~$
```

2. Consider a cryptosystem where the user enters a key in the form of a password.

a. Assume a password consists of 10 latin characters, where each one is encoded using the ASCII scheme (7 bits per character). What is the size of the key space? (2 marks)

The size of the key space is the number of possible key combinations given the parameters. In this case, the size of the key space indicates the number of possible cryptosystem passwords.

Based on the description, a password consists of 10 letters, where each letter is encoded by the ASCII scheme (7 bits per character). Since for each bit, there can only be either 0 or 1 so the number of different sets of 7 bits can be generated is 2^7. As each letter is encoded by 7 bits, number of choices for each letter in the password is 2^7.

The password consists of 10 letters, and there are 2^7 choices for each letter. Therefore, the number of possible options for the password, or the size of the key space is:

$$2^7 \mathbf{x} 2^7 \mathbf{x} 2^7 \mathbf{x} 2^7 \mathbf{x} 2^7 \mathbf{x} 2^7 \mathbf{x} 2^7 \mathbf{x} 2^7 \mathbf{x} 2^7 \mathbf{x} 2^7 = (2^7)^{10} = 2^{70}$$

b. What is the corresponding key length in bits? (2 marks)

Because each character in the password is encoded by the ASCII scheme, using 7 bits. As the password is 10-letter long, the number of bits in the key (key length in bits) is then:

10 X 7 = 70 bits

c. Assume that most users use only 26 lowercase letters from the alphabet instead of the full 7 bits of the ASCII encoding. What is the corresponding key length in bits in this case? (2 marks)

Each letter in the password in the case of this question now uses only 26 characters instead of 2^7 characters. Because $16 = 2^4 < 26 < 2^5 = 32$, we need 5 bits to encode each of the letter in the password (having 26 choices for each letter).

Therefore, to be able to build a key for a password with 10 letters, where each letter is one of the 26 lowercase lettes, the key length in bits is:

5 X 10 = 50 bits

d. At least how many characters are required for a password in order to generate a key length of 128 bits in case of letters consisting of (4 marks)

The key has a length of 128 bits, capable of having a totally of 2^{128} different options for the key space.

(i). 7- bit characters?

Each character in the password is encoded using 7 bits, so having 2^7 choices. Suppose the password is n-letter long, the key space in terms of n is:

$2^7 \text{ x } 2^7 \text{ x } \ldots \text{ x} 2^7$ (n times) $= 2^{7n}$

As the totally different options available for the key space is 2^{128}, we have:

$2^{7n} = 2^{128}$
$7n = 128$

n = 128/7 (approximate) = 18.286 characters < 19 characters

Therefore, at least 19 7-bit characters are required for a password in order to generate a key length of 128 bits.

(ii). 26 lowercase letters from the alphabet?

Each character in the password has 26 different choices. As $16 = 2^4 < 26 < 2^5$, we need 5 bits to encode each letter in the password. Suppose the password is n-letter long, the key space in terms of n is: 2^{5n}.

(Note: While longer keys may offer more alternatives, a minimum of 5 bits is necessary to encode each letter independently.)

As the total different options available for the key space is 2^{5n}, we have:

$2^{5n} = 2^{128}$
$5n = 128$

n = 128/5 = 25.6 characters < 26 characters

Therefore, at least 26 lowercase letters are required for a password in order to generate a key length of 128 bits

3. The following ASCII bits (8 bits per character) is obtained using a stream cipher to encrypt an English plaintext message. (The spaces are only inserted for readability). The encryption key is a single alphabet character (8 bits in ASCII). Using frequency analysis, or otherwise, obtain the plaintext. State and explain the weakness in the way this cipher is used.

(10 marks)

```
00010111 00011111 00011111 00001110 00011011 00001110 00011111
00010110 00011111 00001100 00011111 00010100 00011011 00010111
```

The encryption key is 8-bits long, thus every 8-bits in the cipher are the result of an XOR operation between every 8-bits in the plaintext (which corresponds to a letter in the plaintext) and the encryption key.

Because each operation is conducted on an 8-bits ASCII representation of a single character in the plaintext, the most common block of 8-bits in the cipher corresponds to the most frequent letter in the plaintext.

From the cipher, the most common block of 8 bits is the 00011111 (appeared 5 times / 14 blocks).

This 8-bits block is the result of a XOR operation between the plaintext's most common letter (8-bit representation) and the encryption key. Using frequency analysis, imagine that the letter e is the most commonly used letter in English.

The 8-bit ASCII representation of letter e is: 01100101

So if we assume that this is the most frequent letter in the plaintext, the encryption will then be:

Block of 8-bits in cipher	0	0	0	1	1	1	1	1
8-bit ASCII representation of letter e	0	1	1	0	0	1	0	1
Tested encryption key	0	1	1	1	1	0	1	0

Assume the encryption key is 01111010, try to find the plaintext by reversing the XOR operation and examining the meaningfulness of the plaintext received.

Cipher	00010111	00011111	00001110	00011011	00010110	00001100	00010100
Encryption key	01111010	01111010	01111010	01111010	01111010	01111010	01111010
Plaintext	01101101	01100101	01110100	01100001	01101100	01110110	01101110
Letter	m	e	t	a	l	v	n

In this attempt, the plaintext is: meetatelevenam. Surprisingly, this plaintext is meaningful.

Plaintext: MEET AT ELEVEN AM

The weekness in the way this cipher is used:

- The same technique is done on each letter in the plaintext, thus the frequency of letter distribution will continue during encryption and appear in the encrypted text. This means that by applying frequency analysis, the number of brute-force assaults may be greatly reduced, taking into account the frequency distribution of popular letters in English.

Solution: Using One time pad (OTP) to prevent keystream reuse (the keystream is completely random and is used only once), preventing the frequency analysis can be applied to recover parts of the messages.

- The length of the encryption key is identical to the representation of each letter in plaintext. In this scenario, the frequency distribution is represented by letters, making brute-force attacks considerably easier because the number of attempts is greatly reduced.

Solution: Using an encryption key that is either as long as the plaintext or as short as each character representation (8-bits). Although frequency distribution persists in the second option, it is not the frequency of the distribution of letters that persist in the cipher text, but of blocks of n-bit (with n being the length of the key), which does not rapidly reduce the number of choices in brute-force attacks based on the real language.

4. Use md5 to check for file integrity.
   Using your browser, go to https://scopius.aut.ac.nz. Click on `Information Securities Technologies, Software` directory. Use *`student`* for both username and password.

   There are two *putty-0.70* files, one of which is corrupted. The md5 sum of the good file is also available on the website.

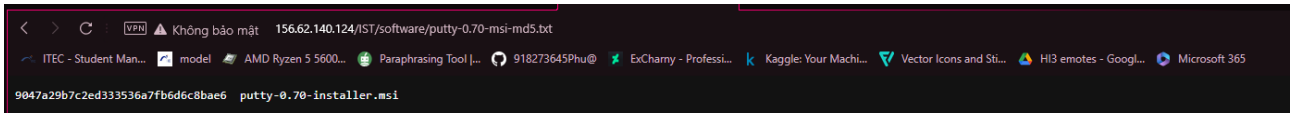
   Download both versions of the putty-0.70 files and identify the good copy.

(a) Run the md5sum on the downloaded file and save the output to a file called `putty.md5` show the screenshot of the content of this md5 file.
`$ md5sum puttyputty-0.70-installer.msi`

(6 marks)

```
kdr8943@Scopius:~$ md5sum putty-0.70-installer.msi > putty.md5
kdr8943@Scopius:~$ md5sum putty-0.70-installed.msi >> putty.md5
kdr8943@Scopius:~$ cat putty.md5
06260b3982589c5714aeded7a40853ba  putty-0.70-installer.msi
9047a29b7c2ed333536a7fb6d6c8bae6  putty-0.70-installed.msi
kdr8943@Scopius:~$
```

(b) Research how to use md5 to check the integrity of files. Check which file is the good copy. Show screen shots of your work. (4 marks)



5. There are two files in Scopius server in *home/pub* folder (and also in *https://scopius.aut.ac.nz*) called n*otice1.txt* and *notice2.txt* and their related HMACs. One of the notices is fake. You know that the authentic copy was made by the person who has the secret key ***comp607*** shared with you. You are required to determine which copy is authentic.

Change to your working directory and copy theses files into it.

```
$ cp /home/pub/notice*.* .
```

```
kdr8943@Scopius:~$ cp /home/pub/notice*.* .
kdr8943@Scopius:~$ ls
data              notice2.hmac.txt  putty.md5          secretLetter_aes_ecb.enc  secretLetter.txt  test1.txt
notice1.hmac.txt  notice2.txt       secretLetter1.txt  secretLetter_aes.enc      test              test2.txt
notice1.txt       notice2.txt.sig   secretLetter2.txt  secretLetter_des_ofb.enc  test1md5.txt
kdr8943@Scopius:~$
```

```
kdr8943@scopius.aut.ac.nz's password:
notice1.hmac.txt                                              100%   91     0.1KB/s   00:00
notice1.txt                                                   100%  178     0.2KB/s   00:00
notice2.hmac.txt                                              100%   91     0.1KB/s   00:00
notice2.txt                                                   100%  200     0.2KB/s   00:00
notice2.txt.sig                                               100%  256     0.3KB/s   00:00
kdr8943@Scopius:~$
```

```
kdr8943@Scopius:~$ ls | grep 'notice*.*'
notice1.hmac.txt
notice1.txt
notice2.hmac.txt
notice2.txt
notice2.txt.sig
kdr8943@Scopius:~$
```

(a) Display both files. Paste the screenshots.

```
kdr8943@Scopius:~$ cat notice1.txt
Exam Notice

The exam is very easy and you do not need to study for it.

No need to work hard to understand the material.

Enjoy and have a good time.

Warmest regards.
Dr. Yang
kdr8943@Scopius:~$ cat notice2.txt
Exam Notice

The exam is not easy and you do need to study for it.

Do work hard to understand the material.

After the exam then you deserve to enjoy and have a good time.

Warmest regards.
Dr. Yang
kdr8943@Scopius:~$
```

(b) Use openssl tools to determine their HMACs and determine which is the authentic one. First, generate the HMAC for each one using the shared key, ***comp607***

```
$ openssl dgst -hmac sharedkey notice1.txt
```

```
kdr8943@Scopius:~$ openssl dgst -hmac comp607 notice1.txt
HMAC-SHA256(notice1.txt)= 91f22871f20bb8561f04d0b25bf188d5ceac5d9a2fb5e68ae6fae825df93b2a0
kdr8943@Scopius:~$ cat notice1.hmac.txt
HMAC-SHA256(notice1.txt)= 1616c9336bdc562083f52f627326ca0180e3f796d24595ec6269340e017a1961
kdr8943@Scopius:~$
```

```
kdr8943@Scopius:~$ openssl dgst -hmac comp607 notice2.txt
HMAC-SHA256(notice2.txt)= a9c9ae0e63165ccdc13132f933d605dec8863127a6c7857bf7fa98103af734fc
kdr8943@Scopius:~$ cat notice2.hmac.txt
HMAC-SHA256(notice2.txt)= a9c9ae0e63165ccdc13132f933d605dec8863127a6c7857bf7fa98103af734fc
kdr8943@Scopius:~$
```

Compare with the given HMACs, *notice1.hmac.txt* and *notice2.hmac.txt*

```
$ cat notice1.hmac.txt
```

Which notice is the authentic version? (10 marks)

- Compare the created HMACs with the given HMACs, the notice2.txt file is the authentic.

6. In your Linux working directory, create a text file using one of the following:
Using a text editor such as `pico` enter or copy and paste some text into it, and and save.

```
$ pico test1.txt
```

Alternatively you can do as follows:

```
$ echo "This is some text file" > test1.txt
```

View the file:

```
$ cat test1.txt
```

```
kdr8943@Scopius:~$ echo "This is some text file" > test1.txt
kdr8943@Scopius:~$ cat text1.txt
cat: text1.txt: No such file or directory
kdr8943@Scopius:~$ cat test1.txt
"This is some text file"
kdr8943@Scopius:~$
```

(a) Obtain the md5 hash of your file: `(To get help: $ md5sum -h)`

```
$ md5sum test1.txt
```

(2 marks)

```
kdr8943@Scopius:~$ md5sum test1.txt
eefda21409d77429098cc7368915ad1f  test1.txt
kdr8943@Scopius:~$
```

(b) Make a small change in your file such as adding a space, dot, etc.
Save the file a with a new name, e.g. "`test2.txt`" and obtain the new md5 hash.
Compare them. Are they same, different, or very different? Do this visually:

```
$ cat test1.txt
$ cat test2.txt
```

(4 marks)

```
kdr8943@Scopius:~$ echo "This is some text file.." > test2.txt
kdr8943@Scopius:~$ cat text1.txt
cat: text1.txt: No such file or directory
kdr8943@Scopius:~$ cat test1.txt
"This is some text file"
kdr8943@Scopius:~$ cat test2.txt
"This is some text file.."
kdr8943@Scopius:~$
```

```
eefda21409d77429098cc7368915ad1f  test1.txt
kdr8943@Scopius:~$ md5sum test2.txt
ba6c05db27aa545314ac1b486ad03dbb  test2.txt
kdr8943@Scopius:~$
```

It is clear that although the majority of the content are the same, the md5 sums of the 2 files are completely different.

(c) Make an MD5 integrity check hash for the file `test1.txt`

```
$ md5sum test1.txt > test1md5.txt
```

Verify the integrity of `test1.txt`:

```
$ md5sum -c test1md5.txt
```

(4 marks)

```
kdr8943@Scopius:~$ md5sum test1.txt > test1md5.txt
kdr8943@Scopius:~$ md5sum -c test1md5.txt
test1.txt: OK
kdr8943@Scopius:~$
```

7. RSA algorithm. *Alice* wishes to send a message $M = 513$ to *Bob* by encrypting it with *Bob's* public key. Use the following parameters to obtain the RSA keys for *Bob*. Assume that *Alice* is able to obtain *Bob's* public key securely.

a). Use $p = 23$, $q = 29$, obtain suitable values for *Bob's* private key d and public key e exponents. What is *Bob's* public key? What is ciphertext that *Alice* obtains by encrypting M using *Bob's* public key? Show how *Bob* can decrypt this ciphertext correctly using his private key. (5 marks)

Solving for private key d and public key e of Bob:

$n = p \times q = 23 \times 29 = 667$

$\phi(n) = (p - 1) \times (q - 1) = 22 \times 28 = 616$

Since the private key d and the public key e of Bob is chosen to satisfy the condition $d \times e \equiv 1 \ (mod\ 616)$, the basic condition for choosing d and e is that $d \times e = 616k + 1 (k\ is\ an\ integer)$

Testing the factorization results for some values of k to choose the suitable couple of d and e

k	d x e	Factorization result
1	617	617×1
2	1233	$3^2 \times 137$

When $k = 2$, one of the possible options to choose for d and e is $d = 137$ and $e = 9$. Choose the numbers as the private key and public key of Bob.

Encrypting the message $M = 513$ using Bob's public key $e = 9$

$C \equiv Me\ mod\ n \equiv 5139\ mod\ 667 = 429$

(The above result is calculated using Python)

```
>>> 513 ** 9 % 667
429
```

Therefore, the ciphertext that Alice obtains by encrypting M using Bob's public key is $C = 429$

Bob's process of decrypt the ciphertext

$C^d \equiv 429^{137}\ mod\ 667 = 513 = M$

```
>>> 429 ** 137 % 667
513
```

The correct message $M = 513$ has successfully been decrypted.

b). Use $p = 11$, $q = 23$, $M = 109$. Calculate *Bob's* private and public keys exponents d and e. The public key exponent e should be a small number.

Bob 'signs' the message *M* by encrypting it with his private key. What is the 'signature' *s*, obtained by *Bob*?
When *Alice* obtains the message *M* and the signature *s*, show how *Alice* would verify the signature. (5 marks)

Solving for private key d and public key e of Bob

$n = p \times q = 11 \times 23 = 253$

$\phi(n) = (p - 1) \times (q - 1) = 10 \times 22 = 220$

Since the private key d and the public key e of Bob is chosen to satisfy the condition $d \times e \equiv 1\ (mod\ 220)$, the basic condition for choosing d and e is that $d \times e = 220k + 1(k\ is\ an\ integer)$

Testing the factorization results for some values of k to choose the suitable couple of d and e

k	d x e	Factorization result
1	221	13×17
2	441	$3^2 \times 7^2$

When $k = 2$, one of the possible options to choose for d and e is $d = 49$ and $e = 9$. Choose the numbers as the private key and public key of Bob.

- Bob signs the message M by encrypting it with his private key $d = 49$. The signature s is obtained by the following calculations:

$$s \equiv M^d\ mod\ n \equiv 109^{49}\ mod\ 253 = 21$$

```
>>> 109 ** 49 % 253
21
```

Therefore, the signature is $s = 21$

- Alice would verify the signature by the following calculations:
  - Calculating M' and compare with M. The calculation of M' is as follows:
    $M' \equiv s^e\ mod\ n \equiv 219\ mod\ 253 = 109 = M$

```
>>> 21 ** 9 % 253
109
```

  - As $M' = M$, Alice would verify the signature and guarantee that the communication is from Bob and not anybody else.

8. Describe how *Alice* and *Bob* are able to exchange a secret key using the DH algorithm.

- Alice and Bob can exchange a secret key using DH algorithm by the following steps:
  + Step 1: They agree to use public parameters G and prime modulus p
  + Step 2: Both parties generate his/her private key and then compute his/her public key by the calculations:
    - Alice: Generate private key $a < p$, then compute public key $A = G^a \pmod p$
    - Bob: Generate private key b, then compute public key $B = G^b \pmod p$
  + Step 3: Both parties exchange their public keys (can be through an insecure channel)
  + Step 4: Both parties obtained the shared secret key Ks by the calculations:
    - Alice: $K_S = K_{As} = B^a = G^{ba} \pmod p$
    - Bob: $K_S = K_{Bs} = A^b = G^{ba} \pmod p$

(a) Demonstrate the process by using generator $g=2$, and prime modulus $n=$ 4787

- Both parties agree to use the public parameters mentioned in the question: $g = 2$ and $n = 4787$
- Alice and Bob generate his/her private key and then compute the public keys
  + Alice: Suppose Alice chooses her private key $a = 4500 < n = 4787$, Alice's public key is $A = 2^{4500} \pmod{4787} = 2666$

  + Bob: Suppose Bob chooses his private key $b = 2750 < n = 4787$, Bob's public key is $B = 2^{2750} \ (mod\ 4787) = 2679$

- Alice and Bob exchange their public keys, Bob obtains $A = 2666$ from Alice and Alice obtains $B = 2679$ from Bob
- Alice and Bob calculates the shared secret key K_S

(b) Which party, Alice, Bob, both, or none, can determine the value of the shared key?

- The value of the shared key is obtained through a set of calculations and is equal to the remainder of G^{ba} when divided by p.
- The public prime modulus (p) and generator (G) must be agreed upon by both parties before to the exchange. In addition, the value of Gba depends on the value of a and b, which are Alice and Bob's private keys and only known by them. Because of the reasons stated above, the value of the shared key can only be established by both parties, Alice and Bob, because the private keys of each are necessary for the computations to derive the shared key.

(10 marks)

9. The DH algorithm can also be used for encryption as well using the ElGamal scheme. Demonstrate this encryption scheme using a numerical example as follows.

Alice wishes to encrypt a secret message, *M = 215* to *Bob*. They have chosen the parammeters and private keys as follows:
Bob: private key *b = 231*, generator *G=2*, prime modulus *p = 443*.
Alice: private key *a = 198*

Show how the method works by displaying what each side computes and sends to each other, including the cipher texts and decrypted messages.

(i) using the above numbers for *M, a, b*

M = 215, a = 198, b = 231

- Step 1: Bob defines $G = 2$, $p = 443$, secret key $b = 231$. Then Bob computes his public key
$B = G^b \pmod p = 2^{231} \pmod{443} = 305$
Bob then sends his public key $k_{Bpub} = (G, p, B) = (2, 443, 305)$

- Step 2: Alice selects her private key $a = 198$. Then Alice computes her public key
$A = G^a \pmod p = 2^{198} \pmod{443} = 144$
Then Alice computes the encryption key $K_S = B^a \equiv G^{ba} \pmod p = 305^{198} \pmod{443} = 321$

- Step 3: Alice encrypts the message $M = 215$, obtaining the cipher
$C = M \times K_S \pmod p = 215 \times 321 \pmod{443} = 350$
Alice then sends her public key $A = 144$ and the cipher text $C = 350$ to Bob

- Step 4: Bob derives the encryption key by the calculation:
$K_S = A^b \pmod p = 144^{231} \pmod{443} = 321$

Bob then decrypts to cipher to receive the message:
$C \times K_S^{-1} \pmod p = 350 \times 374 \pmod{443} = 215 = M$

```
>>> pow(321, -1, 443)
374
>>> 350 * 374 % 443
215
```

The message has successfully been decrypted. The message is $M = 215$, the cipher is $C = 350$

(ii) using your own choice of numbers for *M, a, b* (10 marks)

My choice of numbers: M = 453, a = 444, b = 203

- Step 1: Bob defines $G = 4$, $p = 144$, secret key $b = 203$. Then Bob computes his public key
$B = G^b \pmod p = 4^{203} \pmod{144} = 16$
Bob then sends his public key $k_{Bpub} = (G, p, B) = (4, 144, 16)$

- Step 2: Alice selects her private key $a = 444$. Then Alice computes her public key
$A = G^a \pmod p = 4^{444} \pmod{144} = 64$

Then Alice computes the encryption key $K_S = B^a \equiv G^{ba} \pmod{p} = 16^{444} \pmod{144} = 16$

- Step 3: Alice encrypts the message $M = 453$, obtaining the cipher

$$C = M \times K_S \pmod{p} = 453 \times 16 \pmod{144} = 64$$

Alice then sends her public key $A = 144$ and the cipher text $C = 64$ to Bob

- Step 4: Bob derives the encryption key by the calculation:

$K_S = A^b \pmod{p} = 64^{231} \pmod{144} = 64$

Bob then decrypts to cipher to receive the message:

$C \times K_S^{-1} \pmod{p} = 64 \times 1 \pmod{144} = 64 = M$

The message has successfully been decrypted. The message is $M = 64$, the cipher is $C = 64$