Information Security Technologies COMP607

Assignment 1

Student Name: **Dao Minh Duc**

AUT Student ID: **23199710**

1.

a.

2 Rows																														
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	
3 Rows																														
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
4 Rows																														
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			

Key: 4

Plaintext message: **A GOOD NAME IS BETTER THAN GOOD HABITS**

b.

Plaintext message: **KNO WLE DGE ISN EVE RWA STE D**

Key: **EQUATORIAL**

$$Ks = 4; 16; 20; 0; 19; 14; 17; 8; 0; 11$$

P	10	13	14	22	11	4	3	6	4	8	18	13	4	21	4	17	22	0	18	19	4	3
Ks	4	16	20	0	19	14	17	8	0	11	4	16	20	0	19	14	17	8	0	11	4	16
C	14	3	8	22	4	18	20	14	4	19	22	3	24	21	23	5	13	8	18	4	8	19
	O	D	I	W	E	S	U	O	E	T	W	D	Y	V	X	F	N	I	S	E	I	T

The cipher text is: **ODI WES UOE TWD YVX FNI SEI T**

c.

i.

```
sbt7974@Scopius:~/1cii$ echo "IF YOU ARE STANDING STRAIGHT, DON'T WORR
Y IF YOUR SHADOW IS CROOKED" > secretLetter.txt
sbt7974@Scopius:~/1cii$ openssl enc -aes-192-ecb -in secretLetter.txt
-out secretLetter_aes_ecb.enc
enter aes-192-ecb encryption password:
Verifying - enter aes-192-ecb encryption password:
sbt7974@Scopius:~/1cii$ openssl enc -d -aes-192-ecb -in secretLetter_a
es_ecb.enc -out secretLetter_aes_ecb.dcr
enter aes-192-ecb decryption password:
sbt7974@Scopius:~/1cii$ cat secretLetter_aes_ecb.enc
Salted__[I}7����f_], RJ^`�:.��6�4����[�'�^�B��[U9I��}[K�Y���Lÿ�_5��q��a
ю�!���"�k��}�V�osbt7974@Scopius:~/1cii$
sbt7974@Scopius:~/1cii$ cat secretLetter_aes_ecb.dcr
IF YOU ARE STANDING STRAIGHT, DON'T WORRY IF YOUR SHADOW IS CROOKED
```

ii.

```
sbt7974@Scopius:~/1cii$ echo "IF YOU ARE STANDING STRAIGHT, DON'T WORR
Y IF YOUR SHADOW IS CROOKED" > secretLetter.txt
sbt7974@Scopius:~/1cii$ openssl enc -des-ofb -in secretLetter.txt -out
 secretLetter_des_ofb.enc
enter des-ofb encryption password:
Verifying - enter des-ofb encryption password:
sbt7974@Scopius:~/1cii$ openssl enc -d -des-ofb -in secretLetter_des_o
fb.enc -out secretLetter_des_ofb.dcr
enter des-ofb decryption password:
sbt7974@Scopius:~/1cii$ cat secretLetter_des_ofb.enc
Salted__鄃sNC�A�En��2�m��b~�c4B<�\D5�|�P��Yfc����6 <%���6���%���SY3�h�
Isbt7974@Scopius:~/1cii$ cat secretLetter_des_ofb.dcr
IF YOU ARE STANDING STRAIGHT, DON'T WORRY IF YOUR SHADOW IS CROOKED
```

2.

a.

With 7 bits per character, there are $2^7 = 128$ possible values for each character in the password. With 10 such characters in a password, the key space of the password will have the size of 128^{10}.

b.

There are 10 7-bit characters in the password. Therefore, the key length in bits of it is: $10 \times 7 = 70$ bits.

c.

If only 26 lowercase characters are used, we only need at most 5 bits per character to encode the password. Therefore, the key length in bits in this case will be: $10 \times 5 = 50$ bits.

d.

(i) $128/7 = 18$. Therefore, we need at least 18 7-bit characters.

(ii) $128/5 = 25.2$. Therefore, we need at least 26 characters.

3.

The bit sequence that appears the most frequent is `00011111`, so using frequency attack, we can assume that this sequence represents letter "E", which has the ASCII encode of `01000101`. Therefore, we can assume that the key of this cipher text is: `00011111 XOR 01000101 = 01011010`.

With that key, we can work out the rest of the message by adding it to the bits in the cipher text.

- `00010111 XOR 01011010 = 01001101 = m`
- `00001110 XOR 01011010 = 01010100 = t`
- `00011011 XOR 01011010 = 01011010 = a`
- `00010110 XOR 01011010 = 01001100 = l`
- `00001100 XOR 01011010 = 01010110 = v`
- `00010100 XOR 01011010 = 01001110 = n`

The decrypted cipher text is: **MEETATELEVENAM**

This can be interpreted in plaintext as: **Meet at eleven AM**

4.

```
~/Documents                                        [ 12:19:10 ]
❯  cat putty.md5
9047a29b7c2ed333536a7fb6d6c8bae6  putty-0.70-installed.msi

~/Documents                                        [ 12:19:12 ]
❯  cp putty.md5 putty_r.md5
sending incremental file list
putty.md5
             59 100%    0.00kB/s    0:00:00 (xfr#1, to-chk=
0/1)
```

```
~/Documents                              [ ⏲ 12s ] [ 12:20:15 ]
❯  cat putty_r.md5
9047a29b7c2ed333536a7fb6d6c8bae6  putty-0.70-installer.msi

~/Documents                                        [ 12:20:19 ]
❯  md5sum -c putty.md5 putty_r.md5
putty-0.70-installed.msi: OK
putty-0.70-installer.msi: FAILED
md5sum: WARNING: 1 computed checksum did NOT match
```

Therefore, `putty-0.70-installed.msi` is the good copy.

5.

a.

```
sbt7974@Scopius:~/data$ mkdir 05
sbt7974@Scopius:~/data$ cd 05
sbt7974@Scopius:~/data/05$ cp ../notice*.* .
sbt7974@Scopius:~/data/05$ ls
notice1.hmac.txt  notice2.hmac.txt  notice2.txt.sig
notice1.txt       notice2.txt
sbt7974@Scopius:~/data/05$ 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
```

```
sbt7974@Scopius:~/data/05$ 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 ti
me.

Warmest regards.
Dr. Yang
```

b.

```
sbt7974@Scopius:~/data/05$ openssl dgst -hmac comp607 notice1
.txt > notice1.hmac.test
sbt7974@Scopius:~/data/05$ openssl dgst -hmac comp607 notice2
.txt > notice2.hmac.test
sbt7974@Scopius:~/data/05$ cmp notice1.hmac.txt notice1.hmac.
test
notice1.hmac.txt notice1.hmac.test differ: byte 27, line 1
sbt7974@Scopius:~/data/05$ cmp notice2.hmac.txt notice2.hmac.
test
```

Therefore, `notice2.hmac.txt` is the authentic version.

6.

a.

```
~/Documents/06                                    [ 12:36:19 ]
❯ echo "This is some text file" > test1.txt

~/Documents/06                                    [ 12:36:20 ]
❯ md5sum test1.txt > test1.md5

~/Documents/06                                    [ 12:36:31 ]
❯ cat test1.md5
6b101d0e17e56ea3db991454fda2fb5f  test1.txt
```

b.

```
sbt7974@Scopius:~/test$ cat test1.txt
This is a message!
sbt7974@Scopius:~/test$ sed '$ s/.$//' test1.txt > test2.txt
sbt7974@Scopius:~/test$ cat test2.txt
This is a message
sbt7974@Scopius:~/test$ md5sum test1.txt > test1.md5
sbt7974@Scopius:~/test$ md5sum test2.txt > test2.md5
sbt7974@Scopius:~/test$ cat test1.md5
f1ce17046c1dab5dba96e37d02938b96  test1.txt
sbt7974@Scopius:~/test$ cat test2.md5
9b7eb2f1b70b39d14a53846bddea2f4e  test2.txt
```

Conclusion: `test1.txt` and `test2.txt` plaintext files only differs in the last byte (`!` character), but their MD5 hash values are completely different.

c.

```
~/Documents/06                                    [ 12:45:02 ]
❯ md5sum test1.txt > test1md5.txt

~/Documents/06                                    [ 12:45:18 ]
❯ md5sum -c test1md5.txt
test1.txt: OK
```

7.

a.

$M = 513, p = 23, q = 29$

$\Rightarrow N = p \times q = 23 \times 29 = 667$

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

Select $e = 5$

$\Rightarrow d = e^{-1} \mod \phi(N) = 5^{-1} \mod 616 = 493$

Public key: $(e, N) = (5, 667)$

Private key: $(d, N) = (493, 667)$

Encrypt $M = 513$ using public key:

$C = M^e \mod N = 513^5 \mod 667 = 546$

Decrypt $C = 546$ using private key:

$M = C^d \mod N = 546^{493} \mod 667 = 513$

b.

$M = 109, p = 11, q = 23$

$\Rightarrow N = p \times q = 11 \times 23 = 253$

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

Select $e = 3$

$\Rightarrow d = e^{-1} \mod \phi(N) = 3^{-1} \mod 220 = 147$

Signature: $S = M^d \mod N = 109^{411} \mod 253 = 109$

Verify signature:

$$M = S^e \mod N = 109^3 \mod 253 = 109$$

The signature is verified.

8.

a.

First, Alice and Bob agree on a prime number $n = 4787$ and a generator $g = 2$.

Then, they each choose a secret number, a and b, respectively.

Alice chooses $a = 3$ and Bob chooses $b = 5$.

They then calculate their public keys as follows:

Alice: $A = g^a \mod n = 2^3 \mod 4787 = 8$

Bob: $B = g^b \mod n = 2^5 \mod 4787 = 32$

They then exchange their public keys.

Alice receives Bob's public key, $B = 32$, and calculates the shared key as follows:

$$K = B^a \mod n = 32^3 \mod 4787 = 4046$$

Bob receives Alice's public key, $A = 8$, and calculates the shared key as follows:

$$K = A^b \mod n = 8^5 \mod 4787 = 4046$$

Both Alice and Bob now have a shared key, $K = 4046$.

b.

Both Alice and Bob can determine the value of the shared key. The shared key is calculated using the public key of the other party and the secret key of the party itself.

9.

i.

$M = 215, a = 198, b = 231, G = 2, p = 443$

Alice calculates her public key:

$A = G^a \mod p = 2^{198} \mod 443 = 144$

Bob calculates his public key:

$B = G^b \mod p = 2^{231} \mod 443 = 305$

Alice sends her public key, $A = 144$, to Bob.

Bob sends his public key, $B = 305$, to Alice.

Alice computes the shared key:

$K = B^a \mod p = 305^{198} \mod 443 = 321$

Alice encrypts the message:

$C = M \times K \mod p = 215 \times 321 \mod 443 = 350$

Alice sends the cipher text, $C = 350$, to Bob.

Bob derives the shared key:

$K = A^b \mod p = 144^{231} \mod 443 = 321$

Bob decrypts the message: $M = C \times K^{-1} \mod p = 350 \times 321^{-1} \mod 443 = 215$

ii.

$M = 67, a = 11, b = 13, G = 2, p = 17$

Alice calculates her public key:

$A = G^a \mod p = 2^{11} \mod 17 = 8$

Bob calculates his public key:

$B = G^b \mod p = 2^{13} \mod 17 = 15$

Alice sends her public key, $A = 8$, to Bob.

Bob sends his public key, $B = 15$, to Alice.

Alice computes the shared key:

$$K = B^a \mod p = 15^{11} \mod 17 = 9$$

Alice encrypts the message:

$$C = M \times K \mod p = 67 \times 9 \mod 17 = 8$$

Alice sends the cipher text, $C = 8$, to Bob.

Bob derives the shared key:

$$K = A^b \mod p = 8^{13} \mod 17 = 9$$

Bob decrypts the message: $M = C \times K^{-1} \mod p = 8 \times 9^{-1} \mod 17 = 67$