CS 458 A3

Name: Thomas Lim Jun Wei
ID: 20700973

Question 1

1a) XOR of 2 original plaintexts done using a Xor application. Labelled as "xor"

1b) To derive out my two plaintexts, firstly, I used an XOR application to get the XOR hex string from my two-cipher text. Next, I applied a method known as crib dragging, which is first start from deriving a frequently use key word such as " the "(do note I padded space before and after "the" as my 5-character key crib. By running a python program that xor the hex string with my crib to find if there is any string can be found. In this instance, I found "the s" around the 54^{th} byte area. From there, I increase the crib by padding vowels of 5 character to see if any other characters can be spotted and use guessing and checking iteration around the 54^{th} byte region to derive out my plaintexts.

Question 2 - GnuPG

2a) Generated GnuPG private key as key.asc.

2b) Signed nikita.volodin@uwaterloo.ca key to nikita.volodin-signed.asc

2c) Task to do:

1) Create a message containing my name and userid.
2) Sign with key.asc.
3) encrypt with nikita.volodin key

Instruction ran: gpg --recipient nikita.volodin@uwaterloo.c --armor --sign --encrypt message

2d) The importance of fingerprints in GnuPG is to ensure that a key's fingerprint is verified with the key's owner. This is to guarantee that any communication done with the key's user is authenticated and it's the true owner to the key. If the fingerprint is identical to the fingerprint of the other party's owner, this ensure that both parties received the correct copy of the key to ensure a more secure communication.

Source learned:

https://www.digitalocean.com/community/tutorials/how-to-use-gpg-to-encrypt-and-sign-messages-on-an-ubuntu-12-04-vps
https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/4/html/Step_by_Step_Guide/s1-gnupg-export.html
http://www.thegeekstuff.com/2013/04/gnupg-digital-signatures/
https://www.gnupg.org/gph/en/manual/x110.html

Question 3 – Diffie-Hellman

3a) If p = 83, base = 50, and Alice's secret parameter = 23, Bob's secret parameter = 12.

Then public value that Alice gives to Bob:

$$A = base^{Alice's\ secret\ para}\ mod\ p = 50^{23} mod\ 83 = 35$$

Then public value that Bob gives to Alice:

$$B = base^{Bob'ssecret\ para}\ mod\ p = 50^{12} mod\ 83 = 16$$

To get the secret key:

$$S = Bob\ Public^{Alice's\ secret} mod\ 83 = Alice\ Public^{Bob\ secret} mod\ 83$$

$$S = 16^{23} mod\ 83 = 35^{12}\ mod\ 83 = 28$$

Solution of working is attached behind.

3b) No, Malloy can't recover original secret values of a and b as these two parameters wasn't mutually agreed. To find out the secret values would be computationally difficult for that him to determine the values. Similarly, it would be computationally difficult for Alice to find out about Bob's secret parameter and vice versa.

Each multiplication varies as the square of n, which must be very large. The number of multiplications required by the exponentiation increases with increasing values of the exponent, Alice and Bob in this case.

3c) If Mallory behaves as active MITM attacker, she can abuse DH protocol by forcing both Alice and Bob to communicate in a small subgroup that uses a small modulus prime number. This would give Mallory an advantage to test by doing a brute force search to derive the secret parameter of Alice and Bob.

To prevent this, Both Alice and Bob make use of digital signature to authenticate as DH key exchange main vulnerability is its prone to MITM attack. This authentication protocol allows

the establishment of a shared secret key between two parties with mutual entity authentication and mutual key authentication.

Working for QN 3