Damon Del Priore

I pledge my honor that I have abided by the Stevens Honor System

Homework 1

Problem 2:

1.) Confidentiality and Integrity should be considered in the design of a secure protocol for solving the problem. These ensure that the asset is only viewed and modified by authorized parties. Confidentiality becomes relevant because they must use encryption to not give any information to an eavesdropper. Integrity becomes relevant because it is crucial that the information sent by Alice to Bob stays intact, as well as what Alice will receive from Bob. This would result in Alice receiving the correct information in verifying the two keys are in fact the same, and more importantly not revealing any - information to someone that has intercepted both messages. This ensures that they are safely communicating on the insecure channel.
2.) The above protocol does not satisfy the two security properties mentioned in one. The attacker will have access to X (sent by Alice) and access to Y (sent by Bob). Alice uses xor on her key, K_a with a random value r and sends it to Bob. The attacker has the value X, but knows nothing about the message. When Bob receives the message, Bob takes his key, K_b, and uses xor on what Alice sent to generate Y. $Y = (K_b \oplus K_a \oplus r)$. Given the two keys are in fact equal, this means that Y = r. When Bob sends Alice Y, the attacker has just been given K_a, by simply using the first message he received, X. In fact, $(X \oplus Y) = K_a$. Therefore, Alice would not be able to confirm they have the same key with this method, because if they do have the same key, the key would have been leaked to the attacker. Now that the attacker has the key, he could also prevent Alice from knowing Bob has the key, or allow Alice to know. If he allows Alice to know, the attacker now has access to all of their messages. The fundamental of a one-time-pad use like this is that it only will work in one direction, as Bob having the same key simply decrypted for the attacker.

Problem 3:

1.) Due to small size of the message space, I believe any attacker would be able to decide between the two messages. Because the attacker has two possible messages, he will be able to narrow down the key space. If the period is one, then the key would be revealed right away. If the key is two, then the key would be revealed by the third letter. If the key is three or four, then the key would be revealed on the fourth letter. Due to the key size being less than or equal to the message space, this is going to offer something equivalent to one time pad, but due to the narrowed key space and simplicity of the Vigenère Cipher, it would be likely for the attacker to receive the key with period sizes 1,2,3, or 4.
2.) The mono-alphabetic substitution cipher is trivial to break with chosen plain text attacks because the attacker would be able to encrypt very specific messages to obtain the entire key. The attacker would only need 25 letters of chosen plaintext to receive the entire key. The plaintext “ABCDEFGHIJKLMNOPQRSTUVWXY” would reveal the entire key.

The key would lie uniformly below the plain text, as a monoalphabetic substitution cipher is going to have each letter of the alphabet mapped to a unique letter in the alphabet. This means that A and B in plaintext may not map to the same letter in ciphertext. Whichever letter is not in the returned plaintext is going to be mapped to Z, for Z is the final letter in the alphabet that does not appear in "ABCDEFGHIJKLMNOPQRSTUVWXY." The shortest chosen single-message plaintext, which is a valid modern English message, that would successfully recover the entire secret key is "Pack my box with five dozen liquor jugs." This message contains 32 letters and the entire alphabet. Under the conditions that that the key is unknown, the language in transmission is unknown, and the attacker is unable to perform chosen-plaintext attacks, the mono-alphabetic substitution cipher is perfectly secure because if the attacker is in possession of any ciphertext, it will reveal nothing about the underlying plaintext. It is also equally likely for any given plaintext to map to the same cipher text, based on the unknown key for the encryption. The mono-alphabetic substitution cipher has a key space of 26! which is incredibly large for a brute force attack to be time efficient, especially on a very large cipher text. Intuitively, the mono-alphabetic substitution cipher will be perfectly secure, due to the unknown key, the unknown language to prevent letter frequencies narrowing the key space, and the attacker not being allowed to encrypt their own messages.

Problem 4:

Testing testing can you read this
Yep I can read you perfectly fine
Awesome one time pad is working
Yay we can make fun of Nikos now
I hope no student can read this
That would be quite embarrassing
Luckily OTP is perfectly secure
Didnt Nikos say there was a catch
Maybe but I didnt pay attention
We should really listen to Nikos
Nah we are doing fine without him

1.) My cryptanalysis strategy was to write an algorithm that will find the exclusive or of every combination of every cipher text. With every combination of exclusive or, I was able to fill in some blanks to complete message one "Testing testing can you read this." With regards to one-time pad, I have obtained a message along with its cipher text. This allowed me to have $M_1 \oplus C_1 = M_1 \oplus M_1 \oplus Key = Key$. After obtaining the key, I simply had to run $K \oplus C_i$ for all i cipher texts. The 11 exclusive or statements with the key and each cipher text then printed the entirety of the plaintext.

Cryptoanalzye.java functions ….

inAlphabet takes in a char and returns true if it is in the alphabet, otherwise it returns false
splitLetters takes in a string creates an array list of strings size 2
xorCiphertext computes the exclusive or of every combination of cipher text and prints out the results in hex, as well as the results with respect to the alphabet using inAlphabet
printPlaintext takes in an array list of string containing the plain text I solved and prints out the plain text using the xor mentioned above with the inAlphabet function 00100001

String: youfoundthekey!congratulations!!!

Hex: 796f75666f756e647468656b657921636f6e67726174756c6174696f6e73212121

2.) September 24 KEY FROM CIPHERTEXT - 796f75666f756e647468656b657921636f6e67726174756c6174696f6e73212121
September 25 - 8c2d1eac1bf0b0910888b354d516600f1c9dedd5804def51c7931360dea6dee021
September 26 - 5ff020e301b99b85c2bb9cef4d0e9041fcd22bcb2d970e5b2c84f03132b9d62521
September 27 - e18f8411d780971547ecde5e6008f98dd790959dfee4ac85c47092323836b8be21
September 28 - 9b703b55679ddc9e8d54a42b6827eaf8512e309c6a513e2a6faca2bc736f864c21
September 29 - 2b62a55851f5992443b85bb38e311e0d2fd6be888e1e475efcde53b5ac9d5e6c21
September 30 - afe2d9750f320661fa5fb4461c8ae71c9d52abb65e609278177faa7bb04f0d8421
October 1 – 4941a11cbec279da75e878460876b94d91e3b396933cb3897afbb3ffa04194b921
October 2 – cd53eb2a0e13c77231b9e071f2a32ce70b9abff8c5d5fbd1efcfc666d729057d21
October 3 – 1b0ab6072540a4df150adbfe28654b127fb032e71b2cdc5cd924723fb0c0bc8021
October 4 - 50a0ee2f6b5721a86e7da7e395ba6fb14ae9f0b4e6e58cd5a7b97c546ca802a321
October 5 – 548e06c5e50d1fda04ebbfe201d0177b491998bfb5e960e479ea95db1200526721
October 6 – ef5e37d6b3f5b9e2048352a885c87ca1fe47d07ae743594007882bc9be96ba2021
October 7 – 6120196bc620b54c7b9ead0ed98f22b77689df10c163014a6118c91684bef9c521
October 8 KEY TWO WEEKS LATER– 683587a47e3db2fda56325dbcb84b0541eb4d673abf437be60691d41d269456421