Assignment 1

SENG2250

Sam Dolbel – 3130069

1. Brute-Force Attacks

a) For a sequence of 7 characters where there are 42 possibilities for each character, the total number of password combinations is 42^7, or **230,539,333,248** combinations.
If hypothetically the password consisted of 7 **unique** characters, the total number of passwords would be (42 * 41 * 40 * 39 * 38 * 37 * 36), or 135,970,773,120.

b) To calculate the **average** time that it would take an adversary to guess the password, we add the best-case scenario (the 1st guess is correct) and the worst-case scenario (the 230,539,333,248th guess is correct) and divide by 2. In other words, ((1 + 230539333248) / 2) = **115,269,666,624.5** guesses**.**

The next step is to divide the result by 1,000,000. This calculates the **average time in seconds** to guess the password. (115269666624.5 / 1000000) = **115,270 seconds** to the nearest whole second.

Finally we convert this value to days. Since there are (60 * 60 * 24) = **86,400** seconds in a day, brute-forcing the password will take (115270 / 86400) = **1.334 days** (to 3 decimal places).

c) Suppose that the password is being guessed sequentially – that is, the adversary attempts to guess the first character, finds the character, attempts to guess the second character, and so forth.
In this case, the first character can be guessed in ((42 + 1) / 2) = 21.5 attempts on average.
The average number of attempts for the password to be guessed will be (((42 + 1) / 2) * 7) = 150.5 attempts.
At 2,000,000 password character attempts per second, the password will be guessed in (150.5/2000000) = **0.00007525 seconds**.

2. Block Cipher and Operation Modes

a) As hex numbers are composed of pairs of 4-bit characters, the 128-bit key is:
bbfdf2a13c14e0e43594accb783e16de.
The 512-bit hexadecimal plaintext is:
7468656c6966656f66616d616e756e62
757264656e6564627973757276697661
6c616e6466726565746f637265617465
69736275696c746f6e616e6f74686572.

b) The Initialisation Value is a random figure with length equal to the plaintext block – that is the size of the block used **per stage**. Using a hexadecimal generator located at https://www.browserling.com/tools/random-hex and specifying a 128-bit string, we choose the following IV:
49967cabd3446562c016b1fe22b86f79.

c) The process starts by selecting 1 512-bit string of hexadecimal characters – this is your plaintext string to be encrypted. Next, we choose 2 128-bit strings of hexadecimal characters: one for an IV and the other for the secret key. The size of the IV must match the size of a plaintext block. In this example we will split the 512-bit plaintext into 4 128-bit plaintext blocks, which will serve as our input over 4 rounds of AES. The key is secret and must be exchanged privately between the parties involved, but the IV is public and will be sent with the plaintext.

In the 1st round, we input the first block of plaintext: 7468656c6966656f66616d616e756e62. This will be XORed with the IV: 49967cabd3446562c016b1fe22b86f79**.** The result of this will be encrypted using our secret key: bbfdf2a13c14e0e43594accb783e16de. After this is encrypted, we have our first block of ciphertext: 0930b66fb540f878ac24e6c3da36dd1b**.**

In the 2nd round, the block of ciphertext we received from the 1st round will be XORed with the 2nd block of plaintext: 757264656e6564627973757276697661. The secret key remains the same across all 4 rounds, so we encrypt the XORed result using that. The result will be our 2nd block of ciphertext: 369b015977867dcec3a97b223acbf2a6.

In the 3rd round, the block of ciphertext we received from the 2nd round will be XORed with the 3rd block of plaintext: 6c616e6466726565746f637265617465. The encrypting with the key results in our 3rd block of ciphertext: aea9bb29a20d801d70196f218b29fa3a.

In the 4th and final round, we XOR the 3rd block of ciphertext with the final block of plaintext, encrypt it with the key and output our final block of ciphertext: 4649b88615abdd9592015c0483b6e234.

d) The full 512-bit ciphertext is:
0930b66fb540f878ac24e6c3da36dd1b
369b015977867dcec3a97b223acbf2a6
aea9bb29a20d801d70196f218b29fa3a
4649b88615abdd9592015c0483b6e234

e)

Key: **bbfdf2a13c14e0e43594accb783e16de**

Plaintext:
7468656c6966656f66616d616e756e62757264656e656462797375727669766 16c616e6466726565746f6372656174656973627369 6c746f6e616e6f74686572

IV:
49967cabd3446562c016b1fe22b86f7956741a870fe5a87a699420e57ad06e485fd710b5e54e6c9
f45a9876975979c302eba9a7a428f0915cb0d63553a6b6951

Round 1:
Input to AES: **7468656c6966656f66616d616e756e62**
Output of AES: **0930b66fb540f878ac24e6c3da36dd1b**
Round 2:
Input to AES: **757264656e6564627973757276697661**
Output of AES: **369b015977867dcec3a97b223acbf2a6**
Round 3:
Input to AES: **6c616e6466726565746f637265617465**
Output of AES: **aea9bb29a20d801d70196f218b29fa3a**
Round 4:
Input to AES: **69736275696c746f6e616e6f74686572**
Output of AES: **4649b88615abdd9592015c0483b6e234**

Entire ciphertext:
0930b66fb540f878ac24e6c3da36dd1b369b015977867dcec3a97b223acbf2a6aea9bb29a20d801
d70196f218b29fa3a4649b88615abdd9592015c0483b6e234

3. Hash Functions and Digital Signatures

a) Given K_3, we can easily compute K_4 and K_5 but not K_1 and K_2. This is the idea of a one-way hash function – with the value of data *K* known it is easy to calculate function *H(K)*, but with the value of function *H(K)* it is very difficult to calculate data *K*. Since we know the value of K_3, we can calculate $K_4 = H(K_3)$. Since we now know the value of K_4, we can calculate $K_5 = H(K_4)$.
For example, suppose that the hash function *H(K)* is (K^2 mod 4000). If $K_3 = 366$, we can easily calculate $K_4 = (366^2 \bmod 4000) = x$ (which is 1956). However, it is much more difficult to accurately calculate $K_2 = (x^2 \bmod 4000) = 366$.

b) This demonstration of Alice's power can be expressed with a Message Authentication Code.
Alice shares with the people the *Time.* This can be represented as the secret key ***k***.
Although she cannot leak the *Event*, she can provide a cryptic clue. For example, "sunset cats and diamond dogs" may be code for "a devastating rainstorm in a Western Australian mining town". This can be represented as the encrypted message ***m***.
Alice shares her message in the form *Mac*(*k, m*) -> ***t***, where *t* is a tag sent to the people.

The people then attempt to decode this message. They have the *Time* (*k*) and Alice's message (*Mac*(*k, m*)). Now they can **verify** her message. Suppose that at the time that Alice stated, the town of Kununurra and the nearby Argyle Diamond Mine were flooded by unusual levels of rainfall – the message can be verified as accurate. On the other hand, if nothing relevant happened, the message can be proved inaccurate. The people decrypt her message in the form *Ver*(*k*, *m'*, *t*) -> True/False – that is, they take Alice's tag ***t***, the secret key ***k*** and the decrypted message ***m'***, and they determine whether it is true or false.

To clarify, this covers the three requirements in the following ways:
1) The message is cryptic enough that nobody knows what the event is until it happens, much less act upon it.
2) At the same time, the message is specific enough that it cannot be mistaken when the event comes to pass.
3) Alice cannot deny the prediction because she has already sent the cryptic clue and provided a time.

4. Cryptanalysis on Monoalphabetic Cipher

a) The plaintext reads:
THERE IS A STORY ABOUT TWO PEOPLE WHO DECIDED TO PLAY A GAME IN WHICH THE ONE WHO CALLS THE HIGHEST NUMBER WINS.

b) **E**: When performing cryptanalysis on a monoalphabetic cipher, this is the easiest letter to start with. Given any sufficiently long and meaningful English sentence, E will appear significantly more often than any other letter. Since F has the highest frequency, it can be deduced that **F = E**.
T: Now that we are reasonably certain that F = E, we can see that the first word is ??**E**?**E** (letters in bold are decrypted). The first letter U appears the second most frequently in the sentence so, based on frequency grouping, we can deduce that U = T, A, O, I, N, S, H or R. Of these, the most obvious choice would appear to be T?E?E, used in words such as "there" and "these". This is supported by the rest of the sentence, where we can see the group T?E appear twice. It is very likely that **U = T**.
H: Since we have decrypted **T** and **E**, we have a group of words **T**?**E** that appears twice. The frequency of "I" is relatively high – we can place it in frequency group II. There are 3 words that can be generated from this group: "TIE", "TOE" and "THE". If we compare this to the first word in the ciphertext, we have "**TIE**?**E**", "**TOE**?**E**" and "**THE?E**". Given that only the last group makes an obvious English word, and that the most common English word is "the", we can assume that **I = H**.
O: Given the values we currently have, the tenth word "UP" becomes **T?**. There is only one English word that this can be – "to". Therefore **P = O**;
W: Now that we know that P = O, we can see that the 7th word "XIP" has become "?**HO**". Similar to the previous example, there is only one word that this can be – "who". This means that **X = W**.

At this point it becomes clear that this cipher is a Caesar cipher, or shift cipher. That is, the plaintext is encoded by shifting the value of each letter one place to the right. The exception to this would be Z = A, as it would shift around to the start of the alphabet.