Pledge: *I pledge my honor that I have abided by the Stevens Honor System.* - **Eric Altenburg**

Problem 1: Shared or forgotten keys?
Long ago, Alice and Bob shared an n-bit secret key but not they are no longer sure they still possess the same key. To verify that the key k_a currently held by Alice is the same as the key k_b currently held by Bob, they need to communicate over an insecure channel.

(1)

1. Confidentiality: This is relevant because since they are using an insecure channel now, an attacker, say Eve, can eavesdrop on their conversation now and figure out the key. If confidentiality is thus considered, then Eve will be less likely to find the key.
2. Integrity: The key is going to be used for all of their communications, therefore, it is important that it is secure or else all of their messages are not going to be secure.

(2)

For this case, the property of confidentiality is not satisfied. Alice generates the n-bit value r which is random, then performs an $\oplus$ and sends the result to Bob. If there is an attacker, Eve, who is listening, then she can perform the same operation as Bob and obtain her own key. Integrity is also not satisfied because if their key was compromised as mentioned previously, then none of their messages are secure anymore.

Problem 2: Perfect or imperfect ciphers?

(1)

1. $t = 1$: The algorithm is insecure because an attacker can easily brute force it. This has a similar structure to a simple cipher where the first and second characters could be used and the difference between the two can give an attacker the information they need.
2. $t = 2$: Because there are only two passwords where *ab* and *cd* are at most two letters apart or *be* and *dg* are at most two letters apart as well, an attacker can pick one password or the other. Since the key repeats itself every two letters, it is equally likely for both passwords to be used, and so it is easy to figure out which one is being used for a shift in the cipher.
3. $t = 3$: This means there are two patterns: $x, y, z, x + 3$ or $x, y, z, x + 5$. Because there are only two passwords, an attacker can easily figure out the passwords.
4. $t = 4$: Both passwords are of length four, therefore, the attacker does not gain any new information.

(2)

A mono-alphabetic substitution cipher is trivial to break because a chosen-plaintext attack consisting of 25 letters can be used to figure out the key. With this, the attacker can then crack the cipher and then be able to decrypt all future messages. The reason behind this is because there are 26 distinct letters that will correspond to some other letter, and so 25 letters is all that is needed because the 26^{th} letter would be the only one left with the process of elimination. The shortest single-message plaintext that is a valid English message that can recover the key is *jived fox nymph grabs quick waltz*. This mono-alphabetic substitution cipher would only be perfectly secure against a cipher-text only attacker if the probability that the mapping of the letters and the ciphertext are equally as likely to occur.

Problem 3: Crypt-analyze this!

The text reads:

- 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

To successfully decipher this, I took the XOR of the first 2 ciphertexts. Then using crib dragging, I would use a guess word to go over the XOR to see if I can see if any probable ciphertext can be found. This gave me a single plaintext and the 2 initial ciphertexts, then I did another XOR between them to get the key. To get the rest of the words, I mapped the key over the rest of the ciphertexts to produce the rest of the messages with the help of an ASCII converter. The key is *youfoundthekey!congratulations!!!*.