Assignment 6

1. **This problem deals with storing passwords in a file.**
   1. Why is it a good idea to hash passwords that are stored in a file?

Because by storing hash passwords in a file, it will prevent intruders from directly getting the password if by any chance they get the file. In order for the intruder to find the password, they would need to find a hash a lot of passwords in order to find the one on file.

* 1. Why is it a much better idea to hash passwords stored in a file than to encrypt the password file?

It is better idea to hash the password because if we encrypt the password file, that means that we would need a key, and since we need a key, it is most likely that the key is also contained on the same computer as the password file, therefore if Trudy can get the password file, the chances that she will also get the key are very high. But if we use hash instead, Trudy would have to try to hash all possible keys to find the password that matches with the hash on the file.

* 1. What is a salt and why should a salt be used whenever passwords are hashed?

Salt is a random but not secret number that we use to hash it together with the password, yielding a different hash for the same password if the salt is different. Salt should be used whenever passwords are hashed because if we have a file full of password hashes without salt, it would be very easy for Trudy to compute hashes of possible passwords and then compare it to all the password hashes, But if we use Salt, Trudy would not be able use precomputed hash of passwords and as well as when she compute hashes for possible passwords, she will not be able to compare her hash to all the ones on file because the salt is different, therefore making the job a lot more difficult for Trudy to find the passwords because she would need to compute a new hash for every salt.

1. **Research has shown that most people cannot accurately identify an individual from a photo. For example, one study found that most people will accept an ID with any photo that has a picture of a person of the same gender and race as the presenter.**
2. It has also been demonstrated that when photos are included on credit cards, the fraud rate drops significantly. Explain this apparent contradiction.

When having photos on credit card, the fraud rate drops significantly, because if credit cards didn’t have photos, it would be very easy for people to use stolen credit card and they would only need signature. But if the credit card contains a photo, it will limit the amount of people that can use stolen credit card to the owner’s gender and race. In the case of no picture, the chances for sellers to tell that the credit card is stolen are very low, but if the credit card has a picture, sellers will at least be able to sometimes tell if the credit card does not belong to the right owner.

* 1. Your easily amused author frequents an amusement park that provides each season passholder with a plastic card similar to a credit card. The park takes a photo of each season passholder, but the photo does not appear on the card. Instead, when the card is presented for admission to the park, the photo appears on a screen that is visible to the park attendant. Why might this approach be better than putting the photo on the card?

The approach of showing the picture on the screen instead of the card can improve the attendant’s recognition and also prevent forging. By putting the picture on the screen, it will allow attendant to look at a more detailed picture instead of a really small picture on the card, which will help compare if the pass really belongs to that person. Another benefit is that the picture is stored on a database instead of printed in the card, that way people cannot forge the card with their own picture on it, because the picture is on the database, not on the card.

1. **Suppose all passwords on a given system are 8 characters and that each character can be any one of 64 different values. The passwords are hashed (with a salt) and stored in a password file. Now suppose Trudy has a password cracking program that can test 64 passwords per second. Trudy also has a dictionary of 230 common passwords and the probability that any given password is in her dictionary is 1/4. The password file on this system contains 256 password hashes.**
   1. How many different passwords are possible?

The number of different possible passwords is 648 = 248

* 1. How long, on average, will it take Trudy to crack the administrator's password?

To crack the administrator’s password, Trudy only needs to crack one password. The administrator’s password has a ¼ chance to be on the dictionary of 230 passwords, and there is ¾ chance that it is not on the dictionary which means that Trudy would have to compute possible passwords which is a total of 248.

By doing the math ¼ \*(229) + ¾ \*(247) = 246.58 hashes, since we can do 26 hashes per second, it will take 246.5 / 26 = 240.58 seconds

* 1. What is the probability that at least one of the 256 passwords in the password file is in the dictionary?

1-(3/4)256  = 1, the probability of at least one if the 256 passwords to appear on the dictionary is 1

* 1. What is the expected work for Trudy to recover any one of the passwords in the password file?

The expected work for Trudy to recover any password in the file is **230 / (1/4) = 232**

1. **Let h be a secure cryptographic hash function. For this problem, a password consists of a maximum of 14-characters and there are 32 possible choices for each character. If a password is less than 14 characters, it's padded with nulls until it is exactly 14 characters. Let P be the resulting 14 character password. Consider the following two password hashing schemes.**

**(i) The password P is split into two parts, with X equal to the first 7 characters and Y equal to the last 7 characters. The password is stored as (h(X), h(Y)). No salt is used**

**(ii) The password is stored as h(P). Again, no salt is used.**

**Note that the method in scheme (i) is used in Windows to store the so-called LANMAN password.**

* 1. Assuming a brute force attack, how much easier is it to crack the password if scheme (i) is used as compared with scheme (ii)?

In scheme (i) we would need 2\*(327) = 236 hashes to crack the password, while in scheme (ii) it would require 3214= 270 hashes to crack the password.

* 1. If scheme (i) is used, why might a 10-character password be less secure than a 7-character password?

In LANMAN the algorithm automatically pads the password with nulls, in a 7 character password, the 2nd half would all be null and would tell nothing about the actual password. But if we have a password with 10 characters, the 2nd half of the password will be easier to crack because it only contains 3 characters from the actual password, after cracking the 2nd half, we have 3 letters from the actual password and with that information, it can lead us into guessing what the first part of the password is.

1. **Many websites require users to register before they can access information or services. Suppose that you register at such a website, but when you return later you've forgotten your password. The website then asks you to enter your email address, which you do. Later, you receive your original password via email.**
   1. Discuss several security concerns with this approach to dealing with forgotten passwords.

This approach is insecure, because if Trudy by any chance get access to Alice’s email because Alice did not log out, Trudy could figure out the original password by just saying that she forgot it, and she would receive the original password which might compromise other accounts. Another security concern is the way the website stores their user’s password, because websites are supposed to only store the hashes of the passwords, there shouldn’t be a way for the websites to send the original password of the user back to their email.

* 1. The correct way to deal with passwords is to store salted hashes of passwords. Does this website use the correct approach? Justify your answer.

No, the correct way to store passwords, is by hashing the original password with salt, this way will prevent intruders from getting the file with actual passwords, and by using salt, it will increase the difficulty of cracking all the passwords. But in this website, if the website is able to email the original password to the user, it means that the website is storing the password in a way that it can be recovered, which is not the correct approach.

1. **Suppose that Trudy has a dictionary of 2n passwords and the probability that a given password is in her dictionary is p. If Trudy obtains a file containing a large number of salted password hashes, show that the expected work to recover a password is bounded by 2n-1(1+2(1-p)/p). Hint: As in Section 7.3.5, Case IV, ignore the highly improbable case where none of the passwords in the file appears in Trudy's dictionary. Then make use of the fact that = 1/(1 - x) and also = x/(1 - x)2, provided |x| < 1.**

We want to use the formula, but is missing (1-p)0, so we can’t use the formula yet. To solve this problem we will temporary give (1-p)0 to it which equals to 1. Therefore: , since we temporarily gave 1 to it, we have to take it back now, so… , substituting the above progress, we get:

, Hence proven.

1. **Suppose that when a fingerprint is compared with one other (nonmatching) fingerprint, the chance of a false match is 1 in 1010, which is approximately the error rate when 16 points are required to determine a match (the British legal standard). Suppose that the FBI fingerprint database contains 107 fingerprints.**
   1. How many false matches will occur when 100,000 suspect fingerprints are each compared with the entire database?

With 100000 suspects, it would mean 100000 \* 107 comparisons which equals **1012 comparisons,** since there is 1 false match with each 1010 comparisons, the total false matches that would occur when 100000 suspect’s fingerprint is compared to the database is **1012 / 1010 = 100 false matches**

* 1. For any individual suspect, what is the chance of a false match?

The chance for one individual to get a false match is **107/1010 or 100/100000**, which both equals to **0.001** or **0.1%**

1. **This problem deals with biometrics.**
2. Define fraud rate.

Fraud rate is the percentage times that unauthorized users are mis-authenticated as someone else and provided access. It doesn’t actually have to be a “fraud” it just means that the system is authenticating the user/intruder as someone else in the system.

1. Define insult rate.

Insult rate is the percentage of times that an authorized user is not authenticated as the user itself.

1. What is the equal error rate, how is it determined, and why is it useful

Equal error rate is the rate where fraud and insult rate are the same, it is determined by the adjusting the accuracy requirement of a match until both fraud and insult rate are the same. This measurement is useful to compare different biometrics to see which ones are better.

1. **In one episode of the television show MythBusters, three successful attacks on fingerprint biometrics are demonstrated.**
2. Briefly discuss each of these attacks.

On the first attack, they first print the fingerprint on acetate, make sure that the edges are clear enough and then use a copper coated circuit board to etch the fingerprints and then use ballistic gel to stick the copper finger prints and then use it as a finger, it was used on the normal computer to login and it was successful. This same attack was used on the door and it also worked.

The 2nd attack, they used latex with the copper edges imprinted to it to unlock the door, at start it did not work, but after licking it and put it directly to his thumb, and it worked.

The 3rd attack, they tried just using the photocopied paper, licked it and stick it on the door lock’s reader and it also worked.

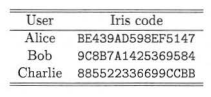
1. Discuss possible countermeasures for each of the attacks in part a. That is, discuss ways that the biometric systems could be made more robust against the specific attacks.

For the first attack, a counter measurement would be to be able to accurately measure pulse and temperature of the finger, as seen in the TV show, the ballistic gel was really thick which obviously had no pulse nor temperature, yet it was still able to fool the door lock.

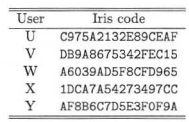
For the second attack, a counter measurement would be to find a difference between latex and human flesh, because the latex attack could provide temperature, pulse and they could lick it for sweat.

For the third attack, a counter measurement would be to actually do the jobs that the lock was promising, it has to measure pulse, temperature, etc. it was ridiculous to be able to unlock the door with a piece of licked paper, the security was lower than the computer thumb scanner.

1. **Suppose that a particular iris scan system generates 64-bit iris codes instead of the standard 2048-bit iris codes mentioned in this chapter. During the enrollment phase, the following iris codes (in hex) are determined**

****

**During the recognition phase, the following iris codes are obtained.**

****

**Use the iris codes above to answer the following questions.**

* 1. Use the formula in equation (7. 1) to compute the following distances:

d(Alice, Bob), d(Alice, Charlie), d(Bob, Charlie).

Alice: 1011111001000011100110101101010110011000111011110101000101000111  
Bob: 1001110010001011011110100001010000100101001101101001010110000100  
Charlie: 1000100001010101001000100011001101100110100110011100110010111011

d(Alice, Bob) = 29/64 = 0.45  
d(Alice, Charlie) = 39/64 = 0.61  
d(Bob, Charlie) = 34/64 = 0.53

* 1. Assuming that the same statistics apply to these iris codes as the iris codes discussed in Section 7.4.2.3, which of the users, U,V,W,X,Y, is most likely Alice? Bob? Charlie? None of the above?

d(Alice,U) = 36/64 = 0.56  
d(Bob, U) = 37/64 = 0.578  
**d(Charlie, U) = 11/64 = 0.17**

d(Alice, V) = 30/64 = 0.468  
d(Bob, V) = 28/64 = 0.437  
d(Charlie, V) = 30/64 = 0.468

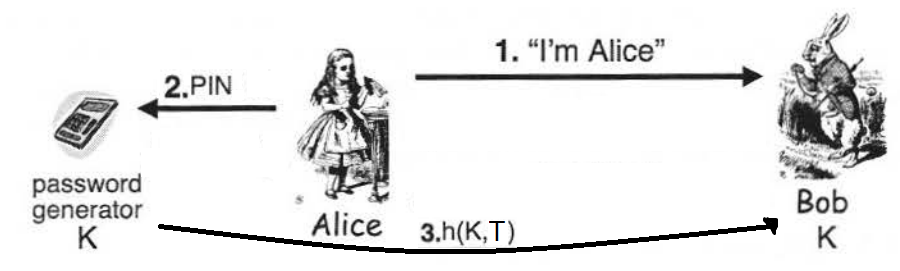
**d(Alice, W) = 10/64 = 0.156**  
d(Bob, W) = 31/64 = 0.48  
d(Charlie, W) = 35/64 = 0.55

d(Alice, X) = 34/64 = 0.53  
**d(Bob, X) = 10/64 = 0.156**  
d(Charlie, X) = 36/64 = 0.56

d(Alice, Y) = 32/64 = 0.5  
d(Bob, Y) = 27/64 = 0.42  
d(Charlie, Y) = 31/64 = 0.484

**Alice = W, Bob = X, Charlie = U**

1. **A popular "something you have" method of authentication is the RSA SecurID. The SecurID system is often deployed as a USB key. The algorithm used by SecurID is similar to that given for the password generator illustrated in Figure 7.8. However, no challenge R is sent from Bob to Alice; instead, the current time T (typically, to a resolution of one minute) is used. That is, Alice's password generator computes h(K, T) and this is sent directly to Bob, provided Alice has entered the correct PIN (or password).**
   1. Draw a diagram analogous to that in Figure 7.8 illustrating the SecurID algorithm.



* 1. Why do we need T? That is, why is the protocol insecure if we remove T?

As we can see, the password generator is creating a hash of (K,T), if we remove T, we the password generator will be giving us h(K) every time and since K doesn’t change, that means that we would get the same hash/password every time, therefore making insecure.

* 1. What are the advantages and disadvantages of using the time T as compared to using a random challenge R?

The advantage of using of using T is that we don’t need to wait for Bob to give us the challenge to generate the password/hash. The disadvantage is that when the key K is compromised, it becomes easy to predict the next output of the generator.

* 1. Which is more secure, using a random challenge R or the time T? Why?

Overall, I think the challenge R will be more secure because if we use Time T, intruders will have more information about what information was used to calculate the hash/password. But if we use Challenge R, the intruder will not know what R is and therefore making it more secure.