CPSC 329 Assignment 2
David Ng
30009245
T04

1.

1. We recall that entropy is log_2(N), where N is the number of guesses required. However, it is difficult to determine the exact minimal number of guesses that are required to be sure that one can obtain access to the system given that the system can repeatedly change the faces that appear after each set of five challenges. Thus, we make the simplifying assumption that the set of five challenges does not change between successive attempts to access the system. In this case, there are 9 possible choices for one challenge. Since we have a total of 5 challenges, this means that we need 9^5 = 59049 guesses. Thus, entropy is log_2(59049), which is approximately 15.85. Alternatively, we could use the definition of entropy that states that N is the number of different possible passwords. In this case, we need to find the number of possible combinations of 5 faces out of 100. This is 100 choose 5 = 100!/(95!5!) = 75287520. Thus, the entropy of this is log_2(75287520), which is around 26.17. Since this acts as an upper bound, we know for certain that the entropy of the passwords given that the face database is public is no greater than 26.17.

2. We note the distinction between obtaining access to the system (impersonating the user by correctly responding to the set of 5 challenges), and obtaining the user's password. Obtaining access to the system implies that the adversary is repeatedly attempting to enter passwords in order to access the system. In the previous answer, it was found that this would require 59049 guesses to exhaust, assuming that the system does not repeatedly change the challenges between sets. Thus, the probability that the adversary succeed on their first attempt is 1/59049. To guess the user's password however, the adversary knows that there are 75287520 unique passwords. The probability that the adversary guess the one password of the particular user out of this total number of possible passwords is therefore 1/75287520, which is approximately 0.00000001328.

3. The security of the database is independent of the total number of different passwords possible assuming that the system does not change the set of 5 challenges that it presents to the user. If this was the case, then it is clear that the security is not increased by keeping the database secret. This is because the adversary still has to repeatedly guess from the 9 choices presented in each of the 5 challenges. This still leaves a total of 9^5 total possibilities. This is the same compared to when the database was public. If we do not make this simplifying assumption, then unlike before, the adversary does not know with certainty the number of different faces that they could be presented. Before, when the adversary knew that there were a total of 100 faces, they could determine the total number of unique passwords that would act as the upper bound of the number of guesses that was required. If they did not know there are only 100 faces, then they could assume that the total number of possible faces was much greater. This could therefore enhance security in the sense that it could prevent adversaries from attempting to access the system in the first place because there is less useful information that they could exploit. If the adversary does indeed choose to attack the system, then it is clear that hiding the database information does not change the number of guesses required to break into the system.

4. For this question, we consider the probability that the adversary succeeds in logging in if the system allows for one incorrectly answered challenge. That is, the user may either get all challenges right with a probability of (1/9)^5, or get either the first wrong, the second wrong, the third wrong, the fourth wrong, or the fifth wrong. There are 5 cases where the adversary

could get one of the challenges incorrectly and still obtain access to the system. The probability that the adversary guess 4 correct but 1 wrong is (1/9)^4*(8/9). Thus, the probability that the adversary can log in given that 1 challenge could be answered incorrectly is (1/9)^5+5*(1/9)^4*(8/9) = 41/59049, which is around 0.0006943. This could also be arrived at by considering this a binomial distribution, where the probability of success for each of the five trials is 1/9. We want at least 4 of the 5 trials to result in success. This is 41/59059, thus giving the same probability that the adversary succeeds in logging in. Another way of obtaining this would be to find that there are 9^5 total possibilities. One of these possibilities is that the attacker gets them all correct. 8 of these possibilities is that the attacker gets the first four correct, and the last one wrong (last one could be any of 8 that are wrong). We multiply this by 5, since either the first, second, third, fourth, or fifth challenge are answered incorrectly. Thus, the probability is 41/59049.

5. Shoulder surfing is among the most recognizable and obvious attack that is more effective in Passfaces compared to traditional text based passwords. This is because the entering of the passwords requires the user to select the face while it is shown on the screen. This could be easily seen by onlookers, since the images could be recorded or observed when selected by the user during login. Furthermore, the images are larger discrete units that may be more easily identifiable. When entering text, it may be more difficult for an onlooker to track the keys pressed, especially when the password entered is displayed as dots on the screen. Similarly, malware that captures screen output would be more effective against Passfaces than against traditional password systems that do not display the password entered on the screen. If a keyboard is used for Passface entry, this is less of an issue, but still susceptible to screen scraping software along with keystroke logging. Since the system must know the images that belong to the user in order to display it, this information must be stored in a way so that the original form is available to the system. Thus, it may be available to anyone who has access to the stored information. An attacker with these files may gain access to the user-specific images. Alternatively, social engineering attacks could ask the users to take screenshots of the Passfaces for sharing. Since this is a relatively new technology, users may be more susceptible and willing to provide this information compared to a text password to an entity posing as an authority figure.

2.

1. For the first method, we need to find the number of possible passwords. This is equal to 100 choose 20 = 535983370403809682970, since out of a total of 100 different pictures, 20 pictures in no particular order are chosen. Thus, the probability that the adversary guess the password is 1/535983370403809682970, which is around 1.866*10^(-21). The probability of impersonating the user by correctly responding to the set of 20 challenges presented by the system can be found by first considering the probability that the user succeeds in answering one challenge. The probability of correctly answering one challenge in 1/2 = 0.5. Thus, for 20 challenges, the probability of correctly answering all of them is (1/2)^(20) = 1/1048576, which is around 9.537*10^(-7). Since Alice determines that the level of security is the highest success chance (for accessing the system), the level of security is the one obtained from Method 2. This is because there is a higher probability of success in impersonating the user than there is to find the password (because 9.537*10^(-7) < 1.866*10^(-21) by several orders of magnitude).

2. Initially, it seems more difficult to use this system than the Passface system. Aside from requiring the user to remember 4 times more pictures, it also requires the user to successfully pass 20 challenges as opposed to 5, so it potentially increases the time required to log in. One could argue however that selecting 1 out of 2 is simpler than 1 out of 9. At first glance, it appears that this system is more secure than the Passface system, because 2^20 >

9^5 and 100 choose 20 > 100 choose 5. Assuming that both systems lock after 3 invalid attempts, we consider the success chance of an adversary in an online attack. **The first case is when the challenges remain the same between attempts.** The success chance of breaking into this system is much less than the Passface system, since the attacker would have exhausted 3 out of 2^20 possibilities compared to 3 out of 9^5 possibilities. **The second case that we consider is when the challenges do not remain the same.** That is, the challenges change between attempts. In this system, the success chance starts out lower than the Passface system on the first attempt, but likely increases drastically after each attempt due to the reasoning explained in Part 3 of Question 2 (Next Part). The Passface system on the other hand, starts out at a higher success chance, but the probability of success does not increase as much with each successive attempt. Assuming that around 50% of the pictures in this system can be eliminated as possibilities after one attempt (see reasoning in Part 3), this leaves 2^10 = 1024 possibilities, which is already less than 9^4 = 6561 possibilities, assuming somewhat arbitrarily that after one attempt on Passfaces, one of the challenges is known. Since greater security corresponds to a lower probability of success, the security of Passfaces can be considered higher when the challenges change, and the security of Passfaces is lower when the challenges do remain the same. In terms of usability, Passfaces seems to take less time to remember (since only 5 are required) compared to 20 pictures (even though we do get to select the 20 pictures to remember). Given enough familiarity with both password systems however, it seems that usability is around the same. This is because while one has to pick the correct face out of 9 for a total of 5 challenges in Passfaces, this is balanced by the need to pick the correct picture out of 2 for a total of 20 challenges.

3. The upper bound on the number of guesses required to login by fully learning the user's password is 2^20 = 1048576 guesses. This occurs when the system gives the same set of 20 challenges after each time the attempt was unsuccessful. The adversary therefore has to go through answering the exact same way, but each time making one change from what was done previously so as to cover all 1048576 possibilities. That is, in the worst case, the adversary needs to list all 1048576 possibilities and cross them off as incorrect one by one as they try them. Otherwise, we assume that the system changes the challenges each time, so even though the 20 pictures remain in the challenges in a mixed up order, the wrong picture (from 80 incorrect pictures) that is randomly paired with each is free to change. In this case, it is relatively simple for an adversary to gain access to the specific account. They simply need to record the pairs that they first enter, giving them a total of 20 pairs of 40 faces in total. Assuming that they failed the first time, the second attempt would involve going through and selecting only among these 40 pictures. That is, any pictures that did not previously show up on first trial is wrong. Thus, for each challenge, if one picture was previously shown and the other was not, then the picture that was shown is definitely correct. The picture that was originally paired with the correct picture is therefore wrong. Furthermore, if they see any pair where both pictures were previously shown but not together in the same pair, they can still deduce that only one of them is correct. Thus, if they do not encounter the missing half belonging to either of the two original pairs, they know immediately that the picture with the missing half is the correct one, so the other is false, and the missing half of the other is true. Unless the system does not change the pictures up and only swaps between them, it should be possible to fully learn the user's password in only a few tries. Assuming that the user can eliminate around 50% of the remaining pictures after each attempt, this will take less than 10 attempts to fully learn the user's password.

3.

1. Note the a XOR b = b XOR a. Thus, since z = y XOR s, we can write this as z = s XOR y. Applying a XOR to both sides, we obtain z XOR y = s XOR y XOR y = s. Since z = 0x1100dd0d and y = 3344ffac, we calculate 0x1100dd0d XOR 3344ffac = 224422a1. This is accomplished by first converting to binary, then performing bitwise XOR. Thus, key s can be retrieved to impersonate the tag.

2. No, a passive eavesdropper cannot learn the secret keys from observing a single execution of the protocol. This is because after a single execution, the eavesdropper knows y and z, where y = x XOR s1 and z = x XOR s2. There are many possibilities for x, s1, and s2. For instance, consider y = 10 and z = 11. Then we can have x = 10, s1 = 00, s2 = 01 as a possibility, but we can also have x = 00, s1 = 10, s2 = 11. Thus, if the goal of the passive eavesdropper is to identify s1 and s2, they cannot do so from observing a single execution of the protocol that gives them y and z.

3. No. They cannot learn the secret key. We note this is the case since with each execution of the protocol, the attacker only obtains access to y and z. Thus, after n elections of the protocol, the attacker has y1, y2, …, yn and z1, z2, …, zn. This grants them x1 XOR s1, x2 XOR s1, …, xn XOR s1 along with x1 XOR s2, x2 XOR s2, …, xn XOR s2. With any combination of these, there is no way to isolate just one variable. XOR can be used to determine a single value when one of the operands is known (as in the first part), but in the case where we have two unknowns each, XOR can only remove the common operand, so we are still left with the XOR of the two other unknowns. For instance, (x1 XOR s1) XOR (x2 XOR s1) = x1 XOR x2. We note however, that although the attacker cannot determine the secret keys s1 or s2, they can still impersonate the tag. From just one execution, the attacker can obtain

(x1 XOR s1) XOR (x1 XOR s2) = (s1 XOR x1) XOR (x1 XOR s2)
= s1 XOR (x1 XOR x1) XOR s2
= s1 XOR s2

That is, while the attacker does not have the keys individually, they do have the XOR of the keys together. Thus, for the next execution when the reader sends y2 = x2 XOR s1, we calculate

y2 XOR (s1 XOR s2) = (x2 XOR s1) XOR (s1 XOR s2)
= x2 XOR (s1 XOR s1) XOR s2
= x2 XOR s2
= z2

Thus, we can still respond with z2, successfully impersonating the tag even though we do not know what the secret keys s1 and s2 are individually.

4. In all of the following questions for Part 4, we assume that the attacker is able to intercept the random challenge r from the reader, and also the hashed result and Id from the tag. That is, the secret key is the only thing that is unknown. Additionally, we assume that the attacker is knowledgeable of the hash function h(). In the following responses, "//" refers to concatenation. For the item to be authenticated (so that there is no suspicion of tampering), the reader must obtain from the tag an Id and a hashed result. It computes the hashed result separately using the Id obtained from the tag and compares this to the hashed result sent by the tag. Only when these two hashed results match is the tag accepted. The goal of the adversary is therefore to be able to trick the reader into believing that the tag has not been tampered with.

1. Yes the attack will work. In Attack 1, the adversary tampers with the tags' responses during the scanning round. During the scanning round, the reader broadcasts a challenge so

that all nearby tags receive the same challenge r. Thus, the attacker will have access to r from the Reader, as well as h(r//Id_1), Id_1 for Tag_Id1, h(r//Id_2), id_2 for Tag_Id2, etc. That is, all the attacker needs is a way to somehow store this data temporarily before sending out the false responses. Since the goal of the attacker is simply to corrupt the shop's database, they could make Tag_Id1 respond with h(r//Id_2), Id_2 for example. This is possible because we assume that the attacker was able to intercept all of the responses from the different tags. Minimally, we only need one other Id, since this would allow this tag to respond with the Id of the other tag, as well as a hashed result that uses this Id instead of the correct one. Thus, since the challenge is the same, the attacker is free to swap the responses of each tag as they please. In this case, the reader will then verify h(r//Id_2), Id_2 and be convinced that Tag_Id1 is Tag_Id2. By repeating this procedure, the attacker could corrupt the shop's database by altering the Id associated with each tag.

2. Assuming that the attacker has access to some Id_2 associated with a product that costs less than the item, then it is possible for this attack to work. This Id_2 could be obtained previously. For example, during the scanning phase. This is a resource that is required before the attack is executed, since there is no way to obtain another Id during the checkout phase. The reader sends the challenge r to the tag. However, since we do not want the tag to respond correctly, we intercept the challenge and instead use our knowledge of the Id_2 of the other cheaper product. Since we know the hash function, the challenge r, and the Id_2 of the other item, we can make the tag send h(r//Id_2), Id_2 as opposed to sending the correct h(r//Id_1), Id_1. This attack therefore supposes that the attacker already has access to the Id of a cheaper item. Note that knowledge of the hashed result from a previous challenge will be of no use, since the challenge r was different. We cannot simply use h(r0//Id_2) the hashed result from a previous challenge since this will not be the same as h(r//Id_2) from the current challenge. When the challenge r is sent, the attacker intercepts this r and makes the tag return the computed hashed value using Id_2 instead.

3. Yes, the attack will work. Similar to Attack 1 on Protocol 1, the attacker gains access to the challenge r, along with the responses from many tags. For instance, the attacker would obtain h(Id_1//k1) XOR r, Id_1 from Tag_Id1, h(Id_2//k2) XOR r, Id_2 from Tag_Id2, etc. In contrast to Protocol 1, we do not know all of the individual components that are sent to the hash function, since we do not know the unique secret keys k1 and k2. Thus, it is not possible to trick the reader without the Id and the hashed result of another item. If we only have the Id of another item, we cannot compute h(Id_2//k2) XOR r, even through we have Id_2 and r. Thus, for Attack 1 on Protocol 2 to succeed, the user would swap the output of Tag_Id1. For instance, by making it return h(Id_2//k2) XOR r, Id_2 (We have access to this since we record the responses of various tags when the reader sends out the challenge), the attacker could trick the reader into believing that Tag_Id1 is Tag_Id2. One would therefore minimally require another set of h(Id//k), Id. This is because one needs the Id along with the corresponding h(Id//k). One can then perform the XOR with the current challenge r. Note that h(Id//k) could be obtained by applying XOR r to the response by the tag that provides h(Id//k) XOR r, so they could save this to use for future attacks. This would then allow one to repeatedly corrupt the database over different scans, since the attacker would only need to replace the Id with the fake Id, and replace the hashed result with h(Id//k) and a XOR r for the current challenge. Unlike Attack 1 on Protocol 1, this attack requires that the attacker obtain the responses from multiple items in order to swap the values. For Protocol 1, the attacker only required another Id which they could pair up with the challenge r in the hash function to trick the reader.

4. No. Assuming that the user has another Id of a cheaper item but not the corresponding h(Id//k), it is not possible for Attack 2 to work on Protocol 2. This is because at checkout, we are not given the hashed result of any other item. Since we do not know the secret

key associated with the cheaper item, we cannot compute the corresponding hash value to return to the reader to trick it into believing that the item is the cheaper item. Only if the attacker had obtained beforehand the Id of a cheaper item along with h(Id//k) could the attack work. This is the only way for the attacker to trick the reader, since they could make the tag return this cheaper Id, along with the XOR of h(Id//k) with the current challenge r. Even if the attacker had access to h(Id//k) XOR r0 for a previous challenge r0, they could not obtain the result for h(Id//k) XOR r, since there is no way to remove the XOR with r0 and replace it with r (unless they had stored r0 as well). This of course, means that the attacker would need some way to remove the XOR with r0. This could be accomplished by applying XOR r0 to h(Id//k) XOR r0 during a previous challenge and response to isolate h(Id//k) for future use. Thus, if the attacker had stored this information when attempting Attack 1 on Protocol 2, they could use this to accomplish Attack 2 on Protocol 2. Without both Id and h(Id//k) for a cheaper item, it is not possible to perform Attack 2 on Protocol 2.

5.

1. No. According to http://www.cic.gc.ca/english/passport/help/epassport.asp, the Canadian ePassport does not currently store biometric information such as an iris scan or fingerprints. The only biometric information that is stored in the Canadian ePassport is the photo of the passport holder's face. However, this picture along with identifying information is already contained on page 2 of the passport.

2. While biometrics is emerging as a powerful form of authentication, there are also many security and privacy issues related to biometrics. In terms of security, increased security often leads to decreased convenience, and vice versa. For instance, since a similarity (match) score may be required to authenticate a user, there has to be a threshold above which the user is accepted. If this threshold is too high, it may be inconvenient for the user since it may take more time to be verified. If the threshold is too low however, the security of the system may be at risk. False match and false non-match are some issues related to the security of biometrics. Another security issue concerns the fact that the biometrics cannot be easily changed. If someone obtained a fingerprint to pretend to be a legitimate user, even if the user found out, it would be tedious, time consuming, and potentially costly to change the information. Some privacy concerns include the covert collection of personal information such as face scans and fingerprints without informing the user of this data collection. Additionally, secondary information revealing a person's health from iris scans, employability based on their fingerprint and other information would violate the privacy of the individual. Another privacy concern would be cross matching, since biometric data could be collected for one purpose and latter used for another purpose altogether without the consent of the user. In terms of security, biometrics may be hindered by duplicates or prosthetics copies of the user's biometric information. For instance, obtaining the user's fingerprint may be enough to pose as the user. This acts as a serious security issue. Another security issue relates to the leakage of biometric data. This compromises the security of the system that was breached, as well as other external biometric systems as well. That is, if the attacker can obtain the biometric information of the user, they could potentially attack multiple systems that use this data for authentication.

3. The RFID chip used in the passport is the same as those used in retail in the sense that they operate using the same fundamental principles. That is, they both use RFID chips that use electromagnetic fields to identify and track tags that are attached to objects. For ePassports, there are standards that the RFID passports adhere to, as determined by the International Civil Aviation Organization (ICAO). The Canadian ePassport operates at a frequency of 13.56 MHz. To read the information, the ePassport reader must be held within 10 centimeters of the reader. ePassports may have additional security features, such as a metal

lining to prevent unauthorized readers from scanning the information. For added security, passive authentication is used to prevent tampering by requiring a data signature be verified, basic access control is used to prevent unauthorized scanning by requiring a key be entered into the reader beforehand, and active authentication is used to prevent cloning. In retail, RFID acts as a way to identify items without manual entry, and is used to tag items. The RFID chips in retail may contain anti-theft technology such that exit-scanners can detect when an active RFID passes through the entrance. An alarm could then be set off, with the chip relaying information such as what the stolen product is.

4. It is important that the RFID chip in the ePassport can be read in Canada and other countries. To accomplish this, the chip has been rigorously tested to ensure that it meets various international standards. Among them is ICAO 9303, which is the ICO technical standard for machine-readable passports. Thus, the cryptography does conform to the ICAO standard, since Doc 9303 http://www.icao.int/publications/Documents/9303_p11_cons_en.pdf reveals that many standards are required, such as those pertaining to the authentication of data, secure messaging, key agreement algorithms, and implementation. The passive authentication process also uses ePassport certificate distribution through the ICAO Public Key Directory (PKD) service. In Canada, the document signer certificates and revocation lists are shared through this secure directory.

5. In 2008, it was shown by a security researcher at Amsterdam university that biometric passport chips could be cloned in under an hour. The “attacker” was Jeroen van Beek, a researcher who was asked by the Times to conduct tests. van Beek proceeded to expose security flaws in the British e-passports by cloning the chips and implanting digital images of Osama bin Laden and a suicide bomber. The altered chips were recognized as genuine by United Nation’s (the victim’s) software. The attack demonstrated that fake biometrics could be inserted into a fake or blank passport. van Been claimed that he had used his own software, a £40 card reader and two £10 radio frequency chips to clone and alter the chips so that they could be planted in fake or stolen passports. The intent of the demonstration was not to claim that terrorists would be able to perform this on passports, but to simply raise the issue related to security on ePassports. Furthermore, it was performed to reveal to the public that, contrary to the claim that the British biometric passport was "one of the most secure passports available”, there are indeed security flaws that could be exploited. This “attack” was revealed by the “attacker” as opposed to being detected or contained. There was no ill will, or a desire to exploit the system, since this was never the intention. van Beek simply demonstrated that it was possible to exploit the lack of security on British e-passports, with the consequence of this demonstration being greater awareness of the issue and the introduction of additional security measures to protect the integrity of the data stored on the e-passports. This incident was reported on https://www.theguardian.com/technology/2008/aug/06/news.terrorism by James Meikle on August 6, 2008. Some indicators that this source is credible include the fact that the author is willing to stand behind the claims made, a date is presented so readers could cross reference the events discussed, and was published by a well known newspaper company.