# **COMS3000: 2014 exam answer**

**Q1) [2 marks]** Compute the Annualised Loss Expectancy (ALE) for the following scenario. It is estimated that on average, a user account in a company is compromised once every 5 months. It is further estimated that the total loss to the company due to each such event is $9,000.

I agree with the person in Red, ARO = 12/5, SLE = 9000, ALE = ARO \* SLE = 12/5 \* 9000 = 21600

**Q2) [2 marks]** Which of the 5 aspects of information security is compromised by a phishing email? Explain your answer.

The phisher is not Authentic. They are trying to get your sensitive information by by masquerading as a trustworthy entity in an [electronic communication](https://en.wikipedia.org/wiki/Electronic_communication).

Could this also cover Confidentiality? They are stealing a legitimate user’s information

**Q3) [2 marks]** Explain the concept of a “Trusted Path” in information security and give an example.

I don’t remember seeing this in lectures but WikiPedia’s definition is “a mechanism that provides confidence to the user that they are communicating with whom they intended to communicate with and ensures attackers can’t intercept or modify the information being communicated”

Slide 21 in Lecture 2 covers the example of a bogus login interface on Windows, but CTRL-ALT-DELETE is a system level interrupt that can’t be hijacked, so you’re guaranteed to get a real login prompt (a trusted path).

An example could be an SSL Certificate, verifies that the user is communicating with who they want to over a secure connection.

And you are trusting the CA

**Q4) [7 marks]** For this question, you can assume that the function h() is an ideal cryptographic oneway hash function which produces 128-bit outputs. (You can assume the ‘random oracle model’.)

**a) [2 marks]** For two different, randomly selected 2048-bit inputs x1 and x2, what is the probability of a collision, i.e. what is the probability that h(x1) = h(x2)?

0.5^128

**b) [2 marks]** Let’s assume the same function h() is used for Bitcoin mining, where the goal is to find an input x so that the first n bits of h(x) are equal to ‘0’. Let’s assume you just bought commercial Bitcoin mining hardware which can do 1TH/s, i.e. 1,000,000,000,000 hashes per second. For n=60, and using this hardware, how long does it take on average to be successful, i.e. to find a value x so that the first n bits of h(x) are equal to ‘0’?

On average we have to try 2^n inputs

n = 60

time in seconds = average hashes required/hashes per second

= 2^60 / 1,000,000,000,000

= 1,155,431.26

= 19257 min

= 320 hours

= 13.37 days

= 1,152,921.505s

= 19215.36min

=320.25hr

=13.34 days

**c)** [3 marks] Is the problem in Question b) related to the Birthday Paradox? If yes, explain how. If not, explain why not and what the difference is.

The Birthday Paradox relates to strong collision, where you’re trying to find two matching inputs. However, part b is looking for a match to an input, and therefore is based around weak collision. As such, part b is not related to the birthday paradox since it is looking for weak collision.

**Q5) [2 marks]** Explain the difference between an online password attack versus an offline attack. Give an example for each type.

An offline attack requires no communication with the server under attack (holding the key).

Example: you’ve got a downloaded database dump and can throw as much power at the attack as you can afford.

An online attack requires (typically significant) work of the system under attack.

Example: trying to guess someone’s PIN at an ATM. You only get 4 tries before you lose the card (so it doesn’t matter how fast you can try a PIN combination).

**Q6) [2 marks]** Explain the concept of SALT in the context of passwords. What are the aims of SALT?

In [cryptography](https://en.wikipedia.org/wiki/Cryptography), a **salt** is [random](https://en.wikipedia.org/wiki/Random_Number_Generator) data that is used as an additional input to a [one-way function](https://en.wikipedia.org/wiki/One-way_function) that [hashes](https://en.wikipedia.org/wiki/Hash_function) a [password](https://en.wikipedia.org/wiki/Password) or [passphrase](https://en.wikipedia.org/wiki/Passphrase). The primary function of salts is to defend against [dictionary attacks](https://en.wikipedia.org/wiki/Dictionary_attacks) versus a list of password hashes and against pre-computed [rainbow table](https://en.wikipedia.org/wiki/Rainbow_table) attacks.

Also prevents two users with the same password from having the same password hash.

**Q7) [2 marks]** Assuming an attacker has access to a hashed password file with only a single hashed password. Does SALT improve the security in this case? Explain your answer.

Yes?

I’m thinking no. The main advantage of salting is that the hash function between separate passwords then becomes different, so an entire database can’t be compromised, and multiple passwords can’t be received. When there’s only one password, neither of these are relevant.  
So in this case it essentially make no difference?

This would still provide protection against dictionary attacks. If you didn’t use a salt, then you could just compare the password against precomputed hashes. By having the salt, this won’t work.

Therefore, in summary, yes if the attacker plans on using a dictionary attack or rainbow table but if the attack was brute force, it would make no difference. Consequently, the security is improved, by using SALT.

**Q8) [2 marks]** Explain the concept and goal of multi-factor authentication. Describe a practical example.

Multi-factor allows the system to utilize two guards against an attacker. THis is much stronger. An example would be to require a password and a time sensitive code that is messaged to a mobile phone (something you know, pword, and something you have, mobile phone). A third factor could be a biometric (something you are).

Multi-factor authentication presents several separate authentication stages,

-Something you know

-Something you have

-Something you are/do

**Q9) [2 marks]** You are given a ciphertext that you know has been encrypted using a Vigenère cipher. You observe that the word ‘SVTYR’ is repeated 4 times, and appears at positions 23, 41, 95, and 131. What is the most likely codeword length?

Appears 4 times 23, 41, 95, 131  
Distance between word appearances 41-23 = 18 95-41 = 36 131-95 = 36

Gcd(54, 36, 18) = 18

therefore most likely code length will be 18

**Q10) [3 marks]** Consider the challenge-response authentication protocol as shown below. Is this protocol secure? If not, what is the problem and how can it be fixed?

Note:

· *h*() is a secure, 256-bit cryptographic one-way hash function.

· ‘||’means concatenation

Protocol:

1. Client sends *username* to Server

2. Server computes challenge *c* as follows: *c = h(username)*

3. Server sends challenge *c* to Client

4. Client computes response *r = h(password || c)*

5. Client sends *r* to Server

6. Server verifies *r*

Challenge should be random, salt that?

Is this vulnerable to a replay attack?

Yes replay attack, the challenge code can be re-used. If an eavesdropper listens in and retrieves the username, they can then send the request to the server gaining the challenge code. Having this information, they can identify the response sent from the client.

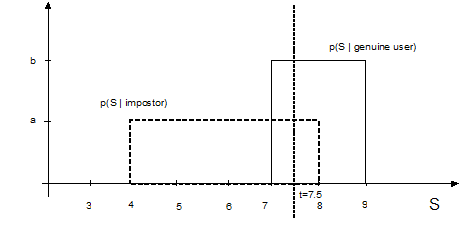
The challenge is always the same for the same user. Therefore, a replay attack can be possible without knowing the raw username or password.

The tutorial answer:

The problem is that c is the same for each login attempt. An attacker can easily launch a replay attack. The eavesdropper can listen and hear the username, challenge and reply.

The nonce, or challenge should not be reused.

**Q11) [5 marks]** Consider a biometric system with the following (somewhat unrealistic) conditional probability density functions for the matching score *S* for an impostor and a genuine user.



**a)** [2 Marks] For a threshold of *t* = 7.5, what is the False Non Match Rate (FNMR)?

a x 4 = 1 -> a = 0.25

b x 2 = 1 -> b = 0.5

FMR = 0.5 x a = 0.125 = 12%

FNMR = 0.5 x b = 0.25 = 25%

**b)** [3 Marks] What is the Crossover Error Rate of the system, and what is the corresponding value of the threshold *t* ?

The error rate at which FMR = FNMR (solve for t)

FNMR = ( t - 7) \* b

FMR = (8 - t) \* a

(8 - t) \* 0.25 = (t - 7) \* 0.5

2 - 0.25t = 0.5t - 3.5

0.75t = 5.5

t = 7.333

Subbing t back into one of the functions,

FMR = (8 - t) \* a

FMR = (8-7.333)\*0.25 = 0.16675

we get CER ~= 16.7%

**Q12) [2 marks]** In a Linux system, based on the information provided below, what are the access rights to the file ‘script’ for the user ‘alice’, who is a member of the group ‘student’.

Also, explain what the letter ‘s’ in the fourth position means.

-rwsr-xr-- bill student script

Alice has access to read and execute.  
s instead of x in the owner permissions means that the 'sticky bit' (suid) is enabled, so this file will be executed with root permissions by all users.

^ Pretty confident the above explanation is wrong. Having the SUID bit set like this means that when the file is executed, the file owner’s permissions/UID will be used to execute it, rather than the permissions/UID of the current user. It would only execute as root, if root was the owner. But in this case the owner of script is bill, so it would execute with his permissions/UID. +1

So technically in this case, alice can also write? So she can do all read, execute and write since she’s running with the owner’s(Bill) permission?

No, the SUID bit means when the file is `executed` it will run with the same permissions as the file owner `bill` in this case (whereas traditionally, if you execute a file as `alice`, it will run with your user `alice` permissions). This means if the script tries to access any other files, it will have the full rights of `bill` in this case when it accesses them. It does not grant Alice those rights, merely the code running within the file those rights.

The only time it will grant root privelege is if the SUID is for 0, or the root id. This means the file will run with root permissions, this is used to change your password for example, as it requires accessing the shadow file (root permission required), but allows you to do it from your account (as the change password script runs with setuid root). This does not grant you access to the shadow file (other users passwords), merely grants the script access. And as you cannot modify the change password script (no write access), it is secure.

**Q13) [6 marks]** Consider a system where user passwords are automatically and uniformly randomly generated, i.e. all possible passwords are equally likely. The passwords consist of the 26 letters of the alphabet, and are all 8 characters long.

**a)** [2 marks] Compute the password Entropy H(X).

H(X) = 26^8 \* -(26^-8 \* log(26^-8)) = ~12 bits

^ I don’t agree. Since the probabilities of all outcomes are equally likely H(X) = I(X).

H(X) = I(X) = log\_2(26^8) = 37.60 bits

-(26^8 \* (26^-8 log\_2(26^-8)) = 37.60 bits

I agree with green +1

**b)** [2 marks] Now consider an alternative approach to randomly generating passwords. In this case, pasdswords are uniformly randomly chosen from a list of all words (of any length) in the English language. As a very rough estimate, we assume the English language has 1 Million words. Does this approach provide better password security than the approach in question a) ? Support your answer using Information Theory.

Working out the entropy for this shows it’s less than that of the password with 8 characters, since only 1 thing is being chosen, and even when it’s chosen from a large number of choices, it still scales less than having more inputs.

Not sure… H(X) = log2(1000000) = 19.93 bits. Which is worse than 8 random characters.

Isn’t it as simple as noting that the number of possible passwords w/ 8 random chars is 26^8, which is more than the number of possible passwords w/ 1,000,000 words? (and if you’re supporting your answer using information theory you just take the entropy of both, and then quote the password entropy slide and say that a higher entropy means it’s more random/unpredictable/harder to guess)

Wouldn't this also make it more vulnerable to dictionary attack? I guess the question asked for information theory, so oh well.

Text taken from the english language does not have the same entropy per character. Because certain characters are more frequently used than others, and because certain trigraphs are used more frequently than others, choosing a character at random from ‘any’ english text will only give you ~1.5 bits of entropy (as certain characters are much more likely), whereas choosing a character at random from the alphabet as in a will give you 4.7 bits of entropy per character (as it’s truly 1/26 per character). That means that you have reduced your entropy from 37.6 to 12, so this is much less secure.

**c)** [2 marks] Aiming to increase the Entropy of the 8-character passwords as described in Question a), a system administrator has the following idea. He suggests adding 6 additional characters at the end of each 8 character password, increasing the length from 8 to 14 characters. These 6 additional characters are computed from the first 8 characters using a hash function and encoded using Base64 encoding. (Base64 encoding is simply a method to encode binary data as ASCII strings, but the details are not relevant for this question.) Discuss if this idea increases the Entropy of the passwords.

This does not affect the randomness of the password at all, since the additional characters are based off the input, and as such the input is the same size. Therefore, the entropy will remain the same.

“If a [compression](https://en.wikipedia.org/wiki/Data_compression) scheme is lossless—that is, you can always recover the entire original message by decompressing—then a compressed message has the same quantity of information as the original, but communicated in fewer characters.” — <https://en.wikipedia.org/wiki/Entropy_(information_theory)>

^ since we have a trivial lossless compression scheme for the passwords (we simply don’t need to store the 6 addition characters as they are a function of the other 8 characters). This would imply that the Entropy of the passwords have not changed.

Fundamentally, the main thing is that the base64 of a PW is always the same for that PW, so we still only have N = 26^8 and Pi = 1/26^8 (therefore entropy is the same)

**Q14) [2 marks]** You are given a binary ciphertext message C = 11001100 which has been encrypted with a one-time pad using the key K = 10101010. What is the corresponding plaintext message M?

Tutorial 6 has several problems like this, uses XOR for a one-time pad

C = 11001100

K = 10101010

M = C XOR K = 01100110

**Q 15) [2 marks]** What is the Discrete Logarithm of 6 to the base 5 if we are calculating modulo 7, i.e.

*log5 6 mod 7* = ?

log5(6) mod7 = x

log w (y) (mod z) = x

w^x (mod z) = y

5^x (mod 7) = 6

5^1 (mod 7) = 5

5^2 (mod 7) = 4 (5\*5 = 25 = 7\*3 + 4)

5^3 (mod 7) = 6 (5^3 = 5^2 \* 5^1 = 5 \* 4 = 20 = 7\*2 + 6)

Therefore, log\_5 6 (mod 7) = 3

**Q16) [2 marks]** Consider the RSA system with the following parameters: p =11 , q = 5 , e = 7  
Encrypt the following numerical plaintext message: m = 5

N = p\*q = 55

C = memod n = 57 mod 55 = (52mod 55 \* 52mod55 \* 52mod55 \* 51mod55) mod 55 = 25

**Q17) [3 marks]** What is the main purpose of Public Key Certificates? Explain how they are used in the context of TLS/SSL.

They are used for authentication?

Taken from lecture slide:

Purpose - links together identity and public key

Signed by a trusted CA (Certification Authority). Public Key certificates are used to authenticate the server.

**Q18) [2 marks]** Describe differences between an Intrusion Prevention System (IPS) and a Next Generation Intrusion Prevention System (Next Gen IPS).

This wasn’t covered in lectures, therefore not covered in exam?

**END OF EXAMINATION**