# **COMS3000 2016 Final Exam Answers**

## [**UQAttic**](http://uqattic.net)

## **Get more out of your study time. ~~Join UQAttic's revision chat.~~ UQCS’s Slack.**

#### [**Other exam papers**](https://docs.google.com/folder/d/0B6_D4T6LJ-uwYzY1YWMzNjYtMzUyZC00OTEyLWJlMjktOGExYWUwOTc4NDE3/edit)

### Please **contribute** to these documents.

### If you're looking for an effective way to familiarise yourself with the course material, you can't go past collaborating with fellow students. We have laboured to put these up, and so at the very least point out where you think we are wrong!

### You'll get more out of the course, you'll do better in the exam, and other students will benefit from your input as well.

### To get editing permissions, simply go to the [chatroom](http://uqattic.net) and provide us with your Google Account address.

### **Style.**

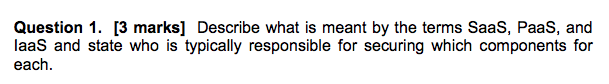
### Type answers in blue beneath each question.

### If you're unsure of your answer, highlight your answer text then hit Ctrl+Alt+M to create a comment beside the text. Once you're satisfied with the answer, click the "Resolve" button on the comment.

### If you want some extra explanation from someone else on their answer, highlight the other person's answer and repeat the procedure above.

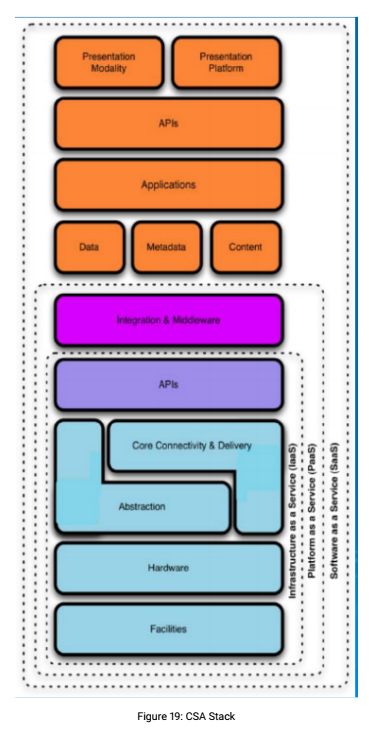
### **Communicate.**

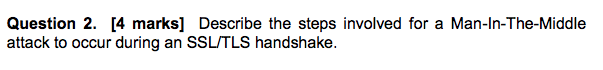
### Head over to [uqattic.net](http://uqattic.net/) and click "Chat Now!". You'll find a chatroom full of students just like you. Talk about a revision document (like this one) or swap prep tips. If you have your own IRC client, point it to irc.uqattic.net, port 6667, channel #attic.



SaaS, PaaS and IaaS are software as a service, platform as a service and infrastructure as a service respectively. They represent cloud service categories. For SaaS, the vendor provides all of the underlying components and the user just uses the software as provided. It is the most abstracted cloud service of the 3. With PaaS the general infrastructure is provided along with integration software and APIs that allow clients to build their own software and data on. Finally, with IaaS, only the infrastructure hardware and basic APIs are provided by the vendor and it is the client’s responsibility to provide both the platform integration functionality along with software applications as desired.

I think this answer needs to include “who is responsible for securing” the stated components - Lewis

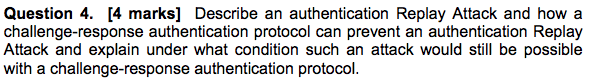
 The vendor is responsible for the parts inside the dotted lines. The client is responsible for anything on top.



A MITM attack on TLS depends on the client receiving the imposter’s public key and accompanying certificate instead of the response that would be sent by the legitimate server. If the client accepts this fraudulent certificate as genuine and sends the session key encrypted using the attacker’s public key, then the attacker will be able to decrypt that key and subsequent communications.



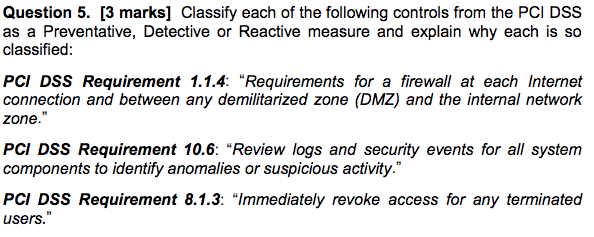
Signature based detection relies on detecting attacks using a stored database of known attacks and comparing possibly suspicious activity against it to detect these attacks based on their signature. Anomaly based detection looks for suspicious changes in behaviour of systems in order to detect an attack, such as a customer’s bank account being accessed from an overseas location immediately after a withdrawal has been made locally.



An authentication replay attack subverts communication by replaying hashed communications from the valid communicating parties. If Bob sends Alice a hashed password and Trudy is listening in, Trudy could potentially “replay” the hashed password to Alice and be granted access later. With challenge-response, the server challenges the client to verify themselves but includes a nonce to be hashed along with any verification communication such as a password. This imbues communication sessions with a unique/temporary quality that means replaying old hashes will not work.

A replay attack will still be possible if the nonce (number used only once) is reused. As an example, an eavesdropper observes the response to n=100 as h(n || p). The attacker will be able to gain access through a replay attack if at any point the server sends n=100 as a challenge to the eavesdropper. It does NOT matter if n is predictable, as long as the eavesdropper never sees n again (ie. it is never ever reused). Isn’t a nonce by definition only used once so this isn’t a problem? - Lewis

I think this considers when the protocol is not implemented correctly, there’s one example in the tutorial question so this might refer to that.



PCI DSS Requirement 1.1.4 (Firewalls) --> Preventative; packets from an unauthorised source are *prevented* from having access.

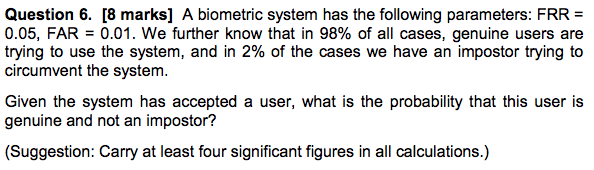
PCI DSS Requirement 10.6 (Review logs) --> Detective; looking at transpired events to *detect* whether anything bad such as in intruder gaining access has occurred.

PCI DSS Requirement 8.1.3 (Revoke access) --> Preventative/Reactive; update user privileges as they change to *prevent* any inappropriate access. Reactive is only after damage is done 8.1.3 does not imply that so i would say this is just preventative. - Lewis

(not sure about ^^^^ these ones)

I would agree with them all. 8.1.3 seems situational, might be good to get clarification from David. I would say it fits under the preventative definition from W1 slide 31: “prevent assets from being damaged”. It doesn’t fit reactive, since the sentence doesn’t indicate to only revoke AFTER the assets (cardholder data) have been compromised ie. to recover from that happening.

Check the 2010 exam paper answers, theres an example which was reactive (implement an incident response plan).



FRR = 0.05, FAR = 0.01

98% are genuine, 2% are imposters

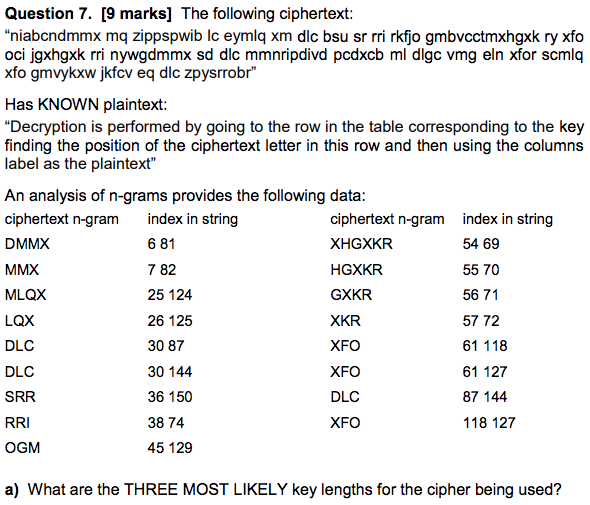
Correct Accept rate = 1 – 0.05 = 0.95

portion of % genuine users accepted = 0.95 \* 98 = 93.1

portion of % imposters accepted = 0.01 \* 2 = 0.02

P(accepted user is an imposter) = 0.02/(93.1+0.02) = 0.0002 or 0.02%

P(accepted user is genuine) = 93.1/(93.1+0.02) = 0.9998 or 99.98%



Top three most common n-grams are:

DLC, XFO and XKR (there are varying n-grams that end in XKR, e.g. XHGXKR )

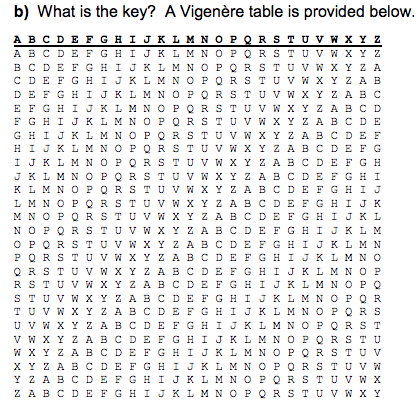
XKR has distances 69-54 = 15, 70-55 = 15, 71-56 = 15, 72-57 = 15 gcd = 15

DLC has distances 87-30 = 57, 144-30 = 114, 144-87 = 57 gcd = 57

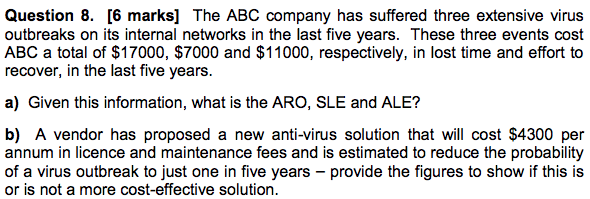
XFO has distances 118-61 = 57, 118-61 = 57, 127-118 = 9 gcd = 3

Most likely key lengths are 15, 57 and 3

I’m not sure this is correct. Wouldn’t the 3 most likely key lengths be the factors that appear most often across all of the n-grams? Does anyone else agree with this? I agree



Using the cipher text and cross-checking the table with the plaintext, the key is “KEY”



a)

ARO = 3/5 = 0.6

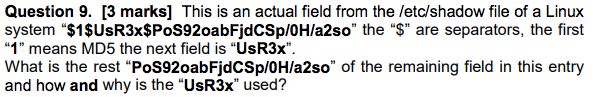
SLE = average cost of incidents = ($17,000 + $7,000 + $11,000)/3 = $11666.67

ALE = SLE \* ARO = 11666.67 \* 0.6 = $7,000

b)

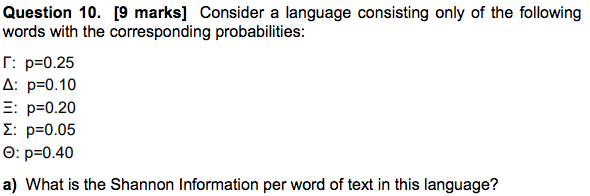
ALE with anti-virus solution = 11666.67 \* 0.2 + $4,300 = $6,633

The anti-virus solution appears to be more cost effective based on ALE alone.



“PoS92oabFjdCSp/0H/a2so” is the user’s MD5 hashed password. “UsR3x” is the salt. A salt is a piece of random data that is hashed along with the user’s password such that if the password of one user is compromised, other users with the same password are kept safe because their salts were different at the time of hashing.

“PoS92oabFjdCSp/0H/a2so” is the user’s MD5 hashed password concatenated with salt. The purpose of the salt is to protect against rainbow table attacks if the user’s password is common, not to protect other passwords. The salt is stored as plaintext anyway.

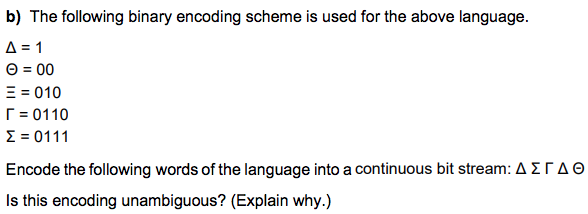


Γ: p=0.25 Δ: p=0.10 Ξ: p=0.20 Σ: p=0.05 Θ: p=0.40

H(x) = -(0.25\*log(0.25) + 0.10\*log(0.10) + 0.20\*log(0.20) + 0.05\*log(0.05) + 0.40\*log(0.40))

H(x) = -(-0.5 – 0.3322 – 0.464386 – 0.216096 – 0.52877)

H(x) = 2.04 bits



Δ Σ Γ Δ Θ encodes to 1 0111 0110 1 00

The coding is unambiguous because it is a prefix free code. This means that no word is a prefix of any other word meaning that a stream of bits can be decoded without ambiguity.



0.25\*4 + 0.1\*1 + 0.2\*3 + 0.05\*4 + 0.4\*2 = 2.7bits



2.7 – 2.04 = 0.66 bits



By using an insurance company, or forcing another party to undertake the full financial responsibility of an asset.

Blame the IT guy when millions of credit history data is stolen ;)



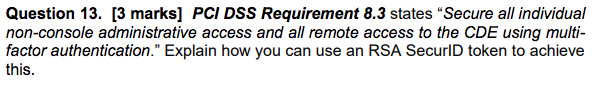
An attacker can send a deauthentication frame at any time to a wireless access point, with a spoofed address for the victim. This disconnects the victim from the access point at which time an alternate rogue access point mimicking the credentials of the legitimate may be connected to instead of connecting to the original legitimate network.

Organizations can install wireless intrusion prevention systems to monitor the radio spectrum for unauthorized access points.

To detect rogue access points, two conditions need to be tested:

1. whether the access point is in the managed access point list

2. whether it is connected to the secure network



RSA SecurID tokens can be used in conjunction with a regular password to constitute multi-factor authentication. At the time of login, the user has their password (something they know) and their personal token (something they have) to authenticate themselves as a valid user. The RSA token uses a random seed and timer such the digits displayed on the token at any given time are known by the login server and the sequence of numbers cannot be determined by the currently displayed number. This two-factor authentication strengthens security as any would be attacker needs to have both the password and the token to gain access.



An APT or advanced persistent threat is a set of stealthy and continuous computer hacking processes, often orchestrated by a person or persons targeting a specific entity. Stuxnet was an advanced suite of combined 0-day exploits combined specifically by the US government to run continued and covert interference on the Iranian nuclear program. By these merits, Stuxnet was an APT.