# **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.**

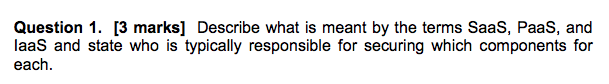
### Type answers in blue beneath each question.

### 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.

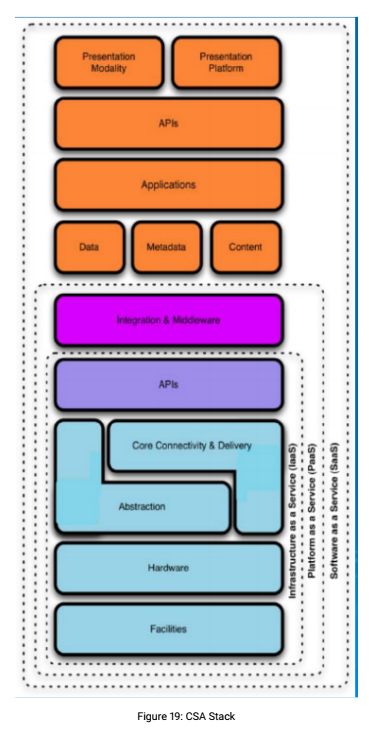
### **Communicate.**

### 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.

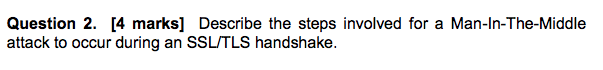
The vendor is responsible for the parts inside the dotted lines. The client is responsible for anything on top.



In SaaS, software as a service, the cloud customer has security responsibility. The responsibility falls fully under the cloud provider. The only control the customer has is through the contract with the provider.

In PaaS, platform as a service, the customer is provided hardware (real or virtual) and an operating system, the customer is responsible for securing the application layer, whilst the provider is responsible for securing everything else.

In IaaS, infrastructure as a service, the customer is provided hardware only. The customer is responsible for all security that does not encompass the physical security of the hardware (real or virtualised), which the provider is responsible for.

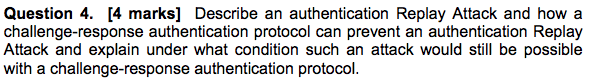


A MITM attack on TLS depends on the client receiving the imposter’s public key and accompanying certificate instead of the response that would be sent by the legitimate server. If the client accepts this fraudulent certificate as genuine and sends the session key encrypted using the attacker’s public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

1. The attacker inserts themselves in between the client and the server.
2. The attacker authenticates with the server.
3. The attacker presents a certificate (real or forged, may be signed or unsigned) to the client alongside a nonce.
4. The client will either accept or reject the certificate, if the client accepts the certificate, the attack is successful. The attacker is authenticated as the server to the client and to the server as a client. Note that in TLS/SSL the client does NOT have to authenticate itself, this is typically handled in the application layer, ie. username/password.



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

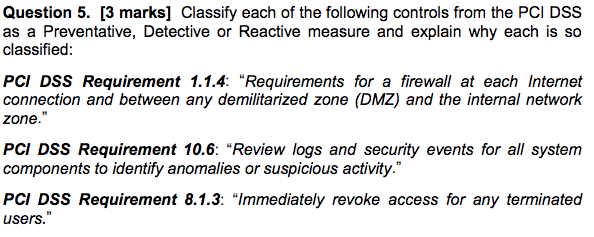
* The lecture slides explicitly mentioned that if the nonce is ever reused, the auth is broken. This is actually how the latest WPA2 attacks work, so you can’t assume a nonce is by definition only used once.

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.

I think the question is looking for nonce reuse, a nonce being reused counts as not being implemented correctly, this is how KRACK works.

A good example is the HTTP Basic vs Digest authentication pg 31 of week 3:

If basic HTTP authentication is used, a replay attack can be performed after eavesdropping a client sending their authentication details to a server. In HTTP digest authentication a challenge (nonce) is sent in the server reply to the GET request. The client authenticates themselves by sending the hash (digest) of their username, password, realm, nonce, URL) back to the server. The attacker can see the nonce that is used and the hash which is returned, due to the cryptographic one-way hash function, they cannot feasibly obtain the client password. A replay attack is possible if a nonce is reused as the attacker could replay a hash that uses that nonce. - Paul @zcoot on UQCS



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.

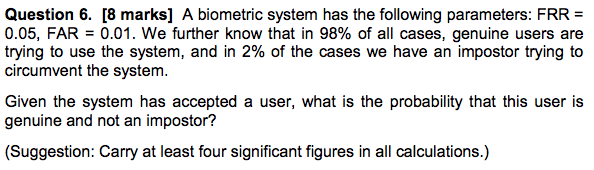
PCI DSS Requirement 8.1.3 (Revoke access) --> **Preventative**/Reactive; update user privileges as they change **to *prevent*** any inappropriate access. Reactive is only after damage is done 8.1.3 does not imply that so i would say this is just preventative. - Lewis

(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, there's an example which was reactive (implement an incident response plan).

As long as you have a reasonable justification you would still get the marks.



FRR = 0.05, FAR = 0.01

98% are genuine, 2% are imposters

Correct Accept rate = 1 – 0.05 = 0.95

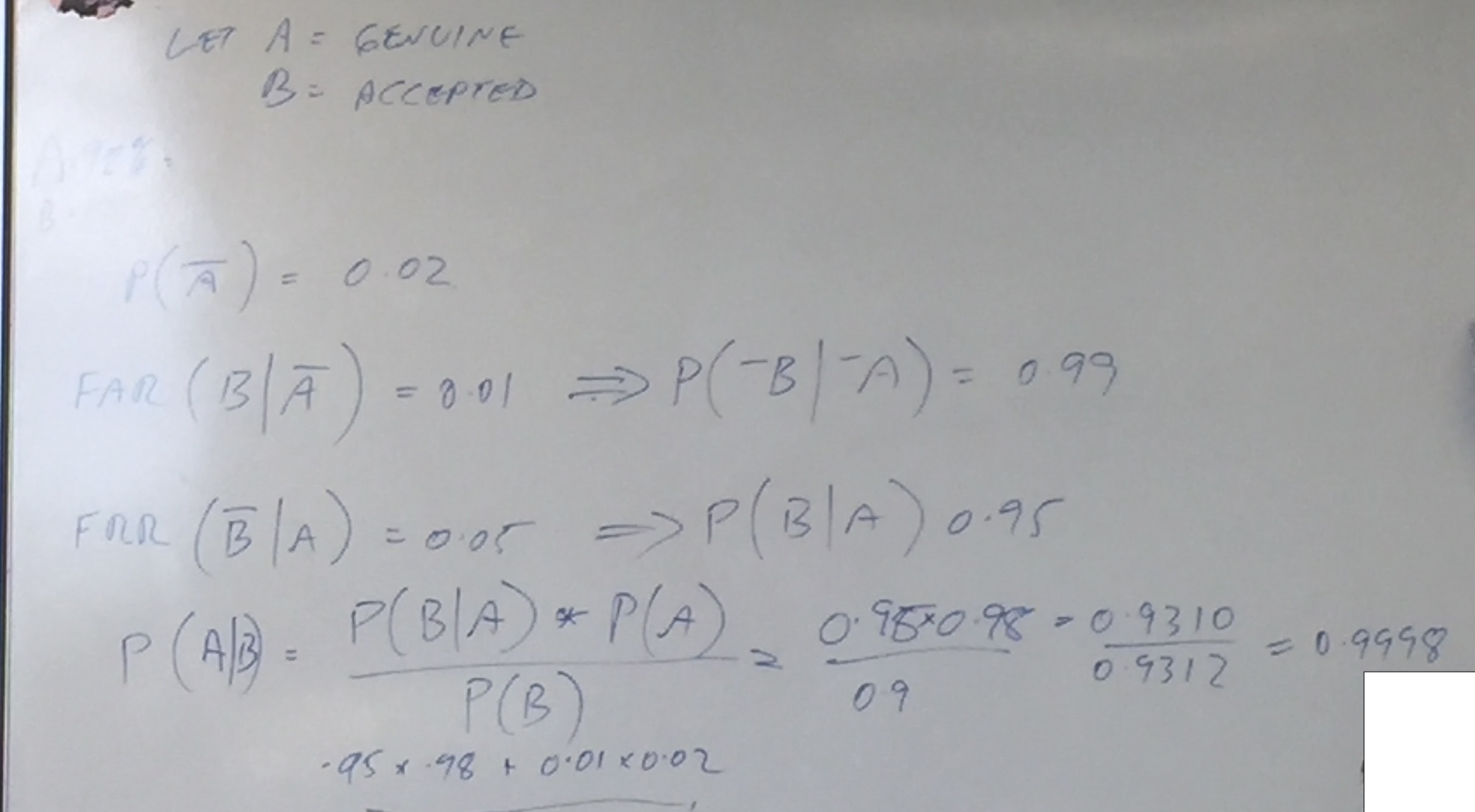
portion of % genuine users accepted = 0.95 \* 0.98 = 0.931

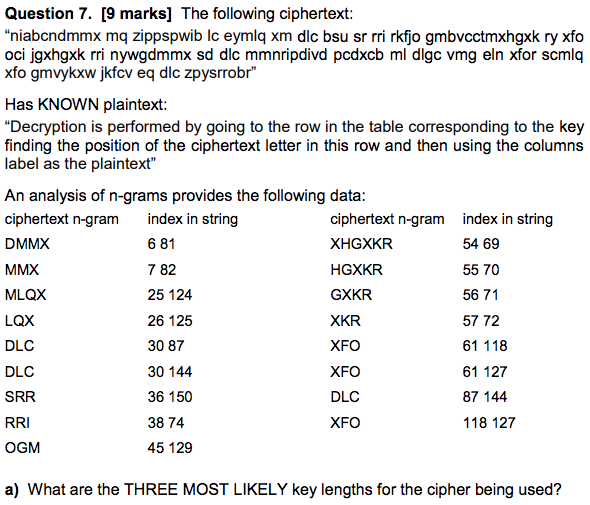
portion of % imposters accepted = 0.01 \* 0.02 = 0.0002

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

P(genuine | accepted) = 0.931/(0.931+0.002) = 0.9998 or 99.98 %

Lecturer’s working in revision session:





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

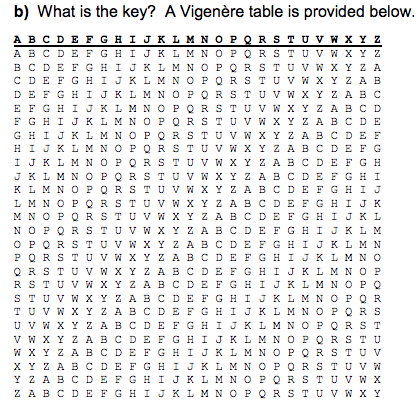
XFO has distances 118-61 = 57, 118-61 = 57, 127-118 = 9 gcd = 3

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

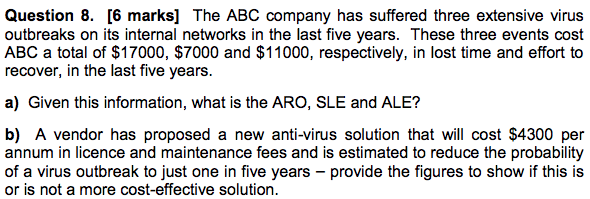
I got 3, 5, 15, as did csa

I (different person) got 3, 5 and 15 as well. Realistically I think you would get some most of the marks from just calculating the distances and seeing some pattern.



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

You can find this by working with the the encrypted dlc (the in decrypted) and then decrypting some text directly after it.



a)

ARO = 3/5 = 0.6

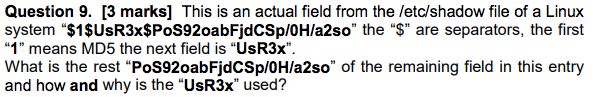
SLE = average cost of incidents = ($17,000 + $7,000 + $11,000)/3 = $11666.67

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

b)

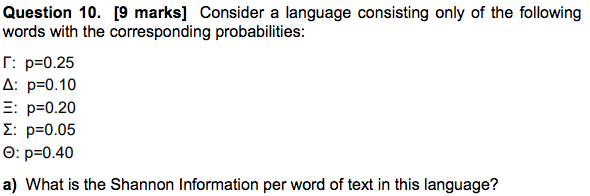
ALE with anti-virus solution = 11666.67 \* 0.2 + $4,300 = $6,633 (the ABC saves 7000-6633 = $366.66 / month by buying AV

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 (its concatenated 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.



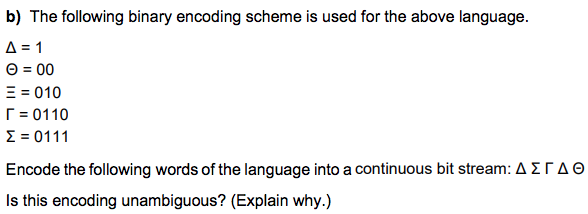
Γ: p=0.25 Δ: p=0.10 Ξ: p=0.20 Σ: p=0.05 Θ: p=0.40

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)

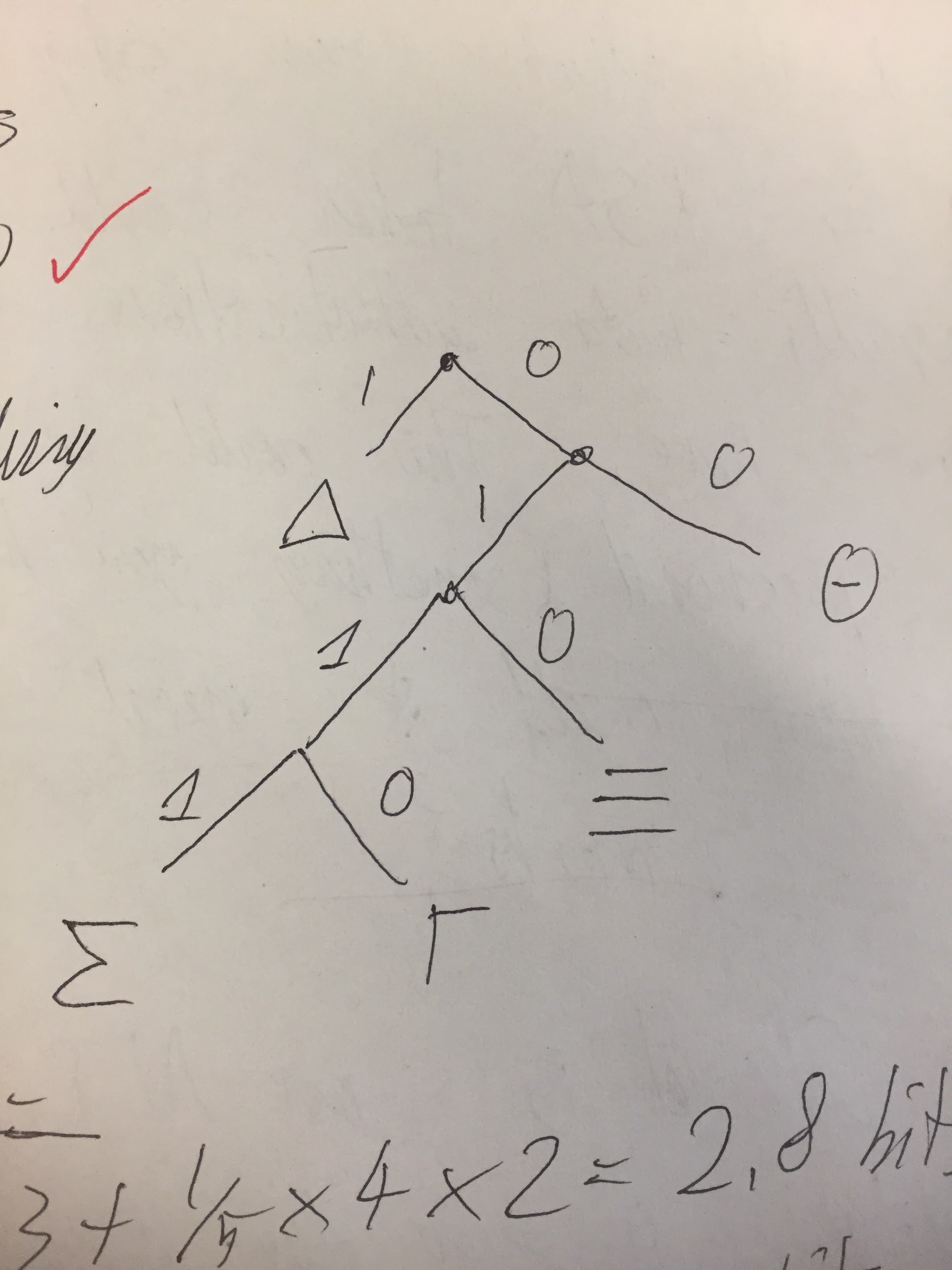
H(x) = 2.04 bits

|  |  |
| --- | --- |
| Word | Information (bits), from |
|  | 0.5 |
|  | 0.332 |
|  | 0.464 |
|  | 0.216 |
|  | 0.529 |
| **Total** | **2.041** |



Δ Σ Γ Δ Θ encodes to 1 0111 0110 1 00

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





(1 + 2 + 3 + 4 + 4)/5 = 2.8 bits

Wouldn’t the probabilities of the words have to be used as the weightings? I thought it would be :

0.25 x 1 + 0.1 x 2 + 0.2 x 3 + 0.05 x 4 + 0.4 x 4 = 2.85 bits

Does anyone else agree with this?

I agree

Should it not be:

0.25\*4 + 0.1\*1 + 0.2\*3 + 0.05\*4 + 0.4\*2 = 2.7bits? The probabilities above aren’t matching the right word lengths

This came up during revision session, either answer is acceptable as the question is ambiguous.



2.8 – 2.04 = 0.76 bits

Therefore this one = 2.85-2.04 = 0.71? Should be 0.81? I got ~0.76 or so

10) Jeevan’s Answer

p1 = 0.25 (4); p2 = 0.1(1); p3 = 0.2(3); p4 = 0.05(4); p5 = 0.4(2);

1. H(x) = - (0.25\*log2(0.25) + 0.1\*log2(0.1) + 0.2\*log2(0.2) + 0.05\*log2(0.05) + 0.4\*log2(0.4))

= 2.04 bits of information

1. 1 0111 0110 1 00; prefix free encoding
2. Average Code Length (ACL) = 0.25\*4 + 0.1\*1 + 0.2\*3 + 0.05\*4 + 0.4\*2

= 2.7

1. Average Redundancy = ACL - H(x) = 2.7 - 2.04 = 0.66



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 ;) (Equifax)



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:

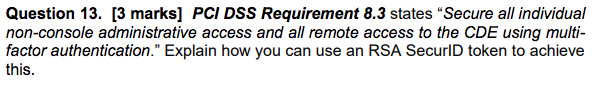
1. whether the access point is in the managed access point list

2. whether it is connected to the secure network

I don’t think this is the answer that Davo was looking for ^^, the above person is correct in explaining the deauthentication attack. However, the prevention method isn’t the one they’re looking for. From 2017 lecture 12 slides and audio the correct answer is something along the lines of:

Authentication frames are a type of management frame which are not encrypted, data frames ARE encrypted, the attacker can therefore launch a deauthentication attack without knowing any secret keys. The 802.11w extension to the 802.11 standard encrypts select management frames, including authentication frames which negate this attack.

I vouch for the blue solution as well.



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.

Is this question a trick or am I reading into it to much? Is Stuxnet itself technically the threat or is it the group that developed it the threat?

Stuxnet is the worm and not the group.

Stuxnet is an APT. It is advanced as it utilises 4 zero-days and the first ever PLC root-kit, this would require specialised knowledge and large technical resources. Persistent as it targeted 5 specific Iranian uranium enrichment plants, politically motivated by the United States of America and was planned. Threat because it had effective actions in place like root-kits to hide presence and intent to cause damage to nuclear program.