The University of Queensland
School of Information Technology and Electrical Engineering

Semester 2, 2017

COMS3000/7003 – Tutorial 7, Answers

Q1) What do the following UNIX directory listing lines mean in terms of access rights?

a)

```
-rwxr-x---  bill    student       thefile.txt
```

User *bill* has read, write and execute access, the group *students* have read and execute access, and all other users have no access to the file *thefile.txt*.

b)

```
-r-xrwxr--  bill    student       thefile.txt
```

User *bill* has read and execute access, the group *students* have read, write and execute access, and all other users have read access to the file *thefile.txt*.

Bill is a member of the group *student*. Does bill have write access to the file in this case?

This is a special case where the owner has fewer access rights than the group (or other users). There is a conflict here. Bill as owner of the file does not have write access, but he has access as a member of the group students.

Unix resolves this as follows:

if user = owner then apply owner permission
else if user in group then group permission
else other permission

That means that in this case, bill does not have write access to the file, but all other members of the group student do. Try it out!

Q2) Explain what problem the Set User Id feature in Unix[1] solves, how it works and what security problems it has.

The SUID feature allows a program to execute with the permissions of its owner rather than the permissions of the user running it. This allows giving users controlled access to files that they would not have otherwise (e.g. write access to the password file).

A security problem arises if an attacker can get control of a SUID program. The attacker can then execute arbitrary code with the access rights of the owner of the SUID program. A common method of gaining control of SUID programs is via the Buffer Overflow attack.

Q3) Assume that the Bell-LaPadula security model has been implemented in a system. Alice has a '*secret'* clearance and Bob's clearance is '*classified'*.

Which of the following operations are not allowed, assuming that both Alice and Bob operate at their highest clearance level?

Alice reads a document written by Bob.
Read down, OK.

Bob reads a document written by Alice.
Not allowed! Alice can only write secret or higher documents (No write down). Bob cannot read documents higher than classified (No read up rule).

Bob sends Alice a document that he has written.
OK

Alice sends Bob a document that she has written.
Not allowed.

Alice reads a document with the label 'secret'.
OK

Bob reads an unclassified document and sends it to Alice.
OK

[1] Throughout this tutorial "Unix" includes any UNIX®-like operating system.

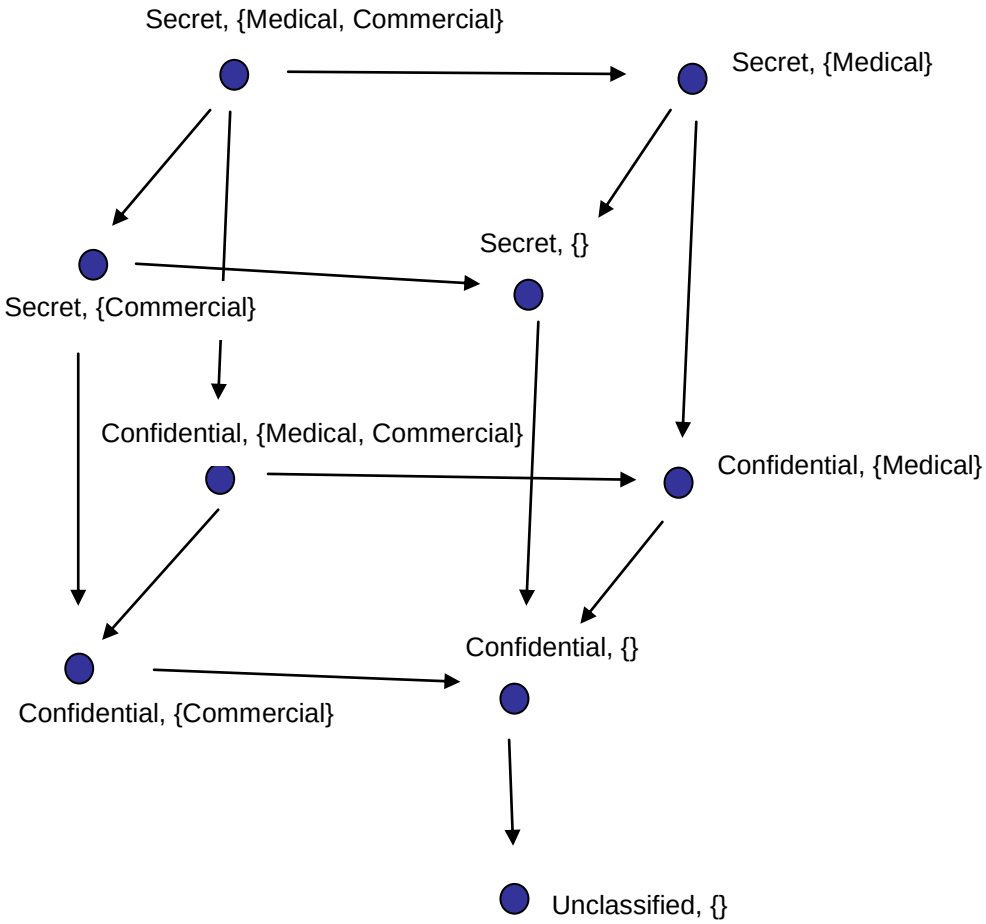
Q4) Draw a lattice with the levels “Secret”, “Confidential”, and “Unclassified”, where Secret and Confidential levels may have any combination of “Commercial” and “Medical” compartments.

Does *(Secret, {Medical})* dominate *(Confidential, {Commercial})* ?

No, because it doesn’t include the “Commercial” category, even though Secret is higher than Confidential. There is no path in the lattice from (Secret, {Medical}) to (Confidential, {Commercial}.

Does (Secret, {Commercial, Medical}) dominate (Confidential, {Medical})?

Yes, because it includes the compartment “Medical”, and Secret is a higher level than Confidential. There exists a corresponding path in the lattice.



Q5) In a multilevel policy (assume BLP), what action would cause the subject's "high water mark" to change? Give an example.

Actions that cause a high water mark to change are reading an object that has a classification higher than the current high water mark. Note that the file would have to have a classification equal or lower than the subject's clearance.
For example, Bob has a clearance of top-secret. His current level is classified. Since a security level is never downgraded, his current level is equal to his highest level, i.e. his High Water Mark. By reading a secret file, his level is automatically upgraded to secret.

Q6) Consider a random variable with the following equally likely outcomes: A, B, C, D, E.

a) What is the Entropy?

(Hint: If your calculator does not have a $\log_2$ function, you can use the following relationship: $\log_2 x = \log_{10} x \ / \log_{10} 2$)

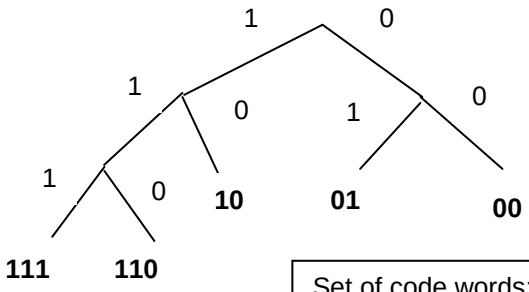
I(X) = log2(5) = 2.3219 bits

b) Find an encoding scheme that uses the smallest possible number of bits per outcome.

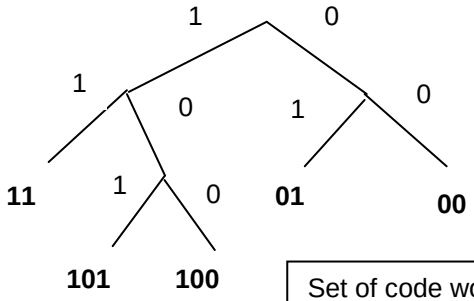
If we use a fixed length code, we need 3 bits per outcome.
For example:
A → 000
B → 001
C → 010
D → 011
E → 100

We can do better by using a variable length code. The code needs to be prefix-free, so that it allows unambiguous decoding. We can do this by using the leaf nodes of a binary tree as code words.

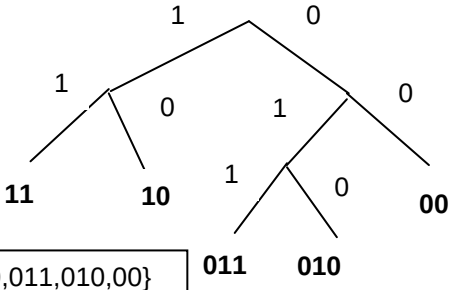
The following four solutions are equivalent:



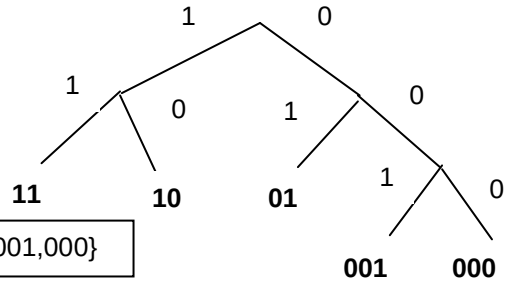
Set of code words: {111,110,10,01,00}



Set of code words: {11,101,100,01,00}



Set of code words: {11,10,011,010,00}



Set of code words: {11,10,01,001,000}

So, for example we can use the following mapping:
A → 111
B → 110
C → 10
D → 01
E → 00
The average number of bits per code word is 3/5 * 2 + 2/5 *3 = 2.4 bits
This comes pretty close to the theoretical minimum of $I(X) = \log_2(5) = 2.3219$ bits

Verify for yourself that for prefix-free codes, decoding is always unambiguous.
If you have any bit string such as 1001011011101011110111001 1…, you can decode it by starting at the top of the binary tree and move downwards in the tree. For a 1 in the bit string, branch to the left, for a 0, branch to the right. Continue this until you reach a leaf node, i.e. code word. Then start again at the top of the tree for the next code word.

Q7) In the lecture (Week 7 slide 74 said "See Tutorial" – this is it), it was mentioned that for a random variable X with two outcomes x_1 and x_2 with probabilities p_1 and p_2, we get maximum Entropy if $p_1 = p_2$. Can you prove this?

Hint! The derivative of $\log_a(x)$: $d(\log_a(x))/dx = 1/x\ln(a)$, where ln() is the natural logarithm $\log_e()$

Some tips regarding derivatives:
h(x) = g(x) * f(x)

Product Rule:

$$\frac{dh}{dx} = h'(x) = f(x)g'(x) + g(x)f'(x)$$

chain Rule:

$$\frac{dh}{dx} = h'(x) = f'(g(x))\ g'(x)$$

Answer:

For a maximum (or minimum) of a function, the first derivative of a function is typically zero. (This is the case in our scenario.)

We are trying to find p_1 for which $H(p_1)$ has the maximum value.

$$p_1 + p_2 = 1$$
$$p_2 = 1 - p_1$$
$$H(X) = -p_1 \log(p_1) - p_2 \log(p_2)$$
$$H(p_1) = -p_1 \log(p_1) - (1 - p_1)\log(1 - p_1)$$
$$\frac{d}{dp_1} H(p_1) = -\log(p_1) - \frac{p_1}{p_1 \ln 2} + \log(1 - p_1) + (1 - p_1)\frac{1}{(1 - p_1)\ln 2} =$$
$$-\log(p_1) - \frac{1}{\ln 2} + \log(1 - p_1) + \frac{1}{\ln 2} = -\log(p_1) + \log(1 - p_1) = 0$$
$$\log(p_1) = \log(1 - p_1)$$
$$\Rightarrow p_1 = 1 - p_1$$
$$p_1 = 0.5$$
$$p_2 = 0.5$$

Since $p_1 = 1 - p_1$ and [line 2] $p_2 = 1 - p_1$, **therefore $p_1 = p_2$** **Q.E.D.** (quod erat demonstrandum)

In general, for a random variable with N outcomes, the entropy is maximal if all outcomes are equally likely. The proof of this is a bit trickier and we will skip it (a proof with N>2 will not be examined). However, it makes sense intuitively. If all outcomes are equally likely, we have maximum uncertainty about the outcome, i.e. maximum Entropy.

Q8) Alice and Bob want to communicate via an insecure channel and decide to use a secret key encryption algorithm with a 128 bit key. They decide on the following method to determine their secret key: They toss a coin 16 times. If the result is 'heads' they write down a '1' if the result is 'tails', they write a '0'. Then, they compute the MD5 hash of this 16 bit number, which results in a 128 bit output.

What's the entropy of their 128 bit secret key?

Even though the resulting key has 128 bits, it only contains 16 bits of Entropy, 'Randomness' or 'Unpredictability'. Applying a deterministic (non-random) function such as MD5 to a random input does not add any Entropy or Randomness.

You can also simply calculate the Entropy using Shannon's formula. The input to the MD5 function is a 16-bit number, which has 2^{16} possible different values. The output of the MD5 function therefore has at the most 2^{16} different values (could potentially be less if there are collisions, but this is extremely unlikely). Assuming a perfect cryptographic one-way hash function (random oracle model), all of these 2^{16} 128-bit values are equally likely with probability $p=1/2^{16}$. All other 128 bit keys cannot occur and therefore have probability p=0. Events of probability 0 (or 1) do not contribute to the Entropy.
(Mathematically, we can say that $\lim_{p\to 0}(-p\log p)=0$)

We therefore have H(X) = 2^{16} x -($1/2^{16}$ x log ($1/2^{16}$) = 16 bits

An attacker who knows how Alice and Bob create their secret keys would only have to search a 16 bit key space in a brute force attack.

This is an important and fundamental problem in Information Security. Keys should be random to have maximum Entropy and therefore maximum unpredictability and security. However, computers are completely deterministic machines and it is therefore impossible to algorithmically generate truly random numbers.

The only truly random number sources are those related to physical phenomena such as the rate of radioactive decay of an element or the thermal noise of a semiconductor diode. Some systems use user input (mouse movement, or key stroke timing) or physical phenomenon of hardware, e.g. hard disk seek times as a source of randomness.

An early version of the Netscape Web browser had a serious security flaw because the developers did not understand the concept of Entropy. The secret key used for encrypting data was chosen as follows:

The seed for the algorithm which computed the secret keys was derived by calculating the MD5 hash of the time of day in microseconds and the process ID. In essence, the secret key contained very little Randomness and Entropy and could easily be guessed. Since the source code was not revealed, the developers probably also relied on security through obscurity (unsuccessfully).

This is a very old paper explaining those issues: http://www.cs.berkeley.edu/~daw/papers/ddj-netscape.html