Name: *Sample Answers*

ID:

THE UNIVERSITY OF AUCKLAND

FIRST SEMESTER, 2010
Campus: City

COMPUTER SCIENCE
Modern Data Communications
(Time allowed: 45 minutes)

NOTE: Examination conditions: Closed book, no calculators permitted.

Attempt **ALL** questions. Total possible: **50 marks.**

Please write your **name and UPI** on all pages of this answer book.

In the questions on security protocols,

- a and b are public keys for Alice and Bob,
- a' and b' are the corresponding private keys,
- t_a and t_b are timestamps,
- E is a public-key encryption function, and D is the corresponding decryption function.

A. The following questions refer to the series of three messages (M1, M2, M3) described below.

M1. Alice → Bob: E_b("Hi Bob, I have a new public key. Please use my new key c.", a , t_a)

M2. Bob → Alice: E_c("Hi Alice, no worries, I have updated my records.", t_a , t_b)

M3. Alice → Bob: E_b("Thanks!", a, t_a , t_b)

1. If Eve is able to intercept messages, will she know the values of a, b, c, t_a , and t_b? Explain briefly. **(5 marks)**

Eve can't learn anything from reading these messages, because they are encrypted. She'd need to know b' to decrypt M1 and M3, and she'd need to know c' to decrypt M2. However a, b, and c are public keys – they're not secrets, and indeed they may have been published by Alice or Bob. So we cannot be sure that Eve knows these public keys; nor can we be sure that Eve does ***not*** *know them. Usually, in a cryptoprotocol analysis, we make reasonable assumptions about what Eve might be able to do – in this case, it would be usual to assume that Eve knows a, b, c. If we can prove that our protocol is safe under this assumption, then we also know it is safe under the other reasonable assumption: that Eve doesn't know these values.*

Full marks were awarded for answers that show a good understanding of public-key cryptography, and of the M1-M3 protocol as specified above. No marks were deducted for varying assumptions about Eve's knowledge of a, b, c from sources other than M1-M3.

Some students assumed that Eve was able to know (or to guess) timestamps. This is a reasonable assumption if these clocks are low-precision, for example Eve might guess someone's clock to the nearest hour if she knows their timezone and time system (i.e. the Common Era / Western Calendar with timestamps in a particular format). However if the timestamps have high-precision (to the nanosecond) then it is not reasonable to assume that

Name:

ID:

Eve can guess t_a and t_b, unless she knows b' and c' (respectively), or unless she has some additional attacking power (such as physical access to the computers being used by Alice or Bob (respectively).

2. If Eve is able to fabricate messages, will she be able to impersonate Alice? Explain briefly. **(5 marks)**

Possibly. Eve would need to know Alice's public key a, and she'd also need to know Bob's public key b, in order to create an M1 and M3 which would fool Bob into accepting Eve's public key e. Here's the full attack:

M1'. Eve → Bob: E_b("Hi Bob, I have a new public key. Please use my new key e.", a , t_e)

M2'. Bob → Alice: E_c("Hi Alice, no worries, I have updated my records.", t_e , t_b)

M3'. Alice → Bob: E_b("Thanks!", a, t_e , t_b)

In the usual protocol of this type, Bob sends a "challenge message" to anyone who claims to be Alice. Alice knows a', and Eve does not, so a message like M2 could be a challenge if it had been encrypted under Alice's old key a. Eve would not be able to read t_b from such an M2, and thus she couldn't construct an M3 containing the value of t_b. By the way: Bob usually encrypts a randomly-generated number in his challenge message, in addition to a timestamp, because some portion of his timestamp might be guessed by Alice, because generating a random number is inexpensive, and because including some randomly-generated information (called "salt") in every encrypted message will decrease his exposure to known-plaintext attacks (by an Eve who can intercept many of his messages, then analyse them offline). Also: Bob should ***not*** *change Alice's key until he has some proof (from M3 in a well-constructed challenge-response protocol of this type) that he has been talking to Alice in M1-M2!*

Full marks were awarded for answers that demonstrated a student's understanding of the 3-message identity-authentication protocol discussed in the lecture slides. Some students discussed signed messages and signed message hashes in their answers, but if they didn't make it clear that no signatures are used in M1-M3, I generally concluded that the student had a very weak understanding of these concepts and gave them no marks for mentioning them in their answer.

I'd encourage all students to re-read section 7.4 of the textbook when preparing for the final exam. I believe that any student who ***doesn't*** *initially find the concepts of public-key cryptography to be very difficult is either fooling themselves, or else they are extremely clever!*

*The 3-step protocol for user authentication shown in your lecture slides (slide #42 of the "Encryption, Authentication" series) is the same as the one discussed in Section 10.4.1 of the 5th edition of Halsall's textbook (*Computer Networking and the Internet, *Addison-Wesley, 2005, pp. 652-3). The protocol in this problem is an insecure variation of that user-authentication protocol.*

B. The following questions refer to a channel that can carry analog signals from 20 Hz to 10020 Hz with a SNR of 30 dB.

3. If amplitude shift keying (ASK) is used on this channel, with one bit per symbol at a frequency of 1 kHZ, what is its bit rate? Explain briefly. **(5 marks)**

(1 b/symbol)(1 kHz)(1 (cycle/s)/Hz)(1 ASK symbol/cycle) = 1 kb/s.

Name:

ID:

Students who had difficulty with this question either didn't know that ASK encodes one symble per cycle, or else they don't understand how to do a dimensional analysis.

I'd strongly encourage students to read http://en.wikipedia.org/wiki/Dimensional_analysis if they're having trouble with dimensional analysis. This was the most important concept I learned while taking first-year physics when I was an undergraduate, and if you can "get the hang of it" I think you'll find most of the formulas in networking are essentially definitional and therefore trivial – so you'll have very little to memorise, aside from the names of units such as Baud – and you'll be able to recognise the remaining bits-and-pieces (such as the factor of 2 in the Nyquist bound, and the $\log_2(1 + S/N)$ factor in the Shannon bound) that are actually non-trivial.

4. Can we use amplitude shift keying with 1 bit per symbol, at a frequency of 10 kHz, on this channel? Hint: use the Nyquist theorem. To receive full credit, you must use the word "Baud" correctly in your answer. **(5 marks)**

The Nyquist limit for signalling (or sampling) on any channel is twice its maximum frequency. The maximum frequency on this channel is 10020 Hz, so we could signal at up to 20020 symbols/second = 20020 Baud if the channel were noise free and non-distorting. However the channel has a little noise (the noise is 30 dB below the signal) and a little distortion (it has a highpass at 20 Hz), so signalling at 20.020 kBaud would not be feasible. Signalling at less than half this rate, at only 1 bit/sample, seems very likely to be feasible but we should do a Shannon bitrate calculation to be certain.

Full marks were awarded for a correct application of the Nyquist limit. A more advanced version of Nyquist's result is that signalling (or sampling) is limited (from above, or from below, respectively) by twice the bandwidth of the channel, so I accepted 2(10020-20) = 20000 Baud as a Nyquist limit -- even though this advanced version was not discussed either in the lecture or the textbook, as far as I know.

5. What is the Shannon capacity (maximum bitrate) of this channel? Show your work. **(5 marks)**

(20020 Hz – 20 Hz) ($\log_2(1 + SNR)$ bits/cycle) = (20 kHz) ($\log_2(1 + 10^{30db/(10db/Bel)}$ bits/cycle)) (1 (cycle/s)/Hz) = 20 $\log_2(1001)$ kb/s ≈ 20(10) kb/s = 200 kb/s.

As noted during the revision at the end of my lecture series, I expect students to know how to simplify base-2 logs and how to perform base-10 exponentiations. A few students lost marks for writing Mb/s rather than kb/s. This is an "understandable" mistake (for anyone who knows Roman numerals) but is not acceptable. 1 Mb/s has a well-defined meaning (= 1000 kb/s).

C. Consider the following code:

0	0000
1	0001
2	0010
3	0011

4	0100
5	0101
6	0110
7	0111

8	1000
9	1001
A	1010
B	1011

C	1100
D	1101
E	1110
F	1111

Name:

ID:

6. Is this code prefix-free? Explain briefly. **(5 marks)**

Yes. This is a fixed-length code (all codewords have length 4), so no codeword is a proper prefix of any other codeword.

There are no equal codewords, i.e. this code is uniquely decodable, implying that this code would be "prefix-free" even if we considered non-proper prefixes (i.e. considering that any string is a prefix of itself). Although the definition of prefix-free given in the lecture slide was limited to proper prefixes, I was encouraged by the students (perhaps 5%) who mentioned that there were no equal codewords, for it showed strong understanding of the fundamentals of coding theory even if it is (strictly speaking) irrelevant to the problem at hand.

7. We can use this code to transmit a sequence of positive integers, in the following way. Each integer is represented as a string of symbols in the range 0-E. The symbol F indicates the end of the string that represents an integer. The value of the integer is the sum of the (hexadecimal) values of the symbols (in the range 0-9 and A-E) in its string. For example, the sequence (14, 9, 0, 15) can be represented as the string "EF9FFE1F", which is encoded as 1110 1111 1001 1111 1111 1110 0001 1111. Using this method, encode the sequence (14, 30). Explain briefly. **(5 marks)**

The integer 14 (with its terminator F) can be represented as EF, and the integer 30 can be represented as EE2F. The sequence (14,30) can thus be encoded at EFEE2F.

Full marks were awarded for EFAAAF, and also for longer encodings such as EFE1E1F because the question didn't require a minimal-length encoding. Some students seemed to be using a positional notation, i.e. encoding (30) as 30F. A positional notation could allow a more efficient code, especially for numbers larger than 42, than the one given here. However there's a lot more involved in changing a network protocol than devising an alternative that (seems to be) a "better idea"! In this case, if you sent 30F, it would be interpreted as (3).

When revising for the final exam, I'd encourage students to review the example of a run-length code found on slide 61 of the Compression sequence. That slide uses a somewhat similar method for encoding integers, which might be explained as follows. Code F means "15+", and codes 0 through E are the self-terminating integer values 0 through 14 (respectively).

D. The following questions refer to a Hamming code with the following properties.

- Each codeword is 7 bits long: c7 c6 c5 c4 c3 c2 c1. Four data bits (d4, d3, d2, d1) are transmitted in positions c7, c6, c5, and c3 of the codeword. Three check bits (r4, r2, r1) are transmitted in positions c4, c2, and c1 of the codeword.

c7	c6	c5	c4	c3	c2	c1
d4	d3	d2	r4	d1	r2	r1

- Check bit r1 is even parity on c1, c3, c5, and c7.
- Check bit r2 is even parity on c2, c3, c6, and c7.
- Check bit r4 is even parity on c4, c5, c6, and c7.

Name:

ID:

8. If d1 = 1, d2 = 1, d3 = 1, and d4 =1 are being transmitted, what are the values of r1, r2, and r3? Show your work. **(5 marks)**

We are given c7 = d4 = 1, c6 = d3 = 1, c5 = d2 = 1, and c3 = d1 = 1. Thus r1 = c1= XOR(c7,c5,c3) = XOR(1,1,1) = 1; r2 = c2 = XOR(c7,c6,c3) = 1; and r4 = c4 = XOR(c7,c6,c5) = 1.

9. If the codeword 1000000 is received (that is, if c7 = 1 and all other codeword bits are 0), what is the corrected data? Show your work. **(5 marks)**

It's moderately difficult to see how to find the codeword at minimal Hamming distance from 1000000, unless you know Hamming's algorithm (as presented in the lecture slides). There's clearly an error in 1000000, since all three checkbits are received as 0 but would be 1 if the received codeword were error-free. Students who guessed (or who recognised) this problem as specifying a Hamming 1-correcting code would realise that the error signature of this codeword is 111 (since all three checkbits are apparently in error), which can be interpreted in binary as implying that the seventh codebit (c7) is in error. The corrected codeword is 0000000, and the corrected data (d4-d1) is 1000.

3 marks were awarded to students who found codewords (such as 1001011) that are at Hamming distance 2 or 3 from 1000000. Such "corrections" are very unlikely to be accurate, if we assume that bit-errors are equally likely to occur at any point in a word. However these students did show some understanding of data correction.

10. Is there any chance of an uncorrected error? Explain briefly. **(5 marks)**

Yes, no error-correcting code can correct all possible errors.

A couple of students noticed that this code has three checkbits, allowing it to distinguish $2^3 - 1 = 7$ errors from the no-error case. It assigns a different error signature to each single-bit error that can occur in the 7-bit codeword, implying any multiple-bit error case will be handled incorrectly.

Frequencies

Statistics

Test

N	Valid	202
	Missing	8
Percentiles	10	15.00
	20	21.00
	30	25.00
	40	29.00
	50	32.00
	60	33.00
	70	36.00
	80	40.00
	90	42.00