ELEN 330 – Information Engineering Techniques
Examination, June 2002

Instructions to candidates:

1. Answer all questions. Total marks for this paper is 75. Full marks is 65.
2. Show all working: marks will be allocated for all working and logical reasoning and not just for the correct answer.
3. Give reasons for any assumptions that are made.
4. Type 2 exam – this is a closed book exam. Candidates are allowed to use an engineering calculator and one handwritten A4 information sheet.

Question 1

(a) The following small sample of plaintext is considered to be representative of a particular alphabet and language. Assuming that the characters found in the sample are the only characters in this alphabet, devise a translation table for a homophonic substitution cipher and use it to encipher the plaintext. You may choose to use spare characters from the normal English alphabet in your translation table. By what factor is the ciphertext larger than the plaintext? (Assume that a whole number of bits is required to represent the members of the alphabet in each case).

SHE SELLS SEA SHELLS **(5 marks)**

(b) Design a simple "paper and pencil" cipher algorithm which makes use of classical elements of confusion and diffusion. The algorithm itself must be fixed but the agents who will use it must be able to select a secret key without which it would be difficult to break the ciphertext. Take care to ensure that:

- The key space is large enough to make it impractical to break ciphertext by a brute force attack within a reasonable period of time.
- The cipher is robust against common classical cryptanalytical techniques such as letter frequency and digram/trigram analysis.
- Your cipher is specified completely (i.e. it must be possible to create unambiguous ciphertext from plaintext and vice-versa).

(7 marks)

(c) The permutation function below is used by Rufus to generate translation tables for a monoalphabetic substitution cipher. The value **a** (ranging from 0 to 25) represents one of the 26 letters of the alphabet and different values of the key **k** give different translation tables. Using your knowledge of the rules for modular arithmetic, determine the keyspace of this cipher.

$$\mathbf{P_1(a) = (k*a + 7) \bmod 26}$$ **(5 marks)**

Question 2

(a) In a particular language there are 12 letters. Three of these are used with relative frequency 4, five are used with relative frequency 2, and the remaining four are used with relative frequency 1. Compute the index of coincidence for monoalphabetic substitutions in this language.

(4 marks)

(b) The IC (Index of Coincidence) per number of alphabets used in a polyalphabetic substitution encipherment for a particular language is as follows:

No. Alphabets:	1	2	3	4	5	Large
IC:	0.080	0.071	0.061	0.049	0.042	0.038

The table below shows factors obtained from a Kasiski analysis on a ciphertext produced by enciphering a sample plaintext of this language. The IC for the associated monoalphabetic ciphers is shown alongside each of these factors.

Kasiski Distance Factors	**IC per monoalphabetic**
7	0.079
3	0.042
13	0.039
21	0.081

Decide what the most likely number of alphabets is and fully explain your decision. Why would a table of digram frequencies *not* be a useful tool for cryptanalysing the monoalphabetic ciphers suggested by the analysis?

(3 marks)

(c) Someone proposes that the block size of the IDEA cipher be reduced but the key size be kept the same. In what way could this compromise the security of the cipher?

(2 marks)

(d) To date it has not been proven that IDEA is not a closed cipher. Devise and describe an experiment (even if not practical, computationally) which would decide whether or not it is closed. Describe your algorithm/experiment in enough detail for an implementation. What would be the implication of the result for 3IDEA (triple-IDEA, an EDE scheme like triple-DES)?

(4 marks)

(e) The table below shows the relationship between IDEA's encryption and decryption subkeys. Explain the meaning of the symbols in the decryption column and how the decryption subkeys are determined from the encryption subkeys.

(3 marks)

IDEA Encryption and Decryption Subkeys

Round	Encryption Subkeys	Decryption Subkeys
1st	$Z_1^{(1)}\ Z_2^{(1)}\ Z_3^{(1)}\ Z_4^{(1)}\ Z_5^{(1)}\ Z_6^{(1)}$	$Z_1^{(9)-1}\ -Z_2^{(9)}\ -Z_3^{(9)}\ Z_4^{(9)-1}\ Z_5^{(8)}\ Z_6^{(8)}$
2nd	$Z_1^{(2)}\ Z_2^{(2)}\ Z_3^{(2)}\ Z_4^{(2)}\ Z_5^{(2)}\ Z_6^{(2)}$	$Z_1^{(8)-1}\ -Z_3^{(8)}\ -Z_2^{(8)}\ Z_4^{(8)-1}\ Z_5^{(7)}\ Z_6^{(7)}$
3rd	$Z_1^{(3)}\ Z_2^{(3)}\ Z_3^{(3)}\ Z_4^{(3)}\ Z_5^{(3)}\ Z_6^{(3)}$	$Z_1^{(7)-1}\ -Z_3^{(7)}\ -Z_2^{(7)}\ Z_4^{(7)-1}\ Z_5^{(6)}\ Z_6^{(6)}$
4th	$Z_1^{(4)}\ Z_2^{(4)}\ Z_3^{(4)}\ Z_4^{(4)}\ Z_5^{(4)}\ Z_6^{(4)}$	$Z_1^{(6)-1}\ -Z_3^{(6)}\ -Z_2^{(6)}\ Z_4^{(6)-1}\ Z_5^{(5)}\ Z_6^{(5)}$
5th	$Z_1^{(5)}\ Z_2^{(5)}\ Z_3^{(5)}\ Z_4^{(5)}\ Z_5^{(5)}\ Z_6^{(5)}$	$Z_1^{(5)-1}\ -Z_3^{(5)}\ -Z_2^{(5)}\ Z_4^{(5)-1}\ Z_5^{(4)}\ Z_6^{(4)}$
6th	$Z_1^{(6)}\ Z_2^{(6)}\ Z_3^{(6)}\ Z_4^{(6)}\ Z_5^{(6)}\ Z_6^{(6)}$	$Z_1^{(4)-1}\ -Z_3^{(4)}\ -Z_2^{(4)}\ Z_4^{(4)-1}\ Z_5^{(3)}\ Z_6^{(3)}$
7th	$Z_1^{(7)}\ Z_2^{(7)}\ Z_3^{(7)}\ Z_4^{(7)}\ Z_5^{(7)}\ Z_6^{(7)}$	$Z_1^{(3)-1}\ -Z_3^{(3)}\ -Z_2^{(3)}\ Z_4^{(3)-1}\ Z_5^{(2)}\ Z_6^{(2)}$
8th	$Z_1^{(8)}\ Z_2^{(8)}\ Z_3^{(8)}\ Z_4^{(8)}\ Z_5^{(8)}\ Z_6^{(8)}$	$Z_1^{(2)-1}\ -Z_3^{(2)}\ -Z_2^{(2)}\ Z_4^{(2)-1}\ Z_5^{(1)}\ Z_6^{(1)}$
output transformation	$Z_1^{(9)}\ Z_2^{(9)}\ Z_3^{(9)}\ Z_4^{(9)}$	$Z_1^{(1)-1}\ -Z_2^{(1)}\ -Z_3^{(1)}\ Z_4^{(1)-1}$

Question 3

A bag contains 1 000 000 green balls and 10 red balls. Given that 1000 draws will be made from the bag *without replacement*,

(a) Derive an exact numerical expression for the chance of drawing at least one red ball in all 1000 draws. Do not use a calculator to reduce the above quantities to a floating point number – give a numerical expression which represents the exact value required.

(4 marks)

(b) Generalise your result and give it in terms of T, the total number of balls, R, the number of red balls and n, the number of draws. Assuming that $R \ll n \ll T$, make and justify approximations which give the final result as a simple expression with one term which is linear in each of the parameters R, n and T. (Your first approximation will be that for a very large population and a small number of draws, probabilities are weakly conditional and that $P(A|B) \approx P(A)$). **(6 marks)**

(c) A cryptological problem which parallels this one exactly is that of determining the chance of a meaningful but incorrect plaintext arising in a brute-force (entire keyspace) search of a ciphertext. Suppose that a ciphertext of 32 bytes has been produced by a cipher of key length 64 bits. The plaintext is to be found by searching the entire key space of the cipher. Given that it is possible to construct a total of 10 000 000 000 (10 billion) different but meaningful plaintexts which are 32 bytes long, use the solution to part (b) of this question (the approximation) to determine the chance of an incorrect plaintext arising in an entire keyspace search. Begin answering this question by identifying analogous items in the two scenarios.

(4 marks)

Question 4

(a) Explain how one would attempt to derive the private part of an RSA key from its public part, and why this is a computationally difficult task. **(2 marks)**

(b) A public key is (143,103). From this information, determine the corresponding private key and write down the encryption and decryption functions. (Note: starting with small values of private key, this takes only a minute using a calculator). Hence find the value S_i obtained on signing the block of information M = 14. **(6 marks)**

(c) Describe how you would go about determining the keyspace of this cipher (having n = 143). **(2 marks)**

(d) Below are three lines from the Java program introduced in lab I2 (and given in the information sheet). Explain what the method contained in each line does, give an example of where it may be used in the context of RSA and give the meaning of each argument.

```
static BigInteger p = new BigInteger(512,10,rr);
e = new BigInteger(511,rr);
a = e.gcd(phi);
```

(7 marks)

Question 5

(a) Every Internet user is certain that their copy of Trent (the "trusted arbitrator's") public key really is his. Trent is able to vouch for the identity of Internet users and their associated public keys. Explain how Trent could use cryptographic functions to render this valuable service to the public.

(2 marks)

(b) Sketch the Huffman tree corresponding to the message DBDBDADBD. Using this tree, decode the message below. Calculate the compression factor of the Huffman code in this case.

(4 marks)

11011100110111010100

(c) The Morse Code (given in the information sheet) represents a form of statistical compression coding used for low bandwidth communication. Treating dots and dashes as binary digits and ignoring the fact that inter-symbol and inter-word pauses are also a vital part of the information in Morse Code, determine the compression factor achieved when used with the English Language. Comment on the suitability of Morse to English transmission – can you suggest any improvements which could be made to the code? What comment would non-English speaking nations likely have made on the establishment of Morse Code as an "International Standard"?

(5 marks)

Information Sheet

The Morse Code

A	B	C	D	E	F	G
.-	-...	-.-.	-..	.	..-.	--.
H	**I**	**J**	**K**	**L**	**M**	**N**
....	..	.---	-.-	.-..	--	-.
O	**P**	**Q**	**R**	**S**	**T**	**U**
---	.--.	--.-	.-.	...	-	..-
V	**W**	**X**	**Y**	**Z**		
...-	.--	-..-	-.--	--..		

Mnemonic popular among amateur cryptanalysts: **ETAOIN**

IC vs Number of alphabets for the English language:

No. Alphabets:	1	2	3	4	5	10	Large
IC:	0.068	0.052	0.047	0.044	0.044	0.041	0.038

Letter frequency distribution for the English Language

A	B	C	D	E	F	G
0.0749	0.0129	0.0354	0.0362	0.1400	0.0218	0.0174
H	**I**	**J**	**K**	**L**	**M**	**N**
0.0422	0.0665	0.0027	0.0047	0.0357	0.0339	0.0674
O	**P**	**Q**	**R**	**S**	**T**	**U**
0.0737	0.0243	0.0026	0.0614	0.0695	0.0985	0.0300
V	**W**	**X**	**Y**	**Z**		
0.0116	0.0169	0.0028	0.0164	0.0004		

Index of Coincidence:

$$IC = \sum_{i=a}^{i=z} \frac{(Freq_i)(Freq_i - 1)}{N(N-1)}$$

RSA3.JAVA

```
/*
      The RSA3 class implements the RSA algorithm using
      Java's BigInteger methods.

      Copyright (c) G.D.Agnew 2001

*/

import java.math.BigInteger;
import java.util.Random;
class RSA3
{


//  Define the objects to be used: all of these have the same meaning
//  as the symbols used in the class notes on RSA:


      static Random rr = new Random(73621781266L); //  Random no gen for later
      static BigInteger p = new BigInteger(512,10,rr);
      static BigInteger q = new BigInteger(512,10,rr);
      static BigInteger n = new BigInteger("1");

      static BigInteger phi = new BigInteger("1");
      static BigInteger e = new BigInteger("1");
      static BigInteger d = new BigInteger("1");

      static BigInteger M = new BigInteger("1");      //  Message text
      static BigInteger C = new BigInteger("1");      //  ciphertext
      static BigInteger M2 = new BigInteger("1");     //  Recovered message
      static BigInteger a,b,c = new BigInteger("1");  //  Intermediate vars

      static BigInteger ONE = new BigInteger("1");
      static BigInteger ZERO = new BigInteger("0");

    public static void main(String[] args) {

      System.out.println("KEY GENERATION:");
      System.out.println("Generating random prime p:");

//  Two large random numbers p and q are generated. This is done
//  using the BigInteger(,,) function described here:
//
//    BigInteger(int bitLength,
//          int certainty,
//          Random rnd)
//     Constructs a randomly generated positive BigInteger that is probably
//     prime, with the specified bitLength.

      p = new BigInteger(512,10,rr);
      System.out.println("  p = "+p);

      System.out.println("Generating random prime q:");
      q = new BigInteger(512,10,rr);
      System.out.println("  q = "+q);

      System.out.println("Calculating product n:");
//    Find the product of p and q:   n = p*q:
      n = p.multiply(q);
      System.out.println("  n = "+n);
```

```
// Calculate (p-1):
      p = p.subtract(ONE);
// Calculate (q-1):
      q = q.subtract(ONE);

//  Calculate (p-1)(q-1):
      phi = p.multiply(q);
      System.out.println("  (p-1)(q-1) = "+phi);

//  The method below randomly selects and encryption key e using
//  BigInteger(). This key must be relatively prime to phi: relative
//  primeness is easily tested using the gcd function (see below).
//  If gcd returns 1 then they ARE relatively prime. If it does not
//  return 1, another prospective key must be randomly selected
//  and the gcd test repeated; until the requirement is satisfied.
//  The code below does not implement the while loop required to
//  ensure that the generation & test are repeated until a good
//  value of e is found.
//
//  Description of BigInteger(,) function:
//    BigInteger(int numBits, Random rnd)
//    Constructs a randomly generated BigInteger, uniformly distributed over
//          the range 0 to (2^numBits - 1), inclusive.
//
      e = new BigInteger(512,rr);
      System.out.println("  Randomly chosen encryption key e = "+e);

//      gcd(BigInteger val)
//             Returns a BigInteger whose value is the greatest
//             common divisor of abs(this) and abs(val).

      a = e.gcd(phi); //  Make sure that e and (p-1)(q-1) are relatively prime
      System.out.println("  a (gcd) = "+a); //  Show gcd outcome. 1 is good

//  Find the inverse of e, modulo (p-1)(q-1). This is the decryption
//  key d:
      d = e.modInverse(phi);
      System.out.println("  Decryption key d = "+d);

      System.out.println("ENCRYPTION:");
//  Generate an arbitrary message M: (normally the message would be a block
//  of bytes representing a binary file). 512 specifies the message size.
//
//
      M = new BigInteger(1023,rr);
      System.out.println("  Plaintext M =  "+M);

//  Do the encryption:
      C = M.modPow(e,n);
      System.out.println("  Ciphertext C =  "+C);

      System.out.println("DECRYPTION:");
//  Do the decryption:
      M2 = C.modPow(d,n);
      System.out.println("  Recovered plaintext M2 =  "+M2);

   }
}
```