SOLUTIONS MANUAL
for
INTRODUCTION TO CRYPTOGRAPHY
with Coding Theory, 2nd edition

Wade Trappe
Wireless Information Network Laboratory
and the Electrical and Computer Engineering Department
Rutgers University

Lawrence C. Washington
Department of Mathematics
University of Maryland

August 26, 2005

Contents

Exercises

Mathematica problems

Maple problems

0

MATLAB problems

Chapter 2 - Exercises

1. Among the shifts of *EVIRE*, there are two words: *arena* and *river*. Therefore, Anthony cannot determine where to meet Caesar.

2. The inverse of 9 mod 26 is 3. Therefore, the decryption function is $x = 3(y-2) = 3y-2 \pmod{26}$. Now simply decrypt letter by letter as follows. $U = 20$ so decrypt U by calculating $3 * 20 - 6 \pmod{26} = 2$, and so on. The decrypted message is 'cat'.

3. Changing the plaintext to numbers yields 7, 14, 22, 0, 17, 4, 24, 14, 20. Applying $5x+7$ to each yields $5 \cdot 7+7 = 42 \equiv 16 \pmod{26}$, $5 \cdot 14+7 = 77 \equiv 25$, etc. Changing back to letters yields *QZNHOBXZD* as the ciphertext.

4. Let $mx + n$ be the encryption function. Since $h = 7$ and $N = 13$, we have $m \cdot 7 + n \equiv 13 \pmod{26}$. Using the second letters yields $m \cdot 0 + n \equiv 14$. Therefore $n = 14$. The first congruence now yields $7m \equiv -1 \pmod{26}$. This yields $m = 11$. The encryption function is therefore $11x + 14$.

5. Let the decryption function be $x = ay + b$. The first letters tell us that $7 \equiv a \cdot 2 + b \pmod{26}$. The second letters tell us that $0 \equiv a \cdot 17 + b$.Subtracting yields $7 \equiv a \cdot (-15) \equiv 11a$. Since $11^{-1} \equiv 19 \pmod{26}$, we have $a \equiv 19 \cdot 7 \equiv 3 \pmod{26}$. The first congruence now tells us that $7 \equiv 3 \cdot 2 + b$, so $b = 1$. The decryption function is therefore $x \equiv 3y + 1$. Applying this to *CRWWZ* yields *happy* for the plaintext.

6. Let $mx+n$ be one affine function and $ax+b$ be another. Applying the first then the second yields the function $a(mx + n) + b = (am)x + (an + b)$, which is an affine function. Therefore, successively encrypting with two affine functions is the same as encrypting with a single affine function. There is therefore no advantage of doing double encryption in this case. (Technical point: Since $\gcd(a, 26) = 1$ and $\gcd(m, 26) = 1$, it follows that $\gcd(am, 26) = 1$, so the affine function we obtained is still of the required form.)

7. For an affine cipher $mx + n \pmod{27}$, we must have $\gcd(27, m) = 1$, and we can always take $1 \leq m \leq 27$. So we must exclude all multiples of 3, which leaves 18 possibilities for m. All 27 values of n are possible, so we have $18 \cdot 27 = 486$ keys. When we work mod 29, all values $1 \leq m \leq 28$ are allowed, so we have $28 \cdot 29 = 812$ keys.

8. (a) In order for α to be valid and lead to a decryption algorithm, we need $\gcd(\alpha, 30) = 1$. The possible values for α are $1, 7, 11, 13, 17, 19, 23, 29$.

(b) We need to find two x such that $10x \pmod{30}$ gives the same value. There are many such possible answers, for example $x = 1$ and $x = 4$ will work.

This corresponds to the letters 'b' and 'e'.

9. If $x_1 = x_2 + (26/d)$, then $\alpha x_1 + \beta = \alpha x_2 + \beta + (\alpha/d)26$. Since $d = \gcd(\alpha, 26)$ divides α, the number $\alpha/26$ is an integer. Therefore $(\alpha/d)26$ is a multiple of 26, which means that $\alpha x_1 + \beta \equiv \alpha x_2 + \beta \pmod{26}$. Therefore x_1 and x_2 encrypt to the same ciphertext, so unique decryption is impossible.

10. (a) In order to find the most probable key length, we write the ciphertext down on two strips and shift the second strip by varying amounts. The shift with the most matches is the most likely key length. As an example, look at the shift by 1:

B	A	B	A	B	A	A	A	B	A	
	B	A	B	A	B	A	A	A	B	A
						*	*			

This has a total of 2 matches. A shift by 2 has 6 matches, while a shift by 3 has 2 matches. Thus, the most likely key length is 2.

(b) To decrypt, we use the fact that the key length is 2 and extract off every odd letter to get BBBAB, and then every even letter to get AAAAA. Using a frequency count on each of these yields that a shift of 0 is the most likely scenario for the first character of the Vigenere key, while a shift of 1 is the most likely case for the second character of the key. Thus, the key is *AB*. Decrypting each subsequence yields BBBAB and BBBBB. Combining them gives the original plaintext BBBBBBABBB.

11. If we look at shifts of 1, 2, and 3 we have 2, 3, and 1 matches. This certainly rules out 3 as the key length, but the key length may be 1 or 2.

In the ciphertext, there are 3 *A*'s, 5 *B*'s, and 2 *C*'s. If the key length were 1, this should approximate the frequencies .7, .2, .1 of the plaintext in some order, which is not the case. So we rule out 1 as the key length.

Let's consider a key length of 2. The first, third, fifth, ... letters are *ACABA*. There are 3 *A*'s, 1 *B*, and 1*C*. These frequencies of .6, .2, .2 is a close match to .7, .2, .1 shifted by 0 positions. The first element of the key is probably *A*. The second, fourth, ... letters of the ciphertext are *BBBBC*. There are 0 *A*'s, 4 *B*'s, and 1 *C*. These frequencies .0, .8, .2 and match .7, .2, .1 with a shift by 1. Therefore the second key element is probably *B*.

Since the results for length 2 match the frequencies most closely, we conclude that the key is probably *AB*.

12. Since the entries of $\mathbb{A}_i$ are the same as those in $\mathbb{A}_0$ (shifted a few places) the two vectors have the same length. Therefore

$$\mathbb{A}_0 \cdot \mathbb{A}_i = |\mathbb{A}_0||\mathbb{A}_i| \cos\theta = |\mathbb{A}_0|^2 \cos\theta.$$

Note that $\cos\theta \leq 1$, and equals 1 exactly when $\theta = 0$. But $\theta = 0$ exactly when the two vectors are equal. So we see that the largest value of the cosine is when $\mathbb{A}_0 = \mathbb{A}_i$. Therefore the largest value of the dot product is when $i = 0$.

13. Change *YIFZMA* to pairs of numbers: (24, 8), (5, 25), (12, 0). Invert the matrix to get $N = \begin{pmatrix} 3 & -13 \\ -2 & 9 \end{pmatrix} \equiv \begin{pmatrix} 3 & 13 \\ 24 & 9 \end{pmatrix} \pmod{26}$. Calculate $(24, 8)N = (4, 20)$, $(5, 25)N = (17, 4)$, $(12, 0)N = (10, 0)$. Change back to letters: *eureka*.

14. Suppose the encryption matrix M is $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$. Change the ciphertext to numbers: (6, 4), (25, 23), (3, 18). Change the plaintext to numbers: (18, 14), (11, 21), (4,3). We know $(18,14)M \equiv (6,4)$, etc. We'll use $(11,21)M \equiv (25,23)$ and $(4,3)M \equiv (3,18)$ to get equations for a, b, c, d, which are most easily put in matrix form: $\begin{pmatrix} 11 & 21 \\ 4 & 3 \end{pmatrix}\begin{pmatrix} a & b \\ c & d \end{pmatrix} \equiv \begin{pmatrix} 25 & 23 \\ 3 & 18 \end{pmatrix}$. The inverse of $\begin{pmatrix} 11 & 21 \\ 4 & 3 \end{pmatrix}$ mod 26 is $\begin{pmatrix} 3 & 5 \\ 22 & 11 \end{pmatrix}$. Multiply by this matrix to obtain $M = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \equiv \begin{pmatrix} 12 & 3 \\ 11 & 2 \end{pmatrix}$.

15. Suppose the matrix has the form

$$M = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}$$

Then the encryption of a plaintext $x = (b, a) = (1, 0)$ yields (α, β). We know this corresponds to HC, and hence $\alpha = 7$ and $\beta = 2$. The second piece of information is that zz encrypts to GT. This corresponds to a plaintext of $(25, 25)$ or equivalently $(-1, -1)$. Using this yields $-\alpha - \gamma = 6$ and $-\beta - \delta = 19$. Thus, $\gamma = 13$ and $\delta = 5$.

16. (a) The plaintext is (3,14), (13, 19). The ciphertext is (4,11), (13, 8). We have $\begin{pmatrix} 3 & 14 \\ 13 & 19 \end{pmatrix} M \equiv \begin{pmatrix} 4 & 11 \\ 13 & 8 \end{pmatrix}$. The inverse of $\begin{pmatrix} 3 & 14 \\ 13 & 19 \end{pmatrix}$ mod 26 is $\begin{pmatrix} 19 & 12 \\ 13 & 3 \end{pmatrix}$. Multiplying by this inverse yields $M \equiv \begin{pmatrix} 10 & 9 \\ 13 & 23 \end{pmatrix}$.

(b) We have $\begin{pmatrix} 3 & 14 \\ 13 & 19 \end{pmatrix} M \equiv \begin{pmatrix} 4 & 11 \\ 13 & 10 \end{pmatrix}$. Proceeding as in part (a), we find $M \equiv \begin{pmatrix} 10 & 19 \\ 13 & 19 \end{pmatrix}$.

17. Suppose the plaintext is of the form (x, y), then the ciphertext is of the form $(x + 3y, 2x + 4y)$ (mod 26). There will be many possible plaintexts that will map to the same ciphertext. We will try to make plaintexts that yield a ciphertext of the form $(0, *)$. To do so, we will have the relationship $x = -3y$ (mod 26). Now we need to find two y values that produce the same $2(-3y) + 4y = -2y$ (mod 26). If we take $y = 4$ and $y = 17$ then we get the same value for $-2y$ (mod 26). Thus, $(14, 4)$ and $(1, 17)$ are two plaintexts that map to $(0, 18)$.

18. We will need to use three different plaintexts. First, choose $(x, y) = (0, 0)$. This will produce a ciphertext that is precisely (e, f). Next, try $(x, y) = (1, 0)$. This will produce a ciphertext that is $(a, b) + (e, f)$. We may subtract off (e, f) to find (a, b). Finally, use $(x, y) = (0, 1)$ to get $(c, d) + (e, f)$ as the ciphertext. We may subtract off (e, f) to get (c, d).

19. As is Section 2.11, set up the matrix equation

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \end{pmatrix} \equiv \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}.$$

This yields $c_0 = 1, c_1 = 0, c_2 = 1$, so the recurrence is $k_{n+3} \equiv k_n + k_{n+2}$. The next four terms of the sequence are 1, 0, 0, 1.

20. The sequence is 1,0,1,0,1,0,1,... . The matrix equation is $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \end{pmatrix} \equiv \begin{pmatrix} 1 \\ 0 \end{pmatrix}$. This yields $c_0 = 1, c_1 = 0$, so $k_{n+2} \equiv k_n$.

21. Set up the matrix equation

$$\begin{pmatrix} x_n & x_{n+1} \\ x_{n+1} & x_{n+2} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \end{pmatrix} = \begin{pmatrix} x_{n+2} \\ x_{n+3} \end{pmatrix}.$$

Using the values provided, we obtain

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 2 \end{pmatrix}.$$

The inverse of the matrix can be found to be

$$-\begin{pmatrix} 0 & -1 \\ -1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix} \pmod 3$$

Multiplying both sides of by the inverse matrix, yields $c_0 = 2$ and $c_1 = 1$.

22. Use x_1, x_2 and x_3 to solve for c_1 by obtaining $c_1 + 2 \equiv 1 \pmod 5$. Thus, $c_1 = 4$. Next, use x_2, x_3 and x_4 to solve for c_0. We get $c_0 + c_1 + 2 \pmod 5 \equiv 0$, and hence $c_0 = 4$.

23. The number of seconds in 120 years is

$$60 \times 60 \times 24 \times 365 \times 120 \approx 3.8 \times 10^9.$$

Therefore you need to count $10^{100}/(3.8 \times 10^9) \approx 2.6 \times 10^90$ numbers per second!

24. (a) The ciphertext will consist of one letter repeated. However, there is no way of deducing what the key is.

(b) The ciphertext will consist of one letter repeated. However, there is no way of deducing what the key is.

(c) The ciphertext will consist of a continuous stream of the letter A. This is easy to detect. However, there will be no way to tell what the key is.

25. (a) The ciphertext will correspond to a shifted version of the key word that is repeated many times. The periodic nature of the resulting ciphertext will cause Eve to suspect the plaintext is a single letter, while the period of the repeating ciphertext will correspond to the key length.

(b) Using the fact that no English word of length six is the shift of another English word, simply treat the Vigenere key as if it were the plaintext and the

single character plaintext as if it were the shift in a shift cipher. Decrypting can be done by trying all possible shifts of the first six characters of the ciphertext. One of these shifts will be a word that corresponds to the Vigenere key.

(c) If we use the method of displacement, then shifting by six will have the highest number of matches. In fact, every place will match up. This will be easy to detect. However, shifting the ciphertext by one place will just yield the amount of matches that occur when the repeated key is shifted by one place. In particular, the key word will most likely not have that many matches with itself when shifted over one place. Similarly for shifts of two, three, four, and five. As a result, other shifts will have a much smaller amount of matches.

Chapter 3 - Exercises

1. (a) Apply the Euclidean algorithm to 17 and 101:

$$101 = 5 \cdot 17 + 16$$

$$17 = 1 \cdot 16 + 1.$$

Working back yields $1 = 17 - 16 = 17 - (101 - 5 \cdot 17) = (-1) \cdot 101 + 6 \cdot 17$.

(b) Since $-101+6\cdot 17 = 1$, we have $6\cdot 17 \equiv 1 \pmod{101}$. Therefore $17^{-1} \equiv 6 \pmod{101}$.

2. (a) Apply the Euclidean algorithm to 7 and 30:

$$30 = 4 \cdot 7 + 2$$

$$7 = 3 \cdot 2 + 1.$$

Working backwards yields $1 = 7 - 3 \cdot 2 = 7 - 3 \cdot (30 - 4 \cdot 7) = 13 \cdot 7 + (-3) \cdot 30$. Therefore $13 \cdot 7 \equiv 1 \pmod{30}$, so $d = 13$.

(b) Let $c \equiv m^7 \pmod{31}$ be the ciphertext. Claim: $c^{13} \equiv m \pmod{31}$. Proof: $c^13 \equiv (m^7)^{13} \equiv m^{91} \equiv (m^{30})^3 m$. If $m \not\equiv 0 \pmod{31}$ then $m^{30} \equiv 1 \pmod{31}$ by Fermat. Then $c^{13} \equiv 1^3 m \equiv m$. If $m \equiv 0 \pmod{31}$, then $c \equiv m^7 \equiv 0$, so $c^{13} \equiv 0^{13} \equiv 0 \equiv m$. Therefore $c^{13} \equiv m$ for all m. Therefore decryption is performed by raising the ciphertext to the 13th power mod 31.

3. 3. (a) $\gcd(12, 236) = 4$, so divide both sides by 4 to obtain $3x \equiv 7 \pmod{59}$. The inverse of 3 mod 59 is 20, so multiply both sides by 20 to obtain $x \equiv 140 \equiv 22 \pmod{59}$. This yields $x \equiv 22, 81, 140, 199 \pmod{236}$.
(b) 30 is not divisible by $4 = \gcd(12, 236)$, so there are no solutions.

4. (a)

$$\begin{aligned} 30030 &= 116 \cdot 257 + 218 \\ 257 &= 1 \cdot 218 + 39 \\ 218 &= 5 \cdot 39 + 23 \\ 39 &= 1 \cdot 23 + 16 \\ 23 &= 1 \cdot 16 + 7 \\ 16 &= 2 \cdot 7 + 2 \\ 7 &= 3 \cdot 2 + 1 \\ 2 &= 2 \cdot 1 + 0. \end{aligned}$$

Therefore, $\gcd(30030, 257) = 1$.
(b) If 257 is composite, it is divisible by a prime $p \leq \sqrt{257} = 16.03\ldots$. The primes satisfying this are exactly the prime factors of 30030. Since the gcd is 1, none of them divide 257, so 257 is prime.

5. (a)

$$\begin{aligned} 4883 &= 1 \cdot 4369 + 514 \\ 4369 &= 8\dot{5}14 + 257 \\ 514 &= 2 \cdot 257 + 0. \end{aligned}$$

Therefore, the gcd is 257.
(b) We know that both numbers have 257 as a factor. This yields $4883 = 257 \cdot 19$ and $4369 = 257 \cdot 17$.

6. (a) The first two steps of the Euclidean algorithm are

$$\begin{aligned} F_n &= 1 \cdot F_{n-1} + F_{n-2} \\ F_{n-1} &= 1 \cdot F_{n-2} + F_{n-3}. \end{aligned}$$

It continues in this way until

$$\begin{aligned} 2 &= 2 \cdot 1 + 1 \\ 1 &= 1 \cdot 1 + 0. \end{aligned}$$

Therefore, the gcd is 1.
(b)

$$\begin{aligned} 11111111 &= 1000 \cdot 11111 + 111 \\ 11111 &= 100 \cdot 111 + 11 \\ 111 &= 10 \cdot 11 + 1 \\ 11 = 11 \cdot 1 + 0. & \end{aligned}$$

Therefore, the gcd is 1.
(c) The first step of the Euclidean algorithm is

$$a = 10^{F_{n-2}} \cdot b + c,$$

where c consists of F_{n-2} repeated 1's. Continuing in this way, in each step we divide F_{j-1} repeated 1's into F_j repeated 1's and get a remainder consisting of F_{j-2} repeated 1's. Eventually, we get down to the computations of part (b), and then obtain that the gcd is 1.

7. (a) If $ab \equiv 0 \pmod{p}$, then $p|ab$. By the Corollary on page 64, since p is prime, either $p|a$ or $p|b$. Therefore, either $a \equiv 0 \pmod{p}$ or $b \equiv 0 \pmod{p}$.

(b) We follow the proof of the Corollary on page 64. Since $\gcd(a, n) = 1$, there are integers x, y such that $ax+ny = 1$. Multiply by b to obtain $abx+bny = b$. Since $n|ab$, both terms on the left are multiples of n. Therefore $n|b$.

8. $(x+1)(x-1) \equiv 0 \pmod{p}$ implies, by 3(a), that either $x+1 \equiv 0 \pmod{p}$ or $x - 1 \equiv 0 \pmod{p}$. Therefore $x \equiv \pm 1 \pmod{p}$.

9. One solution is to look at the numbers congruent to 3 (mod 10) until we find one that is 2 (mod 7): 3, $13 \equiv 6$, $23 \equiv 2 \pmod 7$. Therefore $x \equiv 23 \pmod{70}$.

10. Suppose there are x people. Then $x \equiv 1 \pmod 3, x \equiv 2 \pmod 4, x \equiv 3 \pmod 5$. The last two congruences combine to $x \equiv 18 \pmod{20}$. List the numbers that are $18 \pmod{20}$ until you find one that is $1 \pmod 3$. The answer is $x \equiv 58 \pmod{60}$.

11. If $a \not\equiv 0 \pmod p$, then Fermat says that $a^{p-1} \equiv 1 \pmod p$. Multiply by a to get $a^p \equiv a \pmod p$. If $a \equiv 0 \pmod p$, then $a^p \equiv 0^p \equiv 0 \equiv a \pmod{}$. Therefore $a^p \equiv a \pmod p$ for all a.

12. By Fermat's theorem, $2^{100} \equiv 1 \pmod{101}$. Therefore, $2^{10203} \equiv (2^{100})^{102}2^3 \equiv 1^{102}2^3 \equiv 8$. Therefore, the remainder is 8.

13. "Last two digits" means we work mod 100. Since $\phi(100) = 40$, Euler's theorem says that $123^{40} \equiv 1 \pmod{100}$. Therefore, $123^{562} \equiv (123^{40})^{14}123^2 \equiv 123^2 \equiv 23^2 \equiv 529 \equiv 29$. The last two digits are 29.

14. (a) $7^7 \equiv (-1)^7 \equiv -1 \equiv 3 \pmod 4$.
(b) $7^7 = 3 + 4k$ for some k. By Euler's theorem, $7^4 \equiv 1 \pmod{10}$. Therefore,

$$7^{7^7} = 7^3(7^4)^k \equiv 7^3 \cdot 1^k \equiv 343 \equiv 3 \pmod{10}.$$

The last digit is 3.

15. (a) $\phi(1) = 1, \phi(2) = 1, \phi(5) = 4, \phi(10) = 4$. The sum is 10.
(b) $\phi(1) = 1, \phi(2) = 1, \phi(3)2, \phi(4) = 2, \phi(6) = 2, \phi(12) = 4$. The sum is 12.
(c) The sum of $\phi(d)$, for all of the divisors d of n, equals n.

16. (a) Since $p \nmid a$, Fermat says that $a^{p-1} \equiv 1 \pmod p$. For $p = 7$, we have $a^6 \equiv 1 \pmod 7$, so $a^1728 \equiv (a^6)^{288} \equiv 1^{288} \equiv 1 \pmod 7$. Since $1728 = 12 \cdot 144$ and $1728 = 18 \cdot 96$, a similar argument works for $p = 13$ and for $p = 19$.

(b) If $p \nmid a$, then multiply the result of (a) by a to get $a^{1729} \equiv a \pmod p$. If $p|a$, then a^{1729} and a are both $0 \pmod p$, so $a^{1729} \equiv a$ in this case, too.

(c) Fix a number a. The Chinese Remainder Theorem says that $x \equiv a^{1729} \pmod 7, x \equiv a^1729 \pmod{13}, x \equiv a^{1729} \pmod{19}$ has a unique solution $x \pmod{1729}$, since $1729 = 7 \cdot 13 \cdot 19$. We know two such solutions: $x = a$ (from part (b) and $x = a^1729$ (trivially). Since x is unique mod 1729, we must have $a \equiv a^{1729} \pmod{1729}$.

17. (a) The powers of 2 mod 11 are $2, 4, 8, 5, 10, 9, 7, 3, 6, 1$. This gives all nonzero congruence classes mod 11, so 2 is a primitive root mod 11.
(b) The inverse of $3 \pmod{10}$ is 7. We obtain

$$8^7 \equiv (2^3)^7 \equiv 2^{21} \equiv (2^{10} \cdot 2^1 \equiv 1 \cdot 2 \equiv 2 \pmod{11}.$$

Therefore, $x = 7$.
(c) This can be done directly, but here is another way. If $c \not\equiv 0 \pmod{11}$, then $c \equiv 2^j$ for some j. Therefore, $c \equiv (8^7)^j \equiv 8^{7j} \pmod{11}$, so c is a power of 8.
(d) Write $xy = 1 + (p-1)k$. Then

$$h^x \equiv (g^y)^x \equiv g \cdot (g^{p-1})^k \equiv g \cdot 1^k \equiv g \pmod p.$$

(e) Let c be nonzero mod p. Then $c \equiv g^j \pmod{p}$ for some j, so $c \equiv (h^x)^j \equiv h^{xj} \pmod{p}$. Since every nonzero congruence class is a power of h, we have that h is a primitive root mod p.

18. (a) The determinant is $1 \cdot 1 - 1 \cdot 6 = -5 \equiv 21 \pmod{26}$. The inverse of the determinant is $5 \pmod{26}$. The inverse of the matrix is therefore

$$\frac{1}{21}\begin{pmatrix} 1 & -1 \\ -6 & 1 \end{pmatrix} \equiv 5\begin{pmatrix} 1 & -1 \\ -6 & 1 \end{pmatrix} \equiv \begin{pmatrix} 5 & 21 \\ 22 & 5 \end{pmatrix}.$$

(b) The determinant is $1 - b$. The matrix is invertible mod 26 exactly when $\gcd(1-b, 26) = 1$. This happens when $1 - b$ is odd and not 0 mod 13, so $b \equiv 0, 2, 4, 6, 8, 10, 12, 16, 18, 20, 22, 24 \pmod{26}$.

19. The determinant is $9 - 35 = -26$. This is divisible by 2 and by 13, so these are the two primes for which the matrix is not invertible mod p.

20. (a) We know $a^{\phi(n)} \equiv 1 \pmod{n}$, by Euler. Since r is smallest, $r \leq \phi(n)$.

(b) Since $a^r \equiv 1$, we have $a^m \equiv (a^r)^k \equiv 1^k \equiv 1 \pmod{n}$.

(c) By (b), $a^{qr} \equiv 1$. Therefore, $1 \equiv$ (by assumption)$a^t \equiv a^{qr}a^s \equiv 1 \cdot a^s \equiv a^s$.

(d) Since $a^s \equiv 1$ and r is the smallest positive integer with $a^r \equiv 1$, we must have $s = 0$. Therefore $t = qr$, so $r|t$.

(e) By Euler, $a^{\phi(n)} \equiv 1 \pmod{n}$. From part (d) with $t = \phi(n)$, we obtain $\mathrm{ord}_n(a)|\phi(n)$.

21. (a) If r divides 600, then $r = 2^a \cdot 3^b \cdot 5^c$ with $a \leq 3, b \leq 1, c \leq 2$. If $r < 600$, then we cannot have $a = 3, b = 1, c = 2$. If $a \leq 2$, then $r|2^2 \cdot 3 \cdot 5^2 = 300$. If $b = 0$ then $r|2^3 \cdot 5^2 = 200$. If $c \leq 1$ then $r|2^3 \cdot 3 \cdot 5 = 120$.

(b) From 9(e), we know that $\mathrm{ord}_{601}(7)|600$. If $\mathrm{ord}_{601}(7) < 600$, then it divides 300, 200, or 120, by (a).

(c) Suppose $\mathrm{ord}_{601}(7)|300$. By 9(b), we must have $7^{300} \equiv 1 \pmod{601}$, which is not the case. Therefore, $\mathrm{ord}_{601}(7) \nmid 300$. Similarly, $\mathrm{ord}_{601}(7) \nmid 200$ and $\mathrm{ord}_{601}(7) \nmid 120$.

(d) From (b) and (c), we cannot have $\mathrm{ord}_{601}(7) < 600$. Since $\mathrm{ord}_{601}(7) \leq 600$ from 9(a), we must have $\mathrm{ord}_{601}(7) = 600$. Therefore the numbers $7^j \pmod{601}$, for $j = 0, 1, \ldots, 599$ are distinct, so there are 600 of them. This implies that 7 is a primitive root mod 601.

(e) Calculate $g^{(p-1)/q_i} \pmod{p}$ for $i = 1, 2, \ldots, s$. If each of these is $\not\equiv 1 \pmod{p}$, then g is a primitive root mod p. (Of course, perhaps we should check first that $p \nmid g$.)

22. (a) We have $3^{16k} \equiv 2^k \not\equiv 1 \pmod{65537}$ and $3^{32k} \equiv 2^{32} \equiv 1 \pmod{65537}$. Therefore $65536 \nmid 16k$ and $65536|32k$. Write $32k = 65536\ell$ for some ℓ. Divide by 32 to obtain $k = 2048\ell$, so $2^{11} = 2048|k$. If $2^{12} = 4096|k$, then $16k$ is a multiple of $16 \cdot 4096 = 65536$, which we showed doesn't happen. Therefore k is a multiple of 2048, but is not a multiple of 4096.

(b) From (a), we see that k is an odd multiple of 2048. We also know that $0 \leq k < 65536$, since every nonzero number mod 65537 can be written as a power of 3 with exponent in this range. There are 65536/2048=32 multiples of 2048 in this range. Of these, 16 are multiples of 4096. The remaining 16 are possibilities for k. We now calculate $3^{2048m} \pmod{65537}$ for $m = 1, 3, 5, \cdots$. We find (with the help of a computer) that $m = 27$ works. So $k = 2048 * 27 = 55296$.

23. (a) We claim that after the kth step 2, we have $r_k \equiv y^{b_1 b_2 \cdots b_k} \pmod{n}$. This is easily seen to be the case for $k = 1$. Assume it is true for $k-1$. We'll show it's true for k. We have $s_k \equiv r_{k-1}^2 \equiv y^{2 \cdot b_1 b_2 \cdots b_{k-1}}$. Then $r_k \equiv s_k y^{b_k} \equiv r_k^2 y^{b_k} \equiv y^{2 \cdot b_1 \ldots b_{k-1} + b_k} \equiv y^{b_1 \cdots b_{k-1} b_k}$. Therefore, when $k = w$ we have $r_w \equiv y^x$, as desired.

(b) Write $x = b_1 \ldots b_w$ in binary as in (a). We assume $b_1 = 1$. The algorithm is easily seen to work when $x = 0$, so we may assume $w \geq 1$. We claim that after step 2, $a = b_1 \ldots b_{w-k}$, $b \equiv y^{b_{w-k+1} \cdots b_w}$ and $c \equiv y^{2^k}$ for some value of k. When we start, we have $k = 0$.

Suppose we arrive at step 2 with $a = b_1 \ldots b_{w-k}$, $b \equiv y^{b_{w-k+1} \ldots b_w}$, and $c \equiv y^{2^k}$. If a is odd, then the output of step 2 is the same as the input, hence of the desired form. If a is even, then $b_{w-k} = 0$. We obtain the new $a = b_1 \ldots b_{w-k-1}$, $b \equiv y^{b_{w-k} b_{w-k+1} \ldots b_w}$ (we may include the extra bit at the beginning since it is 0), and $c \equiv y^{2^{k+1}}$. Therefore the output has a, b, c in the desired form with $k+1$ in place of k.

Now let's look at what happens in step 3. The output of step 2 is of the form $a = b_1 \ldots b_{w-j}$, $b \equiv y^{b_{w-j+1} \cdots b_w}$ and $c \equiv y^{2^j}$ for some value of j. If a is even, step 3 does nothing, so the output still has the desired form. If a is odd, then the last bit b_{w-j} of a is 1. The new a is $a = b_1 \ldots b_{w-j-1} 0$. Also, the new b is $y^{b_{w-j} b_{w-j+1} \ldots b_w} \cdot y^{2^j} \equiv y^{1 b_{w-j} b_{w-j+1} \ldots b_w} \equiv y^{b_{w-j} b_{w-j+1} \ldots b_w}$. The new c is still y^{2^j}.

If the new $a = 0$, then $j = w - 1$, so $b \equiv y^x$. Therefore step 5 outputs y^x, as desired. Otherwise, step 4 sends us to step 2, which outputs $a = b_1 \ldots b_{w-j-1}$, $b \equiv y^{b_{w-j} b_{w-j+1} \ldots b_w}$, and $c \equiv y^{2^{j+1}}$. This is of the desired form with $k = j + 1$.

Therefore, the output of step 2 always has the desired form, as claimed.

Since a gets smaller at each application of steps 2 and 3, eventually $a = 0$ and the algorithm stops. As pointed out above, the output of step 5 is then $y^x \pmod{n}$.

24. Since $z_j \equiv 0 \pmod{m_i}$ when $j \neq i$, we have $x \equiv a_i y_i z_i \equiv a_i z_i^{-1} z_i \equiv a_i \pmod{m_i}$.

25. (a) Solve $x^2 \equiv 133 \pmod{11}$ to obtain $x \equiv \pm 1 \pmod{11}$. Now solve $x^2 \equiv 133 \pmod{13}$ to obtain $x \equiv \pm 4 \pmod{13}$. There are four ways to combine these via the Chinese Remainder Theorem: $x \equiv 43, 56, 87, 100 \pmod{143}$.

(b) This reduces to $x^2 \equiv 77 \pmod{11}$ and $x^2 \equiv 77 \pmod{13}$. The solutions satisfy $x \equiv 0 \pmod{11}$ and $x \equiv \pm 5 \pmod{13}$. These combine to yield $x \equiv 44, 99 \pmod{143}$.

26. Suppose $x^2 \equiv -1 \pmod{p}$. Raise both sides to the $(p-1)/2$ power to obtain $x^{p-1} \equiv (-1)^{(p-1)/2} \pmod{p}$. By Fermat, $x^{p-1} \equiv 1$. Since $p \equiv 3] pmod 4$, the exponent $(p-1)/2$ is odd. Therefore $(-1)^{(p-1)/2} = -1$. This yields $1 \equiv -1 \pmod{p}$, which is a contradiction. Therefore x cannot exist.

27. (a) There are at most 4 square roots of $x \pmod{n}$. Therefore, several random selections of these square roots should include the meaningful message m.

(b) Being able to find square roots mod n is computationally equivalent to factoring n, which is presumed to be hard.

(c) Eve chooses a random number m and computes $x \equiv m^2 \pmod{n}$. She gives x to the machine, which outputs a square root m' of x. If $\gcd(m,n) = 1$, there are four possible m', namely m, $-m$, and two others. After a few tries, Eve should obtain m' with $m' \not\equiv \pm m \pmod{n}$. Since $m^2 \equiv x \equiv m'2 \pmod{n}$, a nontrivial factor of n is given by $\gcd(m - m', n)$.

28. (a) Since $r_1 = a - bq_1$ and $d|a, b$, we have $d|r_1$. Since $r_2 = b - q_2 r_1$ and $d|b, r_1$, we have $d|r_2$.

(b) Suppose $d|r_1, \ldots, r_j$. Since $r_{j+1} = r_{j-1} - q_{j+1} r_j$ and $d|rj - 1, r_j$, we have $d|r_{j+1}$. By induction, we have $d|r_i$ for all i.

(c) Since $r_{k-1} = q_{k+1} r_k$, we have $r_k | r_{k-1}$. Assume $r_k | r_{k-i}$ for $i = 1, 2, \ldots, j$. Since $r_{k-j-1} = q_{k-j+1} r_{k-j} + r_{k-j+1}$ and $r_k | r_{k-j}, r_{k-j+1}$ by assumption, we have $r_k | r_{k-j-1}$. By induction, $r_k | r_i$ for all i.

(d) Since $b = q_2 r_1 + r_2$ and $r_k | r_1, r_2$, we have $r_k | b$. Since $a = q_1 b + r_1$ and $r_k | a, r_1$, we have $r | a$.

(e) Since $d|r_k$ for each common divisor d, we have $r_k \geq d$ for all common divisors d. Since r_k is a common divisor, it is the largest.

29. (a)

$$\left(\frac{123}{401}\right) = \left(\frac{401}{123}\right) = \left(\frac{32}{123}\right) = \left(\frac{2}{123}\right)^5 = -1.$$

Therefore, there is no solution.

(b)

$$\left(\frac{43}{179}\right) = -\left(\frac{179}{43}\right) = -\left(\frac{7}{43}\right) = \left(\frac{43}{7}\right) = \left(\frac{1}{43}\right) = 1.$$

Therefore, there is a solution.

(c)

$$\left(\frac{1093}{65537}\right) = \left(\frac{65537}{1093}\right) = \left(\frac{2}{1093}\right)\left(\frac{525}{1093}\right) = -\left(\frac{43}{525}\right) = -\left(\frac{9}{43}\right) = -1.$$

Therefore, there is no solution.

30. (a) If $a \equiv b^2 \pmod{n}$, then

$$\left(\frac{a}{n}\right) = \left(\frac{b}{n}\right)^2 = (\pm 1)^2 = 1.$$

Therefore, if $\left(\frac{a}{n}\right) = -1$, then a cannot be a square mod n.

(b)

$$\left(\frac{3}{35}\right) = -\left(\frac{35}{3}\right) = -\left(\frac{2}{3}\right) = 1.$$

(c) Since 3 is not a square mod 5, it cannot be a square mod 35.

31. $\left(\frac{2}{15}\right) = 1$, but $2^7 \equiv 8 \pmod{15}$.

32. (a)

$$\left(\frac{3}{65537}\right) = \left(\frac{65537}{3}\right) = \left(\frac{2}{3}\right) = -1.$$

(b) Since 65537 is prime, $3^{(65537-1)/2} \equiv \left(\frac{3}{65537}\right) = -1 \pmod{65537}$.
(c) The order of 3 mod 65537 divides $65536 = 2^{16}$, hence is a power of 2. If the order is not 2^{16}, then it divides $2^{15} = 32768$, so $3^{32768} \equiv 1 \pmod{65537}$. But part (b) says that this is not the case. Therefore, the order of 3 must be 2^{16}, and 3 is a primitive root mod 65537.

33. (a) The only polynomials of degree 1 are X and $X+1$, and they are irreducible. The only polynomials of degree 2 are $X^2, X^2+1, X^2+X, X^2+X+1$. But X^2 and $X^2+1 \equiv (X+1)^2$ are reducible, and so is $X^2+X \equiv X(X+1)$. Only X^2+X+1 remains. If it factors, it must be divisible by a degree one polynomial. Clearly X does not divide it. A simple calculation shows that $X+1$ does not divide it either. Therefore X^2+X+1 is irreducible.

(b) If X^4+X+1 factors, it must have an irreducible factor of degree at most 2. Since none of the polynomials from part (a) divide it, it must be irreducible.

(c) $X^4 \equiv -(X+1) \equiv X+1$, since we are working with coefficients mod 2. Square both sides to obtain $X^8 \equiv (X+1)^2 \equiv X^2+1$. Square again to obtain $X^{16} \equiv (X^2+1)^2 \equiv X^4+1 \equiv (X+1)+1 \equiv X$.

(d) Since X^4+X+1 is irreducible, polynomials mod X^4+X+1 form a field. Since $X \not\equiv 0 \pmod{X^4+X+1}$, it has a multiplicative inverse. Therefore, we can divide $X^{16} \equiv X$ by X to obtain $X^{15} \equiv 1$.

34. (a) If X^2+1 is reducible, it must factor as a product of two degree one polynomials. But none of the polynomials X, $X+1$, $X+2$ divides X^2+1 mod 3. Therefore it is irreducible.

(b) Apply the Euclidean algorithm to $2X+1$ and X^2+1:

$$X^2+1 = (2X)(2X+1) + (X+1)$$

$$2X+1 = (2)(X+1) + 2$$

$$X+1 = (2X+2)(2) + 0.$$

Working backwards, we obtain $2 = (2X+1) - 2(X+1) = (2X+1) - 2((X^2+1) - (2X)(2X+1)) = (-2)(X^2+1) + (X+1)(2X+1)$. Therefore, $(X+1)(2X+1) \equiv 2 \pmod{X^2+1}$. Multiply by 2 to obtain $(2X+2)(2X+1) \equiv 1 \pmod{X^2+1}$. Therefore, $2X+2$ is the multiplicative inverse of $1+2X$.

35. $a = q_1 b + r_1$ with $0 \le r_1 < b$. This means that

$$\frac{a}{b} = q_1 + \frac{r_1}{b}$$

with $0 \le r_1/b < 1$. Therefore, $a_0 = q_1$. Similarly, at each step of the algorithm, in the notation of page 67, we have

$$\frac{r_{j-2}}{r_{j-1}} = q_j + \frac{r_j}{r_{j-1}},$$

which yields $a_{j-1} = q_j$.

36. (a) The values of $a_0, a_1, a_2, \ldots$ for $\sqrt{3}$ are $1, 1, 2, 1, 2, 1, 2, \cdots$. For $\sqrt{7}$, they are $2, 1, 1, 1, 4, 1, 1, 1, 4, \cdots$. The first keeps repeating $1, 2$. The second keeps repeating $1, 1, 1, 4$.

(b) For $d = 3$: $n = 1$, and $p_1/q_1 = 1 + 1/1 = 2/1$. We have $2^2 - 3 \cdot 1^1 = 1$. For $d = 7$: $n = 3$, and

$$\frac{p_3}{q_3} = 2 + \frac{1}{1 + \frac{1}{1+\frac{1}{1}}} = \frac{8}{3}.$$

We have $8^2 - 7 \cdot 3^2 = 1$.
(c) The continued fraction for $\sqrt{19}$ starts with $a_0, a_1, a_2, a_3, a_4, a_5, a_6$ equal to $4, 2, 1, 3, 1, 2, 8$. We have $n = 5$. A calculation yields $p_5 = 170$ and $q_5 = 39$. We have $170^2 - 19 \cdot 39^2 = 1$. (Note: this method sometimes yields $x^2 - dy^2 = -1$. In this case $x_1 = x^2 + dy^2$ and $y_1 = 2xy$ satisfy $x_1^2 - dy_1^2 = +1$.)

37. Use a decimal approximation for e to obtain

$$a_0, a_1, a_2, a_3, \cdots = 2, 1, 2, 1, 1, 4, 1, 1, 6, 1, 1, 8, 1, 1, 10, 1, 1, 12, 1, 1, 14, \ldots.$$

After the initial 2, we get blocks of $1, 2n, 1$ for $n = 1, 2, 3, \ldots$.

38. The continued fraction has $a_0, a_1, a_2, \cdots$ equal to $1, 1, 1, \cdots$. We have $p_n = F_{n+2}$ and $q_n = F_{n+1}$, where F_n is the nth Fibonacci number (see Exercise 6).

39. (a) The multiples of p are $p, 2p, 3p, \cdots, (q-1)p$. There are $q - 1$ of them. Similarly for the multiples of q.
(b) The only factors of pq are $1, p, q, pq$. If $\gcd(m, pq) > 1$, then the gcd must be p, q, or pq. Therefore, m is a multiple of p, q, or pq, hence a multiple of p or of q (possibly both).
(c) If m is a multiple of both p and q, then it is a multiple of pq, hence $m \geq pq$. If $1 \leq m < pq$, this is impossible.
(d) We have pq numbers m with $1 \leq m \leq pq$. Remove $m = pq$, which is the only number satisfying $\gcd(m.pq) = pq$. Remove the $q - 1$ multiples of p and the $p - 1$ multiples of q. By part (c), these two sets of numbers do not overlap. We have therefore removed $1 + (q-1) + (p-1)$ numbers. There are $pq - 1 - (q-1) - (p-1) = (p-1)(q-1)$ numbers remaining. These are exactly the m with $\gcd(m, pq) = 1$.

40. (a) $x \equiv 1 \pmod 4$, $x \equiv 2 \pmod 6$ has no solution.
(b) $x \equiv 2 \pmod 4$, $x \equiv 4 \pmod 6$ has the solution $x = 10$ (in fact, $x \equiv 10 \pmod{12}$ gives all solutions).

Chapter 4 - Exercises

1. (a) Switch left and right halves and use the same procedure as encryption. The switch the left and right of the final output. Verification is the same as that on pages 115-116.

(b, c) 1st round: $M_0M_1 \to M_1[M_0 \oplus K \oplus M_1]$

2nd round: $[M_0 \oplus K \oplus M_1][M_1 \oplus M_0 \oplus K \oplus M_1 \oplus K] = [M_0 \oplus K \oplus M_1][M_0]$

3rd round: $[M_0][M_0 \oplus K \oplus M_1 \oplus K \oplus M_0] = [M_0][M_1]$, which is the plaintext.

Therefore 3 rounds is very insecure! After 2 rounds, the ciphertext alone lets you determine M_0 and therefore $M_1 \oplus K$, but not M_1 or K individually. If you also know the plaintext, you know M_1 are therefore can deduce K.

2. If someone discovers the fixed key and obtains the encrypted password file, this person can easily decrypt by the usual decryption procedure. However, knowing the ciphertext and the plaintext does not readily allow one to deduce the key.

3. CBC: We have $D_K(C_j) \oplus C_{j-1} = D_K(E_K(P_j \oplus C_{j-1})) \oplus C_{j-1} = P_j \oplus C_{j-1} \oplus C_{j-1} = P_j$.

CFB: $C_j \oplus L_8(E_K(X_j)) = (P_j \oplus L_8(E_K(X_j))) \oplus L_8(E_K(X_j)) = P_j$.

4. Let I denote the string of all 1's. Note that the expansion $E(\overline{R_{i-1}}) = \overline{E(R_{i-1})} = E(R_{i-1}) \oplus I$. Therefore $E(\overline{R_{i-1}}) \oplus \overline{K_i} = E(R_{i-1}) \oplus I \oplus K_i \oplus I = E(R_{i-1}) \oplus K_i$, so the input to the S-boxes doesn't change. Therefore the output doesn't change. But $\overline{L_{i-1}} = L_{i-1} \oplus I$, so the resulting right side is $\overline{L_{i-1}} \oplus f(R_{i-1}, K_i) = R_i \oplus I = \overline{R_i}$. Also, clearly the new left side is the complementary string. So each round of DES gives the complementary string, so this is true for the final result.

5. (a) The keys $K_1, \ldots, K_{16}$ are all the same (all 1's). Decryption is accomplished by reversing the order of the keys to $K_{16}, \ldots, K_1$. Since the K_i are all the same, this is the same as encryption, so encrypting twice gives back the plaintext.

(b) The key of all 0's , by the same reasoning.

6. Let (m, c) be a plaintext-ciphertext pair. Make one list of $E_K(E_K(m))$, where K runs through all possible keys. Make another list of $D_{K'}(c)$, where K' runs through all possible keys. A match between the two lists is a pair K, K' of keys with $E_{K'}(E_K(E_K(m))) = c$. There should be a small number of such pairs. For each such pair, try it on another plaintext and see if it produces the corresponding ciphertext. This should eliminate most of the incorrect pairs.

Repeating a few more times should yield the pair K_1, K_2.

7. (a) To perform the meet in the middle attack, you need a plaintext m and ciphertext c pair (its a known plaintext attack). So, make two lists. The left list consists of encryptions using the second encryption E^2 with different choices for K_2. Similarly, the right side contains decryptions using different keys for the first encryption algorithm. Thus, the lists look like:

$$\begin{array}{ll} E_1^2(m) = y_1 & z_1 = D_1^1(c) \\ E_2^2(m) = y_2 & z_2 = D_2^1(c) \\ \vdots & \vdots \\ E_{788}^2(m) = y_{788} & z_{788} = D_{788}^1(c) \\ \vdots & \vdots \end{array}$$

Note: The two lists need not be the same size, as the different algorithms might have different key lengths, and hence different amount of keys (see part b). Now, look for matches between y_j and z_l. A match using K_2' for E^2 and K_1' for D^1 indicates

$$E_{K_2'}^2(m) = y = D_{K_1'}^1(c)$$

and hence

$$E_{K_1'}^1\left(E_{K_2'}^2(m)\right) = c.$$

(b) Observe that there are 26 possibilities for β and 12 possibilities for α. Let $E_\alpha^2(x) = \alpha x \pmod{26}$ and let $E_\beta^1(x) = x + \beta \pmod{26}$. The composition of these two gives the affine cipher. The total computation needed involves producing 26 encryptions for E^2 and 12 decryptions for E^1. The total is 38.

8. It suffices to look at an arbitrary round of the encryption process. Suppose we look at the ith round, which involves $L_i = R_{i-1}$, $M_i = L_{i-1}$ and $R_i = f(K_i, R_{i-1}) \oplus M_{i-1}$. To undo this round, that is to go from $\{L_i, M_i, R_i\}$ to $\{L_{i-1}, M_{i-1}, R_{i-1}\}$ we do the following:

$$L_{i-1} = M_i$$

$$M_{i-1} = f(K_i, R_{i-1}) \oplus R_i = f(K_i, L_i) \oplus R_i$$

$$R_{i-1} = L_i.$$

This is of the form of the decryption algorithm specified in the problem.

9. (a) At the decryption side, the decryptor has $\{C_1, C_2, \cdots\}$ and the initial X_1. To decrypt, the decryptor starts with $j = 1$ and calculates

$$P_j = C_j \oplus L_{32}(E_K(X_j))$$

$$X_{j+1} = R_{32}(X_j) \| C_j.$$

(b) To solve this problem, it is easiest to step through registers step by step. We start with X_1 and a sequence of ciphtertext $\tilde{C}_1, C_2, C_3, \cdots$. To decrypt the first block, we calculate:

$$\tilde{P}_1 = \tilde{C}_1 \oplus L_{32}(E_K(X_1))$$

$$\tilde{X}_2 = R_{32}(X_1)\|\tilde{C}_1.$$

Observe that the decrypted plaintext $\tilde{P}_1$ is corrupted because it has the corrupted $\tilde{C}_1$ as part of it, and also that $\tilde{X}_2 \neq X_2$ since it has the corrupted $\tilde{C}_1$ as part of it. The next couple steps of decryption proceed as

$$\tilde{P}_2 = C_2 \oplus L_{32}(E_K(\tilde{X}_2))$$

$$\tilde{X}_3 = R_{32}(\tilde{X}_2)\|C_2 = \tilde{C}_1\|C_2$$

$$\tilde{P}_3 = C_3 \oplus L_{32}(E_K(\tilde{X}_3))$$

$$X_4 = R_{32}(\tilde{X}_3)\|C_3 = C_2\|C_3.$$

At this point, we have three corrupted plaintexts $\tilde{P}_1$, $\tilde{P}_2$, and $\tilde{P}_3$. However, note that by the end of the third round, the register X_4 is no longer corrupted. The subsequent decryption step is

$$P_4 = C_4 \oplus L_{32}(E_K(X_4))$$

$$X_5 = R_{32}(X_4)\|C_4.$$

Thus, the fourth step of decryption is uncorrupted. All subsequent decryption steps also will be free of errors.

10. In CBC, suppose that an error occurs (perhaps during transmission) in block C_j to produce the corrupted $\tilde{C}_j$, and that the subsequent blocks C_{j+1} and C_{j+2} are ok.

Now start decrypting. If we try to decrypt to get P_j we get

$$\tilde{P}_j = D_K(\tilde{C}_j) \oplus C_{j-1}$$

which is corrupted because the decryption of $\tilde{C}_j$ will be junk. Next, try to decrypt to get P_{j+1}:

$$\tilde{P}_{j+1} = D_K(C_{j+1}) \oplus \tilde{C}_j$$

which, although $D_K(C_{j+1})$ is correct, when we add the corrupted $\tilde{C}_j$ we get a corrupted answer. Now proceed to try to decrypt C_{j+2} to get P_{j+2}:

$$P_{j+2} = D_K(C_{j+2}) \oplus C_{j+1}$$

which is uncorrupted since each of the components are $D_K(C_{j+2})$ and C_{j+1} are uncorrupted.

11. Let K be the key we wish to find. Use the hint. Then $C_1 = E_K(M_1)$ and $C_2 = E_K(\overline{M_1})$. Now, suppose we start a brute force attack by encrypting M_1 with different keys. If, when we use K_j we get $E_{K_j}(M_1) = C_1$ then we are done and the key we desire is $K = K_j$. However, when we use K_j we can eliminate another key. Here is how. If $E_{K_j}(M_1) = \overline{C_2}$ then we know (by complementation property) that $E_{\overline{K_j}}(\overline{M_1}) = C_2$. Hence, if this happens, we know the key is $\overline{K_j}$ since $\overline{K_j}$ would decrypt C_2 to get $\overline{M_1}$. We are effectively testing two keys for the price of one! Hence, the key space is cut in half and we only have to search an average of 2^{54}.

Chapter 5 - Exercises

1. (a) We have $W(4) = W(0) \oplus T(W(0)) = T(W(0))$. In the notation in Subsection 5.2.5, $a = b = c = d = 0$. The S-box yields $e = f = g = h = 99$ (base 10) = 01100011 (binary). The round constant is $r(4) = 00000010^0 = 00000001$. We have $e \oplus r(4) = 01100100$. Therefore,

$$W(4) = T(W(0)) = \begin{pmatrix} 01100100 \\ 01100011 \\ 01100011 \\ 01100011 \end{pmatrix}.$$

We have $W(5) = W(1) \oplus W(4) = W(4)$, and similarly $W(7) = W(6) = W(5) = W(4)$.
(b) We have $W(9) = W(5) \oplus W(8) \neq W(8)$, since $W(5) \neq 0$. But $W(10) = W(6) \oplus W(9) = W(6) \oplus W(5) \oplus W(8) = W(8)$, since $W(6) = W(5)$. Also, $W(11) = W(7) \oplus W(10) = W(7) \oplus W(6) \oplus W(9) = W(9)$, since $W(7) = W(6)$.

2. (a) We have $W(4) = W(0) \oplus T(W(0)) = \overline{T(W(0))}$. In the notation in Subsection 5.2.5, $a = b = c = d = 11111111$. The S-box yields $e = f = g = h = 22$ (decimal) = 00010110 (binary). The round constant is $r(4) = 00000010^0 = 00000001$. We have $e \oplus r(4) = 00010111$. Therefore,

$$W(4) = \overline{T(W(0))} = \begin{pmatrix} 11101000 \\ 11101001 \\ 11101001 \\ 11101001 \end{pmatrix}.$$

We have $W(5) = W(1) \oplus W(4) = \overline{W(4)}$. Also, $W(6) = W(2) \oplus W(5) = W(2) \oplus W(1) \oplus W(4) = W(4)$, since $W(1) = W(2)$. Finally, $W(7) = W(3) \oplus W(6) = W(3) \oplus W(2) \oplus W(5) = W(5)$, since $W(2) = W(3)$.
(b) $W(10) = W(6) \oplus W(9) = W(6) \oplus W(5) \oplus W(8) = \overline{W(8)}$, since the entries of $W(5) \oplus W(6)$ are strings of all 1's. Finally, $W(11) = W(7) \oplus W(10) = W(7) \oplus W(6) \oplus W(9) = \overline{W(9)}$.

3. (a) Since addition in $GF(2^8)$ is the same as $\oplus$, we have $f(x_1) \oplus f(x_2) = \alpha(x_1 + x_2) = \alpha(x_3 + x_4) = f(x_3) \oplus f(x_4)$. (b) The ShiftRow transformation permutes the entries of the matrix, which has the effect of permuting the results of the XOR. If $x_1 \oplus x_2 = x_3 \oplus x_4$, then this still holds after permuting the entries. The MixColumn transformation has the form $f(x) = Mx$, where M

is a fixed matrix and x is a binary string represented as a matrix. Therefore, $f(x_1) \oplus f(x_2) = Mx_1 \oplus Mx_2 = Mx_1 + Mx_2$, since addition in $GF(2^8)$ is XOR. This yields $M(x_1 + x_2) = M(x_1 \oplus x_2)$. If $x_1 \oplus x_2 = x_3 \oplus x_4$, then we can reverse the above steps to obtain $f(x_3) \oplus f(x_4)$. The RoundKey Addition has the form $f(x) = x \oplus K$. We have $f(x_1) \oplus f(x_2) = x_1 \oplus x_2 = x_3 \oplus x_4 = f(x_3) \oplus f(x_4)$.

4. (a) It is easy to see that if functions f and g have the equal difference property, then the composition $f \circ g$ has the equal difference property. Since all steps in this modified AES algorithm have the equal difference property, the composition of all the steps has the property.
(b) The steps in E involve permuting, multiplying by a matrix, and adding a matrix. Let $E_j(x)$ represent the result after j steps (there are 30 such steps). Let F_j denote the similar encryption, but where nothing is done (that is, $F_j(x) = F_{j-1}(x)$) in the jth step if the E_j algorithm ends with AddRoundKey. We start with x_1 and x_2. Suppose that $E_{j-1}(x_1) \oplus E_{j-1}(x_2) = F_{j-1}(x_1 \oplus x_2)$ for some $j \geq 1$. If the jth step is ShiftRow, then the entries of all matrices are given the same permutation, so we have $E_j(x_1) \oplus E_j(x_2) = F_j(x_1 \oplus x_2)$. If the jth step is MixColumn, then everything is multiplied on the left by a matrix M. This again yields the relation with j in place of $j - 1$. Finally, if the jth step is AddRoundKey, then a matrix K is XORed with $E_{j-1}(x_1)$ and with $E_{j-1}(x_2)$. These K's cancel each other. So $E_j(x_1) \oplus E_j(x_2) = E_{j-1}(x_1) \oplus E_{j-1}(x_2)$. Since $F_{j-1} = F_j$ in this case, we obtain $E_j(x_1) \oplus E_j(x_2) = F_j(x_1 \oplus x_2)$. Therefore, this relation holds for all j (the case $j = 0$ represents no encryption, so it holds trivially). In particular, since $E = E_{30}$, and since F_{30} is encryption with the AddRoundKey steps also removed, we have $E(x_1) \oplus E(x_2) = F(x_1 \oplus x_2)$, as desired.
(c) Eve uses part (b). She computes $E(x_1) \oplus E(x_2)$. This is the encryption of $x_1 \oplus x_2$ using only ShiftRow and MixColumn, and is independent of the key. These steps are easily reversed to yield $x_1 \oplus x_2$. By XORing with x_1, Eve obtains x_2.

5. $BS(x_1) = 99 = 01100011$ and $BS(x_2) = 124 = 01111100$, so $BS(x_1) \oplus BS(x_2) = 00011111$. But $BS(x_3) = 119 = 01110111$ and $BS(x_4) = 123 = 01111011$, so $BS(x_3) \oplus BS(x_4) = 00001100$. Therefore, BS does not satisfy the equal difference property. By 3(a), affine maps satisfy this property, so BS is not affine.

Chapter 6 - Exercises

1. We have $\phi(n) = (p-1)(q-1) = 100 * 112 = 11200$. A quick calculation shows that $3 \equiv 7467^{-1} \pmod{11200}$. We have $5859^3 \equiv 1415 \pmod{11413}$, so the plaintext was $1415 = no$.

2. (a) Here $\phi(n) = 4 \cdot 10 = 40$. We are looking for a number d such that $ed = 1 \pmod{40}$. Thus, we want to solve for d in $3d = 1 \pmod{40}$. Observe that $d = 27$ gives $3 \cdot 27 = 81 = 1 \pmod{40}$. Hence $d = 27$.

(b) Here, you use Euler's Theorem. d is such that $3d = 1 + k\phi(n)$ for some k. Then, $c^d = m^{3d} = m^{1+k\phi(n)} = m \pmod{n}$ by Euler's Theorem.

3. The two possible plaintexts are 8 and 9. Encrypt each to get $8^3 \pmod{437} = 75$ and $9^3 \pmod{437} = 292$. Hence, the correct plaintext is 8.

4. Here, we want a number d such that $(m^3)^d \pmod{101} = m^{3d} = m \pmod{101}$. By Fermat's Little Theorem, we need to find d such that $3d = 1 \pmod{100}$. Solving, we get $d = 67$ and thus decryption is accomplished by $c^{67} \pmod{101}$.

5. Choose d with $de \equiv 1 \pmod{p-1}$. Then $y^d \equiv x^{de} \equiv x^1 \equiv x \pmod{p}$, since we work mod $p-1$ in the exponent.

6. The number e is $m^{aba_1} \pmod{n}$. Since $aa_1 \equiv 1 \pmod{\phi(n)}$, and we work mod $\phi(n)$ in the exponent, we have $e \equiv m^b \pmod{n}$. Therefore Bob finds b_1 with $bb_1 \equiv 1 \pmod{\phi(n)}$ and computes e^{b_1}. This will be m.

7. Nelson decrypts $2^e c$ to get $2^{ed}c^d \equiv 2c^d \equiv 2m \pmod{n}$, and therefore sends $2m$ to Eve. Eve divides by 2 mod n to obtain m.

8. We have $c_2 \equiv c_1^{e_2} \equiv m^{c_1 c_2} \pmod{n}$. Therefore, this double encryption is the same as single encryption with encryption exponent $e_1 e_2$. So the security is at the same level as single encryption.

9. (a) $x^{\frac{1}{2}\phi(n)} \equiv (x^{p-1})^{(q-1)/2} \equiv 1^{(q-1)/2} \equiv 1 \pmod{p}$, and similarly for q. Note that since q is odd, the exponent $(q-1)/2$ is an integer. The following 'proof' doesn't work: $(x^{1/2})^{\phi(n)} \equiv 1 \pmod{n}$ by Euler. The problem is that $x^{1/2}$ might not make sense mod n. Fractional exponents must be avoided.

(b) Use the Chinese Remainder Theorem to combine the two congruences from (a).

(c) If $ed \equiv 1 \pmod{\frac{1}{2}\phi(n)}$, then $ed = 1 + \frac{1}{2}\phi(n)k$ for some integer k. Therefore

$$x^{ed} \equiv x \cdot (x^{\frac{1}{2}\phi(n)})^k \equiv x \cdot 1^k \equiv x \pmod{n},$$

where the middle congruence used part (b).

10. $e = 1$ means that the ciphertext is the same as the plaintext, so there is no encryption. The exponent $e = 2$ does not satisfy $\gcd(e, (p-1)(q-1)) = 1$, so it is not allowed in RSA (no d will exist).

11. Since $n_1 \neq n_2$, and since they are not relatively prime, we have $\gcd(n_1, n_2)$ must be a nontrivial common factor of n_1 and n_2. Therefore, we can factor n_1 and n_2 and break the systems.

12. We have $(516107 \cdot 187722)^2 \equiv (2 \cdot 7)^2 \pmod{n}$. Compute $\gcd(516107 \cdot 187722 - 2 \cdot 7, 642401) = 1129$. Therefore, $642401 = 1129 * 569$.

13. Let $a = 880525$, $b = 2057202$, and $c = 648581$. Then $(abc)^2 = 2^2 * 3^2 \pmod{2288233}$. Next, we need to check that $abc \neq \pm 6$. After that, factoring is accomplished by just calculating $\gcd(x - y, n)$. The information $668676^2 = 77 \pmod{2288233}$ was just trick information.

14. Use the Chinese remainder theorem to solve

$$x \equiv 7 \pmod{p}, \qquad x \equiv -7 \pmod{q}.$$

Then $x^2 \equiv 49 \pmod{p}$ and also $\pmod{q}$, hence $\pmod{pq}$.

15. (a) Note, if n were prime, then $k^2 \equiv 2^{n-1} \equiv 1 \pmod{n}$. This would contradict the assumption in part (a), and hence n must not be prime.

(b) Suppose $k^2 \equiv 1 \pmod{n}$, then we have a case of the form $x^2 \equiv y^2 \pmod{n}$ yet $x \not\equiv y \pmod{n}$, and hence may factor by calculating $\gcd(x - y, n)$. Here, $x = k$ and $y = 1$. Thus, to factor, we just calculate $\gcd(k - 1, n)$.

16. Since $\gcd(e_a, e_B) = 1$, there are integers x and y with $e_A x + e_B y = 1$. Therefore, $m = m^1 = (m^{e_A})^x (m^{e_B})^y \equiv c_A^x c_B^y \pmod{n}$. Since Eve can calculate this last quantity, she can calculate m.

17. Make a list of $1^e, 2^e, \ldots, 26^e \pmod{n}$. For each block of ciphertext, look it up on the list and write down the corresponding letter. The message given is *hello*.

18. Let $d = \gcd(x + y, n)$. If $d = n$, then $n | x + y$, hence $x \equiv -y$, contradiction. If $d = 1$, then Exercise 3.3(b) implies that $n | x - y$, so $x \equiv y \pmod{n}$, contradiction. Since $d \neq 1, n$, we find that d is a nontrivial factor of n.

19. (a) m is a multiple of $(p-1)(q-1)$, hence a multiple of $(p-1)$. Note that $\gcd(a, n) = 1$ implies that $\gcd(a, p) = 1$. Since $a^{p-1} \equiv 1 \pmod{p}$, we also have $a^m \equiv 1 \pmod{p}$. Similarly, $a^m \equiv 1 \pmod{q}$.

(b) If $a \not\equiv 0 \pmod{p}$, then $a^m \equiv 1 \pmod{p}$, from (a). Multiply by a to get $a^{m+1} \equiv a \pmod{p}$. If $\equiv 0 \pmod{p}$, then this congruence still holds, since both sides are 0 mod p. Similarly, $a^{m+1} \equiv a \pmod{q}$. The Chinese Remainder Theorem allows us to combine these to get $a^{m+1} \equiv a \pmod{pq}$.

(c) Let $m = ed - 1$, which is a multiple of $\phi(n)$. From (b), we have $a^{ed} = a^m \equiv a \pmod{n}$.

(d) If p and q are large, then the probability is only $1/p$ that a is a multiple of p and $1/q$ that a is a multiple of q. Both $1/p$ and $1/q$ are small, so the probability that $\gcd(a, n) \neq 1$ is small $(1/p + 1/q - (1/(pq)$, to be precise).

20. We would need $ed \equiv 1 \pmod{(p-1)(q-1)(r-1)}$. The verification is the same as the one for RSA.

21. Note that $d = e$, so Alice sends $m^{e^2} \equiv m^{ed} \equiv m \equiv 12345$.

22. (a) Note that $\gcd(e, 24) = 1$ leaves only $1, 5, 7, 11, 13, 17, 19$, and 23 as possibilities. These may be paired into e and $-e \pmod{24}$ pairs. Note that $e^2 \equiv (-e)^2 \pmod{24}$. Hence, it suffices to check just $1, 5, 7$, and 11 to see that $e^2 \equiv 1 \pmod{24}$. This can be easily verified by hand.

(b) The encryption exponents e are precisely those e such that $\gcd(e, 24) = 1$. In RSA, we seek to find a d such that $ed \equiv 1 \pmod{24}$. We already know that $e \cdot e \equiv 1 \pmod{24}$ from part (a). Since $\gcd(e, 24) = 1$, inverses are unique and hence $d = e$.

23. The spy tells you that $m^{12345} \equiv 1 \pmod{n}$. Hence $\psi = 12345$ acts like $\phi(n)$ (in the sense of Euler's Theorem). Now, if we can find a δ such that $e\delta \equiv 1 \pmod{\psi}$, then we have that $e\delta = k\psi + 1$ for some k, and thus $c^\delta = m^{e\delta} = (m^\psi)^k m \pmod{n} = m$. Therefore, all that is needed to decrypt is to use the publicly available e and solve $e\delta = 1 \pmod{12345}$, and then use δ as the decryption exponent.

24. (a) Write $de = 1 + 1000k$ for some integer k. Then $m^{de} \equiv m \cdot (m^{1000})^k \equiv m \cdot 1^k \equiv m \pmod{n}$.

(b) There are 1000 solutions to $x^{1000} \equiv 1 \pmod{p}$ and 1000 solutions to $x^{1000} \equiv 1 \pmod{q}$. There are $1000 \times 1000 = 10^6$ ways to combine them using the Chinese Remainder Theorem. So there are 10^6 solutions to $m^{1000} \equiv 1 \pmod{n}$.

25. Since $ed \equiv 1 \pmod{270300}$ we have $ed = 1 + 270300k$ for some integer k. Then $c^d \pmod{1113121} \equiv m^{ed} \equiv (m^{270300})^k m = m \pmod{1113121}$.

26. Let c_A and c_B be the outputs of the two machines. Then $c_A - c_B \equiv 0 \pmod{p}$ but $c_B - c_A \equiv 1 \pmod{q}$. Therefore $\gcd(c_a - c_b, n) = p$, and $q = n/p$.

27. (a) The new ciphertext is $c_1 \equiv 10^{100e} m^e$. Eve makes two lists: 1. $c_1 x^{-e}$ for all x with $1 \le x \le 10^9$, and 2. $(100y)^e$ for all y with $1 \le y \le 10^9$. A match gives $c_1 \equiv (100xy)^e$, so $m = xy$.

Another way: Eve divides c_1 by $10^{100e} \pmod{n}$ and then uses the short message attack from Section 6.2.

(b) Suppose the length of m is k. Then $m||m = (10^k + 1)m$. Therefore, the encrypted message is $(10^k+1)^e m^e$. Eve simply divides this by $(10^k+1)^e \pmod{n}$ and then uses the short message attack.

28. (a) Suppose $0 \le x < (\sqrt{2} - 1)\sqrt{n} - 1$ and $s = \lceil \sqrt{n} \rceil$. Then $x + s < (\sqrt{2} - 1)\sqrt{n} - 1 + s < (\sqrt{2} - 1)\sqrt{n} - 1 + \sqrt{n} + 1 = \sqrt{2n}$.

(b) Suppose $p|f(x)$, then $(x+s)^2 - n = kp$ for some integer k. Thus $(x+s)^2 - kp = n$. Operating modulo p gives $n \equiv (x + s)^2 \pmod{p}$.

(c) From (b), n is a square mod p, so $n \equiv a^2$. Since $p \nmid n$, we have $a \not\equiv 0 \pmod{n}$. We obtain

$$f(x) \equiv (x + s)^2 - a^2 \equiv (x + s + a)(x + s - a) \pmod{p}.$$

Since p is prime, either $x + s + a \equiv 0$ or $x + s - a \equiv 0 \pmod{p}$. This yields $x \equiv \pm a - s \pmod{p}$. Since p is odd and $a \not\equiv 0$, these two values of x are distinct mod p.

(d) We subtract $\log p$ exactly for those p for which $f(x) \equiv 0 \pmod{p}$, so we subtract once for each prime factor of $f(x)$. Since we are assuming that $f(x) =$

$p_1 p_2 \cdots p_r$ is a product of distinct primes, we have $\log f(x) - \log p_1 - \cdots - \log p_r = 0$.
(e) If $f(x)$ has a large prime factor p, then the register will be at least $\log p$, which is large. If all of the factors of $f(x)$ are in $\mathcal{B}$, then the register is 0 if the prime factors are distinct. In general, the register will contain a sum of logs of some primes from $\mathcal{B}$. These will tend to be small. Moreover, the multiple prime factors of $f(x)$ will tend to be the small primes in the factor base (for example, it is much more likely that 2^2 or 3^2 divides $f(x)$ than 65537^2 divides $f(x)$). Therefore, the register will tend to be small.
(f) The procedure in d only works with 2 out of each p values of x, rather than with each x, which is what would happen with trial division. Subtracting is a much faster operation than dividing. Also, the subtraction can be done in floating point, while the division is done with large integer arithmetic.

Chapter 7 - Exercises

1. (a) Perhaps the easiest way to do this is to list the powers of 2 mod 13 until we get 3: $2, 2^2 \equiv 4, 2^3 \equiv 8, 2^4 \equiv 16 \equiv 3 \pmod{13}$. Therefore $L_2(3) = 4$.
(b) $2^7 = 128 \equiv 11 \pmod{13}$, which implies that $L_2(11) = 7$.

2. (a) $6^2 \equiv 3 \pmod{11}$, $6^4 \equiv 3^2 \equiv 9$, $6^5 \equiv 6 \cdot 6^4 \equiv 6 \cdot 9 \equiv 10$.
(b) Since 2 is a primitive root, $2^5 = 2^{(11-1)/2} \equiv -1 \pmod{11}$. Therefore, $-1 \equiv 6^5 \equiv (2^x)^5 \equiv (-1)^x$, so x is odd.

3. Since 5 is a primitive root, $5^{611} \equiv -1 \pmod{1223}$. Therefore, $1 \equiv 3^{611} \equiv 5^{611x} \equiv (-1)^x$, so x is even.

4. $p - 1 = 2 \cdot 3^2$. First we compute $L = L_2(14) \bmod 2$: We have $14^{(p-1)/2} \equiv -1 \pmod{19}$, so $L \equiv 1 \pmod{2}$.

Now compute $L \bmod 3$: We have $14^{(p-1)/3} \equiv 7 \pmod{19}$. Since $2^6 \equiv 7 \pmod{19}$, we have $L \equiv 1 \pmod{3}$. Now write $L = 1+3x_1$. Let $\beta_1 \equiv 14 \cdot 2^{-1} \equiv 7 \pmod{19}$. Then $\beta_1^{(p-1)/9} \equiv 11 \equiv (2^{(p-1)/3})^2$, so $x_1 = 2$. Therefore $L \equiv 1+3 \cdot 2 = 7 \pmod{9}$. Since $L \equiv 1 \pmod{2}$ from above, we use the Chinese Remainder Theorem to obtain $L = 7$.

5. (a) Let $x = L_\alpha(\beta_1)$, $y = L_\alpha(\beta_2)$, and $z = L_\alpha(\beta_1\beta_2)$. Then $\alpha^{x+y} \equiv \alpha^x\alpha^y \equiv \alpha^z \pmod{p}$. By the proposition in Section 3.7, we have $x + y \equiv z \pmod{p-1}$, which is what we wanted to prove.

(b) First, we need the fact that $\alpha^u \equiv \alpha^v \pmod{p}$ if and only if $u \equiv v \pmod{ord_p(\alpha)}$. This is proved by rewriting the congruence as $\alpha^{u-v} \equiv 1 \pmod{p}$ and using Exercise 3.9(d), which says that $\alpha^{u-v} \equiv 1 \pmod{p}$ if and only if $u - v \equiv 0 \pmod{ord_p(\alpha)}$. Now use the proof of part (a) above, with $p-1$ replaced by $\mathrm{ord}_p(\alpha)$. Instead of using the proposition in Section 3.7, use the fact just proved (about u and v).

6. (a) $L_2(24) \equiv 3L_2(2) + L_2(3) \equiv 3 + 69 \equiv 72 \pmod{100}$. Therefore, $L_2(24) = 72$.
(b) $L_2(24) \equiv 3L_2(5) \equiv 3 \cdot 24 \equiv 72 \pmod{100}$. Therefore, $L_2(24) = 72$.

7. Since $11 = 44/2^2$, and since $3^{136} \equiv 1 \pmod{137}$, we have $3^x \equiv 3^{6-2\cdot10} \equiv 3^{-14} \equiv 3^{-14+136} \equiv 3^{122}$. Therefore, $x = 122$.

8. (a) If someone knows $2^x \pmod{p}$, it is difficult to find x since this is the discrete logarithm problem.

(b) If p has only 5 digits, it is easy to compute $2^k \pmod{p}$ for $k = 1, 2, \ldots p-1$ until the number $2^x \pmod{p}$ is found. Then $k = x$ is the password.

9. (a, b) We have $p - 1 = 65536 = 2^{16}$. Note that $2^{16} \equiv -1 \pmod{p}$, and

$2^{32} \equiv 1 \pmod{p}$. We have $2^{(p-1)/2} \equiv 2^{2^{15}} \equiv 1 \equiv (-1)^0 \pmod{p}$, so $x_0 = 1$. Therefore $\beta_1 \equiv \beta \equiv 2 \pmod{p}$. Then $\beta_1^{(p-1)/4} \equiv 2^{2^{14}} \equiv 1 \pmod{p}$, so $x_1 = 0$ and $\beta_2 = 2$. Continuing in this manner, we get $x_3 = \cdots = x_{10} = 0$ and $\beta_2 = \cdots = \beta_{11} = 2$. Then

$$\beta_{11}^{(p-1)/2^{12}} = 2^{16} \equiv -1 \pmod{p},$$

so $x_{11} = 1$. Therefore

$$\beta_{12} \equiv \beta_{11}\alpha^{-x_{11}2^{11}} \equiv 2 \cdot 3^{-2048} \equiv 16384 \pmod{p}.$$

We have

$$\beta_{12}^{(p-1)/2^{13}} \equiv -1 \pmod{p},$$

so $x_{12} = 1$. Then

$$\beta_{13} \equiv 16384 \cdot 3^{-2^{12}} \equiv 256 \pmod{p}.$$

We have

$$\beta_{13}^{(p-1)/2^{14}} \equiv 1 \pmod{p},$$

so $x_{13} = 0$ and $\beta_{14} = \beta_{13} = 256$. Then

$$\beta_{14}^{(p-1)/2^{15}} \equiv -1 \pmod{p},$$

so $x_{14} = 1$ and

$$\beta_{15} \equiv 256 \cdot 3^{-2^{14}} \equiv 65536 \pmod{p}.$$

Then

$$\beta_{15}^{(p-1)/2^{16}} \equiv -1 \pmod{p},$$

so $x_{15} = 1$ and

$$\beta_{16} \equiv 65536 \cdot 3^{-2^{15}} \equiv 1 \pmod{p}.$$

This means we can stop. We have

$$L_3(2) = x_0 + \cdots 2^{15}x_{15} = 2^{11} + 2^{12} + 2^{14} + 2^{15} = 55296.$$

10. Eve computes b_1 with $bb_1 \equiv 1 \pmod{p-1}$. Then $x_2^{b_1} \equiv \alpha^{bb_1} \equiv \alpha^1 \equiv \alpha \pmod{p}$.

11. $m \equiv tr^{-a} \equiv 6 \cdot 7^{-6} \equiv 12 \pmod{17}$.

12. (a) Write d in base N, so $d = a_0 + a_1N$ with $0 \leq a_i < N$. Then $m \equiv c^d \equiv c^{a_0+a_1N}$ implies that $mc^{-a_1N} \equiv c^{a_0}$. Therefore, there is a match for $j = a_0$ and $k = a_1$. This gives $a_0 + a_1N$ as a candidate for d.
(b) $c = m = 1$ gives a match for every j, k, so it is unlikely that the first match yields the correct d.
(c) The lists are of length approximately $\sqrt{N}$. This is approximately the time required to factor n by dividing by all the primes up to $\sqrt{N}$.

Chapter 8 - Exercises

1. It is easy to construct collisions: $h(x) = h(x+p-1)$, for example. (However, it is fairly quickly computed (though not fast enough for real use), and it is preimage resistant.)

2. (a) Finding a preimage is the same as finding a square root mod pq. This is computationally equivalent to factoring (see page 88).
(b) $h(x) \equiv h(n-x)$ for all x, so it is easy to find collisions.

3. h can be computed quickly, so (1) is satisfied. However, $h(x\|0\|0\|\cdots) = x$, so it is not preimage resistant. Taking different numbers of 0-blocks yields collisions, so it is not strongly collision-free.

4. The probability that no two have birthdays in the same month is

$$\left(1-\frac{1}{12}\right)\left(1-\frac{2}{12}\right)\left(1-\frac{3}{12}\right) = \frac{165}{288} \approx .573.$$

5. (a) $f'(x) = -1/(1-x)+1 = -x/(1-x) \le 0$ when $0 \le x < 1$. Also, $g'(x) = -1/(1-x)+1+2x = x(1-2x)/(1-x) \ge 0$ when $0 \le x \le 1/2$.
(b) f is decreasing and $f(0) = 0$, so $\ln(1-x)+x = f(x) \le 0$ for $0 \le x \le 1/2$. Therefore, $\ln(1-x) \le -x$. The other inequality follows similarly.
(c) From (b), we have

$$-\frac{j}{N} - \frac{j^2}{N^2} \le \ln\left(1-\frac{j}{N}\right) \le -\frac{j}{N}.$$

Summing for $1 \le j \le r-1$ yields

$$-\frac{(r-1)r}{2N} - \frac{(r-1)r(2r-1)}{6N^2} \le \sum_{j=1}^{r-1} \ln\left(1-\frac{j}{N}\right) \le -\frac{(r-1)r}{N}.$$

Since $(r-1)r(2r-1) < (r)r(2r) = 2r^3$, the result follows.
(d) Exponentiate the result in (c) and rearrange the exponents.
(e) As $N \to \infty$, we have $c/\sqrt{N} \to 0$, so $e^{c/\sqrt{N}} \to e^0 = 1$. Therefore, both ends of the inequalities in (d) are close to $e^{-\lambda}$.

6. (a) The probability is $(1/2)^j$ that we succeed on the jth try, so the expected number of tries is $1(1/2)+2(1/4)+3(1/8)+\cdots$. To evaluate this sum S, consider

$$S - \frac{1}{2}S = (1\cdot 12 + 2\cdot 14 + 3\cdot 18 + \cdots) - (1\cdot 14 + 2\cdot 18 + 3\cdot 116 + \cdots)$$

$$= \frac{1}{2} + (2-1)\frac{1}{4} + (3-2)\frac{1}{8} + \cdots = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots = 1.$$

(b) By (a), we expect 2 repetitions of the $2^{n/2}$ steps. Each such step requires an evaluation of f, which takes time a constant times n. The total expected time is a constant times $n2^{n/2}$.
(c) We perform t birthday attacks, each of which takes expected time a constant times $n2^{n/2}$.

7. Use the same formulas as in Section 8.5, but with $||g(i)$ inserted at the appropriate places.

8. (a) $\lfloor 2^{30}\sqrt{2} \rfloor = 1518500249 = 5 \cdot 16^7 + 10 \cdot 16^6 + 8 \cdot 16^5 + 2 \cdot 16^4 + 7 \cdot 16^3 + 9 \cdot 16^2 + 9 \cdot 16^1 + 9$. This gives `5A827999` in hexadecimal.
(b) These can be evaluated in the same way as in part (a).

9. (a) Make two lists. The first: $D_{K_1}(c)$ for $\sqrt{N}$ random keys K_1. The second: $E_{K_2}(m)$ for $\sqrt{N}$ random keys K_2. There is a good chance of a match, which yields $D_{K_1}(c) = E_{K_2}(m)$, hence $c = E_{k_1}(E_{K_2}(m))$. This can be changed to $c = E_{K_3}(m)$. By hypothesis, K_3 is probably the only such key, so we have probably found the key K_3.
(b) The hypotheses of (a) are easily checked. The attack makes two lists. The first: the ciphertext shifted by 6 random shifts. The second: the plaintext shifted by 6 random shifts. A match gives a shift that sends the plaintext to the ciphertext, as desired.

10. (a) and (b) Let h be either of the hash functions. Given y of length n, we have $h(y\|000\cdots) = y$, so h is not preimage resistant. Varying the number of 0's gives collisions (or you could add some 1's that cancel each other).

11. (a) Use the Chinese remainder theorem to solve $x_0 \equiv -2^{512}x_i \pmod{p_i}$ for $i = 1, 2$.
(b) kp_1p_2 has approximately $30 + 240 + 240 = 510$ bits, x_0 has at most 480 bits, and $2^{512}x_1$ has approximately $512 + 512 = 1024$ bits. This last term dominates, so the sum has approximately 1024 bits. Dividing by p_1 subtracts around 240 bits, yielding $1024 - 240 = 784$ bits.
(c) Regard q_1 as a random integer near 2^{784}. The probability that it is prime is approximately $1/\ln(2^{784}) \approx 1/543$, by the prime number theorem. The probability that both q_1 and q_2 are prime is approximately $1/543^2 \approx 1/30000$.
(d) There are $2^{30} \approx 10^9$ values of k. Each has probability around $1/30000$ of yielding two primes. Therefore, we expect around $10^9/30000 \approx 30000$ examples with both q_1 and q_2 prime.
(e) Let $x_0' = x_0 + kp_1p_2$. Then n_i is written as $x_i\|x_0'$ in binary, with each of x_1, x_2, x_0' a string of 512 bits. For $i = 1, 2$, we have $H(n_i) = H(x_i\|x_0') = h(h(IV, x_i), x_0')$. Since $h(IV, x_1) = h(IV, x_2)$, we have $H(n_1) = H(n_2)$.

Chapter 9 - Exercises

1. If Eve discovers k, then she can use r, s, m to write $ar \equiv m - ks \pmod{p-1}$ and solve for a. There will be $\gcd(r, p-1)$ solutions a. This will probably be a small number. Each of these possible values a can then be tested until one is found that satisfies $\beta \equiv \alpha^a \pmod{p}$.

2. (a) We know that $\beta^r r^s \equiv \alpha^m \pmod{p}$, since (m, r, s) is a valid signature. Let $m_1 \equiv m r_1 r^{-1} \pmod{p-1}$. Then

$$\beta^{r_1} r_1^{s_1} \equiv (\beta^r r^s)^{r_1 r^{-1}} \equiv (\alpha^m)^{r_1 r^{-1}} \equiv \alpha^{m_1}.$$

(b) It is unlikely that m_1 is a meaningful message.

3. $(3^3 \pmod{11}) \pmod{5} = (5) \pmod{5} = 0$, but $(3^3 \pmod{5}) \pmod{11} = (2) \pmod{11} = 2$.

4. (a) $(\alpha^a)^s r^r \equiv \alpha^{m-kr} \alpha^{kr} \equiv \alpha^m \pmod{p}$.

(b) $(\alpha^a)^m r^r \equiv \alpha^{am} \alpha^{kr} \equiv \alpha^s \pmod{p}$.

(c) $(\alpha^a)^r r^m \equiv \alpha^{ar} \alpha^{km} \equiv \alpha^s \pmod{p}$.

5. (a) We have $v_1 \equiv \beta^r r^{-rv^{-1}} \equiv \beta^r (\beta^v \alpha^u)^{-rv^{-1}} \equiv \alpha^{ar - arvv^{-1} - urv^{-1}} \equiv \alpha^{-urv^{-1}}$, and $v_2 \equiv \alpha^m \equiv \alpha^{su} \equiv \alpha^{-ruv^{-1}}$. Therefore $v_1 \equiv v_2$, so the signature is valid.

(b) In part (a), we choose the message by letting $m \equiv su \pmod{p-1}$ Therefore once u, v are chosen, it is unlikely that $\alpha^{h(m)} \equiv \alpha^m \pmod{p-1}$. So the signature will probably not be valid.

6. First, since $k = a$, she'll have $r = \beta$, so Eve can see this. Then $s \equiv k^{-1}(m - ar) \equiv k^{-1}m - r \pmod{p-1}$. Since Eve knows r, s, m, she can solve this for $k-1$. Actually, she obtains $\gcd(m, p-1)$ possibilities for k^{-1}, hence for k and for a. She tries each of these until she finds the value of a such that $\alpha^a \equiv \beta \pmod{p}$.

7. (a) The density of primes near x is approximately $1/\ln x$. With $x \approx 10^{100}$, we have a density of around $1/230$. Since we only look at odd numbers, the density if around $1/115$.

(b) By the reasoning in (a), the density of primes in the odd numbers near 2^{512} is approximately $2/\ln(2^{512}) \approx 1/177$.

8. (a) $\beta^{f(r)} r^s \equiv \alpha^{af(r)+ks} \equiv \alpha^m \pmod{p}$.

(b) Eve needs arrange that $r^s \equiv \alpha^m$. For example, she can take $k = 1$, $r = \alpha$, and $s = m$ and have a valid signature (m, r, s).

Chapter 10 - Exercises

1. First calculate a_A, a_B, a_C, b_A, b_B and b_C to get $a_A = 10$, $a_B = 17$, $a_C = 14$, $b_A = 14$, $b_B = 6$ and $b_C = 5$. Using $g_A(x) = a_A + b_A x$, and similarly for $g_B(x)$, we may calculate the keys by $K_{AB} = g_A(r_B) = 21$, $K_{AC} = g_A(r_C) = 7$ and $K_{BC} = g_B(r_C) = 29$.

2.

(a) To show $K_{AB} = a + b(r_A + r_B) + cr_A r_B$ simply substitute the definition for a_A and b_A into $K_{AB} = g_A(r_B)$.

(b) Using part (a) it is clear that $K_{AB} = K_{BA}$.

(c) $K_{AB} = f(r_A, r_B)$.

3. We must solve for a, b, and c. To do this, we may use $18 = a + 9b \pmod{31}$ and $24 = a + 2b \pmod{31}$ to get $a = 8$ and $b = 8$. Then, using $29 = b + 9c$ we get $c = 23$.

4.

(a) Alice calculates $(\alpha^{yq})^x$ and Bob calculates $(\alpha^{xq})^y$. These are the same.

(b) Observe that the key is $(\alpha^q)^{xy}$. Since $\alpha^{Mq} = \alpha^{p-1} = 1$ there are only M possible values for α^{kq} for different values of k. Eve may find the key by calculating α^q and raising this to successive powers.

5. Suppose we arrange the participants in a ring, like

$$Bob \to Ted \to Carol \to Alice \to Bob.$$

Each person starts with α and raises it to their private exponent, e.g. Bob calculates α^b. They each send their respective result to the person to their right. The next round, they take what they received and raise it to their private exponent and pass it to their right. If we repeat this two more times, they will all have calculated α^{btca}.

6.

(a) Use $K = M_1 \oplus K_N$ and substitute $M_1 = M_2 \oplus K_H$ to get $K = M_2 \oplus K_H \oplus K_N$. Now substitute $M_2 = M_3 \oplus K_H$ and use the fact that $K_N \oplus K_N = 0$ to get $K = M_3 \oplus K_H$.

(b) Observe that $K_N = M_3 \oplus M_2$ and $K = M_1 \oplus K_N$. Thus Eve can calculate the key.

Chapter 11 - Exercises

1. (i) We have $r \equiv \alpha_1(cx+w)+\alpha_2 \equiv Hx+\alpha_1 w+\alpha_2 \pmod q$. Therefore

$$g^r \equiv g^{w\alpha_1} g^{\alpha_2} g^{xH} \equiv g_w^{\alpha_1} g^{\alpha_2} g^{xH} \equiv a h^H \pmod p.$$

(ii) Since $c_1 \equiv w + xc \pmod q$, we have $\alpha_1 c_1 \equiv w\alpha_1 + xH \pmod q$. Therefore,

$$r \equiv \alpha_1 c_1 + \alpha_2 \equiv xH + w\alpha_1 + \alpha_2 \pmod q.$$

Multiply by s and raise Ig_2 to these exponents to obtain

$$(Ig_2)^r s \equiv (Ig_2)^{xsH} (Ig_2)^{ws\alpha_1} (Ig_2)^{s\alpha_2} \pmod p.$$

This may be rewritten as

$$A^r \equiv z^H b \pmod p.$$

(iii) Since $r_1 \equiv usd + x_1$ and $r_2 \equiv sd + x_2 \pmod q$, we have

$$g_1^{r_1} g_2^{r_2} \equiv (g_1^u g_2)^{sd} g_1^{x_1} g_2^{x_2} \equiv (I g_2^{sd} g_1^{x_1} g_2^{x_2} \equiv\equiv A^d B \pmod p.$$

2. The hacker can calculate a number I and the corresponding number z'. The hacker can then perform the six steps needed to create a valid coin, since the hacker knows x, which is needed in step 5. The fact that c_1, c, x, w are related as in step 5 is the key to the security. If that equation can be satisfied (and the hacker performs all the computations required by the Spender), all the verification equations work.

3. (a) The only place r_1 and r_2 are used in the verification procedure is in checking that $g_1^{r_1} g_2^{r_2} \equiv A^d B$. If $g_1 = g_2$, then this becomes $g_1^{r_1+r_2} \equiv A^d B$. Therefore any pair of numbers r_1', r_2' with the same sum as $r_1 + r_2$ will also work.

(b) The Spender spends the coin correctly once, using r_1, r_2. The Spender then chooses any two random numbers r_1', r_2' with $r_1' + r_2' = r_1 + r_2$ and uses the coin with the Vendor, with r_1', r_2' in place of r_1, r_2. All the verification equations work. Since there are no relations $r_1' \equiv dus + x_1, \quad r_2' \equiv ds + x_2$, the equations given in part 1 of Fraud Control do not exist in this case. There is no way to identify the Spender. The point is that the Spender can double spend without

giving away any extra information, since the new numbers r_1', r_2' can be chosen randomly and therefore contain no new information.

4. The only places that the bank uses the value of I are in the calculations of z' and β. The Sender uses these to obtain z and b. If we are ignoring z and b, the sender never needs the information provided by the bank to produce a legitimate coin. Therefore, any value of I, hence of u, can be used.

5. r_2 and r_2' are essentially random numbers (depending on hash values involving the clock), the probability is around $1/q$ that $r_2 \equiv r_2' \pmod{q}$. Since q is large, $1/q$ is small.

Chapter 12 - Exercises

1. To approach this problem, one should use a $(2, 4)$ threshold scheme. If we use a Shamir $(2, 4)$ scheme, the polynomial is of the form:

$$s(x) = 5 + a_1x + a_2x^2 + a_3x^3 \pmod{p}.$$

Let us take $p = 7$ and choose the polynomial $s(x) = 5 + x + x^2 + x^3 \pmod 7$ (there are many other possible choices for polynomials). Then the secret value is $s(0) = 5$, and we may choose the shares $(1, 1)$, $(2, 5)$, $(3, 2)$, and $(4, 5)$.

2. In this problem, the $(2, 30)$ scheme requires solving for a line that interpolates the points $(1, 13)$ and $(3, 12)$. The slope can be calculated to be 50, and the intercept to be 64. The resulting line is given by $s(x) = 50x+64 \pmod{101}$. To fix the lost data, we evaluate $s(x)$ at $x = 2$ to get $s(2) = 63$. Hence the secret is $(2, 63)$.

3. The polynomial is

$$8\frac{(x-3)(x-5)}{(1-3)(1-5)} + 10\frac{(x-1)(x-5)}{(3-1)(3-5)} + 11\frac{(x-1)(x-3)}{(5-1)(5-3)}.$$

The secret is the constant term, obtained by letting $x = 0$:

$$8\frac{15}{8} + 10\frac{5}{-4} + 11\frac{3}{8} = \frac{53}{8} \equiv \frac{2}{8} \equiv 13 \pmod{17}.$$

4. No information is obtained on the secret until there are 5 people, so there are still 1093 possibilities.

5. We have $97 = M + si = M + 20s$. Since M and s are assumed to be positive, $s = 1, 2, 3, 4$ and $M = 77, 57, 37, 17$.

6. (a) Suppose A, B, C get together and C is the unknown cheater. They compute the secret for the pairs AB, AC, BC. Only AB is correct, so if they try all three, they will find the correct combination within three tries.

(b) Clearly we need more than three people, since part (a) shows that there is only one chance in 3 of getting it right on the first try. Let's try 4, so there is another person D. The pairs AB, AD, BD give the correct secret while the pairs AC, BC, CD give incorrect secrets. If the three incorrect secret are not all the same, then the correct secret can be obtained by seeing which secret occurs most often among the 6 pairs. However, suppose that all three pairs AC, BC, CD give

the same secret. That means the lines through A and C, through B and C, and through C and D all have the same y-intercept. There are two possibilities: A, B, C, D lie on the same line, which means D is legitimate, or C lies on the y-axis. This latter possibility is not allowed, since it would mean that C has the share $(0, y)$ for some y. This type of share is never given out since $(0, f(0))$ is the secret.

7. Take a (10, 30) scheme and give the general 10 shares, the colonels 5 shares each, and the clerks 2 each. Then each of the desired groups, and no smaller groups, have the 10 shares needed to launch the missile.

8. The slopes of the lines AB and AD are equal to 3/2 and -1/3, which are congruent mod 11. Therefore A, B, D lie on a line. C is not on this line, so C is the foreign agent. The line through A and B is $y \equiv 8 + 7x$, so the secret is 8.

9. Split the launch code into three equal components using a 3 party secret splitting scheme. One component will be given to each of A, B, and C. For the component that belongs to Government A, use a $(3, 10)$ secret sharing scheme to give shares to the delegates of Government A. Similarly, use a $(4, 10)$ secret sharing scheme to give shares of the component for Government B to B's delegates. Finally, use a $(2, 10)$ scheme to share government C's component amongst C's delegates.

10. The Newton form of the interpolant has the interpolating polynomial of the form $p(x) = c_0 + c_1(x - x_1) + c_2(x - x_1)(x - x_2) + \cdots + c_n(x - x_1)(x - x_2)\cdots(x - x_n)$. Take the shares (x_k, y_k) and evaluate $p(x)$ at x_k. For example:

$$y_1 = p(x_1) = c_0$$

and

$$y_2 = p(x_2) = c_0 + c_1(x_2 - x_1).$$

We may put this in a matrix representation $Nc = y$ by taking the matrix N to be:

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & (x_2 - x_1) & 0 & \cdots & 0 \\ 1 & (x_3 - x_1) & (x_3 - x_1)(x_3 - x_2) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & (x_{n+1} - x_1) & (x_{n+1} - x_1)(x_{n+1} - x_2) & \cdots & (x_{n+1} - x_1)(x_{n+1} - x_2)\cdots(x_{n+1} - x_n) \end{pmatrix},$$

the vector $c^T = (c_0, c_1, \cdots, c_t)$ and the vector $y^T = (y_0, y_1, \cdots, y_t)$ (where c^T denotes the transpose of c). The matrix N is lower triangular, which makes it easier to solve since we can solve for the coefficients by starting with c_0, using c_0 to solve for c_1, and then using c_0 and c_1 to solve for c_2, and so on. This technique, known as forward-substitution, is commonly used in solving linear systems.

11. Recall that in the Blakley scheme we are looking for the intersection of the planes described by $z = 2x + 3y + 13$ and $z = 5x + 3y + 1$. Setting these two equations equal to each other, the $3y$ terms cancel, yielding $3x = 12$, which gives $x = 4$. This is the secret.

Chapter 13 - Exercises

1. (a) Write $x \equiv g^j \pmod{p}$. Then $g^i \equiv x^2 \equiv g^{2j} \pmod{p}$, which implies that $i \equiv 2j \pmod{p-1}$. Since $p-1$ is even and $2j$ is even, i must be even (since $i = 2j + (p-1)n$ for some n).

(b) If $g^{(p-1)/2}$ cannot be $\equiv 1 \pmod{p}$, since $p-1$ is the smallest exponent k such that $g^k \equiv 1$.

(c) Let $x = g^{(p-1)/2}$. Then $x^2 \equiv g^{p-1} \equiv 1 \pmod{p}$. Therefore, $x \equiv \pm 1 \pmod{p}$. By (b), $g^{(p-1)/2} \not\equiv 1$. Therefore $g^{(p-1)/2} \equiv -1$.

(d) Write $x \equiv g^i \pmod{p}$ for some i. Then $x^{(p-1)/2} \equiv (g^{(p-1)/2})^i \equiv (-1)^i \pmod{p}$. From (a), we have that x is a quadratic residue if and only if i is even, which happens if and only if $x^{(p-1)/2} \equiv (-1)^i \equiv +1$.

2. (a) If y has a square root mod p and mod q, then these can be combined by the Chinese Remainder Theorem to obtain a square root of y mod n. Similar reasoning applies to $-y$.

(b) This is a restatement of the proposition in Section 3.9.

(c) By (b), one of y and $-y$ is a square mod p. Suppose it is y. By (a), y cannot be a square mod q. By (b), this means that $-y$ is a square mod q. Similarly, if, instead, $-y$ is a square mod p then $-y$ is not a square mod q, hence y is a square mod q.

(d) Bob calculates $\gcd(b^2 - y, n)$, which equals p. Therefore, he can factor n.

3. (a) $x \equiv -x$ implies $2x \equiv 0$. Since p is odd, $\gcd(2, p) = 1$, so we can divide by 2 to obtain $x \equiv 0 \pmod{p}$.

(b) We have $(x+y)(x-y) \equiv 0 \pmod{p^2}$. If $p \nmid x+y$, then the proposition on page 68 implies that $x - y \equiv 0 \pmod{p^2}$. Now suppose $p | x+y$, so $x + y \equiv 0 \pmod{p}$. If also $x - y \equiv 0 \pmod{p}$, then $x - y \equiv x + y$, hence $-y \equiv y$. By (a), $y \equiv 0$, contradicting our assumption. Therefore, $p \nmid x - y$. By the proposition on page 68, $x + y \equiv 0 \pmod{p^2}$.

(c) In the usual protocol, there are four square roots (two pairs). When $p = q$, there are only two square roots (one pair), by (b), so $\pm x$ are the only choices for the square roots.

Chapter 14 - Exercises

1. Victor stands outside so that he cannot see into the tunnels. Peggy goes in and goes through a tunnel to the central chamber. If she can unlock the door, she enters the central chamber. If not, she waits outside the door. Victor, after giving Peggy time to do this, goes into the entrance to the system of tunnels. He call out the name of one of the tunnels. Peggy should come out of the central chamber via that tunnel. They repeat this procedure several time until Victor is convinced.

If Peggy has a key, she can always come out the requested tunnel. If she doesn't have the key, she has only 1 chance out of 4 to be able to come out the right tunnel.

2. (a) If Peggy does not know a, then she cannot know both r such that $h_1 \equiv \alpha^r$ and $a - r$ such that $h_2 \equiv \alpha^{a-r}$ (otherwise, she would know their sum: $a = r + (a - r)$). When Peggy sends numbers h_1 and h_2 with $h_1 h_2 \equiv \beta$, half the time she will not be able to produce the correct number in step 3.

(b) Since the probability that Peggy gives a correct response is $1/2$ when she doesn't know a, the probability is $(1/2)^t$ after t iterations.

(c) Without the h_i, Victor is unable to check whether Nelson knows anything. Nelson could be sending random numbers. The point of sending the h_i's first is that then Victor can check that Peggy is not sending random numbers r_i.

3. (a) Nelson computes a square root of y mod p and mod q, then combines them to obtain a square root of y mod n.

(b) There are four square roots of y mod n. After a few iterations, it is very likely that nelson will return an integer s with $s \not\equiv \pm x \pmod{n}$. Since Victor knows x and s, he can compute $\gcd(x - s, n)$ to obtain a nontrivial factor of n.

(c) What Eve hears is essentially a list of random integers s along with their squares y mod n. The numbers y give her no extra information, since she could have computed them by squaring the s's. Therefore, the only information Eve has obtained is a random list of numbers. That should not help factor n. Therefore, Eve probably has not obtained any useful information.

4. (a) $\alpha^y \beta^r \equiv \alpha^{k-ar} \alpha^{ar} \equiv \alpha^k \equiv \gamma$.

(b) Victor needs to know k if he wants to use y to compute a. But to find k, he needs to solve a discrete log problem. Therefore, it is unlikely (but perhaps hard to prove) that Victor obtains any useful information.

(c) Eve has no more information than Victor, so she should not be able to determine a.

(d) Eve knows that $y_1 - y_2 \equiv (k - ar_1) - (k - ar_2) \equiv a(r_2 - r_1) \pmod{p-1}$. Divide by $r_2 - r_1$ to get $\gcd(r_2 - r_1, p-1)$ choices for a. Try each one and see which yields $\beta \equiv \alpha^a$.

5. Step 4: Victor randomly chooses $i = 1$ or 2 and asks Peggy for r_i.
Step 5: Victor checks that $x_i \equiv r_i^e \pmod{n}$.
They repeat steps 1 through 5 at least 7 times (since $(1/2)^7 < .01$).

6. One way: Step 4: Victor chooses $i \neq j$ at random and asks for r_i andr_j.
Step 5: Victor checks that $x_i \equiv r_i^2$ and that $x_j \equiv r_j^2$. If Peggy does not know x, then she can choose two of r_1, r_2, r_3 at random and compute the corresponding x_k's. She then chooses the third x_k so that $x_1x_2x_3 \equiv s$. Therefore, Peggy can know 2 out of the 3 r_k's. There is 1/3 that the r_k that Peggy does not know is the one that Victor does not ask for. Since $(1/3)^5 < .01$, five repetitions are enough.

Another way: Victor asks for only one of the r_k's. Then the probability is 2/3 that Peggy can cheat in a round. Since $(2/3)^12 < .01$, twelve repetitions suffice.

Chapter 15 - Exercises

1. There are four possible outcomes for the pair (X_1, X_2), namely HH, TH, HT, TT. Each occur with probability 1/4. Hence the entropy $H(X_1, X_2)$ is

$$-4\left(\frac{1}{4}\log_2\frac{1}{4}\right) = 2$$

Observe that $H(X_1) = H(X_2) = 1$, and that $H(X_1, X_2) = H(X_1) + H(X_2)$. This happens since the two events occur independently of each other.

2. (a) There are four possible outcomes: HH, HT, TH, and TT. The probabilities are

$$\begin{aligned} p(HH) &= p^2 \\ p(HT) &= p(1-p) \\ p(TH) &= p(1-p) \\ p(TT) &= (1-p)^2 \end{aligned}$$

(b) The entropy is

$$-\left(p^2\log_2 p^2 + 2p(1-p)\log_2(p(1-p)) + (1-p)^2\log_2(1-p)^2\right).$$

By expanding and manipulating this can be expressed as $-2p\log_2 p - 2(1-p)\log_2(1-p)$. This could have been calculated easier using the fact that $H(X,Y) = H(X) + H(Y)$ when X and Y are independent. Now observe that when X and Y are independent flips of the unfair coin, that $H(X,Y) = 2H(X) = -2p\log_2 p - 2(1-p)\log_2(1-p)$.

3. The entropy is of the form

$$H(X) = \left(\frac{1}{2} + 2\frac{1}{4} + 3\frac{1}{8} + \cdots\right).$$

Now, use the identity

$$\sum_{n=1}^{\infty} nx^n = \frac{x}{(1-x)^2}.$$

Thus $H(X) = 2$.

4. X comes from two sets A and B, where $A = [0, 2^8-1]$ and $B = [2^8, 2^{32}-1]$. The size of A is 2^8 while the size of B is $2^{32} - 2^8$. The probability over A is $1/2$ and the probability over B is $1/2$. The entropy is

$$H(X) = 2^8 \left(\frac{9}{2^9}\right) - (2^{32} - 2^8) \left(\frac{1}{2(2^{32} - 2^8)} \log_2 \left(\frac{1}{2(2^{32} - 2^8)}\right)\right).$$

This can be simplified to

$$5 + \frac{1}{2} \log_2(2^{32} - 2^8).$$

5. (a) $Y = 2^X$, so the possible outcomes for Y are $1/4$, $1/2$, 0, 2, and 4. The probabilities for Y are the same as the probabilities for X. Since the entropy only uses the probabilities, $H(X) = H(Y)$.

(b) Observe that the possible outcomes for Y are 4, 1, and 0. In particular, two elements of X get mapped to $Y = 4$ and two elements get mapped to $Y = 1$. Since x^2 is not a one-to-one function, we have $H(Y) \leq H(X)$. (See problem 14.6 for more detail).

6. (a) The first step $H(X, f(X)) = H(X) + H(f(X)|X)$ comes from using the Chain Rule. Then observe that $H(f(X)|X) = 0$ since X determines $f(X)$ and there is thus no uncertainty. $H(X, f(X)) = H(f(X)) + H(X|f(X))$ also comes from the Chain Rule. The final step is $H(X|f(X)) \geq 0$ since entropy is nonnegative.

(b) Let X take the values 1 and -1 with equal probability, and $H(X) = 1$. Then take $f(X) = X^2$. There is only one possible value for $f(X)$, namely $f(X) = 1$ which happens with probability 1. Hence $H(f(X)) = 0$.

(c) To solve this problem, refer back to part (a). We must show that $H(X|f(X)) = 0$. Intuitively, this makes sense since $f(x)$ completely determines x because f is one-to-one. To see this, observe that

$$p(X = x|f(X) = y) = \begin{cases} 1 & \text{if } x = f^{-1}(y) \\ 0 & \text{elsewise} \end{cases}$$

Now

$$H(X|f(X)) = -\sum_y \sum_x p(y)p(x|f(X) = y) \log p(y)p(x|f(X) = y).$$

Observe that the inside terms are always 0 since either the probability is 0 or the logarithm term yields 0.

(d) Run length coding is a useful and practical example that illustrates the above concepts. Observe that $\mathbf{L}$ is a function of $(X_1, X_2, \cdots, X_n)$ and hence $H(\mathbf{L}) \leq H(X_1, X_2, \cdots, X_n)$. Observe that if you have any single X_k and $\mathbf{L}$ then you completely determine $(X_1, X_2, \cdots, X_n)$. Thus $H(X_1, \mathbf{L}) = H(X_1, X_2, \cdots, X_n)$. However, $H(X_1, X_2, \cdots, X_n) \leq H(\mathbf{L}) + H(X_1)$. Since $H(X_1) \leq 1$ we have $H(X_1, X_2, \cdots, X_n) \leq H(\mathbf{L}) + 1$.

7. (a) There are 9 possible outcomes: RR, RW, RB, WR, WW, WB, BR, BW, and BB. One could calculate the probabilities for each of these and then

calculate the entropy. A quicker method is to observe that since we are replacing the balls, each draw occurs independently of each other. If X is the first draw, and Y is the second draw, then $H(X,Y) = H(X) + H(Y) = 2H(X)$. The entropy for $H(X)$ is 1.4855, and hence $H(X,Y) = 2.9710$.

(b) There are 9 possible outcomes: RR, RW, RB, WR, WW, WB, BR, BW, and BB. In this case, it is easiest to calculate all the probabilities.

$$\begin{array}{lll} P(RR) = 2/9 & P(RW) = 1/6 & P(RB) = 1/9 \\ P(WR) = 1/6 & P(WW) = 1/15 & P(WB) = 1/15 \\ P(BR) = 1/9 & P(BW) = 1/15 & P(BB) = 1/45 \end{array}$$

The entropy can be calculated to be 2.9517. Observe that the entropy with replacement is greater than the entropy without replacement.

8. (a) Using independence $H(X_1, X_2, \cdots, X_n) = H(X_1) + \cdots + H(X_n)$. Since they have the same distribution, we get $H(X_1) = H(X_2) = \cdots = H(X_n)$. Thus $H(X_1, X_2, \cdots, X_n) = nH(X_1)$. Substituting and canceling gives $H_\infty(\mathbf{X}) = H(X_1)$.

(b) Use the Chain Rule to get $H(X_1, X_2, \cdots, X_n) = H(X_1) + H(X_2|X_1) + H(X_3|X_2, X_1) + \cdots H(X_n|X_{n-1}, \cdots, X_1)$. Now observe that $H(X_2|X_1) \leq H(X_2)$, $H(X_3|X_2, X_1) \leq H(X_3)$, and similarly for the other terms. Hence $H(X_1, X_2, \cdots, X_n) \leq \sum_k H(X_k)$. Now since the X_k are identically distributed, but not necessarily independent, we have $H(X_k) = H(X_1)$ for all k, and thus $H_\infty(\mathbf{X}) \leq H(X_1)$.

9. (a) The plaintext entropy is

$$H(P) = -\left(\frac{1}{3}\log_2\frac{1}{3} + \frac{2}{3}\log_2\frac{2}{3}\right).$$

(b) Observe that this system matches up with the one-time pad, and hence $H(P|C) = H(P)$.

10. (a) $H(P,K) = H(C,P,K)$ since knowledge of the plaintext and the key determine the ciphertext. $H(P,K) = H(P) + H(K)$ since the keys are chosen independently of the plaintext in a cryptosystem.

(b) $H(C,P) = H(C) + H(P|C)$ by the Chain Rule. Since we have perfect secrecy, $H(P|C) = H(P)$, and thus $H(C,P) = H(C) + H(P)$. For the last part, refer to the solution to problem 14.11 to get $H(C|P) = H(K) - H(K|C,P)$. We must show that in a system with perfect secrecy, that $H(C|P) = H(C)$. To see this, observe that $H(C,P) = H(C) + H(P|C) = H(P) + H(C|P)$. Since $H(P) = H(P|C)$, we have $H(C) = H(C|P)$.

(c) Use the last part of (b) and observe that the stated condition implies that $H(K|C,P) = 0$.

11. First, we tackle the left-hand side of the equation.

$$\begin{aligned} H(P,K,C) &= H(P) + H(K,C|P) \\ &= H(P) + H(C|P) + H(K|C,P) \end{aligned}$$

Now, we may rearrange this to get $H(C|P) = H(P,K,C) - H(P) - H(K|C,P)$. To get the right-hand side, observe $H(P,K,C) = H(P,K) = H(P) + H(K)$, and substitute.

12. $H(K) = H(K|S_1)$ since one secret reveals no information about the shared secret K.

13. a) $H(X) = \log_2 36$.

b) Observe that $Y = 1$ for all choices of X by Fermat's Little Theorem. Hence $H(Y) = 0$.

14. (a) The derivative with respect to p is

$$-\log_2(p) - \frac{1}{\log_2 p} + \log_2(1-p) + \frac{1}{\log_2 p} = -\log_2(p) + \log_2(1-p).$$

This equals 0 when $\log_2 p = \log_2(1-p)$, which means $p = 1-p$. This yields $p = 1/2$. The function is 0 at the endpoints $p = 0$ and $p = 1$, so the maximum is at $p = 1/2$, where the value is $\log_2 2 = 1$.

(b) Let $\mathbf{p} = (p_1, p_2, \cdots, p_n)$. The function we want to maximize is $f(\mathbf{p}) = -\sum_j p_j \log_2 p_j$ subject to the constraint $g(\mathbf{p}) = \sum_j p_j - 1 = 0$. The Lagrange dual function is

$$J(\mathbf{p}) = f(\mathbf{p}) + \lambda g(\mathbf{p}),$$

where λ is the multiplier. Taking the derivative of J with respect to p_j gives

$$\frac{dJ}{dp_j} = -\left(\log_2 p_j + \frac{1}{\ln 2}\right) + \lambda$$

We set the derivative equal to 0 and solve for $p_j = 2^{\lambda - (\ln 2)^{-1}}$ for each j. Since the p_j are equal to the same value for each j, we have $p_j = 1/n$, which is the desired result.

15. (a) Due to the independence of X and Y we have $p_Y(y|x) = p_Y(y)$, and

$$\begin{aligned} \tilde{H}(Y|X) &= -\sum_x \sum_y p_Y(y) \log_2 p_Y(y) \\ &= |\mathcal{X}| H(Y) \geq H(Y) \end{aligned}$$

(b) According to the result in part (a), we the release of information X would introduce more uncertainty to the knowledge of Y. Since X and Y are independent, the information in X should have no impact on the uncertainty in Y, and hence the definition $\tilde{H}(Y|X)$ is not a good description of conditional entropy.

Chapter 16 - Exercises

1. (a) $x^3 + ax^2 + bx + c = (x - r_1)(x - r_2)(x - r_3) = x^3 - (r_1 + r_2 + r_3)x^2 + (r_1r_2r_1r_3 + r_2r_3)x - r_1r_2r_3$. Therefore, $-a = r_1 + r_2 + r_3$.
(b) Expand $(x_1 - a/3)^3 + a(x_1 - a/3)^2 + b(x_1 - a/3) + c$ to get the result.

2. (a) Let $x = 0, 1, 2, \cdots 6$ and see which yield values of y. We obtain $(3, 2), (3, 5), (5, 2), (5, 5), (6, 2), (6, 5), \infty$.
(b) The slope is $(5 - 2)/(5 - 3) \equiv 5 \pmod 7$. The line is $y \equiv 5x + 1$. Intersect with the curve: $(5x - 1)^2 \equiv x^3 - 2$, which becomes $0 \equiv x^3 - 4x^2 + \cdots$. The sum of the roots is $x + 3 + 5 \equiv 4$ (negative the coefficient of x^2. Therefore, $x = 3$ and $y \equiv 5x + 1 \equiv 2$. Reflect to obtain $(3, 5)$.
(c) The slope of the tangent line is computed by implicit differentiation: $2yy' \equiv 3x^2$. This gives $4y' \equiv 27$, so $y' \equiv 5$. The line is $y = 5x + 1$. Its third point of intersection is computed as in (a) to be $(5, 5)$ (alternatively, this can be deuced from (a)). Reflect to obtain $(5, 2)$.

3. We have $2yy' \equiv 3x^2 + 2ax + b$. Substituting yields $0y' \equiv 3x^2 + 2ax + b$, so $y' = \infty$. This means that the tangent line is vertical. The third point of intersection is ∞, which reflects to give ∞. But what if $3x^2 + 2ax + b \equiv 0$? Let $f(x) = x^3 + ax^2 + bx + c$. If $f(x) \equiv 0 \equiv f'(x)$ for some value of x, then it can be shown that x is a multiple root of $f(x)$, which is not allowed. Therefore, $f'(x) \neq 0$ if $y = 0$.

4. Add (3,5)+(3,5) to obtain (129/100, -383/1000).

5. The point Q should be $(2, 3)$, not $(0, 1)$ (as in the first printing)
(a) $2Q = (0, 1)$ and $3Q = 2Q + Q = (-1, 0)$. By Exercise 3, $6Q = 3Q + 3Q = \infty$.
(b) Note that $4Q = 6Q - 2Q = \infty - 2Q = -2Q \neq \infty$ and $5Q = -Q \neq \infty$. If $iQ = jQ$, then $(i - j)Q = \infty$. But $i - j$ cannot be 1, 2, 3, 4, 5 since $q, 2Q, 3Q, 4Q, 5Q \neq \infty$. There the points on the list are distinct. Of course, this could also be shown by a straightforward calculation.

6. (a) $P + P = (5, 16)$. Now compute $3P = 2P + P$. The slope is $(9 - 16)/(10 - 5) = -7/5$. But $\gcd(5, 35) = 5$, so we have the factorization $35 = 5 \cdot 7$.
(b) We have $2yy' \equiv 3x^2 + 5$, which yields $56y' \equiv 8$, so $y' = 1/7$. But $\gcd(7, 35) = 7$, so we have the factorization $35 = 5 \cdot 7$.

7. The tangent line at (2,0) has vertical slope, so $2P = \infty$. This is infinity mod all factors of n, so no factor is singled out. The gcd that we hope will give us a factor, will give us the factor n of n.

8. Choose an elliptic curve E mod some large prime p, and choose a random

point Q on E. Store a password (written as a number x) as xP. When y is given as a password, yP is computed and compared with the entry for the user in the file. The security of the system is based on the difficulty of discrete logarithms on elliptic curves.

9. Suppose $Q = kP$, where P and Q are points on an elliptic curve E mod p. Let $N \geq \sqrt{p}+1$. Then $N^2 \geq p+1+2\sqrt{p} > \#(E \bmod p)$. In particular, $N^2 > k$. Make two lists: First: ip for $0 \leq i < N$. Second: $Q - jNP$ for $0 \leq j < N$. A match gives $k = i + jN$. This works because we can write $k = a_0 + a_1 N$ with $0 \leq a_i < N$.

10. $-R$ is obtained by reflecting across the x-axis. If P, Q, R are collinear, then $P + Q$ is obtained by finding the third point, which is Q, and reflecting to get $P + Q = -R$. Therefore $P + Q + R = \infty$. Conversely, if $P + Q + R = \infty$, then $P + Q = -R$. That means that the reflection of $-R$, namely R, is the third point on the line through P and Q. Therefore P, Q, R are collinear.

11. (a) There are only p choice for the x-coordinate of a point (other than ∞) and p choices for the y-coordinate. So the number of points is at most $1+p^2$.

(b) Look at the points $P, 2P, 3P, \ldots$. Since there are only finitely many distinct points, two of them must be the same. This means $iP = jP$ for some $i > j$. Putting everything on one side yields $(i - j)P = \infty$.

(c) Write $m = kq + r$ with $0 \leq r < k$. Then $\infty = mP = q(kP) + rP = q\infty + rP = rP$. Since k is smallest, we must have $r = 0$. Therefore, $k|m$.

(d) The order of P is the order of the subgroup of E generated by P. Lagrange's theorem says that the order of this subgroup must divide the order of the group E, which is the number of points on E.

12. (a) Since $k/p < k$ and the order is the smallest, k cannot be the order.

(b) Since $m|k$ and $k < m$, it follows that k/m is an integer ¿1. Let p be a prime divisor of k/m. Then $k/(mp)$ is an integer, which means that $p|k/m$.

(c) We know that the order of P, call it ℓ, divides k, by 8(c). If $\ell < k$, then $\ell|k/p$ for some prime divisor p of k. Since $\ell P = \infty$, we have $(k/p)P = \infty$, contradicting our assumption. Therefore $\ell = k$.

13. (a) We claim that after the kth step 2, we have $R_k == b_1 b_2 \cdots b_k P \pmod{n}$. This is easily seen to be the case for $k = 1$. Assume it is true for $k-1$. We'll show it's true for k. We have $S_k = 2R_{k-1} = 2(\cdot b_1 b_2 \cdots b_{k-1})P$. Then $R_k = S_k + b_k P = 2R_k + b_k P = 2(b_1 \ldots b_{k-1} + b_k) = b_1 \cdots b_{k-1} b_k P$. Therefore, when $k = w$ we have $R_w = xP$, as desired.

(b) Write $x = b_1 \ldots b_w$ in binary as in (a). We assume $b_1 = 1$. The algorithm is easily seen to work when $x = 0$, so we may assume $w \geq 1$. We claim that after step 2, $a = b_1 \ldots b_{w-k}$, $B = b_{w-k+1} \ldots b_w P$ and $C = 2^k P$ for some value of k. When we start, we have $k = 0$.

Suppose we arrive at step 2 with $a = b_1 \ldots b_{w-k}$, $B = b_{w-k+1} \ldots b_w P$, and $C = 2^k P$. If a is odd, then the output of step 2 is the same as the input, hence of the desired form. If a is even, then $b_{w-k} = 0$. We obtain the new $a = b_1 \ldots b_{w-k-1}$, $B = b_{w-k} b_{w-k+1} \ldots b_w P$ (we may include the extra bit at the beginning since it is 0), and $c = 2^{k+1} P$. Therefore the output has a, B, C in the desired form with $k + 1$ in place of k.

Now let's look at what happens in step 3. The output of step 2 is of the form $a = b_1 \dots b_{w-j}$, $B = b_{w-j+1} \dots b_w P$ and $c = 2^j P$ for some value of j. If a is even, step 3 does nothing, so the output still has the desired form. If a is odd, then the last bit b_{w-j} of a is 1. The new a is $a = b_1 \dots b_{w-j-1}0$. Also, the new B is $b_{w-j}b_{w-j+1} \dots b_w P + 2^j P = (1b_{w-j}b_{w-j+1} \dots b_w)P = b_{w-j}b_{w-j+1} \dots b_w P$. The new C is still $2^j P$.

If the new $a = 0$, then $j = w-1$, so $B = xP$. Therefore step 5 outputs xP, as desired. Otherwise, step 4 sends us to step 2, which outputs $a = b_1 \dots b_{w-j-1}$, $B = b_{w-j}b_{w-j+1} \dots b_w P$, and $C = 2^{j+1}P$. This is of the desired form with $k = j + 1$.

Therefore, the output of step 2 always has the desired form, as claimed.

Since a gets smaller at each application of steps 2 and 3, eventually $a = 0$ and the algorithm stops. As pointed out above, the output of step 5 is then xP.

14. (a) $V = u_1 A + u_2 B = s^{-1}mA + s^{-1}xB = s^{-1}(mA + xaA) = s^{-1}(m + ax)A = kA = R$. We used the fact that if $a \equiv b \pmod q$, then $aP = bP$ (proof: $a = b + qt$, so $aP = bP + tqP = bP + t\infty = bP$).

(b) Since q is prime and $1 \le k < q$, we must have $\gcd(k, q) = 1$. Therefore, $k^{-1} \pmod q$ exists.

(c) If $s^{-1} \pmod q$ does not exist, then $q|s$. When q is large, this is unlikely. If $s \equiv 0 \pmod q$, then $m + ax \equiv \pmod q$. (In this case, since m, x, q are known publicly, an attacker can quickly determine a and break the system.)

(d) There are two such operations made in the present scheme. There are three such operations in the ElGamal scheme.

15. (a) Write $k' = k + 2^n \ell$. Then $k'A = kA + \ell 2^n A == kA$.

(b) If $j = 2j_1$ is even, then $jT = j_1 2T = j_1 2^n A = \infty$. If $j = 2j_1 + 1$ is odd, then $jT = 2j_1 T + T = \infty + T = T$.

(c) $2^{n-1}B = k2^{n-1}A = kT$, which equals ∞ if and only if k is even, which is if and only if $x_0 = 0$.

(d) $2^{n-m-1}Q_m = 2^{n-m-1}(kA - (x_0 + \dots + 2^{m-1}x_{m-1})A) = 2^{n-m-1}(k - (x + 0 + \dots + 2^{m-1}x_{m-1}))A = 2^{n-m-1}(2^m x_m + \dots + 2^{n-1}x_{n-1})A = (x_m + \dots 2^{n-1}x_{n-1})T$, which equals ∞ if and only if $x_m = 0$.

16. (a) Since $p-1 \equiv -2 \not\equiv 0 \pmod 3$¡ we have $\gcd(3, p-1) = 1$. Therefore, $3d \equiv 1 \pmod{p-1}$ has a solution.
(b) Write $3d = 1 + (p-1)k$. If $a^3 \equiv b$, then $b^d \equiv a^{3d} = a(a^{p-1})^k \equiv a(1)^k \equiv a \pmod p$. Conversely, if $a \equiv b^d$ then $a^3 \equiv b^{3d} \equiv b(b^{p-1})^k \equiv b \pmod p$.

(c) For each y, there is a unique x with $x^3 \equiv y^2 - 1$, by (b). There are p values of y, hence p points (x, y). Including ∞ gives $p + 1$ points.

17. (a) There are $p + 1 = 6q$ points, so $E \bmod p$ is a group of order $6q$. By Cauchy's theorem, the group has an element of order q, which means that there is a point $P_0 \neq \infty$ with $qP_0 = \infty$.

(b) $E \bmod p$ is an abelian group of order $6q$, hence is cyclic since $6q$ is squarefree. In such a cyclic group, there are 6 elements with $6P = \infty$, in this case $\infty, Q, 2Q, \dots, 5Q$. For all other points P, we have $6P \neq \infty$. Since

$6qP = \infty$ by Lagrange's theorem, $6P$ has order q. The points of order dividing q form a cyclic group generated by P_0. Therefore, $6P$ is a multiple of P_0.

18. Eve knows rP_0, P_1, k. She computes

$$\tilde{e}(rP_0, P_1)^k = \tilde{(}kP_0, P_1)^r = \tilde{e}(H_1(\texttt{bob@computer.com}), P_1)^r = g^r.$$

Eve now computes $H_2(g^r)$ and XORs it with t to get m.

19. Use the has function to produce y, The find x with $x^3 = y^2 - 1$. The compute $6(x, y)$. This will be a multiple of P_0, as in Exercise 13.

20. (a) Compute $\tilde{e}(aP_0, bP_0) = \tilde{e}(P_0, P_0))^{ab}$ and $\tilde{e}(cP_0, P_0) = \tilde{e}(P_0, P_0)^c$. These are equal if and only if $ab \equiv c \pmod{q}$ since $\tilde{e}(P_0, P_0)$ is a nontrivial qth root of unity. Since P_0 has order q, we have $abP_0 = cP_0$ if and only if $ab \equiv c \pmod{q}$. Therefore, the two values of $\tilde{e}$ are equal if and only if $abP_0 = cP_0$.
(b) Let $z = \tilde{e}(P_0, P_0)$. Then $\tilde{e}(kP_0, P_0) = \tilde{e}(P_0, P_0)^k = z^k$. Therefore, we can find k by solving the multiplicative discrete logarithm problem $z^k = \tilde{e}(kP_0, P_0)$.

21. We know from Chapter 6 that knowledge of e and d allows the factorization of n. Therefore, each user can factor n and read all messages to everyone.

Chapter 17 - Exercises

1. Let $v_1 = (58, 19)$ and $v_2 = (168, 55)$. Then $v_1 \cdot v_2 / v_1 \cdot v_1 = 2.896$, so $t = 3$. We have $v_2 - 3v_1 = (-6, -2)$. Then we calculate $(58, 19) + 10(-6, -2) = (-2, -1)$, then $(-6, -2) - 3(-2, -1) = (0, 1)$, finally $(-2, -1) + (0, 1) = (-2, 0)$. The basis $(0, 1), (-2, 0)$ is reduced. The vector $(0, 1)$ is a shortest vector.

2. (a) $(107, 205) - 2(53, 88) = (1, 29)$, then $(53, 88) - 3(1, 29) = (50, 1)$. The basis $(1, 29), (50, 1)$ is reduced.
(b) $3(50, 1) + (1, 29) = (151, 32)$ is a distance 1 from $(151, 33)$, so it is the closest vector.

3. $\overline{a}M = a_1v_1 + \cdots + a_nv_n$, which is in the lattice. Conversely, any element of the lattice can be expressed as a linear combination of $v_1, \ldots, v_n$ with integer coefficients, hence in the form $\overline{a}M$.

4. (a) $dw_1 - bw_2 = d(av_1 + bv_2) - b(cv_1 + dv_2) = (ad - bc)v_1 = \pm v_1$. Similarly for v_2.
(b) If w_1, w_2 are not linearly independent, then w_2 is a multiple of w_1 (or vice versa), hence v_1 and v_2 are multiples of w_1, contradicting the linear independence of v_1, v_2. Therefore, w_1, w_2 are linearly independent. Since w_1, w_2 are in the lattice, every integral linear combination of them is in the lattice. Conversely, every element of the lattice is an integral linear combination of v_1, v_2. By (a), such a linear combination can be expressed as an integral linear combination of w_1, w_2. Therefore, the lattice is exactly the integral linear combinations of w_1, w_2. This means that they form a basis of the lattice.

5. (a) Write $j + k = i + Nm$. Assume $m \geq 0$ (the case $m < 0$ is similar). Then

$$X^{j+k} - X^i = X^i(X^{Nm} - 1) = X^i(X^{N(m-1)} + X^{N(m-2)} + \cdots + 1)(X^N - 1).$$

(b) By (a), $c_iX^i - \sum a_jb_kX^{j+k} = \sum a_jb_k(X^i - X^{j+k})$ is a multiple of $X^N - 1$.
(c) Let $f = a_0 + a_1X + \cdots + a_{N-1}X^{N-1}$ and $g = b_0 + b_1X + \cdots + b_{N-1}X^{N-1}$. Then $f * g = c_0 + c_1X + \cdots + c_{N-1}X^{N-1}$, where c_i is defined as in (b). Also, $fg = d_0 + d_1X + \cdots + d_{2N-2}X^{2N-2}$, where $d_m = \sum_{j+k=m} a_jb_k$. Therefore,

$$f * g - fg = \sum_{i=0}^{N-1} (c_iX^i - \sum_{m \equiv i \pmod N} d_mX^m).$$

By (b), each term of this sum is a multiple of $X^N - 1$.

6. $f * F \equiv 1 \pmod{p}$ means that there is a polynomial A such that $f * F = 1 + pA$. By 5(c), there is a polynomial B such that $f * F = fF + (X^N - 1)B$. Therefore,

$$f(X)F(X) + (X^N - 1)B(X) = 1 + pA(X).$$

Let $X = 1$. We obtain $f(1)F(1) = 1 + pA(1)$, so $f(1)F(1) \equiv 1 \pmod{p}$. Therefore, $f(1) \not\equiv 0 \pmod{p}$.

7. (a) The attacker simply computes $c \pmod{p}$ and obtains m.
(b) The attacker again computes $c \pmod{p}$ and obtains m. This is because $c = p\phi * h + m + qA$ for some A. If $q \equiv 0 \pmod{p}$, then we have $c \equiv m \pmod{p}$.

Chapter 18 - Exercises

1. (i) Multiply $(0,1,0,0,1,1,1)$ times the matrix H^T of example 4, Section 16.1, to obtain $(0,1,0)$. This means there is an error in the 6th bit. The correct codeword is $(0,1,0,0,1,0,1)$. The original message is the first four bits: 0,1,0,0.

(ii) Multiply $(0,1,0,1,0,1,0)$ times the matrix H^T of example 4, Section 16.1, to obtain $(0,0,0)$. This means there are no errors. The original message is the first four bits: 0,1,0,1.

2. $1\cdot 0+2\cdot 1+3\cdot 3+4\cdot 1+5\cdot 1+6\cdot 6+7\cdot 0+8\cdot 9+9\cdot 3+10\cdot 8\equiv 5\not\equiv 0 \pmod{11}$. Therefore, it is not a correct ISBN number. If we change the number to 0-13-117093-8 or to 0-13-106093-8 we get valid numbers.

3. (a) In the notation of Section 16.4, the matrix P has size $k\times(n-k)$, so P^T has size $(n-k)\times k$. The parity check matrix $[-P^T, I_{n-k}]$ has size $(n-k)\times n$. In the problem, the parity check matrix has size 3×5. Therefore, $n=5$ and $k=2$.

(b) Remove the 3×3 identity matrix to obtain

$$-P^T=\begin{pmatrix}1&1\\1&0\\0&1\end{pmatrix}.$$

We can ignore the minus sign since we are working mod 2. We have $P=\begin{pmatrix}1&1&0\\1&0&1\end{pmatrix}$ so the generating matrix is

$$G=\begin{pmatrix}1&1&0&1&0\\1&0&1&0&1\end{pmatrix}.$$

(c) The codewords in C are linear combinations of the rows of G, namely

$$(0,0,0,0,0),(1,1,0,1,0),(1,0,1,0,1),(0,1,1,1,1).$$

(d) There are $M=4$ codewords, each of length 5. The code rate is $R=\frac{\log_2(4)}{5}=0.4$.

4. (a) The generating matrix for C is $(1,1,1)$. Therefore $P=(1,1)$, so a parity check matrix is $[-P^T, I_2]=\begin{pmatrix}1&1&0\\1&0&1\end{pmatrix}$.

(b) The cosets are

$$(0,0,0),(1,1,1)$$
$$(1,0,0),(0,1,1)$$
$$(0,1,0),(1,0,1)$$
$$(0,0,1),(1,1,0)$$

and the coset leaders are the first elements of each row.

(c) Multiply the coset leader by the transpose of the parity check matrix to get the syndrome. The syndrome of the first row is (0,0). The syndrome of the second row is (1,1). The syndrome of the third row is (1,0). The syndrome of the fourth row is (0,1).

(d) Multiply (1,1,0) by the transpose of the parity check matrix to obtain (0,1). This is the syndrome of (0,0,1). Subtract $(1,1,0)-(0,0,1)$ to obtain $(1,1,1)$ as the codeword.

5. (a) C does not contain $(0,0,0)$, so it is not a subspace.

(b) Calculate the Hamming distances from the first vector to the other three, the second vector to the last two, and the third to the fourth. These distances are all 2, so the minimum distance is $d(C)=2$.

(c) We have $q=2$, $n=3$, $M=4$, and $d=2$. Therefore

$$M=4=2^{3-2+1}=q^{n-d+1}.$$

6. Note that $d(x,y)=d(x+z,y+z)$ for any vectors x,y,z. This is because adding z to both x and y does not affect whether or not the bits in a given position are equal. Therefore, $wt(u+v)=d(u+v,0)=d(u+v+v,v)=d(u,v)\leq d(u,0)+d(0,v)=wt(u)+wt(v)$.

7. $A_q(n,n)$ is the largest M such that an (n,M,n) code exists. If v_1 is a codeword in any (n,M,n) code, then it is a vector of length n. Take another codeword v_2. It must differ from v_1 in all n places since it has distance n from v_1. Now suppose we have codewords $v_1,\ldots,v_q$. Any $v_i\neq v_j$ are different in all places, in particular, in the first place. Since there are only q possibilities for the first place, all of them are now taken. There is no room for any more codewords. Therefore, $M\leq q$. An example with $M=q$ is obtained by repeating each letter of the alphabet n times.

Therefore, $M=q$ is the largest possible M, so $A_q(n,n)=q$.

8. A vector $(x_1,\ldots,x_n)$ is in $C^\perp$ if and only if $(x_1,\ldots,x_n)\cdot(1,\ldots,1)=0$, which is exactly the equation defining the parity check code of length n.

9. Suppose $u-v\in C$. Let $u+c\in u+C$. Then $u+c=v+(c+(u-v))\in v+C$, since $c+(u-v)$ is the sum of two elements of C, hence is in C. Therefore $u+C\subseteq v+C$. Now let $v+c_1\in v+C$. Then $v+c_1=u+(c_1-(u-v))\in u+C$, since the difference $c_1=(u-v)$ of two elements of C is in C. Therefore, $v+C\subseteq u+C$. It follows that $u+C=v+C$.

Conversely, suppose $u+C=v+C$. Since $u=u+0\in u+C$, we have $u\in v+C$, so $u=v+c$ for some $c\in C$. Therefore $u-v=c\in C$.

10. We have $\dim(C)+\dim(C^\perp)=n$. Since C is self-dual, $C=C^\perp$. Therefore, $n=2\dim(C)$, which is even.

11. The polynomials obtained from the $[n, 1]$ repetition code are of those of the form $a + aX + \cdots + aX^{n-1} = a \cdot g(X)$ with $a \in \mathbb{F}$. Clearly $g(X)$ is the polynomial with the smallest degree (also the same degree) among the nonzero polynomials of this form. Since the leading coefficient of $g(X)$ is 1, it is the generating polynomial.

12. (a) A calculation (i.e., long division) shows that

$$(X^3 + X + 1)(X^4 + X^2 + X + 1) = X^7 - 1$$

in $\mathbb{Z}_2[X]$.

(b) The method on page 334 gives the generating matrix

$$G = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}.$$

(c) Using the polynomial found in (a), we obtain a parity check matrix as on page 334:

$$H = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

(d) This is a straightforward calculation.

(e) The 1st, 5th, 6th, 7th columns of G' form the 4×4 identity matrix, so G' must have rank 4. The space generated by the rows of G' is therefore 4-dimensional and lies in the null space of H^T. Since H is the parity check matrix for C, the null space of H^T is the 4-dimensional code C. Therefore, C must be the space generated by the rows of G'.

(f) Taking the columns of G' in the order 1,5,6,7,2,4,3 yields the generating matrix for the Hamming $[7, 4]$ code.

13. In the notation of the theorem on page 332, the polynomial $h(X) = X^2 + 1$. Calculate

$$h(X)(1 + X + X^3 \equiv 1 + X^2$$

$$h(X)(1 + X + X^2 + X^3) \equiv 0$$

$$h(X)(X^2 + X^3) \equiv 1 + X + X^2 + X^3.$$

These calculations are all in $\mathbb{Z}_2[X] \pmod{X^4 - 1}$. The fact that $X^4 \equiv 1$ and therefore $X^5 \equiv X$ was used.

By the theorem, we find that $1 + X + X^2 + X^3$ is in C and the other two polynomials are not in C.

14. The proposition on page 334 says that the matrix H defined on that page is a parity check matrix for C. This means that $cH^T = 0$ for all $c \in C$, so the rows of H generate a subspace of $C^\perp$. The rank of H is clearly k, since it contains the upper triangular submatrix with $b_\ell = 1$ down the diagonal. Since $C^\perp$ has dimension k, the rows of H generate $C^\perp$ (most of this was proved during

the proof of the proposition on page 334). But H is the generating matrix for the code with generating polynomial $\tilde{h}_r(X)$.

15. (a) Let $n = 2t + 1$. Any vector of length n either has at most t nonzero entries, in which case it has distance at most t from $(0, 0, \ldots, 0)$, or has more than t nonzero entries, in which case it has at most t 0's. Then it has distance at most t from $(1, 1, \ldots, 1)$. Therefore, the spheres of radius t around $(0, 0, \ldots, 0)$ and $(1, 1, \ldots, 1)$ cover the space of all n-tuples. Since $d = 2t + 1$ for this code, it is perfect (see the bottom of page 307).

(b) We know that a perfect code satisfies the Hamming bound with equality. This means that

$$2 = M = \frac{2^n}{\sum_{j=0}^{t} \binom{n}{j}},$$

which implies that $\sum_{j=0}^{t} \binom{n}{j} = 2^{n-1}$.

16. (a) Let C_1 consist of all vectors of the form $(a, a, \ldots, a, 0, \ldots, 0)$ with $a \in \mathbb{F}$, where the first d entries of each vector are a and the last $n - d$ entries are 0. This is clearly an $[n, 1, d]$ code.

(b, f) There are q^{j-1} elements in C_{j-1}. Put a Hamming sphere of radius $d-1$ around each codeword. The union of these spheres has $q^{j-1}V_q(n, d-1) < q^n$ elements, so there are vectors outside these spheres. Choose one such vector v. Then $d(v, c) \geq d$ for all $c \in C_{j-1}$. By adjusting some of the entries of v, we may assume that $d(v, c) = d$ for at least one $c \in C$. (This takes care of the technical point raised in (f).)

(c) Since $v \notin C$, $\dim(C_j = \dim)C_{j-1} + 1 = j$. By definition of the space spanned by v and C_{j-1}, every element of C_j can be written in the form $av + c$ with $a \in \mathbb{F}$ and $c \in C_{j-1}$.

(d) Since multiplication by a nonzero scalar does not change the number of nonzero entries in a vector, $wt(av + c) = wt(v + a^{-1}c) = d(v + a^{-1}c, 0) = d(v, -a^{-1}c)$. The last equality follows from the fact that adding a vector z to vectors x and y does not change the number of places where x and y differ, so $d(x + z, y + z) = d(x, y)$. In the present case, we have taken $z = -a^{-1}c$. Since $d(v, c) \geq d$ for all $c \in C_{j-1}$, and $a^{-1}c \in C_{j-1}$, we have $d(v, -a^{-1}c) \geq d$.

We have implicitly assumed $a \neq 0$ If $a = 0$, then $wt(av + c) = wt(c) \geq d$, since C_{j-1} is an $[n, j-1, d]$ code.

(e) From (d), every nonzero codeword has weight at least d. In (b), we chose one distance to be exactly d. Therefore the minimum weight of C_j is d. The dimension is j, by (c). Therefore, C_j is an $[n, j, d]$ code.

(f) See (b) above.

17. $\mathcal{G}_{23}$ is a $[23, 12, 7]$ code. A short calculation shows that

$$2^{12} = M = \frac{2^{23}}{\binom{23}{0} + \binom{23}{1} + \binom{23}{2} + \binom{23}{3}}.$$

Therefore, $\mathcal{G}_{23}$ is perfect.

18. (a) Write $X^7 - 1 = (X^3 + X + 1)h(X)$ for some $h(X)$. Then $\alpha^7 - 1 = (\alpha^3 + \alpha + 1)h(\alpha) = 0$, since α is a root of $X^3 + X + 1$. Therefore $\alpha^7 = 1$.

(b) Since $1^3 + 1 + 1 \neq 0$, 1 is not a root of $X^3 + X + 1$. Therefore $\alpha \neq 1$.
(c) $\alpha^1 = (\alpha^j)^a(\alpha^7)^b = 1^a 1^b = 1$.

19. In the notation of section 16.8, we have

$$H = \begin{pmatrix} 1 & 1 & \dots & 1 \\ 1 & \alpha & \dots & \alpha^6 \end{pmatrix}.$$

A calculation yields $(s_1, s_2) = (1,0,1,1,0,1,1)H^T = (1, \alpha^2)$. The value of s_2 was calculated as follows: Since $\alpha^3 = \alpha+1$, we have $\alpha^4 = \alpha^2+\alpha$, $\alpha^5 = \alpha^3+\alpha^2 = \alpha^2+\alpha+1$, $\alpha^6 = \alpha^3+\alpha^2+\alpha = \alpha^2+1$. Therefore $s_2 = 1+\alpha^2+\alpha^3+\alpha^5+\alpha^6 = \alpha^2$, and $\alpha^{j-1} = s_2/s_1 = \alpha^2$, so $j = 3$. The error is in the 3rd position. The corrected vector is (1,0,0,1,0,1,1).

20. (a) If $\deg(g) < 1$, then g is a constant polynomial, hence $g(X) = 1$ (we have implicitly assumed $C \neq 0$), so $C = \mathbb{F}^n$, contradicting our assumption. Therefore, $\deg(g) \geq 1$.

(b) Since $\deg(g) \geq 1$, we must have $\deg(h) \leq n - 1$. Therefore $h(X)$ has at most $n - 1$ roots. There are n elements on the list $1, \alpha, \dots, \alpha^{n-1}$, and all of them are roots of $X^n - 1 = g(X)h(X)$. Since at most $n - 1$ are roots of $h(X)$, at least one must be a root of $g(X)$.

(c) We have implicitly assumed that $p \nmid n$, where the number of elements of $\mathbb{F}$ is p^m for some m. In the BCH bound in Section 16.8, we take $\delta = 0$ and obtain $d(C) \geq 2$.

Chapter 19 - Exercises

1. (a) The sequence is 1, 2, 4, 8, 1, 2, 4, 8, ... The period is 4.

(b) We need $15^2 \leq 2^m < 2 \cdot 15^2$. This yields $m = 8$.

(c) Since $c/2^m = 192/256 = 3/4$, we obtain $r = 4$, which agrees with (a).

(d) Write $r = 2^k m$, so we have $k = 2, m = 1$. We have $a = 2$. Compute $b_0 \equiv a^m \equiv 2 \pmod{15}, b_1 \equiv b_0^2 \equiv 4, b_2 \equiv b_1^2 \equiv 1$. Therefore, $4^2 \equiv 1^2 \pmod{15}$, but $4 \not\equiv \pm 1$. We find that $\gcd(4-1, 15) = 3$ is a nontrivial factor of 15.

2. (a) In the sum, write $c = c_0 + j2^s$ with $0 \leq j < 2^{m-s}$. Then

$$\sum_c e^{2\pi i c x/2^m} = e^{2\pi i c_0 x/2^m} \sum_j e^{2\pi i j x/2^{m-s}}.$$

If $x \equiv 0 \pmod{2^{m-s}}$, then all terms in the sum are 1, so we obtain $2^{m-s} e^{2\pi i c_0 x/2^m}$.

Otherwise, we have a geometric sum, which equals

$$\frac{e^{2\pi i (2^{m-s}) x/2^{m-s}} - 1}{e^{2\pi i x/2^{m-s}} - 1} = 0.$$

(b) The sum in the Fourier transform can be written in the form

$$a_0 \sum_{c \equiv 0 \pmod{2^s}} + a_s u m_{c \equiv 1 \pmod{2^s}} + \cdots a_{2^s-1} \sum_{c \equiv 2^s - 1 \pmod{2^s}}.$$

When $x \not\equiv 0 \pmod{2^{m-s}}$, each of these sums is 0 by (a).

3. (a) We have $|j_1/r_1 - j/r| = |j_1 r - j r_1|/(r_1 r)$. Since the two fractions are assume to be unequal, the numerator is a nonzero integer, hence ≥ 1. Therefore, $|j_1/r_1 - j/r| \geq 1/(r_1 r) > 1/n^2$.

(b) We have $|j_1/r_1 - j/r| \leq |j_1/r_1 - c/2^m| + |c/2^m - j/r| \leq 1/2n^2 + 1/2n^2 = 1/n^2$. From (a), this is impossible unless $j_1/r_1 = j/r$.

Chapter 2 - Mathematica

Problem 1. Let's call up the ciphertext, then try all shifts of it.

```
In[1]:= ycve
Out[1]= ycvejqwvhqtdtwvwu
```

```
In[2]:= allshifts[ycve]
```

```
ycvejqwvhqtdtwvwu

zdwfkrxwirueuxwxv

aexglsyxjsvfvyxyw

bfyhmtzyktwgwzyzx

cgzinuazluxhxazay

dhajovbamvyiybabz

eibkpwcbnwzjzcbca

fjclqxdcoxakadcdb

gkdmryedpyblbedec

hlenszfeqzcmcfefd

imfotagfradndgfge

jngpubhgsbeoehghf

kohqvcihtcfpfihig

lpirwdjiudgqgjijh

mqjsxekjvehrhkjki

nrktyflkwfisilklj

osluzgmlxgjtjmlmk

ptmvahnmyhkuknmnl

qunwbionzilvlonom

rvoxcjpoajmwmpopn

swpydkqpbknxnqpqo

txqzelrqcloyorqrp

uyrafmsrdmpzpsrsq

vzsbgntsenqaqtstr

watchoutforbrutus

xbudipvugpscsvuvt
```

The plaintext was "watch out for Brutus"

Problem 2. Here is the ciphertext:

```
In[3]:= lcll
```

```
Out[3]= lcllewljazlnnzmvyiylhrmhza

In[4]:= frequency[lcll]
Out[4]= {{a,2},{b,0},{c,1},{d,0},{e,1},{f,0},{g,0},{h,2},{i,1},{j,1},{k,0},{l,6},
          {m,2},{n,2},{o,0},{p,0},{q,0},{r,1},{s,0},{t,0},{u,0},{v,1},{w,1},{x,0},
          {y,2},{z,3}}
```

The most common letter is l, which is 7 places after e. Try shifting back by 7:

```
In[5]:= shift[lcll,-7]
Out[5]= eveexpectseggsforbreakfast
```

Therefore the plaintext is "Eve expects eggs for breakfast"

Problem 3. Let the decryption function be x=ay+b. The plaintext "if" corresponds to the numbers 8,5. The ciphertext "ed" corresponds to 4,3. Therefore 8=4a+b and 5=3a+b mod 26. Subtract to get a=3. Then b=22.
Decrypt:

```
In[6]:= affinecrypt[edsg,3,22]
Out[6]= ifyoucanreadthisthankateacher
```

Problem 4. Solve y=3x+b mod 26 for x to obtain x=9y-9b mod 26. Therefore, the plaintext can be found by computing 9y, then trying all shifts:

```
In[7]:= allshifts[affinecrypt[tcab,9,0]]

psajpuoetlkooexehepeao

qtbkqvpfumlppfyfifqfbp

ruclrwqgvnmqqgzgjgrgcq

svdmsxrhwonrrhahkhshdr

twentysixpossibilities

uxfouztjyqpttjcjmjujft

vygpvaukzrquukdknkvkgu

wzhqwbvlasrvvlelolwlhv

xairxcwmbtswwmfmpmxmiw

ybjsydxncutxxngnqnynjx

zcktzeyodvuyyohorozoky

adluafzpewvzzpipspaplz

bemvbgaqfxwaaqjqtqbqma

cfnwchbrgyxbbrkrurcrnb

dgoxdicshzyccslsvsdsoc

ehpyejdtiazddtmtwtetpd

fiqzfkeujbaeeunuxufuqe

gjraglfvkcbffvovyvgvrf
```

```
hksbhmgwldcggwpwzwhwsg

iltcinhxmedhhxqxaxixth

jmudjoiynfeiiyrybyjyui

knvekpjzogfjjzszczkzvj

lowflqkaphgkkatadalawk

mpxgmrlbqihllbubebmbxl

nqyhnsmcrjimmcvcfcncym

orziotndskjnndwdgdodzn
```

Therefore the plaintext is "twentysixpossibilities". (The encryption function was y=3x+14 mod 26.)

Problem 5. For example, encrypt the string "abcde" with various possibilities:

```
In[8]:= affinecrypt["abcde",267,11]
Out[8]= lszgn

In[9]:= affinecrypt["abcde",7,11]
Out[9]= lszgn

In[10]:= affinecrypt["abcde",33,11]
Out[10]= lszgn
```

As expected, all three encryptions are the same.

Problem 6. Look at the program and extract the relevant commands, then modify the commands and command names (to avoid changing the existing commands):

```
In[11]:= alphabetDNA = "ACGT";

         keyDNA[n_] :=
           Table[StringTake[alphabetDNA,{i}] -> 
              StringTake[alphabetDNA,{Mod[i - 1 + n,4] + 1}],{i,1,4}];

         shiftDNA[plaintext_,n_] :=
           StringReplace[plaintext, keyDNA[n]];

         affkeyDNA[m_,n_] :=
           Table[StringTake[alphabetDNA,{i}] -> 
              StringTake[alphabetDNA,{Mod[m * (i - 1) + n,4] + 1}],
             {i,1,4}];

         affinecryptDNA[plaintext_,m_,n_] :=
           StringReplace[plaintext, affkeyDNA[m,n]];
```

Part(a): Shift the plaintext by 1 (the plaintext has been stored as DNA):

```
In[12]:= shiftDNA[DNA,1]
Out[12]= TCCAAGTGTTGGTGCCAACCGGGAGCGACCCTTTCAGAGACTCCGA
```

Part (b): Let the affine function be y=mx+n. Then we need
gcd(m,4)=1.

Here is an affine encryption:

```
In[13]:= affinecryptDNA[DNA,3,2]
Out[13]= AGGTTCACAACCACGGTTGGCCCTCGCTGGGAAAGTCTCTGAGGCT
```

Here is what could happen if gcd(m,4) is not 1:

```
In[14]:= affinecryptDNA[DNA,2,1]
Out[14]= CCCTTTCTCCTTCTCCTTCCTTTTTCTTCCCCCCCTTTTTCCCCTT
```

Notice that only C and T appear in the ciphertext. Both A and G are encrypted to C.

Problem 7. Let's find the key length by computing coincidences:

```
In[15]:= coinc[hdsf,1]
Out[15]= 11

In[16]:= coinc[hdsf,2]
Out[16]= 14

In[17]:= coinc[hdsf,3]
Out[17]= 15

In[18]:= coinc[hdsf,4]
Out[18]= 25

In[19]:= coinc[hdsf,5]
Out[19]= 14

In[20]:= coinc[hdsf,6]
Out[20]= 14
```

The key length is probably 4.

Look at frequencies of letters in positions 1 mod 4:

```
In[21]:= vigvec[hdsf,4,1]
Out[21]= {0.151163,0.0465116,0.0116279,0,0.0348837,0.127907,0.0697674,0.0465116,0,
           0.0232558,0,0.0348837,0,0.0581395,0.0232558,0.0232558,0,0,
           0.0465116,0.0697674,0.0465116,0.0813953,0,0.0116279,0.0465116,0.0465116}
```

Find the dot products with the alphabet frequency vector:

```
In[22]:= corr[%]
Out[22]= {0.0448023,0.0458953,0.0421163,0.0296977,0.0289302,0.039,0.0378837,0.0494302,
           0.0342558,0.0348605,0.0262674,0.0363953,0.0443721,
           0.0526628,0.0421279,0.035186,0.0303256,0.0400233,0.045686,0.0419186,
           0.0390349,0.0345698,0.0404884,0.0370465,0.0346512,0.0333721}

In[23]:= Max[%]
Out[23]= 0.0526628
```

The max is in the 14th position.

Repeat with 2 mod 4:

```
In[24]:= corr[vigvec[hdsf,4,2]]
Out[24]= {0.0385465,0.0349767,0.0458372,0.0457209,0.033407,0.0222093,0.0342558,0.0318023,
           0.0392093,0.044314,0.0510814,0.0346395,0.0306512,
           0.0430465,0.0577674,0.0414419,0.0286395,0.0323605,0.032593,
           0.0311395,0.0359535,0.046186,0.0366628,0.0381977,0.0502326,0.0401279}

In[25]:= Max[%]
Out[25]= 0.0577674
```

The max is in the 15th position.

Now do 3 mod 4:

```
In[26]:= corr[vigvec[hdsf,4,3]]
Out[26]= {0.0374302,0.031,0.0290465,0.034093,0.0576628,0.0466163,0.0311512,0.0375116,
           0.0435349,0.0381163,0.041907,0.0430465,0.0363023,
           0.0336163,0.0410349,0.0306628,0.035407,0.0395116,0.048686,
           0.0426628,0.0364535,0.0293256,0.0341977,0.0352791,0.0468023,0.0399419}

In[27]:= Max[%]
Out[27]= 0.0576628
```

The max is in the 5th position.

Finally, do 4 mod 4:

```
In[28]:= corr[vigvec[hdsf,4,4]]
Out[28]= {0.0387674,0.0367326,0.0448837,0.0393488,0.0366163,0.0353837,0.0386628,
           0.0450581,0.0327442,0.0279535,0.0380814,0.0372326,0.0434535,
           0.0428837,0.0482558,0.0301395,0.0322791,0.0349651,0.0572209,
           0.0333488,0.0307093,0.031686,0.0378605,0.0377093,0.0416744,0.0473488}

In[29]:= Max[%]
Out[29]= 0.0572209
```

The max is in the 19th position.

Since the first position corresponds to a shift of 0, etc., the shifts are by 13, 14, 4, 18. Decrypt as follows:

```
In[30]:= vigenere[hdsf,-{13,14,4,18}]
Out[30]= uponthisbasisiamgoingtoshowyouhowabunchofbrightyoungfolksdidfinda
              championamanwithboysandgirlsofhisownamanofso
              dominatingandhappyindividualitythatyo
              uthisdrawntohimasisaflytoasugarbowlitisastoryabouta
              smalltownitisnotagossipyyarnnorisitadrymonotonous
              accountfullofsuchcustomaryfillinsasromantic
              moonlightcastingmurkyshadowsdownalongwindingcountryroad
```

Observe that the plaintext has no e's (the encryption key was "noes"). This is why the maxes of the dot products (around 0.57) were not as large as they usually are (around .065).

Problem 8. Find the key length:

```
In[31]:= coinc[ocwy,1]
Out[31]= 13
```

```
In[32]:= coinc[ocwy,2]
Out[32]= 13

In[33]:= coinc[ocwy,3]
Out[33]= 11

In[34]:= coinc[ocwy,4]
Out[34]= 6

In[35]:= coinc[ocwy,5]
Out[35]= 15

In[36]:= coinc[ocwy,6]
Out[36]= 23

In[37]:= coinc[ocwy,7]
Out[37]= 8
```

We guess that the key length is 6.

Now, perform the same calculations as in Problem 7:

```
In[38]:= corr[vigvec[ocwy,6,1]]
Out[38]= {0.0417308,0.0452115,0.0372308,0.0451731,0.0294615,0.0301731,0.0419808,
           0.0560577,0.038,0.0398654,0.0351731,0.0366538,0.0316731,
           0.0392308,0.0411154,0.0388654,0.0409808,0.0322115,0.0350769,
           0.0384423,0.0475577,0.0323846,0.0448846,0.0393654,0.033,0.0295}

In[39]:= Max[%]
Out[39]= 0.0560577
```

This occurs in the 8th position.

```
In[40]:= corr[vigvec[ocwy,6,2]]
Out[40]= {0.0360577,0.0435192,0.0398846,0.03825,0.0359423,0.0313269,0.0271538,
           0.0385,0.0441154,0.0411346,0.0473077,0.0308462,0.0300769,
           0.0466154,0.0641923,0.0341923,0.0263269,0.0384231,0.0416538,
           0.0361154,0.0362308,0.0365577,0.0325962,0.04275,0.0442308,0.037}

In[41]:= Max[%]
Out[41]= 0.0641923
```

This occurs in the 15th position.

```
In[42]:= corr[vigvec[ocwy,6,3]]
Out[42]= {0.0474231,0.0376731,0.0362885,0.0412692,0.0347885,0.0285192,0.0333462,
           0.0411731,0.0326154,0.0307885,0.0447115,0.0608269,0.0389231,
           0.0308654,0.03975,0.0458077,0.0332308,0.0335,0.0406154,
           0.0319423,0.0270769,0.0443846,0.0481538,0.0370962,0.0356731,0.0445577}

In[43]:= Max[%]
Out[43]= 0.0608269
```

This occurs in the 12th position.

```
In[44]:= corr[vigvec[ocwy,6,4]]
```

```
Out[44]= {0.0461346,0.0405,0.03875,0.0313654,0.0392692,0.0342885,0.0341154,0.0291154,
           0.0478269,0.0391346,0.0355769,0.0348077,0.0608846,0.0437885,
           0.0291731,0.0347115,0.0438846,0.0282692,0.0314615,0.0426154,
           0.0351538,0.0313846,0.0409231,0.0447885,0.0410577,0.0420192}

In[45]:= Max[%]
Out[45]= 0.0608846
```

This occurs in the 13th position.

```
In[46]:= corr[vigvec[ocwy,6,5]]
Out[46]= {0.0506346,0.0364423,0.032,0.0352692,0.0602308,0.0388846,0.0328077,
           0.0376731,0.0451154,0.0265385,0.0355769,0.0444615,0.0360577,
           0.0361154,0.0383654,0.0446154,0.0360192,0.0417308,0.0380769,
           0.0432692,0.0394038,0.0364038,0.0366346,0.03375,0.0353846,0.0295385}

In[47]:= Max[%]
Out[47]= 0.0602308
```

This is in the 5th position.

```
In[48]:= corr[vigvec[ocwy,6,6]]
Out[48]= {0.03125,0.0328654,0.0522885,0.0472308,0.0258077,0.0328654,0.0490577,
           0.03925,0.0310192,0.0324808,0.0397885,0.0301154,0.0384038,0.0357692,
           0.0403846,0.0394808,0.0348269,0.0372115,0.0636923,0.0385962,
           0.0274423,0.0348269,0.0485385,0.0293846,0.0417115,0.0467115}

In[49]:= Max[%]
Out[49]= 0.0636923
```

This is in the 19th position.

Since the 1st position corresponds to a shift of 0, etc., we find that the shifts were by 7, 14, 11, 12, 4, 18. The key word was "holmes". Decrypt:

```
In[50]:= vigenere[ocwy,-{7,14,11,12,4,18}]
Out[50]= holmeshadbeenseatedforsomehoursinsilencewithhislongthinbackcurved
              overachemicalvesselinwhichhewasbrewingaparticularly
              malodorousproducthisheadwassun
              kuponhisbreastandhelookedfrommypointofviewlikeastrange
              lankbirdwithdullgreyplumageandablacktopknot
              sowatsonsaidhesuddenlyyoudonotproposetoinvesti
              nsouthafricansecurities
```

Problem 9. Follow the procedure of Problem 8:

```
In[51]:= coinc[xkju,1]
Out[51]= 15

In[52]:= coinc[xkju,2]
Out[52]= 17

In[53]:= coinc[xkju,3]
Out[53]= 13

In[54]:= coinc[xkju,4]
```

```
Out[54]= 10

In[55]:= coinc[xkju,5]
Out[55]= 23

In[56]:= coinc[xkju,6]
Out[56]= 5
```

The key length is 5.

```
In[57]:= corr[vigvec[xkju,5,1]]
Out[57]= {0.0444545,0.0683636,0.0388333,0.0296212,0.0382121,0.0411364,0.0349394,
           0.0312273,0.0357576,0.0299697,0.035697,0.0370758,0.0492424,
           0.0400758,0.0421818,0.0406212,0.0474091,0.0393788,0.0313636,
           0.0313333,0.0409091,0.0331364,0.0299394,0.0440909,0.0357121,0.0303182}

In[58]:= Max[%]
Out[58]= 0.0683636
```

This is in position 2.

```
In[59]:= corr[vigvec[xkju,5,2]]
Out[59]= {0.0344545,0.0240909,0.0334242,0.0716667,0.0435606,0.0306212,0.0311212,
           0.0461364,0.0347727,0.0348333,0.0411818,0.0387121,0.0308182,
           0.0297424,0.047303,0.0450152,0.0387273,0.0358485,0.0459545,
           0.0402879,0.0320606,0.0377727,0.0440909,0.0332727,0.0285,0.0470303}

In[60]:= Max[%]
Out[60]= 0.0716667
```

This is in position 4.

```
In[61]:= corr[vigvec[xkju,5,3]]
Out[61]= {0.0350606,0.0485606,0.0283485,0.0278788,0.0457576,0.0665455,0.0414242,
           0.0331061,0.0355152,0.0432121,0.0326212,0.0329394,0.0387576,
           0.0306212,0.0387576,0.0391212,0.05,0.0347879,0.0407727,
           0.044303,0.0455455,0.0367879,0.0289848,0.0274697,0.034803,0.0393182}

In[62]:= Max[%]
Out[62]= 0.0665455
```

This is in position 6.

```
In[63]:= corr[vigvec[xkju,5,4]]
Out[63]= {0.0387121,0.0288485,0.031697,0.0451061,0.0367727,0.0276667,0.0360303,
           0.0726667,0.0411818,0.0353485,0.0297121,0.0448788,0.0301667,
           0.0385909,0.0381667,0.0333636,0.0274394,0.0365455,0.0512424,
           0.0434242,0.0419394,0.034197,0.0507273,0.0409848,0.0342576,0.0313333}

In[64]:= Max[%]
Out[64]= 0.0726667
```

This is in position 8.

```
In[65]:= corr[vigvec[xkju,5,5]]
```

```
Out[65]= {0.0326818,0.030697,0.0388485,0.0368485,0.0343485,0.0436818,0.0343788,
           0.0369394,0.0385303,0.0671818,0.0381212,0.0336061,0.0333182,
           0.042,0.0283485,0.0359848,0.0382424,0.0324394,0.0378788,
           0.0393333,0.0444697,0.0388788,0.0395909,0.044,0.0430606,0.0375909}

In[66]:= Max[%]
Out[66]= 0.0671818
```

This is in position 10.

The key is {1,3,5,7,9}. Decrypt:

```
In[67]:= vigenere[xkju,-{1,3,5,7,9}]
Out[67]= wheninthecourseofhumaneventsitbecomesnecessaryforonepeopletodissolve
             thepoliticalbandswhichhaveconnectedthemwithanotherand
             toassumeamongthepowersoft
             heearththeseparateandequalstationtowhichthelawsofnature
             andofnaturesgodentitlethemadecentrespecttotheopinions
             ofmankindrequiresthattheyshoulddecl
             arethecauseswhichimpelthemtotheseparation
```

(this is the start of the Declaration of Independence)

Problem 10. Change the ciphertext to numbers:

```
In[68]:= txt2num0["zirkzwopjjoptfapuhfhadrq"]
Out[68]= 250817102522141509091415190500152007050700031716
```

Invert the matrix:

```
In[69]:= Inverse[{{1,2,3,4},{4,3,2,1},{11,2,4,6},{2,9,6,4}}]
Out[69]= {{-8/55, 2/55, 1/11, 0}, {-86/55,
-116/55, 8/11, 1}, {37/11, 54/11, -19/11,
-2}, {-16/11, -29/11, 10/11, 1}}
```

As on page 379, we need to remove denominators:

```
In[70]:= 55*%
Out[70]= {{-8,2,5,0},{-86,-116,40,55},{185,270,-95,-110},{-80,-145,50,55}}

In[71]:= invM = Mod[PowerMod[55,-1,26]*%,26]
Out[71]= {{6,18,19,0},{6,22,22,1},{1,12,3,24},{8,21,8,1}}
```

Break the ciphertext numbers into blocks of 4 and multiply by invM mod 26:

```
In[72]:= Mod[{25,8,17,10}.invM,26]
Out[72]= {9,0,2,10}

In[73]:= Mod[{25,22,14,15}.invM,26]
Out[73]= {0,13,3,9}

In[74]:= Mod[{9,9,14,15}.invM,26]
Out[74]= {8,11,11,22}

In[75]:= Mod[{19,5,0,15}.invM,26]
Out[75]= {4,13,19,20}
```

```
In[76]:= Mod[{20,7,5,7}.invM,26]
Out[76]= {15,19,7,4}

In[77]:= Mod[{0,3,17,16}.invM,26]
Out[77]= {7,8,11,11}
```

Change back to letters:

```
In[78]:= num2txt0[900021000130309081111220413192015190704070811111]
Out[78]= jackandjillwentupthehill
```

Problem 11. Find the length:

```
In[79]:= lfsrlength[L101,10]

{1,1}

{2,1}

{3,1}

{4,0}

{5,1}

{6,1}

{7,0}

{8,0}

{9,0}

{10,0}
```

The length is 10.

```
In[80]:= lfsrsolve[L101,6]
Out[80]= {1,1,0,1,1,0}
```

The recurrence is x_{n+6}=x_n + x_{n+1} + x_{n+3} + x_{n+4} mod 2.

Problem 12. Find the length:

```
In[81]:= lfsrlength[L100,15]

{1,1}

{2,0}

{3,1}

{4,0}

{5,0}

{6,1}

{7,0}

{8,1}

{9,0}

{10,0}
```

```
{11,0}

{12,0}

{13,0}

{14,0}

{15,0}
```

The length is 8.

```
In[82]:= lfsrsolve[L100,8]
Out[82]= {1,1,0,0,1,0,0,0}
```

These are the coefficients of the recurrence.

Problem 13. Here is the ciphertext:

```
In[83]:= L011
Out[83]= {0,1,1,0,0,0,1,0,1,0,1,1,1,0,0,1,1,1,0,1,0,1,0,0,0,1,0,
0,0,1,1,0,0,0,1,0,1,0,1,1,1,0,0,1,1,1,0,1,0,1}
```

XOR the fist 15 terms with the plaintext to get the LFSR output:

```
In[84]:= Mod[{0,1,1,0,0,0,1,0,1,0,1,1,1,0,0}+{1,0,0,1,0,0,1,0,0,1,0,0,1,0,0},2]
Out[84]= {1,1,1,1,0,0,0,0,1,1,1,1,0,0,0}
```

Find the length of the recurrence:

```
In[85]:= lfsrlength[%,8]

{1,1}

{2,0}

{3,0}

{4,1}

{5,1}

{6,0}

{7,0}

{8,0}
```

The length is 5.

```
In[86]:= lfsrsolve[%%,5]
Out[86]= {1,1,0,0,1}
```

Now generate the LFSR output using these coefficients and the first 5 terms of the LFSR output:

```
In[87]:= lfsr[{1,1,0,0,1},{1,1,1,1,0},50]
Out[87]= {1,1,1,1,0,0,0,0,1,1,1,1,0,0,0,0,1,1,1,1,0,0,0,0,1,1,1,
1,0,0,0,0,1,1,1,1,0,0,0,0,1,1,1,1,0,0,0,0,1,1}
```

XOR this with the ciphertext to get the plaintext:

```
In[88]:= Mod[% + L011,2]
```

```
Out[88]= {1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,1,
1,0,1,1,0,1,1,0,1,1,0,1,1,0,1,1,0,1,1,0,1,1,0}
```

This is the plaintext.

Chapter 3 - Mathematica

Problem 1.

```
In[89]:= GCD[8765,23485]
Out[89]= 5
```

Problem 2. (a)

```
In[90]:= ExtendedGCD[65537,3511]
Out[90]= {1,{-1405,26226}}
```

The gcd is 1, and x=-1405, y=26226.

Part (b)

```
In[91]:= 17 * (-1405)
Out[91]= -23885

In[92]:= 17 * 26226
Out[92]= 445842
```

Therefore, x=-23885, y=26226 is one solution.

Problem 3.

```
In[93]:= PowerMod[3,1234567,100000]
Out[93]= 40587
```

Problem 4.

```
In[94]:= Solve[{314 * x == 271, Modulus == 11111},x,Mode → Modular]
Out[94]= {{Modulus → 11111,x → 10298}}
```

The answer is 10298.

Problem 5. Note that

```
In[95]:= GCD[216,606]
Out[95]= 6
```

Therefore there are 6 solutions.

```
In[96]:= Solve[{216 * x == 66, Modulus == 606},x,Mode → Modular]
Out[96]= {{Modulus → 606,x → 48},{Modulus → 606,x → 149},{Modulus → 606,x →
250},{Modulus → 606,x → 351},{Modulus → 606,x → 452},{Modulus → 606,x →
553}}
```

Problem 6.

```
In[97]:= ChineseRemainderTheorem[{17,18,19},{101,201,301}]
Out[97]= 61122
```

Problem 7.

```
In[98]:= PowerMod[2,390,391]
Out[98]= 285

In[99]:= EulerPhi[391]
Out[99]= 352

In[100]:= PowerMod[2,352,391]
Out[100]= 1
```

Therefore, j=352 works.

Problem 8. Pick a random x,say x=123. Now compute y via the Chinese Remainder Theorem as follows:

```
In[101]:= ChineseRemainderTheorem[{123,-123},{84047,65497}]
Out[101]= 5495329171
```

Call this y. The square of y will be congruent to 123^2 mod each of the primes, therefore mod their product.

```
In[102]:= PowerMod[%,2,84047*65497]
Out[102]= 15129

In[103]:= PowerMod[123,2,84047*65487]
Out[103]= 15129
```

Therefore x^2 = y^2 mod 84047*65497.

Problem 9.

```
In[104]:= FactorInteger[65537-1]
Out[104]= {{2,16}}
```

Therefore 2 is the only prime factor of p-1.
The method of Exercise 10 says that we need to raise 3 to the (p-1)/2 power:

```
In[105]:= PowerMod[3,65536/2,65537]
Out[105]= 65536
```

Since this is not 1, Exercise 10 tells us that 3 is a primitive root for 65537.

Problem 10. (a) We follow the procedure from pages 384-385.

```
In[106]:= Inverse[{{1,2,4},{1,5,25},{1,14,196}}]
Out[106]= {{35/18,-28/27,5/54},{-19/36,16/27,-7/108},{1/36,-1/27,1/108}}
```

Multiply by 108:

```
In[107]:= %*108
Out[107]= {{210,-112,10},{-57,64,-7},{3,-4,1}}
```

Multiply by 108^{-1} mod 101:

```
In[108]:= Mod[PowerMod[108,-1,101]*%,101]
Out[108]= {{30,85,88},{64,38,100},{87,86,29}}
```

This is the inverse mod 101.

Part (b). There is no inverse mod p exactly when p divides the determinant:

```
In[109]:= Det[{{1,2,4},{1,5,25},{1,14,196}}]
Out[109]= 324

In[110]:= FactorInteger[324]
Out[110]= {{2,2},{3,4}}
```

Therefore, the matrix is not invertible mod 2 and mod 3.

Problem 11. Since p=34807 is 3 mod 4, we can raise to the (p+1)/4 power:

```
In[111]:= PowerMod[26055,(34807+1)/4,34807]
Out[111]= 33573
```

The other square root is

```
In[112]:= Mod[-33573,34807]
Out[112]= 1234
```

Problem 12. First, factor the modulus:

```
In[113]:= FactorInteger[2325781]
Out[113]= {{523,1},{4447,1}}
```

Both primes are 3 mod 4,so we find square roots by raising to the (p+1)/4 power:

```
In[114]:= PowerMod[1522756,(523+1)/4,523]
Out[114]= 335

In[115]:= PowerMod[1522756,(4447+1)/4,4447]
Out[115]= 1234
```

Now recombine using the Chinese Remainder Theorem:

```
In[116]:= ChineseRemainderTheorem[{335,1234},{523,4447}]
Out[116]= 437040

In[117]:= ChineseRemainderTheorem[{-335,1234},{523,4447}]
Out[117]= 1234

In[118]:= ChineseRemainderTheorem[{335,-1234},{523,4447}]
Out[118]= 2324547

In[119]:= ChineseRemainderTheorem[{-335,-1234},{523,4447}]
Out[119]= 1888741
```

These are the four square roots.

Problem 13.

```
In[120]:= PowerMod[48382,(83987+1)/4,83987]
Out[120]= 60555

In[121]:= PowerMod[60555,2,83987]
Out[121]= 35605
```

This is -48382 mod 83987:

```
In[122]:= Mod[-48382,83987]
Out[122]= 35605
```

Therefore, we found the square root of -48382 mod 83987, rather than the square root of 48382. This is because 48382 does not have a square root mod 83987.

Chapter 6 - *Mathematica*

Note: There are two commands that can be used to change text to numbers: num1 and txt2num1. They are the same function. Similarly, both alph1 and num2txt1 change numbers back to text and are the same function.

Problem 1. Encrypt each of the two messages and see which one it is:

```
In[123]:= PowerMod[txt2num1["one"],6551,712446816787]
Out[123]= 273095689186

In[124]:= PowerMod[txt2num1["two"],6551,712446816787]
Out[124]= 709427776011

In[125]:= Therefore the plaintext was "one".
```

Problem 2. Use the Universal Exponent Factoring method. First, let's store n:

```
In[126]:= n = 718548065973745507
Out[126]= 718548065973745507
```

Compute e*d-1:

```
In[127]:= 3449 * 543546506135745129 - 1
Out[127]= 1874691899662184949920
```

Express it as a power of 2 times an odd number:

```
In[128]:= %/8
Out[128]= 234336487457773118740

In[129]:= %/4
Out[129]= 58584121864443279685
```

We see that e*d-1 is 32 times an odd number.

```
In[130]:= b0 = %
Out[130]= 58584121864443279685
```

Pick a random number, say 2, and raise it to the b0 power:

```
In[131]:= PowerMod[2,b0,n]
Out[131]= 511846277294206136
```

Successively square until we get 1 mod n:

```
In[132]:= PowerMod[%,2,n]
Out[132]= 551183936117094424

In[133]:= PowerMod[%,2,n]
Out[133]= 576566739470926048

In[134]:= PowerMod[%,2,n]
Out[134]= 1
```

The last number before the 1 was not -1, so we use the gcd to factor n:

```
In[135]:= GCD[%% - 1, n]
Out[135]= 740876531
```

This is one of the factors of n. The other is

```
In[136]:= n/%
Out[136]= 969862097
```

Problem 3.

```
In[137]:= p = NextPrime[Random[Integer, {10^29, 10^30}]]
Out[137]= 402705413885710376590247255323

In[138]:= q = NextPrime[Random[Integer, {10^29, 10^30}]]
Out[138]= 981658456965686427441951359623

In[139]:= n = p * q
Out[139]= 39531917520677456119358471043920692079268372617274387402322 9
```

Let’s choose a random 10-digit prime as the encryption exponent:

```
In[140]:= e = NextPrime[Random[Integer, {10^9, 10^10}]]
Out[140]= 8043107833
```

Just to be sure:

```
In[141]:= GCD[e, (p - 1) * (q - 1)]
Out[141]= 1
```

Encrypt:

```
In[142]:= PowerMod[txt2num1["cat"], e, n]
Out[142]= 974733122694798100911519166262913782976484474058460294449 61

In[143]:= PowerMod[txt2num1["bat"], e, n]
Out[143]= 154014014076091742858892655592524685510646363741074986515366

In[144]:= PowerMod[txt2num1["encyclopedia"], e, n]
Out[144]= 876375172732070859990445455439494310245138480501680777797 39

In[145]:= PowerMod[txt2num1["antidisestablishmentarianism"], e, n]
Out[145]= 346249201616666622428262988027121744553143608807213087607118
```

There is no obvious relation between the ciphertexts and the plaintexts.

Problem 4. Let’s choose our bound to be B=50 and let a=2:

```
In[146]:= PowerMod[2, Factorial[50], 618240007109027021]
Out[146]= 394571778315865637

In[147]:= GCD[% - 1, 618240007109027021]
Out[147]= 250387201
```

The other factor is

```
In[148]:= 618240007109027021/%
Out[148]= 2469135821
```

The reason this worked is

```
In[149]:= FactorInteger[250387201 - 1]
Out[149]= {{2,8},{3,5},{5,2},{7,1},{23,1}}
```

Therefore, p-1 has only small prime factors.

Problem 5. We'll use B=100 and a=2:

```
In[150]:= PowerMod[2,Factorial[100],n1]
Out[150]= 96162382165780005507702322927071548424 8143

In[151]:= GCD[% - 1,n1]
Out[151]= 364438989216827965440001
```

The other factor is

```
In[152]:= n1/%
Out[152]= 24242424242468686907
```

Problem 6. We use the Basic Principle:

```
In[153]:= GCD[85975324443166 - 462436106261,537069139875071]
Out[153]= 9876469
```

The other factor is

```
In[154]:= 537069139875071/%
Out[154]= 54378659
```

Problem 7. Pick a random x, say x=123. Now compute y via the Chinese Remainder Theorem as follows:

```
In[155]:= ChineseRemainderTheorem[{123,-123},{985739879,1388749507}]
```

312089563232070143

Call this y.The square of y will be congruent to 123^2 mod each of the primes,therefore mod their product.

```
In[156]:= PowerMod[%,2,985739879 * 1388749507]
Out[156]= 15129

In[157]:= PowerMod[123,2,985739879 * 1388749507]
Out[157]= 15129
```

Therefore x^2=y^2 mod 985739879*1388749507.

Problem 8. (a)

```
In[158]:= GCD[33335 - 670705093,670726081]
Out[158]= 54323
```

The other factor is

```
In[159]:= 670726081/%
Out[159]= 12347
```

Part (b) Observe that 3 = -670726078 mod 670726081. Let's try the gcd anyway:

```
In[160]:= GCD[3 - 670726078, 670726081]
```

```
Out[160]= 1
```

So we get the trivial factorization.

Problem 9. This is what happens in the exponent factorization method. We see that 1488665^2 is 1 mod 3837523, so we use the gcd to factor:

```
In[161]:= GCD[1488665 - 1, 3837523]
Out[161]= 3511
```

The other factor is

```
In[162]:= 3837523/%
Out[162]= 1093
```

Problem 10. (a) Find the square root of n. The next prime will be a factor and the previous prime will be the other factor.

Part (b)

```
In[163]:= n = 10993522499
Out[163]= 10993522499

In[164]:= q = NextPrime[N[Sqrt[n]]]
Out[164]= 104851
```

We used N[] in order to change Sqrt[n] to a number. The other factor is

```
In[165]:= p = n/q
Out[165]= 104849
```

Now find the decryption exponent:

```
In[166]:= d = PowerMod[113, -1, (p - 1) * (q - 1)]
Out[166]= 5545299377
```

Decrypt:

```
In[167]:= PowerMod[10787770728, d, n]
Out[167]= 5011925

In[168]:= num2txt1[%]
Out[168]= easy
```

Part (c) We use the same procedure as in (b). We use N[,75] to get 75-digit accuracy in the evaluation of the square root.

```
In[169]:= q = NextPrime[N[Sqrt[naive], 75]]
Out[169]= 123456789000000314159265001 43

In[170]:= p = naive/q
Out[170]= 123456789000000314159265000 31

In[171]:= d = PowerMod[9007, -1, (p - 1) * (q - 1)]
Out[171]= 660462771866258534689069380246851318997849360492799256 83

In[172]:= PowerMod[cnaive, d, naive]
Out[172]= 200809190014211302051800230119001415200019050321 1805
```

```
In[173]:= num2txt1[%]
Out[173]= this number was not secure
```

Problem 11. (a)

```
In[174]:= p = 123456791
Out[174]= 123456791

In[175]:= q = 987654323
Out[175]= 987654323

In[176]:= m = 14152019010605
Out[176]= 14152019010605

In[177]:= PowerMod[m,127,p]
Out[177]= 104120308

In[178]:= PowerMod[m,127,q]
Out[178]= 812538893

In[179]:= c = ChineseRemainderTheorem[{%,%%},{q,p}]
Out[179]= 18001903071439991
```

Part (b) Let's change one digit by adding 100000:

```
In[180]:= PowerMod[m,127,p] + 100000
Out[180]= 104220308

In[181]:= PowerMod[m,127,q]
Out[181]= 812538893

In[182]:= f = ChineseRemainderTheorem[{%,%%},{q,p}]
Out[182]= 17982149984979991

In[183]:= GCD[f - c,p * q]
Out[183]= 987654323
```

This found the factor q because f=c mod q but f is not congruent to c mod p. Therefore gcd(f-c,pq)=q.

Problem 12.

```
In[184]:= p = 76543692179
Out[184]= 76543692179

In[185]:= q = 343434343453
Out[185]= 343434343453

In[186]:= e = 457
Out[186]= 457
```

We'll try all possibilities for the last digit until we get a meaningful message. It will be easiest to append a 0 to the received text, then add 0, 1, 2, ... and decrypt.

```
In[187]:= c = 23043293280169369471950
Out[187]= 2304329328016936947195 0
```

The decryption exponent is

```
In[188]:= d = PowerMod[e, -1, (p - 1) * (q - 1)]
Out[188]= 1553104555909567360609
```

Now try to decrypt the various possibilities. Note that the incorrect ones will probably contain two-digit blocks larger than 26 hence they cannot be changed back to letters.

```
In[189]:= PowerMod[c + 0, d, p * q]
Out[189]= 4174116046631355447474
```

```
In[190]:= PowerMod[c + 1, d, p * q]
Out[190]= 8579710266419129803917
```

```
In[191]:= PowerMod[c + 2, d, p * q]
Out[191]= 1913091205
```

This one can be changed back to letters:

```
In[192]:= num2txt1[%]
Out[192]= smile
```

Therefore the missing digit was 2 and the plaintext was "smile".

Problem 13.

```
In[193]:= n = 38200901201
Out[193]= 38200901201
```

Write n-1 as a power of 2 times an odd number:

```
In[194]:= (n - 1)/8
Out[194]= 4775112650
```

```
In[195]:= %/2
Out[195]= 2387556325
```

Therefore n-1=16*odd.

```
In[196]:= PowerMod[2, (n - 1)/16, n]
Out[196]= 1
```

The Miller-Rabin test says that n is probably prime. Now try a=3:

```
In[197]:= PowerMod[3, (n - 1)/16, n]
Out[197]= 6099632610
```

```
In[198]:= PowerMod[%, 2, n]
Out[198]= 9753514238
```

```
In[199]:= PowerMod[%, 2, n]
Out[199]= 5489404833
```

```
In[200]:= PowerMod[%, 2, n]
Out[200]= 22630222508
```

```
In[201]:= PowerMod[%, 2, n]
Out[201]= 767945134
```

Thus us 3^{n-1} mod n. It is not 1, so Miller-Rabin (or Fermat) says that n is composite.

Problem 14. (a) and (b) Let m be the message. We know mˆ3 mod n1, mod n2 ,and mod n3. By the Chinese Remainder Theorem, we know mˆ3 mod n1*n2*n3. Since m< n1, n2, n3,we have mˆ3< n1*n2*n3. Therefore we know mˆ3. Taking the cube root, we obtain m.

Part (c) We follow the steps outlined in parts (a, b).

```
In[202]:= ChineseRemainderTheorem[{359335245251,10436363975495,5135984059593},
{2469247531693,11111502225583,44444222221411}]
Out[202]= 52189581153668510460961337 5

In[203]:= %^(1/3)
Out[203]= 805121215

In[204]:= num2txt1[%]
Out[204]= hello
```

Chapter 7 - Mathematica

Problem 1.

```
In[205]:= PowerMod[3,1234,53047]
Out[205]= 8576
```

Problem 2.

We do an exhaustive search:

```
In[206]:= For[i = 1,i < 30,Print[i," ", PowerMod[3,i,31]]; i + +]

1 3

2 9

3 27

4 19

5 26

6 16

7 17

8 20

9 29

10 25

11 13

12 8

13 24

14 10

15 30

16 28

17 22
```

```
18 4

19 12

20 5

21 15

22 14

23 11

24 2

25 6

26 18

27 23

28 7

29 21
```

Therefore L_3(24)=13.

Problem 3. (a)

```
PowerMod[2,2000,3989]

3925

PowerMod[2,3000,3989]

1046
```

Part (b)

```
Mod[2000 + 3000, 3988]

1012
```

Problem 4. We follow the notation in the text. Let x=L_11(2). First we need the prime factors of 1200:

```
In[207]:= FactorInteger[1200]
Out[207]= {{2,4},{3,1},{5,2}}
```

Now we follow the Pohlig-Hellman algorithm, starting with the powers of 2:

```
In[208]:= PowerMod[2,1200/2,1201]
Out[208]= 1
```

Therefore the first binary digit of x is 0, and x=0 mod 2, and b1=2.

```
In[209]:= PowerMod[2,1200/4,1201]
Out[209]= 1
```

Therefore the second binary digit of x is 0, and x=0 mod 4, and b2=2.

```
In[210]:= PowerMod[2,1200/8,1201]
```

```
Out[210]= 1200
```

This is -1 mod 1201, so the next bit in the binary expansion is 1. Therefore x=4 mod 8.

```
In[211]:= b3 = Mod[2 * PowerMod[11, -4, 1201], 1201]
Out[211]= 729

In[212]:= PowerMod[b3, 1200/16, 1201]
Out[212]= 1200
```

This says that the next bit in the binary expansion is 1. Therefore x=4+8=12 mod 16. Now we work with 3:

```
In[213]:= PowerMod[2, 1200/3, 1201]
Out[213]= 570
```

We now need

```
In[214]:= w = PowerMod[11, 1200/3, 1201]
Out[214]= 570
```

This matches the 570 above, so x=1 mod 3. Now we work mod 5 and mod 25:

```
In[215]:= PowerMod[ 2, 1200/5, 1201]
Out[215]= 105

In[216]:= w = PowerMod[11, 1200/5, 1201]
Out[216]= 1062
```

We compute powers of w until one matches 105:

```
In[217]:= PowerMod[w, 2, 1201]
Out[217]= 105
```

This matches the 105 computed above. Therefore x=2 mod 5.

```
In[218]:= b1 = Mod[2 * PowerMod[11, -2, 1201], 1201]
Out[218]= 536

In[219]:= PowerMod[b1, 1200/25, 1201]
Out[219]= 1062
```

This is w, so the next digit in the 5-adic expansion is 1. Therefore x=2+1*5 =7 mod 25. Now put everything together with the Chinese Remainder Theorem:

```
In[220]:= ChineseRemainderTheorem[{12, 1, 7}, {16, 3, 25}]
Out[220]= 1132
```

Let’s check the answer:

```
In[221]:= PowerMod[11, 1132, 1201]
Out[221]= 2
```

Therefore L_11(2)=1132.

Chapter 8- Mathematica

Problem 1. (a)

```
1 - Product[1. - i/365, {i, 29}]
```

```
0.706316
```

The formula on page 230 gives 0.708547.

Part (b). We use the formula on page 230 to get an approximate value of r. We need

1-e^(-r^2/2N)=.99, so r^2/2N =-ln(.01):

```
λ = - log[0.01]
```

```
4.60517
```

We want r to be approximately sqrt(2*lambda*N):

```
Sqrt[2 * λ * 365]
```

```
57.9808
```

Let's try r=58:

```
1 - Product[1. - i/365, {i, 57}]
```

```
0.991665
```

Let's try r=57:

```
1 - Product[1. - i/365, {i, 56}]
```

```
0.990122
```

Let's try r=56:

```
1 - Product[1. - i/365, {i, 55}]
```

```
0.988332
```

Therefore r=57 is the number needed to have at least a .99 probability.

Part (c). We need 366 people to guarantee a match (ignore February 29).

Problem 2.

```
1. - Product[1. - i/10^4, {i, 199}]
```

```
0.86512
```

Chapter 9- Mathematica

Problem 1. (a) Since r is the same in both messages, the values of k are the same.

Part (b) We follow the procedure on page 181.

```
In[222]:= m1 = 809; m2 = 22505; s1 = 1042; s2 = 26272; r = 18357; p = 65539;
```

We want to solve (s1-s2)k=m1-m2 mod p-1. First compute a gcd:

```
In[223]:= GCD[s1 - s2, p - 1]
Out[223]= 6
```

Since the gcd is 6, there are 6 solutions to the congruence. Divide the congruence by 6 to obtain ((s1-s2)/6)k=(m1-m2)/6 mod (p-1)/6. This has the solution

```
In[224]:= Mod[((m1 - m2)/6) * PowerMod[(s1 - s2)/6, -1, (p - 1)/6], (p - 1)/6]
Out[224]= 1814
```

Therefore k=1814 mod (p-1)/6. The solutions mod p-1 are

```
In[225]:= 1814
Out[225]= 1814

In[226]:= 1814 + (p - 1)/6
Out[226]= 12737

In[227]:= 1814 + 2 * (p - 1)/6
Out[227]= 23660

In[228]:= 1814 + 3 * (p - 1)/6
Out[228]= 34583

In[229]:= 1814 + 4 * (p - 1)/6
Out[229]= 45506

In[230]:= 1814 + 5 * (p - 1)/6
Out[230]= 56429
```

We can see which is the correct value for k by computing alpha to the k-th power mod p and seeing which one gives r:

```
In[231]:= PowerMod[2, 1814, p]
Out[231]= 51656

In[232]:= PowerMod[2, 12737, p]
Out[232]= 33299

In[233]:= PowerMod[2, 23660, p]
Out[233]= 47182

In[234]:= PowerMod[2, 34583, p]
Out[234]= 13883

In[235]:= PowerMod[2, 45506, p]
Out[235]= 32240

In[236]:= PowerMod[2, 56429, p]
Out[236]= 18357
```

This last one is r, so k=56429. Now solve r*a=m1-s1*k mod p-1. First compute a gcd:

```
In[237]:= GCD[r, p - 1]
Out[237]= 3
```

There are 3 solutions to our congruence. Divide by 3 and solve:

```
In[238]:= Mod[(m1 - s1 * 56429)/3 * PowerMod[r/3, -1, (p - 1)/3], (p - 1)/3]
Out[238]= 9871
```

The solutions mod p-1 are

```
In[239]:= 9871
Out[239]= 9871

In[240]:= 9871 + (p - 1)/3
Out[240]= 31717

In[241]:= 9871 + 2 * (p - 1)/3
Out[241]= 53563
```

Raise alpha to each of these three exponents and see which gives beta:

```
In[242]:= PowerMod[2, 9871, p]
Out[242]= 33384
```

This is beta, so we stop here.The value of a is 9871.

Problem 2. First we need to find Bob's encryption exponent dB:

```
In[243]:= dB = PowerMod[87697, -1, (sigpb - 1) * (sigqb - 1)]
Out[243]= 259959042568078902255663939554592635205071473
```

The message is m1 raised to the dB power mod nb:

```
In[244]:= PowerMod[sigpairm1, dB, signb]
Out[244]= 19012507151504022505

In[245]:= num2txt1[%]
Out[245]= saygoodbye
```

The decrypted signed document is the pair (m,s), where m is obtained as above and s=mˆda mod na. To verify the signature, check that m=sˆeA mod na. First we need to find s=s1ˆdB mod nb:

```
In[246]:= s = PowerMod[sigpairs1, dB, signb]
Out[246]= 150270996499036309478023705411245214416829627
```

Now compute sˆeA:

```
In[247]:= PowerMod[s, 1571, signa]
Out[247]= 19012507151504022505
```

This matches m, so the verification works.

Problem 3. Look at s from Problem 2:

```
In[248]:= s
Out[248]= 150270996499036309478023705411245214416829627
```

This is bigger than the new value of nb:

```
In[249]:= 7865712896579 * sigqb
```

```
Out[249]= 668254407055724785349502432497421 47
```

When s1 is decrypted to find s, we only obtain s mod the new nb, rather than the full value of s. Therefore we do not have the value of s needed to use in verifying the signature.

Chapter 12 - Mathematica

Problem 1. The secret is the sum of the shares mod 2110763:

```
In[250]:= Mod[1008369 + 593647 + 631870, 2110763]
Out[250]= 123123
```

Therefore the secret is 123123.

Problem 2. Let's take the 1st, 3rd, 5th, and 7th shares:

```
In[251]:= InterpolatingPolynomial[{{1, 214}, {3, 6912}, {5, 3904}, {7, 510}}, x]
```

$$\text{Out[251]= } 214 + \left(3349 + \left(-\frac{4853}{4} + \frac{1165}{6}\,(-5+x)\right)(-3+x)\right)(-1+x)$$

Now evaluate at x=0 to find the constant term:

```
In[252]:= %/. x → 0
```

$$\text{Out[252]= } -\frac{38749}{4}$$

```
In[253]:= Mod[Numerator[%] * PowerMod[Denominator[%], -1, 8737], 8737]
Out[253]= 1234
```

The secret is 1234.

Let's try with the last 4 shares:

```
In[254]:= InterpolatingPolynomial[{{4, 8223}, {5, 3904}, {6, 3857}, {7, 510}}, x]
Out[254]= 8223 + (-4319 + (2136 - 1262 (-6 + x)) (-5 + x)) (-4 + x)

In[255]:= %/. x → 0
Out[255]= 219659

In[256]:= Mod[%, 8737]
Out[256]= 1234
```

As expected, this matches the previous answer.

Problem 3. Let's compute the secrets for the pairs AB, BC, CD, and DA:

```
In[257]:= InterpolatingPolynomial[{{38, 358910}, {3876, 9612}}, x]
```

$$\text{Out[257]= } 358910 - \frac{174649\,(-38+x)}{1919}$$

```
In[258]:= %/. x → 0
```

$$\text{Out[258]= } \frac{36599208}{101}$$

```
In[259]:= Mod[Numerator[%] * PowerMod[Denominator[%], -1, 984583], 984583]
Out[259]= 69918

In[260]:= InterpolatingPolynomial[{{3876, 9612}, {23112, 28774}}, x]
```

$$\text{Out[260]= } 9612 + \frac{9581\,(-3876+x)}{9618}$$

```
In[261]:= %/. x → 0
```

```
Out[261]= 9218710/1603

In[262]:= Mod[Numerator[%] * PowerMod[Denominator[%], -1, 984583], 984583]
Out[262]= 927070

In[263]:= InterpolatingPolynomial[{{23112, 28774}, {432, 178067}}, x]
Out[263]= 28774 - 149293 (-23112 + x)/22680

In[264]:= %/. x → 0
Out[264]= 18995621/105

In[265]:= Mod[Numerator[%] * PowerMod[Denominator[%], -1, 984583], 984583]
Out[265]= 21502

In[266]:= InterpolatingPolynomial[{{432, 178067}, {38, 358910}}, x]
Out[266]= 178067 - 180843/394 (-432 + x)

In[267]:= %/. x → 0
Out[267]= 74141287/197

In[268]:= Mod[Numerator[%] * PowerMod[Denominator[%], -1, 984583], 984583]
Out[268]= 21502
```

We suspect that the secret is 21502 and that B's share is incorrect. This can be proved as follows: The line through (0,21502) and D passes through both C and A by the above calculation(actually, we showed that the line through C and D passes through (0, 21502), which is really the same result). Therefore A, C, and D must be collinear, and B must be incorrect.

Chapter 16 - *Mathematica*

Problem 1 (a).

```
In[269]:= addell[{1, 5}, {9, 3}, 2, 3, 19]
Out[269]= {15, 8}
```

Part (b)

```
In[270]:= addell[{9, 3}, {9, -3}, 2, 3, 19]
Out[270]= {infinity, infinity}
```

Part (c). From (b) we know that (9,-3)=-(9,3), so subtracting (9,3) is the same as adding (9,-3):

```
In[271]:= addell[{1, 5}, {9, -3}, 2, 3, 19]
Out[271]= {10, 4}
```

Part (d)

```
multsell[{1, 5}, 20, 2, 3, 19]
```

```
Out[271]= {1, {1, 5}, 2, {3, 13}, 3, {12, 8}, 4, {10, 15}, 5,

            {9, 3}, 6, {15, 8}, 7, {14, 18}, 8, {5, 10}, 9,

            {11, 11}, 10, {18, 0}, 11, {11, 8}, 12, {5, 9}, 13,

            {14, 1}, 14, {15, 11}, 15, {9, 16}, 16, {10, 4}, 17,

            {12, 11}, 18, {3, 6}, 19, {1, 14}, 20,

            {infinity, infinity}}
```

This tells us that 5(1,5)=(9,3),so k=5.

Part (e). The output in part (d) shows that (1,5) has exactly 20 multiples.

Part (f). We have shown that (1,5) has order 20 (terminology of Exercise 8). Therefore, by 8(d), the number N of points on E is a multiple of 20. Hasse's theorem says that |N-20|<2*sqrt(19), so 11<N<29. The only multiple of 20 in this range is N=20.

Problem 2. To test whether a number z is a square mod p,we have two options.The easier is to use Exercise 11.1(d)and raise z to the (p-1)/2 power.The other is to use Section 3.9: raise z to the (p+1)/4 power to calculate a potential square root, then square it to see if it is actually a square root. We'll use the easier method. We successively substitute the number 12345*into xˆ3+7x+11 with*running through 0,1,2,... until we get a square mod 593899:

```
In[272]:= p = 593899
Out[272]= 593899

In[273]:= z = 123450^3 + 7 * 123450 + 11
Out[273]= 1881365964489161

In[274]:= PowerMod[z, (p - 1)/2, p]
Out[274]= 593898

In[275]:= z = 123451^3 + 7 * 123451 + 11
Out[275]= 1881411684567019

In[276]:= PowerMod[z, (p - 1)/2, p]
Out[276]= 1
```

Therefore z is a square mod p. To find a square root:

```
In[277]:= PowerMod[z, (p + 1)/4, p]
Out[277]= 423090
```

We find that (123451,423090) is a point on the curve.

Problem 3 (a). Pick a random point,say (1,1), and a random value of a, say a=123. Now choose b so that (1,1) lies on the curve yˆ2=xˆ3+a*x+b. We find b=-123. Now choose a bound, say B=30 and compute B!(1,1):

```
In[278]:= multell[{1,1},Factorial[30],123,-123,3900353]
Out[278]= {factor =,1109}
```

The other factor is

```
In[279]:= 3900353/1109
```

```
Out[279]= 3517
```

Part (b). Let's take a=2 and B=30:

```
In[280]:= PowerMod[2,Factorial[30],3900353]
Out[280]= 609067

In[281]:= GCD[%-1,3900353]
Out[281]= 1
```

This produces the trivial factorization. The factorizations of 1109-1 and 3517-1 are

```
In[282]:= FactorInteger[1108]
Out[282]= {{2,2},{277,1}}

In[283]:= FactorInteger[3516]
Out[283]= {{2,2},{3,1},{293,1}}
```

Both have large prime factors. We probably would have needed to use B=277 to find the factor 1109.

Problem 4. (a)

```
In[284]:= multell[{2,3},189,-10,21,557]
Out[284]= {infinity,infinity}

In[285]:= multell[{2,3},63,-10,21,557]
Out[285]= {38,535}

In[286]:= multell[{2,3},27,-10,21,557]
Out[286]= {136,360}
```

Part (b). The only prime factors of 189 are 3 and 7. We have shown that (189/3)P and (189/7)P are not the point at infinity. By Exercise 9, the order of P is 189.

Part (c) By Exercise 8(d), the number N of points on the elliptic curve is a multiple of 189. Hasse's theorem says that |N-558| <2*sqrt(557), so 510<N<606. The only multiple of 189 in this range is N=567.

Problem 5. The negative of (1,1) is (1,-1). See page 275. Therefore we compute

```
In[287]:= addell[{5,9},{1,-1},11,-11,593899]
Out[287]= {148475,222715}
```

Chapter 18 - *Mathematica*

Problem 1. We follow the procedure given on page 326. Start with the first vector.

```
In[288]:= Mod[{0,1,0,0,0,0,1,1,0,1,0,1,1,0,1,1,0,1,0,0,0,0,1,1}.Transpose[golay],2]
Out[288]= {0,0,0,1,0,1,1,0,0,0,0,0}
```

Since wt(s)=3,we can go to step 3. The errors are in positions 4,6,7. The corrected vector is therefore
(0,1,0,1,0,1,0,1,0,1,1,0,1,1,0,1,0,0,0,0,1,1)
The first 12 entries give the original message:
(0,1,0,1,0,1,0,1,0,1,0,1)

The second vector:

```
In[289]:= s = Mod[{0,0,1,0,1,0,1,0,1,0,1,0,0,1,0,0,1,1,1,0,1,1,0,0}.Transpose[golay],2]
Out[289]= {0,0,1,0,1,1,0,0,1,0,1,0}

In[290]:= sB = Mod[s.golayb,2]
Out[290]= {1,1,1,0,1,0,1,1,0,0,1,0}
```

Now compute the vectors s+c_jˆT. One way to obtain the jth column of a matrix M is to take the jth row of its transpose using Transpose[M][[j]]:

```
In[291]:= Mod[s + Transpose[golay][[13]],2]
Out[291]= {1,1,0,1,0,0,1,1,0,1,0,0}

In[292]:= Mod[s + Transpose[golay][[14]],2]
Out[292]= {1,0,0,0,1,1,1,1,0,0,0,1}

In[293]:= Mod[s + Transpose[golay][[15]],2]
Out[293]= {1,1,1,1,1,1,0,1,0,1,1,1}

In[294]:= Mod[s + Transpose[golay][[16]],2]
Out[294]= {0,1,0,0,0,1,0,0,0,1,0,1}

In[295]:= Mod[s + Transpose[golay][[17]],2]
Out[295]= {1,0,0,1,1,0,0,0,1,1,0,1}

In[296]:= Mod[s + Transpose[golay][[18]],2]
Out[296]= {1,1,1,1,0,1,1,0,1,0,0,1}

In[297]:= Mod[s + Transpose[golay][[19]],2]
Out[297]= {1,1,0,0,0,0,0,1,1,0,1,1}

In[298]:= Mod[s + Transpose[golay][[20]],2]
Out[298]= {0,1,0,1,1,0,1,0,0,0,1,1}

In[299]:= Mod[s + Transpose[golay][[21]],2]
Out[299]= {0,0,0,1,0,1,1,1,1,1,1,1}

In[300]:= Mod[s + Transpose[golay][[22]],2]
Out[300]= {0,0,1,1,0,0,0,1,0,0,0,1}

In[301]:= Mod[s + Transpose[golay][[23]],2]
Out[301]= {1,0,1,0,0,0,1,0,0,1,1,1}

In[302]:= Mod[s + Transpose[golay][[24]],2]
Out[302]= {0,1,1,0,1,0,1,1,1,1,0,1}
```

We now compute the vectors sB+b_jˆT. This time, we need the rows of the matrix B,so we don’t need to take a transpose.

```
In[303]:= Mod[sB + golayb[[1]],2]
Out[303]= {0,0,0,0,0,1,0,1,0,0,0,0}
```

This has weight 2,so we can stop.This says that e1=1.
The vector we just computed has nonzero entries in positions k=6 and k=8.
Therefore e18=1 and e20=1. The corrected received vector is therefore
(1,0,1,0,1,0,1,0,1,0,1,0,0,1,0,0,1,0,1,0,1,1,1,1,0,0).

The first 12 entries give the message: (1,0,1,0,1,0,1,0,1,0,1,0)

The third vector:

```
In[304]:= s = Mod[{1,1,1,0,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1}.Transpose[golay],2]
Out[304]= {0,1,1,1,1,0,0,0,1,1,1,1}

In[305]:= sB = Mod[s.golayb,2]
Out[305]= {1,0,1,1,1,1,0,1,1,1,0,0}

In[306]:= Mod[s + Transpose[golay][[13]],2]
Out[306]= {1,0,0,0,0,1,1,1,0,0,0,1}

In[307]:= Mod[s + Transpose[golay][[14]],2]
Out[307]= {1,1,0,1,1,0,1,1,0,1,0,0}

In[308]:= Mod[s + Transpose[golay][[15]],2]
Out[308]= {1,0,1,0,1,0,0,1,0,0,1,0}

In[309]:= Mod[s + Transpose[golay][[16]],2]
Out[309]= {0,0,0,1,0,0,0,0,0,0,0,0}
```

We can stop here since this vector has weight 1. We have e16=1 and e4=1. The corrected vector is (1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)

The first 12 entries give the message: (1,1,1,1,1,1,1,1,1,1,1,1)

Problem 2. Multiply the received vector by the transpose of the parity check matrix:

```
In[310]:= Mod[{0,1,1,0,0,0,1,1,0,0,0,1,0,1,0}.Transpose[hammingpc],2]
Out[310]= {1,1,0,0}
```

This is the 8th column of the parity check matrix, so the error is in the 8th entry of the received vector. The corrected received vector is

```
In[311]:= (0,1,1,0,0,0,1,0,0,0,0,1,0,1,0)
```

The first 11 entries are the message:

```
In[312]:= (0,1,1,0,0,0,1,0,0,0,0).
```

Chapter 2 - Maple

```
> Problem 1.  Let's call up the ciphertext:
> ycve;
```

"ycvejqwvhqtdtwvwu"

```
> Try all shifts and see which is an English text:
> allshifts(ycve);
```

"ycvejqwvhqtdtwvwu"
"zdwfkrxwirueuxwxv"
"aexglsyxjsvfvyxyw"
"bfyhmtzyktwgwzyzx"
"cgzinuazluxhxazay"
"dhajovbamvyiybabz"
"eibkpwcbnwzjzcbca"
"fjclqxdcoxakadcdb"
"gkdmryedpyblbedec"
"hlenszfeqzcmcfefd"
"imfotagfradndgfge"
"jngpubhgsbeoehghf"
"kohqvcihtcfpfihig"
"lpirwdjiudgqgjijh"
"mqjsxekjvehrhkjki"
"nrktyflkwfisilklj"
"osluzgmlxgjtjmlmk"
"ptmvahnmyhkuknmnl"
"qunwbionzilvlonom"
"rvoxcjpoajmwmpopn"
"swpydkqpbknxnqpqo"
"txqzelrqcloyorqrp"
"uyrafmsrdmpzpsrsq"
"vzsbgntsenqaqtstr"
"watchoutforbrutus"
"xbudipvugpscsvuvt"

```
> The plaintext was  "watch out for Brutus"

> Problem 2.  Here is the ciphertext:
```

```
>  lcll;
```

"lcllewljazlnnzmvyiylhrmhza"

```
>  frequency(lcll);
```

[2, 0, 1, 0, 1, 0, 0, 2, 1, 1, 0, 6, 2, 2, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 2, 3]

```
>  The most common letter is l, which is 7 places after e.  Try
shifting
>  back by 7:


>  shift(lcll,-7);
```

"eveexpectseggsforbreakfast"

```
>  Therefore the plaintext is  "eve expects eggs for breakfast"


>  Problem 3.
>  Let the decryption function be x=ay+b.  The plaintext "if" corresponds
>  to the
>  numbers 8, 5.  The ciphertext "ed" corresponds to 4, 3.  Therefore
>  8= 4a+b and 5= 3a+b mod 26.  Subtract to get a=3.  Then b=22.
>  Decrypt:
>  affinecrypt(edsg,3,22);
```

"ifyoucanreadthisthankateacher"

```
>  Problem 4.
>  Solve y=3x+b mod 26 for x to obtain  x=9y-9b mod 26.  Therefore,
the
>  plaintext
>  can be found by computing 9y, then trying all shifts:
>  allshifts(affinecrypt(tcab,9,0));
```

"psajpuoetlkooexehepeao"

"qtbkqvpfumlppfyfifqfbp"

"ruclrwqgvnmqqgzgjgrgcq"

"svdmsxrhwonrrhahkhshdr"

"twentysixpossibilities"

"uxfouztjyqpttjcjmjujft"

"vygpvaukzrquukdknkvkgu"

"wzhqwbvlasrvvlelolwlhv"

"xairxcwmbtswwmfmpmxmiw"

"ybjsydxncutxxngnqnynjx"

"zcktzeyodvuyyohorozoky"

"adluafzpewvzzpipspaplz"

"bemvbgaqfxwaaqjqtqbqma"

“cfnwchbrgyxbbrkrurcrnb”

“dgoxdicshzyccslsvsdsoc”

“ehpyejdtiazddtmtwtetpd”

“fiqzfkeujbaeeunuxufuqe”

“gjraglfvkcbffvovyvgvrf”

“hksbhmgwldcggwpwzwhwsg”

“iltcinhxmedhhxqxaxixth”

“jmudjoiynfeiiyrybyjyui”

“knvekpjzogfjjzszczkzvj”

“lowflqkaphgkkatadalawk”

“mpxgmrlbqihllbubebmbxl”

“nqyhnsmcrjimmcvcfcncym”

“orziotndskjnndwdgdodzn”

```
>  Therefore the plaintext is "twentysixpossibilities".
>  (The encryption function was y=3x+14 mod 26.)



>  Problem 5.
>  For example, encrypt the string "abcde" with various
>  possibilities:
>  affinecrypt("abcde",267,11);
```

“lszgn”

```
>  affinecrypt("abcde",7,11);
```

“lszgn”

```
>  affinecrypt("abcde",33,11);
```

“lszgn”

```
>  As expected, all three encryptions are the same.

>  Problem 6.  Look at the program and extract the relevant commands,
>  then modify the commands and command names (to avoid changing
the
>  existing commands):
>  numbDNA:=table([" "=0,"A"=1, "C"=2, "G"=3, "T"=4]):
>  alphDNA:=table([0=" ",1="A",2="C",3="G",4="T"]):
>  shiftDNA:=proc(txt,n)
>  local i, z;
>  z :=NULL;
>  for i from 1 while i<= length(txt) do
>  z:=cat(z,(alphDNA[(numbDNA[substring(txt,i)]+n-1 mod 4)+1]));
```

```
> end do;
> return(z);
> end:
> affinecryptDNA:= proc(txt,m,n)
> local i,z;
> z:=NULL;
> for i from 1 while i<=length(txt) do
> z:=cat(z,(alphDNA[((numbDNA[substring(txt,i)]-1)*m+n mod
> 4)+1]));
> end do;
> return(z);
> end:
> Part (a): Shift the plaintext by 1 (the plaintext has been stored
as
> DNA):
> shiftDNA(DNA,1);
```

“TCCAAGTGTTGGTGCCAACCGGGAGCGACCCTTTCAGAGACTCCGA”

```
> Part (b) Let the affine function be y=mx+n. Then we need
> gcd(m,4)=1.
> Here is an affine encryption:
> affinecryptDNA(DNA,3,2);
```

“AGGTTCACAACCACGGTTGGCCCTCGCTGGGAAAGTCTCTGAGGCT”

```
> Here is what could happen if gcd(m,4) is not 1:
> affinecryptDNA(DNA,2,1);
```

“CCCTTTCTCCTTCTCCTTCCTTTTTCTTCCCCCCCTTTTTCCCCTT”

```
> Notice that only C and T appear in the ciphertext.Both A and
G are
> encrypted to C.


> Problem 7. Let’s find the key length by computing coincidences:
> coinc(hdsf,1);
```

$$11$$

```
> coinc(hdsf,2);
```

$$14$$

```
> coinc(hdsf,3);
```

$$15$$

```
> coinc(hdsf,4);
```

$$25$$

```
> coinc(hdsf,5);
```

$$14$$

```
> coinc(hdsf,6);
```

$$14$$

```
> The key length is probably 4.
> Look at the frequencies of letters in positions 1 mod 4:
> vigvec(hdsf,4,1);
```

[0.1511627907, 0.04651162791, 0.01162790698, 0., 0.03488372093, 0.1279069767,
0.06976744186, 0.04651162791, 0., 0.02325581395, 0., 0.03488372093, 0.,
0.05813953488, 0.02325581395, 0.02325581395, 0., 0., 0.04651162791,
0.06976744186, 0.04651162791, 0.08139534884, 0., 0.01162790698,
0.04651162791, 0.04651162791]

```
> Find the dot products with the alphabet frequency vector:
> corr(%);
```

0.04480232562, 0.04589534885, 0.04211627904, 0.02969767444, 0.02893023256,
0.03900000000, 0.03788372094, 0.04943023257, 0.03425581396,
0.03486046511, 0.02626744186, 0.03639534885, 0.04437209302,
0.05266279070, 0.04212790697, 0.03518604652, 0.03032558140,
0.04002325579, 0.04568604652, 0.04191860465, 0.03903488372,
0.03456976745, 0.04048837209, 0.03704651162, 0.03465116280,
0.03337209301

```
> max(%);
```

0.05266279070

```
> The max is in the 14th position.
> Repeat with 2 mod 4:
> corr(vigvec(hdsf,4,2));
```

0.03854651164, 0.03497674419, 0.04583720930, 0.04572093024, 0.03340697675,
0.02220930232, 0.03425581395, 0.03180232560, 0.03920930232,
0.04431395349, 0.05108139533, 0.03463953489, 0.03065116280,
0.04304651163, 0.05776744186, 0.04144186046, 0.02863953489,
0.03236046511, 0.03259302326, 0.03113953489, 0.03595348837,
0.04618604651, 0.03666279068, 0.03819767444, 0.05023255815,
0.04012790698

```
> max(%);
```

0.05776744186

```
> The max is in the 15th position.
> Now do 3 mod 4:
> corr(vigvec(hdsf,4,3));
```

0.03743023255, 0.03100000000, 0.02904651161, 0.03409302325, 0.05766279069,
0.04661627907, 0.03115116279, 0.03751162791, 0.04353488372,
0.03811627908, 0.04190697673, 0.04304651164, 0.03630232557,
0.03361627904, 0.04103488371, 0.03066279070, 0.03540697674,
0.03951162791, 0.04868604651, 0.04266279071, 0.03645348836,
0.02932558139, 0.03419767441, 0.03527906976, 0.04680232560,
0.03994186046

```
> max(%);
```

0.05766279069

```
> The max is in the 5th position.
> Finally, do 4 mod 4:
> corr(vigvec(hdsf,4,4));
```

0.03876744184, 0.03673255816, 0.04488372091, 0.03934883723, 0.03661627905,
0.03538372094, 0.03866279070, 0.04505813955, 0.03274418605,
0.02795348839, 0.03808139536, 0.03723255815, 0.04345348839,
0.04288372093, 0.04825581396, 0.03013953488, 0.03227906979,
0.03496511629, 0.05722093024, 0.03334883722, 0.03070930233,
0.03168604651, 0.03786046509, 0.03770930233, 0.04167441860,
0.04734883721

```
> max(%);
```

0.05722093024

```
> The max is in the 19th position.
> Since the first position corresponds to a shift of 0, etc., the
> shifts are by 13, 14, 4, 18.  Decrypt as follows:
> vigenere(hdsf,-[13,14,4,18]);
```

“uponthisbasisiamgoingtoshowyouhowabunchofbrightyoungfolksdidfindachampi \
onamanwithboysandgirlsofhisownamanofsodominatingandhappyindividua \
litythatyouthisdrawntohimasisaflytoasugarbowlitisastoryaboutasmalltowni \
tisnotagossipyyarnnorisitadrymonotonousaccountfullofsuchcustomaryfilli \
nsasromanticmoonlightcastingmurkyshadowsdownalongwindingcountryro \
ad”

```
> Observe that the plaintext has no e's (the encryption key was
> "noes").  This is why the maxes of the dot products (around .057)
were
> not as large as they usually are (around .065).


> Problem 8.  Find the key length:
> coinc(ocwy,1);
```

13

```
> coinc(ocwy,2);
```

13

```
> coinc(ocwy,3);
```

11

```
> coinc(ocwy,4);
```

6

```
> coinc(ocwy,5);
```

15

```
> coinc(ocwy,6);
```

23

```
> coinc(ocwy,7);
```

8

```
> We guess that the key length is 6.
> Now, perform the same calculations as in Problem 7:
> corr(vigvec(ocwy,6,1));
```

0.04173076921, 0.04521153846, 0.03723076924, 0.04517307693, 0.02946153846,
0.03017307693, 0.04198076922, 0.05605769227, 0.03799999999,
0.03986538462, 0.03517307692, 0.03665384615, 0.03167307692,
0.03923076923, 0.04111538460, 0.03886538463, 0.04098076924,
0.03221153848, 0.03507692307, 0.03844230769, 0.04755769229,
0.03238461538, 0.04488461539, 0.03936538462, 0.03300000001,
0.02950000002

```
> max(%);
```

0.05605769227

```
> This occurs in the 8th position.
> corr(vigvec(ocwy,6,2));
```

0.03605769230, 0.04351923078, 0.03988461538, 0.03825000001, 0.03594230769,
0.03132692308, 0.02715384617, 0.03850000000, 0.04411538461,
0.04113461539, 0.04730769231, 0.03084615384, 0.03007692308,
0.04661538462, 0.06419230766, 0.03419230770, 0.02632692308,
0.03842307693, 0.04165384614, 0.03611538461, 0.03623076922,
0.03655769233, 0.03259615385, 0.04275000000, 0.04423076922,
0.03700000000

```
> max(%);
```

0.06419230766

```
> This occurs in the 15th position.
> corr(vigvec(ocwy,6,3));
```

0.04742307692, 0.03767307693, 0.03628846154, 0.04126923077, 0.03478846153,
0.02851923077, 0.03334615385, 0.04117307691, 0.03261538463,
0.03078846153, 0.04471153843, 0.06082692306, 0.03892307692,
0.03086538463, 0.03974999999, 0.04580769233, 0.03323076924,
0.03349999999, 0.04061538460, 0.03194230768, 0.02707692308,
0.04438461537, 0.04815384616, 0.03709615386, 0.03567307693,
0.04455769231

```
> max(%);
```

0.06082692306

```
> This occurs in the 12th position.
> corr(vigvec(ocwy,6,4));
```

0.04613461538, 0.04050000000, 0.03875000000, 0.03136538462, 0.03926923078,
0.03428846153, 0.03411538462, 0.02911538462, 0.04782692310,
0.03913461539, 0.03557692310, 0.03480769231, 0.06088461538,
0.04378846153, 0.02917307694, 0.03471153847, 0.04388461540,
0.02826923077, 0.03146153847, 0.04261538460, 0.03515384615,
0.03138461538, 0.04092307692, 0.04478846153, 0.04105769231,
0.04201923077

```
> max(%);
```

0.06088461538

```
> The max is in the 13th position.
> corr(vigvec(ocwy,6,5));
```

0.05063461539, 0.03644230769, 0.03200000000, 0.03526923078, 0.06023076924,
0.03888461539, 0.03280769234, 0.03767307693, 0.04511538463,
0.02653846153, 0.03557692309, 0.04446153845, 0.03605769231,
0.03611538462, 0.03836538462, 0.04461538460, 0.03601923077,
0.04173076923, 0.03807692308, 0.04326923076, 0.03940384617,
0.03640384616, 0.03663461540, 0.03375000001, 0.03538461539,
0.02953846155

```
> max(%);
```

0.06023076924

```
> This is in the 5th position.
> corr(vigvec(ocwy,6,6));
```

0.03125000000, 0.03286538462, 0.05228846153, 0.04723076924, 0.02580769230,
0.03286538461, 0.04905769229, 0.03924999998, 0.03101923078,
0.03248076923, 0.03978846156, 0.03011538460, 0.03840384616,
0.03576923075, 0.04038461539, 0.03948076925, 0.03482692308,
0.03721153846, 0.06369230769, 0.03859615385, 0.02744230769,
0.03482692308, 0.04853846154, 0.02938461540, 0.04171153847,
0.04671153845

```
> max(%);
```

0.06369230769

```
> This is in the 19th position.
> Since the 1st position corresponds to a shift of 0, etc, we find
that
> the shifts were by 7, 14, 11, 12, 4, 18.  The key word was "holmes".
> Decrypt:
> vigenere(ocwy,-[7,14,11,12,4,18]);
```

"holmeshadbeenseatedforsomehoursinsilencewithhislongthinbackcurvedoverache \
micalvesselinwhichhewasbrewingaparticularlymalodorousproducthishead \
wassunkuponhisbreastandhelookedfrommypointofviewlikeastrangelankbir \
dwithdullgreyplumageandablacktopknotsowatsonsaidhesuddenlyyoudonot \
proposetoinvestinsouthafricansecurities"

```
> Problem 9.  Follow the procedure of Problem 8:
> coinc(xkju,1);
```

15

```
> coinc(xkju,2);
```

17

```
> coinc(xkju,3);
```

13

```
> coinc(xkju,4);
```

10

```
> coinc(xkju,5);
```

23

```
> coinc(xkju,6);
```

5

```
> The key length is 5.
> corr(vigvec(xkju,5,1));
```

0.04445454548, 0.06836363638, 0.03883333335, 0.02962121212, 0.03821212121,
0.04113636362, 0.03493939395, 0.03122727272, 0.03575757575,
0.02996969698, 0.03569696969, 0.03707575758, 0.04924242425,
0.04007575758, 0.04218181818, 0.04062121212, 0.04740909092,
0.03937878789, 0.03136363635, 0.03133333333, 0.04090909092,
0.03313636363, 0.02993939393, 0.04409090911, 0.03571212122,
0.03031818182

```
> max(%);
```

0.06836363638

```
> This is in position 2.
> corr(vigvec(xkju,5,2));
```

0.03445454546, 0.02409090911, 0.03342424244, 0.07166666668, 0.04356060607,
0.03062121212, 0.03112121212, 0.04613636366, 0.03477272727,
0.03483333336, 0.04118181819, 0.03871212120, 0.03081818182,
0.02974242423, 0.04730303031, 0.04501515152, 0.03872727275,
0.03584848487, 0.04595454547, 0.04028787880, 0.03206060605,
0.03777272729, 0.04409090910, 0.03327272729, 0.02850000001,
0.04703030303

```
> max(%);
```

0.07166666668

```
> This is in position 4.
> corr(vigvec(xkju,5,3));
```

0.03506060606, 0.04856060606, 0.02834848486, 0.02787878789, 0.04575757577,
0.06654545455, 0.04142424243, 0.03310606063, 0.03551515152,
0.04321212121, 0.03262121215, 0.03293939395, 0.03875757577,
0.03062121213, 0.03875757575, 0.03912121214, 0.05000000001,
0.03478787880, 0.04077272726, 0.04430303031, 0.04554545456,
0.03678787878, 0.02898484850, 0.02746969696, 0.03480303031,
0.03931818182

```
> max(%);
```

0.06654545455

```
> This is in position 6.
> corr(vigvec(xkju,5,4));
```

0.03871212120, 0.02884848486, 0.03169696970, 0.04510606060, 0.03677272728,
0.02766666668, 0.03603030304, 0.07266666669, 0.04118181818,
0.03534848485, 0.02971212122, 0.04487878788, 0.03016666668,
0.03859090909, 0.03816666667, 0.03336363635, 0.02743939394,
0.03654545456, 0.05124242425, 0.04342424243, 0.04193939395,
0.03419696971, 0.05072727273, 0.04098484849, 0.03425757577,
0.03133333334

```
> max(%);
```

0.07266666669

```
> This is in position 8.
> corr(vigvec(xkju,5,5));
```

0.03268181818, 0.03069696970, 0.03884848486, 0.03684848485, 0.03434848485,
0.04368181821, 0.03437878790, 0.03693939394, 0.03853030306,
0.06718181820, 0.03812121213, 0.03360606061, 0.03331818182,
0.04199999999, 0.02834848486, 0.03598484850, 0.03824242425,
0.03243939395, 0.03787878789, 0.03933333332, 0.04446969698,
0.03887878789, 0.03959090911, 0.04400000002, 0.04306060609,
0.03759090910

```
> max(%);
```

0.06718181820

```
> This is in the 10th position.
> The key is [1,3,5,7,9]. Decrypt:
> vigenere(xkju,-[1,3,5,7,9]);
```

"wheninthecourseofhumaneventsitbecomesnecessaryforonepeopletodissolvethepo \
liticalbandswhichhaveconnectedthemwithanotherandtoassumeamongthepo \
wersoftheearththeseparateandequalstationtowhichthelawsofnatureandofnat \
uresgodentitlethemadecentrespecttotheopinionsofmankindrequiresthatthey \
shoulddeclarethecauseswhichimpelthemtotheseparation"

```
> (this is the start of the Declaration of Independence)

> Problem 10.  Change the ciphertext to numbers.  Since text2num
uses
> a=1 and z=26, we shift back by 1 first.  (This whole process
could be
> done by hand; but the present method helps avoid errors.  However,
the
> a's in the 15th and 21st position become 26 instead of 0.)
> text2num(shift("zirkzwopjjoptfapuhfhadrq",-1));
```

25081710252214150909141519052615200705072603171 6

```
> Invert the matrix:
> M:=matrix(4,4,[1,2,3,4,4,3,2,1,11,2,4,6,2,9,6,4]);
```

$$M := \begin{bmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 \\ 11 & 2 & 4 & 6 \\ 2 & 9 & 6 & 4 \end{bmatrix}$$

```
> invM:=map(x->x mod 26, inverse(M));
```

$$invM := \begin{bmatrix} 6 & 18 & 19 & 0 \\ 6 & 22 & 22 & 1 \\ 1 & 12 & 3 & 24 \\ 8 & 21 & 8 & 1 \end{bmatrix}$$

```
> Break the ciphertext numbers into blocks of 4 and multiply by
> invM:
> map(x-> x mod 26,evalm([25,8,17,10]&*invM));
```

[9, 0, 2, 10]

```
> map(x-> x mod 26,evalm([25,22,14,15]&*invM));
```

[0, 13, 3, 9]

```
> map(x-> x mod 26,evalm([9,9,14,15]&*invM));
```

[8, 11, 11, 22]

```
> map(x-> x mod 26,evalm([19,5,0,15]&*invM));
```

[4, 13, 19, 20]

```
> map(x-> x mod 26,evalm([20,7,5,7]&*invM));
```

$$[15, 19, 7, 4]$$

```
> map(x-> x mod 26,evalm([0,3,17,16]&*invM));
```

$$[7, 8, 11, 11]$$

```
> Change back to letters, reversing the above procedure (and using
26
> in place of 0):
> shift(num2text(092602102613030908111122041319201519070407081111),1);
```

"jackandjillwentupthehill"

```
> Problem 11.  Find the length:
> lfsrlength(L101,10);
```

$$[1, 1]$$
$$[2, 1]$$
$$[3, 1]$$
$$[4, 0]$$
$$[5, 1]$$
$$[6, 1]$$
$$[7, 0]$$
$$[8, 0]$$
$$[9, 0]$$
$$[10, 0]$$

```
> The length is 6.
> lfsrsolve(L101,6);
```

$$[1, 1, 0, 1, 1, 0]$$

```
> The recurrence is x_{n+6}=x_n + x_{n+1} + x_{n+3} + x_{n+4}
> mod 2.
```

```
> Problem 12.  Find the length:
> lfsrlength(L100,15);
```

$$[1, 1]$$
$$[2, 0]$$
$$[3, 1]$$
$$[4, 0]$$
$$[5, 0]$$
$$[6, 1]$$
$$[7, 0]$$

[8, 1]

[9, 0]

[10, 0]

[11, 0]

[12, 0]

[13, 0]

[14, 0]

[15, 0]

```
> The length is 8.
> lfsrsolve(L100,8);
```

[1, 1, 0, 0, 1, 0, 0, 0]

```
> These are the coefficients of the recurrence.


> Problem 13.  Here is the ciphertext:
> L011;
```

[0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1]

```
> XOR the first 15 terms with the plaintext to get the LFSR output:
> [0,1,1,0,0,0,1,0,1,0,1,1,1,0,0]+[1,0,0,1,0,0,1,0,0,1,0,0,1,0,0]
mod
> 2;
```

[1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0]

```
> Find the length of the recurrence:
> lfsrlength(%,8);
```

[1, 1]

[2, 0]

[3, 0]

[4, 1]

[5, 1]

[6, 0]

[7, 0]

[8, 0]

```
> The length is 5.
> lfsrsolve(%%,5);
```

[1, 1, 0, 0, 1]

```
>  Now generate the LFSR output using these coefficients and the
first
>  five terms of the LFSR output:
>  lfsr([1,1,0,0,1],[1,1,1,1,0],50);
```

[1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1]

```
>  XOR this with the ciphertext to get the plaintext:
>  % + L011 mod 2;
```

[1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0]

```
>  This is the plaintext.
```

Chapter 3 - Maple

```
> Problem 1.
> gcd(8765,23485);
```

$$5$$

```
> Problem 2 (a)
> igcdex(65537,3511,'x','y');x;y;
```

$$1$$
$$-1405$$
$$26226$$

```
> The gcd is 1, x=-1405, y=26226
> Part (b)
> 17*x; 17*y;
```

$$-23885$$
$$445842$$

```
> x:='x';y:='y';
```

$$x := x$$
$$y := y$$

```
> Problem 3.
> 3&^1234567 mod 100000;
```

$$40587$$

```
> Problem 4.
> solve(314*x=271) mod 11111;
```

$$10298$$

```
> Problem 5.  There are 6 solutions because
> gcd(216,606);
```

$$6$$

```
> Divide the congruence by 6 and solve:
> solve(36*x=11) mod 101;
```

$$48$$

```
> This is the solution mod 101.  The solutions mod 606 are
> 48; 48+101; 48+202; 48+303; 48+404; 48+505;
```

$$48$$
$$149$$
$$250$$
$$351$$
$$452$$
$$553$$

```
> Problem 6.
```

```
> chrem([17,18,19],[101,201,301]);
```

$$61122$$

```
> Problem 7.
> 2&^390 mod 391;
```

$$285$$

```
> By Euler's theorem, j=phi(391) should work:
> phi(391);
```

$$352$$

```
> 2&^352 mod 391;
```

$$1$$

```
> Problem 8.
> Pick a random x, say x=123.
> chrem([123,-123],[84047,65497]);
```

$$5495329171$$

```
> This is y.  By construction, x^2 = y^2 mod each prime factor,
> therefore mod n:
> %&^2 mod (84047*65497);
```

$$15129$$

```
> 123^2;
```

$$15129$$

```
> We see that x^2 = y^2 mod n.
> Problem 9.  We use the procedure of Exewrcise 10:
> ifactor(65537-1);
```

$$(2)^{16}$$

```
> Therefore, 2 is the only prime factor of p-1.
> 3&^(65536/2) mod 65537;
```

$$65536$$

```
> This is not 1 mod p.  By Exercise 10, 3 is a primitive root mod
p.
> Problem 10.  (a)
> M:=matrix(3,3,[1,2,4,1,5,25,1,14,196]);
```

$$m := \begin{bmatrix} 1 & 2 & 4 \\ 1 & 5 & 25 \\ 1 & 14 & 196 \end{bmatrix}$$

```
> inverse(M);
```

$$\begin{bmatrix} \frac{35}{18} & \frac{-28}{27} & \frac{5}{54} \\ \frac{-19}{36} & \frac{16}{27} & \frac{-7}{108} \\ \frac{1}{36} & \frac{-1}{27} & \frac{1}{108} \end{bmatrix}$$

```
> map(x->x mod 101,%);
```

$$\begin{bmatrix} 30 & 85 & 88 \\ 64 & 38 & 100 \\ 87 & 86 & 29 \end{bmatrix}$$

```
> Part (b).  M has an inverse mod p exactly when p does not divide
> det(M).
> det(M);
```

$$324$$

```
> ifactor(324);
```

$$(2)^2\ (3)^4$$

```
> Therefore, M  is not invertible mod 2 and mod 3.
> Problem 11.  We use the procedure of Section 3.9:
> 26055&^((34807+1)/4) mod 34807;
```

$$33573$$

```
> This is one square root.  The other is
> -33573 mod 34807;
```

$$1234$$

```
> Problem 12.  The number 2325781n is composite:
> ifactor(2325781);
```

$$(523)\ (4447)$$

```
> Find a square root mod each prime factor by the method of Section
> 3.9:
> 1522756&^(524/4) mod 523;
```

$$335$$

```
> 1522756&^(4448/4) mod 4447;
```

$$1234$$

```
> Now combine the square roots with the Chinese Remainder Theorem:
> chrem([335,1234],[523,4447]);
```

$$437040$$

```
> chrem([-335,1234],[523,4447]);
```

$$1234$$

```
> chrem([335,-1234],[523,4447]);
```

$$2324547$$

```
> chrem([-335,-1234],[523,4447]);
```

$$1888741$$

```
> These are the four square roots.
> Problem 13.
> 48382&^(83988/4)mod 83987;
                          60555
> 60555&^2 mod 83987;
                          35605
> This is the negative of 48382 mod 83987:
> -48382 mod 83987;
                          35605
> Therefore, we found the square root of -48382 mod 83987.
```

Chapter 6 - Maple

```
> Problem 1.  Encrypt each of the two messages and see which one
it
> is:
> text2num("one") &^6551 mod 712446816787;
```

$$273095689186$$

```
> text2num("two") &^6551 mod 712446816787;
```

$$709427776011$$

```
> Therefore the plaintext was "one".

> Problem 2.  Use the universal exponent factoring method.  First,
let's
> store n:
> n:=718548065973745507;
```

$$n := 718548065973745507$$

```
> Compute e*d-1:
> 3449*543546506135745129-1;
```

$$1874691899662184949920$$

```
> Express it as a power of 2 times an odd number:
> %/8;
```

$$234336487457773118740$$

```
> %/4;
```

$$58584121864443279685$$

```
> We see that e*d-1 is 32 times an odd number.
> b0:=%;
```

$$b0 := 58584121864443279685$$

```
> Pick a random number, say 2, and raise it to the b0 power:
> 2&^b0 mod n;
```

$$511846277294206136$$

```
> Successively square until we get 1 mod n:
> %&^2 mod n;
```

$$551183936117094424$$

```
> %&^2 mod n;
```

$$576566739470926048$$

```
> %&^2 mod n;
```

$$1$$

```
> The last number before the 1 was not -1 mod n, so we use the
gcd to
> factor n:
> gcd(%%-1,n);
```

$$740876531$$

```
> This is one of the factors of n.  The other is
> n/%;
```

$$969862097$$

```
> Problem 3.
> p:=nextprime(rand(10^29..10^30)());
```

$$p := 769081321110693270343632599093$$

```
> q:=nextprime(rand(10^29..10^30)());
```

$$q := 343563558458718976746753303637$$

```
> n:=p*q;
```

$$n := 264228315424922488460852715316219596666650690368066519801241$$

```
> Let's choose a random 10-digit prime as the encryption exponent:
> e:=nextprime(rand(10^9..10^10)());
```

$$e := 6062222113$$

```
> Just to be sure:
> gcd(e,(p-1)*(q-1));
```

$$1$$

```
> Encrypt:
> text2num("cat")&^e mod n;
```

$$221108395229623543553316522699162115906870683738440529748061$$

```
> text2num("bat")&^e mod n;
```

$$899421085504261939456198062304525371248333332392508946072l3$$

```
> text2num("hat")&^e mod n;
```

$$981601443644283760613822426595873393897220813183577026593l1$$

```
> text2num("encyclopedia")&^e mod n;
```

$$836441293697217139261469836467935233222364084247624476l0688$$

```
> text2num("antidisestablishmentarianism")&^e mod n;
```

$$200243403124679637361892833093193955827052809600006148320576$$

```
> There is no obvious relation between the ciphertexts and the
> plaintexts.

> Problem 4.  Let's choose our bound to be B=50 and let a=2:
> 2&^factorial(50) mod 618240007109027021;
```

$$394571778315865637$$

```
> gcd(%-1,618240007109027021);
```

$$250387201$$

```
> The other factor is
> 618240007109027021/%;
```

$$2469135821$$

```
> The reason this worked is
```

```
> ifactor(%%-1);
```

$$(2)^8\ (3)^5\ (5)^2\ (7)\ (23)$$

```
> The first prime factor p is such that p-1 has only small prime
> factors.


> Problem 5.  We'll use B=100 and a=2:
> 2&^factorial(100) mod n1;
```

$$961623821657800055077023229270715484248143$$

```
> gcd(%-1,n1);
```

$$36443898921682796544 0001$$

```
> The other factor is
> n1/%;
```

$$2424242424246868690 7$$

```
> Problem 6.  We use the Basic Principle:
> gcd(85975324443166-462436106261, 537069139875071);
```

$$9876469$$

```
> The other factor is
> 537069139875071/%;
```

$$54378659$$

```
> Problem 7.  Choose a random x, say x=123.  Do the following:
> chrem([123,-123],[985739879,1388749507]);
```

$$312089563232070143$$

```
> The square of this is congruent to the square of 123 mod each
prime
> factor, therefore mod their product:
> 123&^2 mod 985739879*1388749507 ;
```

$$15129$$

```
> 312089563232070143&^2 mod 985739879*1388749507;
```

$$15129$$

```
> However, by construction, 5495329171 is not congruent to 123
or
> -123.


> Problem 8.  (a)
> gcd(33335-670705093,670726081);
```

54323

```
> The other factor is
> 670726081/%;
```

12347

```
> Part (b) Observe that 3 = -670726078 mod 670726081.
> Let's try the gcd anyway:
> gcd(3-670726078,670726081);
```

1

```
> So we get the trivial factorization.

> Problem 9.  This is what happens in the exponent factorization
method.
> We see that 1488665^2 is 1 mod 3837523, so we use the gcd to
> factor:
> gcd(1488665-1,3837523);
```

3511

```
> The other factor is
> 3837523/%;
```

1093

```
> Problem 10.  (a) Find the square root of n.  The next prime will
be a
> factor and the previous prime will be the other factor.
> Part (b)
> n:=10993522499;
```

$n := 10993522499$

```
> q:=nextprime(round(sqrt(n)));
```

$q := 104851$

```
> p:=n/q;
```

$p := 104849$

```
> d:=1/113 mod ((p-1)*(q-1));
```

$d := 5545299377$

```
> 10787770728&^d mod n;
```

5011925

```
> num2text(%);
```

"easy"

```
> Part(c) We need higher precision here for calculating a square
> root:
> Digits := 50;
```

$Digits := 50$

```
> q:=nextprime(round(sqrt(naive)));
```

$$q := 1234567890000003141592650014 3$$

```
> Let's reset to the default precision:
> Digits := 10;
```

$$Digits := 10$$

```
> p:=naive/q;
```

$$p := 1234567890000003141592650003 1$$

```
> d:=1/9007 mod ((p-1)*(q-1));
```

$$d := 660462771866258534689069380246851318997849360492799256 83$$

```
> cnaive &^d mod naive;
```

$$2008091900142113020518002301190014152000190503211805$$

```
> num2text(%);
```

"this number was not secure"

```
> Problem 11.(a)
> p:=123456791; q:=987654323; e:=127; m:=14152019010605;
```

$$p := 123456791$$

$$q := 987654323$$

$$e := 127$$

$$m := 14152019010605$$

```
> m&^e mod p;
```

$$104120308$$

```
> m&^e mod q;
```

$$812538893$$

```
> c:=chrem([%%,%],[p,q]);
```

$$c := 18001903071439991$$

```
> Part (b) Let's change 1 digit by adding 100000:
> m&^e mod p + 100000;
```

$$104220308$$

```
> m^e mod q;
```

$$812538893$$

```
> f:=chrem([%%,%],[p,q]);
```

$$f := 17982149984979991$$

```
> gcd(f-c,p*q);
```

$$987654323$$

```
> This found the factor q because f=c mod q but f is not congruent
to c
> mod p.  Therefore gcd(f-c,pq)=q.
```

```
>  Problem 12.
>  p:=76543692179; q:=343434343453; e:=457;
```

$$p := 76543692179$$

$$q := 343434343453$$

$$e := 457$$

```
>  We'll try all possibilities for the last digit until we get a
>  meaningful message.  It will be easiest to append a 0 to the
received
>  text, then add 0, 1, 2, ...  and decrypt.
>  c:=23043293280169369471950;
```

$$c := 2304329328016936947195 0$$

```
>  The decryption exponent is
>  d:=1/e mod ((p-1)*(q-1));
```

$$d := 1553104555909567360609$$

```
>  Now try to decrypt the various possibilities.  Note that the
incorrect
>  ones will probably contain 2-digit blocks larger than 26, hence
they
>  cannot be changed back to letters:
>  (c+0)&^d mod p*q;
```

$$4174116046631355447474$$

```
>  (c+1)&^d mod p*q;
```

$$8579710266419129803917$$

```
>  (c+2)&^d mod p*q;
```

$$1913091205$$

```
>  This one can be changed back to letters:
>  num2text(%);
```

"smile"

```
>  Therefore the missing digit was 2 and the plaintext was "smile".


>  Problem 13.
>  n:=38200901201;
```

$$n := 38200901201$$

```
>  Write n-1 as a power of 2 times an odd number:
>  (n-1)/8;
```

$$4775112650$$

```
>  %/2;
```

$$2387556325$$

```
>  Therefore n-1 = 16*odd.
>  2&^((n-1)/16) mod n;
```

1

```
>  The Miller-Rabin test says that n is probably prime.  Now try
a=3:
>  3&^((n-1)/16) mod n;
```

6099632610

```
>  %&^2 mod n;
```

9753514238

```
>  %&^2 mod n;
```

5489404833

```
>  %&^2 mod n;
```

22630222508

```
>  %&^2 mod n;
```

767945134

```
>  This is 3^(n-1) mod n.  It is not 1, so Miller-Rabin (or Fermat)
says
>  that n is composite.


>  Problem 14.  (a) and (b) Let m be the message.  We know m^3 mod
n1, mod
>  n2, and mod n3.  By the Chinese Remainder Theorem, we know m^3
mod
>  n1*n2*n3.  Since m< n1, n2, n3, we have m^3 < n1*n2*n3.  Therefore
we
>  know m^3.  Taking the cube root, we obtain m.


>  Part (c).  We follow the steps outined in parts (a, b).
>  chrem([359335245251,10436363975495,5135984059593],[2469247531693,1111
>  1502225583,44444222221411]);
```

$$521895811536685104609613375$$

```
>  %^(1/3);
```

$$521895811536685104609613375^{(1/3)}$$

```
>  simplify(%);
```

805121215

```
>  num2text(%);
```

"hello"

Chapter 7 - Maple

```
> Problem 1.
> 3&^1234 mod 53047;
```

8576

```
> Problem 2.  We do an exhaustive search:
> for i from 1 to 29 do print(i, 3&^i mod 31) end do;
```

1, 3
2, 9
3, 27
4, 19
5, 26
6, 16
7, 17
8, 20
9, 29
10, 25
11, 13
12, 8
13, 24
14, 10
15, 30
16, 28
17, 22
18, 4
19, 12
20, 5
21, 15
22, 14
23, 11
24, 2
25, 6
26, 18
27, 23
28, 7
29, 21

```
> Therefore, L_3(24)=13.
```

```
>  Problem 3.(a)
>  2&^2000 mod 3989; 2&^3000 mod 3989;
```

$$3925$$

$$1046$$

```
>  Part (b)
>  2000+3000 mod 3988;
```

$$1012$$

```
>  Problem 4.  We follow the notation in the text.  Let x =L_11(2).
First,
>  we need to know the prime factors of 1200:
>  ifactor(1200);
```

$$(2)^4\ (3)\ (5)^2$$

```
>  Now we follow the Pohlig-Hellman algorithm, starting with the
powers
>  of 2.
>  2&^(1200/2) mod 1201;
```

$$1$$

```
>  Therefore the first binary bit of x is 0, and x = 0 mod 2, and
>  b1=2.
>  2&^(1200/4) mod 1201;
```

$$1$$

```
>  Therefore the second binary bit of x is 0, and x=0 mod 4, and
>  b2=2.
>  2&^(1200/8) mod 1201;
```

$$1200$$

```
>  This is -1 mod 1201, so the next bit in the binary expansion
is 1.
>  Therefore x= 4 mod 8.
>  b3:= 2/11^4 mod 1201;
```

$$b3 := 729$$

```
>  b3&^(1200/16) mod 1201;
```

$$1200$$

```
>  This says that the next bit in the binary expansion is 1.
>  Therefore x= 4+8=12 mod 16.
>  Now we work mod 3:
>  2&^(1200/3) mod 1201;
```

$$570$$

```
>  We now need
>  w:=11&^(1200/3) mod 1201;
```

$$w := 570$$

```
> This matches the 570 above.  Therefore x= 1 mod 3.
> Now we work mod 5 and 25:
> 2&^(1200/5) mod 1201;
```

$$105$$

```
> w:=11&^(1200/5) mod 1201;
```

$$w := 1062$$

```
> We compute powers of w until one matches 105:
> w^2 mod 1201;
```

$$105$$

```
> This matches the 105 computed above.  Therefore x= 2 mod 5.
> b1:=2/11^2 mod 1201;
```

$$b1 := 536$$

```
> b1&^(1200/25) mod 1201;
```

$$1062$$

```
> Therefore x=2+1*5 = 7 mod 25.
> Now put everything together with the Chinese Remainder Theorem:
> chrem([12, 1, 7],[16,3,25]);
```

$$1132$$

```
> Let's check the answer:
> 11&^1132 mod 1201;
```

$$2$$

```
> Therefore, L_11(2)=1132.
```

Chapter 8 - Maple

```
> Problem 1.  (a)
> 1-mul(1.-i/365, i=1..29);
```

$$0.7063162428$$

```
> The formula on page 230 gives 0.708547.
> Part (b) Use the formula (8.1) to get an approximate value of
r.  We
> need 1-e^(-r^2/2N)=.99, so r^2/2N= -ln(.01):
> lambda:=-ln(.01);
```

$$\lambda := 4.605170186$$

```
> We want r to be approximately sqrt(2*lambda*N):
> sqrt(2*lambda*365);
```

$$57.98080920$$

```
> Let's try r=58:
> 1-mul(1.-i/365,i=1..57);
```

$$0.9916649794$$

```
> Let's try r=57:
> 1-mul(1.-i/365,i=1..56);
```

$$0.9901224593$$

```
> Let's try r=56:
> 1-mul(1.-i/365,i=1..55);
```

$$0.9883323549$$

```
> Therefore r=57 is the number needed to have at least .99
> probability.
> Part (c).  We need 366 people to guarantee a match (ignore February
> 29).


> Problem 2.
> 1-mul(1.-i/10^4,i=1..199);
```

$$0.8651196384$$

Chapter 9 - Maple

```
> Problem 1.  (a) Since r is the same in both meassages, the values
of k
> are the same.
> Part (b) We follow the procedure on page 181.
> m1:=809; m2:=22505; s1:=1042; s2:=26272; r:=18357; p:=65539;
```

$$m1 := 809$$
$$m2 := 22505$$
$$s1 := 1042$$
$$s2 := 26272$$
$$r := 18357$$
$$p := 65539$$

```
> We want to solve (s1-s2)k=m1-m2 mod p-1.  First, compute a gcd:
> gcd(s1-s2,p-1);
```

$$6$$

```
> Since the gcd is 6, there are 6 solutions to the congruence.
Divide
> the congruence we want to solve by 6 to obtain ((s1-s2)/6)k =
> (m1-m2)/6 mod (p-1)/6.  This has the solution
> (m1-m2)/(s1-s2) mod (p-1)/6;
```

$$1814$$

```
> Therefore k=1814 mod (p-1)/6.  The solutions mod p-1 are
> 1814; 1814+(p-1)/6; 1814+2*(p-1)/6; 1814+3*(p-1)/6; 1814+4*(p-1)/6;
> 1814+5*(p-1)/6;
```

$$1814$$
$$12737$$
$$23660$$
$$34583$$
$$45506$$
$$56429$$

```
> We can see which is the correct value for k by computing alpha
to the
> k-th power mod p and seeing which one gives r:
> 2&^1814 mod p;
```

$$51656$$

```
> 2&^12737 mod p;
```

$$33299$$

```
> 2&^23660 mod p;
```

$$47182$$

```
> 2&^34583 mod p;
```

$$13883$$

```
>  2&^45506 mod p;
```

32240

```
>  2&^56429 mod p;
```

18357

```
>  This last one is r, so k=56429.
>  Now solve r*a=m1-s1*k mod p-1.  First, compute a gcd:
>  gcd(r,p-1);
```

3

```
>  There are three solutions to our congruence.  Divide by 3 and
>  solve:
>  (m1-s1*56429)/r mod (p-1)/3;
```

9871

```
>  The solutions mod p-1 are
>  9871; 9871+(p-1)/3; 9871+2*(p-1)/3;
```

9871
31717
53563

```
>  Raise alpha to each of these exponents and see which gives beta:
>  2&^9871 mod p;
```

33384

```
>  This is beta, so we stop here.  The value of a is 9871.

>  Problem 2.  First, we need to find Bob's decryption exponent
dB:
>  dB:=1/87697 mod ((sigpb-1)*(sigqb-1));
```

$dB := 259959042568078902255663939554592635205071473$

```
>  The message is m1 raised to the dB power mod nb:
>  sigpairm1&^dB mod signb;
```

19012507151504022505

```
>  num2text(%);
```

"saygoodbye"

```
>  The decrypted signed document is the pair (m,s), where m is obtained
>  above and s=m^dA mod na.  To verify the signature, check that
m = s^eA
>  mod na:
>  First, we need to find s= s1^dB mod nb:
>  s:=sigpairs1&^dB mod signb;
```

$s := 150270996499036309478023705411245214416829627$

```
>  Now compute s^eA:
>  s&^1571 mod signa;
```

19012507151504022505

```
>  This matches m, so the verification works.


>  Problem 3.  Look at s from Problem 2:
>  s;
```

$$150270996499036309478023705411245214416829627$$

```
>  This is bigger than the new value of nb:
>  7865712896579*sigqb;
```

$$6682544070557247853495024324974214 7$$

```
>  When s1 is decrypted to find s, we only obtain s mod the new
nb,
>  rather than the full value of s.  Therefore we do not have the
value of
>  s needed to use in verifying the signature.
```

Chapter 12 - Maple

```
> Problem 1.
> The secret is the sum of the shares  mod 2110763:
> 1008369+593647+631870 mod 2110763;
```

$$123123$$

```
> Therefore the secret is 123123.

> Problem 2.  Let's take the 1st, 3rd, 5th, and 7th shares:
> interp([1, 3, 5, 7],[214, 6912, 3904, 510],x);
```

$$\frac{1165}{6}\,x^3 - \frac{11843}{4}\,x^2 + \frac{76007}{6}\,x - \frac{38749}{4}$$

```
> eval(%,x=0);
```

$$\frac{-38749}{4}$$

```
> % mod 8737;
```

$$1234$$

```
> The secret is 1234.
> Let's try with the last 4 shares:
> interp([4,5,6,7],[8223,3904,3857,510],x);
```

$$-1262\,x^3 + 21066\,x^2 - 116931\,x + 219659$$

```
> eval(%,x=0);
```

$$219659$$

```
> % mod 8737;
```

$$1234$$

```
> As expected, this matches the previous answer.

> Problem 3.
> Let's compute the secrets for the pairs AB, BC, CD, and DA:
> eval(interp([38, 3876],[358910,9612],x),x=0) mod 984583;
```

$$69918$$

```
> eval(interp([3876,23112],[9612,28774],x),x=0) mod 984583;
```

$$927070$$

```
> eval(interp([23112,432],[28774,178067],x),x=0) mod 984583;
```

$$21502$$

```
> eval(interp([432,38],[178067,358910],x),x=0) mod 984583;
```

$$21502$$

```
>  We suspect that the secret is 21502, and that B's share is incorrect.
>  This can be proved as follows.  The line through (0,21502) and
D passes
>  through both C and A by the above calculation (actually, we showed
>  that the line through C and D passes through (0,21502), which
is
>  really the same result).  Therefore, A,C,D are collinear, and
B must be
>  incorrect.
```

Chapter 16 - Maple

```
> Problem 1.(a)
> addell([1,5],[9,3],2,3,19);
```

[15, 8]

```
> Part (b)
> addell([9,3],[9,-3],2,3,19);
```

["infinity", "infinity"]

```
> Part (c) From (b), we know that (9,-3)=-(9,3), so subtracting
(9,3)
> is the same as adding (9,-3):
> addell([1,5],[9,-3],2,3,19);
```

[10, 4]

```
> Part (d)
> multsell([1,5],20,2,3,19);
```

[[1, [1, 5]], [2, [3, 13]], [3, [12, 8]], [4, [10, 15]], [5, [9, 3]], [6, [15, 8]], [7, [14, 18]], [8, [5, 10]], [9, [11, 11]], [10, [18, 0]], [11, [11, 8]], [12, [5, 9]], [13, [14, 1]], [14, [15, 11]], [15, [9, 16]], [16, [10, 4]], [17, [12, 11]], [18, [3, 6]], [19, [1, 14]], [20, ["infinity", "infinity"]]]

```
> This tells us that 5(1,5)=(9,3), so k=5.
> Part (e) The output in part (d) shows that (1,5) has exactly
20
> multiples.
> Part (f) We have shown that (1,5) has order 20 (terminology of
> Exercise 8).  Therefore, by 8(d), the number N of points on E
is a
> multiple of 20.  Hasse's theorem says that |N-20|< 2*sqrt(19),
so 11 <
> N < 29.  The only multiple of 20 in this range is N=20.


> Problem 2.  To test whether a number z is a square mod p, we
have two
> options.  The easier is to use Exercise 11.1(d) and raise z to
the
> (p-1)/2 power.  The other is to use Section 3.9:  raise z to
the (p+1)/4
> power to calculate a potential square root, then square it to
see if
> it is actually a square root.  We'll use the easier method.  We
> successively substitute the number 12345* into x^3 +7x +11 with
*
> running through 0,1,2,...  until we get a square mod 593899:
> p:=593899;
```

$$p := 593899$$

```
>  z:=123450^3+7*123450+11;
```

$$z := 1881365964489161$$

```
>  z&^((p-1)/2) mod p;
```

$$593898$$

```
>  z:=123451^3+7*123451+11 mod p;
```

$$z := 426106$$

```
>  z&^((p-1)/2) mod p;
```

$$1$$

```
>  Therefore z is a square mod p.  To find a square root:
>  z&^((p+1)/4) mod p;
```

$$423090$$

```
>  We find that (123451,423090) is a point on the curve.

>  Problem 3 (a) Pick a random point, say (1,1), and a random value
of
>  a, say a=123.  Now choose b so that (1,1) lies on the curve y^2=x^3
+
>  a*x +b.  We find b=-123.  Now choose a bound, say B=30 and compute
>  B!(1,1):
>  multell([1,1],factorial(30),123,-123,3900353);
```

$$[\text{“factor=”}, 1109]$$

```
>  The other factor is
>  3900353/1109;
```

$$3517$$

```
>  Part (b) Let's take a=2 and B=30:
>  2&^factorial(30) mod 3900353;
```

$$609067$$

```
>  gcd(%-1,3900353);
```

$$1$$

```
>  This produces the trivial factorization.  The factorizations
of 1109-1
>  and 3517-1 are
>  ifactor(1108);
```

$$(2)^2\ (277)$$

```
>  ifactor(3516);
```

$$(2)^2\ (3)\ (293)$$

```
>  Both have large prime factors.  We probably would have needed
to use
>  B=277 to find the factor 1109.

>  Problem 4.  (a)
```

```
>  multell([2,3],189,-10,21,557);
```

["infinity", "infinity"]

```
>  multell([2,3],63,-10,21,557);
```

[38, 535]

```
>  multell([2,3],27,-10,21,557);
```

[136, 360]

```
>  Part (b) The only prime factors of 189 are 3 and 7.  We have
shown
>  that (189/3)P and (189/7)P are not the point at infinity.  By
Exercise
>  9, the order of P is 189.
>  Part (c) By Exercise 8(d), the number N of points on the elliptic
>  curve is a multiple of 189.  Hasse's theorem says that |N-558|
<
>  2*sqrt(557), so 510< N <606.  The only multiple of 189 in this
range is
>  N=567.

>  Problem 5.  The negative of (1,1) is (1,-1).  See page 275.  Therefore
>  we compute
>  addell([5,9],[1,-1],11,-11,593899);
```

[148475, 222715]

Chapter 18 - Maple

```
> Problem 1.  We follow the procedure given on page 326.  Start
with the
> first vector.
> s:=map(x->x mod
> 2,evalm([0,1,0,0,0,0,1,1,0,1,0,1,1,0,1,1,0,1,0,0,0,0,1,1]&*transpose(g
> olay)));
```

$$s := [0,\ 0,\ 0,\ 1,\ 0,\ 1,\ 1,\ 0,\ 0,\ 0,\ 0,\ 0]$$

```
> Since wt(s)=3, we can go to step 3.  The errors are in positions
4, 6,
> 7.  The corrected vector is therefore
> (0,1,0,1,0,1,0,1,0,1,0,1,1,0,1,1,0,1,0,0,0,0,1,1)
> The first 12 entries give the original message:
> (0,1,0,1,0,1,0,1,0,1,0,1)


> The second vector:
> s:=map(x->x mod
> 2,evalm([0,0,1,0,1,0,1,0,1,0,1,0,0,1,0,0,1,1,1,0,1,1,0,0]&*transpose(g
> olay)));
```

$$s := [0,\ 0,\ 1,\ 0,\ 1,\ 1,\ 0,\ 0,\ 1,\ 0,\ 1,\ 0]$$

```
> sB:=map(x->x mod 2, evalm(s&*golayb));
```

$$sB := [1,\ 1,\ 1,\ 0,\ 1,\ 0,\ 1,\ 1,\ 0,\ 0,\ 1,\ 0]$$

```
> Now compute the vectors s+c_j^T. Use the command col(M,j) to
obtain
> the j-th column of the matrix M.
> map(x->x mod 2, evalm(s+col(golay,13)));
```

$$[1,\ 1,\ 0,\ 1,\ 0,\ 0,\ 1,\ 1,\ 0,\ 1,\ 0,\ 0]$$

```
> map(x->x mod 2, evalm(s+col(golay,14)));
```

$$[1,\ 0,\ 0,\ 0,\ 1,\ 1,\ 1,\ 1,\ 0,\ 0,\ 0,\ 1]$$

```
> map(x->x mod 2, evalm(s+col(golay,15)));
```

$$[1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 0,\ 1,\ 0,\ 1,\ 1,\ 1]$$

```
> map(x->x mod 2, evalm(s+col(golay,16)));
```

$$[0,\ 1,\ 0,\ 0,\ 0,\ 1,\ 0,\ 0,\ 0,\ 1,\ 0,\ 1]$$

```
> map(x->x mod 2, evalm(s+col(golay,17)));
```

$$[1,\ 0,\ 0,\ 1,\ 1,\ 0,\ 0,\ 0,\ 1,\ 1,\ 0,\ 1]$$

```
> map(x->x mod 2, evalm(s+col(golay,18)));
```

$$[1,\ 1,\ 1,\ 1,\ 0,\ 1,\ 1,\ 0,\ 1,\ 0,\ 0,\ 1]$$

```
> map(x->x mod 2, evalm(s+col(golay,19)));
```

$$[1,\ 1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 1,\ 1,\ 0,\ 1,\ 1]$$

```
> map(x->x mod 2, evalm(s+col(golay,20)));
```

$$[0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1]$$

```
>  map(x->x mod 2, evalm(s+col(golay,21)));
```

$$[0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1]$$

```
>  map(x->x mod 2, evalm(s+col(golay,22)));
```

$$[0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1]$$

```
>  map(x->x mod 2, evalm(s+col(golay,23)));
```

$$[1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1]$$

```
>  map(x->x mod 2, evalm(s+col(golay,24)));
```

$$[0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1]$$

```
>  We now compute the vectors sB+b_j^T:
>  map(x->x mod 2, evalm(sB+row(golayb,1)));
```

$$[0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0]$$

```
>  This has weight 2, so we can stop.  This says that e1=1.  The
vector we
>  just computed has nonzero entries in positions k=6 and k=8.  Therefore
>  e18=1 and e20=1.  The corrected received vector is therefore
>  [1,0,1,0,1,0,1,0,1,0,1,0,0,1,0,0,1,0,1,1,1,1,0,0]
>  The first 12 entries give the message:
>  [1,0,1,0,1,0,1,0,1,0,1,0]


>  The third vector:
>  s:=map(x->x mod
>  2,evalm([1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1]&*transpose(g
>  olay)));
```

$$s := [0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1]$$

```
>  sB:=map(x->x mod 2,evalm(s&*golayb));
```

$$sB := [1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0]$$

```
>  map(x->x mod 2, evalm(s+col(golay,13)));
```

$$[1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1]$$

```
>  map(x->x mod 2, evalm(s+col(golay,14)));
```

$$[1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0]$$

```
>  map(x->x mod 2, evalm(s+col(golay,15)));
```

$$[1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0]$$

```
>  map(x->x mod 2, evalm(s+col(golay,16)));
```

$$[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]$$

```
>  We can stop here since this vector has weight 1.  We have e16=1
and
>  e4=1.  The corrected received vector is
>  [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]
>  The first 12 entries give the message:
```

```
>  [1,1,1,1,1,1,1,1,1,1,1,1]


>  Problem 2.  Multiply the received vector by the transpose of
the
>  parity check matrix:
>  map(x->x mod
>  2,evalm([0,1,1,0,0,0,1,1,0,0,0,1,0,1,0]&*transpose(hammingpc)));
```

$$[1, 1, 0, 0]$$

```
>  This is the 8th column of the parity check matrix, so the error
is in
>  the 8th entry of the received vector.  The corrected received
vector
>  is
>  [0,1,1,0,0,0,1,0,0,0,0,1,0,1,0]
>  The first 11 entries are the message:
>  [0,1,1,0,0,0,1,0,0,0,0]
```

Chapter 2 - MATLAB

Before you begin these problems, you should load in all of the needed ciphertexts. To do this, type

```
>> ciphertexts
```

1. After loading in all of the ciphertexts type in:

```
>> allshift(ycve)
```

```
ycvejqwvhqtdtwvwu
zdwfkrxwirueuxwxv
aexglsyxjsvfvyxyw
bfyhmtzyktwgwzyzx
cgzinuazluxhxazay
dhajovbamvyiybabz
eibkpwcbnwzjzcbca
fjclqxdcoxakadcdb
gkdmryedpyblbedec
hlenszfeqzcmcfefd
imfotagfradndgfge
jngpubhgsbeoehghf
kohqvcihtcfpfihig
lpirwdjiudgqgjijh
mqjsxekjvehrhkjki
nrktyflkwfisilklj
osluzgmlxgjtjmlmk
ptmvahnmyhkuknmnl
qunwbionzilvlonom
rvoxcjpoajmwmpopn
swpydkqpbknxnqpqo
txqzelrqcloyorqrp
uyrafmsrdmpzpsrsq
```

```
vzsbgntsenqaqtstr
watchoutforbrutus
xbudipvugpscsvuvt
```

The plaintext was 'watchoutforbrutus'.

2.

```
>> fr=frequency(lcll);
```

```
  a    2
 b   0
 c   1
 d   0
 e   1
 f   0
 g   0
 h   2
 i   1
 j   1
 k   0
 l   6
 m   2
 n   2
 o   0
 p   0
 q   0
 r   1
 s   0
 t   0
 u   0
 v   1
 w   1
 x   0
 y   2
 z   3
```

The most common common letter is l, which is 7 places after e. Try shifting back by 7:

```
>> shift(lcll,-7)
```

```
ans =
   eveexpectseggsforbreakfast
```

3. Let the decryption function be $x = ay + b$. The plaintext 'if' corresponds to the numbers 8 and 5. The ciphertext 'ed' corresponds to 4 and 3. Therefore $8 = 4a + b$ and $5 = 3a + b \pmod{26}$. Subtract to get $a = 3$. Then $b = 22$. Decrypt by

```
>> affinecrypt(edsg,3,22)

ans =
   ifyoucanreadthisthankateacher
```

4. Solve $y = 3x+b \pmod{26}$ for x to obtain $x = 9y-9b \pmod{26}$. Therefore the plaintext can be found by computing $9y$ and then trying all shifts.

```
>> allshift(affinecrypt(tcab,9,0))

psajpuoetlkooexehepeao
qtbkqvpfumlppfyfifqfbp
ruclrwqgvnmqqgzgjgrgcq
svdmsxrhwonrrhahkhshdr
twentysixpossibilities
uxfouztjyqpttjcjmjujft
vygpvaukzrquukdknkvkgu
wzhqwbvlasrvvlelolwlhv
xairxcwmbtswwmfmpmxmiw
ybjsydxncutxxngnqnynjx
zcktzeyodvuyyohorozoky
adluafzpewvzzpipspaplz
bemvbgaqfxwaaqjqtqbqma
cfnwchbrgyxbbrkrurcrnb
dgoxdicshzyccslsvsdsoc
ehpyejdtiazddtmtwtetpd
fiqzfkeujbaeeunuxufuqe
gjraglfvkcbffvovyvgvrf
hksbhmgwldcggwpwzwhwsg
iltcinhxmedhhxqxaxixth
jmudjoiynfeiiyrybyjyui
knvekpjzogfjjzszczkzvj
lowflqkaphgkkatadalawk
mpxgmrlbqihllbubebmbxl
nqyhnsmcrjimmcvcfcncym
orziotndskjnndwdgdodzn
```

Therefore the plaintext is 'twentysixpossibilities'.

5. For example, encrypt the string 'abcde' with various possibilities:

```
>> affinecrypt('abcde',267,11)

ans =
   lszgn
>> affinecrypt('abcde',7,11)

ans =
   lszgn
```

6.

a)

The message can be found in the file ciphertexts.m under the variable gaat. The following code performs the conversion to 0, 1, 2, 3, and performs the shifting.

```
ind0=find(gaat=='A');
ind1=find(gaat=='C');
ind2=find(gaat=='G');
ind3=find(gaat=='T');
vec=gaat;
vec(ind0)=0;
vec(ind1)=1;
vec(ind2)=2;
vec(ind3)=3;
vec=mod(vec+1,4);
ind0=find(vec==0);
ind1=find(vec==1);
ind2=find(vec==2);
ind3=find(vec==3);
output=vec;
output(ind0)='A';
output(ind1)='C';
output(ind2)='G';
output(ind3)='T';
output=char(output);
```

The answer is 'TCCAAGTGTTGGTGCCAACCGGGAGCGACCCTTTCAGAGACTCCGA'.

b)

The following code assumes that the affine cipher is of the form $y = ax+b \pmod 4$. The parameters a and b should be entered in. The restrictions are that a is relatively prime to 4 which means that a is either 1 or 3.

```
ind0=find(gaat=='A');
ind1=find(gaat=='C');
ind2=find(gaat=='G');
ind3=find(gaat=='T');
vec=gaat;
vec(ind0)=0;
vec(ind1)=1;
vec(ind2)=2;
vec(ind3)=3;
vec=mod(a*vec+b,4);
ind0=find(vec==0);
ind1=find(vec==1);
ind2=find(vec==2);
ind3=find(vec==3);
output=vec;
output(ind0)='A';
output(ind1)='C';
output(ind2)='G';
output(ind3)='T';
output=char(output);
```

7.

We first compute the coincidences for shifts of 1, 2, 3, 4, 5, and 6:

```
>> coinc(hdsf,1)

ans =
   11
>> coinc(hdsf,2)

ans =
   14
>> coinc(hdsf,3)

ans =
   15
>> coinc(hdsf,4)

ans =
   25
```

```
>> coinc(hdsf,5)

ans =
   14
>> coinc(hdsf,6)

ans =
   14
```

From this we conclude that the key length is probably 4.

```
>> vigvec(hdsf,4,1)

ans =
   0.1512
   0.0465
   0.0116
   0
   0.0349
   0.1279
   0.0698
   0.0465
   0
   0.0233
   0
   0.0349
   0
   0.0581
   0.0233
   0.0233
   0
   0
   0.0465
   0.0698
   0.0465
   0.0814
   0
   0.0116
   0.0465
   0.0465
>> corr(ans)

ans =
```

```
   0.0448
   0.0459
   0.0421
   0.0297
   0.0289
   0.0390
   0.0379
   0.0494
   0.0343
   0.0349
   0.0263
   0.0364
   0.0444
   0.0527
   0.0421
   0.0352
   0.0303
   0.0400
   0.0457
   0.0419
   0.0390
   0.0346
   0.0405
   0.0370
   0.0347
   0.0334
```

The maximum occurs in the 14th position and has value 0.0527. Thus the shift for the first letter is by 13. We now do the same for the other positions of hdsf.

```
>> vigvec(hdsf,4,2)

ans =
   0.0233
   0.0698
   0.1512
   0.0349
   0
   0.0233
   0.0814
   0.0581
```

```
   0.0349
   0
   0.0116
   0
   0.0233
   0
   0.1395
   0.0349
   0.0465
   0.1047
   0
   0.0116
   0.0116
   0.0349
   0.0814
   0
   0
   0.0233
>> corr(ans)

ans =
   0.0385
   0.0350
   0.0458
   0.0457
   0.0334
   0.0222
   0.0343
   0.0318
   0.0392
   0.0443
   0.0511
   0.0346
   0.0307
   0.0430
   0.0578
   0.0414
   0.0286
   0.0324
   0.0326
```

```
   0.0311
   0.0360
   0.0462
   0.0367
   0.0382
   0.0502
   0.0401
```

The max occurs at the 15th position, so the shift is by 14.

```
>> vigvec(hdsf,4,3)

ans =
   0.0349
   0
   0.0233
   0
   0.0698
   0.0116
   0.0116
   0
   0
   0.0465
   0.0349
   0.0698
   0.1395
   0
   0
   0.0349
   0.0349
   0.0698
   0.1512
   0
   0
   0.0581
   0.0698
   0.0698
   0.0698
   0
>> corr(ans)

ans =
```

```
   0.0374
   0.0310
   0.0290
   0.0341
   0.0577
   0.0466
   0.0312
   0.0375
   0.0435
   0.0381
   0.0419
   0.0430
   0.0363
   0.0336
   0.0410
   0.0307
   0.0354
   0.0395
   0.0487
   0.0427
   0.0365
   0.0293
   0.0342
   0.0353
   0.0468
   0.0399
```

The max occurs in the 5th position, so the shift for the third letter is by 4.

```
>> vigvec(hdsf,4,4)

ans =
   0.0930
   0
   0.0116
   0.0465
   0.0349
   0.0698
   0.1279
   0.0116
   0
```

```
   0.0349
   0.0465
   0.0930
   0.0233
   0.0116
   0.0465
   0
   0.0814
   0
   0.1395
   0
   0.0233
   0.0581
   0
   0
   0.0233
   0.0233
>> corr(ans)

ans =
   0.0388
   0.0367
   0.0449
   0.0393
   0.0366
   0.0354
   0.0387
   0.0451
   0.0327
   0.0280
   0.0381
   0.0372
   0.0435
   0.0429
   0.0483
   0.0301
   0.0323
   0.0350
   0.0572
   0.0333
```

```
   0.0307
   0.0317
   0.0379
   0.0377
   0.0417
   0.0473
```

The max occurs in the 19th position. So the shift is by 18. Using these four shifts of 13, 14, 4, and 18 to decrypt we get:

```
>> vigenere(hdsf,-[13 14 4 18])

ans =

uponthisbasisiamgoingtoshowyouhowabunchofbright
youngfolksdidfindachampionamanwithboysandgirlso
fhisownamanofsodominatingandhappyindividualityt
hatyouthisdrawntohimasisaflytoasugarbowlitisast
oryaboutasmalltownitisnotagossipyyarnnorisitadr
ymonotonousaccountfullofsuchcustomaryfillinsasr
omanticmoonlightcastingmurkyshadowsdownalongwin
dingcountryroad
```

Observe that this plaintext does not have any letter 'e', and that this causes the maximum values of the dot products to be smaller than their usual values.

8.

We first calculate the coincidences to determine the most likely key length:

```
>> coinc(ocwy,1)

ans =
   13
>> coinc(ocwy,2)

ans =
   13
>> coinc(ocwy,3)

ans =
   11
>> coinc(ocwy,4)

ans =
```

```
   6
>> coinc(ocwy,5)

ans =
   15
>> coinc(ocwy,6)

ans =
   23
>> coinc(ocwy,7)

ans =
   8
```

Observe that the key length is most likely 6.

```
>> vigvec(ocwy,6,1)

ans =
   0.0577
   0.0769
   0.0385
   0
   0
   0.0192
   0
   0.0962
   0.0385
   0.0192
   0.0577
   0.0769
   0
   0.0577
   0.0769
   0.0769
   0
   0.0192
   0.0192
   0.0385
   0.1346
   0.0192
   0.0385
   0
```

```
   0.0192
   0.0192
>> corr(ans)

ans =
   0.0417
   0.0452
   0.0372
   0.0452
   0.0295
   0.0302
   0.0420
   0.0561
   0.0380
   0.0399
   0.0352
   0.0367
   0.0317
   0.0392
   0.0411
   0.0389
   0.0410
   0.0322
   0.0351
   0.0384
   0.0476
   0.0324
   0.0449
   0.0394
   0.0330
   0.0295
```

The maximum occurs at the 8th position.

```
>> vigvec(ocwy,6,2)

ans =
   0.0577
   0.1346
   0.0962
   0
   0
```

```
   0.0769
   0.0385
   0.0385
   0
   0
   0.0192
   0
   0.0385
   0
   0.0769
   0.0192
   0.0192
   0.0577
   0.0962
   0.0192
   0
   0.1154
   0.0769
   0
   0.0192
   0
>> corr(ans)

ans =
   0.0361
   0.0435
   0.0399
   0.0382
   0.0359
   0.0313
   0.0272
   0.0385
   0.0441
   0.0411
   0.0473
   0.0308
   0.0301
   0.0466
   0.0642
   0.0342
```

```
   0.0263
   0.0384
   0.0417
   0.0361
   0.0362
   0.0366
   0.0326
   0.0427
   0.0442
   0.0370
```

The maximum occurs in the 15th position. We now perform the same procedure for the remaining positions. The shifts that were used were 7, 14, 11, 12, 4, 18 and correspond to the name 'holmes'. To decrypt we do:

```
>> vigenere(ocwy,-[7 14 11 12 4 18])

ans =

holmeshadbeenseatedforsomehoursinsilencewithhi
slongthinbackcurvedoverachemicalvesselinwhichh
ewasbrewingaparticularlymalodorousproducthishe
adwassunkuponhisbreastandhelookedfrommypointof
viewlikeastrangelankbirdwithdullgreyplumageand
ablacktopknotsowatsonsaidhesuddenlyyoudonotpro
posetoinvestinsouthafricansecurities
```

9.

We first calculate the coincidences to determine the most likely key length:

```
>> coinc(xkju,1)

ans =
   15
>> coinc(xkju,2)

ans =
   17
>> coinc(xkju,3)

ans =
   13
>> coinc(xkju,4)
```

```
ans =
   10
>> coinc(xkju,5)

ans =
   23
>> coinc(xkju,6)

ans =
   5
```

The key length is most likely 5.

```
>> vigvec(xkju,5,1)

ans =
   0
   0.0606
   0.0303
   0.0303
   0.0606
   0.1515
   0.0303
   0
   0.1061
   0.0303
   0
   0
   0.0303
   0.0152
   0.0758
   0.0606
   0.0152
   0.0152
   0.0606
   0.0606
   0.1212
   0
   0
   0.0303
   0
   0.0152
>> corr(ans)
```

```
ans =
   0.0445
   0.0684
   0.0388
   0.0296
   0.0382
   0.0411
   0.0349
   0.0312
   0.0358
   0.0300
   0.0357
   0.0371
   0.0492
   0.0401
   0.0422
   0.0406
   0.0474
   0.0394
   0.0314
   0.0313
   0.0409
   0.0331
   0.0299
   0.0441
   0.0357
   0.0303
```

The maximum occurs in the 2nd position. We perform the same procedure for the other positions to get the shifts of 1, 3, 5, 7, and 9. To decrypt we do:

```
>> vigenere(xkju,-[1 3 5 7 9])

ans =

wheninthecourseofhumaneventsitbecomesnecessary
foronepeopletodissolvethepoliticalbandswhichha
veconnectedthemwithanotherandtoassumeamongthep
owersoftheearththeseparateandequalstationtowhi
chthelawsofnatureandofnaturesgodentitlethemade
centrespecttotheopinionsofmankindrequiresthatt
```

```
heyshoulddeclarethecauseswhichimpelthemtothese
paration
```

10.

```
>> M=[1 2 3 4; 4 3 2 1; 11 2 4 6; 2 9 6 4]
   M =
     1    2    3    4
     4    3    2    1
    11    2    4    6
     2    9    6    4
>> Minv=inv(M)
   Minv =
    -0.1455    0.0364    0.0909    0.0000
    -1.5636   -2.1091    0.7273    1.0000
     3.3636    4.9091   -1.7273   -2.0000
    -1.4545   -2.6364    0.9091    1.0000
```

We are working mod 26 and so we can't deal with fractional values. What we need to do is multiply out the fractional values by multiplying by the determinant of M. The determinant is calculated by:

```
>> det(M)

ans =
   -55
```

We will multiply Minv by -55 and then multiply by $(-55)^{-1} \pmod{26} = 17$.

```
>> M1=Minv*(-55)
   M1 =
      8.0000      -2.0000      -5.0000      -0.0000
     86.0000     116.0000     -40.0000     -55.0000
   -185.0000    -270.0000      95.0000     110.0000
     80.0000     145.0000     -50.0000     -55.0000
>> M2=round(mod(M1*17,26))
   M2 =
    6   18   19   26
    6   22   22    1
    1   12    3   24
    8   21    8    1
```

A quick check shows that this is indeed the inverse matrix mod 26.

```
>> mod(M2*M,26)
```

```
ans =
    1   0   0   0
    0   1   0   0
    0   0   1   0
    0   0   0   1
```

The ciphertext is stored in the variable zirk in the file ciphertexts.m. We can convert it to numerical values by:

```
>> text2int(zirk)

ans =
   Columns 1 through 12
   25 8 17 10 25 22 14 15 9 9 14 15
   Columns 13 through 24
   19 5 0 15 20 7 5 7 0 3 17 16
```

We now take these numbers in blocks of 4 and multiply each block on the right by the inverse matrix we just calculated:

```
>> mod([25 8 17 10]*M2,26)

ans =
   9 0 2 10
>> mod([25 22 14 15]*M2,26)

ans =
   0 13 3 9
>> mod([9 9 14 15]*M2,26)

ans =
   8 11 11 22
>> mod([19 5 0 15 ]*M2,26)

ans =
   4 13 19 20
>> mod([20 7 5 7 ]*M2,26)

ans =
   15 19 7 4
>> mod([0 3 17 16]*M2,26)

ans =
   7 8 11 11
```

Now we convert to text by:

```
>> int2text([9 0 2 10 0 13 3 9 8 11 11 22 4 13 19 20 15 19 7 4 7
8 11 11])
```

```
ans =
   jackandjillwentupthehill
```

11.

```
>> lfsrlength(L101,10)
   Order Determinant
   1 1
   2 1
   3 1
   4 0
   5 1
   6 1
   7 0
   8 0
   9 0
   10 0
```

Thus, we guess that the recurrence length is 6. To determine the recurrence coefficients, we do:

```
>> lfsrsolve(L101,6)

ans =
   1 1 0 1 1 0
```

This gives the recurrence as

$$x_{n+6}\ x_n + x_{n+1} + x_{n+3} + x_{n+4} \pmod 2.$$

12.

```
>> lfsrlength(L100,12)
   Order Determinant
   1 1
   2 0
   3 1
   4 0
   5 0
   6 1
   7 0
   8 1
   9 0
   10 0
   11 0
```

```
   12 0
```

The recurrence length is most likely 8. To determine the recurrence coefficients, we do:

```
>> lfsrsolve(L100,8)

ans =
   1 1 0 0 1 0 0 0
```

This gives the recurrence as

$$x_{n+8} = x_n + x_{n+1} + x_{n+4}.$$

13.

We start by XORing the ciphertext with the known part of the plaintext to obtain the beginning of the LFSR output:

```
>> x=mod(L011(1:15)+[1 0 0 1 0 0 1 0 0 1 0 0 1 0 0],2)
   x =
   Columns 1 through 12
   1 1 1 1 0 0 0 0 1 1 1 1
   Columns 13 through 15
   0 0 0
```

This is the beginning of the LFSR output. Now we find the length of the recurrence:

```
>> lfsrlength(x,8)
   Order Determinant
   1 1
   2 0
   3 0
   4 1
   5 1
   6 0
   7 0
   8 0
```

The length is most likely 5. We now solve for the coefficients of the recurrence:

```
>> lfsrsolve(x,5)

ans =
   1 1 0 0 1
```

Now we can generate the full output of the LFSR using the coefficients we just found plus the first five terms of the LFSR output:

```
>> lfsr([1 1 0 0 1],[1 1 1 1 0],50)

ans =
   Columns 1 through 12
   1 1 1 1 0 0 0 0 1 1 1 1
   Columns 13 through 24
   0 0 0 0 1 1 1 1 0 0 0 0
   Columns 25 through 36
   1 1 1 1 0 0 0 0 1 1 1 1
   Columns 37 through 48
   0 0 0 0 1 1 1 1 0 0 0 0
   Columns 49 through 50
   1 1
```

When we XOR the LFSR output with the ciphertext, we get back the plaintext:

```
>> mod(ans+L011,2)

ans =
   Columns 1 through 12
   1 0 0 1 0 0 1 0 0 1 0 0
   Columns 13 through 24
   1 0 0 1 0 0 1 0 0 1 0 0
   Columns 25 through 36
   1 0 1 1 0 1 1 0 1 1 0 1
   Columns 37 through 48
   1 0 1 1 0 1 1 0 1 1 0 1
   Columns 49 through 50
   1 0
```

This is the plaintext.

Chapter 3 - MATLAB

1.

```
>> gcd(8765,23485)

ans =
   5
```

2.(a)

```
>> [g,x,y]=gcd(65537,3511)
   g =
   1
   x =
   -1405
   y =
   26226
```

(b) Multiply both x and y by 17 to get

```
>> 17*x, 17*y

ans =
   -23885

ans =
   445842
```

3.

```
>> powermod(3,1234567,100000)

ans =
   40587
```

4.

```
>> powermod(314,-1,11111)
```

```
ans =
   7254
>> mod(7254*271,11111)

ans =
   10298
```

5. Turn $216x = 66 \pmod{606}$ into $36x = 11 \pmod{101}$

```
>> powermod(36,-1,101)

ans =
   87
>> mod(87*11,101)

ans =
   48
```

Now add 101, 202, 303, 404, and 505 to 48 to get the answers:

```
>> 48 + [0, 101, 202, 303, 404, 505]

ans =
   48 149 250 351 452 553
```

6.

```
>> crt([17 18 19],[101 201 301])

ans =
   61122
```

7.

```
>> powermod(2,390,391)

ans =
   285
```

Take $j = \phi(n)$.

```
>> eulerphi(391)

ans =
   352
```

8.

```
>> crt([123,-123],[84047,65497])

ans =
   5495329171.00
```

9. Factor 65536 as 2^{16}.

```
>> powermod(3,65536/2,65537)

ans =
   65536.00
```

Since this is not 1 mod 65537 we have a primitive root.

10. (a)

```
>> M=[1 2 4; 1 5 25; 1 14 196];
>> format rat
>> Minv=inv(M)
   Minv =
     35/18    -28/27     5/54
    -19/36     16/27    -7/108
      1/36     -1/27     1/108
>> M1=(Minv*108)*powermod(108,-1,101)
   M1 =
     6090    -3248    290
    -1653     1856   -203
       87     -116     29
>> mod(M1,101)

ans =
    30   85    88
    64   38   100
    87   86    29
```

(b) These are precisely the factors of 324 which are 2 and 3.

11.

```
>> powermod(26055,(34807+1)/4,34807)

ans =
   33573
>> mod(-33573,34807)

ans =
   1234
```

12.

```
>> factor(2325781)

ans =
   523 4447
```

```
>> powermod(1522756,(523+1)/4,523)

ans =
   335
>> powermod(1522756,(4447+1)/4,4447)

ans =
   1234
```

Now we put together the four combinations

```
>> crt([335 1234],[523 4447])

ans =
   437040
>> crt([-335 1234],[523 4447])

ans =
   1234
>> crt([335 -1234],[523 4447])

ans =
   2324547
>> crt([-335 -1234],[523 4447])

ans =
   1888741
```

13.

```
>> powermod(48382,(83987+1)/4,83987)

ans =
   60555
>> powermod(60555,2,83987)

ans =
   35605
```

We found a square of -48382.

Chapter 6 - MATLAB

1.

First, we convert the two text messages to numerical values by:

```
>> text2int1('one')

ans =
   151405
>> text2int1('two')

ans =
   202315
```

The modulus is too large to be able to perform the computations in MATLAB without using the Maple Kernel. To perform the exponentiations using the Maple Kernel, do:

```
>> maple('151405&^6551 mod 712446816787')

ans =
   273095689186
>> maple('202315&^6551 mod 712446816787')

ans =
   709427776011
```

Therefore the message was "one".

2.

Use the universal exponent factoring method. Compute e*d-1:

```
>> maple('3449*543546506135745129-1')

ans =
   1874691899662184949920
```

Express it as a power of 2 times an odd number. Trial and error produces

```
>> maple('1874691899662184949920/32')
```

```
ans =
   58584121864443279685
```

and hence e*d-1 is 32 times an odd number. Call the odd number b_0 Pick a random number, say 2, and raise it to b_0th power.

```
>> maple('2&^58584121864443279685 mod 718548065973745507')

ans =
   511846277294206136
```

Successively square until we get 1 mod n:

```
>> maple('%&^2 mod 718548065973745507')

ans =
   551183936117094424
>> maple('%&^2 mod 718548065973745507')

ans =
   576566739470926048
>> maple('%&^2 mod 718548065973745507')

ans =
   1
```

The last number before the 1 was not -1 mod n, so we use the gcd to factor n:

```
>> maple('gcd(%% -1, 718548065973745507)')

ans =
   740876531
```

This is one of the factors of n. The other is

```
>> maple('718548065973745507/%')

ans =
   969862097
```

3.

We first choose some RSA parameters:

```
>> maple('p:=nextprime(rand(10^29..10^30)())')

ans =
   p := 769081321110693270343632599093
>> maple('q:=nextprime(rand(10^29..10^30)());')

ans =
   q := 34356355845871897676746753303637
```

```
>> maple('n:=p*q')

ans =
   n := 264228315424922488460852715316219596666650690368066519801241
>> maple('e:=nextprime(rand(10^9..10^10)())')

ans =
   e := 6062222113
```

We convert the words to numerical values as:

```
>> text2int1('cat')

ans =
   30120
>> text2int1('bat')

ans =
   20120
>> text2int1('hat')

ans =
   80120
```

Observe that we cannot use the MATLAB function text2int1 to convert encyclopedia or antidisestablishmentarianism into numerical values since their numerical representations are too long for MATLAB to represent with double precision. Therefore, we only present results for the first cat, bat and hat.

Now we calculate the ciphertexts:

```
>> maple('30120&^e mod n')

ans =
   221108395229623543553316522699162115906870683738440529748061
>> maple('20120&^e mod n')

ans =
   899421085504261939456198062304525371248333332392508946072l3
>> maple('80120&^e mod n')

ans =
   98160144364428376061382242659587339389722081318357702659331
```

There is no obvious relationship between the messages.

4.

Let's choose our bound to be B=50 and let $a = 2$.

```
>> maple('2&^factorial(50) mod 618240007109027021')
```

```
ans =
   394571778315865637
>> maple('gcd(%-1,618240007109027021)')

ans =
   250387201
```

The other factor is

```
>> maple('618240007109027021/%')

ans =
   2469135821
```

The reason this worked is

```
>> maple('ifactor(%%-1)')

ans =
   ``(2)^8*``(3)^5*``(5)^2*``(7)*``(23)
```

The first prime factor p is such that p-1 has only small prime factors.

5.

We'll use B=100 and a=2&^factorial(100) mod n1:

```
>> maple('n1:=8834884587090814646372459890377418962766907');
>> maple('a:2&^factorial(100) mod n1')

ans =
   96162382165780005507702322927071548424814 3
>> maple('gcd(%-1,n1);')

ans =
   364438989216827965440001
```

The other factor is

```
>> maple('n1/%')

ans =
   24242424242468686907
```

6.

We use the Basic Principle:

```
>> maple('gcd(85975324443166-462436106261, 537069139875071);')

ans =
   9876469
```

The other factor is

```
>> maple('537069139875071/%;')

ans =
   54378659
```

7.

Choose a random x, say x=123. Do the following

```
>> n=985739879*1388749507;
>> y=crt([123,-123],[84047,65497])
   y =
   312089563232070143
```

The square of this is congruent to the square of 123 mod each prime factor, therefore mod their product:

```
>> powermod(123,2,n)

ans =
   15129

>> powermod(312089563232070143,2,n)

ans =
   15129
```

However, by construction, 312089563232070143 is not congruent to 123 or -123.

8.

(a)

```
>> gcd(33335-670705093,670726081)

ans =
   54323
```

The other factor is:

```
>> 670726081/ans

ans =
   12347
```

(b)

Observe that 3 = -670726078 mod 670726081. Let's try the gcd anyway:

```
>> gcd(3-670726078,670726081)

ans =
   1
```

9.

This is what happens in the exponent factorization method. We see that $1488665^2 \equiv 1 \pmod{3837523}$, so we use the gcd to factor:

```
>> n=3837523
   n =
   3837523
>> gcd(1488665-1,n)

ans =
   3511
>> n/ans

ans =
   1093
```

10.

(a)

Find the square root of n. The next prime will be a factor and the previous prime will be the other factor.

(b)

```
>> n=10993522499
   n =
   10993522499
>> q=nextprime(round(sqrt(n)))
   q =
   104851
>> p=n/q
   p =
   104849
>> d=powermod(113,-1,(p-1)*(q-1))
   d =
   5545299377
```

The numbers are too big to use MATLAB without the Maple Kernel:

```
>> maple('10787770728&^5545299377 mod 10993522499 ')

ans =
   5011925
```

To convert it to a string, we use:

```
>> int2text1(5011925)

ans =
```

easy

(c)

First we load the variables into the Maple Kernel in MATLAB:

```
>> maple('naive:= 152415787501905985701881832150835089037
858868621211004433')

ans =
   naive := 152415787501905985701881832150835089037
858868621211004433

>> maple('cnaive:= 141077461765569500241199505617854673388
398574333341423525')

ans =
   cnaive := 141077461765569500241199505617854673388
398574333341423525
```

Since p and q are consecutive primes we can find them by:

```
>> maple('q:=nextprime(round(sqrt(naive)))')

ans =
   q := 123456789000000314159265001 43
>> maple('p:=naive/q')

ans =
   p := 123456789000000314159265000 31
```

Now we calculate the decryption exponent as:

```
>> maple('d:=1/9007 mod ((p-1)*(q-1))')

ans =
   d := 660462771866258534689069380246851318997
849360492799256 83
```

and the message as:

```
>> maple('cnaive &^d mod naive')

ans =
   200809190014211302051800230119001415200
0190503211805
```

Using the int2text1 function on portions of the this number allows us to translate it into the text as 'this number was not secure'.

11.

(a)

```
>> maple('p:=123456791;');
>> maple('q:=987654323;');
>> maple('e:=127;');
>> maple('m:=14152019010605;');
>> maple('m&^e mod p')

ans =
   104120308
>> maple('m&^e mod q')

ans =
   812538893
```

Now we use the Chinese Remainder Theorem to combine:

```
>> maple('c:=chrem([%%,%],[p,q])')

ans =
   c := 18001903071439991
```

(b) Let's change 1 digit by adding 100000

```
>> maple('m&^e mod p + 100000')

ans =
   104220308
>> maple('m&^e mod q')

ans =
   812538893
>> maple('f:=chrem([%%,%],[p,q])')

ans =
   f := 17982149984979991
>> maple('gcd(f-c,p*q)')

ans =
   987654323
```

This found the factor q because $f = c \pmod{q}$ but f is not congruent to c mod p.

12.

We first enter the numbers into the Maple Kernel by:

```
>> maple('p:=76543692179;');
>> maple('q:=343434343453;');
>> maple('e:=457;');
```

We'll try all possibilities for the last digit until we get a meaningful message. It will be easiest to append a 0 to the received text, then add 0, 1, 2, ... and decrypt.

```
>> maple('c := 23043293280169369471950');
>> maple('d:=1/e mod ((p-1)*(q-1))')

ans =
   d := 1553104555909567360609
```

Now try to decrypt the various possibilities. Note that the incorrect ones will probably contain 2-digit blocks larger than 26, hence they cannot be changed back to letters:

```
>> maple('(c+0)&^d mod p*q')

ans =
   4174116046631355447474
>> maple('(c+1)&^d mod p*q')

ans =
   8579710266419129803917
>> maple('(c+2)&^d mod p*q')

ans =
   1913091205
```

This last one can be translated to the message 'smile' using int2text1.

13.

```
>> maple('n:=38200901201;');
```

Write n-1 as a power of 2 times an odd number. By dividing successively by 2 we get that $n-1$ is 16 times an odd number.

```
>> maple('2&^((n-1)/16) mod n')

ans =
   1
```

The Miller-Rabin test says that n is probably prime. Now try a=3:

```
>> maple('3&^((n-1)/16) mod n')

ans =
   6099632610
>> maple('%&^2 mod n')

ans =
   9753514238
```

```
>> maple('%&^2 mod n')

ans =
   5489404833
>> maple('%&^2 mod n')

ans =
   22630222508
>> maple('%&^2 mod n')

ans =
   767945134
```

This is $3^{(n-1)} \pmod{n}$. It is not 1, so Miller-Rabin (or Fermat) test says that n is composite.

14.

(a) and (b) Let m be the message. We know $m^3 \pmod{n_1}$, $m^3 \pmod{n_2}$, and $m^3 \pmod{n_3}$. By the Chinese Remainder Theorem, we know $m^3 \pmod{n_1 n_2 n_3}$. Since $m < n_1, n_2, n_3$, we have $m^3 < n_1 n_2 n_3$. Therefore we know m^3. Taking the cube root, we obtain m.

(c)

We follow the steps outlined in parts (a,b).

```
>> maple('chrem([359335245251,10436363975495,5135984059593],
   [2469247531693,11111502225583,44444222221411])')

ans =
   52189581153668510460961337 5
>> maple('%^(1/3);')

ans =
   52189581153668510460961337 5^(1/3)
>> maple('simplify(%)')

ans =
   805121215
```

This translates to the text 'hello'.

Chapter 7 - MATLAB

1.

In order to verify, we just need to check that $3^{1234} = 8576 \pmod{53047}$.

```
>> powermod(3,1234,53047)

ans =
   8576
```

2.

One way to do this is the following:

```
>> for j=1:30,
   [j, powermod(3,j,31)]
   end
```

The output has been suppressed, but you will find that the logarithm happens at $j = 13$.

3. (a)

The verification is done by checking that $2^{2000} = 3925 \pmod{3989}$ and $2^{3000} = 1046 \pmod{3989}$.

```
>> powermod(2,2000,3989)

ans =
   3925
>> powermod(2,3000,3989)

ans =
   1046
```

(b)

We use the property that $L_\alpha(\beta_1\beta_2) = L_\alpha(\beta_1) + L_\alpha(\beta_2)$:

```
>> mod(2000+3000,3988)

ans =
```

```
   1012
```

4.

```
>> factor(1200)

ans =
   Columns 1 through 5
   2 2 2 2 3
   Columns 6 through 7
   5 5
>> powermod(2,(1200)/2,1201)

ans =
   1
```

Therefore $x = 0 \pmod 2$.

```
>> powermod(2,(1200)/4,1201)

ans =
   1
```

Therefore $x = 0 \pmod 4$.

```
>> powermod(2,(1200)/8,1201)

ans =
   1200
```

This is -1 mod 1201, and hence $x = 4 \pmod 8$.

```
>> b3=mod(2*powermod(11,-4,1201),1201)
   b3 =
   729
>> powermod(b3,1200/16,1201)

ans =
   1200
```

Therefore $x = 4 + 8 \pmod{16}$. Now we need to work mod 3.

```
>> powermod(2,1200/3,1201)

ans =
   570
>> w=powermod(11,1200/3,1201)
   w =
   570
```

Hence $x = 1 \pmod 3$. Now we work mod 5 and 25.

```
>> powermod(2,1200/5,1201)

ans =
   105
>> w=powermod(11,1200/5,1201)
   w =
   1062
```

and w^2 is

```
>> powermod(w,2,1201)

ans =
   105
```

So $x = 2 \pmod 5$. Now mod 25.

```
>> powermod(2,1200/25,1201)

ans =
   443
>> w=powermod(11,1200/25,1201)
   w =
   96
```

We need to find what power of w gives 443. A little trial and error gives:

```
>> powermod(w,7,1201)

ans =
   443
```

Thus $x = 7 \pmod{25}$. Now we do Chinese Remainder Theorem to get:

```
>> crt([12 1 7],[16 3 25])

ans =
   1132
```

This is the answer.

Chapter 8 - MATLAB

1.

(a) The probability that two have the same birthday is $\prod_{i=1}^{29}(1-i/365)$. Subtracting from 1 gives:

```
>> 1 - prod(1 - (1:29)/365)

ans =
   0.7063
```

The formula on page 230 gives 0.708547.

(b) We use the approximation on page 230 to conclude that for a 99% probability of a pair having the same birthday, that $r^2/2N \approx 4.6$. Thus we need roughly $r = \sqrt{2 \cdot 4.6 \cdot 365} = 57.9$ people. In fact, we are close as 57 people gives

```
>> 1 - prod(1 - (1:56)/365)

ans =
   0.9901
```

Try the one above and below and they are further away.

(c) 366 people would be needed.

2. Here there are 10000 objects and 200 students so the probability is given by

```
>> 1- prod(1 - (1:199)/10000)

ans =
   0.8651
```

Chapter 9 - MATLAB

1. (a) Observe that the r values are the same, and therefore the k values are the same.

(b) This problem is similar to the example in Section 8.2. We shall solve $(s_1 - s_2)k = m_1 - m_2 \pmod{p-1}$ for k. Observe that $s_1 - s_2 = -25230$ and $m_1 - m_2 = -21696$. Since $gcd(-25230, p-1) = 6$, there are six possible solutions. These can be found by the method of Section 3.3 by dividing by both sides by the gcd to get:

$$4205k \equiv 3616 \pmod{10923}.$$

This gives $k = 1814$. The other possible values can be found by adding multiples of 10923 to 1814. We now calculate α^k for these possible k values until we find one that gives $\alpha^k = r \pmod{p}$. It turns out that $k = 56429$ gives the desired answer, which can be verified by

```
>> powermod(2,56429,65539)

ans =
   18357
```

Now we solve for a. To do this, use $(m - sk) = ar \pmod{p-1}$. Thus we have

$$54915 \equiv 18357a \pmod{65538}$$

which, since $gcd(18357, 65538) = 3$, has 3 solutions. These can be found by solving

$$18305 \equiv 6119 \pmod{21846}$$

giving $a = 9871$, and the other possible solutions are found by adding multiples of 21846. Only $a = 9871$ satisfies $\beta = \alpha^a \pmod{p}$ which can be verified by

```
>> powermod(2,9871,65539)

ans =
```

```
   33384
```

2.

First we calculate Bob's decryption exponent d_B by

```
>> maple('db:=1/87697 mod (sigpb-1)*(sigqb-1)')

ans =
   db := 25995904256807890225566393955459263520507147 3
>> maple('sigpairm1 &^ db mod signb')

ans =
   19012507151504022505
```

Converting this to text gives 'saygoodbye'. We now decrypt s_1 by

```
>> maple('sigpairs1 &^ db mod signb')

ans =
   150270996499036309478023705411245214416829627
```

and raising this to the e_A power mod n_A should give the message:

```
>> maple('% &^ 1571 mod signa')

ans =
   19012507151504022505
```

Thus, Alice sent the message.

3.

Observe that the new n_B is

```
>> maple('7865712896579*sigpb')

ans =
   776845002924591882150632670282388927
```

and that this is smaller than the value for the signature s calculated in problem 2. Thus, we will immediately lose information about s when we perform encryptions to turn m and s into m_2 and s_2. The information we lose is unrecoverable.

Chapter 12 - MATLAB

1.

First we enter the values:

```
>> n=2110763
   n =
   2110763
>> A=1008369
   A =
   1008369
>> r=593647
   r =
   593647
>> s=631870
   s =
   631870
```

The answer is obtained by adding Alice's share A with r and s:

```
>> mod(A+r+s,n)

ans =
   123123
```

2.

```
>> x=[1 2 3 4 5 6 7];
>> s=[214 7543 6912 8223 3904 3857 510];
>> p=8737;
```

Now we find the interpolating polynomial. We can choose any 4 of the points. For example, if we use the first 4 points, the command is:

```
>> interppoly(x(1:4),s(1:4),p)

ans =
```

```
   1234 5387 3592 7475
```

The secret is 1234. To verify, we shall use the 4,5,6,7 shares:

```
>> interppoly(x(4:7),s(4:7),p)

ans =
   1234 5387 3592 7475
```

3.

We form the interpolating polynomial for different combinations of two users, and then compare the results. The answer that occurs most often is the correct answer.

```
>> p=984583;
>> x=[38 3876 23112 432];
>> s=[358910 9612 28774 178067];
>> interppoly(x([1 2]),s([1 2]),p)

ans =
   69918 318526
>> interppoly(x([1 3]),s([1 3]),p)

ans =
   21502 941642
>> interppoly(x([1 4]),s([1 4]),p)

ans =
   21502 941642
>> interppoly(x([3 4]),s([3 4]),p)

ans =
   21502 941642
```

From these, it is clear that share 2 is the incorrect share.

Chapter 16 - MATLAB

1.

(a)

```
>> p=19;
>> addell([1 5],[9 3],2,3,19)

ans =
   15 8
```

(b)

```
>> addell([9 3],[9 -3],2,3,19)

ans =
   Inf Inf
```

(c)

```
>> addell([1 5],[9 -3],2,3,19)

ans =
   10 4
```

(d)

```
>> multsell([1,5],25,2,3,19)

ans =
   1:  1 5
   2:  3 13
   3:  12 8
   4:  10 15
   5:  9 3
   6:  15 8
   7:  14 18
   8:  5 10
   9:  11 11
```

```
   10:  18 0
   11:  11 8
   12:  5 9
   13:  14 1
   14:  15 11
   15:  9 16
   16:  10 4
   17:  12 11
   18:  3 6
   19:  1 14
   20:  Inf Inf
   21:  1 5
   22:  3 13
   23:  12 8
   24:  10 15
   25:  9 3
```

We see that 5*(1,5)=(9,3).

(e)

From this list, observe that we repeat after 20 iterations.

(f)

Observe that this gives us $(20k - 20) < 2\sqrt{19}$ so $k < (2\sqrt{19} + 20)/20$

```
>> (2*sqrt(19)+20)/20

ans =
   1.4359
```

Hence, $k = 1$, and thus E has exactly 20 points.

2.

First, we try $x = 123450$:

```
>> p=593899;
>> x=123450;
>> y2=mod(powermod(x,3,p)+7*x+11,p)
   y2 =
   474965
>> z=powermod(y2,(p+1)/4,p)
   z =
   371853
>> powermod(z,2,p)

ans =
```

```
   118934
```

This value of x does not lead to a value that has a square root, and hence does not lie on the curve. We therefore try the next value, $x = 123451$.

```
>> x=123451;
>> y2=mod(powermod(x,3,p)+7*x+11,p)
   y2 =
   426106
>> z=powermod(y2,(p+1)/4,p)
   z =
   423090
>> powermod(z,2,p)

ans =
   426106
```

Since $z^2 = y^2 \pmod{p}$, we have a square root, and hence the point (x, z) lies on the curve, i.e. $(123451, 423090)$ belongs to the curve.

3.

(a)

```
>> n=3900353
   n =
   3900353
```

We start by picking a random point and a random curve and a bound B. Lets take (1,1) on the curve $y = x^3 + x + b \pmod{n}$, this forces b to be -1. If we take the bound $B = 12$ we get:

```
>> multell([1 1],factorial(12),1,-1,n)

ans =
   2520576 1519543
```

Since this did not produce a factor, we try another curve until we do find a factor. For example:

```
>> multell([2 4],factorial(12),2,-8,n)

ans =
   477360 2030165
```

doesn't produce a factor, but the following curve does.

```
>> multell([1 2],factorial(12),17,-14,n)
   Elliptic Curve addition produced a factor of n, factor= 1109
Multell found a factor of n and exited

ans =
```

```
[]
```

Hence one factor is 1109. The other is 3517.

(b)

We follow the procedure in section 6.4 of the book. We take $a = 2$ and choose $B = 12$, and get:

```
>> b=powermod(2,factorial(12),n)
   b =
   972999
>> gcd(b-1,n)

ans =
   1
```

Since the gcd is 1, we have not found a factor. Similarly, if we take $a = 3$ and $B = 32$ we get:

```
>> b=powermod(3,factorial(32),n)
   b =
   1879231
>> gcd(b-1,n)

ans =
   1
```

If you try several choices for a and B, it will be unlikely that you find a factor as both $p - 1$ and $q - 1$ have a large prime factor.

4.

(a)

```
>> p=557;
>> multell([2 3],189,-10,21,p)

ans =
   Inf Inf
>> multell([2 3],63,-10,21,p)

ans =
   38 535
>> multell([2 3],27,-10,21,p)

ans =
   136 360
```

(b)

Observe that the prime factors of p are 3 and 7 and that $189/3 = 63$, and $189/7 = 27$ (where /7 means multiply by inverse of 7 mod p). Hence, by Exercise 9, P has order 189.

(c)

Observe that we thus have $189k < 2\sqrt{557} + 558$, hence $k < (2\sqrt{557} + 558)/189$ this gives that $1 \leq k \leq 3$. Observe that k=1 or 2 does not satisfy Hasse's Theorem.

5.

```
>> addell([5 9],[1 -1],11,11,593899)

ans =
   148475 222715
```

Chapter 18 - MATLAB

1. The matrices are stored in the file Golay.m.

```
>> Golay
>> r=[0 1 0 0 0 0 1 1 0 1 0 1 1 0 1 1 0 1 0 0 0 0 1 1];
```

We refer the reader to pg 326 for the decoding algorithm. First, calculate the syndrome:

```
>> s=mod(r*golay',2)
   s =
   0 0 0 1 0 1 1 0 0 0 0 0
```

Observe that s has weight 3. The errors are in the positions 4, 6, 7 of the first half of r (since G is in systematic form). the correct message is therefore

```
>> z=r(1:12);
>> z([4 6 7])= ( z([4 6 7]))
   z =
   0 1 0 1 0 1 0 1 0 1 0 1
```

Now, the second one.

```
>> r=[0 0 1 0 1 0 1 0 1 0 1 0 0 1 0 0 1 1 1 0 1 1 0 0];
>> s=mod(r*golay',2)
   s =
   0 0 1 0 1 1 0 0 1 0 1 0
>> sB=mod(s*golayb,2)
   sB =
   1 1 1 0 1 0 1 1 0 0 1 0
```

Since both the syndrome and sB have weight greater than 3, we must proceed to the other steps.

```
>> sc1=mod(s+golay(:,13)',2);
>> sc2=mod(s+golay(:,14)',2);
```

```
>> sc3=mod(s+golay(:,15)',2);
>> sc4=mod(s+golay(:,16)',2);
>> sc5=mod(s+golay(:,17)',2);
>> sc6=mod(s+golay(:,18)',2);
>> sc7=mod(s+golay(:,19)',2);
>> sc8=mod(s+golay(:,20)',2);
>> sc9=mod(s+golay(:,21)',2);
>> sc10=mod(s+golay(:,22)',2);
>> sc11=mod(s+golay(:,23)',2);
>> sc12=mod(s+golay(:,24)',2);
```

By using sum(sc1) we can calculate the weight of sc1. Doing this for each of them, we find that none of the weights are less than or equal to 2. We therefore must try step 6 from the book.

```
>> sb1=mod(sB+golayb(1,:),2);
>> sb2=mod(sB+golayb(2,:),2);
>> sb3=mod(sB+golayb(3,:),2);
>> sb4=mod(sB+golayb(4,:),2);
>> sb5=mod(sB+golayb(5,:),2);
>> sb6=mod(sB+golayb(6,:),2);
>> sb7=mod(sB+golayb(7,:),2);
>> sb8=mod(sB+golayb(8,:),2);
>> sb9=mod(sB+golayb(9,:),2);
>> sb10=mod(sB+golayb(10,:),2);
>> sb11=mod(sB+golayb(11,:),2);
>> sb12=mod(sB+golayb(12,:),2);
```

Observe that the weight of sb1 is 2, and hence $e_1 = 1$. Also, there are two errors in the parity bits, namely $e_{18} = 1$ and $e_{20} = 1$. We can calculate the original message by changing the first bit of the first 12 bits of r.

```
>> z=r(1:12);
>> z(1)= z(1)
   z =
   1 0 1 0 1 0 1 0 1 0 1 0
```

Finally, the third one.

```
>> r=[1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 ];
>> s=mod(r*golay',2);
>> sB=mod(s*golayb,2);
```

Both s and sB have weight greater than 3. We calculate the weight of $s + c_j^T$ for $13 \leq j \leq 24$ by:

```
>> sc1=mod(s+golay(:,13)',2);
>> sc2=mod(s+golay(:,14)',2);
>> sc3=mod(s+golay(:,15)',2);
>> sc4=mod(s+golay(:,16)',2);
>> sc5=mod(s+golay(:,17)',2);
>> sc6=mod(s+golay(:,18)',2);
>> sc7=mod(s+golay(:,19)',2);
>> sc8=mod(s+golay(:,20)',2);
>> sc9=mod(s+golay(:,21)',2);
>> sc10=mod(s+golay(:,22)',2);
>> sc11=mod(s+golay(:,23)',2);
>> sc12=mod(s+golay(:,24)',2);
```

None of these have weight ≤ 2, and so we calculate $sB + b_j^T$:

```
>> sb1=mod(sB+golayb(1,:),2);
>> sb2=mod(sB+golayb(2,:),2);
>> sb3=mod(sB+golayb(3,:),2);
>> sb4=mod(sB+golayb(4,:),2);
>> sb5=mod(sB+golayb(5,:),2);
>> sb6=mod(sB+golayb(6,:),2);
>> sb7=mod(sB+golayb(7,:),2);
>> sb8=mod(sB+golayb(8,:),2);
>> sb9=mod(sB+golayb(9,:),2);
>> sb10=mod(sB+golayb(10,:),2);
>> sb11=mod(sB+golayb(11,:),2);
>> sb12=mod(sB+golayb(12,:),2);
```

Only sb4 has weight less than 2, hence $e_4 = 1$. Since sb4 has a non-zero entry in the 4th position, we know $e_{16} = 1$.

2.

```
>> r=[0 1 1 0 0 0 1 1 0 0 0 1 0 1 0];
>> s=mod(r*hammingpc',2)
   s =
   1 1 0 0
```

Observe that s corresponds to the 8th column of the parity check matrix. Hence we need to flip the 8th bit of the received vector to correct the error. Only the first 11 bits corresponds to information bits, and the correct message can be obtained by:

```
>> z=r;
>> z(8)= z(8);
```

```
>> z(1:11)

ans =
   0 1 1 0 0 0 1 0 0 0 0
```