APPENDIX A
Mathematica® Examples

These computer examples are written in Mathematica. If you have Mathematica available, you should try some of them on your computer. If Mathematica is not available, it is still possible to read the examples. They provide examples for several of the concepts of this book. For information on getting started with Mathematica, see Section A.1. To download a Mathematica notebook that contains these commands, go to

http://www.prenhall.com/washington

A.1 Getting Started with Mathematica

1. Download the Mathematica notebook crypto.nb that you find using the links starting at *http://www.prenhall.com/washington*

2. Open Mathematica, and then open crypto.nb using the menu options under File on the command bar at the top of the Mathematica window. (Perhaps this is done automatically when you download it; it depends on your computer settings.)

3. With crypto.nb in the foreground, click (left button) on Kernel on the command bar. A menu will appear. Its first line will read Evaluation. Move the arrow so it is on this line. A submenu will appear. Move the arrow down to the line Evaluate Notebook and click (left button). This evaluates the notebook and loads the necessary functions. Ignore any warning messages about spelling. They occur because a few functions have similar names.

4. Go to the command bar at the top and click on File. Move the arrow down to New and click. A new notebook will appear on top of crypto.nb. However, all the commands of crypto.nb will still be working.

5. If you want to give the new notebook a name, use the File command and scroll down to Save As.... Then save under some name with a .nb at the end.

6. You are now ready to use Mathematica. If you want to try something easy, type 1+2*3+4^5 and then press the Shift and Enter keys simultaneously. Or, if your keyboard has a number pad with Enter, probably on the right side of the keyboard, you can press that (without the Shift). The result 1031 should appear (it's $1 + 2 \cdot 3 + 4^5$).

7. Turn to the Computer Examples Section A.3. Try typing in some of the commands there. The outputs should be the same as that in the examples. Remember to press Shift Enter (or the numeric Enter) to make Mathematica evaluate an expression.

8. If you want to delete part of your notebook, simply move the arrow to the blue line at the right edge of the window and click the left button. The highlighted part can be deleted by clicking on Edit on the top command bar, then clicking on Cut on the menu that appears.

9. Save your notebook by clicking on File on the command bar, then clicking on Save on the menu that appears.

10. Print your notebook by clicking on File on the command bar, then clicking on Print on the menu that appears. (You will see the advantage of opening a new notebook in Step 4; if you didn't open one, then all the commands in crypto.nb will also be printed.)

11. If you make a mistake in typing in a command and get an error message, you can edit the command and hit Shift Enter to try again. You don't need to retype everything.

12. If a program seems to be running for a very long time, you can sometimes stop it by clicking on Kernel and Abort Evaluation. If this doesn't work, there is always the Off button on the computer.

13. Look at the commands available through the command bar at the top. For example, Format then Style allows you to change the type font on any cell that has been highlighted (by clicking on its blue bar on the right side).

14. If you are looking for help or a command to do something, try the Help command. The Master Index leads to a lot of useful information. Note that the commands that are built into Mathematica always start with capital letters. The commands that are coming from crypto.nb start with small letters and will not be found in the Help Index.

15. Some of the number theory and plotting commands require that special packages be loaded (for example, see Example 7 for Chapter 3). These are automatically loaded when the notebook from the Web site is evaluated. If the commands are used independently of that notebook, don't forget to load the packages. One way to identify which packages are needed is to look up the commands in the Master Index.

A.2 Som

The following ar
Examples. The
built into Mathe
been written spe
at

addell[{x,y
the elliptic curv
affinecrypt
allshifts[txt
ChineseRe
multaneous cong
choose[txt,
coinc[txt,n
corr[v] the
frequency vector
EulerPhi[n
ExtendedG
$mx + ny = \gcd$.
FactorInteg
frequency[t
txt.
GCD[m,n]
Inverse[M]
lfsr[c,k,n] g
coefficients give
vector k.
lfsrlength[v
of length at mos
lfsrsolve[v,
the binary vecto
Max[v] is t
Mod[a,n] is
multell[{x,
curve $y^2 \equiv x^3 -$
multsell[{x
the elliptic curv
NextPrime
must be loaded
num2text0
must each be at

A.2 Some Commands

The following are some Mathematica commands that are used in the Computer Examples. The commands that start with capital letters, such as **EulerPhi**, are built into Mathematica. The ones that start with small letters, such as **addell**, have been written specially for this text and are in the Mathematica notebook available at

http://www.prenhall.com/washington

addell[{x,y}, {u,v}, b, c, n] finds the sum of the points $\{x, y\}$ and $\{u, v\}$ on the elliptic curve $y^2 \equiv x^3 + bx + c \pmod{n}$, where n is odd.

affinecrypt[txt,m,n] affine encryption of txt using $mx + n$.

allshifts[txt] gives all 26 shifts of txt.

ChineseRemainderTheorem[{a,b,...},{m,n,...}] gives a solution to the simultaneous congruences $x \equiv a \pmod{m}, x \equiv b \pmod{n}, \ldots$.

choose[txt,m,n] lists the characters in txt in positions congruent to $n \pmod{m}$.

coinc[txt,n] the number of matches between txt and txt shifted by n.

corr[v] the dot product of the vector v with the 26 shifts of the alphabet frequency vector.

EulerPhi[n] computes $\phi(n)$ (don't try very large values of n).

ExtendedGCD[m,n] computes the gcd of m and n along with a solution of $mx + ny = \gcd$.

FactorInteger[n] factors n.

frequency[txt] lists the number of occurrences of each letter a through z in txt.

GCD[m,n] is the gcd of m and n.

Inverse[M] finds the inverse of the matrix M.

lfsr[c,k,n] gives the sequence of n bits produced by the recurrence that has coefficients given by the vector c. The initial values of the bits are given by the vector k.

lfsrlength[v,n] tests the vector v of bits to see if it is generated by a recurrence of length at most n.

lfsrsolve[v,n] given a guess n for the length of the recurrence that generates the binary vector v, it computes the coefficients of the recurrence.

Max[v] is the largest element of the vector v.

Mod[a,n] is the value of $a \pmod{n}$.

multell[{x,y}, m, b, c, n] computes m times the point $\{x, y\}$ on the elliptic curve $y^2 \equiv x^3 + bx + c \pmod{n}$.

multsell[{x,y}, m, b, c, n] lists the first m multiples of the point $\{x, y\}$ on the elliptic curve $y^2 \equiv x^3 + bx + c \pmod{n}$.

NextPrime[x] gives the next prime $> x$ (the NumberTheoryFunctions package must be loaded).

num2text0[n] changes a number n to letters. The successive pairs of digits must each be at most 25; a is 00, z is 25.

num2text[n] changes a number n to letters. The successive pairs of digits must each be at most 26; *space* is 00, a is 01, z is 26.

PowerMod[a,b,n] computes $a^b \pmod{n}$.

PrimitiveRoot[p] finds a primitive root for the prime p.

shift[txt,n] shifts txt by n.

txt2num0[txt] changes txt to numbers, with $a = 00, \ldots, z = 25$.

txt2num[txt] changes txt to numbers, with *space*=00, $a = 01, \ldots, z = 26$.

vigenere[txt,v] gives the Vigenère encryption of txt using the vector v.

vigvec[txt,m,n] gives the frequencies of the letters a through z in positions congruent to $n \pmod{m}$.

A.3 Examples for Chapter 2

Example 1. A shift cipher was used to obtain the ciphertext **kddmu**. Decrypt it by trying all possibilities.

In[1]:= **allshifts["kddkmu"]**

```
kddkmu
leelnv
mffmow
nggnpx
ohhoqy
piiprz
qjjqsa
rkkrtb
sllsuc
tmmtvd
unnuwe
voovxf
wppwyg
xqqxzh
yrryai
zsszbj
attack
buubdl
cvvcem
dwwdfn
exxego
fyyfhp
gzzgiq
haahjr
ibbiks
jccjlt
```

As you can see, *attack* is the only word that occurs on this list, so that was the plaintext.

Example 2. Encrypt the plaintext message *cleopatra* using the affine function $7x + 8$:

In[2]:=**affinecrypt["cleopatra", 7, 8]**

Out[2]=`whkcjilxi`

Example 3. The ciphertext *mzdvezc* was encrypted using the affine function $5x + 12$. Decrypt it.

Solution: First, solve $y \equiv 5x + 12 \pmod{26}$ for x to obtain $x \equiv 5^{-1}(y - 12)$. We need to find the inverse of 5 (mod 26):

In[3]:= **PowerMod[5, -1, 26]**

Out[3]= `21`

Therefore, $x \equiv 21(y - 12) \equiv 21y - 12 \cdot 21$. To change $-12 \cdot 21$ to standard form:

In[4]:= **Mod[-12*21, 26]**

Out[4]= `8`

Therefore, the decryption function is $x \equiv 21y + 8$. To decrypt the message:

In[5]:= **affinecrypt["mzdvezc", 21, 8]**

Out[5]= `anthony`

In case you were wondering, the plaintext was encrypted as follows:

In[6]:= **affinecrypt["anthony", 5, 12]**

Out[6]= `mzdvezc`

Example 4. Here is the example of a Vigenère cipher from the text. Let's see how to produce the data that was used in Section 2.3 to decrypt it. For convenience, we've already stored the ciphertext under the name *vvhq*.

In[7]:= **vvhq**

Out[7]=

```
vvhqwvvrhmusgjgthkihtssejchlsfcbgvwcrlryqtfsvgahwkcuhwauglqhnslrljs
hbltspisprdxljsveeghlqwkasskuwepwqtwvspgoelkcqyfnsvwljsniqkgnrgybwl
wgoviokhkazkqkxzgyhcecmeiujoqkwfwvefqhkijrclrlkbienqfrjljsdhgrhlsfq
twlauqrhwdmwlgusgikkflryvcwvspgpmlkassjvoqxeggveyggzmljcxxljsvpaivw
ikvrdrygfrjljslveggveyggeiapuuisfpbtgnwwmuczrvtwglrwugumnczvile
```

Find the frequencies of the letters in the ciphertext:

In[8]:= **frequency[vvhq]**

Out[8]=
```
{{a, 8}, {b, 5}, {c, 12}, {d, 4}, {e, 15}, {f, 10}, {g, 27},
{h, 16}, {i, 13}, {j, 14}, {k, 17}, {l, 25}, {m, 7}, {n, 7},
{o, 5}, {p, 9}, {q, 14}, {r, 17}, {s, 24}, {t, 8}, {u, 12},
{v, 22}, {w, 22}, {x, 5}, {y, 8}, {z, 5}}
```

Let's compute the coincidences for shifts of 1, 2, 3, 4, 5, 6:

In[9]:= **coinc[vvhq, 1]**

Out[9]= 14

In[10]:= **coinc[vvhq, 2]**

Out[10]= 14

In[11]:= **coinc[vvhq, 3]**

Out[11]= 16

In[12]:= **coinc[vvhq, 4]**

Out[12]= 14

In[13]:= **coinc[vvhq, 5]**

Out[13]= 24

In[14]:= **coinc[vvhq, 6]**

Out[14]= 12

We conclude that the key length is probably 5. Let's look at the 1st, 6th, 11th, ... letters (namely, the letters in positions congruent to 1 mod 5):

In[15]:= **choose[vvhq, 5, 1]**

Out[15]=
```
vvuttcccqgcunjtpjgkuqpknjkygkkgcjfqrkqjrqudukvpkvggjjivgjggpfncwuce
```

In[16]:= **frequency[%]**

Out[16]=
```
{{a, 0}, {b, 0}, {c, 7}, {d, 1}, {e, 1}, {f, 2},
{g, 9}, {h, 0}, {i, 1}, {j, 8}, {k, 8}, {l, 0}, {m, 0}, {n, 3},
{o, 0}, {p, 4}, {q, 5}, {r, 2}, {s, 0}, {t, 3}, {u, 6}, {v, 5},
{w, 1}, {x, 0}, {y, 1}, {z, 0}}
```

To express this as a vector of frequencies:

In[17]:= **vigvec[vvhq, 5, 1]**

Out[17]=
```
{0, 0, 0.104478, 0.0149254, 0.0149254, 0.0298507,
0.134328, 0, 0.0149254, 0.119403, 0.119403, 0, 0, 0.0447761,
```



```
0, 0.0597015,
0.0746269, 0.0
```

The dot products
computed as follo

In[18]:= **corr[%]**

Out[18]=
```
{0.0250149, 0.
0.051209, 0.03
0.0342687, 0.0
0.0391791, 0.0
0.0488358, 0.0
```

The third entry i
One way to find

In[19]:= **Max[%**

Out[19]= 0.0713

Now it is easy to
once). Since it oc
by 2, correspondi
vigvec[vvhq, 5,2
3, 4, 18. Let's ch

In[20]:= **vigener**

Out[20]=

```
themethodusedf
extremeandatth
eeasewithwhich
fthiscodebytho
estdangerofthe
```

For the record, th

In[21]:= **vigener**

Out[21]=

```
vvhqwvvrhmusgj
hbltspisprdxlj
wgoviokhkazkqk
twlauqrhwdmwlg
ikvrdrygfrjljs
```

```
0, 0.0597015, 0.0746269, 0.0298507, 0, 0.0447761, 0.0895522,
0.0746269, 0.0149254, 0, 0.0149254, 0}
```

The dot products of this vector with the shifts of the alphabet frequency vector are computed as follows:

In[18]:= **corr**[%]

Out[18]=

```
{0.0250149, 0.0391045, 0.0713284, 0.0388209, 0.0274925, 0.0380149,
0.051209, 0.0301493, 0.0324776, 0.0430299, 0.0337761, 0.0298507,
0.0342687, 0.0445672, 0.0355522, 0.0402239, 0.0434328, 0.0501791,
0.0391791, 0.0295821, 0.0326269, 0.0391791, 0.0365522, 0.0316119,
0.0488358, 0.0349403}
```

The third entry is the maximum, but sometimes the largest entry is hard to locate. One way to find it is

In[19]:= **Max**[%]

Out[19]= 0.0713284

Now it is easy to look through the list and find this number (it usually occurs only once). Since it occurs in the third position, the first shift for this Vigenère cipher is by 2, corresponding to the letter *c*. A procedure similar to the one just used (using *vigvec[vvhq, 5,2],...*, *vigvec[vvhq,5,5]*) shows that the other shifts are probably 14, 3, 4, 18. Let's check that we have the correct key by decrypting.

In[20]:= **vigenere[vvhq, -{2, 14, 3, 4, 18}]**

Out[20]=

```
themethodusedforthepreparationandreadingofcodemessagesissimpleinthe
extremeandatthesametimeimpossibleoftranslationunlessthekeyisknownth
eeasewithwhichthekeymaybechangedisanotherpointinfavoroftheadoptiono
fthiscodebythosedesiringtotransmitimportantmessageswithouttheslight
estdangeroftheirmessagesbeingreadbypoliticalorbusinessrivalsetc
```

For the record, the plaintext was originally encrypted by the command

In[21]:= **vigenere**[%, **{2, 14, 3, 4, 18}**]

Out[21]=

```
vvhqwvvrhmusgjgthkihtssejchlsfcbgvwcrlryqtfsvgahwkcuhwauglqhnslrljs
hbltspisprdxljsveeghlqwkasskuwepwqtwvspgoelkcqyfnsvwljsniqkgnrgybwl
wgoviokhkazkqkxzgyhcecmeiujoqkwfwvefqhkijrclrlkbienqfrjljsdhgrhlsfq
twlauqrhwdmwlgusgikkflryvcwvspgpmlkassjvoqxeggveyggzmljcxxljsvpaivw
ikvrdrygfrjljslveggveyggeiapuuisfpbtgnwwmuczrvtwglrwugumnczvile
```

Example 5. The ciphertext

$$22, 09, 00, 12, 03, 01, 10, 03, 04, 08, 01, 17$$

was encrypted using a Hill cipher with matrix

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 10 \end{pmatrix}.$$

Decrypt it.

Solution: A matrix $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ is entered as $\{\{a,b\},\{c,d\}\}$. Type $M.N$ to multiply matrices M and N. Type $v.M$ to multiply a vector v on the right by a matrix M.

First, we need to invert the matrix mod 26:

In[22]:= **Inverse[{{ 1,2,3},{ 4,5,6},{7,8,10}}]**

Out[22]= {{-2/3, -4/3, 1}, {2/3, 11/3, -2}, {1, -2, 1}}

Since we are working mod 26, we can't stop with numbers like 2/3. We need to get rid of the denominators and reduce mod 26. To do so, we multiply by 3 to extract the numerators of the fractions, then multiply by the inverse of 3 mod 26 to put the "denominators" back in (see Section 3.3):

In[23]:= **%*3**

Out[23]= {{-2, -4, 3}, {-2, 11, -6}, {3, -6, 3}}

In[24]:= **Mod[PowerMod[3, -1, 26]*%, 26]**

Out[24]= {{8,16,1}, {8,21,24}, {1,24,1}}

This is the inverse of the matrix mod 26. We can check this as follows:

In[25]:= **Mod[%.{{1, 2, 3}, {4, 5, 6}, {7, 8, 10}}, 26]**

Out[25]= {{1, 0, 0}, {0, 1, 0}, {0, 0, 1}}

To decrypt, we break the ciphertext into blocks of three numbers and multiply each block on the right by the inverse matrix we just calculated:

In[26]:= **Mod[{22, 09, 00}.%%, 26]**

Out[26]= {14, 21, 4}

In[27]:= **Mod[{12, 03, 01}.%%%, 26]**

Out[27]= {17, 19, 7}

In[28]:= **Mod[{10, 03, 04}.%%%%, 26]**

Out[28]= {4, 7, 8}

In[29]:= **Mod[{0**

Out[29]= {11, 1

Therefore, the p
changed back to

In[30]:= **alph0[1**

Out[30]= overth

Note that the fin
three letters.

Example 6. (

The initial value
Solution: Th
given by the vect

In[31]:= **lfsr[{1.**

Out[31]= {0, 1
0, 0, 0, 1, 1,
0, 1, 0, 1, 1,

Example 7. 1, 0, 0, 0. 0. 1. 1.

Solution: Fir
n] calculates the
procedure in Sec

In[32]:=
lfsrlength[{1. 0

{1, 1}
{2, 1}
{3, 0}
{4, 1}
{5, 0}
{6, 1}
{7, 0}
{8, 0}
{9, 0}
{10, 0}

The last nonzero
length 6. To find

In[33]:= **lfsrsolv**

In[29]:= **Mod[{08, 01, 17}.%%%%%, 26]**

Out[29]= `{11, 11, 23}`

Therefore, the plaintext is 14, 21, 4, 17, 19, 7, 4, 7, 8, 11, 11, 23. This can be changed back to letters:

In[30]:= **alph0[142104171907040708111123]**

Out[30]= `overthehillx`

Note that the final `x` was appended to the plaintext in order to complete a block of three letters.

Example 6. Compute the first 50 terms of the recurrence

$$x_{n+5} \equiv x_n + x_{n+2} \pmod 2.$$

The initial values are $0, 1, 0, 0, 0$.

Solution: The vector of coefficients is $\{1, 0, 1, 0, 0\}$ and the initial values are given by the vector $\{0, 1, 0, 0, 0\}$. Type

In[31]:= **lfsr[{1, 0, 1, 0, 0}, {0, 1, 0, 0, 0}, 50]**

Out[31]= `{0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1,`
`0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0,`
`0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1}`

Example 7. Suppose the first 20 terms of an LFSR sequence are 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1. Find a recurrence that generates this sequence.

Solution: First, we find the length of the recurrence. The command *lfsrlength[v, n]* calculates the determinants mod 2 of the first n matrices that appear in the procedure in Section 2.11:

In[32]:=
lfsrlength[{1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1}, 10]

```
{1, 1}
{2, 1}
{3, 0}
{4, 1}
{5, 0}
{6, 1}
{7, 0}
{8, 0}
{9, 0}
{10, 0}
```

The last nonzero determinant is the sixth one, so we guess that the recurrence has length 6. To find the coefficients:

In[33]:= **lfsrsolve[{1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1}, 6]**

```
Out[33]= {1, 0, 1, 1, 1, 0}
```

This gives the recurrence as

$$x_{n+6} \equiv x_n + x_{n+2} + x_{n+3} + x_{n+4} \pmod 2.$$

Example 8. The ciphertext 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0 was produced by adding the output of a LFSR onto the plaintext mod 2 (i.e., XOR the plaintext with the LFSR output). Suppose you know that the plaintext starts 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0. Find the rest of the plaintext.

Solution: XOR the ciphertext with the known part of the plaintext to obtain the beginning of the LFSR output:

```
In[34]:= Mod[{1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0} + {0, 1, 1, 0,
1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1}, 2]

Out[34]= {1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1}
```

This is the beginning of the LFSR output. Now let's find the length of the recurrence:

```
In[35]:= lfsrlength[%, 8]
    {1, 1}
    {2, 0}
    {3, 1}
    {4, 0}
    {5, 1}
    {6, 0}
    {7, 0}
    {8, 0}
```

We guess the length is 5. To find the coefficients of the recurrence:

```
In[36]:= lfsrsolve[%%, 5]

Out[36]= {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:

```
In[37]:= lfsr[{1, 1, 0, 0, 1}, {1, 0, 0, 1, 0}, 40]

Out[37]= {1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0,
1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0}
```

When we XOR the LFSR output with the ciphertext, we get back the plaintext:

```
In[38]:= Mod[% + {0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1,
0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0}, 2]

Out[38]= {1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1,
1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0}
```

This is the plaintext.