

Cryptography Math 178. Fall 2009. MWF 10:30 - 11:35, O'Connor 104, Section 52266

Professor Ed Schaefer, 309 O'Connor Hall, 554-6899, eschaefer@scu.edu
math.scu.edu/~eschaefer/index.html (or Google "Ed Schaefer").

Office Hours: Monday 1:40 - 2:40, Tuesday 1 - 2, Wednesday 8:40 - 9:40, Thursday 4 - 5 and by appointment.

Midterm, Friday October 30. The final is on Monday, December 7 at 1:30. Your grade is broken up as: Homework 20%, Midterm 30%, Final 50%.

Cryptography is about hiding information. As technology becomes increasingly involved in communication, cryptography becomes increasingly important. There are issues of security relating to cellular phones, e-mail, ATM pin numbers, making purchases over the internet, electronic cash, financial transactions over phone lines, communications among spies and among world leaders, keeping hackers out of bank accounts, authenticating electronic signatures and more. We can say that any message involves three people: the sender, the proper receiver and an eavesdropper. This course is primarily from the point of view of the first two people. The second course, Cryptanalysis (the breaking of codes), is from the point of view of the eavesdropper.

Our goal is to answer four questions: How is encryption done now? How can you arrange to encrypt a message with someone you will never meet and hide it from other people you will never meet? How can you be convinced that the person with whom you are communicating on-line is who they say they are? How can you make sure that a message you receive has not been tampered with? In fact, all four of these questions must be answered in order to understand how a credit card transaction occurs when you purchase something on the internet.

We will spend the first 2.5 weeks covering terminology, history, number theory, simple cryptosystems, and cryptanalysis. We will spend the last 7.5 weeks learning about cryptosystems in use today.

For those interested, I recommend further reading from RSA's FAQ on line. It can be accessed from my website.

For COEN majors, this course counts as one of your three COEN 1XX electives. You need to complete a Petition to Waiver form, requesting permission for the substitution, and then have it signed by your advisor, and then give it to Prof. Lewis (or Prof. Davis, I'm not sure). For CS majors, this counts as one of the two "upper division mathematics courses not on the list".

Those of you who have taken classes from me are used to practice problems for exams. I write those to show students what I think is important and to supplement the text with problems I like. This is not necessary in this course since I wrote all of the homework.

End syllabus

Cryptography computer labs manual

The program `dan.exe` was written by Dan Fisher (who took this class in 1993). It contains programs for this class. PARI was written by 4 Frenchmen who are both number theorists and computer scientists. The Windows version of PARI is an afterthought (ah UNIX!), so if you have a home computer/laptop, you may have trouble running it at home, if so, see me.

`dan`: This program has 4 options at first: 1) letters \leftrightarrow numbers, 2) matrix encryption/decryption, 3) baby DES, x) exit. We won't use baby DES. Don't type **1**), for example, just type **1**.

If you pick 1), then you get 3 new options, 1) letters to numbers, 2) numbers to letters, x) exit.

letters to numbers encodes a string of letters, digits and spaces into (at most) 24 digit integers in a weird way. Once you go over 24 digits, it will start a new number, you won't see this, but when you write the output to a file, that's how it will be broken up. A-Z become 00-25, 0-9 become 26-35 and *space* becomes 36. The program doesn't care if you type in upper or lower case letters. The program puts a *space*=36 at the beginning of each string so that the output numbers won't start with a 0. It would turn THE into 36190704 which is an integer around 36 million. Usually you will want to save the output data. When it asks, type **y** or **n** for yes or no. Call the output file *xxx.txt* but not *xxx.out* where *xxx* could be any string of letters. You will kill PARI and have to delete it and get a new copy if there are files with *.out* floating around or files with strange characters (even digits) in them. So *a.txt* is a nice thing to call the output for example. The output will be saved in your directory.

numbers to letters does the reverse. It takes a number like 190704 and changes it back to THE. You will usually be reading data from an output file that you made in PARI. This program doesn't write to a file, so write out the plaintext message by hand.

If you pick 2) matrix encryption/decryption, then you get 3 new options, 1) matrix encryption, 2) matrix decryption, x) exit.

matrix encryption is a symmetric (shared/secret) key encryption. Since we won't go over matrix encryption in class, you should think of it as a black box symmetric key cryptosystem. (AES is another symmetric key cryptosystem.) You will usually save the output. Remember: **end output files with .txt not .out**. The output file will be saved in the directory. The messages may consist of letters, digits and spaces. A-Z become 00-25, 0-9 become 26-35 and *space* becomes 36. The program doesn't care if you type in upper or lower case letters. *matrix encryption* operates on digraphs. If the number of input characters is odd, it will add a space and tell you that. The program will then decode the numbers back into ciphertext digraphs. This program will usually be used after a key exchange, when we want to focus on the key exchange and not on the enciphering transformation.

matrix decryption reverses this. You will need the same 12 digit key. I believe there is a bug in matrix encrypt/decrypt. So don't get too frustrated if something doesn't work. It does work for all the homework this quarter.

This paragraph is an aside for those who are interested in peeking inside the black box and know how to multiply a matrix times a vector. If the 12 digit key is 123456789055 then the message that will be entered will be encoded as digrams (TH becomes 19 07) and then the

encryption is by

$$\begin{bmatrix} 12 & 34 \\ 56 & 78 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 90 \\ 55 \end{bmatrix} \pmod{37}$$

where x and y might be 19 and 7 for the digraph TH. If the determinant is $0 \pmod{37}$, then the program will fix this problem, you needn't worry about that.

PARI is essentially a calculator. The 2 most important things to know about it are to never end the name of an output file with *.out*, always end them with *.txt* like *a.txt* and use only letters and a period in the filename. The second is that you type $\backslash q$ to get out. If PARI gets hung up because you ask it to do something too hard (like factor a 100 digit number) then you will have to kill it.

PARI does all the usual operations $+, -, *, /, \wedge$ for plus, minus, times, divides, and powers. You can use parentheses too. Below will be a short PARI session, what I type is boldface.

```
? 2+2
%1 = 4
? 3^5
%2 = 243
? %1+%2
%3 = 247
```

Comment: Mod(a,m) means $a \pmod{m}$; you must use a capital M. Back to PARI:

```
? Mod(78,5)
%4=Mod(3,5)
```

Comment: To do powers mod m , do the power outside. To reduce $2^{1000000} \pmod{77}$ do the following:

```
? Mod(2,77)^1000000
%5 = Mod(23,77)
? Mod(2,7)^-1
%6 = Mod(4,7)
```

Comment: nextprime(a) gives the next prime $\geq a$. I like to give things names other than the ones they already have with the % signs. This is a matter of taste. When numbers have names you don't have to retype them, if you use the same name for 2 different numbers, it will only remember the last one (like in BASIC). Back to PARI:

```
? a=nextprime(100)
%7 = 101
? b=nextprime(200)
%8 = 211
? n=a*b
%9 = 21311
? gcd(a,b)
%10 = 1
```

Comment: If $d = \text{Mod}(10,35)$ then the output of lift(d) is 10, it pulls the c out of $\text{mod}(c,n)$.

```
? d=Mod(100,2345)
%11=Mod(100,2345)
? e=lift(d)
%12=100
```

```
? d
%12=Mod(100,2345)
? e
%13=100
```

Do not start a variable name with a number. `r2` is OK, `2r` is not. Do not use math operations in variable names. `a+b` is not an OK VARIABLE name.

To find out about PARI's other functions if you are interested, type `?`

Reading from and writing to files

Say the file `c.txt` already exists in your directory (may be output from *letters to numbers*, for example). It has 3 numbers on 3 lines and looks like the following:

```
12
15
17
```

Let's start a new session of PARI:

```
? \r c.txt
%1=12
%2=15
%3=17
? %1*%3
%4=204
```

In PARI if you type `\w d.txt` after the line `%n=m` it will write `m` to the file `d.txt`.

Example:

```
? a=16
%1=16
? \w d.txt
? b=14
%2=14
? \w d.txt
? n=a*b
%3=224
? \w d.txt
```

Then there will be a file called `d.txt` left in your directory that will have 3 lines, the numbers 16, 14 and 224.

Warning: there are already files called `n.txt` and `e.txt` on your CD so avoid these names for files that you create.

You can type `\l` while in PARI and everything thereafter will appear in a file called `pari.log`. If you want to play with PARI then type `?` and explore. You can program within PARI. A document explaining how is at my website:
math.scu.edu/~eschaefer/index.html.

End computer labs manual

Encoding handout

ASCII

00100000	32	8	00111000	56	P	01010000	80	h	01101000	104	
!	00100001	33	9	00111001	57	Q	01010001	81	i	01101001	105
"	00100010	34	:	00111010	58	R	01010010	82	j	01101010	106
#	00100011	35	;	00111011	59	S	01010011	83	k	01101011	107
\$	00100100	36	<	00111100	60	T	01010100	84	l	01101100	108
%	00100101	37	=	00111101	61	U	01010101	85	m	01101101	109
&	00100110	38	>	00111110	62	V	01010110	86	n	01101110	110
'	00100111	39	?	00111111	63	W	01010111	87	o	01101111	111
(00101000	40	@	01000000	64	X	01011000	88	p	01110000	112
)	00101001	41	A	01000001	65	Y	01011001	89	q	01110001	113
*	00101010	42	B	01000010	66	Z	01011010	90	r	01110010	114
+	00101011	43	C	01000011	67	[01011011	91	s	01110011	115
,	00101100	44	D	01000100	68	\	01011100	92	t	01110100	116
-	00101101	45	E	01000101	69]	01011101	93	u	01110101	117
.	00101110	46	F	01000110	70	^	01011110	94	v	01110110	118
/	00101111	47	G	01000111	71	_	01011111	95	w	01110111	119
0	00110000	48	H	01001000	72	`	01100000	96	x	01111000	120
1	00110001	49	I	01001001	73	a	01100001	97	y	01111001	121
2	00110010	50	J	01001010	74	b	01100010	98	z	01111010	122
3	00110011	51	K	01001011	75	c	01100011	99	{	01111011	123
4	00110100	52	L	01001100	76	d	01100100	100		01111100	124
5	00110101	53	M	01001101	77	e	01100101	101	}	01111101	125
6	00110110	54	N	01001110	78	f	01100110	102	~	01111110	126
7	00110111	55	O	01001111	79	g	01100111	103			

Alphabet encodings

a	0	00000	b	1	00001	c	2	00010	d	3	00011	e	4	00100
f	5	00101	g	6	00110	h	7	00111	i	8	01000	j	9	01001
k	10	01010	l	11	01011	m	12	01100	n	13	01101	o	14	01110
p	15	01111	q	16	10000	r	17	10001	s	18	10010	t	19	10011
u	20	10100	v	21	10101	w	22	10110	x	23	10111	y	24	11000
z	25	11001												

End of Encoding handout

1 Introduction

The U.S. government in the early 1970's wanted an encryption process on a small chip that would be widely used and safe. In 1975 they accepted IBM's Data Encryption Standard Algorithm (DES). DES is a symmetric-key cryptosystem which has a 56-bit key and encrypts 64-bit plaintexts to 64-bit ciphertexts. By the early 1990's, the 56-bit key was considered too short. So people started using Triple-DES. That is using DES three times (with two different keys, first, second, first, so we have 112 key bits) to encrypt a 64-bit plaintext. (Surprisingly, Double-DES with two different keys is not much safer than DES, as I'll explain in the Cryptanalysis course). However, DES was not designed with Triple-DES in mind. Undoubtedly there would be a more efficient algorithm with the same level of safety as Triple-DES. So in 1997, the National Institute of Standards and Technology (NIST) solicited proposals for replacements of DES. In 2001, NIST chose 128-bit block Rijndael with a 128-bit key to become the Advanced Encryption Standard (AES). (If you don't speak Dutch, Flemish or Afrikaans, then the closest approximation to the pronunciation is Rine-doll). Rijndael is a symmetric-key block cipher designed by Joan Daemen and Vincent Rijmen.

2 Simplified AES

Simplified AES was designed by Mohammad Musa, Steve Wedig (two former Crypto students) and me in 2002. It is a method of teaching AES. We published the article *A simplified AES algorithm and its linear and differential cryptanalyses* in the journal *Cryptologia* in 2003. We will learn the linear and differential cryptanalyses in the Cryptanalysis Course.

2.1 The Finite Field

Both the key expansion and encryption algorithms of simplified AES depend on an S-box that itself depends on the finite field with 16 elements.

Let $\mathbf{F}_{16} = \mathbf{F}_2[x]/(x^4 + x + 1)$. The word nibble refers to a four-bit string, like 1011. We will frequently associate an element $b_0x^3 + b_1x^2 + b_2x + b_3$ of \mathbf{F}_{16} with the nibble $b_0b_1b_2b_3$.

2.2 The S-box

The S-box is a map from nibbles to nibbles. It can be inverted. (For those in the know, it is one-to-one and onto or bijective.) Here is how it operates. First, invert the nibble in \mathbf{F}_{16} . The inverse of $x + 1$ is $x^3 + x^2 + x$ so 0011 goes to 1110. The nibble 0000 is not invertible, so at this step it is sent to itself. Then associate to the nibble $N = b_0b_1b_2b_3$ (which is the output of the inversion) the element $N(y) = b_0y^3 + b_1y^2 + b_2y + b_3$ in $\mathbf{F}_2[y]/(y^4 + 1)$. Let $a(y) = y^3 + y^2 + 1$ and $b(y) = y^3 + 1$ in $\mathbf{F}_2[y]/(y^4 + 1)$. Doing multiplication and addition is similar to doing so in \mathbf{F}_{16} except that we are working modulo $y^4 + 1$ so $y^4 = 1$ and $y^5 = y$ and $y^6 = y^2$. The second step of the S-box is to send the nibble $N(y)$ to $a(y)N(y) + b(y)$. So the nibble 1110 = $y^3 + y^2 + y$ goes to $(y^3 + y^2 + 1)(y^3 + y^2 + y) + (y^3 + 1)$

$= (y^6 + y^5 + y^4) + (y^5 + y^4 + y^3) + (y^3 + y^2 + y) + (y^3 + 1) = y^2 + y + 1 + y + 1 + y^3 + y^3 + y^2 + y + y^3 + 1 = 3y^3 + 2y^2 + 3y + 3 = y^3 + y + 1 = 1011$. So $S\text{-box}(0011) = 1011$.

Note that $y^4 + 1 = (y + 1)^4$ is reducible over \mathbf{F}_2 so $\mathbf{F}_2[y]/(y^4 + 1)$ is not a field and not all of its non-zero elements are invertible; the polynomial $a(y)$, however, is. If you read the literature, then the second step is often described by an affine matrix map.

We can represent the action of the S-box in two ways (note we do not show the intermediary output of the inversion)

nib	S-box(nib)	nib	S-box(nib)	
0000	1001	1000	0110	
0001	0100	1001	0010	
0010	1010	1010	0000	
0011	1011	1011	0011	
0100	1101	1100	1100	
0101	0001	1101	1110	
0110	1000	1110	1111	
0111	0101	1111	0111	

or $\begin{bmatrix} 9 & 4 & 10 & 11 \\ 13 & 1 & 8 & 5 \\ 6 & 2 & 0 & 3 \\ 12 & 14 & 15 & 7 \end{bmatrix}$.

The left-hand side is most useful for doing an example by hand. For the matrix on the right, we start in the upper left corner and go across, then to the next row and go across etc. The integers 0 - 15 are associated with their 4-bit binary representations. So 0000 = 0 goes to 9 = 1001, 0001 = 1 goes to 4 = 0100, ... , 0100 = 4 goes to 13 = 1101, etc.

3 Keys

For our simplified version of AES, we have a 16-bit key, which we denote $k_0 \dots k_{15}$. That needs to be expanded to a total of 48 key bits $k_0 \dots k_{47}$, where the first 16 key bits are the same as the original key. Let us describe the expansion. Let $RC[i] = x^{i+2} \in \mathbf{F}_{16}$. So $RC[1] = x^3 = 1000$ and $RC[2] = x^4 = x + 1 = 0011$. If N_0 and N_1 are nibbles, then we denote their concatenation by N_0N_1 . Let $RCON[i] = RC[i]0000$ (this is a byte, a string of 8 bits). So $RCON[1] = 10000000$ and $RCON[2] = 00110000$. These are abbreviations for *round constant*. We define the function $RotNib$ to be $RotNib(N_0N_1) = N_1N_0$ and the function $SubNib$ to be $SubNib(N_0N_1) = S\text{-box}(N_0)S\text{-box}(N_1)$; these are functions from bytes to bytes. Their names are abbreviations for *rotate nibble* and *substitute nibble*. Let us define an array (vector, if you prefer) W whose entries are bytes. The original key fills $W[0]$ and $W[1]$ in order. For $2 \leq i \leq 5$,

$$\begin{aligned} \text{if } i \equiv 0 \pmod{2} & \text{ then } W[i] = W[i - 2] \oplus RCON(i/2) \oplus SubNib(RotNib(W[i - 1])) \\ \text{if } i \not\equiv 0 \pmod{2} & \text{ then } W[i] = W[i - 2] \oplus W[i - 1] \end{aligned}$$

The bits contained in the entries of W can be denoted $k_0 \dots k_{47}$. For $0 \leq i \leq 2$ we let $K_i = W[2i]W[2i + 1]$. So $K_0 = k_0 \dots k_{15}$, $K_1 = k_{16} \dots k_{31}$ and $K_2 = k_{32} \dots k_{47}$. For $i \geq 1$, K_i is the round key used at the end of the i -th round; K_0 is used before the first round.

Recall \oplus denotes bit-by-bit XORing.

4 Key Expansion Example

Let's say that the key is 0101 1001 0111 1010. So $W[0] = 0101\ 1001$ and $W[1] = 0111\ 1010$. Now $i = 2$ so we $\text{Rotnib}(W[1])=1010\ 0111$. Then we $\text{SubNib}(1010\ 0111)=0000\ 0101$. Then we XOR this with $W[0] \oplus \text{RCON}(1)$ and get $W[2]$.

$$\begin{array}{r} 0000\ 0101 \\ 0101\ 1001 \\ \oplus \underline{1000\ 0000} \\ 1101\ 1100 \end{array}$$

So $W[2] = 11011100$.

Now $i = 3$ so $W[3] = W[1] \oplus W[2] = 0111\ 1010 \oplus 1101\ 1100 = 1010\ 0110$. Now $i = 4$ so we $\text{Rotnib}(W[3])=0110\ 1010$. Then we $\text{SubNib}(0110\ 1010)=1000\ 0000$. Then we XOR this with $W[2] \oplus \text{RCON}(2)$ and get $W[4]$.

$$\begin{array}{r} 1000\ 0000 \\ 1101\ 1100 \\ \oplus \underline{0011\ 0000} \\ 0110\ 1100 \end{array}$$

So $W[4] = 01101100$.

Now $i = 5$ so $W[5] = W[3] \oplus W[4] = 1010\ 0110 \oplus 0110\ 1100 = 1100\ 1010$.

5 The Simplified AES Algorithm

The simplified AES algorithm operates on 16-bit plaintexts and generates 16-bit ciphertexts, using the expanded key $k_0 \dots k_{47}$. The encryption algorithm consists of the composition of 8 functions applied to the plaintext: $A_{K_2} \circ SR \circ NS \circ A_{K_1} \circ MC \circ SR \circ NS \circ A_{K_0}$ (so A_{K_0} is applied first), which will be described below. Each function operates on a state. A state consists of 4 nibbles configured as in Figure 1. The initial state consists of the plaintext as in Figure 2. The final state consists of the ciphertext as in Figure 3.

$b_0b_1b_2b_3$	$b_8b_9b_{10}b_{11}$
$b_4b_5b_6b_7$	$b_{12}b_{13}b_{14}b_{15}$

Figure 1

$p_0p_1p_2p_3$	$p_8p_9p_{10}p_{11}$
$p_4p_5p_6p_7$	$p_{12}p_{13}p_{14}p_{15}$

Figure 2

$c_0c_1c_2c_3$	$c_8c_9c_{10}c_{11}$
$c_4c_5c_6c_7$	$c_{12}c_{13}c_{14}c_{15}$

Figure 3

5.1 The Function A_{K_i}

The abbreviation A_K stands for *add key*. The function A_{K_i} consists of XORing K_i with the state so that the subscripts of the bits in the state and the key bits agree modulo 16.

5.2 The Function NS

The abbreviation NS stands for *nibble substitution*. The function NS replaces each nibble N_i in a state by $S\text{-box}(N_i)$ without changing the order of the nibbles. So it sends the state

$$\begin{array}{|c|c|} \hline N_0 & N_2 \\ \hline N_1 & N_3 \\ \hline \end{array} \text{ to the state } \begin{array}{|c|c|} \hline S\text{-box}(N_0) & S\text{-box}(N_2) \\ \hline S\text{-box}(N_1) & S\text{-box}(N_3) \\ \hline \end{array}.$$

5.3 The Function SR

The abbreviation SR stands for *shift row*. The function SR takes the state

$$\begin{array}{|c|c|} \hline N_0 & N_2 \\ \hline N_1 & N_3 \\ \hline \end{array} \text{ to the state } \begin{array}{|c|c|} \hline N_0 & N_2 \\ \hline N_3 & N_1 \\ \hline \end{array}.$$

5.4 The Function MC

The abbreviation MC stands for *mix column*. A column $[N_i, N_j]$ of the state is considered to be the element $N_i z + N_j$ of $\mathbf{F}_{16}[z]/(z^2 + 1)$. As an example, if the column consists of $[N_i, N_j]$ where $N_i = 1010$ and $N_j = 1001$ then that would be $(x^3 + x)z + (x^3 + 1)$. Like before, $\mathbf{F}_{16}[z]$ denotes polynomials in z with coefficients in \mathbf{F}_{16} . So $\mathbf{F}_{16}[z]/(z^2 + 1)$ means that polynomials are considered modulo $z^2 + 1$; thus $z^2 = 1$. So representatives consist of the 16^2 polynomials of degree less than 2 in z .

The function MC multiplies each column by the polynomial $c(z) = x^2 z + 1$. As an example,

$$\begin{aligned} & [((x^3 + x)z + (x^3 + 1))](x^2 z + 1) = (x^5 + x^3)z^2 + (x^3 + x + x^5 + x^2)z + (x^3 + 1) \\ & = (x^5 + x^3 + x^2 + x)z + (x^5 + x^3 + x^3 + 1) = (x^2 + x + x^3 + x^2 + x)z + (x^2 + x + 1) \\ & = (x^3)z + (x^2 + x + 1), \end{aligned}$$

which goes to the column $[N_k, N_l]$ where $N_k = 1000$ and $N_l = 0111$.

Note that $z^2 + 1 = (z + 1)^2$ is reducible over \mathbf{F}_{16} so $\mathbf{F}_{16}[z]/(z^2 + 1)$ is not a field and not all of its non-zero elements are invertible; the polynomial $c(z)$, however, is.

The simplest way to explain MC is to note that MC sends a column

$$\begin{array}{|c|} \hline b_0 b_1 b_2 b_3 \\ \hline b_4 b_5 b_6 b_7 \\ \hline \end{array} \text{ to } \begin{array}{|c|c|c|c|} \hline b_0 \oplus b_6 & b_1 \oplus b_4 \oplus b_7 & b_2 \oplus b_4 \oplus b_5 & b_3 \oplus b_5 \\ \hline b_2 \oplus b_4 & b_0 \oplus b_3 \oplus b_5 & b_0 \oplus b_1 \oplus b_6 & b_1 \oplus b_7 \\ \hline \end{array}.$$

5.5 The Rounds

The composition of functions $A_{K_i} \circ MC \circ SR \circ NS$ is considered to be the i -th round. So this simplified algorithm has two rounds. There is an extra A_K before the first round and the last round does not have an MC ; the latter will be explained in Section 6.

6 Decryption

Note that for general functions (where the composition and inversion are possible) $(f \circ g)^{-1} = g^{-1} \circ f^{-1}$. Also, if a function composed with itself is the identity map (i.e. gets you back where you started), then it is its own inverse; this is called an involution. This is true of each A_{K_i} . Although it is true for our SR , this is not true for the real SR in AES, so we will not simplify the notation SR^{-1} . Decryption is then by $A_{K_0} \circ NS^{-1} \circ SR^{-1} \circ MC^{-1} \circ A_{K_1} \circ NS^{-1} \circ SR^{-1} \circ A_{K_2}$.

To accomplish NS^{-1} , multiply a nibble by $a(y)^{-1} = y^2 + y + 1$ and add $a(y)^{-1}b(y) = y^3 + y^2$ in $\mathbf{F}_2[y]/(y^4 + 1)$. Then invert the nibble in \mathbf{F}_{16} . Alternately, we can simply use one of the S-box tables from Section 2.2 in reverse.

Since MC is multiplication by $c(z) = x^2z + 1$, the function MC^{-1} is multiplication by $c(z)^{-1} = xz + (x^3 + 1)$ in $\mathbf{F}_{16}[z]/(z^2 + 1)$.

Decryption can be done as above. However to see why there is no MC in the last round, we continue. First note that $NS^{-1} \circ SR^{-1} = SR^{-1} \circ NS^{-1}$. Let St denote a state. We have $MC^{-1}(A_{K_i}(St)) = MC^{-1}(K_i \oplus St) = c(z)^{-1}(K_i \oplus St) = c(z)^{-1}(K_i) \oplus c(z)^{-1}(St) = c(z)^{-1}(K_i) \oplus MC^{-1}(St) = A_{c(z)^{-1}K_i}(MC^{-1}(St))$. So $MC^{-1} \circ A_{K_i} = A_{c(z)^{-1}K_i} \circ MC^{-1}$.

What does $c(z)^{-1}(K_i)$ mean? Break K_i into two bytes $b_0b_1 \dots b_7, b_8, \dots b_{15}$. Consider the first byte

$$\begin{array}{|c|} \hline b_0b_1b_2b_3 \\ \hline b_4b_5b_6b_7 \\ \hline \end{array}$$

to be an element of $\mathbf{F}_{16}[z]/(z^2 + 1)$. Multiply by $c(z)^{-1}$, then convert back to a byte. Do the same with $b_8 \dots b_{15}$. So $c(z)^{-1}K_i$ has 16 bits. $A_{c(z)^{-1}K_i}$ means XOR $c(z)^{-1}K_i$ with the current state. Note when we do MC^{-1} , we will multiply the state by $c(z)^{-1}$ (or more easily, use the equivalent table that you will create in your homework). For $A_{c(z)^{-1}K_1}$, you will first multiply K_1 by $c(z)^{-1}$ (or more easily, use the equivalent table that you will create in your homework), then XOR the result with the current state.

Thus decryption is also

$$A_{K_0} \circ SR^{-1} \circ NS^{-1} \circ A_{c(z)^{-1}K_1} \circ MC^{-1} \circ SR^{-1} \circ NS^{-1} \circ A_{K_2}.$$

Recall that encryption is

$$A_{K_2} \circ SR \circ NS \circ A_{K_1} \circ MC \circ SR \circ NS \circ A_{K_0}.$$

Notice how each kind of operation for decryption appears in exactly the same order as in encryption, except that the round keys have to be applied in reverse order. For the real AES, this can improve implementation. This would not be possible if MC appeared in the last round.

7 Encryption Example

Let's say that we use the key in the above example 0101 1001 0111 1010. So $W[0] = 0101\ 1001$, $W[1] = 0111\ 1010$, $W[2] = 1101\ 1100$, $W[3] = 1010\ 0110$, $W[4] = 0110\ 1100$, $W[5] = 1100\ 1010$,

Let's say that the plaintext is my name 'Ed' in ASCII: 01000101 01100100 Then the initial state is (remembering that the nibbles go in upper left, **then lower left**, then upper right, then lower right)

0100	0110
0101	0100

Then we do A_{K_0} (recall $K_0 = W[0]W[1]$) to get a new state:

0100	0110	=	0001	0001
\oplus 0101	\oplus 0111		1100	1110
0101	0100			
\oplus 1001	\oplus 1010			

Then we apply NS and SR to get

0100	0100	$\rightarrow SR \rightarrow$	0100	0100
1100	1111		1111	1100

Then we apply MC to get

1101	0001
1100	1111

Then we apply A_{K_1} , recall $K_1 = W[2]W[3]$.

1101	0001	=	0000	1011
\oplus 1101	\oplus 1010		0000	1001
1100	1111			
\oplus 1100	\oplus 0110			

Then we apply NS and SR to get

1001	0011	$\rightarrow SR \rightarrow$	1001	0011
1001	0010		0010	1001

Then we apply A_{K_2} , recall $K_2 = W[4]W[5]$.

1001	0011	=	1111	1111
\oplus 0110	\oplus 1100		1110	0011
0010	1001			
\oplus 1100	\oplus 1010			

So the ciphertext is 11111110 11110011.

8 The Real AES

For simplicity, we will describe the version of AES that has a 128-bit key and has 10 rounds. Recall that the AES algorithm operates on 128-bit blocks . We will mostly explain the ways

in which it differs from our simplified version. Each state consists of a four-by-four grid of bytes.

The finite field is $\mathbf{F}_{2^8} = \mathbf{F}_2[x]/(x^8 + x^4 + x^3 + x + 1)$. We let the byte $b_0b_1b_2b_3b_4b_5b_6b_7$ and the element $b_0x^8 + \dots + b_7$ of \mathbf{F}_{2^8} correspond to each other. The S-box first inverts a nibble in \mathbf{F}_{2^8} and then multiplies it by $a(y) = y^4 + y^3 + y^2 + y + 1$ and adds $b(y) = y^6 + y^5 + y + 1$ in $\mathbf{F}_2[y]/(y^8 + 1)$. Note $a(y)^{-1} = y^6 + y^3 + y$ and $a(y)^{-1}b(y) = y^2 + 1$.

The real ByteSub is the obvious generalization of our *NS* - it replaces each byte by its image under the S-box. The real ShiftRow shifts the rows left by 0, 1, 2 and 3. So it sends the state

$$\begin{array}{|c|c|c|c|} \hline B_0 & B_4 & B_8 & B_{12} \\ \hline B_1 & B_5 & B_9 & B_{13} \\ \hline B_2 & B_6 & B_{10} & B_{14} \\ \hline B_3 & B_7 & B_{11} & B_{15} \\ \hline \end{array}
 \quad \text{to the state} \quad
 \begin{array}{|c|c|c|c|} \hline B_0 & B_4 & B_8 & B_{12} \\ \hline B_5 & B_9 & B_{13} & B_1 \\ \hline B_{10} & B_{14} & B_2 & B_6 \\ \hline B_{15} & B_3 & B_7 & B_{11} \\ \hline \end{array}.$$

The real MixColumn multiplies a column by $c(z) = (x+1)z^3 + z^2 + z + x$ in $\mathbf{F}_{2^8}[z]/(z^4+1)$. Also $c(z)^{-1} = (x^3 + x + 1)z^3 + (x^3 + x^2 + 1)z^2 + (x^3 + 1)z + (x^3 + x^2 + x)$. The step *MC* appears in all but the last round. The real AddRoundKey is the obvious generalization of our A_{K_i} . There is an additional AddRoundKey with round key 0 at the beginning of the encryption algorithm.

For key expansion, the entries of the array W are four bytes each. The key fills in $W[0], \dots, W[3]$. The function RotByte cyclically rotates four bytes 1 to the left each, like the action on the second row in ShiftRow. The function SubByte applies the S-box to each byte. $RC[i] = x^i$ in \mathbf{F}_{2^8} and $RCON[i]$ is the concatenation of $RC[i]$ and 3 bytes of all 0's. For $4 \leq i \leq 43$,

$$\begin{aligned}
 \text{if } i \equiv 0 \pmod{4} & \quad \text{then } W[i] = W[i-4] \oplus RCON(i/4) \oplus \text{SubByte}(\text{RotByte}(W[i-1])) \\
 \text{if } i \not\equiv 0 \pmod{4} & \quad \text{then } W[i] = W[i-4] \oplus W[i-1].
 \end{aligned}$$

The i -th key K_i consists of the bits contained in the entries of $W[4i] \dots W[4i+3]$.

9 Analysis of Simplified AES

The enemy intercepts a matched plaintext/ciphertext pair and wants to solve for the key. Let's say the plaintext is $p_0 \dots p_{15}$, the ciphertext is $c_0 \dots c_{15}$ and the key is $k_0 \dots k_{15}$. There are 15 equations of the form

$$f_i(p_0, \dots, p_{15}, k_0, \dots, k_{15}) = c_i$$

where f_i is a polynomial in 32 variables, with coefficients in \mathbf{F}_2 which can be expected to have 2^{31} terms on average. Once we fix the c_j 's and p_j 's (from the known matched plaintext/ciphertext pair) we get 16 non-linear equations in 16 unknowns (the k_i 's). On average these equations should have 2^{15} terms.

Everything in simplified AES is a linear map except for the S-boxes. Let us consider how they operate. Let us denote the input nibble of an S-box by $abcd$ and the output nibble as

$efgh$. Then the operation of the S-boxes can be computed with the following equations

$$\begin{aligned}e &= acd + bcd + ab + ad + cd + a + d + 1 \\f &= abd + bcd + ab + ac + bc + cd + a + b + d \\g &= abc + abd + acd + ab + bc + a + c \\h &= abc + abd + bcd + acd + ac + ad + bd + a + c + d + 1\end{aligned}$$

where all additions are modulo 2. Alternating the linear maps with these non-linear maps leads to very complicated polynomial expressions for the ciphertext bits.

Solving a system of linear equations in several variables is very easy (take Math 53). However, there are no known algorithms for quickly solving systems of non-linear polynomial equations in several variables.

10 Design Rationale

The quality of an encryption algorithm is judged by two main criteria, security and efficiency. In designing AES, Rijmen and Daemen focused on these qualities. They also instilled the algorithm with simplicity and repetition. Security is measured by how well the encryption withstands all known attacks. Efficiency is defined as the combination of encryption/decryption speed and how well the algorithm utilizes resources. These resources include required chip area for hardware implementation and necessary working memory for software implementation. Simplicity refers to the complexity of the cipher's individual steps and as a whole. If these are easy to understand, proper implementation is more likely. Lastly, repetition refers to how the algorithm makes repeated use of functions.

In the following two sections, we will discuss the concepts security, efficiency, simplicity, and repetition with respect to the real AES algorithm.

10.1 Security

As an encryption standard, AES needs to be resistant to all known cryptanalytic attacks. Thus, AES was designed to be resistant against these attacks, especially differential and linear cryptanalysis. To ensure such security, block ciphers in general must have diffusion and non-linearity.

Diffusion is defined by the spread of the bits in the cipher. Full diffusion means that each bit of a state depends on every bit of a previous state. In AES, two consecutive rounds provide full diffusion. The ShiftRow step, the MixColumn step, and the key expansion provide the diffusion necessary for the cipher to withstand known attacks.

Non-linearity is added to the algorithm with the S-Box, which is used in ByteSub and the key expansion. The non-linearity, in particular, comes from inversion in a finite field. This is not a linear map from bytes to bytes. By linear, I mean a map that can be described as map from bytes (i.e. the 8-dimensional vector space over the field \mathbf{F}_2) to bytes which can be computed by multiplying a byte by an 8×8 -matrix and then adding a vector.

Non-linearity increases the cipher's resistance against cryptanalytic attacks. The non-linearity in the key expansion makes it so that knowledge of a part of the cipher key or a round key does not easily enable one to determine many other round key bits.

Simplicity helps to build a cipher's credibility in the following way. The use of simple steps leads people to believe that it is easier to break the cipher and so they attempt to do so. When many attempts fail, the cipher becomes better trusted.

Although repetition has many benefits, it can also make the cipher more vulnerable to certain attacks. The design of AES ensures that repetition does not lead to security holes. For example, the round constants break patterns between the round keys.

10.2 Efficiency

AES is expected to be used on many machines and devices of various sizes and processing powers. For this reason, it was designed to be versatile. Versatility means that the algorithm works efficiently on many platforms, ranging from desktop computers to embedded devices such as cable boxes.

The repetition in the design of AES allows for parallel implementation to increase speed of encryption/decryption. Each step can be broken into independent calculations because of repetition. ByteSub is the same function applied to each byte in the state. MixColumn and ShiftRow work independently on each column and row in the state respectively. The AddKey function can be applied in parallel in several ways.

Repetition of the order of steps for the encryption and decryption processes allows for the same chip to be used for both processes. This leads to reduced hardware costs and increased speed.

Simplicity of the algorithm makes it easier to explain to others, so that the implementation will be obvious and flawless. The coefficients of each polynomial were chosen to minimize computation.

End of AES handout

Encoding a message as a point on an elliptic curve with PARI

Plaintext: L. Use LETTERS TO NUMBERS. Let's pretend that the output is 11 (actually it would be 3611 if you used LETTERS TO NUMBERS. You should leave the 36 when doing the homework). Write it to msg.txt.

```
? \r msg.txt
```

```
%1=11
```

```
? p=257
```

```
? ee=[0,0,0,0,-4]
```

```
? e=Mod(ee,p)
```

```
? g=[2,2]
```

```
? w=x^3-4
```

```
? n=Mod(%1*10,p)
```

```
%7=Mod(110,257)
```

```
? subst(w,x,n)
```

In expression w replace x by n

```
%8=Mod(250,257)
```

```
? sqrt(%8)
```

not quadratic residue

```
? n1=n+1
```

```
%9=Mod(111,257)
```

```
? subst(w,x,n1)
```

```
%10=Mod(130,257)
```

```
? sqrt(%10)
```

not quadratic residue

```
? n2=n1+1
```

```
%11=Mod(112,257)
```

```
? subst(w,x,n2)
```

```
%12=Mod(162,257)
```

```
? sqrt(%12)
```

```
%13=Mod(26,257)
```

```
? q=[n2,%13]
```

```
%14=[Mod(112,257),Mod(26,257)]
```

Aside: Q is our encoded plaintext point on e . Since e is defined mod p , you can enter points of the form $[2,2]$ or $[\text{Mod}(2,257),\text{Mod}(2,257)]$, it doesn't matter.

END: Encoding a message as a point

Certificates and SSL handout

Certificates

There's still a problem not solved yet. How does Amazon know that my public key actually belongs to Ed Schaefer. Maybe Eve will completely impersonate me and say "this public key belongs to Ed Schaefer" and then purchase stuff under my name. How can Amazon be sure that the person Ed Schaefer and the public key said to belong to Ed Schaefer are actually connected? This is solved using certificates. In real life, the first time I contact Amazon, I say "this is Ed Schaefer and this is my public key". Why should Amazon believe me? There's a trusted Certification Authority (CA) who acts like an electronic notary. I can show ID to the CA and so can Amazon. The CA has a public key and digitally signs a message like "the public key n_E, e_E belongs to Ed Schaefer who works at SCU". That's my certificate. The CA also signs a message like "the public key n_A, e_A belongs to Amazon.com". That's Amazon's certificate. When I first contact Amazon, Amazon uses the CA's public key to verify the CA's signature on my certificate and I do the same for Amazon's certificate. Now, since I trust the CA, I feel confident that the Amazon is really the owner of Amazon's public key and Amazon believes that the real owner of Ed Schaefer's public key is Ed Schaefer.

Of course it's impractical for everyone to visit the CA with ID. So the CA can certify sub-CA's. For example, someone at SCU could go to the CA and show her ID and get her certificate. Then people at SCU could show ID to SCU's sub-CA. Then my certificate would be signed by the sub-CA. If I contact Amazon, then Amazon will check SCU's sub-CA's signature on my certificate. Then Amazon will check the CA's signature on the sub-CA's certificate. Similarly, Amazon might be under a sub-CA. In practice, there are different levels of certification. For a serious one, you really do show ID. You can get a less serious certificate that basically says "the e-mail address `eschaefer@hotmail.com` and the public key n_E, e_E are connected". This less serious one does not certify that the e-mail address is provably connected to person Ed Schaefer.

The world's largest and most important CA is the company Verisign which is located in Mountain View.

Secure Sockets Layer

So how does cryptography actually happen on the web? The process is called the Secure Sockets Layer (SSL). When you see `https:`, the *s* tells use that SSL is being used. There are several different protocols for doing it. The following is a simplification of how Netscape does it.

1. When Bob first connects to Amazon, Bob sends Amazon his certificate, which includes his RSA public key $n_{\text{Bob}}, e_{\text{Bob}}$. The certificate links Bob's name with his public keys.
2. Amazon sends Bob its certificate. If there is no certificate or there is a problem with the certificate, Bob's browser will notify Bob.
3. Amazon uses Bob's RSA public key to encrypt an AES key and a key for a MAC. So Amazon concatenates the two keys and turns it into a number. That's easy, it could just be the bit string considered as a base 2 number. Then Amazon raises that number to the power e_{Bob} modulo n_{Bob} . Amazon sends this to Bob.

4. Bob decrypts the message by raising it to his d_{Bob} modulo n_{Bob} and gets the AES key and the MAC key.
5. Bob then encrypts a message for Amazon. This message would include his order, his credit card number, his address, etc. This message would be encrypted with AES using the AES key sent by Amazon. It is this encrypted message that Bob sends to Amazon.
6. Bob finds the MAC of the plaintext message he just encrypted using the MAC key sent by Amazon. Bob uses RSA to sign this MAC. Bob then encrypts the signed MAC with AES.
7. Amazon then receives the message from 5 and decrypts it using AES and gets the plaintext message.
8. Amazon then finds the MAC of the plaintext message using the MAC key.
9. Amazon then decrypts the encrypted, signed MAC sent by Bob using AES. Amazon then has the signed MAC.
10. Amazon then verifies signature by raising the signed MAC to Bob's e_{Bob} mod n_{Bob} . Now Amazon has the MAC that Bob computed.
11. Amazon then checks that the MAC it got in 10 is equal to the MAC it computed in 8. The fact that they are equal ensures i) that the message was not tampered with (because of the MAC) and ii) that it really came from Bob (because of the signature).

Notes. Instead of using RSA, finite field Diffie Hellman and elliptic curve Diffie Hellman (now the most common of the three) could be used. For the Diffie Hellmans, Bob and Amazon would need to trade Diffie Hellman public keys at the beginning. Instead of AES, many protocols are still using triple-DES. If Bob gets cheap after he sends the order, he can not deny that he sent it. Such a denial is called repudiation. Only Bob could have signed the MAC and anyone can verify that it was Bob, because the signature only works using Bob's public key (so only Bob could have made it using his private key).

Actually, Bob could deny it if he publishes his private key. Then he could say that anyone could have faked the signature. On the other hand, if Amazon can prove that he published his private key after the order, or if Bob does not want to publish his private key, then he can't repudiate the order.

End SSL handout

Homework

Problems beginning CW are primarily computer work. On homework problems not involving the computer, show enough work so that I can see what you're doing. Obviously you won't show everything you did on your calculator on longer, repetitive problems. On homework involving the computer, there will often be little work to show.

NT-1. Find $\gcd(720, 450)$ i) using the Euclidean algorithm, ii) by factoring each.

NT-2. For each of the following pairs of numbers, find the gcd using the Euclidean algorithm and then write the gcd as an integer linear combination of the pair: i) 21, 30, ii) 126, 129.

Hist-1. Encrypt the following message. Playfair cipher system, key SUBHARMONIC, plaintext: CHRISTIANS

Hist-2. Decrypt the following message. Playfair cipher system, key FACETIOUSLY, ciphertext: HQSMLFTO

Hist-3. Decrypt the following message. ADFGVX ciphersystem, key CREAMY, ciphertext: AFDFVXFAXXDVGXXGXVGXXFG

Hist-4. You are an ancient Spartan and you intercept a thin strip of paper with the following letters. Decrypt the message. There is a file at my website called greek.txt with this text in it.

ATAIWSRSTSIPTSI LAHWNETHLINRHGROHDNDOERRSEBEJWNOONSUAESACDAEL
FRINKARNLAKTASNEDTRSGNIDTSHIOAGTCHTANUSLSAEHTTTTPEWSSEGOAIRIT
MHUTFNOTSAOAHIGNHHLRRESSAHILNDWHHJISEOSAAOUSBRFHTNRTTAF AIEDE

In real life it would look like $\begin{array}{|c} A \\ T \\ \vdots \end{array}$

NT: 3 - 6: if you're working Mod m or in $\mathbf{Z}/m\mathbf{Z}$ all answers should be between 0 and $m - 1$.

NT-3. Find the multiplicative inverses to all elements in $\mathbf{Z}/13\mathbf{Z}^*$. Your answer should look like: $1^{-1} = 1, 2^{-1} = \dots$

NT-4. Use the Euclidean algorithm to find 35^{-1} in $\mathbf{Z}/73\mathbf{Z}$. What's 35^{-2} in $\mathbf{Z}/73\mathbf{Z}$? (We could say these are $35^{-1} \pmod{73}$ and $35^{-2} \pmod{73}$.) Your answers should be between 0 and 72.

NT-5. Make an addition table for $\mathbf{Z}/6\mathbf{Z}$. Make a multiplication table for $\mathbf{Z}/6\mathbf{Z}$.

NT-6. Make a multiplication table for $\mathbf{Z}/8\mathbf{Z}^*$. Make a multiplication table for $\mathbf{Z}/10\mathbf{Z}^*$.

NT-7. Show that the sum of the squares of 2 odd integers is not the square of an integer. Hint: work Mod 4.

NT-8. Show $X^3 + X + 1$ is always odd if X is an integer.

NT-9. Consider the 10 functions $f_n(x) = nx$ from $\mathbf{Z}/12\mathbf{Z}$ to $\mathbf{Z}/12\mathbf{Z}$ where $n = 2, 3, 4, 5, 6, 7, 8, 9, 10, 11$ (consider them one at a time, the first is $f_2(x) = 2x$). How many elements are in the

range (the range is the set of outputs) of f_n for these 9 n 's? Come up with a formula predicting these values based on n and 12. So for example, the elements of $\mathbf{Z}/12\mathbf{Z}$ are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11. The function $2x$ sends them to 0, 2, 4, 6, 8, 10, 0, 2, 4, 6, 8, 10, so the size of the range is 6. The function $3x$ sends them to 0, 3, 6, 9, 0, 3, 6, 9, 0, 3, 6, 9 so the size of the range is 4. Do this up to 11.

NT-10. Find i) $\varphi(32)$, ii) $\varphi(100)$, iii) $\varphi(3600)$, iv) $\varphi(35)$, v) $\varphi(77)$.

NT-11. If $n = pq$ where p and q are different primes, find a formula for $\varphi(n)$ in terms of p and q .

NT-12. This is an incredibly difficult problem: Solve the MU problem. All strings allowed are made up of the letters M,I,U. You start with the string MI. You can apply any of the following 4 rules to change your string. You can use the rules repeatedly, and in any order you like. 1) If last letter of string is I can add a U at the end (so MI can become MIU). 2) Suppose you have Mx (where x is any string of M,I,U's), can change to Mxx. (from MIU can get MIUIU, from MUM can get MUMUM). 3) If III occurs in string can replace with U (from UMIIIMU can get UMUMU, but can't change IIMII with this rule). 4) If UU occurs in a string you can drop it. Puzzle: can you get the string MU? Don't spend more than 30 minutes on this.

NT-13. Solve the following (if possible): i) $3x \equiv 2(\text{Mod } 14)$, ii) $3x \equiv 2(\text{Mod } 15)$, iii) $3x \equiv 6(\text{Mod } 15)$, iv) $37x \equiv 51(\text{Mod } 100)$.

NT-14. For which positive integers m is the following statement true? $27 \equiv 5(\text{Mod } m)$?

NT-15. Reduce $3^{1,000,000}(\text{Mod } 7)$.

NT-16. Find all positive integers n with $\varphi(n) \leq 12$. You need not prove your result.

NT-17. For which positive integers n is $\varphi(3n) = 3\varphi(n)$? No need to explain, just find the pattern. I want you to describe the set of ALL positive integers with this property; don't just give me a few examples.

SmC-1. Use frequency analysis to decrypt the following. I used a simple shift transformation: $C \equiv P + b(\text{Mod } 26)$. I have inserted spaces and some punctuation. Z CVRIEVU KYV WFCFCNZEX AFBV KYV CRJK KZDV Z NFIBVU R IVRC AFS. NYRK UZU KYV WZJY JRP NYVE YV YZK YZJ YVRU? URD

SmC-2. Encrypt TRANSFER FUNDS using $C \equiv 7P + 11(\text{Mod } 26)$.

SmC-3. You intercept NGIPNGZO YGWB DQIQP GE. You know it was encrypted with $C \equiv aP + b(\text{Mod } 26)$ (so you'll use $P \equiv a'C + b'(\text{Mod } 26)$ to decrypt). You know this sender ends every message with the plaintext OK. i) Find the decrypting keys a' and b' . ii) Decrypt the message. iii) Find the encrypting keys a and b and encrypt COME SOON OK.

SmC-4. If I used a 38 symbol alphabet (the letters, the digits, a period and a blank space), how many different key-pairs (a, b) would there be for affine encryption: $C \equiv aP + b(\text{Mod } 38)$?

SmC-5. Let's use a 27-letter alphabet consisting of the standard 26 letters and a blank space, which we'll denote $-$. We encode the letters in the usual way with $A = 0, \dots, Z = 25$, and $- = 26$. Then we encode digraphs as in class (though for a 27-letter alphabet). You

intercept some ciphertext encrypted with $C \equiv aP + b \pmod{729}$. From frequency analysis, you determine that the ciphertexts corresponding to the two most frequent digraphs in English (namely $S-$ and $-T$) are NG and KX, respectively. Decrypt this piece of the ciphertext: TCUGARXKOK. Don't write anything up for the following three sentences. Observe the last two plaintext and ciphertext digraphs. Think about whether that will always happen by thinking simply Mod 27. For comparison, note that this didn't happen with the second and fifth digraphs. (Note I found another document that says that the two most common digraphs are $E-$ and $S-$.)

SmC-6. Now we will prove what we observed in SmC-5, only with a 26 letter alphabet. Let's say that we use $C \equiv aP + b \pmod{26^2}$ to encrypt digraphs with $\gcd(a, 26^2) = 1$. We encode digraphs as $(x, y) \mapsto 26x + y$ where $x, y \in \{0, 1, \dots, 25\}$ are the usual numerical encodings of letters. Assume that two digraphs end with the same letter (like SO and TO). Prove that their corresponding ciphertext digraphs will also end in the same letter.

LM-1. Read the computer labs manual. (This homework problem has no writeup).

CW-LM-2. Use LETTERS TO NUMBERS to encode 'all is well'. Write to the file t.txt (This computer work problem has no writeup).

CW-LM-3. Use GP-PARI to compute the following:

Let a be the nextprime above 2^{50} , b be the nextprime above 3^{50} and $m = 11^{27}$.

i) Reduce $a^b \pmod{m}$ (remember to do $\text{Mod}(a, m) \wedge b$)

ii) Find $a^{-1} \pmod{m}$

iii) Confirm a^{-1} is right by doing the following. Lift $a^{-1} \pmod{m}$ to an integer outside of a Mod. Multiply this integer by a , subtract 1 and then divide by m . You'd better get an integer (think about why), what is it?

iv) Find $\gcd(m, 8689142)$

v) Read file t.txt and add the number 1895888327898213960000 to that number and write it to the file u.txt. **REMEMBER: to get out of PARI** you type $\backslash q$.

vi) Use NUMBERS TO LETTERS to decode the number in u.txt

FF-1. 2 is a generator of \mathbf{F}_{13}^* . i) For $i = 1$ to 12 compute 2^i in \mathbf{F}_{13} .

ii) For each element $b (= 2^i)$ of \mathbf{F}_{13}^* , find the smallest positive integer r so that $b^r = 1$ (0 is not positive). Make a chart $i, b = 2^i, r$.

In the chart, i is called the discrete $\log_2(b)$ since $2^i = b$.

iii) Find a formula for r given i .

iv) Multiply $3 \cdot 12$ in \mathbf{F}_{13} . Find $\log_2(3)$, $\log_2(12)$ and $\log_2(3 \cdot 12)$ (those are **DISCRETE logs**, so $\log_2(3) = 4$). Find $9 \cdot 10$ in \mathbf{F}_{13} . Find $\log_2(9)$, $\log_2(10)$ and $\log_2(9 \cdot 10)$. Find $11 \cdot 5$ in \mathbf{F}_{13} . Find $\log_2(11)$, $\log_2(5)$ and $\log_2(11 \cdot 5)$. Recall with usual (non-discrete) logs, that $\log(ab) = \log(a) + \log(b)$. For discrete logs, how does it seem that the log of the product is related to the two other logs? So if $a, b \in \mathbf{F}_{13}^*$, how are $\log_2(a)$, $\log_2(b)$ and $\log_2(a \cdot b)$ related?

FF-2i) 2 is not a generator of \mathbf{F}_{17}^* ; find the smallest one, call it g .

ii) What is the smallest positive power i of g that gives you 2 (i.e. what's $i = \log_g(2)$)? What power r of 2 gives you 1? Is r as predicted by the formula in 31iii? (generalized from $p = 13$ to $p = 17$).

FF-3. Fact: The size of the group \mathbf{F}_p^* is $p - 1$. Fact: An element $x \in \mathbf{F}_p^*$ is a generator if and only if its order in the group is $p - 1$. Problem: Let p be a prime and g generate the multiplicative group \mathbf{F}_p^* . Let $k \in \mathbf{Z}$ with $k \geq 0$. If $\gcd(k, p - 1) > 1$, then prove that g^k is not a generator of \mathbf{F}_p^* .

For problems SC-1 and SC-2, our random bit generator will be the first one I taught you. 83 is prime and 2 generates \mathbf{F}_{83}^* . $2 \cdot 83 + 1 = 167$ is prime and 5 generates \mathbf{F}_{167}^* . Our symmetric/shared/secret key is $k=7$. We have $s_1 \equiv g^k \equiv 5^7 \pmod{167}$. For $i = 1, 2, \dots$ we have $s_{i+1} \equiv s_i^2 \pmod{167}$. For $i = 1, 2, \dots$ we have $k_i \equiv s_i \pmod{2}$ where we consider s_i to have been reduced Mod 167 already. To start you off, $s_1 = 136$, $s_2 = 126$, $s_3 = 11$ and so $k_1 = 0$, $k_2 = 0$, $k_3 = 1$. As a check, you should have $s_{16} = 62$. We will use the same random stream for problems SC-1 and SC-2.

SC-1. Stream cipher: The ciphertext is 0101 0101 1110 1001 (I put in spaces to make it easier to read). Those are the bits $c_1 c_2 \dots c_{16}$. The plaintext will be called $p_1 p_2 \dots p_{16}$. The rule is $p_i = c_i \oplus k_i$ for $i = 1, 2, \dots, 16$. Decrypt and decode using ASCII.

SC-2. Self-synchronizing stream cipher: The ciphertext is 0110 1101 1111 0111 Those are the bits $c_1 c_2 \dots c_{16}$. The plaintext will be called $p_1 p_2 \dots p_{16}$. We will define $p_{-1} = p_0 = 0$ which are not part of the plaintext but show up in the computations. The rule is shown below for $i = 1, 2, \dots, 16$. Decrypt and decode using ASCII.

$$p_i = k_i \oplus c_i \oplus \begin{cases} p_{i-2} & \text{if } p_{i-1} = 0 \\ p_{i-3} & \text{if } p_{i-1} = 1 \end{cases}$$

SC-3. Say p is a prime for which 2 generates \mathbf{F}_p^* and assume $q = 2p + 1$ is also prime. Let g generate \mathbf{F}_q^* . Let $k \in \mathbf{Z}$ with $\gcd(k, 2p) = 1$. Let $s_1 = g^k \in \mathbf{F}_q$ and for $i \geq 1$, $s_{i+1} = s_i^2 \in \mathbf{F}_q$. If $2 \leq i < j \leq p$, prove $s_i \neq s_j$. Prove $s_{p+1} = s_2$.

FF-4. Factor (mod 2) all eight polynomials of the form $x^3 + b_2 x^2 + b_1 x + b_0$ completely, where $b_i \in \{0, 1\}$. For example, $x^3 + x^2 = x^2(x + 1)$, now you do the other 7.

FF-5. Find the first 7 powers of x^2 in $\mathbf{F}_8 = \mathbf{F}_2[x]/(x^3 + x + 1)$. Is x^2 a generator of \mathbf{F}_8^* ?

FF-6. $x^4 + x + 1$ is irreducible over \mathbf{F}_2 (trust me). (a) Find $(x^3 + x + 1) \cdot (x^3 + x^2 + 1)$ in $\mathbf{F}_{16} = \mathbf{F}_2[x]/(x^4 + x + 1)$. (b) Find the first 15 powers of x in \mathbf{F}_{16} . (c) Does x^3 generate \mathbf{F}_{16}^* ?

FF-7. In $\mathbf{F}_{32} = \mathbf{F}_2[x]/(x^5 + x^2 + 1)$ find the inverse of $x^3 + x^2 + 1$.

AES-1. Read section 10 of the AES handout.

AES-2. Verify by hand that $\text{SBOX}(1100) = 1100$. By this I mean first invert $x^3 + x^2$ and turn that into a nibble. Turn that nibble into an element $N(y)$ of $\mathbf{F}_2[y]/(y^4 + 1)$ and find $(y^3 + y^2 + 1)N(y) + (y^3 + 1)$. Turn it back into a nibble; it ought to be 1100.

AES-3. We know that MC^{-1} is multiplying by $c(z)^{-1} = xz + (x^3 + 1)$ in $\mathbf{F}_{16}[z]/(z^2 + 1)$. It would be nicer to have this in a form

$$\begin{array}{|l} b_0b_1b_2b_3 \\ b_4b_5b_6b_7 \end{array} \text{ to } \begin{array}{|l} b_3 + b_5 \qquad b_1 + b_4 + b_7 \\ b_2 + b_4 \qquad b_0 + b_6 + b_7 \end{array}.$$

So multiply $(b_0x^3 + b_1x^2 + b_2x + b_3)z + (b_4x^3 + b_5x^2 + b_6x + b_7)$ with $xz + (x^3 + 1)$ in $\mathbf{F}_{16}[z]/(z^2 + 1)$ by hand to fill in this table. (I gave you half of them as a check.)

AES-4i) Expand the key 1011 1101 0010 0101 using Simplified AES. As a check, the last four bits of the expanded key should be 1100 (that is $k_{44}k_{45}k_{46}k_{47}$).

ii) Find $c(z)^{-1}K_1$. (Hint, that's really $MC^{-1}K_1$).

iii) Decrypt the string 0111 0001 0011 1001 with the key 1011 1101 0010 0101 using Simplified AES. Use $A_{K_0} \circ SR^{-1} \circ NS^{-1} \circ A_{c(z)^{-1}K_1} \circ MC^{-1} \circ SR^{-1} \circ NS^{-1} \circ A_{K_2}$. (Remember that means A_{K_2} is the first step. Just do ECB Mode). Then decode to a pair of ASCII characters (the message will tell you that you got it right). Give the homework grader a break and write this up neatly with little notes about which step you're doing.

As a midway check: After MC^{-1} , you should have 1000 0001 1100 1011.

AES-5. Encrypt the 32-bit message COEN (in ASCII) using Simplified AES and Cipher Block Chaining (CBC) with initialization vector 1000 1101 0000 1011 (IV) and key 1100 1011 1111 1110. Here are some checks. CT1 is 1000 0110 1101 110*. The final ciphertext (CT2) is *1* 1 *0*1 *0*1 *001. (I'm not telling you the bits under the *'s)

AES-6. Decrypt 1100 0010 1011 0011 0010 0101 1011 0011 using the output feedback (OFB) stream cipher. Use the same key as in the last problem, namely 1100 1011 1111 1110. Use the initialization vector 1100 1110 0100 0100 (Look in the last problem and you'll see that this string was the XOR of the first two bytes of plaintext and the initialization vector from that problem. In that last problem, that XOR is what you encrypted. That's the first step of this problem. So that saves a LOT of work. So you need not show any work to get z_1). Decode to ASCII characters.

AES-7. Explain the middle-in-the-middle attack that shows that triple-DES with three different keys is not much safer than triple-DES with two different keys. For simplicity, assume that the former is $CT = E_{K_3}(E_{K_2}(E_{K_1}(PT)))$ and the latter is $CT = E_{K_1}(E_{K_2}(E_{K_1}(PT)))$.

NT-18. Use repeated squares to reduce $17^{53} \pmod{97}$. Do this problem twice. Use the calculator method first. Then do the computer algorithm (though with your calculator).

RSA-1. Do this problem by hand and calculator. Don't use PARI (though you could check work with it). For RSA, your p is 7, your q is 97 and your e is 257. Find $\varphi(n)$ and d and include them in the writeup. We will use RSA to agree on a key for the affine $C \equiv aP + b \pmod{26}$ cipher. I encode the key(s) a and b by computing $26a + b$ (we didn't do that with the key when discussing this affine cipher; we did it with digraphs when working mod 26^2 , which we're NOT doing). I then encrypt the key using your e and n and send you the reduction, namely 146. Decrypt to a number and then determine a and b and

include them in your writeup. I have encrypted a message with $C \equiv aP + b \pmod{26}$ and got PBEXBI. I send you PBEXBI. Decrypt the message and include in your writeup.

The next day, you want to send me a message using $C \equiv aP + b \pmod{26}$. For safety you decide to use a new a and b , namely $a = 11$, $b = 21$. Turn that into a single number, namely $26a + b$ and send me that number (the key) using RSA. My $n, e = 681, 19$.

LM-4. In dan.exe is a symmetric key cryptosystem called *Matrix encryption* that I no longer cover in class, but it is handy as a generic symmetric key cryptosystem that we have a program for. I will use it with public key cryptosystems. **Think of it like AES.** For plaintexts and ciphertexts it has a 37 letter alphabet, A-Z=0-25, 0-9=26-35, space=36. It puts a space at the beginning of each line of ciphertext. It takes a 12 decimal digit key.

For practice, encrypt 'do not go gentle into that good night' using *Matrix encryption*. using the key 113106212619. Write the output to the file v.txt. On your homework write out the ciphertext output from the program.

Go to the program *Matrix decryption* and enter the same key. (With *Matrix encryption* and *Matrix decryption* you always enter the same enciphering key.) Have *Matrix decryption* decrypt v.txt and make sure you get out the same message. You don't turn anything for this paragraph's work, I just want to be sure you're using the software correctly.

For problems CW-RSA 2 - 4, at the top of your homework, write your name, your and your partner's colors.

R.S.A. directory, (n,e), for problems CW-RSA 2 - 4

Blue (bl): $n = 231440235891660906053817934904337175936493389690535863833485$
 $2359112541601646303546661419068500664331$

$e = 72693169641294392054324089725009715723775238103809310693108$
 $2582280108084558760457289206461584448191$

Green (gr): $n = 8982739355658980450954505563786378902219911689145959$
 $45263169996375998665839252877482856394980059807$

$e = 440795955818489272223347981691856786085455386333305489849$
 $776281698474322567978204145020005718349271$

Orange (or): $n = 45970003270264989688908974131921753900627792978445589$
 $494936157700926851903384528223275277440227877$

$e = 596131982616448003103784710980344171151668293418683956$
 $626438045113323689246259881668637506766829519$

Red (re): $n = 1859486070465862481250084737025244499643330450742207640$
 $127531631364580491764319392630687618608632729$

$e = 57463814268791658055979510125582566711041821111604849929$
 $1854170685862245899201436141624525447657277$

Yellow (ye): $n = 3466436991338570577407597275451416873137600057687491154$
 $54693238560026239259246326048573361535111137$

$e = 8760737903908348835948966304524580678073899022132796982046$
 $30864651783688734202322795082344727508911$

There are files with these numbers if you don't want to type them in, namely n.txt and e.txt

CW-RSA-2. Above is your n, e . There are files called xxp.txt where xx is the first two letters of your color, blue has blp.txt, etc. This file contains your p. Read this into Pari. Find your

q and write it on your homework. Now find $\varphi(n)$, using the formula and write it on your homework. Now find your d and write it on your homework. I recommend that you *lift* your d and write it to the file d.txt so you don't have to keep retyping it.

CW-RSA-3. There's a file called xxkey.txt where xx is the first two letters of your color. This is a 12-digit key for *Matrix encryption* I have encrypted using your R.S.A. keys. Decrypt our shared 12-digit key. In the file xxmat.txt there is a message from me which I encrypted using *Matrix encryption* and our shared 12-digit key. Find the message. Put our key and the plaintext message in your homework. Usually R.S.A. is used for key exchange for a faster conventional cryptosystem as in this problem.

CW-RSA-4. Find your partner's color and encrypt the *Matrix encryption* key 208494469883 for him/her using RSA (and his/her RSA keys, of course). Put the ciphertext in your writeup. As practice, your partner should decrypt the ciphertext and be sure s/he gets 208494469883. In this problem, you won't actually use the *Matrix encryption* key to encrypt a message.

Now encrypt 208494469884 for your partner using RSA (include its encryption in your writeup). Notice the difference in ciphertexts from the two very similar plaintexts.

DH-1. Do this problem by hand/calculator. Diffie-Hellman key exchange. Work in \mathbf{F}_{677} , 677 is prime. $g = 2$ is the generator of \mathbf{F}_{677}^* we'll use. Your private key is 13. (a) What's your public key? (b) My public key is 287. Find our shared key. (c) Turn our shared key into a pair of keys (a,b) where our shared key is $a26 + b$ where $0 \leq a, b \leq 25$. You will encrypt for me using $C \equiv aP + b(\text{Mod } 26)$. Encrypt YO (not as a digraph, but as a two letters, i.e. two numbers in [0,25]) for me and decode to a pair of letters. That is the ciphertext pair you would send to me.

For the problems DH 2 - 3, at the top of your homework, put your color and name, and your partner's color

Discrete log directory of public keys for problems DH 2, 3, ElG 2, 3. We are working with $p = \text{nextprime}(10 \wedge 24)$ and $g=5$ (a generator of \mathbf{F}_p^*). This directory is in the file dlkey.txt.

Ed: 483535020765083780845831	Red: 635364114936087527279104
Yellow: 891128110299929524095054	Blue: 258020014459286684891269
Green: 794096565547742926108933	Orange: 977726291542055604902486

Your secret key (your a) is in a file called dlxxkey.txt where xx is the first two letters of your color. Note, that if you compute $\text{Mod}(5,p) \wedge a$, you should get your public key.

CW-DH-2. Diffie Hellman: Find your shared key with your partner. Write down the first 12 digits (and put in writeup). Use this 12 digit key with your partner to pass messages back and forth with Matrix enc/decrypting (don't turn in anything for that).

CW-DH-3. Diffie Hellman: Find your shared key with Ed. Copy down the first 12 digits of it (put in writeup). Now use that key to MATRIX DECRYPT what's in the file dlmatxx.txt where xx is the 2 letter code for your color. Put output in writeup.

Not a homework problem: You have two files (that I didn't tell you about before - heh, heh). They are called renc and rdec. Those stand for Rijndael encryption and Rijndael decryption. They do simplified AES encryption and decryption, respectively. While in Pari,

you can read those in: for example `\r renc`. You won't see much happen in Pari and that's OK. If you read in `renc` then you can type `aesenc([1,0,1,0,1,1,1,0,1,0,0,0,1,0,1,0],[1,1,0,1,1,1,0,0,1,0,0,1,1,1,0,0])`. The first vector is the plaintext; the second vector is the key. The output (on the next line) will be the corresponding ciphertext. Notes: remember, if the plaintext vector is called %1 and the key is called %2 then you could just type `aesenc(%1,%2)`. It is important (for later) that these vectors not look like `[Mod(1,2),...]`. If you have a vector like that, then *lift* it.

If you read in `rdec` then you can type `aesdec([1,1,1,0,0,1,0,0,1,1,0,0,1,0,1,1],[1,1,0,1,1,1,0,0,1,0,0,1,1,1,0,0])` and it will use simplified AES to decrypt the first vector (the ciphertext) with the second vector (the key).

CW-DH-4. Diffie Hellman: We will not work with the public keys in the table for this problem. We will also not break the class up by color. We will instead work in \mathbf{F}_{25} . The polynomial $x^{25} + x^3 + 1$ is irreducible over \mathbf{F}_2 . A generator of \mathbf{F}_{25}^* is x , which in Pari is `Mod(Mod(1,2)*x, x^25 + x^3 + 1)`. We want to use Diffie Hellman to agree on a simplified AES key. My public key is in the file `dhkey.txt`. Your private key is 8675309. Find our shared key. It will be of the form $a_{24}x^{24} + a_{23}x^{23} + \dots + a_1x + a_0$ with $a_i \in \{0, 1\}$. We want to turn that into a 16-bit simplified AES key. It will help to lift it twice. Our simplified AES key will be $a_{24}a_{23} \dots a_9$ (in that order, left to right). In the file `ct1.txt` are two ciphertext vectors. Decrypt using simplified AES and decode (ASCII) into a four-character message. Put the 16-bit simplified AES key and the plaintext in your writeup.

CW-RSA-5. There's a file you downloaded called `xxb.txt` where `xx` is the first two letters of your color. This is a message I encrypted for you using R.S.A. and your keys n, e , given earlier. Read this into Pari, decrypt, lift, and write to the file `msgrsa.txt` Use NUMBERS TO LETTERS to decode and write the plaintext in your homework.

CW-RSA-6. Find your partner's color and pick the one out of the five following messages that includes his or her color. Blue moon rising, the corn is green, ripe juicy oranges, the lady in red, yellow rose of texas. First use LETTERS TO NUMBERS to encode this message, write to the file `msg.txt` (it should have 2 lines). Go into Pari and use R.S.A. to encrypt each line with your partner's n and e . Write on your homework the numbers and your partner's color.

ElG-1. Do by hand/calculator. ElGamal message exchange. You and I have the same keys as in problem DH-1. You want to send the message PK (abbrev. for public key cryptography) to me. Encode this digraph as an integer in $[0,675]$. You choose a random $k = 11$. (a) What pair of numbers will you send? (b) I send you the pair (310,407). Decrypt and decode.

CW-ElG-2. ElGamal message exchange: In the file `elgxx.txt`, where `xx` is your color, is a pair of numbers (g^k, Mg^{ak}) where $g=5$, a is your secret key (in `dlxxkey.txt`), M is my plaintext message and k is a random number I picked (you'll never need or know k). Find the number M , lift and use NUMBERS TO LETTERS to find the plaintext message. Put output in writeup. Pari is flexible with division with Mod: `Mod(10,11)/3` would give you `Mod(7,11)` also `Mod(10,11)/Mod(3,11)` would give you `Mod(7,11)`. Recall for this and CW-ELG-3 we are using the discrete log directory given earlier.

CW-ElG-3. Now practice with the ElGamal message exchange with each other. I suggest that you keep your messages short. Nothing to turn in here.

CW-ElG-4. ElGamal again, but now we work in $\mathbf{F}_{2^{25}}$. The polynomial $x^{25} + x^3 + 1$ is irreducible over \mathbf{F}_2 . A generator of $\mathbf{F}_{2^{25}}^*$ is $\text{Mod}(\text{Mod}(1, 2) * x, x \wedge 25 + x \wedge 3 + 1)$. I will use ElGamal to send you a simplified AES key. Your secret key is $a=321321$. I send you a pair (g^k, Mg^{ak}) , which is in the file `elgtwo.txt`. Find M and then lift it once and then lift it again to get a polynomial $x^{24} + \dots$. That polynomial is $a_{24}x^{24} + \dots a_1x + a_0$ where $a_i \in \{0, 1\}$. Again, to get the simplified AES key, take the first 16 bits, namely $a_{24}a_{23}a_{22} \dots a_9$. Decrypt the message in `ct2.txt` with this key (it has four ASCII characters). Put the key and the plaintext message in your writeup.

For computer problem MO-1, when I ask you to put a long number in the writeup, you need only write the first 4 digits.

CW-MO-1. Massey-Omura Cryptosystem. This must be done with a partner. Again we use $p=\text{nextprime}(10 \wedge 24)$. The order of the colors is Blue, Green, Yellow, Orange, Red. The partner whose color appears first in that list is the first partner. Both partners will find their randomly chosen encrypting key for this session in the file `moxxkey.txt` where `xx` is the first two letters of your color. Find your decrypting key (remember it is the inverse $\text{Mod } p-1$) and put this in your writeup. The first partner should encode SANTA CLARA using LETTERS TO NUMBERS. Read it into Pari and encrypt this message. Both should put the encrypted number in your writeups. Get this message to your partner. Now the second partner should encrypt that. Put that in both writeups. Now the first partner should start decrypting that. Put that number in both writeups. Now the second partner should decrypt and decode and hope that you got SANTA CLARA. On the top of your homework include your name, your color and your partner's color.

Try the same exercise (with a different message, if you like) starting with the second partner. Nothing to turn in here.

In reality, the message you'd exchange would probably be an AES key.

MO-2. Alice and Bob use Massey-Omura. Alice is the initial sender. She sends message M_1 to Bob. Alice uses e_A and d_A . The next day, Alice uses Massey-Omura with Eve and she (Alice) sends a different message M_2 to Eve and uses the **same** e_A and d_A . How can Eve trick Alice and determine the message from the Alice-Bob exchange? (It's OK if Eve doesn't ever determine the message that Alice is trying to send her.) Alice should not receive anything from Eve that she (Alice) has seen before. This shows why Alice should use a different e_A and d_A pair each time. Assume that all three users use the same finite field.

For the following problems, write your color and your partner's color at the top. Do EC-1 and EC-2 by hand and with a calculator.

EC-1. Add the point $[-2,3]$ to the point $[2,-5]$ on the elliptic curve $y^2 = x^3 + 17$ using line intersections. Now do the same problem using the addition formulas.

EC-2. Double the point $[-2,3]$ on the same elliptic curve using line intersections.

Working with elliptic curves in PARI. Represent the elliptic curve $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$ as a vector $[a_1, a_2, a_3, a_4, a_6]$. You can give the elliptic curve a name like $e=[0,0,0,0,1]$. Represent the point $(0,1)$ on the curve by the vector $[0,1]$. You can name points $q=[0,1]$, $r=[2,-3]$. To find $q+r$ you compute **elladd(e,q,r)**. To find $q-r$ you compute **ellsub(e,q,r)**. To find $7q$ (q added to itself 7 times) you compute **ellpow(e,q,7)**. PARI calls the 0 point $[0]$.

CW-EC-3. Use the elliptic curve $y^2 = x^3 + 17$. Let Q be the point [-2,3] and R be the point [2,5]. Verify in your head that both points are on the curve. Find 2Q (I would first type $e=[0,0,0,0,17]$ and $q=[-2,3]$ and then at this point write $q2=ellpow(e,q,2)$ so I can use 2Q in the future). Then find Q+R, 3Q, 4Q, 2R, Q-R, 2Q-R, 3Q-R, 4Q-R, 2Q-2R. (There are old versions of PARI on campus that want: $ellpow(e,2,q)$. If in doubt, after ? type ?ellpow).

EC-4. Find all points of $y^2 = x^3 - 4$ over \mathbf{F}_2 , \mathbf{F}_3 and \mathbf{F}_5 (don't forget the 0-point). Do by hand.

In PARI: let's say $e=[a1,a2,a3,a4,a6]$. You can work over the finite field \mathbf{F}_p by saying $e=Mod(1,p)*e$. Now e is the same elliptic curve over the finite field \mathbf{F}_p . Once you've done that, you don't need to make the points Mod p.

CW-EC-5. Now we'll work with the elliptic curve $y^2 + y = x^3 - x$. Let Q be the point [0,0]. We will work over \mathbf{F}_7 (so $p = 7$). Find 2Q, 3Q, ... until you get the 0-point. Which multiple of Q is 0? (In the writeup, call a point [a,b] instead of [Mod(a,7),Mod(b,7)]).

For EC 6, 7, we will be working with elliptic curve discrete log cryptosystems. The elliptic curve is

$ee=[0,0,0,0,-4]$ ($y^2 = x^3 - 4$). The point we'll work with is $g=[2,2]$, the finite field we'll work over is \mathbf{F}_p where $p=\text{nextprime}(10\wedge 25)$ (not 24). Let $e=Mod(ee,p)$, now the elliptic curve is defined over the finite field. My public key point on this elliptic curve is [166985550562940165155251, 1303821074460599731155518], it is the file eckey.txt. Your private key is $a=\text{nextprime}(10\wedge 22)$, so your public key is $ag=ellpow(e,g,a)$ (which you don't need to compute).

CW-EC-6. Diffie-Hellman key exchange. Find our shared key. Our shared key is a point on the elliptic curve. Find the first 12 digits of the x-coordinate and write them down. In the file matec.txt is a message that I encrypted with *matrix encryption* for you with this 12 digit key. Put the key and plaintext in your writeup.

CW-EC-7. ElGamal message exchange part 1. For this problem, I am sending a message to you. So I send you two points, the first one is kg where k is a random number I found and g is the point [2,2]. That point is [7386457279431210226870299, 9511859968639266447209535], It is in the file kg.txt. The second point I send you is $Q+kag$ where Q is my plaintext point, and a is your secret key. The second point is [3757051374663529725534547, 4304699078475562946238569], it is also in the file kg.txt. Find the point Q . Carefully retype the x-coordinate without its last digit (omit that last 0) and write it to a file. Now use NUMBERS TO LETTERS to read the message. OR instead of retyping, you could try $Q[1]$. That pulls out the first entry of the vector. Then lift and divide by 10 and write to a file and use NUMBERS TO LETTERS. Put plaintext in your writeup.

CW-EC-8. El Gamal message exchange part 2. For this problem, you will send me the message *switzerland*. First use LETTERS TO NUMBERS and write it to a file. Now read that file into Pari, I will call that number m . If starting a new session you need to tell PARI that $p=\text{nextprime}(10\wedge 25)$, $ee=[0,0,0,0,-4]$, $e=Mod(ee,p)$, $g=[2,2]$. Now compute $n=Mod(10*m,p)$ (you have multiplied m by 10 and started working Mod p). Let $w=x^3-4$ (since the elliptic curve is $y^2 = x^3 - 4$). Now compute $c=\text{subst}(w,x,n)$ (this plugs the number n into $x^3 - 4 \text{ Mod } p$). (I wrote the on-line help for subst, Frenchmen reworded

it :) Now compute $\text{sqrt}(c)$ (this tests to see if there is a point on the elliptic curve with x -coordinate n). If you don't get a sqrt , then compute $n1=n+1$. Do all the same steps with $n1$ instead of n (starting with $c=\text{subst}(w,x,n1)$). If this doesn't work, go to $n2=n1+1$ (hint, you don't need very many n_i 's). Once you get a $\text{sqrt}(c)$, let $y=\text{sqrt}(c)$. Create a point $q=[n_i,y]$ (both n_i and y are Mod expressions). This is a point on the curve. Hope you see why. Now you have encoded your plaintext as a point q on the elliptic curve. You choose a random $k=\text{nextprime}(10\wedge 20)$. Now compute kg . Put that in your writeup (4 digits each is O.K.). Now compute $q+kag$ (remember that's my ag , which is in `eckey.txt`). Put that in your writeup. Those are the 2 points you would send me. As a check, the last point is $(2\dots, 9\dots)$.

CW-EC-9. This is the only realistic elliptic curve cryptography homework problem. We will do ECDH over the finite field $\mathbf{F}_{2^{16}}$. The only unrealistic part is the 16 (which should be a prime at least 163). Let $f = x^{16} + x^6 + x^2 + x + 1$. We represent $\mathbf{F}_{2^{16}}$ as $\mathbf{F}_2[x]/(f)$. The elliptic curve is $y^2 + xy = x^3 + 1$ or in Pari: $e = [1, 0, 0, 0, 1] * \text{Mod}(\text{Mod}(1, 2), f)$. My public key is the point $[x^{11} + x^8 + x^4 + x^3 + x, x^{13} + x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + 1]$. The file `ecdhkey.txt` has two lines. The first is e , the second is my public key point. Your private key is 31415. Find our shared Diffie Hellman Key. Lift its x -coordinate twice to get a 16 bit simplified AES key. The x -coordinate is $a_{15}x^{15} + a_{14}x^{14} + \dots + a_1x + a_0$. Let the simplified AES key be $a_{15}a_{14} \dots a_1a_0$. Use it to decrypt the message in `ecaes.txt`.

CW-MAC-1. MAC practice problem. The plaintext is *memory*.

The MAC key is $[0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1]$. That key is in a file you downloaded called `mackey.txt` Find the MAC using simplified AES.

As a check: the answer is 1011*****0110.

RSA-7. This will give you practice at using RSA for signatures. The two parties involved are Dr. Schaefer, whose enciphering keys are $n_S = 65, e_S = 11$ and Dr. Appleby $n_A = 77, e_A = 13$. Dr. Appleby has an ID number known to everyone and it is 32; Dr. Schaefer's is 30. First S wants to send a msg notifying A that he's about to send a signed message, so he sends his ID number (30), but encrypted, with A's keys, to A. It is simply encrypted with no digital signature. Without this notification, A would have no idea who the upcoming message was from. a) How will this message look to the enemy (Eve, perhaps), i.e. what number will S send? (In your writeup of a), show your work computing d_S and d_A). b) Now A knows who the upcoming message is allegedly from; notice the enemy does not, not knowing A's deciphering key. S now wants to send the message 21 to A; he will sign it and encrypt it for A. This way the enemy can't read it and A can be sure it came from S. Pay attention to the sizes of n_S and n_A when you decide whether to sign first or encrypt first. How will this message look to the enemy. (Back of the book: The answer is '3', but I want you to show work getting it, of course). c) Now have A decrypt both messages (and with the 2nd, confirm it's from S). Do all work for a)-c) by hand and with a calculator, show your work in the writeup. This will give you practice with repeated squares, finding inverses, etc. for the midterm. You may check your work with PARI. You may use PARI (no need for calculator or much writeup) for d)-f) or do them with a calculator if you hate computers.

Now we'll have A return a message to S. First A simply encrypts his ID number 32 for S, so S will know a signed message from A is about to arrive. d) what number will A send? Now A wants to send S the message 15. He will sign it so S knows it's really from A and

he'll encrypt it so the enemy can't read the message. Pay attention to the sizes of n_S and n_A when you decide which to do first. e) what number will A send? f) now have S decrypt both messages and confirm the second is from A.

For ELG 5, at the top of your homework, write your name, your and your partner's colors. CW-ELG-5) ElGamal signature. In this problem both partners should send and receive signatures. Use the discrete log directory from right before problem CW-DH-2: $p = \text{nextprime}(10^{24})$ and $g=5$ and a =your secret key in dlxxkey.txt. Encode ITSME022800 using LETTERS TO NUMBERS. That's your signature S . (This problem is not very realistic. Usually you sign a MAC.) Use the number in moxxkey.txt as your random k . Compute $r = g^k \text{ Mod } p$ and lift. Find your $x = k^{-1}(s - ar) \text{ Mod } p-1$ and lift. Put r, x and S in your writeup. Send your partner (r,x,S) .

ii) Receive (r,x,S) from your partner. The signature receiver only works Mod p . Compute r^x, g^{ar} (where g^a is your partner's public key; it is in dlkey.txt and the directory in earlier homework), and $g^{ar}r^x$. Then compute g^S . It should be the same as $g^{ar}r^x$. Include r^x and g^{ar} in your writeup.

ELG-6 Let's say that on Monday and Tuesday, Alice uses ElGamal signatures to send two different signatures, S_1 and S_2 , to Bob but uses the same k both days. Explain how Eve can determine a_A (Alice's private key). This explains why Alice should use a different k each time she signs something.

CW-MAC-2. We want to sign the MAC from problem CW-MAC-1 using RSA. The MAC above can be considered to be a base two number. Turn it into a base 10 number $P = 1 \cdot 2^{15} + 0 \cdot 2^{14} + 1 \cdot 2^{13} + 1 \cdot 2^{12} + \dots + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0$. Sign that using your private key from problem CW-RSA-2. Put the signature in your writeup. Have your partner verify the signature using your public key.

CW-MAC-3. We want to sign the MAC from problem CW-MAC-1 using ElGamal signatures. Take the number P that you determined in problem CW-MAC-2. Use the private key and k from problem CW-ELG-5). Put the signature S, r, x in your writeup. Have your partner verify the signature. (Note $P=S$ in this problem.)

SSL-1. Nothing to turn in on this one. Go through the Netscape SSL protocol with your partner. Choose an Amazon and a Bob. Have Bob find a message that you will encrypt using simplified AES. Use simplified AES for the MAC. Use RSA or Diffie Hellman initially to agree on the AES and MAC keys. Use ElGamal signature instead of DSS. Have both parties do their parts of the whole process. If you're really gung ho, do it again switching roles (possibly with a new message). Don't bother with certificates.