

Applied Cryptography CSCI 181, 76287. Tuesday/Thursday 11:50 - 1:35. O'Connor 102.

Professor Ed Schaefer, 309 OConnor Hall, 554-6899, eschaefer@scu.edu Office hours: Tuesday 9:20 - 10:20, 2 - 3. Wednesday 8:30 - 9:00. Thursday 11 - 11:30, 3 - 4.

Midterm Thursday February 16. Your decision about what your project will be is due Thursday January 19.

A one page progress report on your project is due on Tuesday February 14.

On Thursday, March 1, you must turn in the equivalent of two message's worth of plaintext, though no write-up is necessary.

Your project is due at the beginning of class on Thursday, March 8. If you miss the deadline, then you can turn the project in any time up to March 15 for a penalty of one grade (A→ B, B+→ C+, etc).

Final Thursday March 22 at 1:30.

Your grade is broken up as: homework 25%, project 25%, midterm 20%, final 30%.

In this class on Applied Cryptography, we will go deeper into cryptography: how it is done and what it is used for. We will also make use of your programming skills. This course will familiarize you with important cryptographic applications that you did not learn in the first course. We will learn some basic cryptanalytic analytic techniques which explain how key sizes are chosen. We will learn running time analysis of algorithms so that we understand better how fast encryption, decryption and cryptanalysis are. We will implement RC4 and MD5 - a popular stream cipher and hash algorithm. We will study how groups of people use cryptography with key management, Kerberos and PGP. We will learn how quantum cryptography works. We will learn new applications of cryptography like time-stamping, secret sharing, and digital cash and elections.

Project: You may do the project alone or in pairs. I expect more plaintext from pairs for an equivalent grade. For people doing the first project: spend more time thinking and less time programming and Googling for the first two thirds of your project time. People doing the second project found it good to start programming immediately.

I am aware that science students often dont like writing. However, I take writing very seriously. It is the primary way in which we, as scientists, communicate our results. Important results have been ignored because of bad writing. Make sure you devote sufficient time before your project write-up is due to write up your project clearly. If you get great results at the last minute and have almost no time to write them up well, I will not be happy.

Here are the two projects you can choose from:

1. Recall the stream cipher where the keystream is the repetition of keyword. You will be given ciphertext (i.e. the keystream XORed with the plaintext) and will find the plaintext.
2. In WWII, the Germans sometimes used the same keystream twice for two messages. If you XOR those together, the keystream drops out and you get two plaintexts XORed together. Peter Hilton, who was interviewed in the documentary we saw last term, used to determine those two plaintexts. You do it too.

End syllabus

## 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**

# Classical cryptanalysis handout

## In-class monoalphabetic substitution cipher

GU P IPY AKJW YKN CJJH HPOJ RGNE EGW OKIHPYGKYW HJVEPHW GN GW  
DJOPZWJ EJ EJPVW P AGUUVJYJN AVZIIJV MJN EGI WNJH NK NEJ IZWGO  
REGOE EJ EJPVW EKRJSJV IJPWZVJA KV UPV PRPB

## Vigenère examples

### Keyword TIN

ABCDEFGHIJKLMNOPQRSTUVWXYZ  
TUVWXYZABCDEFGHIJKLMNOPQRS  
IJKLMNOPQRSTUVWXYZABCDEFGHI  
NOPQRSTUVWXYZABCDEFGHIJKLM

## Vigenère square

ABCDEFGHIJKLMNOPQRSTUVWXYZ  
BCDEFGHIJKLMNOPQRSTUVWXYZA  
CDEFGHIJKLMNOPQRSTUVWXYZAB  
DEFGHIJKLMNOPQRSTUVWXYZABC  
EFGHIJKLMNOPQRSTUVWXYZABCD  
FGHIJKLMNOPQRSTUVWXYZABCDE  
GHIJKLMNOPQRSTUVWXYZABCDEF  
HIJKLMNOPQRSTUVWXYZABCDEFG  
IJKLMNOPQRSTUVWXYZABCDEFGH  
JKLMNOPQRSTUVWXYZABCDEFGHI  
KLMNOPQRSTUVWXYZABCDEFGHIJ  
LMNOPQRSTUVWXYZABCDEFGHIJK  
MNOPQRSTUVWXYZABCDEFGHIJKL  
NOPQRSTUVWXYZABCDEFGHIJKLM  
OPQRSTUVWXYZABCDEFGHIJKLMN  
PQRSTUVWXYZABCDEFGHIJKLMNO  
QRSTUVWXYZABCDEFGHIJKLMNOP  
RSTUVWXYZABCDEFGHIJKLMNOPQ  
STUVWXYZABCDEFGHIJKLMNOPQR  
TUVWXYZABCDEFGHIJKLMNOPQRS  
UVWXYZABCDEFGHIJKLMNOPQRST  
VWXYZABCDEFGHIJKLMNOPQRSTU  
WXYZABCDEFGHIJKLMNOPQRSTUV  
XYZABCDEFGHIJKLMNOPQRSTUVW  
YZABCDEFGHIJKLMNOPQRSTUVWX  
ZABCDEFGHIJKLMNOPQRSTUVWXY

## In-class Vigenère ciphertext

wzggqbuawq pvhveirrbv nysttaknke nxosavvwfw frvxqumhuw wqgwtgziih  
locgpnhjmn nmtzqboavv abcuawohbv rjtampovkl gpigfsmfmw vnniyhzyrv  
qkkiqywweh vjrjwgwegw zhcxucakep wpsnjhvama hkmehnhuww vtzguwac lz  
stsvfxlplz muywzygagk aofkioblwi argtvrgzit xeofswcrqb tllcmiabfk  
ttbwbfenvz snlytxahuw vgtzstghut vrzwrzglpr ariltwxwta mpotgvlqh  
vkhkynwmpm vmwgbjxqnb tnuxhkwas gvbwbntswm pwfdmhnxe zinbdsqarv  
aihojmneqo alfwmpomqd qgmkuwvfgf husrfaqggg vavwzyahgg wbrgjbbake  
axkgovnkww kdwiwhdnbo aumggbgbm v exaogypwe wgzvgymfrf gglbcuaq

## Histograms

The first line gives the frequency in which A, B, ..., Z appears among the 1st, 8th, 15th, 22nd, 29th, etc. letters in the ciphertext.

[10, 0, 0, 1, 1, 3, 7, 0, 0, 5, 7, 3, 2, 2, 0, 0, 1, 0, 4, 1, 2, 3, 10, 0, 1, 6]

[2, 0, 0, 1, 3, 1, 2, 6, 6, 1, 1, 2, 7, 0, 0, 2, 2, 5, 2, 2, 0, 9, 8, 4, 2, 1]

[3, 0, 5, 1, 3, 3, 11, 2, 2, 1, 3, 0, 0, 5, 2, 2, 5, 1, 0, 5, 2, 6, 5, 0, 1, 0]

[4, 4, 0, 0, 1, 2, 7, 6, 0, 0, 4, 4, 7, 2, 0, 2, 0, 2, 0, 6, 1, 2, 3, 7, 3, 1]

[5, 8, 0, 2, 3, 0, 1, 0, 4, 1, 0, 4, 4, 3, 2, 5, 6, 0, 0, 1, 1, 4, 9, 0, 0, 5]

[3, 5, 4, 0, 0, 2, 5, 7, 0, 1, 3, 0, 1, 0, 9, 2, 1, 2, 6, 2, 0, 4, 8, 0, 1, 2]

[4, 4, 1, 0, 2, 4, 5, 1, 2, 0, 0, 1, 1, 9, 3, 1, 1, 7, 1, 4, 9, 3, 0, 1, 3, 1]

End of Classical cryptanalysis handout

## 0.1 The MD5 hash algorithm

One of the most popular hash algorithms at the moment is MD5 (SHA1 is another - this stands for *Secure Hash Algorithm*). It is more efficient than the hash algorithm described last quarter using AES. MD5 is based on the following hash function  $f$ . The function  $f$  maps 512 bit strings to 128 bit strings. Let  $X$  be a 512 bit string. For MD5 we will call a 32 bit string a word. So  $X$  consists of 16 words. Let  $X[0]$  be the first word,  $X[1]$  be the next,  $\dots$ ,  $X[15]$  be the last word.

### Initial buffers

While evaluating  $f$  we continue to update four buffers:  $A$ ,  $B$ ,  $C$ ,  $D$ ; each contains one word. There are four constants  $\text{Const}_A = 0x67452301$ ,  $\text{Const}_B = 0xefcdab89$ ,  $\text{Const}_C = 0x98badcfe$ ,  $\text{Const}_D = 0x10325476$ . The notation  $0x$  indicates that what comes after is a hexadecimal representation which uses the symbols 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f to represent the nibbles 0000, 0001,  $\dots$ , 1111. So  $\text{Const}_A = 01100111010001010010001100000001$ . Initially we let  $A = \text{Const}_A$ ,  $B = \text{Const}_B$ ,  $C = \text{Const}_C$ ,  $D = \text{Const}_D$ . For clarity during the hash function, when we update,  $A, B, C$  or  $D$  we will give it an index. So initial  $A_0 = \text{Const}_A$ ,  $B_0 = \text{Const}_B$ ,  $C_0 = \text{Const}_C$ ,  $D_0 = \text{Const}_D$ .

### The four functions

Let us define four functions. Each takes 3 words as inputs.

$$F(X, Y, Z) = XY \vee \bar{X}Z,$$

$$G(X, Y, Z) = XZ \vee Y\bar{Z},$$

$$H(X, Y, Z) = X \oplus Y \oplus Z,$$

$$I(X, Y, Z) = Y \oplus (X \vee \bar{Z}).$$

Note  $\bar{1} = 0$  and  $\bar{0} = 1$  (this NOT). Note  $0 \vee 0 = 0$ ,  $0 \vee 1 = 1$ ,  $1 \vee 0 = 1$ ,  $1 \vee 1 = 1$  (this is OR).

For example, let us apply  $F$  to three bytes (instead of 3 words). Let  $X = 00001111$ ,  $Y = 00110011$ ,  $Z = 01010101$ .

$X$	$Y$	$Z$	$XY$	$\bar{X}Z$	$XY \vee \bar{X}Z$
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	0	0
0	1	1	0	1	1
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	1	0	1
1	1	1	1	0	1

So  $F(X, Y, Z) = 01010011$ . Note that for 1-bit inputs to  $F$ , we had all 8 possibilities 000,  $\dots$ , 111 and the outputs were 0 half the time and 1 half the time. This is true for  $G$ ,  $H$ , and  $I$  as well.

### The constants

There are 64 constants  $T[1], \dots, T[64]$ . Let  $i$  measure radians. Then  $|\sin(i)|$  is a real number between 0 and 1. Let  $T[i]$  be the first 32 bits after the decimal point in the binary

representation of  $|\sin(i)|$ . Be careful, here the index starts at 1 because  $\sin(0)$  is no good.

### Rotation notation

If  $E$  is a 32-bit string, and  $1 \leq n \leq 31$  then  $E \lll n$  is a left rotational shift of  $E$  by  $n$  bits. So if  $E = 01000000000000001111111111111111$  then  $E \lll 3 = 00000000000001111111111111111010$ .

### More notation

We let  $+$  denote addition modulo  $2^{32}$  where a word is considered to be the binary representation of an integer  $n$  with  $0 \leq n \leq 2^{32} - 1$ .

In computer science  $x := y$  means set  $x$  equal to the value of  $y$  at this moment. So if  $x = 4$  at a particular moment and you write  $x := x + 1$  then after that step,  $x = 5$ .

### Evaluating $f$

$f$  takes as input,  $X$  and 4 words which we denote  $A, B, C, D$ .

First the 4 words are stored:  $AA := A_0, BB := B_0, CC := C_0, DD := D_0$ .

Then there are 64 steps in four rounds.

Round 1 consists of 16 steps.

Let  $[abcd\ k\ s\ i]$  denote the operation  $a := b + ((a + F(b, c, d) + X[k] + T[i]) \lll s)$ .

Those 16 steps are

[ABCD 0 7 1] [DABC 1 12 2] [CDAB 2 17 3] [BCDA 3 22 4]  
 [ABCD 4 7 5] [DABC 5 12 6] [CDAB 6 17 7] [BCDA 7 22 8]  
 [ABCD 8 7 9] [DABC 9 12 10] [CDAB 10 17 11] [BCDA 11 22 12]  
 [ABCD 12 7 13] [DABC 13 12 14] [CDAB 14 17 15] [BCDA 15 22 16].

Round 2 consists of the next 16 steps.

Let  $[abcd\ k\ s\ i]$  denote the operation  $a := b + ((a + G(b, c, d) + X[k] + T[i]) \lll s)$ .

Those 16 steps are

[ABCD 1 5 17] [DABC 6 9 18] [CDAB 11 14 19] [BCDA 0 20 20]  
 [ABCD 5 5 21] [DABC 10 9 22] [CDAB 15 14 23] [BCDA 4 20 24]  
 [ABCD 9 5 25] [DABC 14 9 26] [CDAB 3 14 27] [BCDA 8 20 28]  
 [ABCD 13 5 29] [DABC 2 9 30] [CDAB 7 14 31] [BCDA 12 20 32]

Round 3 consists of the next 16 steps.

Let  $[abcd\ k\ s\ i]$  denote the operation  $a := b + ((a + H(b, c, d) + X[k] + T[i]) \lll s)$ .

Those 16 steps are

[ABCD 5 4 33] [DABC 8 11 34] [CDAB 11 16 35] [BCDA 14 23 36]  
 [ABCD 1 4 37] [DABC 4 11 38] [CDAB 7 16 39] [BCDA 10 23 40]  
 [ABCD 13 4 41] [DABC 0 11 42] [CDAB 3 16 43] [BCDA 6 23 44]  
 [ABCD 9 4 45] [DABC 12 11 46] [CDAB 15 16 47] [BCDA 2 23 48]

Round 4 consists of the last 16 steps.

[abcd k s i] denote the operation  $a := b + ((a + I(b, c, d) + X[k] + T[i]) \lll s)$ .

Those 16 steps are

[ABCD 0 6 49] [DABC 7 10 50] [CDAB 14 15 51] [BCDA 5 21 52]  
 [ABCD 12 6 53] [DABC 3 10 54] [CDAB 10 15 55] [BCDA 1 21 56]  
 [ABCD 8 6 57] [DABC 15 10 58] [CDAB 6 15 59] [BCDA 13 21 60]  
 [ABCD 4 6 61] [DABC 11 10 62] [CDAB 2 15 63] [BCDA 9 21 64]

Lastly, add in the saved words from the beginning.

$A := A + AA$ ,  $B := B + BB$ ,  $C := C + CC$ ,  $D := D + DD$ .

The output of  $f$  is the concatenation ABCD, which has 128 bits.

### Clarification

Let us clarify the notation above. First let us look at Step 1. Before Step 1, we have  $A = A_0 = 67\ 45\ 23\ 01$ ,  $B = B_0 = \dots$ ,  $C = C_0 = \dots$ ,  $D = D_0 = \dots$ . Recall

[abcd k s i] denotes the operation  $a := b + ((a + F(b, c, d) + X[k] + T[i]) \lll s)$ . And Step 1 is [ABCD 0 7 1].

So that means

$A := B + ((A + F(B, C, D) + X[0] + T[1]) \lll 7)$

or

$A_1 := B_0 + ((A_0 + F(B_0, C_0, D_0) + X[0] + T[1]) \lll 7)$ .

So first we evaluate  $F(B_0, C_0, D_0)$ . Then we turn it into an integer. Then we turn  $A_0$ ,  $X[0]$  and  $T[1]$  into integers. Then we add the four integers together modulo  $2^{32}$ . Recall  $X[0]$  is the beginning of the 512-bit input message and  $T[1]$  comes from  $\sin(1)$ . Take the result and convert it to a 32-bit word and shift left by 7. Turn that back into an integer and turn  $B_0$  into an integer and add those two modulo  $2^{32}$  to get  $A_1$ . Turn  $A_1$  back into a 32-bit word. Now  $A = A_1$ ,  $B = B_0$ ,  $C = C_0$ ,  $D = D_0$ .

Now we move onto Step 2. Recall

[abcd k s i] denotes the operation  $a := b + ((a + F(b, c, d) + X[k] + T[i]) \lll s)$ . And Step 2 is [DABC 1 12 2]. So that means

$D := A + ((D + F(A, B, C) + X[1] + T[2]) \lll 12)$  or

$D_1 := A_1 + ((D_0 + F(A_1, B_0, C_0) + X[1] + T[2]) \lll 12)$ .

Now  $A = A_1$ ,  $B = B_0$ ,  $C = C_0$ ,  $D = D_1$ .

### Hash algorithm

We will call the input to the hash algorithm the message. Let us say that the (padded) message has  $3 \cdot 512$  bits and consists of the concatenation of three 512-bit strings  $S_1 S_2 S_3$ . First we take  $S_1$  and break it into 16 words:  $X[0], \dots, X[15]$ .

We start with the buffers  $A = 0x67452301$ ,  $B = 0xefcdab89$ ,  $C = 0x98badcfe$ ,  $D = 0x10325476$ . We go through the algorithm described above. After the 64th steps  $A = A_{16}$ ,



## Homework

CW-Hist-5. Decrypt the following two ciphers. Each uses a different monalphabetic substitution cipher. They are in the files mon1.txt and mon2.txt. For the second one, I found it helpful to note that the most common two letter words in English are OF, TO, IN, IS, HE, IT, AS, ON, BE, AT, BY, OR, AN, WE, NO IF, SO, UP, DO, MY, ME, MR, US, and GO (from most to least frequent).

KD KU JT YTJECB DXOD JCYDTJ MODCB YBTDC DT XTTRC KA K XOFC UCCJ  
AOBDXCB DXOJ ECUZOBDCU KD KU LCZOQUC K XOFC UDTTE TJ DXC  
UXTQMECBU TA HKOJDU.

BKGF TCATYTAGENTDBD BFCA BR LF MGTBF GCRLDFKYECB TB TD BSF DCRL  
BSF HRGNA LF DRXSTDBTUEBF BSF QKTJSBFCFA URCQRKPTDB HSR ZFFXD  
E QEDUTCEBFA RK HRKKTFA FWF RC HSEB TD TC BSF HTCA

CW-Hist-6. The following was encrypted using the Vigenere cipher. (If you want it, the ciphertext is in the files ct2num.txt and ct2let.txt).

ptugycymhz gvvzvfxklz gypvjhzlsd smyckvxvvv atzewfxzld oglzvfrmzv  
rtfqffgprx halaycelwt vhpncoshw welmehhjl m fvojfffhwj ogksmfavqv  
fhvqnolalv tbywkhhlrk skdlzxxdez hbukwckalv fxzxkcvvqv wgalvfxdej  
delrkmhmzx axastcgaid dehvxhalwy oovajowcee qbukrqkvwj halzzzxek  
halfrgxvjk vxhpkokyey zpoiekxmmi gmhviwolhk vxueifhds w halechxyvr  
wejsmsklhf befejatspg ckamfbhmxy smyrbzcp r fmpvgtuhs mmoivbwvjd  
olzecahzxk vxlrkwkxli wtukcsphwz bloeucpdlv bbhwbswtcj semhzrmoij  
vtksnqhccii vtsjkvxhvv oboeubmzxm rblhrbrmsi atskvcflxi mrlxsjimpjz  
uoyiu

Determine the length of the keyword. First use the Kasiski method to make a conjecture. Find several trigraphs and factor the distances between them. Then use the Friedman test to estimate the length of the keyword. The number of characters is 485 the frequencies for A - Z are [19, 16, 16, 11, 22, 21, 12, 33, 14, 15, 27, 32, 23, 5, 15, 13, 7, 16, 20, 13, 10, 41, 21, 26, 12, 25] Now go into PARI and type  $k = \text{the number you believe is the length of the keyword}$ . Then type  $l=485$ . (That's a lower case L). Then read in the file ct2num.txt. (You do that by typing `\r ct2num.txt`). Then read in the file hist.txt. You should have  $k$  output vectors of length 26. The first shows a histogram (frequency of  $[A, \dots, Z]$ ) of the letters appearing in positions 1,  $k+1$ ,  $2k+1$ , etc. The second shows a histogram of the letters appearing in positions 2,  $k+2$ , etc. Now write down those histograms and underline every number that is 6 or more. Use the fact that the distances between A, E, T and A in the alphabet are 4, 15, and 7 to decide how much each shift was. Determine the keyword. Let's say that you

believe the keyword is length 7 (it's not) and that the shifts are by 20, 13, 2, 14, 24, 9, 5. Then type `w=[20,13,2,14,24,9,5]`. Now read in `decvig.txt`. That is the plaintext with `A=0, ..., Z=25`. You'll probably want to write it to a file and print it. Or feel free to use a word processor or write a program to decode.

RT-1. Find the running time required to compute  $5^N$  in terms of  $N$ . The computer program would compute  $((((5 \cdot 5) \cdot 5) \cdots) \cdot 5)$ .

RT-2. Find the running time required to compute  $M^N$ , in terms of  $M$  and  $N$ . The computer program would compute  $((((M \cdot M) \cdot M) \cdots) \cdot M)$ . Note that if  $M, N$  and  $n$  are all similar size, then computing  $M^N$  and reducing it mod  $n$  would be incredibly slow. But using repeated squares to compute  $M^N \pmod n$  is polynomial time:  $O(\log^3(n))$ . In PARI, typing `Mod(M,n) ^ N` tells it to use repeated squares. If you type `Mod(M^N,n)`, it will slowly compute  $M^N$ , then reduce mod  $n$ .

RT-3i). I'm telling you that  $1 + 2 + \dots + N = N(N + 1)/2$ . A stupid programmer doesn't know this and writes a program to compute that sum the slow way  $((((1 + 2) + 3) + \dots + N)$ . Find the running time this will take in terms of  $N$ . ii) Find the running time it would take you to compute  $N(N + 1)/2$ . You'll see this is much faster.

RT-4. The prime number theorem says that the number of primes up to  $N$  is about  $N/\ln(N)$ . Another cool fact is that the product of all the primes up to  $N$  is about  $e^N$ . Using these two facts, find the running time it would take to compute the product of all primes up to  $N$ . So you would compute  $((((2 \cdot 3) \cdot 5) \cdots) \cdot p$  where  $p$  is the largest prime  $\leq N$ . In general,  $p \approx N$ . For simplicity, assume that you already have a list in storage of the primes up to  $N$ , so you don't have to worry about the time to compute them.

RT-5. We want to test if an integer  $N$  is prime. We use a simple trial division. In other words we divide every integer up to  $\sqrt{N}$  into  $N$  to see if  $N$  factors. Find the running time it would take to do this. (Note, a polynomial time algorithm for testing if an integer is prime was developed in 2002 by Prof. Agarwal and his two students Saxena and Kayal in India).

RT-6. Let  $\alpha = (1 + \sqrt{5})/2$ . Let  $F_n$  denote the  $n$ th Fibonacci number. We have  $F_1 = F_2 = 1$  and for  $i \geq 3$ ,  $F_i = F_{i-1} + F_{i-2}$  (so  $F_3 = 2, F_4 = 3, F_5 = 5, F_6 = 8, \dots$ ). Recall  $F_n \approx \alpha^n$ . Find the running time of an algorithm that exactly finds the integer  $\prod_{i=1}^n F_i$  using  $((F_1 \cdot F_2) \cdot F_3) \cdots F_n$ . Your answer should be  $O()$  of a function of  $n$  and not have an  $F$  in it. For simplicity, assume that you already have  $F_1, \dots, F_n$  in storage, so you don't have to worry about the (negligible anyway) time to compute them.

RC4-1. You and Alice are using RC4 with  $n = 3$  and the key 6, 7, 0, 3, 1, 3 (really 110111000011001011). Alice sends you the ciphertext 101 100 001 110 000 000 111 011 111 110 110. Decrypt. In this problem, we use 3-bit encoding for the letters A - H by A = 000, B = 001, ..., H = 111.

RC4-2. i) I want you to write a program that has three inputs: the integer  $n$ , the integer  $l$  and the array of bits which is the key. The output of the program should be an array of bits which is the keystream (it should have length  $n * l$ ). I want your program to include two subroutines. **I want your program to have lots of comments so I can follow it and help with bugs.** Please print out your program and turn it in along with the results.



Time-2. At a time between timestamps, which documents must Trent store? Is that a lot?

Time-3. Is it necessary for the timestamper Trent's public key (which is used for verifying his signature) to be available to everyone or just the people who ask for their documents to be timestamped. Explain.

Time-4. If  $I_{n-1}$  and  $I_{n+1}$  both die, then Alice could cheat. Describe a reasonable adaptation of the protocol that takes care of this problem.

MD5-1. If  $X = 11111111111111111111111111111111$ , and  $Y$  and  $Z$  are arbitrary, then find the simplest formula for  $F(X, Y, Z)$ .

MD5-2. Let's get practice with a simplified MD5 algorithm. Instead of words of length 32 we will only have words of length 8. Now  $+$  means addition mod  $2^8$ . We will work through the 18th and 19th steps (the steps can be numbered from 1 to 64). Instead of left shifts of 9 and 14 we will do shifts of 5 and 2, respectively. At the beginning of the 18th step we have  $A_5 = 10010110$ ,  $B_4 = 00100111$ ,  $C_4 = 01110100$ ,  $D_4 = 11011100$ ,  $X[6] = 01100010$ ,  $X[11] = 00111011$ ,  $T[18] = 11000000$ ,  $T[19] = 00100110$ . Find the output of the 19th step, namely  $A_5, B_4, C_5, D_5$ . Hint:  $D_5 = *01 * *000$  and  $C_5 = *10 * *110$ .

MD5-3i) Let  $y$  be a 128 bit hash value. On average, there ought to be  $n$  different 512-bit messages  $x$  hashing to  $y$ . Find  $n$ .

ii) Let us say that Eve has  $y$  and is trying to solve the one-way problem. So  $y$  is given and there at the start there are no known  $x$ 's such that  $f_{\text{hash}}(x) = y$ . Eve wants to find an  $x$  such that  $f_{\text{hash}}(x) = y$ . Since there are so many different messages hashing to  $y$ , Eve should be able to set  $m$  of the 16  $X[i]$ 's to

00000000000000000000000000000000 (the 0-word) and (on average) still find a 512-bit message

$x = X[0]X[1] \dots X[15]$  hashing to  $y$ . Find  $m$ .

MD5-4. Eve wants to solve the one-way problem for a given 128-bit hash value  $y$ . Again, no  $x$  is known so that  $f_{\text{hash}}(x) = y$ . As we saw in the problem above, Eve can set several message words to the 0-word and still expect to solve the one-way problem. Let us say that Eve sets each of  $X[9], X[2], X[11], X[4], X[13], X[6], X[15], X[8]$  to the 0-word. Eve wants to try to go backwards through the MD5 function in order to try to solve the one-way problem. Given  $y$ , how far back can she solve? Hint: It's easy to determine  $A_{16}, B_{16}, C_{16}, D_{16}$ . Now figure out what the 64th step is using subscripts and see what you can solve for. How far back can she get? Your answer should be of the form: She can find  $C_9$  but not  $D_9$ . (Though that is incorrect).

For MD5-5 and MD5-6, **I want your program to have lots of comments so the grader can follow it and help with bugs.**

MD5-5. i) Create four subroutines named ffunction, gfunction, hfunction and ifunction. The first should take as input three arrays of length 32 and return the output of the  $F$  function on them. The second through fourth subroutines should do the same for the functions  $G, H$  and  $I$ . **Print out the four subroutines for your homework.**

Look in the file *MD5help* and run all four functions with the input xxx, yyy, zzz. **Print**

**the output (for your homework).** Partial answer as a check: The outputs should start [0, 1, 0, 1, 0, 0, 1, 1, . . . , [0, 0, 1, 0, 0, 1, 1, 1, . . . , [0, 1, 1, 0, 1, 0, 0, 1, . . . , [1, 0, 0, 1, 1, 1, 0, 0, . . .

ii) Create a subroutine named shiftvec that takes as input an array of length 32 and an integer  $n$  and rotationally shifts to the left by  $n$ . **Print the subroutine.** Look in the help-file at my website and run this function on a0 with  $n = 3$ . **Print and the output.**

MD5-6. i) Create four subroutines named round1step, round2step, round3step, round4step. round1 should take as input four arrays (say  $a, b, c, d$ ) of length 32 and three integers (say  $k, s, i$ ) and return the array  $b + ((a + F(b, c, d) + X[k] + T[i]) \lll s)$ . **Print the subroutines.** You need NOT name the variables  $a, b, c, d, k, s, i$ .

Be very careful. After evaluating  $F(b, c, d)$  you will need to turn  $a, F(b, c, d), x[k]$  and  $t[i]$  into integers before adding them mod  $2^{32}$ . Then back to an array for the left-shift. Then back to an integer so you can add that mod  $2^{32}$  to  $b$  (represented as an integer). The output should be an array again.

Be careful with ts. If your programming language considers the first element of an array to have index 0, then  $ts[0]=T[1]$ ,  $ts[1]=T[2]$ , etc.

Apply each round to  $a=a0, b=b0, c=c0, d=d0, k=0, s=7, i=1$ . You can find a0, b0, c0, d0, T and an X in the help-file at my website. **Print the outputs.** Partial answer as a check: The outputs should start [0, 1, 0, 0, 0, . . . , [0, 0, 1, 0, 1, . . . , [1, 0, 0, 0, 1, . . . , [1, 0, 1, 0, 0, . . . I am NOT sure of these. If you are getting something different and you are pretty sure you are right, e-mail me your code and your outputs. If a few students get the same thing and it's different from mine then the group is probably right.

Note that if you can do this with one subroutine (instead of four) that takes an the extra input which is a function (ffunction, . . . , ifunction) then be my guest.

ii) Write a program that does MD5 once without padding. **Print the program in your homework.** Use the X in the file. **Print the output.** Partial answer as a check: The outputs should start ( $A =$ )[1, 1, 1, 0, 0, . . . , ( $B =$ )[0, 1, 1, 1, 0, . . . , ( $C =$ )[0, 0, 0, 0, 0, . . . , ( $D =$ )[1, 0, 0, 1, 0, . . .

iii) Now change the first bit of X (so change that first 1 to a 0) but leave everything else the same. Apply MD5 to that. **Print the output.** Notice how different it is from the previous output.

Kerb-1. After step 3 and before step 4, what does TGS do? Answers should look like: First it encrypts/decrypts/checks . . . Second it encrypts/decrypts/checks . . . , etc.

Kerb-2. Can  $C$  forge  $\{c, a, v, K_{C,TGS}\}_{K_{TGS}}$ ? Explain.

Kerb-3. After step 5, what does S do? Answers should be like in Kerb-1.

Kerb-4. Can  $C$  forge  $\{c, a, v, K_{C,S}\}_{K_S}$ ? Explain.

Key-1. There are 26 lowercase letters, 36 lowercase letters and digits, 62 letters and digits, 95 printable characters, 128 standard ASCII characters and 256 8-bit ASCII characters. Still many people have passwords that are 8 characters or less and consist entirely of lowercase letters. Assume that a person's password consists of 8 lowercase letters. You can try 200,000 per second (this is reasonable right now). **On average** how many hours will it take before

you find such a person's password? Professor Schaefer's password consists of 8 printable characters. How many years would it take in order to **guarantee** that you can find his password (assuming no increase in computer speed)?

CW-SS-1. From the Lagrange Interpolating Polynomial Scheme. There are 10 managers and it takes any four to get a secret message. We have  $p = 1237$ . Managers 1, 3, 6 and 7 collude to get the secret. Manager 1 has been given 83, Manager 3 has been given 1153, Manager 6 has been given 1014 and Manager 7 has been given 10. What is the secret message? (It is an integer between 0 and 1023.) In Pari, if you wanted to solve  $3x + 7y = 10$ ,  $8x + 5y = 2 \pmod{11}$  you could type ? m=[3,7;8,5]\*Mod(1,11) ? v=[10;2] ? matsolve(m,v)

DigCash-1. Before I explained RIS's, I gave a solution to problem 1. In that solution, the bank chooses 99 of the 100 messages Alice has sent the bank and asks Alice to unblind them. I want you to figure out if Alice sends the bank 99  $M_j$ s or 99  $r_j$ s in order that the bank unblind.

Alice wants to cheat. So she first sends 100 messages  $M'_j$ , where each corresponding  $M_j$  says "\$20". The bank asks Alice to unblind 99 of them (say all except  $M'_{47}$ ). For  $j = 1, \dots, 46$  and  $48 \dots 100$ . a) Should the bank ask Alice to send 99  $M_j$ s or 99  $r_j$ s (all except  $j = 47$ )? b) Explain why. c) Let's say that Alice wants to cheat and have the bank only deduct \$5 from her account. We are assuming she has already sent  $M'_1, \dots, M'_{100}$  to the bank where  $M'_j = M_j r_j^{e_B}$  where  $M_j$  says "\$20". So Alice decides, for each  $j \neq 47$ , she will make  $\hat{M}_j$  say "\$5". So now she needs to solve for the corresponding  $\hat{r}_j$  so that  $\hat{M}_j \hat{r}_j^{e_B} = M'_j$ . Explain why Alice can not find such a  $\hat{r}_j$ .

DigCash-2. If the bank asks Alice to unblind 99 of the 100 messages Alice has sent the bank then the probability that Alice can still cheat is  $1/100$  or  $1/2^{100}$ ? (By cheat I mean the one of the messages says "This is worth \$20" when the other 99 say "This is worth \$5".)

DigCash-3. a) In step 5 of the withdrawal algorithm, why does the bank check that  $y_{i,j} = H(x_{i,j})$  and  $y'_{i,j} = H(x'_{i,j})$  and  $x_{i,j} \oplus x'_{i,j} = \text{Alice} + \text{salt}_{i,j}$ ? b) Alice sends Carol  $M_{63}$ ,  $M'_{63}$ , and various  $x_{63,j}$ s and  $x'_{63,j}$ s. Why does Carol feel safe that those  $x_{63,j}$ s and  $x'_{63,j}$ s are not forged? Mind you, the bank never saw the  $x_{63,j}$ s, the  $x'_{63,j}$ s, the  $y_{63,j}$ s or the  $y'_{63,j}$ s.

DigCash-4. In class we had Carol challenge Alice with 50 bits. It would not be OK for Carol to instead challenge Alice with only 5 bits. Why? Explain what could go wrong and what is the probability that it does go wrong.

DigCash-4-2010. In class we had an example with  $k = 5$ . Explain what could go wrong with  $k = 5$  and what is the probability that it does go wrong?

DigCash-5. If Alice does not salt her identity, explain how Carol could try to frame Alice.

All election problems refer to Scantegrity II.

Elec-1. Ed is required to commit to the secret  $s$ . He picks a random  $t$  and publishes  $x := g^s h^t \pmod{p}$ . Later he wants to back out of his commitment. So he tries to find  $s'$  and  $t'$  so that  $g^{s'} h^{t'} = x$  with  $s \neq s'$ . Explain why he can't.

Elec-2. i) Why are commitments published for all elements of Table Q? ii) Why are commitments published for all elements of Table R?

Elec-3. It seems worrisome to a given voter than in the partially opened Table R, she will see either the Q-pointer for her CC or the S-pointer to her vote, but not both. So she does not see the connection between her CC and her vote. Though she must trust the secure work station somewhat, there are two other parts of this system that should make her feel secure that indeed in Table R, her Q-pointer and S-pointer are in the same row. What two parts of the system are those?

Elec-4. What would be the bad consequence of opening a side-by-side Q-pointer and S-point in Table R corresponding to a vote ballot (not an audit ballot)?

Elec-5. Let's say that half the Q-pointers and half the S-pointers have already been opened. Assume that someone programs the work station to now tamper with the hidden S-pointers and Table S. Explain how an observer can use probabilities to detect a significant amount of such tampering.