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.

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 permutation (starts with *zero*): 0L9FN2 TD3OPG HI1ZQC VARE45 XYUMSW 6B8K7J, keyword: CREAMY, ciphertext: FDDDFVDGVFXDVAFVAGXFGVDV.

Hist-4. You are an ancient Greek and you intercept a thin strip of paper with the following letters. Decrypt the message. If you prefer not to do this by hand, the message is also in a file called greek.html

ATAIIWSRTSIPTSILAHWNETHLINRHGROHDNDOERRSEBEJWNOONSUAESACDAEL
FRINKARNLAKTASNEDTRSGNIDTSHIOAGTCHTANUSLSAEHTTTPEWSSEGOAIRIT
MHUTFNOTSAOAHIGNHHLRRESSAHILNDWHHJISEOSAAOUSBRFHTNRTTAFAIEDE

In real life it would look like $\begin{vmatrix} \text{A} \\ \text{T} \\ \vdots \end{vmatrix}$

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. (So for i, find integers a and b so that $\gcd(21, 30) = 21a + 30b$.)

NT: 3 - 6, 8, 9: if you're working Mod m or in $\mathbf{Z}/m\mathbf{Z}$ all answers should be between 0 and $m - 1$.

NT-3. 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 $35^{-2} \pmod{73}$.)

NT-4. Make an addition table for $\mathbf{Z}/6\mathbf{Z}$. Make a multiplication table for $\mathbf{Z}/6\mathbf{Z}$.

NT-5. Make a multiplication table for $\mathbf{Z}/8\mathbf{Z}^*$. Make a multiplication table for $\mathbf{Z}/10\mathbf{Z}^*$.

NT-6. 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 10 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-7. For which positive integers $m \geq 2$ is the following statement true? $27 \equiv 5 (\text{Mod}\, m)$?

NT-8. Find the multiplicative inverses to all elements in $\mathbf{Z}/13\mathbf{Z}^*$. Your answer should look like: $1^{-1} = 1$, $2^{-1} = \ldots$. (Answers should all be from 1 to 12.)

NT-9. 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-10. Show that the sum of the squares of 2 odd integers is not the square of an integer. Hint: work Mod 4.

NT-11. Show $X^3 + X + 1$ is always odd if X is an integer.

NT-12. Find i) $\varphi(32)$, ii) $\varphi(100)$, iii) $\varphi(3600)$, iv) $\varphi(35)$, v) $\varphi(77)$.

NT-13. If $n = pq$ where p and q are different primes, find a formula for $\varphi(n)$ in terms of p and q.

NT-14. Omitted

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. This requires an intelligent brute force.

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 WFCCFNZEX 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 $A = 0, \ldots Z = 25$, and $- = 26$. Then we encode digraphs as in class (though for a 27-letter alphabet). You intercept some ciphertext encrypted with $C \equiv aP + b (\text{Mod}\, 729)$. 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 $C \equiv aP + b (\text{Mod}\, 26^2)$ to encrypt digraphs with $\gcd(a, 26^2) = 1$. We encode digraphs as $(x, y) \mapsto 26x + y$ where $x, y \in \{0, 1, \ldots, 25\}$ 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 in the collection of handouts at my website. (This homework problem has no writeup).

CW-LM-2. Use CryptoSoft to encode 'all is well' (without apostrophes) as an integer. Put it in the file t.txt (in newer Windows versions you just create a file called t). (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 $a^b \text{Mod}\, m$ (remember to do Mod(a,m)^b). Put it in your writeup. For this problem and in the future, if you hand write your homework and the answer is a very long number, feel free to write down the first four digits and then ...

ii) Find $a^{-1} \text{Mod}\, m$. Put it in your writeup.

iii) Confirm a^{-1} is right by doing the following. Lift $a^{-1} (\text{Mod}\, m)$ to an integer that is not inside a modular expression. Multiply this integer by a, subtract 1 and then divide by m. You'd better get an integer (think about why), what is it? Put that integer in your writeup.

iv) Find gcd(m,8689142). Put it in your writeup.

v) Read file t.txt and subtract the number 157351220028268828649062 40 from it and write it to the file u.txt. **REMEMBER: to get out of PARI** you type $\backslash q$. No need to write anything up for this one.

vi) Use CryptoSoft to decode the number in u.txt to plaintext and put it in your writeup.

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 postive 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 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 FF-1-iii)? (generalized from $p = 13$ to $p = 17$).

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 $s_1 \equiv g^k \equiv 5^7 (\text{Mod } 167)$. For $i = 1, 2, \ldots$ we have $s_{i+1} \equiv s_i^2 (\text{Mod } 167)$. For $i = 1, 2, \ldots$ we have $k_i \equiv s_i (\text{Mod } 2)$ 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. Read the paragraph above. Stream cipher: The ciphertext is 0101 0101 1110 1001 (I put in spaces to make it easier to read). Those are the bits $c_1 c_2 \ldots c_{16}$. The plaintext will be called $p_1 p_2 \ldots p_{16}$. The rule is $p_i = c_i \oplus k_i$ for $i = 1, 2, \ldots 16$. Decrypt and decode using ASCII.

SC-2. Self-synchronizing stream cipher: The ciphertext is 0110 1101 1111 0111 Those are the bits $c_1 c_2 \ldots c_{16}$. The plaintext will be called $p_1 p_2 \ldots p_{16}$. 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 $i = 1, 2, \ldots 16$. Decrypt and decode using ASCII.

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

FF-4. Factor (mod 2) all eight polynomials of the form $x^3 + b_2 x^2 + b_1 x + b_0$ into polynomials that are irreducible over $\mathbf{F}_2$, where $b_i \in \{0, 1\}$. For example, $x^3 + x^2 = x^2(x + 1)$, now you do the other 7. Recall, the irreducible polynomials over $\mathbf{F}_2$ of degree 3 or less are $x, x + 1, x^2 + x + 1, x^3 + x + 1$, and $x^3 + x^2 + 1$.

For problems FF 5 - 7, if you are working in $\mathbf{F}_2[x]/(x^d + \ldots)$ then all answers should be polynomials of degree $d - 1$ or less.

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 multiplicative inverse of $x^3 + x^2 + 1$.

AES-1. Omitted.

AES-2. For Simplified AES, verify by hand that 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. For Simplified AES, 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

$b_0b_1b_2b_3$			
$b_4b_5b_6b_7$			

to

$b_3 \oplus b_5$		$b_1 \oplus b_4 \oplus b_7$	
	$b_2 \oplus b_4$		$b_0 \oplus b_6 \oplus b_7$

.

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.) 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 in Cipher Block Chaining (CBC) mode 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, 7 omitted.

AES-8. i) Draw a decryption diagram for CBC. ii) Bit errors normally happen during transmission of ciphertext, not during encryption or decryption. Alice has a message that is not nice plaintext (perhaps it is a very long serial number with digits and characters and what not). She breaks the message into 10 blocks (namely $\mathrm{PT}_1, \ldots, \mathrm{PT}_{10}$) and encrypts them using AES in CBC mode and sends Bob $\mathrm{CT}_1, \ldots, \mathrm{CT}_{10}$. There is a single bit error in the transmission of CT_3. Which PT_i's will Bob correctly determine? Briefly explain. iii) Same problem as ii) except that the plaintext is the ASCII encoding of nice standard English. Which PT_i's can Bob correctly determine? Explain.

AES-9. Read the part of the AES handout that we did not cover in class. Mind you, I ALWAYS put something from this on the exam.

NT-18. Use repeated squares to reduce $17^{53} (\mathrm{Mod}\, 97)$.

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 $C \equiv aP + b (\mathrm{mod}\, 26)$ cipher. I encode the key(s) a and b as a single number by computing $26a + b$. 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 $C \equiv aP + b (\mathrm{mod}\, 26)$ 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 $C \equiv aP + b (\mathrm{mod}\, 26)$. 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 CryptoSoft is an implementation of Simplified AES in CBC mode with the initialization vector 1100 1011 1110 1000. Click on the tab 'Encrypt' to get there. For plaintext it allows upper and lower case letters, digits, spaces, commas and periods only. Don't use enter/carriage return.

For practice, encrypt 'do not go gentle into that good night' using the key 1010 1010 1010 1010 (don't type the spaces). On your homework write out the last four Hex ciphertext characters. If you don't know Hex, there's an aside at the end of this problem.

Copy the ciphertext (into a buffer) and click on the tab 'Decrypt' and paste the ciphertext into the appropriate field. Enter the same key and decrypt and make sure you get back the original plaintext. You don't turn anything for this paragraph's work, I just want to be sure you're using the software correctly.

Aside on Hex. Hex is an encoding of nibbles 0000, 0001, ..., 1001 encode to 0, 1, ... 9. 1010, 1011, ..., 1111 encode to A, B, ... F.

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
235911254160164630354666141906850066433 1
$e =$
726931696412943920543240897250097157237752381038093106931 08
258228010808455876045728920646158444819 1
Green (gr): $n =$
898273935565898045095450556378637890221991168914595 9
452631699963759986658392528774828563949800598 07
$e =$
440795955818489272223347981691856786085455386333305489849
776281698474322567978204145020005718349271
Orange (or): $n =$
459700032702649896889089741319217539006277929784455 89
494936157700926851903384528223275277440227877
$e =$
596131982616448003103784710980344171151668293418683956
626438045113323689246259881668637506766829519
Red (re): $n =$
185948607046586248125008473702524449964333045074220764 0
127531631364580491764319392630687618608632729
$e =$
574638142687916580559795101255825667110418211116048499 29
185417068586224589920143614162452544765727 7
Yellow (ye): $n =$
346643699133857057740759727545141687313760005768749115 4
546932385600262392592463260485733615351111 37
$e =$
876073790390834883594896630452458067807389902213279698204 6
308646517836887342023227950823447275089 11

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. Feel free to write only the first four digits of each long number in your writeup.

CW-RSA-3. There's a file called xxAESkey1.txt where xx is the first two letters of your color. This is a key for SAES in CryptoSoft that I have encrypted using your R.S.A. keys. Decrypt our shared key. It is an integer less than 2^{16}. In the file xxAESCT1.txt there is a message from me which I encrypted using SAES and our shared 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. (Remember to do Mod(a,m)∧b).

CW-RSA-4. Find your partner's color and encrypt the SAES key 51913 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 51913. In this problem, you won't actually use the SAES key to encrypt a message.

Now encrypt 51914 for your partner using RSA (include its encryption in your writeup). Notice the difference in ciphertexts from the two very similar plaintexts. Feel free to write only the first four digits of each ciphertext in your writeup.

RSA-4.5 In 2012, some cryptographers tested millions of n's and found that the primes p and q giving the n's had not been chosen randomly. In fact, many people shared their n with someone else. For other people, they shared one prime. Let's say that person 1 has $n_1 = r_1 r_2$, person 2 has $n_2 = r_1 r_2$, person 3 has $n_3 = r_3 r_4$ and person 4 has $n_4 = r_3 r_5$ where the r_i's are distinct primes. Assume that no other person uses r_1, r_2, r_3, r_4 or r_5. For simplicity, assume all people use $e = 17$ ($e = 17$ is very common). Assume that gcd'ing is very fast and factoring is very slow. i) Exactly who is a danger to person 1? Explain. ii) Exactly who is a danger to person 3? Explain. End problem: More detail on the history. They found 266729 certificates that share an n with another certificate (it may be that some of these different certificates belong to the same person). One n is used in 16489 certificates. They found 12720 n's that share exactly one prime with another n. There is one prime p that was found in 4627 different n's, each with a different q_i. So for $1 \leq i \leq 4627$ they had $n_i = pq_i$. They also found 9 primes $t_1, \ldots, t_9$ so that all 36 pairs $t_i t_j$ created someone's n.

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 $C \equiv aP + b (\mathrm{Mod}\, 26)$. 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. We are working with in $\mathbf{F}_p$ with p=nextprime($10 \wedge 24$) and g=5 (a generator of $\mathbf{F}_p^*$). This directory is in the file dlkey.txt.

Ed: 48353502076508378084583 1 Red: 63536411493608752727910 4
Yellow: 89112811029992952409505 4 Blue: 25802001445928668489126 9
Green: 79409656554774292610893 3 Orange: 97772629154205560490248 6

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

CW-DH-2. Diffie Hellman: Find your shared key with your partner. Reduce it mod 2^{16} (and put the reduction in your writeup). Use this integer (that is less than 2^{16}) as the key to pass messages back and forth with your partner using CryptoSoft encryption and decryption. (don't turn in anything for that). Make sure you read the above section **For the problems DH 2 - 3**.

CW-DH-3. Diffie Hellman: Find your shared key with Ed. Reduce it mod 2^{16} (and put the reduction in the writeup). Use this integer (that is less than 2^{16}) as the key for CryptoSoft SAES to decrypt the message in xxDHCT1.txt where xx is the 2 letter code for your color. Put output in writeup.

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 $\mathbf{F}_{2^{25}}$. The polynomial $x^{25} + x^3 + 1$ is irreducible over $\mathbf{F}_2$. A generator of $\mathbf{F}_{2^{25}}^*$ is x, which in Pari is Mod(Mod $(1, 2) * x, x \wedge 25 + x \wedge 3 + 1$). We want to use Diffie Hellman to agree on a SAES 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 $a_{24}x^{24} + a_{23}x^{23} + \ldots a_1x + a_0$ with $a_i \in \{0, 1\}$. We want to turn that into a 16-bit SAES key. It will help to lift it twice. Our simplified AES key will be $a_{24}a_{23} \ldots a_9$ (in that order, left to right). In the file allDHCT2.txt is a ciphertext. Decrypt using CryptoSoft's SAES. Put the 16 bit SAES key and the plaintext in your writeup.

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. Do this problem by hand and with a calculator.

EC-2. Double the point [-2,3] on the same elliptic curve using line intersections. Do this problem by hand and with a calculator.

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].

In newer versions of PARI that you download, you need ellinit to define an elliptic curve. So you could type e = ellinit([0,0,0,0,1]). Also ellpow has been replaced with ellmul.

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). Make sure you read the above paragraph **Working with elliptic curves in PARI**.

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 or e = ellinit(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)]).

CW-EC-6. Elliptic curve Diffie-Hellman key agreement. 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=nextprime(10∧25). Let e=ee*Mod(1,p), now the elliptic curve is defined over the finite field. If necessary, you should e = ellinit(ee*Mod(1,p)). My public key point on this elliptic curve is [166985550562940165155251, 130382107446059973115551 8], it is the file eckey.txt. Your private key is a=nextprime(10∧22), so your public key is ag=ellpow(e,g,a) (which you don't need to compute).

Find our shared key. Our shared key is a point on the elliptic curve. If Q is a point on an elliptic curve, you can get its x-coordinate by typing $Q[1]$. Lift the x-coordinate of our shared key. Reduce it mod 2^{16}. That represents an SAES key. In the file ECCT1.txt is a message that I encrypted with CryptoSoft's SAES using the SAES key you just got. Put the (integer) SAES key and plaintext in your writeup.

There is no EC-8.

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 257). Let $f = t^{16} + t^6 + t^2 + t + 1$. We represent $\mathbf{F}_{2^{16}}$ as $\mathbf{F}_2[t]/(f)$. I use t here instead of the usual x so as not to confuse with the x in the equation of the elliptic curve. The elliptic curve is $y^2 + xy = x^3 + 1$ or in Pari: $E = [1, 0, 0, 0, 1] * Mod(Mod(1, 2), f)$ (or $E = ellinit([1, 0, 0, 0, 1] * Mod(Mod(1, 2), f))$). My public key is the point $[t^{11} + t^8 + t^4 + t^3 + t, t^{13} + t^9 + t^8 + t^7 + t^6 + t^5 + t^4 + t^3 + 1]$. 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 SAES key. The x-coordinate is $a_{15}t^{15} + a_{14}t^{14} + \ldots + a_1 t + a_0$. Let the SAES key be $a_{15}a_{14} \ldots a_1 a_0$. Go to CryptoSoft and decrypt the message in ECAESCT1.txt. In your writeup put the key and plaintext.

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-HASH-1. Using AES as shown in class does not create a secure hash function. It does not have the one way property. I want you to demonstrate that you understand how to break the one way property by doing an example with simplified AES. Assume IV = $[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]$. Find M_1 and M_2, both 16 bit strings, so that the hash is $[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]$. To be precise, find M_1, a 16 bit string, so that when you encrypt it using simplified AES with IV as the key, you get an output C. Now encrypt M_2, a 16 bit string, so that when you encrypt it using simplified AES with C as the key, you get as output $[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]$. Explain to the homework grader how you did it (so s/he does not have to verify all of them using a computer). You should have an elegant solution that would also work for the real AES, where you would not have time to brute force a solution. Make sure you read the above section titled **Not a homework problem**.

HASH-1.5. We saw in HASH-1 that using symmetric key cryptography to create a hash function as described in class is not secure. We can, however, change it in a simple way so that it is a hash function. The security is based on the following fact. Let's say you are using a symmetric key cryptosystem to encrypt a single block of plaintext to get a single block of ciphertext using a key (in ECB mode). If Eve knows both the ciphertext block AND the plaintext block then it should be essentially impossible for Eve to determine the key. Use this property to design a hash algorithm (not a MAC - there should be no secret key - and not just a hash function) that should have the one-way property.

CW-MAC-1. MAC problem. The plaintext is *memory* (converted to ASCII in blocks: me mo ry).
The MAC key is $[0,1,0,0,1,1,0,0,1,0,0,0,0,0,1,1]$. 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. Copy your answer to your notes as you will need it again when you do CW-MAC-2.

HASH-2. Let $f : X \to Y$ be a hash function and assume $|X|/|Y|$ is very large (note $|X|$ and $|Y|$ are the sizes of the sets X and Y, respectively). I want you to give an informal proof that if f has the weakly collision free property then f has the one-way property. We will do this by contrapositive. So Assume that f does NOT have the one-way property. Given an informal proof that f will NOT have the the weakly collision free property.
Recall: One-way property: Given $y \in Y$ it is not feasible to find $x \in X$ such that $f(x) = y$.
Weakly collision free property: Given $x \in X$ it is infeasible to find $x' \in X$ with $x' \neq x$ such that $f(x') = f(x)$.

CW-ElG-5. (Note I have removed ElG-1 - 4.) For ElG 5, at the top of your homework, write your name, your and your partner's colors. ElGamal signature problem. In this problem both partners should send and receive signatures. Use the discrete log directory from right before problem CW-DH-2: p=nextprime(10^24) and g=5 and a=your secret key in xxFFkey.txt. Encode 'It is I' (without apostrophes) as an integer using CryptoSoft. We will call that number H (even though we are not bothering to hash it). Use the number in xxMOkey.txt as your random k. Compute $r = g^k$ Mod p and lift. Find your $x = k^{-1}(H + ar)$ Mod **p-1** and lift. Put r, x and H in your writeup. Send your partner (r,x,H).

ii) Receive (r,x,H) from your partner. The signature receiver works only Mod p. Compute g^H, g^{ar} (where g^a is your partner's public key; it is in dlkey.txt and the directory in earlier homework), and $g^H g^{ar}$. Then compute r^x. It should be the same as $g^H g^{ar}$. 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, H_1 and H_2, to Bob but uses the same k both days. Explain how Eve can determine a_A (Alice's private key). For simplicity, you may assume that the numbers appearing during the problem are relatively prime to p and $p-1$. This explains why Alice should use a different k each time she signs something.

ECDSA-1. In the file ECDSA.txt is the following:

```
p=nextprime(10^20)
E=[0,0,0,1,1]* Mod(1,p)
ell=33333333331225019431
G = [72,611]
aAG=[70115422955790749114, 633062636345802828 18]
kG=[17936361326783132673, 21712802287988569145]
H=49275910948174955198
x=7308730874663469007
```

Note E is our elliptic curve over $\mathbf{F}_p$. The number of points on $E(\mathbf{F}_p)$ is 3ℓ where ℓ (denoted ell in the Pari program) is prime. We have a pseudo-generating point G with $\ell G = 0$. The point $a_A G$ (denoted aAG in the Pari program) is Alice's public key point.

Alice hashed a document and encoded the hash as the integer H. She signed it using ECDSA. Her signature consists of the point kG, H and the x where x solves $kx = H + a_A x_k (\mathrm{mod}\, \ell)$. Note that x_k is the x-coordinate of kG which you can get using kG[1] in Pari. Note, I have not told you how to verify this signature, though it is analogous to doing so for the ElGamal signature system (and DSA). Figure it out and compute both sides of the equation (you better get the same thing both times). Once you know how to do this problem, open a log file (called pari.log) and compute both sides of the equation again (i.e. if you make a bunch of mistakes or have extraneous computations, then be kind to the homework grader and start it again). To create a log file in pari, type

```
? \l
```

Note that is the letter l (as in *log*) after the backslash. Print pari.log and attach it to your homework. (Warning: Pari is case sensitive. Also, if you enter $\backslash l$ a second time, Pari will toggle it off and stop logging what you typed.)

CW-MAC-2. This and MAC-3 are more realistic signature problems since usually what gets signed is a MAC. 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 $H = 1 \cdot 2^{15} + 0 \cdot 2^{14} + 1 \cdot 2^{13} + 1 \cdot 2^{12} + \ldots + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0$ (CryptoSoft can help). Sign that using your private key from problem CW-RSA-2. Put the signature in your writeup (first 4 digits is OK). Have your partner verify the signature using your public key. Copy the value of H to your notes as you will need it for CW-MAC-3.

CW-MAC-3. We want to sign the MAC from problem CW-MAC-1 using ElGamal signatures. Take the number H that you determined in problem CW-MAC-2. Use the p, g, private key and k from problem CW-ElG-5). Put the signature r, x, H in your writeup (first 4 digits is OK). Have your partner verify the signature.

ElG-7. The purpose of this problem is to understand where the security of the ElGamal signature system comes from. Let's say Alice uses the generator g of $\mathbf{F}_p^*$ and that her private key is a_A. So $g^{a_A} (\mathrm{mod}\, p) = PK$ is her public key.

Aside: Let's define the FFDLP a little more broadly. Let $h \in \mathbf{F}_p^*$ and z be some power of h in $\mathbf{F}_p^*$. Given h, z, and p, solving the equation $y^i \equiv z (\mathrm{mod}\, p)$ for i is the FFDLP. If p is large (and $p-1$ has a large prime factor), then for an overwhelming majoring of elements of h of $\mathbf{F}_p^*$, the FFDLP is very difficult to solve. End aside.

Alice has used the ElGamal signature system to sign a hash H (encoded as an integer). Freddy the forger has access to p, g, PK, r, x, H.

i) Let's say, for some freaky reason, that Freddy has access to k. Explain how it is that Freddy can quickly determine a_A.

ii). Now let's assume, as is usually the case, that Freddy does not know a_A or k. Freddy has g, p, and PK. Why can't Freddy use $g^{a_A} (\mathrm{mod}\, p) = PK$ to determine a_A?

iii) Still we assume that Freddy does not know a_A or k. Why can't Freddy use $r^x \equiv (g^H)(PK^r) (\mathrm{mod}\, p)$ to determine a_A?

iv) Still we assume that Freddy does not know a_A or k. Why can't Freddy use $xk \equiv H + a_A r (\mathrm{mod}\, p-1)$ to solve for a_A?

v) Still we assume that Freddy does not know a_A or k. We saw in part i) that if Freddy knows k, then he can quickly determine a_A. Why can't Freddy use $g^k \equiv r (\mathrm{mod}\, p)$ to determine k?

vi) Still we assume that Freddy does not know a_A or k. WHy can't Freddy use $xk \equiv H + a_A r (\mathrm{mod}\, p-1)$ to solve for k?

Cert-1. When your browser checks a certificate, it does four things: 1) hashes the top of the certificate, 2) confirms that equals the hash value at the bottom of the certificate, 3) raises the signature value to e_{Ver} mod n_{Ver} (assuming it's Verisign who signs the certificate), 4) confirms that (from 3)) equals the hash value. I want you to explain why steps 1 and 2 are necessary. Assume that you have a stupid browser that skips 1) and 2) and Eve knows it. I want you to explain how Eve can forge a certificate. At the top she gives a phony name and an n and and e for that phony name. Explain how Eve can fill in the hash value and the signature value so that your browser won't notice anything wrong. One solution is to

simply copy and paste the hash value and signature value from any other certificate. Come up with another solution.

CW-TLS-1. Everything should go in both partners' writeups. You and your partner decide who is Alice and who is Bob. Bob's p and q for RSA are in the file TLS-Bob.txt (pretend you don't have access to each others' files). Bob's RSA $e = 17$. Alice chooses key_{AES}, which is in TLS-Alice.txt as a bit string and as a vector. Alice chooses key_{MAC}, which is in TLS-Alice.txt as a bit string and as a vector. Alice concatenates key_{AES} and key_{MAC}, turns it into an integer (CryptoSoft) and encrypts with Bob's RSA keys (Bob's n is in TLS-Alice.txt) to get M_1. (Hint: $M_1 = 26 \ldots 65$.) Put M_1 in your **writeup**. Bob should find his RSA d and put d in your **writeup**. (Of course Alice shouldn't really know it. Hint: $d = 12 \ldots 53$.) Now Bob decrypts M_1 using his RSA keys and turns that integer into a bit string. He takes the first 16 bits and that's key_{AES} and the last 16 bits and that's key_{MAC}.

Now Bob encrypts the plaintext message *dock* for Alice with Simplified AES in CBC mode with key_{AES} and the initialization vector 1100101111101000 (So you can use CryptoSoft. Note that this initialization vector is automatic in CryptoSoft - you don't need to enter it.) Call the ciphertext M_2 and put it in your **writeup** (Hint: $M_2 = 0 \ldots$ C2 in Hex.) Bob will send that to Alice later.

Now he creates the encrypted, signed, MAC of the ciphertext. The MAC algorithm is the one described in class that exploits a symmetric key cryptosystem. Use simplified AES (you'll need to use renc in Pari - see the description before homework problem CW-HASH-1) with key_{MAC} as the key and the first 16 bits of ciphertext as the plaintext (weird, huh?). The output will act as the key used to encrypt the second 16 bits of ciphertext. The final output is the MAC. Call the MAC M_3 and put it in your **writeup**. (Hint: $M_3 = 010 \ldots 01$.) Bob turns the MAC M_3 into an integer and signs it with RSA. Call the signed MAC M_4 and put it in your **writeup**. (Hint: $M_4 = 17 \ldots 16$.) Bob turns the signed MAC M_4 into a bitstring and encrypts it in CBC mode with key_{AES} and the initialization vector IV=[1,1,0,0,1,0,1,1,1,1,1,0,1,0,0,0] (which is in TLS-Alice.txt and TLS-Bob.txt). Unfortunately you can not use Cryptosoft for that so you will need to use renc. Let M_5 be the encrypted, signed Mac and put it in your **writeup**. (Hint $M_5 = 101101 \ldots 001$ should have 32 bits.)

Bob sends M_2 (the encrypted plaintext message) and M_5 (the encrypted, signed, MAC of the ciphertext) to Alice.

First Alice should find the MAC of the ciphertext Bob has sent (M_2) using key_{MAC}. Let's call the output M_6. It is the putative MAC of the ciphertext.

Alice should now "undo" M_5 to find Bob's MAC. Alice uses rdec for the AES decryption (in CBC mode with IV=[1,1,0,0,1,0,1,1,1,1,1,0,1,0,0,0]) Then Alice uses RSA to undo the signature on the MAC. The output is Bob's MAC. Alice should confirm this equals M_6 - the putative MAC.

If those two MAC's agree, then Alice decrypts M_2 to get the plaintext message.

Assuming that Alice is convinced, via a certificate, that $n = 4290862423$ and $e = 17$ belong to Bob, i) explain in your **writeup** how Alice can be sure that only Bob could have sent this encrypted message and the encrypted, signed, MAC.

Let's say that Eve intercepts the messages Bob is sending to Alice. Eve removes the last

half of M_2 and sends it on to Alice. Alice decrypts the received 16 bits and gets *do*. ii) If Eve also sends the encrypted, signed MAC to Alice, what will Alice know? Explain in your **writeup**. iii) Instead of ii), can Eve forge an encrypted, signed MAC to agree with *do*? (Assume Eve knows about the plaintext *do*.) Explain in your **writeup**.

One lack of realism in this problem is that the message is about the same length as the encrypted, signed, MAC. In real life, the MAC will be, perhaps, 256 bits, the signed MAC will be, perhaps, 2048 bits (size of RSA n) and the encrypted, signed MAC will also be 2048 bits. The message is usually much longer than 256 characters.