Below are an initial set sample final exam questions from prior years' CS/ECE/Math 435 (I will provide additional sample problems during the coming week). These particular questions are courtesy of Professor Nigel Boston, from past years in which he taught 435.

Caution: As was noted for Prof. Boston's questions distributed for the midterm, these questions come from years when the exam format was in-person, with students completing hand-written, hardcopy exams. These are reasonably close in character to the long-format questions for this semester, but recognize that some adaptation for an on-line exam format will be necessary.

Prior Semesters 435 Final Exam Questions

Question 1. Compute the number of elements of the key space set for an affine cipher on Z_{26} (i.e., how many distinct key values?).

Solution 1: This is a relatively straightforward problem. Recall that an affine cipher encryption function, operating on a plaintext character x, will take the form:

$$e(x) = a*x + b$$

where coefficient "a" must be an element of $\mathbf{Z}_{26}$*, and b and element of $\mathbf{Z}_{26}$. The set $\mathbf{Z}_{26}$* has twelve elements (specifically {1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25}). The set $\mathbf{Z}_{26}$ of course has 26 elements. Therefore, the key space size is: 12*26 = 312.

Question 2. Consider a Diffie-Hellman key exchange between entity A and entity B, using parameters: prime number p, generator g, and integers a, and b. Of these four parameters, which ones remain internally secret to A in the Diffie-Hellman algorithm?

Compute the numeric value of the key shared between entities for the specific values of p = 101, g = 2, a = 4, b = 10 (this is the final value of key that A and B both compute to use in encrypting subsequent communication. This is computed after they've shared their public, "partial" keys in the Diffie-Hellman exchange).

Solution 2: Clearly, Prof. Boston used somewhat different notation in his past offerings of 435, so to be a legitimate question for this semester, the notation above should have been better explained. However, in Prof. Boston's notation, integer "a" ($0 \le a < p-1$) would be the internally secret quantity for entity A, and A would send mod(g^a, p) to entity B.

While not requested here, note that prime number p and generator g are made public by the network manager. The integer "b" ($0 \le b < p-1$), would be entity B's internally secret quantity.

The shared key is mod(g^(a*b), p), so here, mod (2^(40), 101). To evaluate, consider the following (certainly not unique) decomposition:

mod (2^(40), 101) = mod[{mod (2^(20), 101)*mod (2^(20), 101)}, 101]

= mod(…
mod[{mod (2^(10), 101)* mod (2^(10), 101)},101]

* mod[{mod (2^(10), 101)* mod (2^(10), 101)},101], 101)

Then mod(2^10,101)=mod(1024,101) = 14,

So mod (2^(20), 101)

= mod[{mod (2^(10), 101)* mod (2^(10), 101)},101] = mod(14^2, 101)

= mod(196, 101) = 95

And mod (2^(40), 101) = mod(95^2, 101) = mod(9025, 101)

If the last evaluation presents any challenge, just perform long division:

101 divides 9025 to yield an integer quotient of 89 (89*101=8989), with remainder 36. And of course, it is the remainder that determines the mod result, i.e. mod(9025, 101) = 36, so 36 is the desired solution; i.e.,36 is the value of the shared key between A and B.

Question 3. A linear feedback shift register (LFSR) produces a pseudo-random sequence in which the first nine bits are: 100011110.

(a) Show that these nine bits cannot be produced by a LFSR of order n=3 (for any possible choice of initial condition and coefficients), but that this sequence can be produced by a LFSR of order n=4. Find the coefficients and initial condition for the order n=4 LFSR consistent with these given nine bits, and use it to compute the next nine bits in the sequence.

(b) Suppose the pseudo-random sequence produced this LFSR is used in a stream cipher applied to an 18-bit plaintext message, which then produces 18-bit ciphertext of 101110111100010101. Find the plaintext.

Solution 3: To establish that the given pseudo-random sequence could NOT be produced by a LFSR of order n=3, simply consider the form of update equations fpr the third order case (note: while it is possible to apply the Berlekamp-Massey algorithm to see what order LFSR it produces here, that would be a poor solution strategy – it requires a very large amount of work for what can be shown much more simply below):

i) $c_0x_0 + c_1x_1 + c_2x_2 = x_3$

ii) $c_0x_1 + c_1x_2 + c_2x_3 = x_4$

iii) $c_0x_2 + c_1x_3 + c_2x_4 = x_5$

iv) $c_0x_3 + c_1x_3 + c_2x_5 = x_6$

…etc.

(recall all computations are mod 2).

Now, specifically consider the second update equation above, in which the left hand side operates on elements x1, x2, x3 in the pseudo-random sequence. In the given sequence, x1=0, x2=0, x3=0. Therefore, for ANY coefficient values for c0,c1, c2, a third order LFSR would have to produce a x4=0; but in the given sequence, x4=1. Therefore, we can conclude that this given sequence could NOT possibly be produced by a LFSR of n=3.

Our sequence is: 100011110.
And for the n=4 case, we can use the four update equations above, with the given x0 to x6 terms in the sequence, to construct linear equations that must be satisfied by c0, c1,c2,c4. The fact that x1=0, x2=0, x3=0 make this set of four equations particularly straightforward to solve. In particular, we have:

c0*1 + c1*0 + c2*0 + c3*0 = 1; conclude c0=1.

Next:

1*0 + c1*0 + c2*0 +c3*1 = 1; conclude c3=1

Then:

1*0 + c1*0 + c2*1 +1*1 = 1; conclude c2=0

1*0 + c1*1 + 0*1 +1*1 = 1; conclude c1=0.

Now, we can doublecheck if these coefficients work for the last given term in the sequence:

1*1 + 0*1 +0*1 + 1*1 mod 2 should = x8 (x8 is given as 0); it does.

<u>Question 4.</u> (a) Suppose that a block cipher is defined as follows. Encrypt bit strings of length 6 by mapping:

$[x_1, x_2, x_3, x_4, x_5, x_6]$ to $[x_4, x_5, x_6, x_1+x_5+x_6, x_2+x_4+x_6, x_3+ x_4 + x_5]$.

Encrypt the six bit plaintext 100111. Decrypt six bit ciphertext 100111.

(b) If $y_1, y_2, y_3, y_4, y_5, y_6$ are the six elements the ciphertext, write the six equations defining each element of the corresponding plaintext, in terms of y_1, y_2,

$y_3, y_4, y_5, y_6.$ What is the name given to this approach for constructing invertible functions?

Solution 4:
For the given cipher, the first three elements of the ciphertext y=e([100111] are easy: we just shift x4 to x6 to yield y1=1, y2=1, y3=1. Then:

y4=mod(x1+x5+x6, 2)=mod(3,2)=1

y5=mod(x_2+x_4+x_6, 2)=mod(2,2)=0

y6=mod(x_3+ x_4 + x_5, 2)=mod(2,2)=0

To invert this encryption function (observing that it is a form of Fiestel cipher), it's probably easiest to first perform the step asked for in (b), and identify the decryption function symbolically as:

d([y1 y2 y3 y4 y5 y6] = [y4–y2–y3, y5–y1–y3;y6–y1–y2;y1; y2; y3]

so the numeric values asked for in the decryption in (a) are:

d([100111])=[100100]

Question 5. Suppose Bob employs the RSA encryption system with public key N = 15, e = 3. Note that in contrast to any real-world, practical application, here N is a very small value that may be factored by hand calculation.

Factor N, and use this result to calculate Bob's internally secret decryption exponent d. Suppose Alice wants to send message x = 3 to Bob - what ciphertext y should she send? Show that decrypting this y does recover the original x=3.

Solution : Here we can easily factor N=15=3*5, so our primes that produced N are p=3 and q=5. The associate phi=(p-1)*(q-1)=8. Knowing that e=3, we're seek an integer d such that mod(d*3,phi)=1; that is, we seek a d such that mod(3*d,8)=1. We conclude d=3.

In sending message x=3 to Bob,Alice creates ciphertext as follows:

y= mod(x^e, N) = mod(3^3,15) = mod(27, 15) =12

Then Bob decrypts by computing:

mod(y^d, N) = mod(12^3,15)=mod(1728,15).

Long division will confirm that 15 divides 1728 with quotient 115 (i.e., 115*15 = 1725), and remainder 3. Hence, mod(12^3,15)=3, as desired.