HW 9 CMSC 456. DUE Nov 19
NOTE- THE HW IS THREE PAGES LONG

1. (0 points) READ the syllabus- Content and Policy. What is your name? Write it clearly. What is the day of the final? READ the slides and notes on Secret Sharing.

2. (30 points) Assume there is an α-SES. From class we know we can, with a hardness assumption, use the α-SES to get a (t, L) secret sharing scheme with shares of size $\frac{n}{t} + \alpha n$.

    1) (10 points) Use the α-SES to get a (t, L) secret sharing scheme with even SHORTER shares (though see next part). (NOTE – whatever you do to make it shorter, just do once. In a later part of this question you'll get to do it again!)

    2) (5 points) Your answer to the above question might not quite yield shorter shares. You need a condition on t, α. What is that condition?

    3) (10 points) Do a protocol that makes the shares even shorter! (See next part)

    4) (5 points) Your answer to the above question might not quite yield shorter shares. You need a condition on t, α. What is that condition?

GOTO NEXT PAGE

3. (40 points) For this problem you can assume that, for any t, L, there is a protocol for (t, L) secret sharing where everyone gets one share of size the size of the secret (or roughly). For the problems below explain it so that someone who has never seen secret sharing can understand it, though she knows that for all t, L there is a (t, L) secret sharing scheme where blah blah. (This is not hypothetical. I am having this one graded by someone outside the course grading this one.)

   Zelda has a secret $s \in \{0,1\}^n$. She wants to share a secret with $A_1, \ldots, A_{L_1}$, $B_1, \ldots, B_{L_2}$. such that the following happens:

   (a) If $\geq k_1$ of $A_1, \ldots, A_{L_1}$ meet with $\geq k_2$ of $B_1, \ldots, B_{L_2}$ then they can learn the secret

   (b) No other set of people can learn the secret.

   (c) Everyone gets a string of length roughly n (the roughly since we are over $\mathbb{Z}_p$ and not the field on 2^n elements.)

GOTO NEXT PAGE

4. (30 points) In this problem you will investigate the low quality and predictability of the Java random number generator. In order for this problem to work, your code must instantiate a single instance of "java.util.Random" and call the "nextInt()" method to generate random numbers.

   (a) Write a function "genList(int N)" which creates an array of N random integers

   (b) Write a function "countEvenSubseq(int[] list , int n)" which will return the number of contiguous subsequences of even numbers. To put it another way, this is the number indicies i so that list[i] ... list[$i + n - 1$] are all even. For example,

      i. countEvenSubseq({0,4,2,3}, 2) = 2
      ii. countEvenSubseq({0,4,2,2,10,8}, 4) = 3
      iii. countEvenSubseq({0,4,2,9,10,8}, 4) = 0

   (c) If the java random number generator were truly random, then what on average would you expect $countEvenSubseq(genList(N), n)$ to be? (Hint: there are $N - n + 1$ indices which could be the start of a contiguous subsequence. What is the chance that each one is all even? Now multiply those two numbers together.)

   (d) Write a program which outputs $countEvenSubseq(genList(N), n)$ for N=3145728, n=14. Try running it a few more times. How do the results differ from what you would have expected from part (c)?

   (e) (Not worth any points, just for your own information) see `https://www.javamex.com/tutorials/random_numbers/java_util_random_algorithm.shtml` to learn more about how the Java random number generator actually works. Can you figure out why part (d) worked the way it did? Hint: what are the factors of N?