Assignment 1 – Due Thursday, February 7

1. Textbook, page 98, problem 3.2.
2. Textbook, page 100, problem 3.15.
3. Textbook, page 142, problems 4.1–4.6.
4. For this problem, you are to explore how well just looking at letter frequencies works for breaking a simple monoalphabetic substitution cipher. You should write a program that does the following: Read an input file containing ciphertext character by character, counting how times each character occurs. Then, using a sorted list of character frequencies in English (readily available online), substitute the most frequent character in the ciphertext with 'e', the second with 't', etc. Your program should output this transformed text and you can view it and see if it makes sense. Does it look like English at all? Is it close enough where you can get some idea about what the plaintext says? The class web page (under "Assignments") has data files of various sizes that you should use, as well as sample code in C++ and Java for doing the input (you're not restricted to these languages, but these are the ones for which sample code is provided). Turn in your code electronically, and include in the paper submission that you turn in a short sample of the outputs (some test data is very large — don't print out the entire output, which could be hundreds of pages!) and a brief description of how well you think this technique works.
5. Even when encryption algorithms are strong, sometimes the way they are used renders a system insecure. Consider the following situation: someone designs a system to encrypt using AES, but makes a poor design decision so that only half of the key is random — in other words, they use AES, but the effective key length is 64 bits rather than 128 bits. In this question you will explore how secure such a system would be. For each part below, show your work — and by this, I don't mean just show raw calculations, but rather consider your writing to be for a situation in which you are trying to show someone (like your boss) how you arrived at your answers and why they are justified.
   (a) Recent Intel processors can perform AES encryptions at approximately 35 million encryptions/decryptions per second per core. How long would it take, on average, to perform a brute force attack on this system at this rate (assumptions: the attack uses a single core/computer, and the attacker knows the plaintext so no time has to be taken testing whether a plaintext is "likely").
   (b) According to the "Top 500" list at `http://www.top500.org/`, which lists the 500 fastest supercomputers in the world (at least the fastest that are publicly known), the world's fastest supercomputer in November 2012 was at Oak Ridge National Laboratory, running 560,640 cores. If all of these cores were put toward the task of a brute force attack on

this system, how long would it take (assumptions: each core can do 35 million encryptions/decryptions per second).

(c) If you were to build a special-purpose encryption-breaking machine, the most efficient design would use special purpose AES hardware that took advantage of pipelining and other efficiency tricks. For example, you could build chips based on the "Open source AES IP core" that is available online[1], which has a throughput of 185 million encryptions/decryptions per second. For fabricating these chips, assume you can put 32 copies ("cores") of this AES engine on a single chip, and can fabricate them in quantity for around $5 per chip. Building a machine has a lot more expense than just the AES chips though (circuit boards, power supplies, circuitry to test results, ...), so assume that the AES chips are half the cost of the final machine. With all of these assumptions, assume you have a budget of $1,000,000 — how long would it take to break this system with such a machine?

(d) Making a few modest improvements in efficiency from the previous scenario, let's say that the current maximum brute force speed for a specially-built $1,000,000 machine is 10^{15} decryptions per second. Now we flip the problem around and ask about minimum acceptable key size: you would like to design a cryptosystem such that it would still take at least 30 years on average to brute force on a "current" special-purpose machine, using a machine that is built 30 years from now (giving a secure lifetime of 60 years). Assume that Moore's Law continues for the next 30 years, doubling performance every 18 months.

6. Imagine encrypting credit card numbers with a straightforward use of AES. If I know a particular transaction uses Joe's credit card, and it's encrypted with a single AES block encryption, then if Joe's credit card is used in a future transaction I will recognize the ciphertext and know that it is Joe — in this way we can track how many transactions Joe makes, which violates Joe's confidentiality goals. The problem with this is that naively performing encryption with a block cipher is *deterministic* — it does the same transformation every time it is used. To counter this, we will add some randomness (or *non-determinism*): Before encrypting the credit card number, we will generate 128 random bits and will exclusive-or this with the plaintext credit card number, and the result is then encrypted with AES. In addition to the ciphertext, we then also transmit this random 128-bit "pad" — notice that the pad is chosen randomly, so it doesn't reveal anything about the plaintext, and randomizes what goes into the AES encryption. Specifically, if pad is a 128-bit random value, then we encrypt using $E(K, \mathsf{pad} \oplus P) \rightarrow C$ and transmit both pad and C. If we are unlucky and this technique picks the same pad twice for Joe, then we'll see the same transmission twice and Joe's confidentiality will be violated. What is the probability of this happening if Joe makes 100 credit card transactions? What is the probability if Joe makes 10,000 credit card transactions? How many transactions would Joe need to make before there is a probability of about $\frac{1}{2}$ of seeing a repeated transmission? (**Hint:** This is closely related to a standard probability problem called the "Birthday problem" or the "Birthday paradox" — look this up online if you are having trouble analyzing this situation.)

[1] See `http://www.rachiddafali.com/en/IP_core_library.html`.