EECS 122: Introduction to Communication Networks

Homework 10 Solutions

Solution 1. There are two ways to solve this problem. We can first use the Nyquist sampling theorem and the quantization noise formula to convert our signal to bits per second (as in homework 9 problem 3), then use the Shannon capacity theorem to determine what link characteristics are needed to support that data rate. Or we can just use Shannon for both conversions. Let's do the latter first. By our definition, high fidelity audio requires a channel with capacity $(20,000 \text{ Hz} - 20 \text{ Hz}) \log_2(1 + 84 \text{ dB})\text{bits}$ (neglecting to subtract the 20 Hz is acceptable, because makes so little difference), and the channel we are given has capacity $B \log_2(1 + 7 \text{ dB})\text{bits}$, where B is the bandwidth of the channel, which was not specified. We need the capacity of the given channel to be at least as great as what is required:

$$B \log_2(1 + 10^{0.7})\text{bits} \geq 19,980 \text{ Hz} \log_2(1 + 10^{8.4})\text{bits}$$

$$B \geq 19,980 \text{ Hz} \frac{\log(1 + 10^{0.7})}{\log(1 + 10^{8.4})} \doteq 215.4 \text{ kHz}$$

(The bases of the logs cancel out.) So we need the given link to have a bandwidth of at least 215.4 kHz. If we use the other method (convert to bits per second explicitly), we find that we need $\frac{84 \text{ dB}}{6 \text{ dB}} = 14$ bits per sample, and $2 \cdot 20,000 = 40,000$ samples per second (by Nyquist's sampling theorem), which is 560,000 bps. Then by Shannon's capacity theorem we need:

$$B \log_2(1 + 10^{0.7})\text{bits} \geq 560,000 \text{ bps}$$

$$B \geq \frac{560,000 \text{ bits/second}}{\log_2(1 + 10^{0.7})\text{bits}} \doteq 216.4 \text{ kHz}$$

The slight difference from the previous answer comes from the slight inaccuracy of the 6 dB per bit rule—it is actually a factor of four, which is $10 \log_{10}(4)\text{dB} \doteq 6.021 \text{ dB}$.

Solution 2. Each additional bit in the key doubles the number of keys, requiring computers twice as fast in order to try them all. If 56-bit key is the largest susceptible to a brute-force attack today, then a 128-bit key will be secure until computers double in speed 72 times, which by the assumptions of the problem will take $72 \cdot 18$ months $= 108$ years. (Of course, the notion of what is "feasible" is fuzzy—what's feasible for the average person today was already feasible for a very wealthy person/corporation/government several years ago, using older technology, but more of it.)

Solution 3.

a) There are 2^{64+56} triples $\langle X,K,Y\rangle$ such that $\mathrm{DES}(X,K)=Y$ (because you can pick any X and K, and then Y is determined). There are 2^{64+64} pairs $\langle X,Y\rangle$. There is a trivial function from $\langle X,K,Y\rangle$ to $\langle X,Y\rangle$ (just remove the K). The reverse relation is not necessarily a function, but it serves as a map from each pair $\langle X,Y\rangle$ to the set of keys that satisfy $\mathrm{DES}(X,K)=Y$. Therefore, there are $2^{64+56}/2^{64+64}=2^{-8}$ keys that fit each pair on average.

b) This is potentially interesting for code breaking because if you somehow learn one of the input blocks corresponding to an output block, and you somehow find a key that maps the input to the output, it is very likely that you have found the actual key that was used, which will then decrypt the rest of the message.

Solution 4. *B* might actually be talking to a man-in-the-middle *C*. Suppose *C* is some other server, and *A* wants to establish a connection to *C*. *C* can at the same time establish a connection to *B*, impersonating *A*, as follows:

A generates symmetric session key *K*
A sends *K* to *C* encrypted with *C*'s public key
C decrypts *K*
C sends *K* to *B* encrypted with *B*'s public key
B generates nonce *N*
B sends *N* to *C* encrypted with *K*
C relays the message to *A*
A signs *N* with *A*'s private key, sends result to *C* encrypted with *K*
C relays the message to *B*

B sees exactly the same messages as if *A* were establishing a connection to *C* (shown in the problem statement), but *B* is talking to *C*. And *C* knows *K*, so it can continue the rest of the conversation with *B*. The way to stop this attack is for *A* to include the name of the server in the list of things it signs. That way the last message sent to *B* would contain *C* instead of *B*, so *B* would know to reject it.