**Explain why the initial and final permutations of DES do not add to its security.**

Let <m, c> be the plain text, cipher text pair for DES. Consider an algorithm same as DES but without Initial and Final permutation. If someone can break this algorithm i.e. guess the key, then s/he can easily break DES as well. All s/he needs to do is to apply the reverse permutations i.e. Final Permutation on m to get m1 and Initial Permutation on c to get c1, and pass <m1, c1> to the code for breaking above algorithm. Thus, DES with Initial and Final Permutation is not any harder to crack than that without the permutations. Hence Initial and Final Permutations of DES do not add to its security.

If the initial permutation had value than it must be a factor in making DES

unbreakable (except by exhaustive key search). If this were the case than

removing the permutation should render DES breakable. However, the removal of

a known permutation which can easily be reversed cannot enable DES to be

broken and therefore it is clear that the permutation has no security value.

**Why must it be the case that the size of the cipher-text block is at least as big as the size of the plain-text block for any encryption algorithm?**

Let size of the plain text block be p and the size of the cipher block be c. Let us assume that c < p. Now, we have 2p different possible inputs and 2c different possible outputs. But 2p > 2c. So, more than one different inputs, say p1 and p2, would map to same output, say ci. In this case, it is impossible to decrypt ci correctly to exactly one of p1 or p2. Thus, the size of the cipher-text block must be at least as big as the size of the plain-text block for any encryption algorithm for correct decryption to be possible.

**3-DES involves an encryption (using K1) followed by a decryption (by K2) and then again an encryption (by K1). Why is the middle operation a decryption and not an encryption?**

Reasons are:

·         If the middle operation is encryption, then its Initial permutation cancels out Final permutation of first encryption and Initial permutation of last encryption cancels out the final permutation of middle encryption. If we instead have decryption in the middle, then Final permutation as well as Initial Permutation is applied twice. This adds some security to the algorithm.

·         With EDE, if k1 = k2, then the 3DES is same as DES with k1. This means that the 3DES system can be made to work with other systems using DES without any change in the implementation.

**What is an upper bound on the worst case number of attempts a brute force algorithm (one that sequentially tries all keys) must make to crack 3DES?**

Upper bound on the worst case number of attempts for brute force on 3DES is 2112.

**What is the output of the first iteration of the DES algorithm when the both plaintext and the key are all zero?**

The 64-bit input is x0=00...0 (64-zeroes). The initial permutation has no effect. Hence L0=00...0 (32-zeroes) and R0=00...0 (32-zeroes). Applying the key schedule which is a fixed permutation on the input bits of the key yields the round key K1= (00..0) (48-zeroes).

The round computes R1=L0 XOR f(R0, K1)  
The f-function

1. First expands R0 into 48-bit long bitstring using a fixed permuted expansion rule. Since only permutations and repetitions are used this will yield a 48-bit 0 string.
2. The result is XORed with K1, which produces a 48-bit zero string.
3. The 48-bit 0 string is divided into eight 6-bit chunks and the ith chunk is transformed under the rule specified in the Si box. 000000 is mapped 8 times with boxes Si i=1,..,8 and produces the following sequence of 4-bit values: 14, 15, 10, 7, 2, 12, 4, 13. In binary we obtain the following sequence:

1110 1111 1010 0111 0010 1100 0100 1101

1. Finally the bit-string is permuted according to the P table to the following:

1101 1000 1101 1000 1101 1011 1011 1100

This is the result of f(R0),K1)

The right half R1=L0 XOR f(R0),K1) is simply the output of f(R0),K1) since the L0 is zero. L1 = R0 = (00..0) (32-zeroes). Concatenating both yields the following 64-bit string, which is the output of round 1.

0000 0000 0000 0000 0000 0000 0000 0000 1101 1000 1101 1000 1101 1011 1011 1100

**(AES) Show the first eight words of the key expansion for a 128-bit key of all zeros in AES.**

, , , 

, , , 

**Compare AES to DES. For each of the following elements of DES, indicate the comparable element in AES or explain why it is not needed in AES.**

* 1. **XOR of subkey material with the input to the *f* function.**
  2. **XOR of the *f* function output with the left half of the block.**
  3. **The *f* function.**
  4. **Permutation *P.***
  5. **Swapping halves of the block**

a. AddRoundKey

b. Since AES is not Feistel cipher this step is not necessary.

c. ByteSub

d. ShiftRow and MixColumn.

e. No Swapping of halves.

**How many DES keys on average encrypt a particular plaintext block to a**

**particular ciphertext block?**

For a particular 64-bit block of plaintext there are 264 possible blocks of

ciphertext. There are only 256 possible DES keys. Each key has a one in 264 chance

of transforming the chosen plaintext into a particular ciphertext. If all possible

keys are tried the overall probability reduces to one in 28.

**What value do the 8 parity bits in DES have?**

The 8 parity bits have no security value.

**8. If keys Ke and Kf are used in an IDEA Even Round what values should these**

**keys take in the equivalent round for decryption?**

In the even round keys Ke and Kf remain the same for encryption or decryption.

**If a brute force attack (i.e. an exhaustive key search) is employed on a 64-bit**

**block encrypted by DES and a second block encrypted by IDEA how much more**

**computing power would be required to break IDEA as against breaking DES?**

If the computing power for encrypting a single block in DES is CDES and that for

encrypting a single block in IDEA is CIDEA then the increase in computing power

required for an exhaustive key search on IDEA is 272xCIDEA/CDES.

A virus is a program that is designed to spread from file to file on a single PC, it does not intentionally try to move to another PC, and it must replicate, and execute itself to be defined as a virus.  
A worm is designed to copy itself (intentionally move) from PC to PC, via networks, internet etc.  
A worm doesnt need a host file to move from system to system, whereas a virus does.  
So worms spread more rapidly then viruses.  
The word 'Virus' has become a common term a lot of people use to refer to worms and trojans too. Which is not exactly correct.

**Why we take 56 not 64 ?**

**Because we take 8 bits parity bits**