4.3 Further comments on DES

The S-boxes $S_1, S_2, ..., S_8$ are lookup tables (for each 6 bit input, they specify a 4 bit output).

- They have been carefully designed.
  For instance, their design already anticipated and protected against differential cryptanalysis (which wasn't publicly known at the time).
- On the other hand, they do not follow any simple rule. In particular, they must not be linear (or close to it). If they were, DES would be entirely insecure.
  [Slightly more specifically, if the S-boxes were linear, then all of DES would be, and, in the usual spirit of linear algebra, a few (m, c) pairs would suffice to recover the key.]
- They are also designed so that if one bit is changed in the input, then at least 2 bits of the output change.
  Important consequence. Go through one application of the round function $f_{k_i}(R)$, and convince yourself that flipping one bit of R has the effect of flipping at least two bits of $f_{k_i}(R)$. Repeating this for 16 rounds, you can see how the goal of diffusion seems to be achieved: changing one bit of the plaintext should change the ciphertext completely.

Example 121. Sometime it is stated that DES works with a 64 bit key size. In that case, every 8th bit is a parity bit, but the algorithm really operates with 56 bit keys.

Comment. Apparently, the NSA was interested in strengthening DES against any attack (recall that developments like differential cryptanalysis were foreseen) except brute-force. Indeed, the NSA seems to have pushed for a key size of 48 bits versus proposed 64 bits, and the result was a compromise for 56 bits.

Example 122. (homework) Can we (easily) break DES if we know one of the round keys?

Solution. Absolutely! Recall that each round key is 48 bits taken from the overall 56 bit DES key. Hence, we know all but 8 bits of the key. We just need to brute-force these $2^8 = 256$ many possibilities.

Example 123. If DES is insecure because of its 56 bit key size, why not just increase that?

Solution. DES was designed specifically for that key size. Increasing it necessitates a completely new analysis on how to choose the S-boxes and so on.

On the other hand. See the next example for how to leverage the original DES to increase the key size.

However. With the advent of powerful successors like AES there are very few reasons to use 3DES for new cryptosystems. (One slight advantage of 3DES is its particular small footprint in hardware implementations.)

Example 124. (3DES) A simple approach to increasing the key size of DES, without the need to design and analyze a new block cipher, is **3DES**. It consists of three applications of DES to each block and is still considered secure.

$$c = E_{k_3}(D_{k_2}(E_{k_1}(m)))$$

The 3DES standard allows three keying options:

- k_1, k_2, k_3 independent keys: $3 \times 56 = 168$ key size, but effective key size is 112
- $k_1 = k_3$: $2 \times 56 = 112$ key size, effective key size is stated as 80 by NIST
- $k_1 = k_2 = k_3$: this is just the usual DES, and provides backwards compatibility (which is a major reason for making the middle step a decryption instead of another encryption).

Comment. The reason for the reduced effective key sizes is the meet-in-the-middle attack. It is also the reason why something like 2DES is not used. See next example!

Example 125. (no 2DES) Explain why "2DES" does not really provide extra security over DES.

Solution. Let's denote DES encryption with E_k and decryption with D_k. The keys k are 56 bits.

Then, 2DES encrypts according to $c = E_{k_2}(E_{k_1}(m))$. The key size of 2DES is $56 + 56 = 112$ bits.

- A brute-force attack, would go through all possibilities for pairs (k_1, k_2), of which there is $2^{56} \cdot 2^{56} = 2^{112}$, to check whether $c = E_{k_2}(E_{k_1}(m))$. That requires 2^{112} DES computations.
- On the other hand, note that $c = E_{k_2}(E_{k_1}(m))$ is equivalent to $D_{k_2}(c) = E_{k_1}(m)$.

  Assuming sufficient memory, we first go through all 2^{56} keys k_2 and store the values $D_{k_2}(c)$ in a lookup table.

  We then go through all 2^{56} keys k_1, compute $E_{k_1}(m)$ and see if we have stored that value before. (Even though this is a huge table, the cost for checking whether an element is in the table can be disregarded; thanks to the magic of hash tables!)

  Whenever we have a candidate key $k = (k_1, k_2, k_3)$ (there will usually be many!), we do additional checks (like testing another block (m', c')) to see if k is really the key we are after.

  The total number of DES computations to break 2DES therefore is $2^{56} + 2^{56} = 2^{57}$, which is hardly more than for breaking DES!

This is known as a meet-in-the-middle attack.

https://en.wikipedia.org/wiki/Meet-in-the-middle_attack

Comment. The price to pay is that this attack also requires memory for storing This sort of approach is referred to as a time-memory trade-off. Instead of brute-forcing 2DES in 2^{112} time, we can attack it in 2^{57} time when using 2^{56} memory.

Comment. This applies to any block cipher, not just DES!

Comment. For some block ciphers it is the case that for any pair of keys k_1, k_2, there is a third key k_3 such that $E_{k_2}(E_{k_1}(m)) = E_{k_3}(m)$. In that case, we say that the cipher is a group, and double (or triple, or quadruple) encryption does not add any additional security! DES, however, is not a group.

Example 126. (homework) Explain why 3DES, used with three different keys, only has effective key size 112.

Solution. (details to be filled in) Instead of going through all k_1, k_2, k_3 to check whether

$$c = E_{k_3}(D_{k_2}(E_{k_1}(m)))$$

(which would take $2^{56} \cdot 2^{56} \cdot 2^{56} = 2^{168}$ DES computations), we can use that the latter is equivalent to

$$D_{k_3}(c) = D_{k_2}(E_{k_1}(m)).$$

Now proceed as in the previous example ... to see that we can break 3DES with 2^{112} DES computations. How much memory do we need?