Univ. Grenoble Alpes/Grenoble INP **2016–2017**
M2 Cybersecurity

Practical Work DES (January 2017)

Purpose : Differential attack of D.E.S. reduced to 3 and 6 rounds.

You need a program for DES following the standard description (therefore without optimization) in order to study its rounds. A reference version is given in in the file dea.c. We begin by perform the preliminary 3 round attack.

Each "binome" should mail a **report in PDF** as well as the source of the programs and example (**as a tar.gz**). with deadline 30/01/2017.

1. Perform a 3 round differential attack

References : see my course about Differential cryptanalysis and Stinson's book 1st edition.
Secret key to use and to recover : K=0x1A624C89520DEC46 ;

three pairs in input, given as : (X,Y) and (XS,YS) with Y=YS :
X[0]=0x748502cd ; XS[0]=0x38747564 ; Y[0]= 0x38451097 ;
X[1]=0x48691102 ; XS[1]=0x375bd31f ; Y[1]= 0x6acdff31 ;
X[2]=0x357418da ; XS[2]=0x12549847 ; Y[2]= 0x013fec86 ;

Here are the corresponding encryption by a DEA (DES without initial and final permutations) with three rounds.

```
ciphertexts :03c70306d8a09f10 78560a0960e6d4cb xor : 7b919fb8464bdb
ciphertexts :45fa285be5adc730 134f7915ac253457 xor : 56b5514e4988f367
ciphertexts :d8a31b2f28bbc5cf 0f317ac2b23cb944 xor : d79261ed9a877c8b
```

Compute the arrays corresponding to the sets $Test_j$ for each pair and by adding these array and looking for the 3 in them deduce 48 bits of the Key. Then recover the complete key.

Second step : write a program which generate a random key, three random pairs the 3R-attack, perform the 3R-attack and output the results.

2. Perform a 6 round differential attack

The purpose is now to implement the 6 round differential attack (as the easiest example of use of characteristics) to extend the previous 3 round attack.

We shall

— generate random pairs,

— use the first charactristic,

— Avoid the need of 2^{30} counter by looking for an admissible set of maximal length.

— for a pair of plaintexts M and M^*, we consider their xor M', their encryptions C and C^* and the xor C' of these encryptions. Notations for the program : $*$ is written S (as star) and $'$ written as P (as "prime") ; so $MP = M \oplus MS$, $CP = C \oplus CS$.

— we perform a 3-R attack using the 3 round characteristic with input 40080000 04000000 and output 04000000 40080000 with probability 1/16.

— The important formula is

$$G'_3 \oplus D'_6 = Z \oplus f(G_6, K_6) \oplus f(G_6, K_6)$$

with Z depending of D_3 and D'_3 therefore the blocks 2,5,6,7 and 8 are 0. We restrict ourselves to the 5 corresponding boxes and a part $\overline{K}_6$ of of the subkey K_6 containig the 30 bits which comes in our 5 boxes.

— We take enough random pairs (Stinson used 120 pairs) with prescribed xor and look for the elements in $Test_j(E_j, E^*, C')$, with

$$C' = P^{-1}(04000000 \oplus D'6),\ E = E(G_6), E^* = E(G^*_6)$$

for suggesting 5 pieces of K_6.

— The simplest idea to count for each value k of $\overline{K}_6$ the number of times it is suggested. So we need 2^{30} counters !

3. Admissible sets

The method we use following Biham shamir and Stinson uses the concept of **admissible sets**. We consider the set of pairs (M_i, M^*_i), $i = 0, ..., NP$ and its subsets defined by $i \in I' \subset \{0, ..., NP\}$. Such a I' is *admissible* if there exists at least one $\overline{K}_6$ suggested by the corresponding pairs. It is obvious that a subset of an admissible I' is again an admissible subset.

We can increase the length of admissible subsets by adding one pair and keeping it only if the new subset is still admissible. In that manner, we get a recursion construction of admissible sets. The good pairs (those compatible with the characteristic) all suggest the good $\overline{K}_6$. The bad one are spread on the 2^{30} possible value.

By increasing the number of elements, we end up by finding a unique maximal admissible subset suggesting only one $\overline{K}_6$.

How to test whether a subset I' with t elements is admissible : As in the 3 round attack, sum for $i \in I'$ the contents of the 5 arrays T_j, $j = 2, 5, 6, 7, 8$ relative to the corresponding pair ; each of sums should be an array containing at least one t.

Steps :

1 We construct 120 random pairs with prescribed xor 40080000 04000000 and encrypt them by DEA.

2 One computes the sets $Test_j(E_j, E^*, C')$, j=2,5,6,7 et 8.

3 The filtered pairs, that is the ones without any empty set $Test_j$, are the admissible sets with 1 element.

4 we enlarge them recursively to get 1, 2,3,....element sets until we reach the maximum set.

The key to use and to recover :
0x34E9F71A20756231