Chapter 11
Differential Crypt Analysis

February 15, 2010

11

Differential crypt-analysis (DCA) is a general form of cryptanalysis applicable primarily to block ciphers, but can also be applicable to stream ciphers and cryptographic hash functions. With respect to a block cipher, it refers to a set of techniques for tracing differences through the network of transformations, discovering where the cipher exhibits non-uniform behaviour, and exploiting such properties to recover the secret key.

11.1

DCA is a technique for attacking symmetric encryption algorithms which are composed of many identical stages or rounds, with each round controlled by a different sub-key. Usually these round sub-keys are derived from a single key, which can be referred to as the secret encryption key.

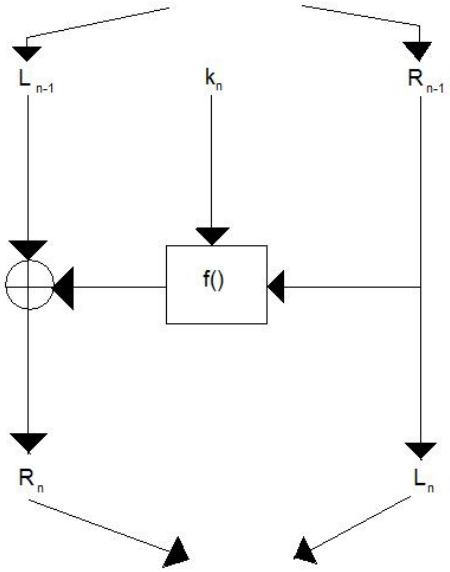
To be successful the attack needs to be able to exploit certain non-uniform characteristics in the target algorithm (as is the case with other similar attacks such as Linear Crypt Analysis). In the specific DCA case these non-uniformities are non-linear differential characteristics. If such characteristics are not present or present only at a very low exploitable probability, DCA cannot be used against the algorithm.

DCA was first described by Eli Biham and Ali Shamir in the Journal of Cryptology Volume 4, Number 1, 1991. They described DCA as applied to Data Encryption Standard (DES), the Data Encryption Algorithm. The discussion that follows here is directly taken from their original paper.

11.2.1

Assume we are analysing DES with n iterations, normally n would be equal to sixteen. The aim of the analysis is to discover the last key, k_n by inputting suitable plain-text P (64 bits) and observing the output cypher-text (R_n, L_n). If k_n is found we can obviously find (R_{n-1}, L_{n-1}) and use the same procedure to try to discover k_{n-1}; etcetera. When the key sequence is found then any cypher-text can decrypted. Differential crypt analysis is a chosen plain-text attack.

It should be noted that it is possible to perform the attack in reverse (i.e. on the decryption process) and try to discover k_1, k_2 etcetera.



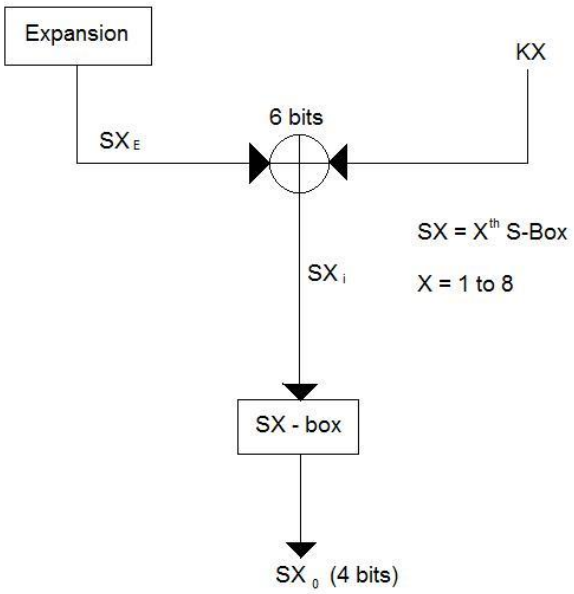
11.2.2

The differential technique involves inputting pairs of plaintext P, P^* such that $P + P^* = P' = constant$. (+ means bit by bit XOR, meaning exclu-

sive or). At the output we have L_n, R_n and L_n^* and the differential values $L_n + L_n^* = L'_n$, $R_n + R_n^* = R'_n$. Thus we input many pairs with constant P' ; and we examine the outputs L'_n, R'_n (which will not be constant because of the non-linear elements in DES) for the clues to k_n.

11.2.3

If we examine DES we see that all the stages are linear except for the S-boxes within $f()$. Note that the various permutations and the expansions move bits around but do not change their values and are therefore linear. Furthermore the keys K_i enter linearly into $f()$.



An S-box is a component that performs substitution. It provides Shannon's property of confusion.

In general, some number of input bits m are entered into an S-box, then they are transformed by the S-box into some number of output bits n. An mn S-box can be set up as a lookup table. Fixed tables are normally used, as in the Data Encryption Standard (DES), but this is not always the case.

We denote the inputs to the SX-Box as SX_i, SX_i^* with differential input $SX_i' = SX_i + SX_i^*$. Note that since KX is constant (the X^{th} 6-bit portion of the 48-bit sub-key) $SX_i' = SX_E'$. We can produce 64 different pairs (SX_i, SX_i^*) with constant SX_i' by varying SX_E and keeping $SX_E^* = SX_E + SX_i'$. (The effect is exactly the same as keeping (SX_E, SX_E^*) constant and varying KX; which of course we cannot do.)

We denote the outputs of the S-box by $SX_0 = SX(SX_i)$. the S-boxes are non-linear; therefore $SX(SX_E+SX_i^*) \neq SX_0+SX_0^* = SX(SX_i)+SX(SX_i^*)$.

Constant SX_i', constructed in 64 different ways, do not produce a constant SX_0', but rather a very skew distribution of SX_0' over the sixteen possible Hex values 0 to F. (This is to be contrasted with inputting 64 different (single) SX_i which produces exactly four occurrences of each of the possible sixteen values for SX).

Examples of this skew distribution for the first S-Box, S1, with $S1_E' = S1_i' = 34(HEX)$ and $S1_E' = S1_i' = OC$.

$S1_0'$	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S1_i' = 34$	0	8	16	6	2	0	0	12	6	0	0	0	0	8	0	6
$S1_i' = OC$	0	0	0	8	0	6	6	0	0	6	6	4	6	6	14	2

The figures in the table give the number of occurrences of the value $S1_0'$ for the input $S1_i'$ when $S1_i$ is put through all 64 possible values. See also table 11.1.

11.2.4

This skew distribution can be exploited to 'guess' $S1_0'$ for example. Thus the 64 possible inputs with $S1_i' = OC$ will give $S1_0' = E$ with probability $\frac{14}{64}$. Suppose we could arrange R_{n-1}' to give rise to $S1_i' = OC$ (actually we can read $R_{n-1}' = L_n'$) then we could 'guess' $S1_0' = E$ and knowing R_n' calculate the corresponding four bits of L_{n-1}'; and so for S2, S3, . . . S8.

We give an example later of choosing initial P' which will 'push forward' our probabilistic knowledge of the input to the S-boxes SX_i' and their outputs SX_0' over several DES iterations.

11.2.5

Suppose we 'know' (probabilistically perhaps) SX'_i and SX'_0, we can also extract information about the key KX from this knowledge. For example $S1'_i = S1'_E = 34$ gives $S1'_0 = D$ eight times. The values for $S1_i$ and $S1^*_i$ which do this are (in hex)

$S1_i$	$S1^*_i$
6 (1)	32 (C)
10 (3)	24 (E)
16 (C)	22 (1)
1C (0)	28 (D)

When $S1_E = 1$, $S1^*_E = 35$ we need Key K1: 7, 11, 17, 1D, 33, 25, 23, 29 to get these $S1_i S1^*_i$.

There are four (not eight) rows in the tables because we have two cases per row. Example $S1_i = 6$, $S1^*_i = 32$ and $S1_i = 32$, $S1^*_i = 6$. The values in brackets are the S1-Box output $S1_0$, $S1^*_0$ for the inputs, and it will be seem that $S1_0 + S1^*_0 = D = S1'_0$.

Suppose we know $S1_E = 1$ and $S1^*_E = 35$, by reading the input to $f()$; example L_n, L^*_n; and we 'know' $S1'_0 = D$. Then the key K1 must be one of the eight values listed to the right, to XOR with the $S1_E = 1$ and $S1^*_E = 35$ inputs and give the output $S1'_0 = D$. This simple test allows us to select eight out of the possible 64 values for K1 as possibly correct.

11.2.6

Now we do another test, still with $S1'_E = 34$, but with $S1_E = 21$, $S1^*_E = 15$ for example; and suppose we find $S1'_0 = 3$. The table becomes

$S1_i$	$S1^*_i$
1 (0)	35 (3)
2 (4)	36 (7)
15 (C)	21 (5)

When $S1_E = 21$, $S1^*_E = 15$ we need Key K1: 20, 23, 34, 14, 17, 00 to

get these $S1_i S1_i^*$.

The key must be one of the six values below the tables if the inputs are $S1_E = 21$, $S1_E^* = 15$ and the output is $S1_0' = 3$. Comparing this with the first test we see that the most likely values for K1 are 23 and 17.

Continuing testing in this way we should establish a frequency distribution for the values of K1. We pick the most frequently occurring value. Similarly for K2,. . .,K8.

We may now summarise the differential crypt analysis procedure as follows:

1. Make inputs P, P^* giving $P' = P + P^*$ to produce L_{n-1}' with a marked probability,
2. Assume this L_{n-1}' occurs
3. Calculate the S-box outputs SX_0' (taking account of the permutation at the end of $f()$) from $L_{n-1}' + R_n'$ where R_n' is observed.
4. Calculate the S-box inputs SX_E, SX_E^* and $SX_E' = SX_i'$ (taking account of the expansion E at the start of $f()$) from L_n, L_n^* and L_n' observed.
5. Pick off the possible key values K_n (and increment the counts of their occurrence) which could have produced the SX_i, SX_i^* from the SX_E, SX_E^* to give the supposed SX_0'.
6. Repeat points one to five above many times with given P' to build up a frequency distribution of K_n (or at least parts of it such as $K1_n$. Pick the most frequent value for K_n.
7. If all the K1, K2, . . . ,K8 of K_n are not found in this way, repeat points one to six above with another chosen P'.

11.2.7

It remains to show how we can 'push forward' our 'knowledge' of (for example) L_{n-1}'. See figure 11.1. Consider the following example (which ignores the initial permutation in DES which is easily accounted for). Initial differrential input = L_0', $R_0' = 60000000$ (Hex). the expansion function transforms R_0' to $S1_E' = OC$ while $S2_E' = S3_E' = \ldots = S8_E' = 0$.

This gives $S1_0' = E$ with probability $\frac{14}{64}$; and $S2_0' = S3_0' = ... = S8_0' = 0$ since identical inputs with zero difference give identical outputs with zero

difference. The permutation function takes the three non zero bits of E to give a differential output to f() of 00808200 (Hex).

Now if we choose $L'_0 = 00808200$ also we go into the next stage with

$L'_1 = 60000000 = R'_0$ and
$R'_1 = 00000000 = L'_0 + Output\ of\ f()$

In the second stage a zero differential input to $f()$ gives a zero differential output, so we go into the third stage with

$L'_2 = 00000000 = R'_1$ and
$R'_2 = 60000000 = L'_1 + Output\ of\ f()$

This input to the third stage still has probability $\frac{14}{64}$.

Since the third stage input R'_2 is that of the first stage we know it produces an output from $f()$ of 00808200 with probability $\frac{14}{64}$. This gives the inputs to the fourth stage

$L'_3 = 60000000 = R'_2$ and
$R'_3 = 00808200 = L'_2 + Output\ of\ f()$.

and this fourth stage input has probability $(\frac{14}{64})^2$.

Considering that in principle the probability of a particular input to any stage is 2^{-64}, we have produced a very marked probable input to the fourth stage giving us a very high confidence when we apply it to find k_4.

This prodecure can be carried forward for many more rounds. Each time our confidence about the value L'_{n-1} decreases, and more tests are necessary. Intuitively we can say that if the probability of such a desired differential input to the final round is p ($\gg 2^{-64}$ for a 64-bit algorithm) then we require at least $n = \frac{1}{p}$ tests (with varying differential initial inputs designed to produce the desired differential input to the final round) to be successful. That is, to produce a sufficiently marked non-uniformity to be able, for example, to pick off preferred (portions of) subkey values. In the case of DES it can be shown that only when the number of rounds n is of the order of fourteen or so, does the computation involved become so heavy that it would be quicker to do a brute force 'try all the keys' attack. DES is 'secure' with sixteen iterations, but only just.

1 11.3 Conclusion

Des (and other algorithms such as FEAL)is susceptible to differential cryptanalysis because:

- It has strongly non-uniform differential characteristics in the S-boxes
- It is linear apart from the S-boxes
- The keys are entered linearly (XOR)

'Good' algorithms should be non-linear. Rijndael is an example of an algorithm which is designed to use non-linearity to frustrate differential crypt analysis. 'Good' algorithms must also have uniform of near uniform differential characteristics.

Example of Projecting L', R' forward in DES

Figure 11.1

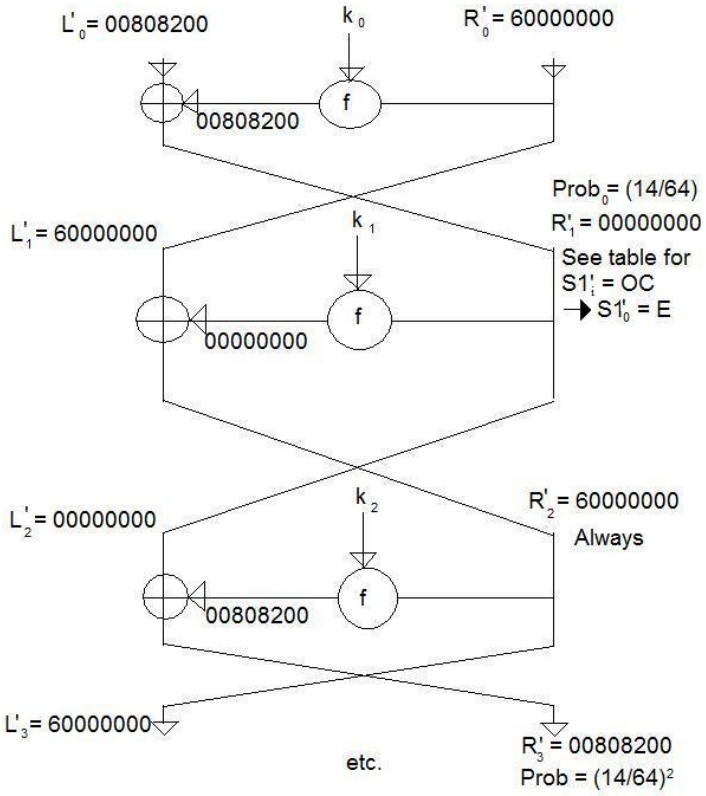


Table 11.1

Some Differential (and linear) Characteristics of DES
S-Box S1 (All numbers in Hex)

$$\begin{aligned} \mathrm{S1}_i &= 6-bit\ Input, \quad \mathrm{S1}_i^* = 6-bit\ input = S1_i + 34 \quad (S1_i' = 34) \\ \mathrm{S1}_0 &= 4-bit\ Output, \quad \mathrm{S1}_0^* = 4-bit\ output, \quad S1_0' = S1_0 + S1_0^* \end{aligned}$$

Figure in brackets in column $S1_i$ is bit 10_{HEX} of input. Figure in brackets in column $S1_0$ is XOR of bits F_{HEX} of Output. (example: $1011 = B \rightarrow 1$ and $1010 = A \rightarrow 0$)

It is only necessary to show half the table $S1$ in this form since

In Differential Cryptology $\mathrm{S1}_i$ and $S1_i^*$ interchange.

In Linear Cryptology RHS of DES Table 4 is "complement" of LHS.

$S1_i$	$S1_i^*$	S_0	$S1_0^*$	$S1_0'$
00 (0)	34	E (1)	9	7
01 (0)	35	0 (0)	3	3
02 (0)	36	4 (1)	7	3
03 (0)	37	F (0)	E	1
04 (0)	30	D (1)	F	2
05 (0)	31	7 (1)	5	2
06 (0)	32	1 (1)	C	D
07 (0)	33	4 (1)	B	F
08 (0)	3C	2 (1)	5	7
09 (0)	3D	E (1)	6	8
0A (0)	3E	F (0)	0	F
0B (0)	3F	2 (1)	D	F
0C (0)	38	B (1)	3	8
0D (0)	39	D (1)	A	7
0E (0)	3A	8 (1)	A	2
0F (0)	3B	1 (1)	0	1
20 (0)	14	4 (1)	6	2
21 (0)	15	F (0)	C	3
22 (0)	16	1 (1)	C	D
23 (0)	17	C (0)	B	7
24 (0)	10	E (1)	3	D
25 (0)	11	8 (1)	A	2
26 (0)	12	8 (1)	A	2
27 (0)	13	2 (1)	6	4
28 (0)	1C	D (1)	0	D
29 (0)	1D	4 (1)	3	7
2A (0)	1E	6 (0)	7	1
2B (0)	1F	9 (0)	8	1
2C (0)	18	2 (1)	5	7
2D (0)	19	1 (1)	9	8
2E (0)	1A	B (1)	9	2
2F (0)	1B	7 (1)	5	2