Brute–Force Search Strategies for Single–Trace and Few–Traces Template Attacks on the DES Round Keys of a Recent Smart Card

Mathias Wagner, Stefan Heyse, Charles Guillemet

mathias.wagner@nxp.com

Abstract. Recently, a new template attack on the DES key scheduling was demonstrated that allows recovery of a sufficiently large portion of the DES key of a widely deployed certified smart card chip using a *single* EM (electromagnetic) trace during the Exploitation Phase. Firstly, in this paper we show how the results can be improved upon when combining them with the analysis of another leakage channel, the total Hamming distance. Remaining rest entropies as low as ≈ 13 bits have been found for some single–trace attacks, meaning that effectively 42 bits of a single–key DES were recovered in a single trace. The nature of single–trace attacks has it that conventional software countermeasures are rendered useless by this attack, and thus the only remaining remedy is a hardware redesign. Secondly, various brute–force search strategies are compared with each other and an extensive analysis of the statistics of the rest entropy is presented. The analysis is also extended to two–key TDES. Moreover, the amount of brute–force effort can be drastically reduced when having more than one trace available for the attack. Already as few as $N = 8$ traces during the Exploitation Phase bring about a reduction of the average brute–force effort of the order of 10 bits for single DES, and 22 bits for two–key TDES. For $N \approx 100$ we achieve an average brute–force effort of less than 50 bits for two–key TDES. Further analysis reveals that this attack is not equally strong for all DES keys, but that quite a number of weaker DES keys exist where the attack is much stronger. Naturally, any assessment of the severity of this attack will have to be made based on the weakest keys. [This last part constitutes an update to a previous version of this paper.]

1 Introduction

In this paper we will present detailed brute–force search strategies and explore a couple of improvements to the original template attack on the DES key schedule as presented in [1, 2], using the very same target smart card as there. In a nutshell, this attack is taking advantage of a weakness in the key schedule of the DES HW coprocessor of that device, where the DES round keys of any two consecutive rounds leak their Hamming distances, with correlation function amplitudes being as large as 70%. This allows to attack this device using a single electromagnetic (EM) trace plus a subsequent brute–force step with a DES cracker or even just a PC. Remaining rest entropies as low as 19 bits have been found in [2] for single–key DES.

The analysis in [1, 2] is based on taking advantage of the mathematical properties of the DES key schedule, leading to the definition of two so–called C rings, which describe the Hamming distance — or $\oplus$ — relationship of key bits in consecutive DES rounds, requiring the creation and handling of 28 overlapping templates of adjustable size for a total of 112 $\oplus$ relationships of key bits. Overlapping templates had to be introduced to keep the size of each template manageable, but still cover all key bits. This led to some challenges how to put all results back together. Please refer to [2] for more details. Here it suffices to say that one of the two C rings corresponds to the so–called C Register of the DES key schedule, whilst the other C ring relates to the D Register, and that this factorisation into two groups helps a lot in the brute–force key search, as outlined in Sec. 2.

In Sec. 3 we put forward an enhanced exploitation of this key schedule leakage by amending the original brute–force search strategy for 28 overlapping templates along the two C rings with the results obtained by additionally exploiting the leakage of the total Hamming distance of the two C rings taken together (which fits into a single template). With total Hamming distance we refer to the sum — over all rounds — of the Hamming distances between two consecutive round keys. On average, this approach yields additionally ≈ 2.5 bits per DES key when using a single trace in the Exploitation Phase, or ≈ 5 bits for a two–key TDES.

Thirdly, in Sec. 4 we analyse the statistics of various brute–force search strategies in more detail, and as a function of the template size, and compare those with the predictions made in [2].

In Sec. 5 — which is a Section added in an update to this paper — we provide strong evidence that this attack does not work equally well for all DES keys, but that there are, in fact, a large number of much weaker keys, of the order of a few % of all keys, where the attack works particularly well. This is an important detail when performing a risk assessment and studying the worst case.

Finally, in Sec. 6 we explore the conventional trade–off between the brute–force effort required, and the number of traces N used in the Exploitation Phase. Obviously, the single–trace attack studied in [2] and in this paper until Sec. 5 corresponds to $N = 1$. We find that already a few dozen traces reduce the brute–force effort drastically. Since SW countermeasures against fault attacks often require repeated calls to the HW DES engine using the same key, it is not unreasonable to have even in a nominally single–trace scenario in fact a couple of traces with the same DES key available for the attack.

2 Brute-Force Algorithm

In [2] some concrete brute–force results were shown for the C and D Register of the DES algorithm, but always separately. The statistics of the combined brute–

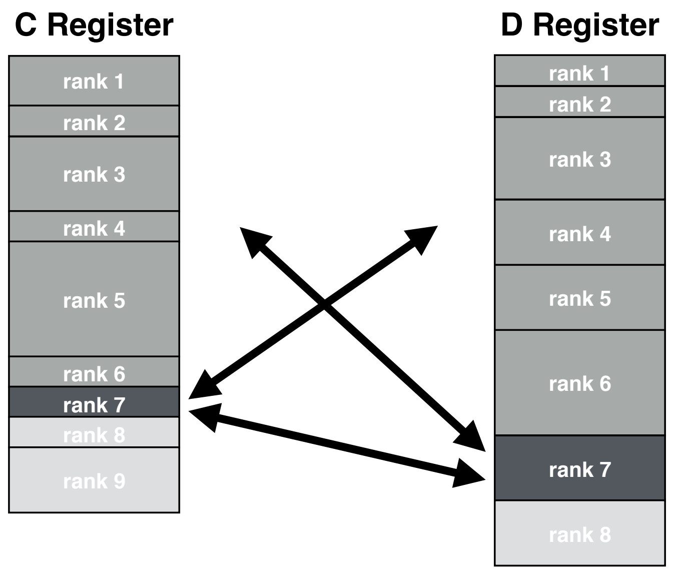


Fig. 1. Search strategy across the C and D Register of the DES key schedule. Both lists are ranked according to some criterion, resulting in classes of 27–bit sub–keys having the same rank (e.g., because they have the same $r_{\max}$, or the same r_{average}). The size of each class represents the number of 27–bit sub–keys in it. Now, suppose all keys have been searched up to rank 6 inclusive already. Then in order to count all keys up to and including rank 7, all the C/D Register combinations indicated by the arrows need to be counted, as spelled out more concretely in Eq. (4).

force attacks on both registers together has only been estimated using Eq. (10) in [2], and the purpose of this Section is to address this point.

As a recap from [2], the two C rings resulting from the DES key schedule look as follows,

$$\begin{tikzcd}[column sep=tiny]
& 7 \arrow[rr]\arrow[dr] & & 21 \arrow[rr]\arrow[dr] & & 35 \arrow[rr]\arrow[dr] & & 49 \arrow[rr]\arrow[dr] & & 38 \arrow[rr]\arrow[dr] & & 52 \arrow[rr]\arrow[dr] & & 9 \arrow[rr]\arrow[dr] & & 23 \arrow[rr]\arrow[dr] & & 37 \arrow[rr]\arrow[dr] & & 51 \arrow[rr]\arrow[dr] & & 8 \arrow[rr]\arrow[dr] & & 22 \arrow[rr]\arrow[dr] & & 36 \arrow[rr]\arrow[dr] & & 50 \arrow[rr]\arrow[dr] & & 7 \\
0 \arrow[rr]\arrow[ur] & & 14 \arrow[rr]\arrow[ur] & & 28 \arrow[rr]\arrow[ur] & & 42 \arrow[rr]\arrow[ur] & & 31 \arrow[rr]\arrow[ur] & & 45 \arrow[rr]\arrow[ur] & & 2 \arrow[rr]\arrow[ur] & & 16 \arrow[rr]\arrow[ur] & & 30 \arrow[rr]\arrow[ur] & & 44 \arrow[rr]\arrow[ur] & & 1 \arrow[rr]\arrow[ur] & & 15 \arrow[rr]\arrow[ur] & & 29 \arrow[rr]\arrow[ur] & & 43 \arrow[rr]\arrow[ur] & & 0 \arrow[ur] & \\
& 10 \arrow[rr]\arrow[dr] & & 24 \arrow[rr]\arrow[dr] & & 11 \arrow[rr]\arrow[dr] & & 25 \arrow[rr]\arrow[dr] & & 39 \arrow[rr]\arrow[dr] & & 53 \arrow[rr]\arrow[dr] & & 12 \arrow[rr]\arrow[dr] & & 26 \arrow[rr]\arrow[dr] & & 40 \arrow[rr]\arrow[dr] & & 54 \arrow[rr]\arrow[dr] & & 13 \arrow[rr]\arrow[dr] & & 27 \arrow[rr]\arrow[dr] & & 41 \arrow[rr]\arrow[dr] & & 55 \arrow[rr]\arrow[dr] & & 10 \\
3 \arrow[rr]\arrow[ur] & & 17 \arrow[rr]\arrow[ur] & & 4 \arrow[rr]\arrow[ur] & & 18 \arrow[rr]\arrow[ur] & & 32 \arrow[rr]\arrow[ur] & & 46 \arrow[rr]\arrow[ur] & & 5 \arrow[rr]\arrow[ur] & & 19 \arrow[rr]\arrow[ur] & & 33 \arrow[rr]\arrow[ur] & & 47 \arrow[rr]\arrow[ur] & & 6 \arrow[rr]\arrow[ur] & & 20 \arrow[rr]\arrow[ur] & & 34 \arrow[rr]\arrow[ur] & & 48 \arrow[rr]\arrow[ur] & & 3 \arrow[ur] &
\end{tikzcd} \tag{1}$$

where the arrows denote an $\oplus$ relation between two key bits (of two consecutive rounds of the DES key schedule).[1] The first of these two rings maps to the C Register of the DES key schedule, the second to the D Register.

[1] The numbering of these key bits is such that we count them as ordered in the original DES key, but ignore parity bits. Note that the C rings do not tell between which two rounds the $\oplus$ occurred, nor how often.

For these two disjoint C rings we construct the templates maximally overlapping. So, if the first template is, e.g.,

$$\begin{tikzcd}
 & 7 \arrow[rr] \arrow[dr] & & 21 \arrow[rr] \arrow[dr] & & 35 \arrow[rr] \arrow[dr] & & 49 \\
0 \arrow[rr] \arrow[ur] & & 14 \arrow[rr] \arrow[ur] & & 28 \arrow[rr] \arrow[ur] & & 42 &
\end{tikzcd} \tag{2}$$

then the next template "to its right" is

$$\begin{tikzcd}
 & 21 \arrow[rr] \arrow[dr] & & 35 \arrow[rr] \arrow[dr] & & 49 \arrow[rr] \arrow[dr] & & 38 \\
14 \arrow[rr] \arrow[ur] & & 28 \arrow[rr] \arrow[ur] & & 42 \arrow[rr] \arrow[ur] & & 31 &
\end{tikzcd} \tag{3}$$

and so on along the ring, until the loop is closed. Incidentally, the templates shown here are 7–bit templates, but clearly they can be made smaller or larger by pruning or extending them to the right. However, even the largest practically feasible template will not be large enough to encompass an entire C ring — in particular, a template targeting a 27–bit value would lead to 2^{27} classes,[2] which is not practical — and because of this we changed tactics and decided to divide the C rings into overlapping templates. In [2] it was extensively discussed how strong neighbouring ranking lists correlate with each other due to this overlap.

The construction we chose results in 14 overlapping templates for each C ring, each having its own ranking list of pattern–template matching candidates created in the Exploitation Phase.

The task then is how to best combine these 14 lists to a single ranking list, or a single key enumeration scheme for the Register at hand, C or D. In a first step we create an unordered list of all possible combinations of ranks across all 14 lists. Obviously, because of overlapping templates and the ring structure imposing some boundary conditions, not all ranking combinations across those 14 lists are allowed. It turns out that creating such a single list is pretty straight–forward to do, though, with the total number of possible entries in this key enumeration list — for one register — being 2^{27}. This makes a lot of sense, since the C and the D Register control 28 sub–key bits each, and one bit is consumed by the $\oplus$ operation already.

The next step is to decide on a suitable criterion to order these two lists of 27–bit sub–keys — the key enumeration scheme. Two approaches were already put forward in [2]: The first one is to order by the Maximum Ranking found in the respective C ring, $r_{\max} = \max_0^{13} r_i$ for the C Register, and $r_{\max} = \max_{14}^{27} r_i$ for the D Register, where r_i is the ranking in the ranking list of template i pertaining to the 27–bit sub–key at hand. The second approach is to order by the Average Ranking, $r_{\text{average}} = 1/14 \sum_0^{13} r_i$, or $r_{\text{average}} = 1/14 \sum_{14}^{27} r_i$. Since r_i is always an integer, the number of possible values for $r_{\max}$ and r_{average} is much

[2] Remember, we are targeting the Hamming distance and not the Hamming weight itself. Hence, it is 27 bits only, not 28.

smaller than 2^{27}, and hence there will, in fact, be many 27–bit sub–keys having the same values of $r_{\max}$ or r_{average} in these list. Because of this collision, the effort and overhead required to sort the 2^{27} entries of each list is much smaller than, e.g., a classical QuickSort algorithm [3] would require, as there is no need of any ordering within a given rank. With ranks we denote the groups of list entries having the same order parameter (e.g., $r_{\max}$ or r_{average}).

Having ordered the lists for the C and D Register separately, the final step is to combine these two lists into one. In Fig. 1 we have depicted the corresponding search strategy across both lists as deployed in this paper. In essence, it is simply repeatedly performing and doing all book keeping in the iterative step

$$\begin{aligned} N_j &= N_j^C \times N_j^D \\ &\to N_{j+1} = (N_j^C + n_{j+1}^C) \times (N_j^D + n_{j+1}^D) \\ &\qquad\quad = N_j^C \times N_j^D \\ &\qquad\quad + N_j^C \times n_{j+1}^D + n_{j+1}^C \times N_j^D \\ &\qquad\quad + n_{j+1}^C \times n_{j+1}^D \quad . \end{aligned} \tag{4}$$

Here N_j^C is the number of 27–bit sub–keys in the C Register tried already up to rank j, and n_{j+1}^C is the number of 27–bit sub–keys in the C Register to be tried for the next rank $j+1$. A similar notion holds for the D Register, and for both registers together. Such an iterative approach can be efficiently implemented as a fast search strategy with very little overhead.

Since in each rank j all keys have by definition the same probability, the correct key may be found at the beginning of a rank, or its end. The following results all take the linear average of these two extreme cases and then apply $\log_2$ to convert this average to an average entropy.

In Sections 4 and 6 we apply this brute–force search strategy using templates of sizes 5, 7, and 9 bits as defined in [2] and above, and using various schemes for ordering the search lists. By applying these brute–force searches to many single traces individually, say 32 k or more traces, we are able to present meaningful statistical results.

But first, we show how to exploit additional total Hamming distance leakage for the target device at hand to make the attack more effective.

3 Hamming Weight Leakage

For reference, a typical EM measurement of a DES calculation of the target device is shown in Fig. 2. In fact, there are four DES calculations seen, all using the same key — possibly because of countermeasures against fault attacks. As described in detail in [2] some careful signal processing is required to arrive at a set of well–aligned traces. In order to be able to create templates and later on have statistical results over many single–trace attacks, this first set of traces uses randomly chosen DES keys for the Profiling as well as the Exploitation Phase of

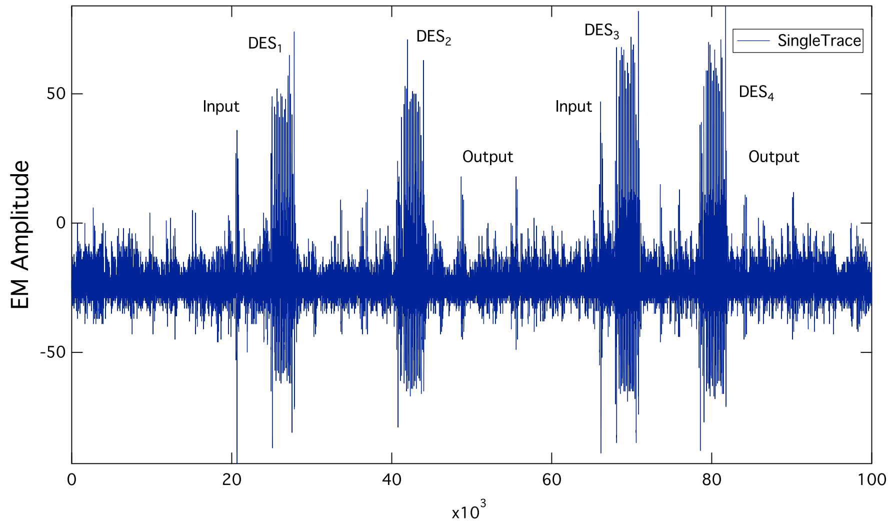


Fig. 2. A typical single EM (electromagnetic) trace of the TOE showing 4 calls to the DES HW engine. It was obtained by placing a Langer EM probe on top of what had previously been identified as a DES coprocessor hard macro in the chip layout. Sampling rate: 5 GS/s.

the template attack. A second set of traces with a single, fixed key will be used in Sec. 6.

Now, what is new in this Section of the paper is that we perform a further signal processing step in that we add up the 16 portions of every trace corresponding to the 16 rounds of each DES call to yield one average round per DES call (backfolding step). The associated results for the correlation function as well as χ^2 are shown in Figs. 3 and 4, respectively. The select function chosen here is based on Table 5 of [2], where the contributions of all $\oplus$ is summed up over all rounds to give a total Hamming distance of the two C rings lumped together. Another way of looking at this is to realise that the total Hamming distance is the Hamming distance between any two rounds, summed over all rounds. It should be noted here that between some key bits there exist many $\oplus$ relationships, whilst for others there are only very few. This depends on whether these $\oplus$ relations belong to the so–called A or the B rings. Again, please refer to [2] for details.

The total Hamming distance counts each $\oplus$ between any pair of key bits, regardless of which DES round it originally occurred in, and hence because of these weights there are effectively many more possible values to the total Hamming distance than one might naively guess at first from a 56–bit DES key. In fact, by way of sheer counting, it turns out there are 612 possible values, with the largest one being 648.[3] Because of this, the maximally possible total

[3] Reference [2] provided a slightly wrong number here.

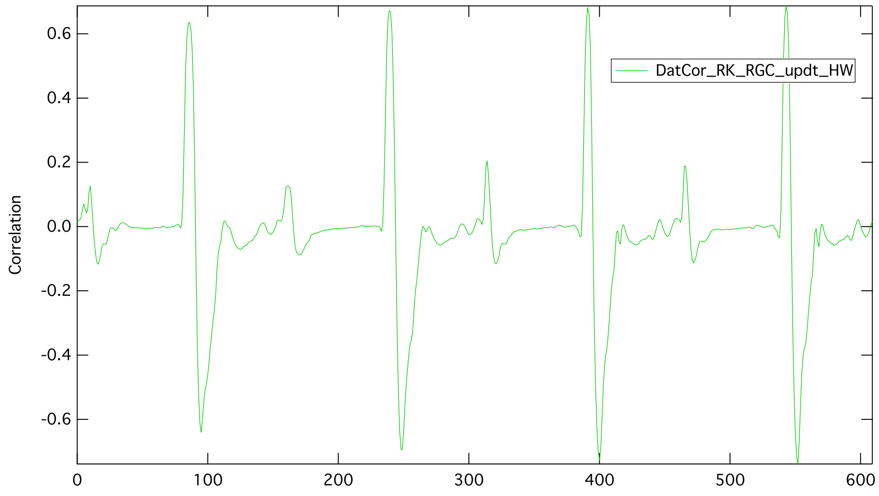


Fig. 3. Correlation for total Hamming distance using backfolded traces.

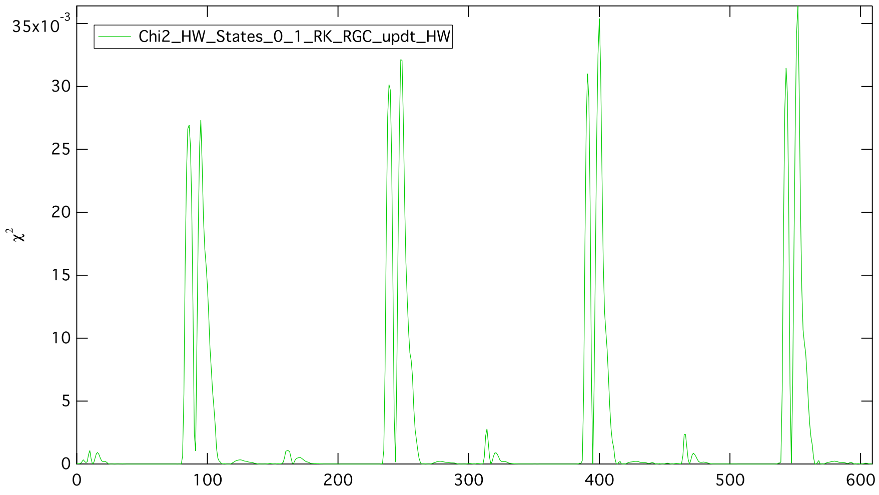


Fig. 4. χ^2 for total Hamming distance using backfolded traces.

Hamming distance leakage is much larger than one might first think, and can be as much as 6–7 bits for perfect leakage.

As before, we perform a template attack, but now targeting the total Hamming distance of both C rings, using the first 4.75 M traces of the first set for creating the templates in the Profiling Phase, and then targeting a few 10 k traces one by one, i.e., as single–trace attacks, in the Exploitation Phase. The

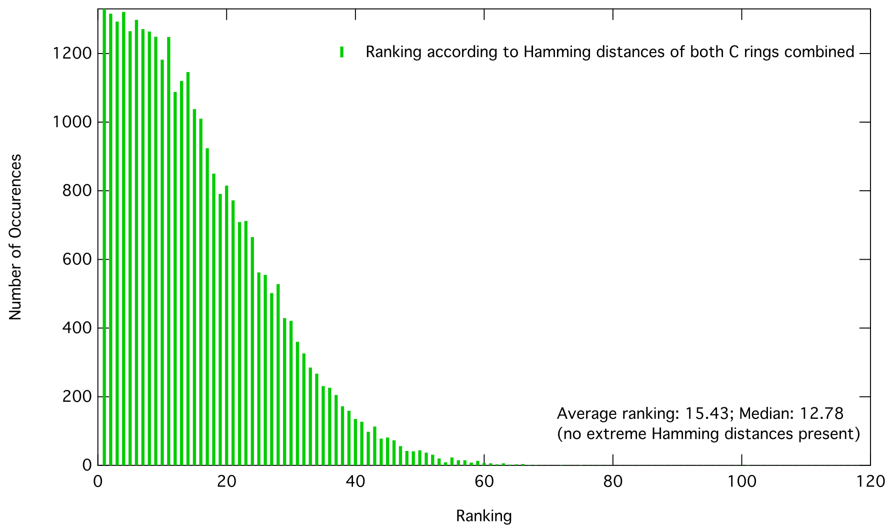


Fig. 5. Ranking for total Hamming distance using backfolded traces.

number of POIs chosen is 352 and has been derived based on setting a threshold for the χ^2 function.

Fig. 5 shows the statistical ranking results for 32 k single traces. Out of the 511 possible ranking values, the majority of traces in the Exploitation Phase show rankings better than 40, with the average ranking being only 15.43.

Based on an implementation of a brute–force search for the correct DES key as put forward in Sec. 2, Table 1 shows the effect on the rest entropy for some selected single traces when additional restrictions are applied to the search candidates such as, e.g., a matching total Hamming distance that is required. The idea is as follows: No matter how the brute–force search algorithm looks like precisely, it will be a search over the combined ranking lists for the C and the D Registers as illustrated in Fig. 1, where for each candidate in the C Register list, a range of entries in the D Register list will have to be tried out. Now, if the total Hamming distance of the correct key was known completely, then for each candidate in the C Register list, it is clear how much of that Hamming distance "is left" for the candidate in the D Register, simply because the total Hamming distances of both candidates added together must equal the total Hamming distance of the correct key. The corresponding results are found in Table 1 in the columns labelled "Exact HW". More generally, if the total Hamming distance is not precisely known, as is evident from Fig. 5, then the condition for the Hamming–distance matching needs to be somewhat relaxed. Given that the average ranking of Fig. 5 is 15.43, we have chosen to include all candidates in the D Register with total Hamming distance "errors" up to ± 7.[4] This approach

[4] The idea here is simply that with such an error of ± 7 a range of 15 different possible total Hamming distances are allowed, which is roughly equal to the average ranking in Fig. 5 of 15.43.

Table 1. Comparison of the rest entropy $E^C_{r_{\max}}$ as estimated by Eq. (10) of [2] for selected traces, with actual brute–force key searches based on minimal Average Ranking r_{average} ($:E_{r_{\text{average}}}$) or minimal Maximum Ranking $r_{\max}$ ($:E_{r_{\max}}$).

Trace ID	T'pl. Size	C– and D–Register			Exact HW		Exact HW ±7	
	[bits]	$E^C_{r_{\max}}$	$E_{r_{\text{average}}}$	$E_{r_{\max}}$	$E_{r_{\text{average}}}$	$E_{r_{\max}}$	$E_{r_{\text{average}}}$	$E_{r_{\max}}$
4750004	5	**58.00**	**52.39**	**55.29**	**45.86**	**48.46**	**49.74**	**52.36**
4750005	5	**58.00**	**51.08**	**55.26**	**43.67**	**47.17**	**47.59**	**51.08**
4750007	5	**58.00**	**50.57**	**55.24**	**44.10**	**48.26**	**47.97**	**52.18**
4750002	5	**57.01**	**54.18**	**54.16**	**47.54**	**47.51**	**51.43**	**51.40**
4750003	5	**57.01**	**46.18**	**54.14**	**39.73**	**47.56**	**43.61**	**51.46**
4750272	5	**57.01**	**52.96**	**54.21**	**46.38**	**47.62**	**50.27**	**51.52**
4750009	5	**56.00**	**48.76**	**53.02**	**42.23**	**46.40**	**46.09**	**50.29**
4750001	5	**54.96**	**44.93**	**52.12**	**38.36**	**45.55**	**42.25**	**49.43**
4750008	5	**53.66**	**44.20**	**49.18**	**37.65**	**42.30**	**41.51**	**46.17**
4750006	5	**51.66**	**53.25**	**49.01**	**46.32**	**42.13**	**50.23**	**46.03**
4750032	5	**50.50**	**41.32**	**48.69**	**34.85**	**41.39**	**38.70**	**45.26**
4753937	5	**49.31**	**31.81**	**46.74**	**25.40**	**39.24**	**29.28**	**43.15**
4750068	5	**48.08**	**36.64**	**46.29**	**30.30**	**39.64**	**34.15**	**43.52**
4750000	5	**46.81**	**46.78**	**45.05**	**40.30**	**38.37**	**44.19**	**42.27**
4750010	5	**45.50**	**35.02**	**43.79**	**28.75**	**37.27**	**32.57**	**41.12**
4750011	5	**44.15**	**48.87**	**42.86**	**42.36**	**36.42**	**46.25**	**40.30**
4750798	5	**39.79**	**31.95**	**38.92**	**25.53**	**32.33**	**29.42**	**36.20**
4756552	5	**39.79**	**31.75**	**38.24**	**25.37**	**31.69**	**29.26**	**35.58**
4763629	5	**38.22**	**29.11**	**37.27**	**23.00**	**29.83**	**26.77**	**33.70**
4754072	5	**36.60**	**28.33**	**36.95**	**21.13**	**30.70**	**25.01**	**34.41**
4750232	5	**34.90**	**31.92**	**34.52**	**25.56**	**28.10**	**29.34**	**32.00**
4760532	5	**34.90**	**29.37**	**34.53**	**23.10**	**28.48**	**26.81**	**32.24**
4750606	5	**33.13**	**31.33**	**33.73**	**25.01**	**27.52**	**28.92**	**31.41**
4780499	5	**29.34**	**19.59**	**28.90**	**13.40**	**22.87**	**17.49**	**26.69**
4777975	5	**27.31**	**27.92**	**27.32**	**20.88**	**20.73**	**24.75**	**24.53**
4763788	5	**27.31**	**26.84**	**27.77**	**20.08**	**21.42**	**24.00**	**25.32**
4781560	5	**25.16**	**24.89**	**24.63**	**18.11**	**18.16**	**22.18**	**22.11**
4763782	5	**25.16**	**24.56**	**23.75**	**18.19**	**17.53**	**22.33**	**21.66**
4756577	5	**22.89**	**25.95**	**22.58**	**19.83**	**16.42**	**23.72**	**20.31**
4935963	5	**22.89**	**17.44**	**22.80**	**8.93**	**16.04**	**13.27**	**20.08**
4757225	5	**20.49**	**21.33**	**19.36**	**15.32**	**13.07**	**19.11**	**17.03**

is not completely correct, the search strategy needs to be a bit more sophisticated than that, but this reasoning will give an idea of where we are heading when including the total Hamming distance leakage as an additional boundary condition in any brute–force search.

From Table 1 we find for, e.g., TraceID 4935963, that the formula given in Eq. (10) of [2] predicts a rest entropy of 22.89 bits, whilst a brute–force search based on minimal Maximum Ranking yields 22.80 bits. On the other hand, when searching based on minimal Average Ranking, one finds the DES key with a rest entropy of only 17.44 bits. When allowing for a total Hamming distance error of ± 7, these rest entropies get further reduced to 20.08 and 13.27 bits, respectively. A more detailed statistical analysis shows that the average gain when using the total Hamming distance restriction in the brute–force search — with a maximal error of ± 7 — is about 2.5 bits per DES key. This result does not appear to depend on the size of the templates used. In the remainder of this paper we will use this rough figure to estimate the effect of the total Hamming distance leakage on the statistics of rest entropies.

So far we have looked at the total Hamming distance of the two C rings. It should be possible to improve these results somewhat by looking at the total Hamming distances of the A and B rings separately, since these rings map to different rounds of the DES key schedule, and hence can be distinguished along the timeline. Consequently, such an approach should reveal more information. However, we have not analysed this route yet. Instead, in an updated version of this paper we followed the approach described in Sec. 5 using 15–tuples, which should reveal even more information.

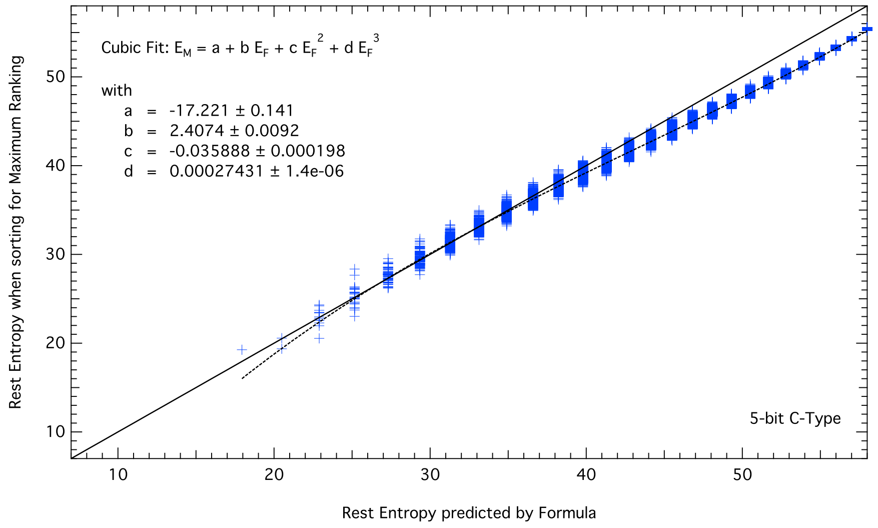
4 Statistics of Brute–Force Searches

In this Section we analyse the statistics of the brute–force effort based on the algorithm(s) shown in Sec. 2, on the basis of up to 297 k individual attacks, for templates of increasing sizes. We will analyse various key enumeration or search strategies in more detail. For simplicity, we will not include the total Hamming distance leakage discussed in Sec. 3 as an additional restriction to the brute–force search, but rather note its additional contribution (i.e., 2.5 bits for single DES, and 5 bits for two–key TDES) where appropriate. Unless stated otherwise, all results pertain to single–key DES. Estimates for two–key TDES will be given at the end of each n–bit template analysis.

4.1 Brute-Force Results for Random Keys (5–Bit Templates)

Firstly, let's have a look at the accuracy of the formula given as Eq. (10) in [2] for predicting the rest entropy based on knowing $r_{\max}$ only. Fig. 6 shows in its top part the rest entropy when using the Maximum Ranking for sorting the C and D Register lists of possible key candidates, as a function of the rest entropy predicted by said Eq. (10). Thus, if Eq. (10) were strictly correct, all results would be on the diagonal. These results were obtained using 297 k single traces and 5–bit templates. Clearly, the formula is fairly accurate for smaller rest entropies, but for larger rest entropies, the formula — on average — is too conservative and actually over–estimates the remaining brute–force effort. Consequently, for 5–bit templates the actual results for the rest entropy should be better than those predicted in [2].

Secondly, in the bottom part of Fig. 6 we see that a search strategy based on ordering the lists by Average Ranking is on average substantially better than ordering by Maximum Ranking — with the possible exception being for very



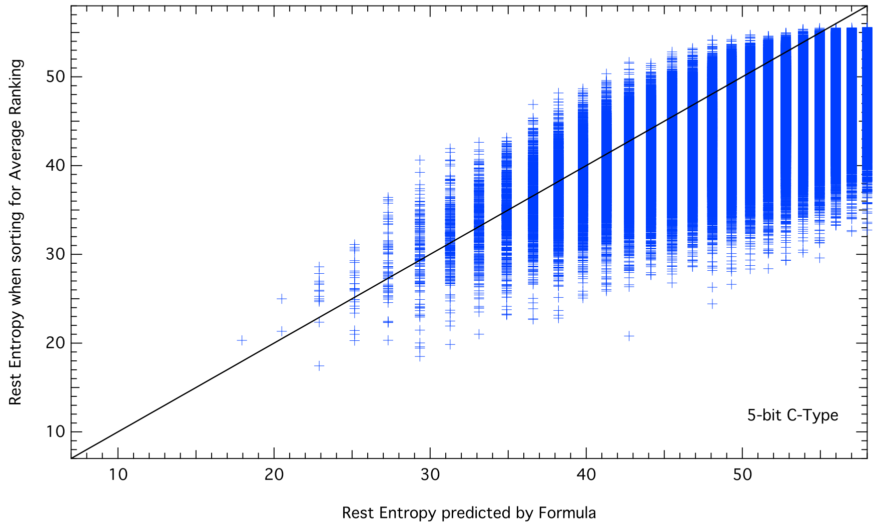


Fig. 6. The top graph shows the rest entropy found when sorting the search lists for Maximum Ranking versus the rest entropy predicted by formula Eq. (10) of [2]. The bottom graph shows the same but instead sorting the lists for Average Ranking. Clearly, the latter is on average a better search strategy for all but perhaps the smallest rest entropies.

small values for the rest entropy, i.e., for very leaky traces. Fig. 7 shows how Average Ranking fares against Maximum Ranking. Again, it is obvious that searching by Average Ranking is generally found to be better than searching by Maximum Ranking.

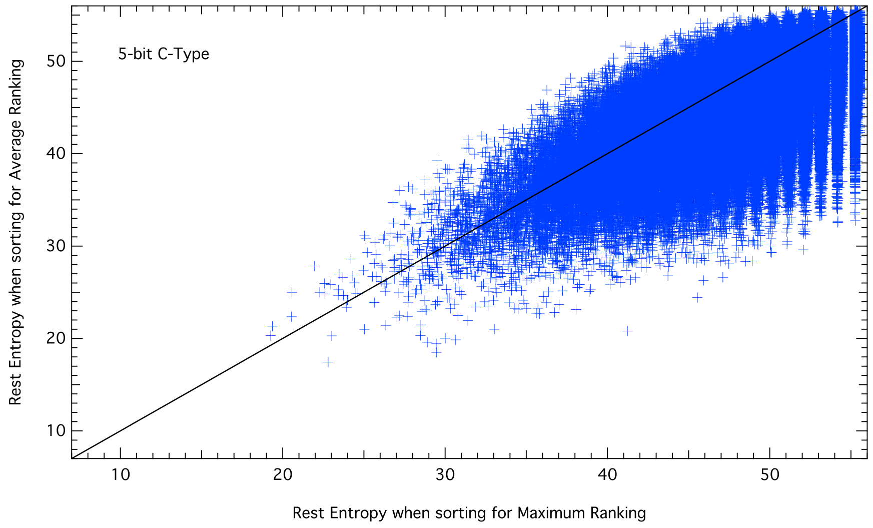
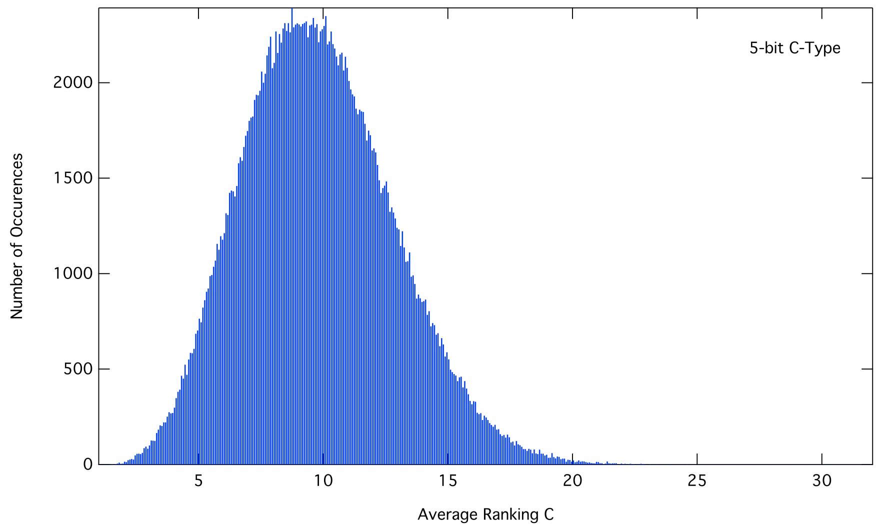


Fig. 7. Same as Fig. 6, but now Average Ranking versus Maximum Ranking.

In Fig. 8 we have plotted the distributions of the Average Ranking of the correct key — separately for the C and D Register — with their averages being 9.86 and 7.78, respectively, and their standard deviations being 3.01 and 2.67. Clearly, the D Register performs better than the C Register, and both are substantially better than random results, i.e. 16.5. The difference of their means is ≈ 2.05488. The fact that the C and D Register distributions differ will later be used to improve the search strategies somewhat. For reference, in Fig. 9 we also show the equivalent results obtained for Maximum Ranking.[5]

In order to try to answer the question whether the value of the secret key may affect the rest entropy in the brute–force search, we have analysed the Hamming distance of the combined C rings as a function of the rest entropy when sorting for (Differential — as explained further below) Average Ranking, as shown in Fig. 10. The correlation between these two variables is very weak, of the order of -1.8%. We suspect that measurement noise and mis-alignment will in parts be responsible for the distribution seen, and improving the quality of the measurement and the subsequent alignment steps should yield improved results. However, further analysis in Sec. 5 reveals that some DES keys are much

[5] It may be tempting to conclude from Fig. 8 that an even better search strategy may be obtained by starting the search at the respective peaks of these averages. However, this would be wrong, since as far as searching is concerned, the relevant probability is not the one shown, but rather Fig. 8 needs to be normalised by the number of 27–bit sub–keys that have the same given Average Ranking to begin with. Clearly, this normalisation curve will be strongly peaked at half the maximally possible ranking. It turns out that after such a normalisation the better search strategy is still to start with the smallest possible average values and not the peak values, as expected.



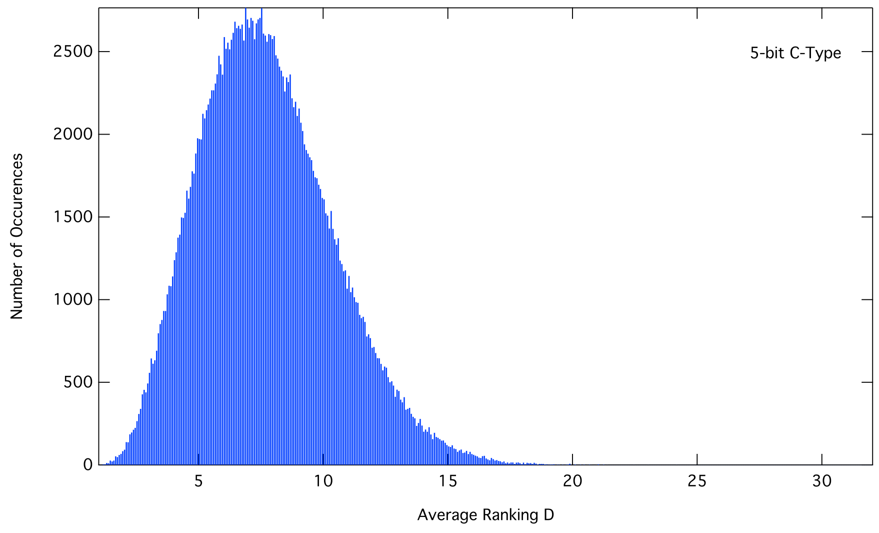
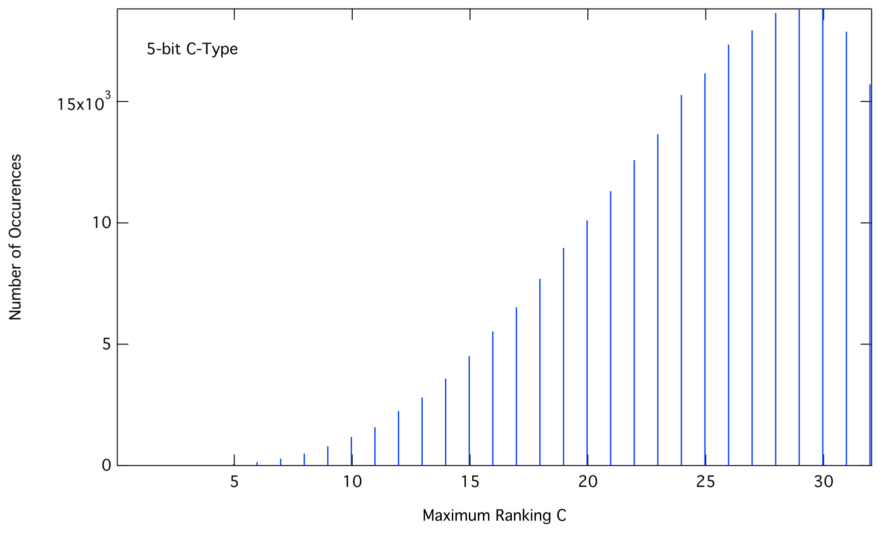


Fig. 8. Distributions found for the Average Ranking of the correct key for C and D Registers.

more susceptible to this attack than others. This is further confirmed in Sec. 6 where the statistics of fixed key attacks will be analysed.

In Fig. 11 we show the resulting distributions of the rest entropy for search strategies based on Maximum Ranking (top), as well as Average Ranking (bottom). Here as well as in the following of this Section, for all results based on 5–bit templates, some 297 k single traces were involved in the analysis. It turns out that the brute–force search based on Average Ranking is substantially better than the one based on Maximum Ranking, the average rest entropy being



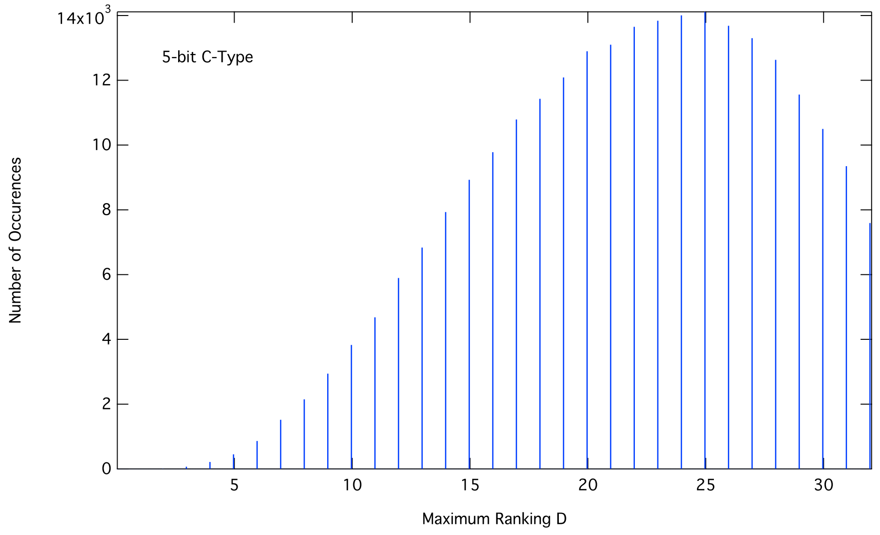


Fig. 9. Distributions found for the Maximum Ranking of the correct key for C and D Registers.

46.16 bits in the former, and 49.72 bits in the latter case, whilst Eq. (10) of [2] yields 52.11 bits. The distributions have long tails towards smaller rest entropies, meaning that statistically some brute–force attacks involve very little effort.

These results can be improved upon a little by taking advantage of the fact that the D Register is performing better on average than the C Register, as evidenced in Fig. 8. Hence, it makes sense to adjust the relative search depth δr

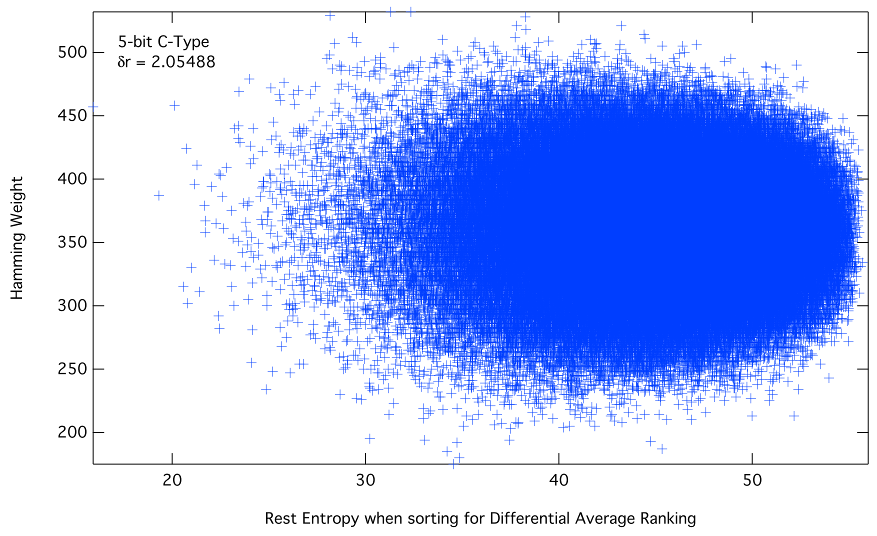


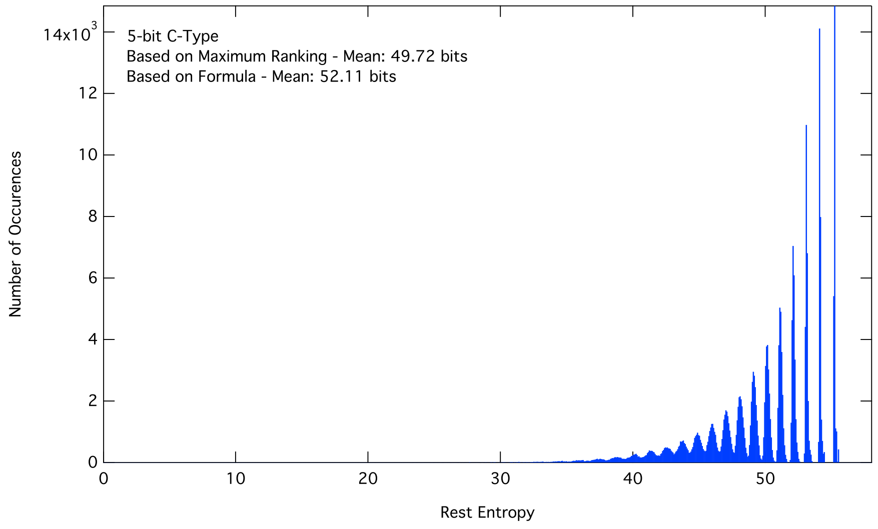
Fig. 10. The Hamming distance versus Differential Average Ranking. The correlation between the two is very weak — of the order of -1.8%, meaning that large Hamming distances will leak only very slightly more than smaller ones.

between the C and D Registers.[6] For analysis purposes, we have picked heuristically a few constant values that scale with the difference seen between the average rankings of the C and D register, $\delta r_{CD} = 2.05488$, such as $\delta r = 0.25 \times \delta r_{CD}$, $0.5 \times \delta r_{CD}$, δr_{CD}, and $1.25 \times \delta r_{CD}$ with the latter two choices resulting in Fig. 12. It thus appears that the average rest entropy improves by about 0.5 bits compared to Fig. 11 (bottom) when accounting for the different quality of the C and D Register results in this somewhat crude way. There is still some room for further improvement here by choosing even slightly larger values of δr, but it seems marginal and we did not follow up on this.

In Fig. 13 we have checked whether the Average Rankings of the C and the D Register correlate — it turns out they does so, but only with $\approx -4.1\%$. Hence, albeit this correlation does imply that a more sophisticated choice of δr would be beneficial for the attack, this is not an avenue to improve the search strategy greatly in this case.

In Figs. 14 to 17 we have plotted the estimated 2–key TDES distributions of the rest entropies, again for various techniques of sorting the lists of the C and the D Register. These estimates have been made by picking any two traces of the single–key DES, taking the highest rest entropy of the two, and then simply doubling it. This is then done for all possible combinations of single–key DES traces. However, since the characteristics of a two–key TDES of the form $\text{DES}(k_1)\text{DES}^{-1}(k_2)\text{DES}(k_1)$ is that the outer key k_1 is used twice, as it were,

[6] In Fig. 1 this simply means that there is a constant difference in the rank j for the C Register and j' for the D Register. E.g., for $\delta r = 2$ rank 7 in C will be matched with rank 5 in D Register, and so on.



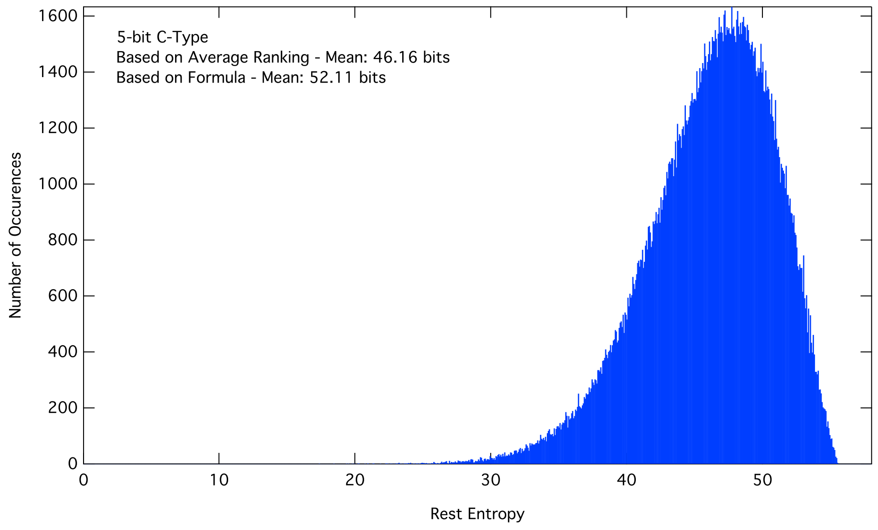
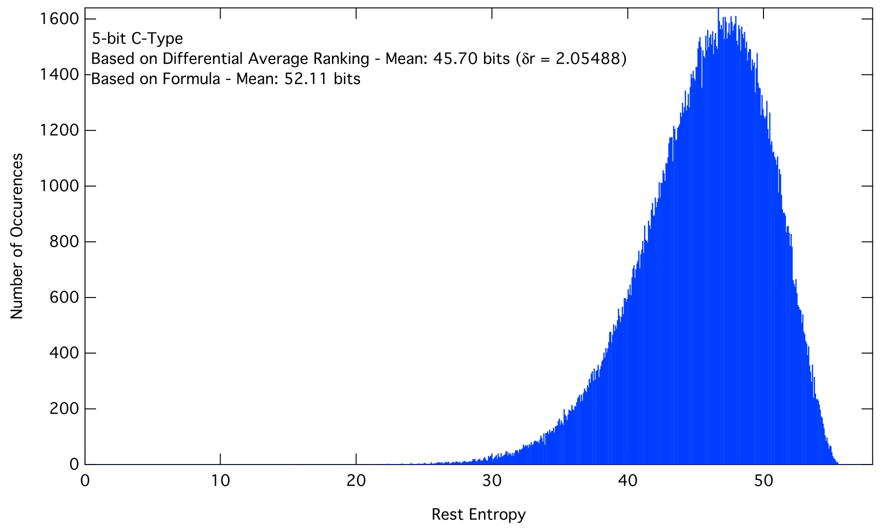


Fig. 11. Distributions for the rest entropy when sorting for Maximum Ranking (top) or Average Ranking (bottom).

there is in fact more leakage present than we have implicitly assumed when extrapolating from a single–key DES to a two–key TDES. The two–key TDES is in fact a mixture of a single–trace and a two–trace attack. This should improve the real attack somewhat compared to the numbers we present in this Section.[7]

[7] In Sec. 6.2 below we study for a given fixed key how the statistics improves when using $N > 1$ traces in the Exploitation Phase. When comparing the $N = 1$ and $N = 2$ case (so Figs. 44 and 47) we find an improvement of about 3.4 bits in the average rest entropy for a single DES key (which goes down to about 1.6 bits when



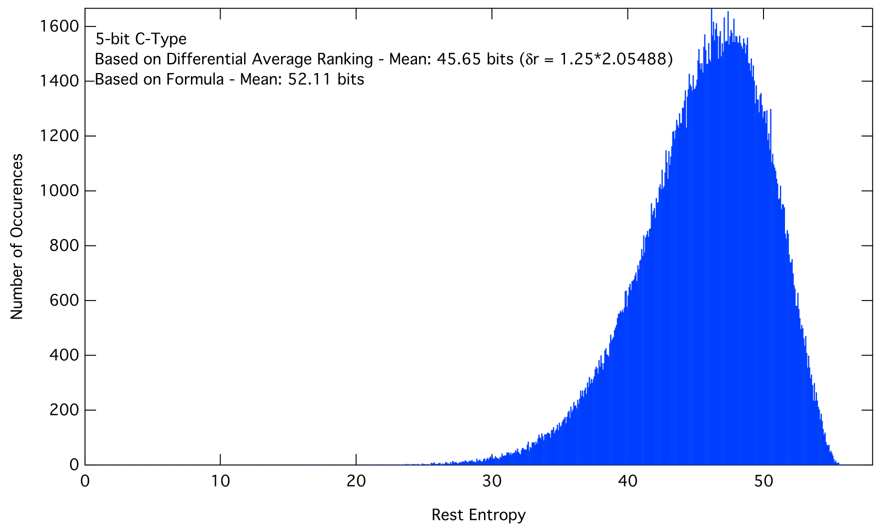


Fig. 12. Distributions for the rest entropy when sorting for Differential Average Ranking with $\delta r = 2.05488$ (top), and $\delta r = 1.25 \times 2.05488$ (bottom).

As expected from the single–key DES results, from all variants analysed, the results for search strategies based on Average Rankings with $\delta r = 1.25 \times 2.05488$ perform best, the average rest entropy for a two–key TDES being 96.48 bits. More importantly, though, also these distributions show a large tail towards

comparing $N = 32$ and $N = 64$). This is then also the improvement we would expect for a two–key TDES when accounting for the fact that the outer DES keys are effectively two traces worth of information leakage.

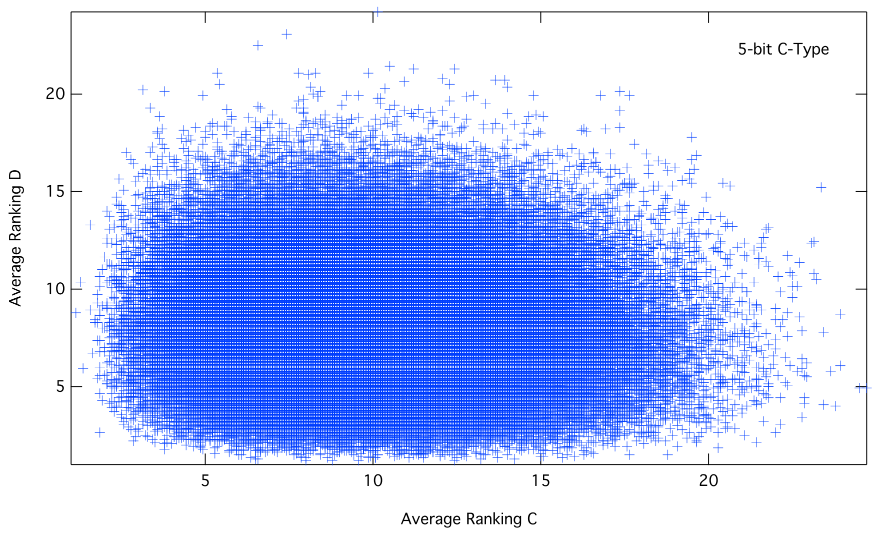


Fig. 13. The correlation between the Average Rankings of the C and the D Register is rather weak, but noticeable at about −4.1%.

smaller rest entropies, with some 5.3% of all traces having a rest entropy of 85 bits or less.

If we now add the results of Sec. 3 regarding the "boost" which the total Hamming distance leakage gives, namely about 2.5 bits per DES key, then we find that about 5.3% of all traces have a rest entropy of $85 - 2 \times 2.5 = 80$ bits or less. A further ≈ 3 bits have to knocked off this figure to account for the single–trace / two–trace effect of two–key TDES referred to above, resulting in about 5.3% of all traces having a rest entropy of $85 - 2 \times 2.5 - 3.4 = 76.4$ bits or less.

It depends on the threat model which of the two is the decisive figure of merit for this attack: The average rest entropy of the entire set of traces, or the tail of the distribution characterised by the relative number of traces below a certain threshold. The latter has to be used in scenarios where either a single successful attack is already not acceptable, almost no matter how small the chance of success is or, alternatively, where many targets will be attacked simultaneously and it does not matter which target will yield in the end or, finally, if one can attack the same target multiple times and thereby increase ones chances. In all these cases the attacker will have compromised about 5.3% of all targets with an effort of 77 bits or less, and (s)he may choose to stop the effort once a few targets have yielded. For completeness, we also provide the figures for smaller threshold values — again all after having accounted for the effect of the total Hamming distance leakage and the single–trace / two–trace effect of two–key

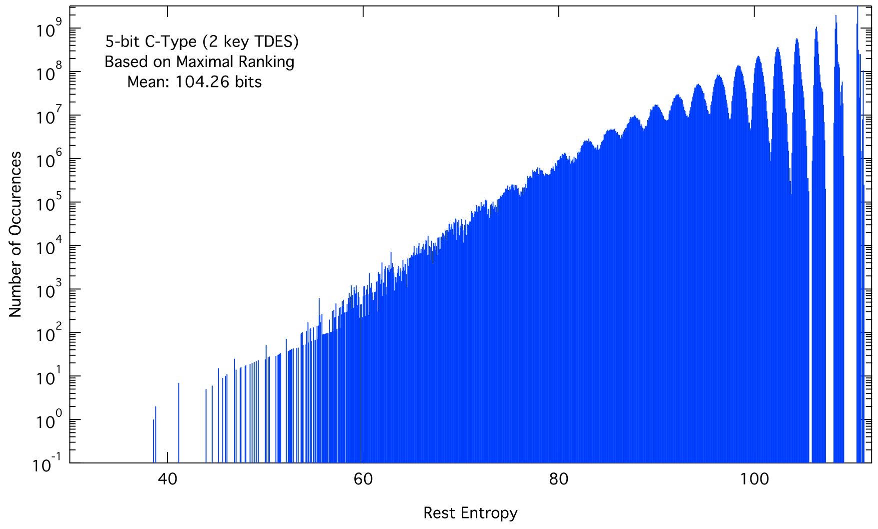


Fig. 14. Distribution of the rest entropy for a 2–key TDES when sorting according to Maximum Ranking.

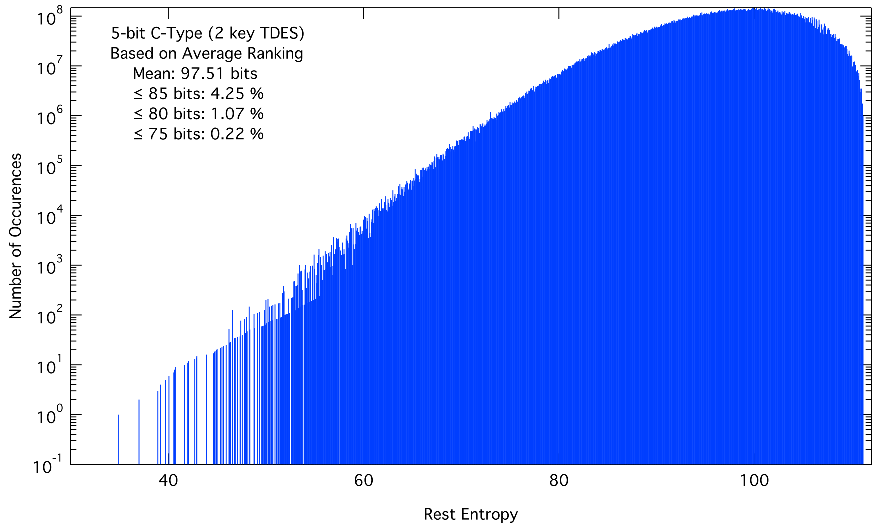


Fig. 15. Distribution of the rest entropy for a 2–key TDES when sorting according to Average Ranking.

TDES: $80 - 2.5 - 3.4 = 71.6$ bits for 1.4% of all traces, and $75 - 2.5 - 3.4 = 66.6$ bits for 0.3% of all traces.[8]

[8] Please see also the discussion in Sec. 5 regarding the existence of weaker keys, which explains the spread seen in these distributions and helps to devise optimal strategies for attackers.

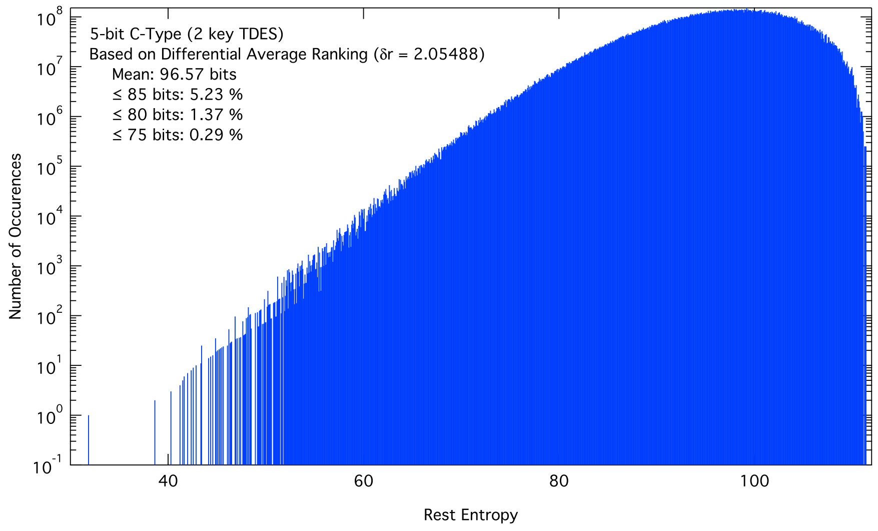


Fig. 16. Distribution of the rest entropy for a 2–key TDES when sorting according to Differential Average Ranking with $\delta r = 2.05488$.

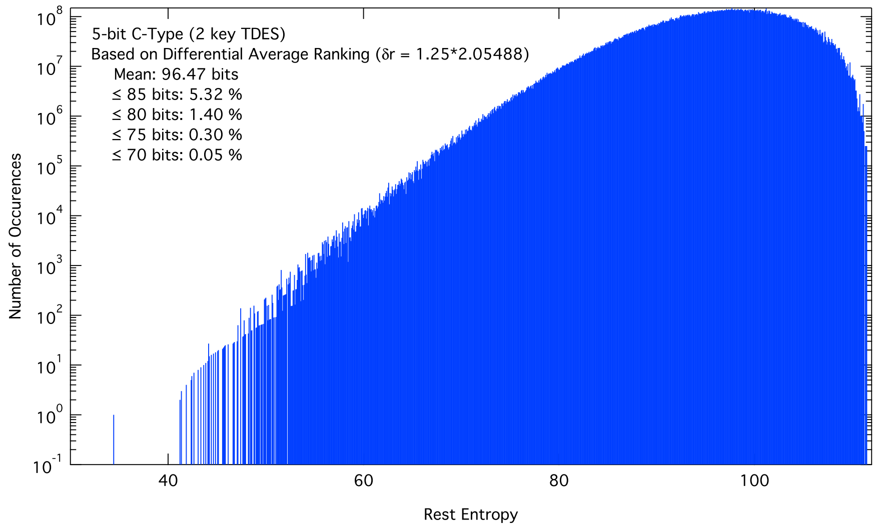
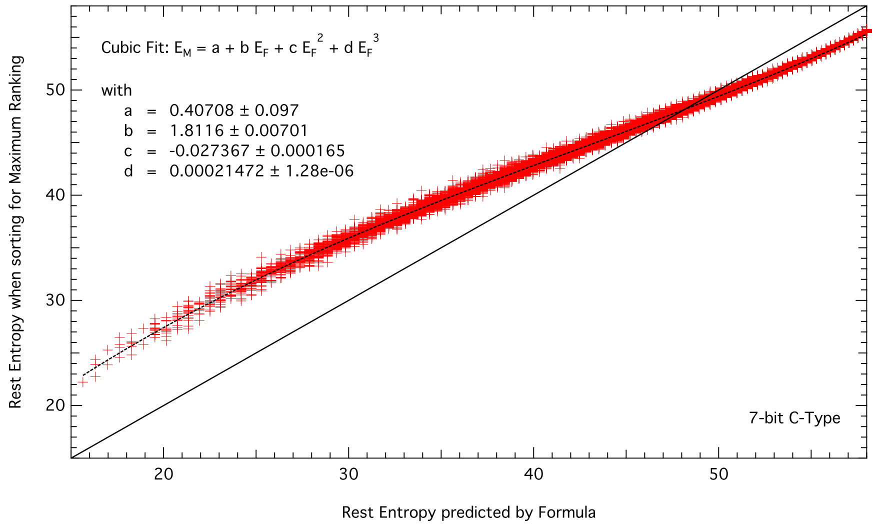


Fig. 17. Distribution of the rest entropy for a 2–key TDES when sorting according to Differential Average Ranking with $\delta r = 1.25 \times 2.05488$.

We conclude that Eq. (10) in [2] is reasonably accurate for this template size of 5 bits, whilst its prediction of the average rest entropy for a 2–key TDES, 109.56 bits, is still some 13 bits too conservative compared to the actual result found with a more optimal search strategy.



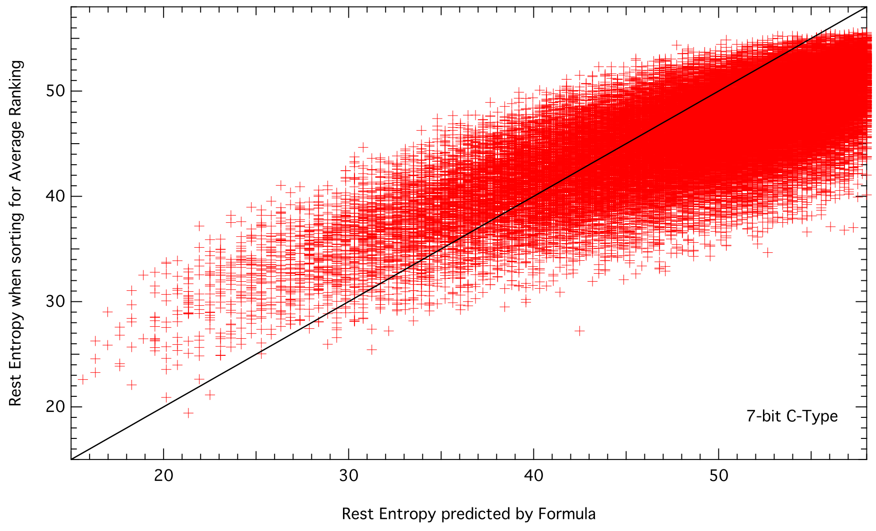


Fig. 18. The top graph shows the rest entropy found when sorting the search lists for Maximum Ranking versus the rest entropy predicted by formula Eq. (10) of [2]. The bottom graph shows the same but instead sorting for Average Ranking. Clearly, the latter is on average a better search strategy for all but perhaps the smallest rest entropies.

4.2 Brute-Force Results for Random Keys (7–Bit Templates)

Next we perform the same analysis as before, but now for 7–bit templates and an ensemble of 64 k single traces. The brute–force effort is largely the same as for the 5–bit templates, only the overhead for creating the unordered 2×2^{27} lists of 27–bit sub–keys is larger. The difference of the average rankings of the C

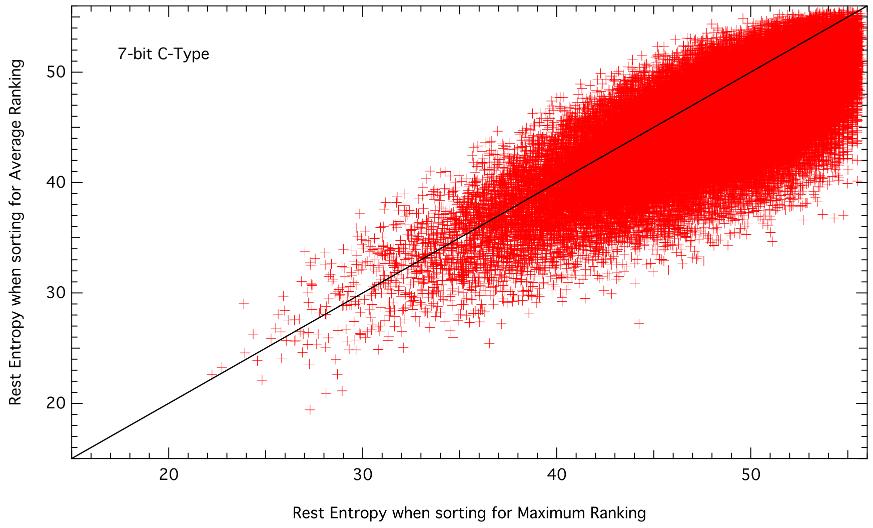


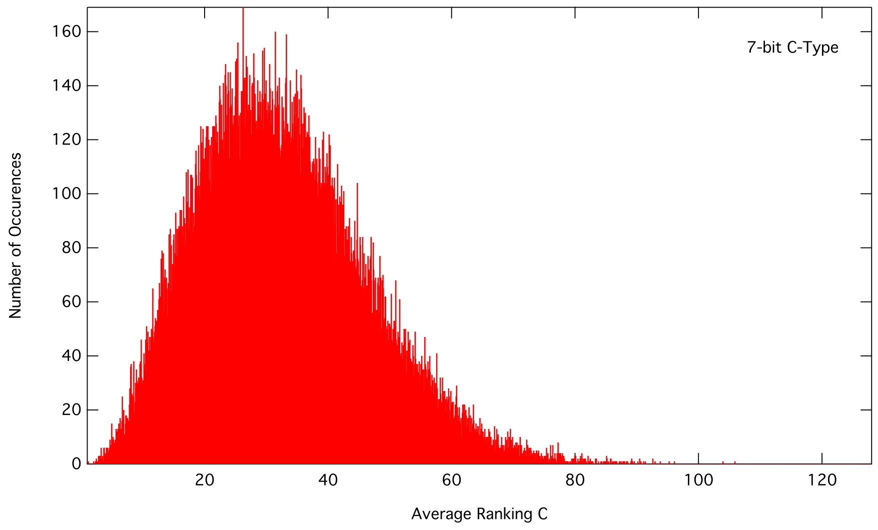
Fig. 19. Same as Fig. 18, but now Average Ranking versus Maximum Ranking.

and D Registers is now 9.1601, with their absolute values being 32.53 and 23.29, respectively.

Fig. 18 shows in its top part the rest entropy when using the Maximum Ranking for sorting the C and D Register lists, as a function of the rest entropy predicted by Eq. (10) in [2]. Clearly, for this template size, the formula is too aggressive for smaller rest entropies, but for larger rest entropies, it still over–estimates the remaining brute–force effort.

According to Figs. 18–29 the 7–bit template results are a little better than those for the 5–bit templates, with the average rest entropy for the single–key DES now being 45.37, and for the 2–key TDES 95.97 bits, almost 1.4 bits better. Some 6.25% traces are having a rest entropy smaller or equal to 85 bits (or 77 bits after again accounting for the additional leakage seen in the Hamming distance of the two C rings, and the single–trace / two–trace effect of two–key TDES). The corresponding values for 80 (72) bits and 75 (67) bits are 1.70% and 0.37%, respectively. All are slightly improved compared to the 5–bit template results.

We conclude that Eq. (10) in [2] is wrong for this template size, yet its prediction of the average rest entropy for a 2–key TDES, 104.56 bits, is still some 8-9 bits too conservative compared to the actual result found with a more optimal search strategy.



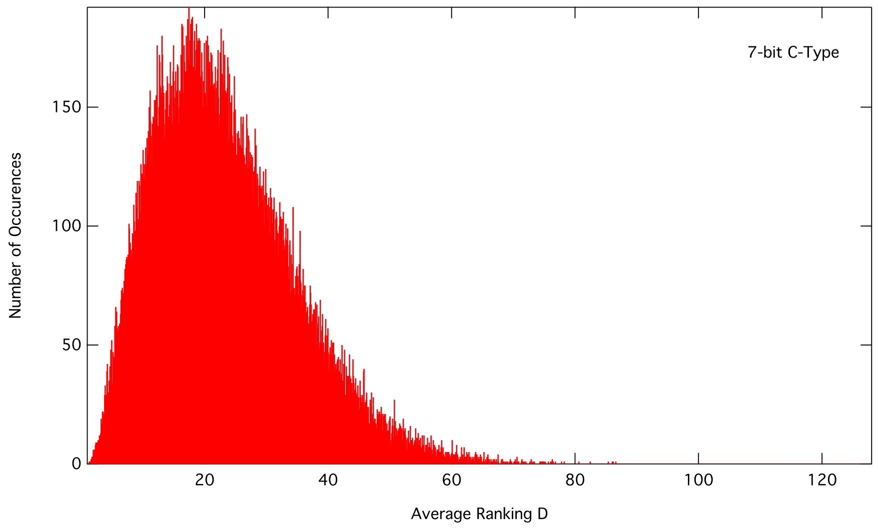


Fig. 20. Distributions found for the Average Ranking of the correct key for C and D Registers.

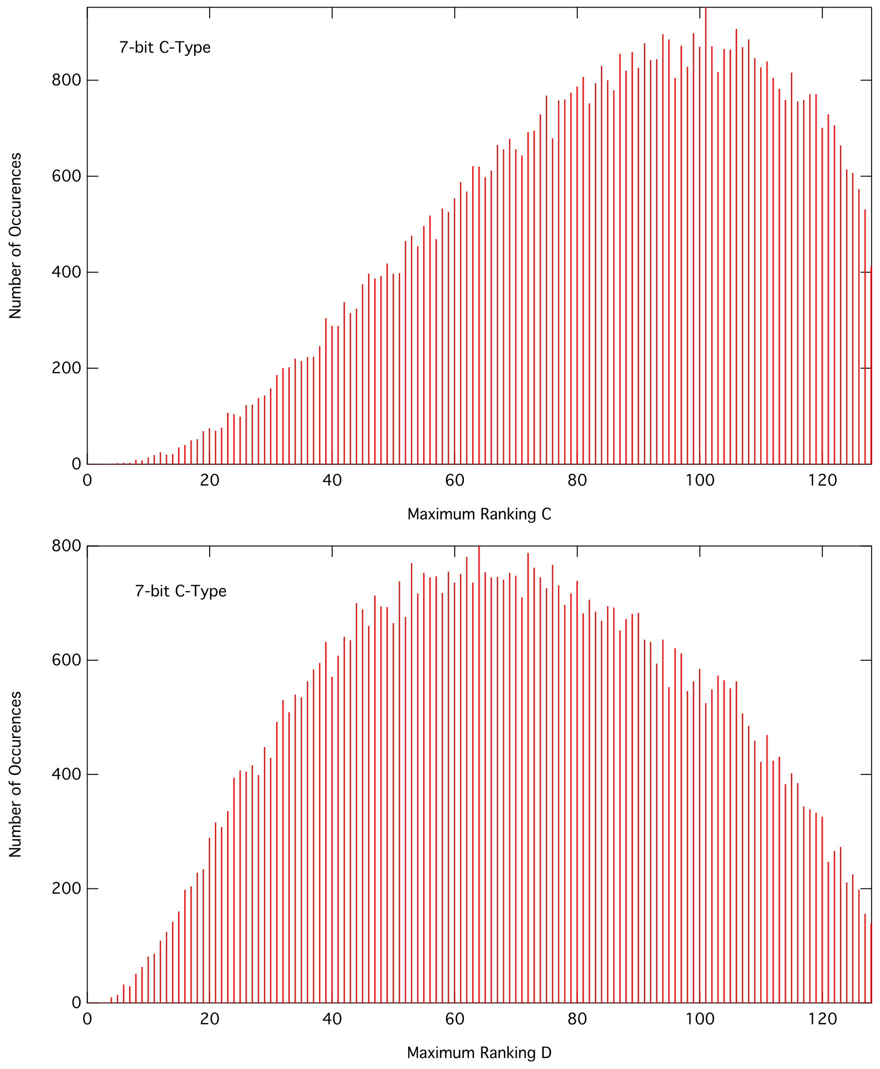


Fig. 21. Distributions found for the Maximum Ranking of the correct key for C and D Registers.

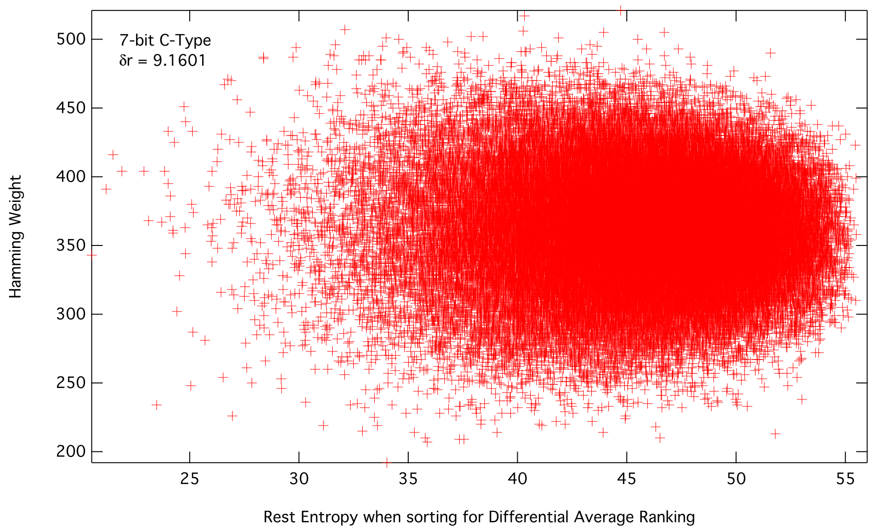
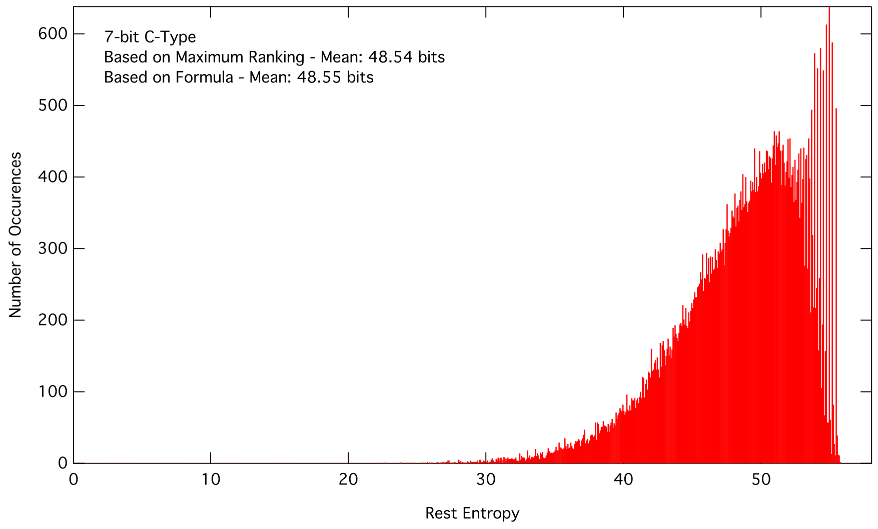


Fig. 22. The Hamming distance versus Differential Average Ranking. The correlation between the two is very weak — of the order of −2.0%, meaning that large Hamming distances will leak only very slightly more than smaller ones.



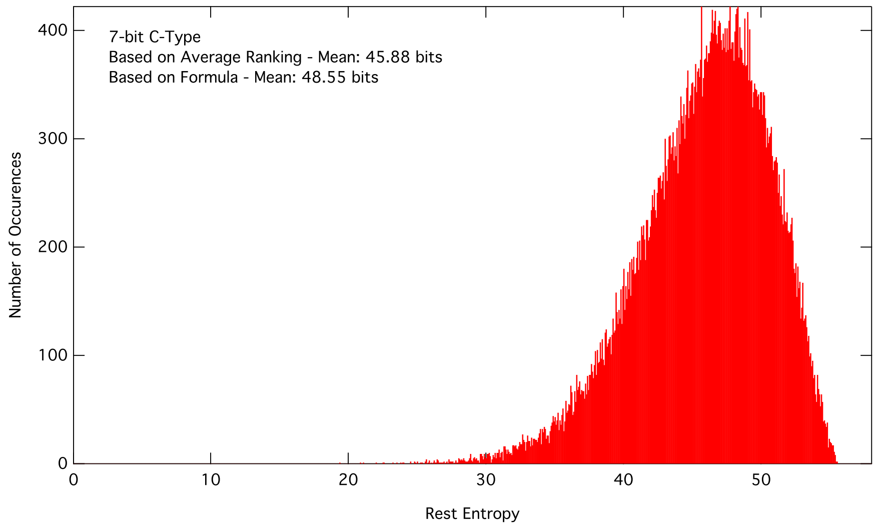
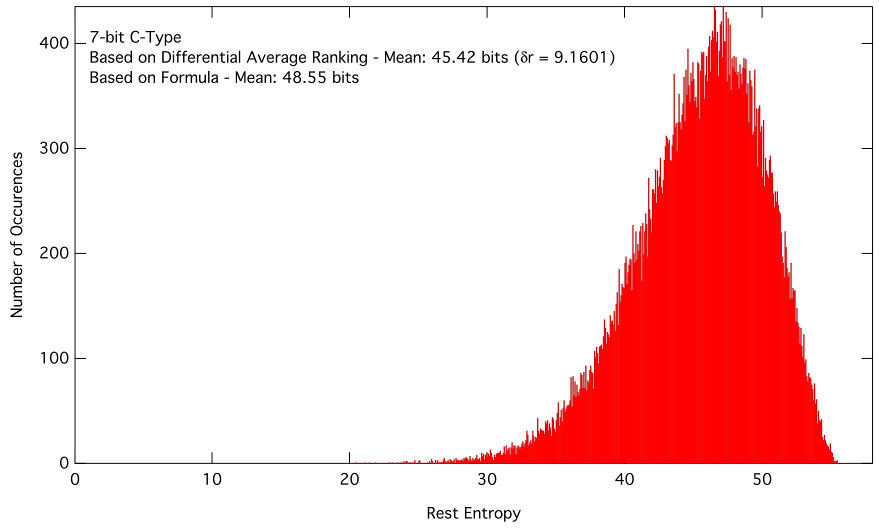


Fig. 23. Distributions for the rest entropy when sorting for Maximum Ranking (top) or Average Ranking (bottom).



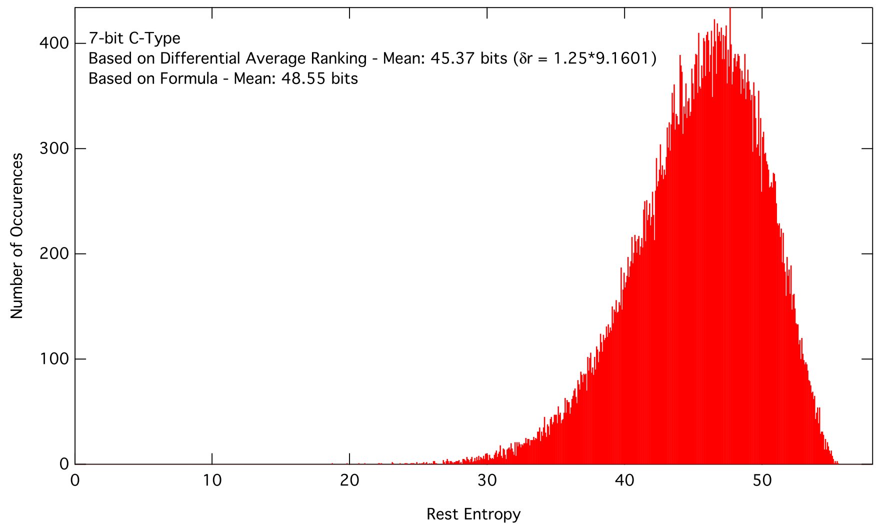


Fig. 24. Distributions for the rest entropy when sorting for Differential Average Ranking with $\delta r = 9.1601$ (top), and $\delta r = 1.25 \times 9.1601$ (bottom).

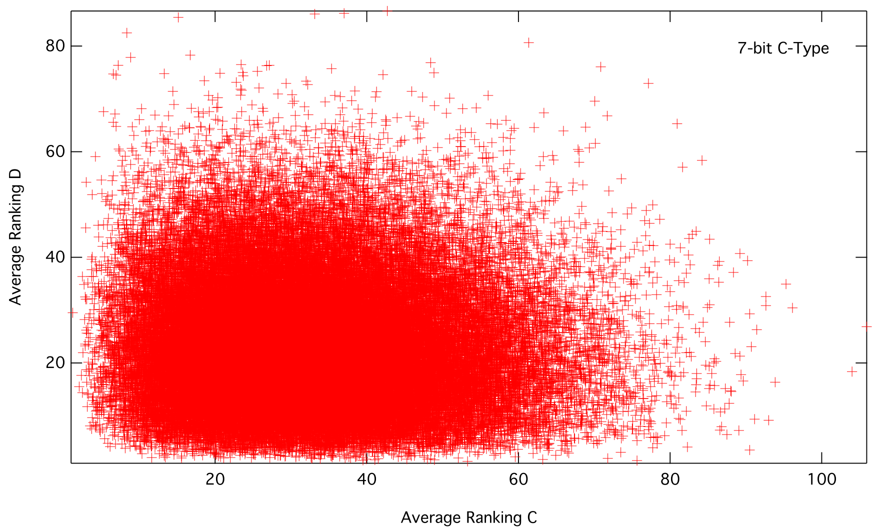


Fig. 25. The correlation between the Average Rankings of the C and the D Register are rather weak, but noticeable at about −3.3%.

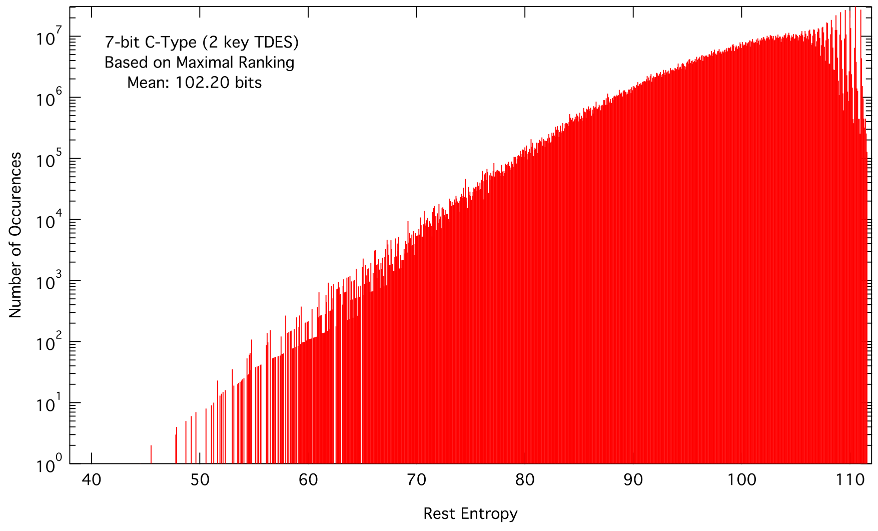


Fig. 26. Distribution of the rest entropy for a 2–key TDES when sorting according to Maximum Ranking.

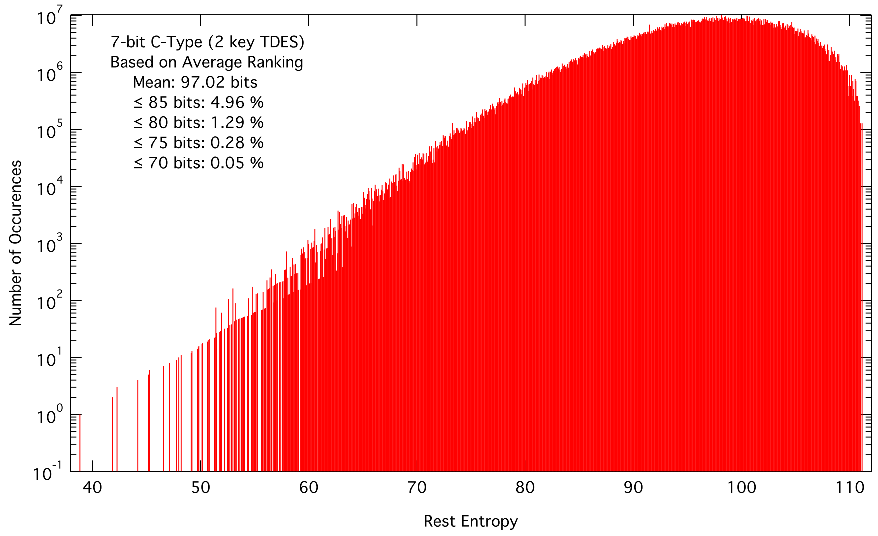


Fig. 27. Distribution of the rest entropy for a 2–key TDES when sorting according to Average Ranking.

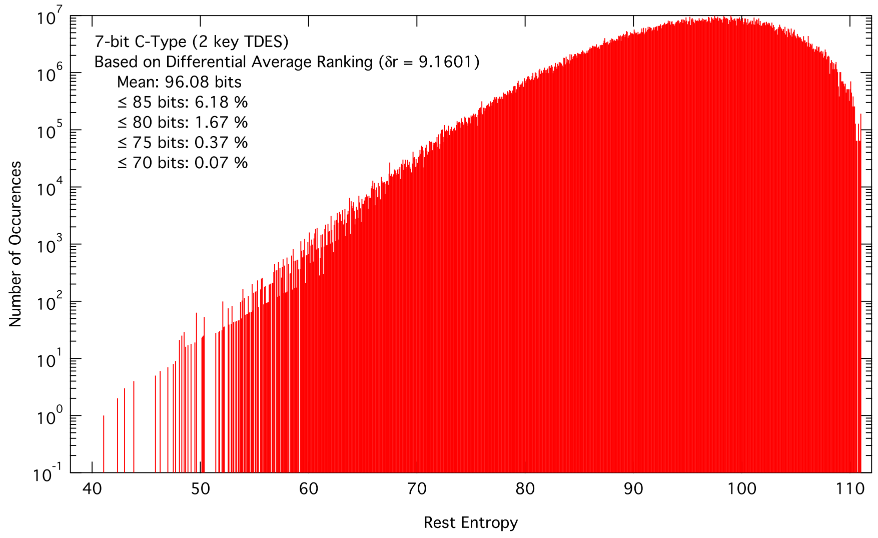


Fig. 28. Distribution of the rest entropy for a 2–key TDES when sorting according to Differential Average Ranking with $\delta r = 9.1601$.

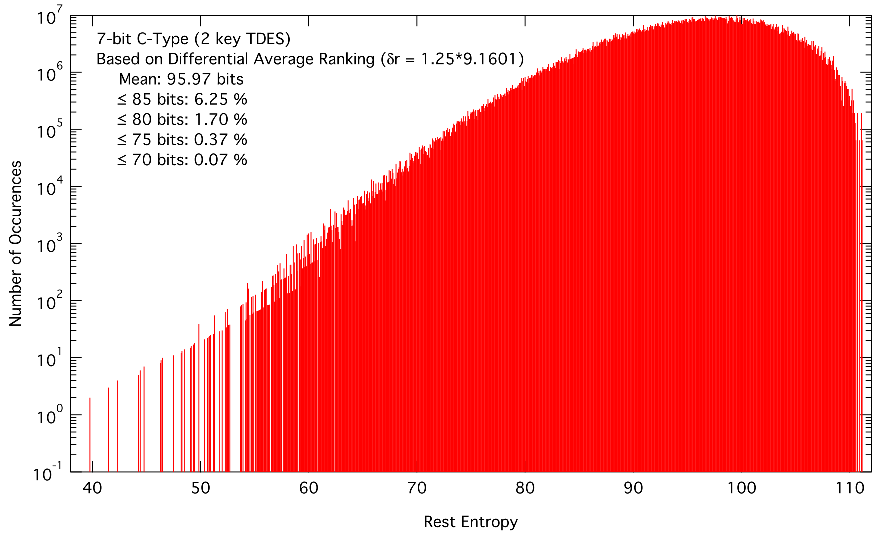
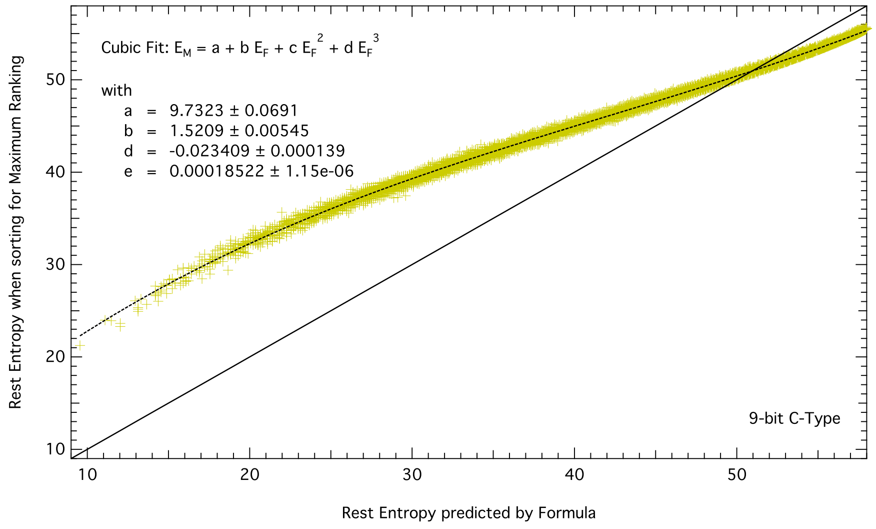


Fig. 29. Distribution of the rest entropy for a 2–key TDES when sorting according to Differential Average Ranking with $\delta r = 1.25 \times 9.1601$.



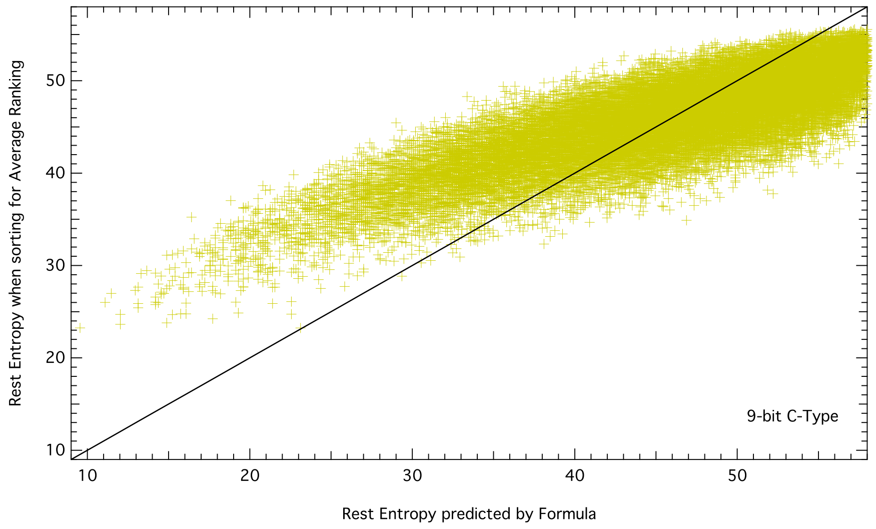


Fig. 30. The top graph shows the rest entropy found when sorting the search lists for Maximum Ranking versus the rest entropy predicted by formula Eq. (10) of [2]. The bottom graph shows the same but instead sorting for Average Ranking. Clearly, the latter is on average a better search strategy for all but perhaps the smallest rest entropies.

4.3 Brute-Force Results for Random Keys (9–Bit Templates)

Next we perform the same analysis as before, but now for 9–bit templates and an ensemble of 32 k single traces. The brute–force effort is largely the same as for the 5–bit and 7–bit templates, only the overhead for creating the unordered 2×2^{27} lists of 27–bit sub–keys is yet larger. The difference of the average rankings

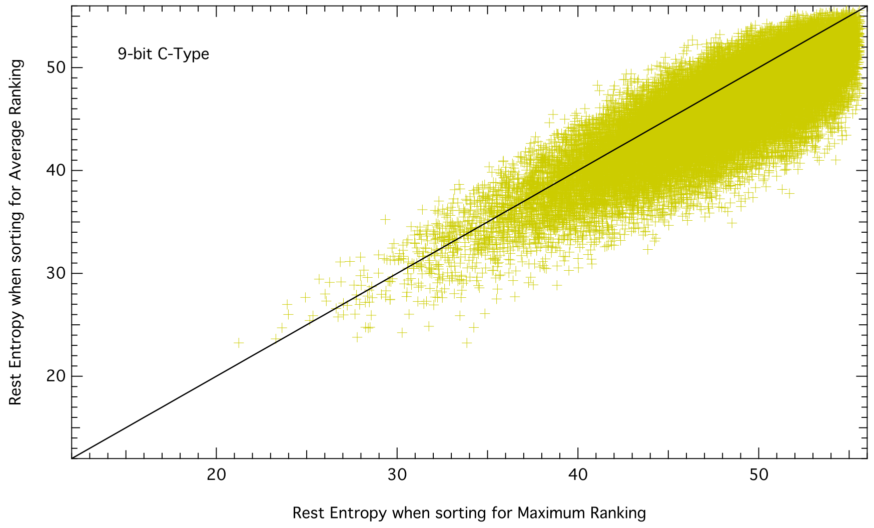


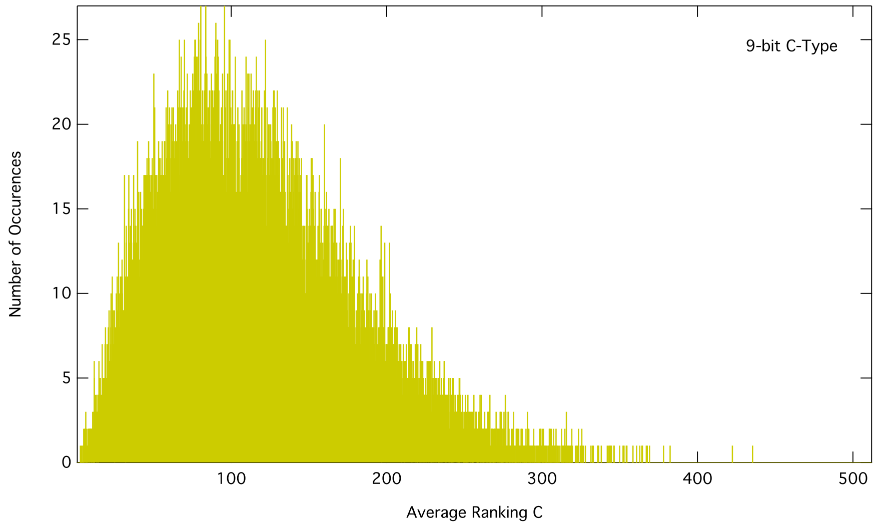
Fig. 31. Same as Fig. 30, but now Average Ranking versus Maximum Ranking.

of the C and D Registers is now 39.3155, with the absolute values being 114.51 and 75.20, respectively. We note that this difference increases faster than by the expected factor 4 when moving from one template size to the next.

Fig. 30 shows in its top part the rest entropy when using the Maximum Ranking for sorting the C and D Register lists, as a function of the rest entropy predicted by Eq. (10) in [2]. Clearly, the formula is too aggressive for smaller rest entropies, but for larger rest entropies, it over–estimates the remaining brute–force effort. The difference between formula prediction and real brute–force results for the rest entropy is even larger than in the 7–bit template case.

According to Figs. 30–41 the 9–bit templates results are yet a little better than those for the 7–bit templates, with the average rest entropy for the single–key DES now being 45.34, and for the 2–key TDES 95.90 bits. Some 6.39% traces are having a rest entropy smaller or equal to 85 bits (or 77 bits after again accounting for the additional leakage seen in the Hamming distance of the two C rings, and the single–trace / two–trace effect of two–key TDES, as discussed in Sec. 3). The corresponding values for 80 (72) bits and 75 (67) bits are 1.72% and 0.38%, respectively. All are again slightly improved compared to the 7–bit template results, but the magnitude of the improvement is smaller compared to the difference between 7–bit and 5–bit results.

We conclude that although Eq. (10) in [2] is not quite accurate anymore for this template size, its prediction of the average rest entropy for a 2–key TDES, 100.27 bits, is still some 4-5 bits too conservative compared to the result found with a more optimal search strategy.



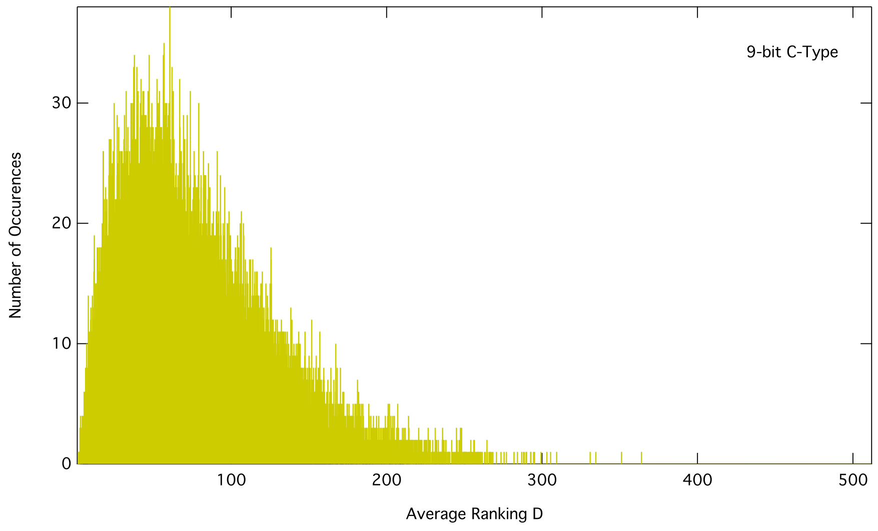
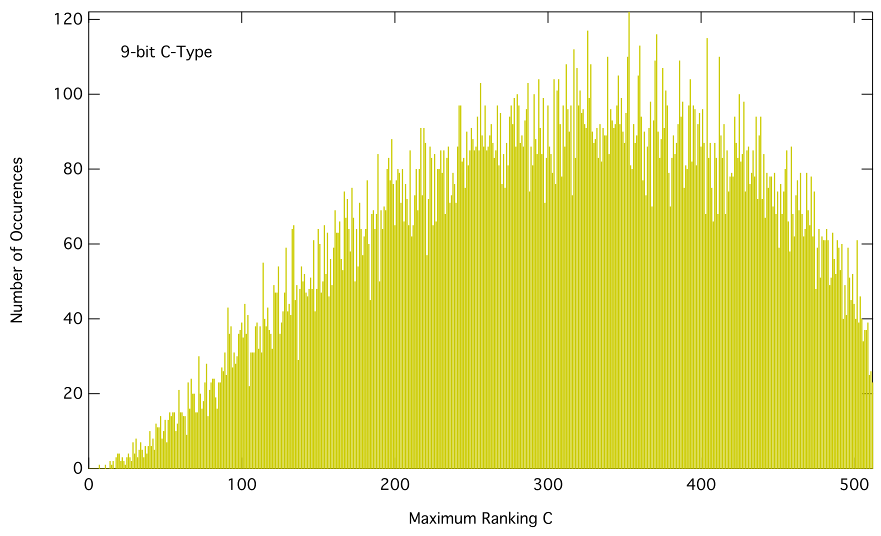


Fig. 32. Distributions found for the Average Ranking of the correct key for C and D Registers.

4.4 Comparing Results for Varying Template Sizes

When comparing the results of the previous Sections it is noticeable that larger template sizes do improve the results, but not dramatically so. There are two possible reasons for that: Firstly, a larger template primarily means that more averaging happens along the two C rings. However, the entire brute–force approach involves a lot of explicit and implicit averaging, anyway, and this may then reduce the impact of further improved averaging due to larger template sizes. Secondly, though, it cannot be excluded that there is still a subtle programming



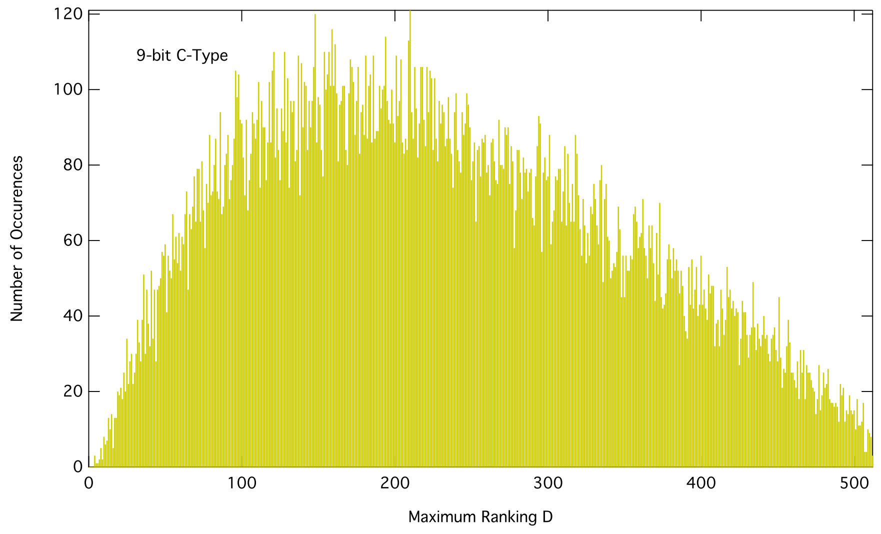


Fig. 33. Distributions found for the Maximum Ranking of the correct key for C and D Registers.

bug for larger template sizes exceeding 8 bits (where for this implementation–specific reasoning also the linearly dependent bits — as defined in [2] — need to be counted).

In any case, it is clear that the interpretation and applicability of the results will depend on the threat model at hand. When the threat model is to attack one particular application of a given device, then the figure of merit for the attack will most likely have to be the average rest entropy, be it single–key or two–key DES, unless the same device can be attacked multiple times. When this

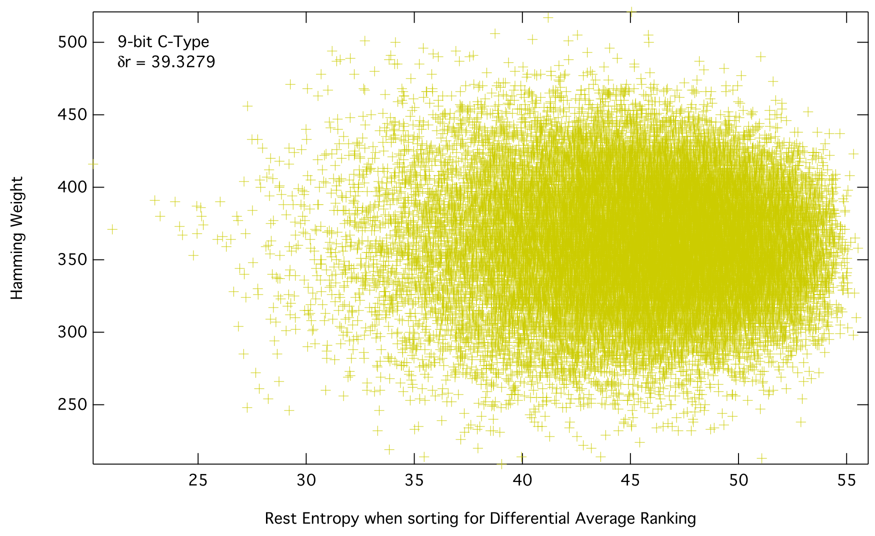


Fig. 34. The Hamming distance versus Differential Average Ranking. The correlation between the two is very weak — of the order of -1.8%, meaning that large Hamming distances will leak only very slightly more than smaller ones.

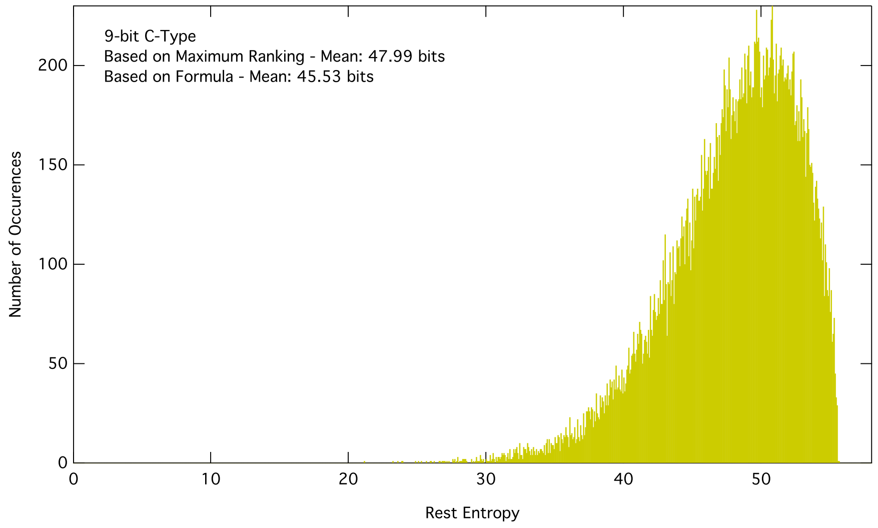
Table 2. Two–key TDES, Differential Average Ranking: For various template sizes 5, 7, and 9, as well as for various rest–entropy thresholds ranging from ≤ 95 bits down to ≤ 50 bits, the effective rest entropy is calculated (rows 5 to 7) that factors in the probability of success (within this threshold), as given in rows 2 to 4 in this table.

bits	≤ 95	≤ 90	≤ 85	≤ 80	≤ 75	≤ 70	≤ 65	≤ 60	≤ 55	≤ 50
5	37.90 %	16.18 %	5.29 %	1.39 %	0.30 %	0.053 %	8.0e-3 %	1.1e-3 %	1.2e-4 %	9.3e-6 %
7	41.11 %	18.28 %	6.25 %	1.70 %	0.37 %	0.066 %	9.7e-3 %	1.2e-3 %	9.5e-5 %	1.0e-5 %
9	41.53 %	18.50 %	6.39 %	1.72 %	0.38 %	0.068 %	9.1e-3 %	1.1e-3 %	8.5e-5 %	8.8e-6 %
5	96.40	92.63	89.24	86.17	83.40	80.89	78.60	76.48	74.66	73.36
7	96.28	92.45	89.00	85.88	83.06	80.56	78.33	76.40	75.00	73.22
9	96.27	92.44	88.97	85.86	83.03	80.51	78.43	76.49	75.17	73.44

is the case, or when the threat model is such that the attacker can attack a large number of similar devices, and has succeeded when just one device yields, as is perhaps the case in banking, where the attacker does not care whose money (s)he steals, then the figure of merit will be different. In such a scenario one needs to assess the long tails in the distribution of rest entropies in more detail.[9]

To this end, we have calculated the *effective* rest entropy by taking a given threshold value for the left–sided tail in the rest–energy distribution, say ≤ 70 bits, and convert the corresponding probability $P_{\leq 70}$ of succeeding with a brute–force attack within this threshold to an additional contribution, $-\log_2(P_{\leq 70})$, to the rest entropy, leading to $E_{\text{effective}} = 70 - \log_2(P_{\leq 70})$. The results of this

[9] See Sec. 5 for details on weak keys.



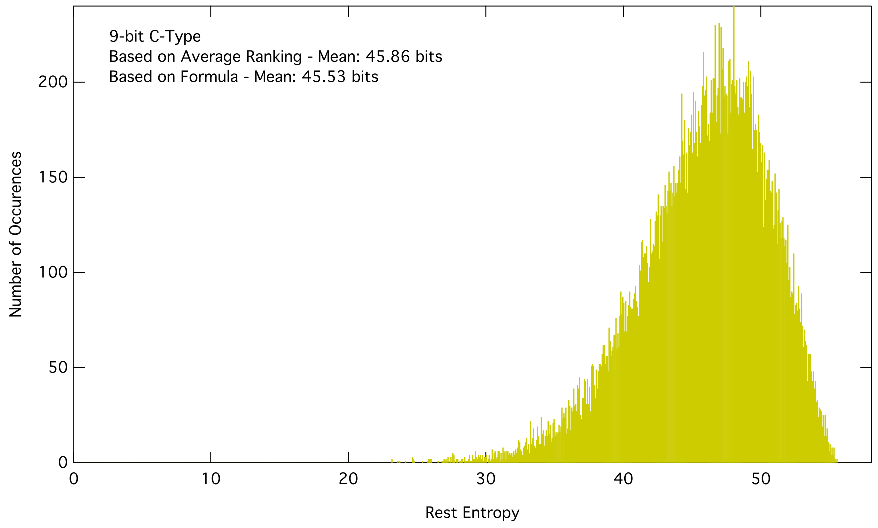
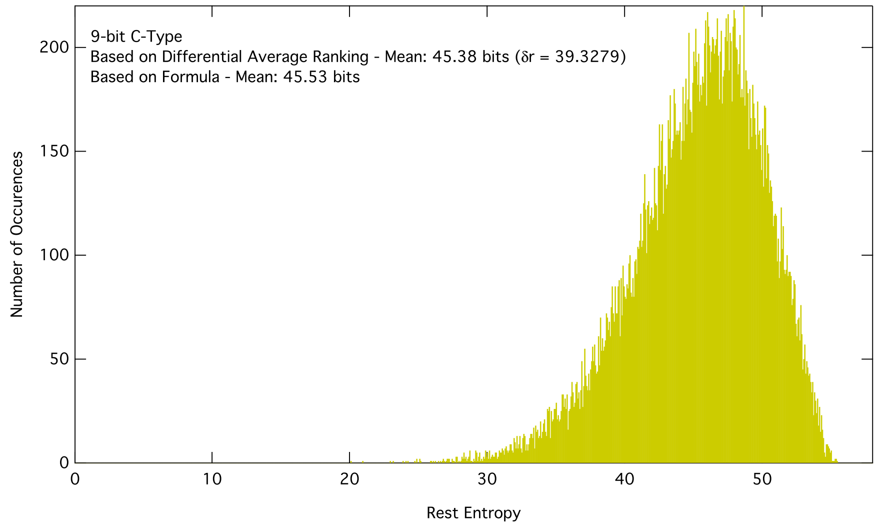


Fig. 35. Distributions for the rest entropy when sorting for Maximum Ranking (top) or Average Ranking (bottom).

approach are shown in Table 2. Interestingly, one finds that this value decreases when moving to lower threshold values.[10] This is simply because the distribution tail is very long.

Concretely, using a 9–bit template and a threshold of 75 bits, if the attacker runs $1/P_{\leq 75} \approx 263$ attacks in parallel on 263 different devices or, alternatively,

[10] It should be noted that the statistics is not very accurate for very small numbers of events, i.e., for $P_{\leq 50}$, $P_{\leq 55}$, and perhaps $P_{\leq 60}$.



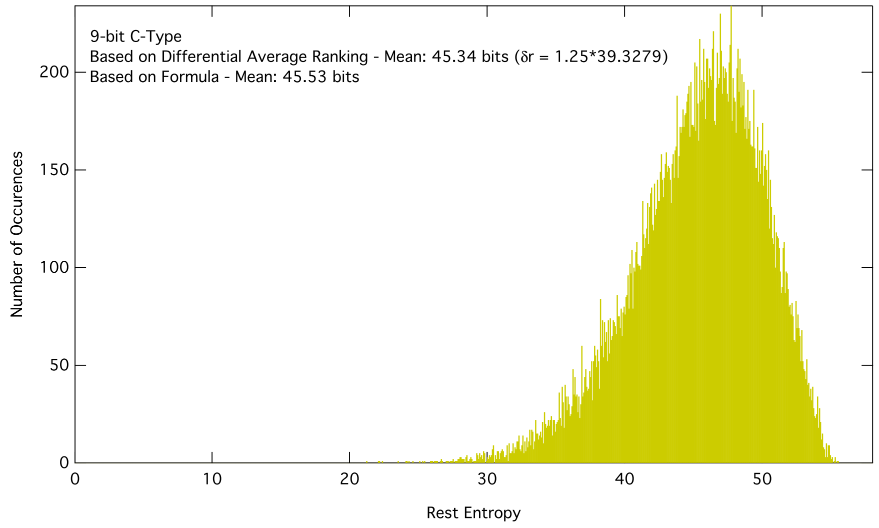


Fig. 36. Distributions for the rest entropy when sorting for Differential Average Ranking with $\delta r = 39.3279$ (top), and $\delta r = 1.25 \times 39.3279$ (bottom).

on 263 different single traces of the same device, (s)he will spend a total effort of about 83 bits across all these 263 attacks.[11]

As before, all two–key TDES brute–force efforts will be reduced by about 5 bits when including the total Hamming distance leakage as discussed in Sec. 3.

[11] It should be noted here that the effort is even slightly smaller than that, since the effort was calculated based on the threshold value, whilst in reality some brute–force efforts will succeed earlier than that, as the distributions show. Due to non–linearity, this will be an effect much smaller than one bit, though, and hence is neglected here.

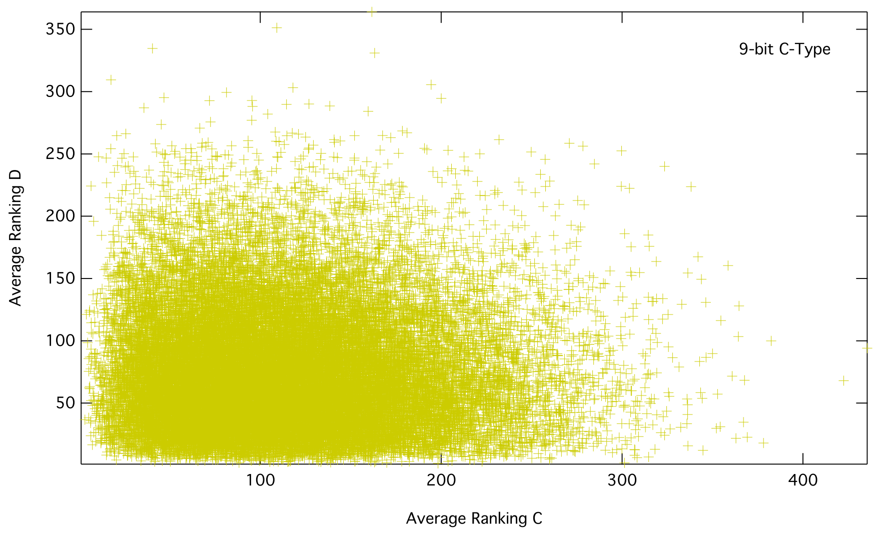


Fig. 37. The correlation between the Average Rankings of the C and the D Register are rather weak, but noticeable at about −1.6%.

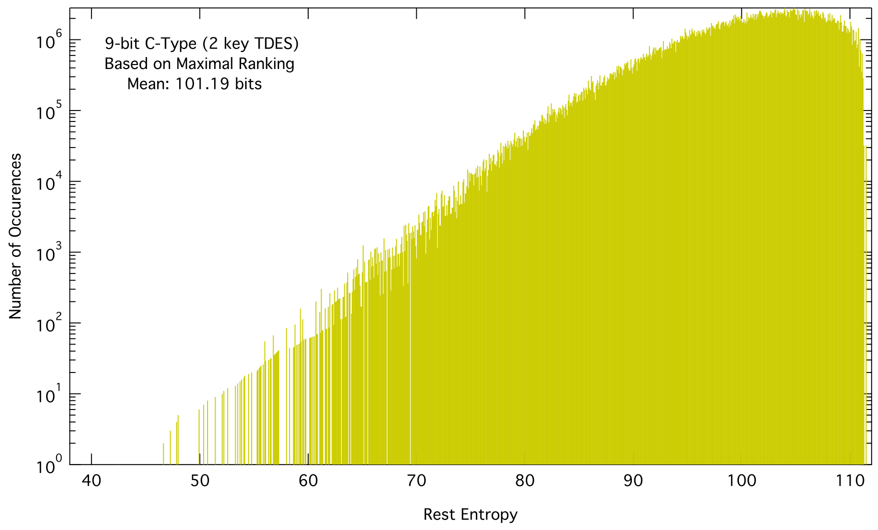


Fig. 38. Distribution of the rest entropy for a 2–key TDES when sorting according to Maximum Ranking.

Secondly, as first indicated in Sec. 4.1 and subsequently further elaborated upon in Sec. 6.2, a more correct analysis of the brute–force effort for the two–key TDES case would result in a further reduction by some 3 bits due to one key in the two–key TDES being used twice.

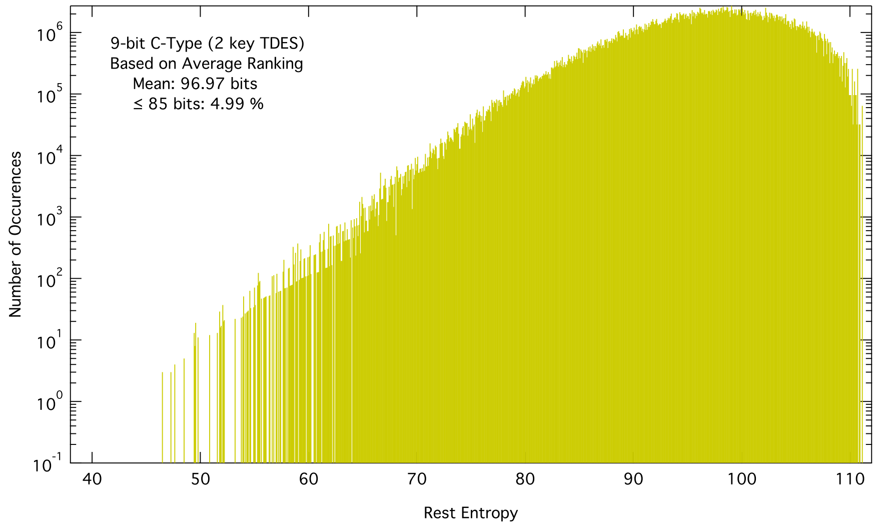


Fig. 39. Distribution of the rest entropy for a 2–key TDES when sorting according to Average Ranking.

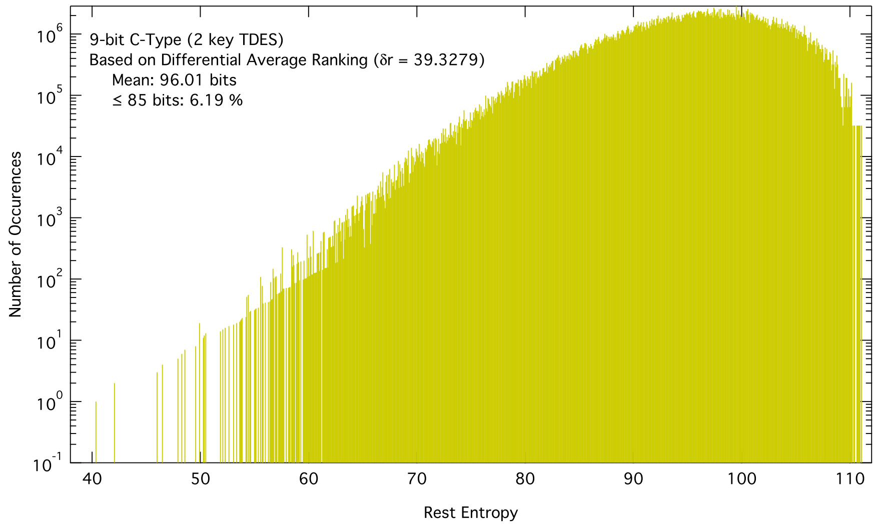


Fig. 40. Distribution of the rest entropy for a 2–key TDES when sorting according to Differential Average Ranking with $\delta r = 39.3279$.

5 Weak Keys

So far, we have established that the brute–force results do not seem to depend much on the value of the total Hamming distance. E.g., Fig. 34 shows only a minor dependency. However, this does not mean that the attack has the same efficiency regardless of the targeted key. Indeed, in this Section we will show

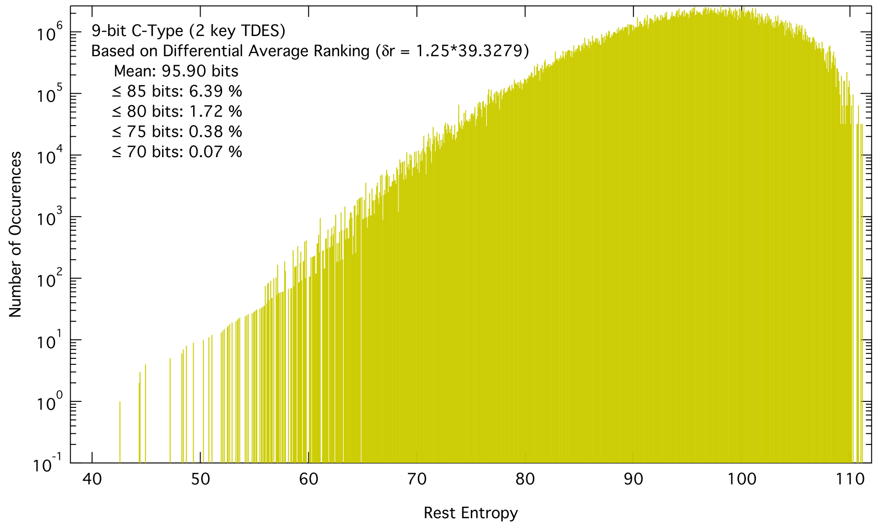


Fig. 41. Distribution of the rest entropy for a 2–key TDES when sorting according to Differential Average Ranking with $\delta r = 1.25 \times 39.3279$.

that some keys are much weaker than others within the framework of our attack, having a rest entropy of the order of 2 bits only. As we will see, there is no clear separation between "weak" and "normal" keys possible, and thus the definition of what constitutes a weak key is somewhat arbitrary, but for all intents and purposes a few percent of the total number of keys need to be regarded as weak keys.

In the following let us assume a perfect leakage of the Hamming distances of the 15 consecutive rounds. This can be represented by a 15–tuple containing the 15 values of the Hamming distances of these 15 consecutive rounds, and can be defined via the function $\mathrm{dist15}(k) : k \to 15$–tuple. The question to be answered then is, for a given DES key k, how many different keys k' will map to the same given 15–tuple such that $\mathrm{dist15}(k') = \mathrm{dist15}(k)$, i.e., how many DES keys will collide with each other in this model. This figure will then determine the lowest possible rest entropy that could be achieved within this leakage model,[12] and it should serve as an indication of how strong this attack can become for a given targeted key.

The most straight–forward approach to determining $\mathrm{dist15}(k)$ is to start from the 16 round keys of the DES key schedule. However, in order to stay consistent with the rest of this paper, we derive $\mathrm{dist15}(k)$ using the ring structure found in the DES key schedule. Thus, in the first step, reusing Table 5 of [2], we define two matrices A and B that together calculate the Hamming distances in the 15

[12] Of course, when the leakage mechanism is more powerful than "perfect Hamming distance" and individual bits leak differently, then the rest entropies will be lower than predicted in the 15–tuple model, and consequently the attack will be even stronger.

consecutive rounds when multiplied with a bit vector representing the $(\oplus)$ bits in the rings R_i^A and R_j^B as defined in [2]. More precisely, let R^A be the concatenation of the four A rings, $\mathrm{R}^A = \mathrm{R}_0^A||\mathrm{R}_1^A||\mathrm{R}_2^A||\mathrm{R}_3^A$, and R^B be the concatenation of the two B rings, $\mathrm{R}^B = \mathrm{R}_0^B||\mathrm{R}_1^B$, with both bit vectors containing 56 bits each.[13]

The A matrix will then yield the Hamming distances in the 12 consecutive rounds that relate to the four A rings, whilst the B matrix will do the same for the 3 consecutive rounds the two B rings are associated with. Thus, in total, we will have generated the desired 15–tuple of Hamming distances of 15 consecutive rounds.

After some bit manipulations one finds the A matrix to be represented by

$$\begin{array}{*{56}{c}}
1&1&1&0&1&1&1&1&1&1&1&0&1&1&1&1&1&1&1&0&1&0&1&1&1&1&1&1&1&1&1&1&0&1&1&1&0&1&1&1&1&1&1&1&1&1&1&1&1&0&1&1&1&1&1&0\\
1&1&0&1&1&1&1&1&1&1&0&1&1&1&1&1&1&1&0&1&0&1&1&1&1&1&1&1&1&1&1&0&1&1&1&0&1&1&1&1&1&1&1&1&1&1&1&1&0&1&1&1&1&1&0&1\\
1&0&1&1&1&1&1&1&1&0&1&1&1&1&1&1&1&0&1&0&1&1&1&1&1&1&1&1&1&1&0&1&1&1&0&1&1&1&1&1&1&1&1&1&1&1&1&0&1&1&1&1&1&0&1&1\\
0&1&1&1&1&1&1&1&0&1&1&1&1&1&1&1&0&1&0&1&1&1&1&1&1&1&1&1&1&0&1&1&1&0&1&1&1&1&1&1&1&1&1&1&1&1&0&1&1&1&1&1&0&1&1&1\\
1&1&1&1&1&1&1&0&1&1&1&1&1&0&1&0&1&0&1&1&1&1&1&1&1&1&1&1&0&1&1&1&0&1&1&1&1&1&1&1&1&1&1&1&1&0&1&1&1&1&1&0&1&1&1&1\\
1&1&1&1&1&1&0&1&1&1&1&1&0&1&0&1&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&1&1&1&1&1&1&1&1&1&1&1&0&1&1&0&1&1&1&1&1&0&1&1&1&1&1\\
0&1&1&1&1&1&1&1&1&1&1&1&0&1&1&1&1&1&1&0&1&1&1&1&1&0&1&1&0&1&1&1&1&1&0&1&1&1&1&1&1&1&1&1&0&1&1&1&1&1&1&1&1&1&0&1\\
1&1&1&1&1&1&1&1&1&1&1&0&1&0&1&1&1&1&0&1&1&1&1&1&0&1&1&1&1&1&1&1&1&0&1&1&1&1&1&1&1&0&1&0&1&1&1&1&1&1&1&1&1&0&1&1\\
1&1&1&1&1&1&1&1&1&1&0&1&0&1&1&1&1&0&1&1&1&1&1&0&1&1&1&1&1&1&1&1&0&1&1&1&1&1&1&1&0&1&0&1&1&1&1&1&1&1&1&1&0&1&1&1\\
1&1&1&1&1&1&1&1&1&0&1&0&1&1&1&1&0&1&1&1&1&1&0&1&1&1&1&1&1&1&1&0&1&1&1&1&1&1&1&0&1&1&1&1&1&1&1&1&1&1&1&0&1&1&1&0\\
1&1&1&1&1&1&1&1&0&1&0&1&1&1&1&0&1&1&1&1&1&0&1&1&1&1&1&1&1&1&0&1&1&1&1&1&1&1&0&1&1&1&1&1&1&1&1&1&1&1&0&1&1&1&0&1\\
1&1&1&1&1&1&1&0&1&0&1&1&1&1&0&1&1&1&1&1&0&1&1&1&1&1&1&1&1&0&1&1&1&1&1&1&1&0&1&1&1&1&1&1&1&1&1&1&1&0&1&1&1&0&1&1
\end{array} \tag{5}$$

and likewise the B matrix to be given by

$$\begin{array}{*{56}{c}}
1&1&1&1&1&1&1&1&1&0&1&1&0&1&1&1&0&1&1&1&1&1&1&1&1&0&1&1&0&1&1&1&1&1&1&1&1&1&1&0&1&1&1&1&0&1&1&0&1&1&1&1&1&1&1&1\\
1&1&1&0&1&1&1&1&1&1&1&1&0&1&1&1&1&1&1&1&1&1&1&1&0&1&1&0&1&1&1&0&1&1&0&1&1&1&1&1&1&1&1&0&1&1&1&1&1&1&1&1&1&1&0&1\\
1&1&1&1&1&1&1&1&1&1&1&0&1&1&0&1&1&1&0&1&1&1&1&1&1&1&1&0&1&1&0&1&1&1&1&1&1&1&1&1&1&0&1&1&1&1&0&1&1&0&1&1&1&1&1&1
\end{array} . \tag{6}$$

As it stands, these two matrices operate on the A and B rings, respectively, which are not independent of each other, but rather connected via the C rings. Therefore, before using these matrices to work out dist15(k), they need to be cast to the same bit vector basis. The easiest way of doing this is to express the R^A vectors in terms of orthogonal and unitary R^B basis vectors. This is easily visualised with the help of the C rings of Eq. (1): Consider, for example, the basis vector on the B ring where only the bit $7 \oplus 14$ is set in the B ring, with all other bits in both B rings being zero. If we then set the two bits $7 \oplus 21$ and $0 \oplus 14$ in the A rings to one, with all other bits being zero, then the two $\oplus$ loops $7 \to 14 \to 21 \to 7$ and $7 \to 14 \to 0 \to 7$ in the C ring are both satisfied, as are all other possible loops, and thus everything is consistent as it should be.

By virtue of the cyclic nature of the B rings, this construction works the same for all possible 56 basis vectors, yielding a transformation matrix L as

$$L = \begin{pmatrix} \tilde{L} & 0 \\ 0 & \tilde{L} \end{pmatrix} \tag{7}$$

[13] For the sake of clarity, this means that $\mathrm{R}^A = (0 \oplus 14, 14 \oplus 28, 28 \oplus 42, ...)$ and $\mathrm{R}^B = (0 \oplus 7, 7 \oplus 14, 14 \oplus 21, ...)$. Note, though, that although these vectors contain 56 bits, only 54 bits of those a linearly independent because of the cyclic nature of the rings.

such that $\mathrm{R}^A = (L\mathrm{R}^B) \mod 2$ for any R^B, where $\tilde{L}$ is given by

$$\begin{array}{*{28}{c}}
0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\
0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\
0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\
0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\
0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\
0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0 \\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0 \\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0 \\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0 \\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0 \\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0 \\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0 \\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1 \\
1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\
0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\
0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\
0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\
0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\
0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0 \\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0 \\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0 \\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0 \\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0 \\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0 \\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1
\end{array} . \tag{8}$$

Taking $A((L\mathrm{R}^B) \mod 2)$ and $B\mathrm{R}^B$ together yields $\mathrm{dist15}(\mathrm{R}^B)$, which is closely related to our originally targeted mapping function $\mathrm{dist15}(k)$. It only differs in that for every possible R^B vector there exist 4 possible DES keys k.

To tackle the challenge of finding all collisions for a given 15–tuple we first notice that $\mathrm{dist15} = \mathrm{dist15}_C + \mathrm{dist15}_D$, where $\mathrm{dist15}_C$ and $\mathrm{dist15}_D$ refer to the contributions to the 15–tuple stemming from the C and the D Register, respectively. They can be obtained, for instance, using Table 5 of [2]. These Hamming distances are simply additive. This allows us to reduce the complexity of the problem by constructing a meet–in–the–middle algorithm. In the next step we create two ordered lists, one for the C Register, and another one for the D Register, where both contain 2^{27} entries of 28–bit sub–keys (k_C or k_D) taken from the B rings, together with their respective values of $\mathrm{dist15}_C(k_C)$ respectively $\mathrm{dist15}_D(k_D)$. We then order each list with respect to two parameters — firstly the total Hamming distance $||\mathrm{dist15}_C(k_C)||_1$ respectively $||\mathrm{dist15}_D(k_D)||_1$ of the sub key as primary criterion, where $||.||_1$ denotes the L_1 norm and, secondly, within a given total Hamming distance, the numerical value of $\mathrm{dist15}_C$ or $\mathrm{dist15}_D$ when interpreting it as a large, multi–digit number. Now, for any target key k given, we first calculate $||\mathrm{dist15}(k)||_1$, and then search within all possible combinations of $||\mathrm{dist15}_C||_1 + ||\mathrm{dist15}_D||_1 = ||\mathrm{dist15}||_1$ as follows: For every k_C satisfying $||\mathrm{dist15}_C(k_C)||_1 = ||\mathrm{dist15}_C||_1$ we search over all k_D satisfying $||\mathrm{dist15}_D(k_D)||_1 = ||\mathrm{dist15}_D||_1$ to find all instances where also $\mathrm{dist15}_C(k_C) + \mathrm{dist15}_D(k_D) = \mathrm{dist15(k)}$ holds.[14] With this approach it is possible to do a reasonably fast brute–force search of all key collisions when given a random DES key R^B as input. On a Mac PC it takes on average about $0.5s$ to find all the collisions for a single key.

[14] The second ordering criterion of interpreting $\mathrm{dist15}_C(k_C)$ and $\mathrm{dist15}_D(k_D)$ as large, multi–digit numbers allows us to find very good initial starting values for each search by using the result k'_D of a previous search as initial guess. But of course, other data structures such as hash tables are also possible and may perhaps be even more efficient solutions.

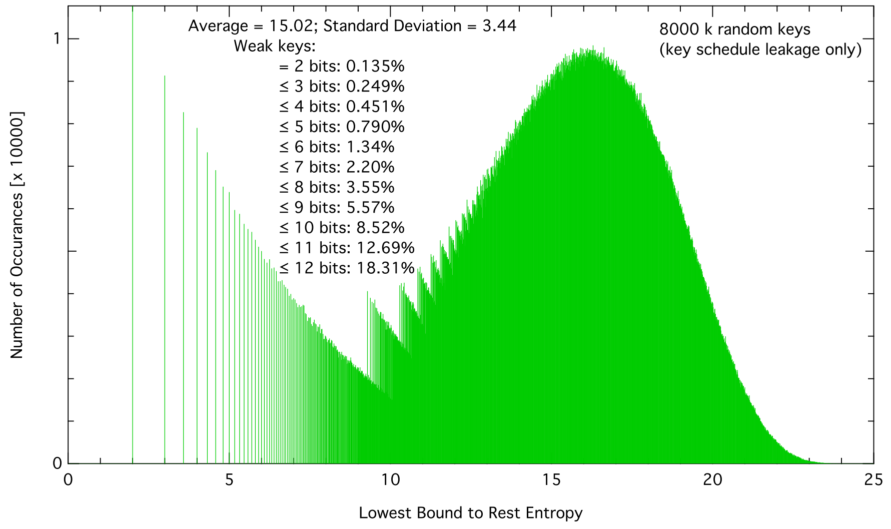


Fig. 42. Distributions for the lowest bound to the rest entropy based on the Hamming distance leakage model in the key schedule. The tall peaks to the left represent the weak keys. Some 0.8% of all keys have a lowest bound of 5 bits or less.

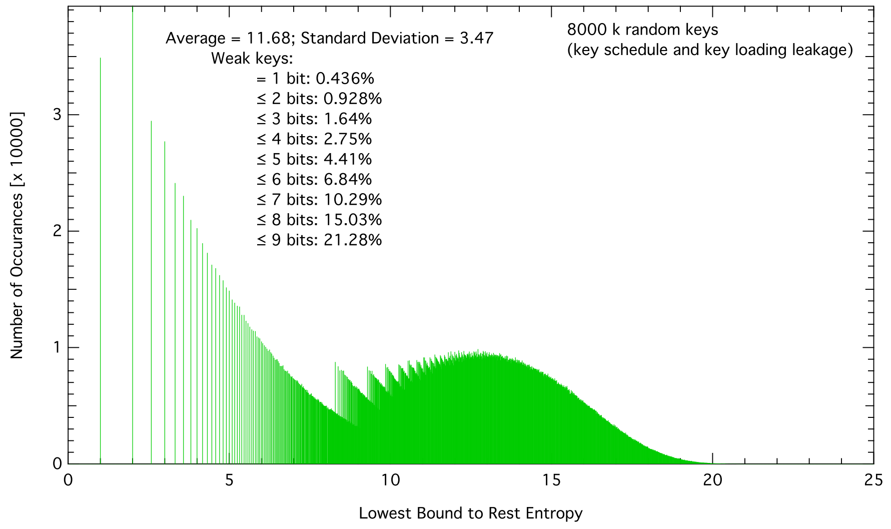


Fig. 43. Distributions for the lowest bound to the rest entropy based on the Hamming distance leakage model in the key schedule as well as the key loading stage ($\oplus$ of the left and right halves of the key).

With such numbers it seems a difficult problem to compute all possible collisions for all possible DES keys and so, in order to tackle this, in what follows a Monte–Carlo simulation is performed.

In Fig. 42 we show the histogram of lowest possible rest entropies — i.e., the number of keys mapping to the same 15–tuple by way of dist15 — based on Monte–Carlo simulations with a randomly chosen set of 8000 k DES keys. It turns out there are quite a few collisions for most 15–tuples, leading to an average lowest possible rest entropy of 15.02 bits. But then again, there are also quite a few weaker keys having very few collisions, e.g., 0.135% of all DES keys have a lowest bound to the rest entropy of only 2 bits instead of the average 15 bits, and as many as 3.55% of all DES keys have only 8 bits or less.[15]

In Fig. 11, bottom graph, we had presented the distribution of the rest entropies of a randomly chosen set of DES keys, using the Average Ranking as the sorting criterion in the key enumeration strategy. Using the same set of DES keys, we calculated the rest entropies predicted by the 15–tuple approach, resulting in a correlation between the two data sets as high as 27.8%. In the case of 7–bit templates, i.e., Fig. 23, this correlation is slightly increased to 28.5%. From this we conclude that the Hamming distance does not perfectly model the leakage observed in the attacked device. But it does go a long way towards explaining why there is such a spread in the distribution seen in Fig. 11 and elsewhere — it has to do with the prolific existence of weak keys.

Furthermore, the average rest entropy of Fig. 11 is 46.16 bits, and if we restrict the analysis only to all weak keys with a rest entropy of 2 bits, this average comes down to 41.61 bits, or to 40.72 bits for the 9–bit templates. This is still substantially higher than the 15.02 bits predicted by the 15–tuple approach, and is most likely explained by a combination of various effects, such as measurement noise in the single traces and — likely more importantly — the ring–based search strategies presented in this paper being far from optimal.[16]

The analysis of a little over ten thousand weak keys seems to suggest that they are characterised by extreme values in the 15 entries of the 15–tuple, and we conjecture that the keys with the highest rest entropy — so the strongest, most resisting keys in the context of this attack — have a 15–tuple with no variation at all, i.e., where all entries equal 24, resulting in a rest entropy of 25.03 bits. From this it seems likely an attacker will know *in advance* whether a key will be weak or not, simply by looking at the 15–tuple that the side–channel attack produces

[15] Some examples of weak keys are `04867C2A1CA4CA36`, `E4D8C44676649046`, `847CA2EC0658BEE6`, `90D086CED4683CDA`, `60D2A80A4452023C`, `22923290164A66D4`, `BE580E120CCCE886`, `20C63E52BECC064C`, `F044BC0E7EC8DA5A`, `F4BC522C745CE806`, `FC2EC3F2A20D58E4`. The bit ordering is done here counting from left to right, with every 8th bit being a (dysfunctional) parity bit. With this, the last weak key in the list above, `FC2EC3F2A20D58E4`, encrypts the plaintext `D576B72B236B94E7` to the ciphertext `383580233DD8715B`. The trace fairing best in this 15–tuple analysis as well as in the 5–bit template analysis of Fig. 11 has the DES key `4BC85940D7C5EEE5`, with TraceID 4935963 in Table 1. Hence, this latter key seems like a good test key to choose in future analysis.

[16] The best trace presented in Table 1, TraceID 4935963, is based on a weak key with a rest entropy of 2 bits. If the total Hamming distance leakage were perfect, according to this table, the ring–based search algorithm based on Average Ranking still yields a rest entropy of 8.93 bits.

as the most likely candidate in the ranking list.[17] Consequently, the attacker is in a position to know whether it is worth spending the brute–force effort or not, and this will effectively reduce the aggregated effort when performing attacks at a large scale on many targets where not all targets need to be compromised in the end, like in banking.

It is further interesting to see how these results improve when additional leakage is present that can be exploited as well. The idea here is that any further leakage mechanism will provide additional constraints to the possible key collisions and thereby reduce the rest entropy. To give an example, in [1] we had analysed the very same smart card device that is the target of the current paper, and we had found it to leak information when loading the DES key into the coprocessor kernel. The model of the leakage mechanism was as follows: Due to the width of the internal memory buses, the key is loaded sequentially in two steps into the coprocessor kernel through a 4 byte register interface. First, e.g., the left 4 bytes get loaded, and then the right 4 bytes, overwriting the first 4 bytes in doing so. In the absence of any countermeasures this results in a typical leakage of the Hamming distance of 28 bit vectors.[18] So, in essence, what we are looking at here is a new mapping function, $\mathrm{dist16}(k)$, where the DES key k is now mapped to 16–tuples, with the first 15 entries being the same as before, whilst the last entry is the Hamming distance of the key–loading leakage just described.

To assess the impact of this additional leakage mechanism, let us again assume perfect Hamming distance leakage of the key loading and not the more powerful leakage by value. Fig. 43 shows the resulting distribution, again based on a Monte–Carlo simulation with randomly chosen DES keys.[19] Compared with Fig. 42 the average lowest bound to the rest entropy is lowered to 11.68 bits, meaning that the additional key–loading leakage has reduced the average brute–force effort by about 3.34 bits. This is also visible in the weak keys — now as many as 15% of all keys have a lower bound to the rest entropy of only 8 bits or less.

Consequently, for a two–key TDES the average brute–force effort has come down by close to 7 bits when combining these two leakages. Additional leakage mechanisms would result in additional constraints to the key, and hence in additional entries appended to the 16–tuple.

[17] Even if this is not quite the correct key, as weak keys tend to be neighbours, this initial guess will not be too far off the truth.

[18] Most of this additional leakage maps directly to one or the other of the two B rings, but some of these additional $\oplus$ bit relations connect the C and D Register — i.e., the two C rings — with each other and thus fix the possible values that the two C rings can have relative to each other. As a result, it is not possible anymore to flip all bits in each C ring separately and still have a valid, undistinguishable combination. It is only possible to flip all 56 bits together at the same time. Hence, the minimally possible rest entropy is not 2 bits anymore, but just 1 bit.

[19] Please note that the binning used to create the histograms of Figs. 42 and 43 is the same. The saw–tooth–like structures seen in both these histograms are an artefact of the discrete, fixed–width binning used.

In summary, we have defined a method for determining all DES keys k having a given value of dist15(k). In a context where an attacker can retrieve the Hamming distances between consecutive DES round keys perfectly, we have devised an optimal strategy that gives us directly the list of all keys having the same, given 15–tuple. Obviously, this method can be extended by searching for nearest neighbours to the 15–tuples in case the side–channel analysis does not yield perfect results. Furthermore, we have generalised this method to combine this key–scheduling leakage with additional leakage seen during the loading of the DES key.

Most importantly, all these results mean that unless precautions are taken to avoid the usage of weak keys in a given eco system — which does not seem practical — then in a risk assessment for this attack the worst case of a weak key needs to be assumed, and hence also such a weak key must be used for performing any vulnerability analysis. The following Section does not adhere to this principle, though, as the fixed key used for further analysis was chosen before the existence of weak keys became known to the authors.

6 Brute-Force Results for a Given Fixed Key (5–Bit Templates)

So far, all results were obtained using random keys in the Exploitation Phase, which is perfectly ok — and in fact from a statistics point of view actually preferable — since we have been looking at single–trace attacks. In this Section, though, we will study the statistics when the DES key is kept constant in this attack phase, usually based on 32 k attacks.

To start with, we will first analyse the single–trace statistics for a given fixed key having a total Hamming distance of 355, and then move on to study how these statistics improve when using more than one trace during the Exploitation Phase of the attack. The key chosen is not a weak key according to Sec. 5, but rather an average key having a lowest bound to its rest entropy of 15.79 bits, which is just a little higher than the average found in Fig. 42 — hence our reference to it being an average key as far as this leakage model is concerned.

6.1 Brute-Force Results for a Given Fixed Key using Single Traces in Exploitation Phase

In Fig. 44 we have plotted the distributions of the Average Rankings of the C and D Register, which have average values of 11.25 and 7.43, respectively, and corresponding standard deviations of 2.85 and 2.15. Comparing this with the case of random keys, Fig. 8, we find good overall agreement of the shapes of the distributions, except for the average value for the C Register having shifted to a somewhat larger value.

In Fig. 45 the distribution of the rest entropy is shown for a single–key DES, with an average rest entropy of 46.69 bit, which needs to be compared with Fig. 12. Again, these distributions are rather similar. This then translates to a

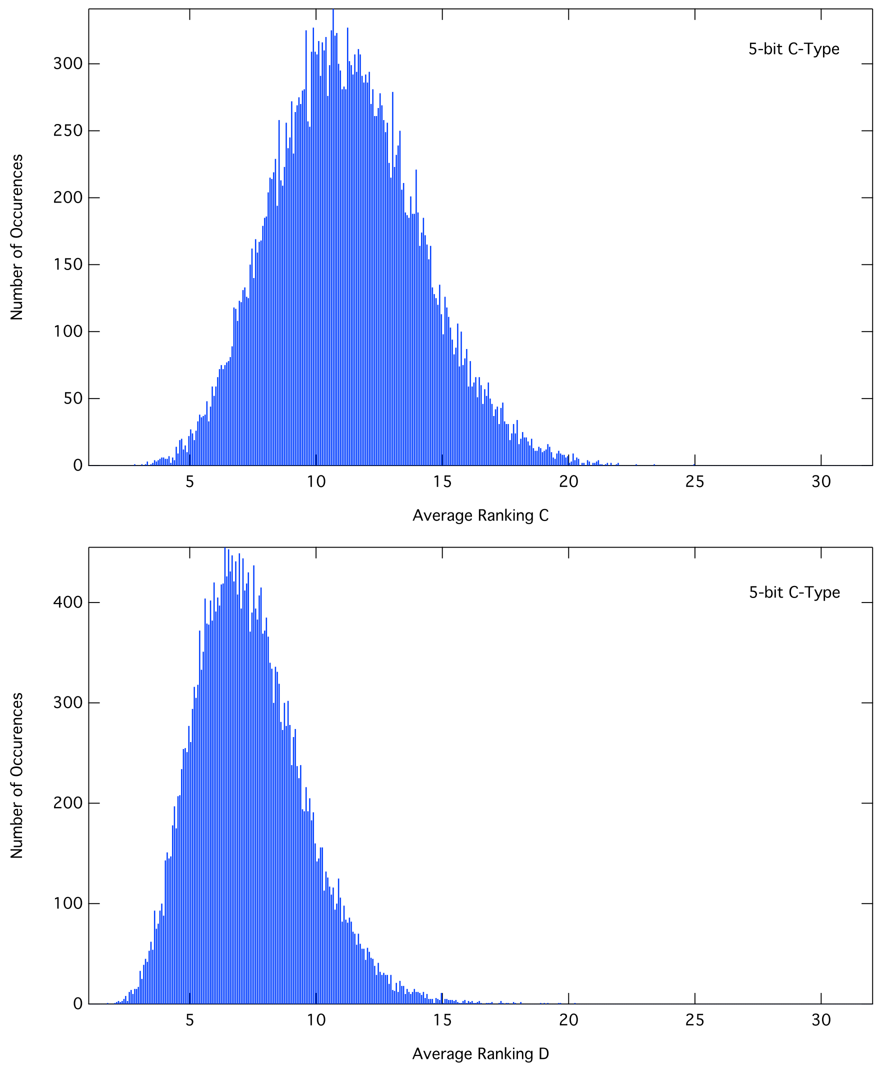


Fig. 44. Distributions found for the Average Ranking of the correct key for C and D Registers for a given fixed key.

distribution of the rest entropy for a two–key TDES, as shown in Fig. 46, where the average rest entropy is 97.59 bit.

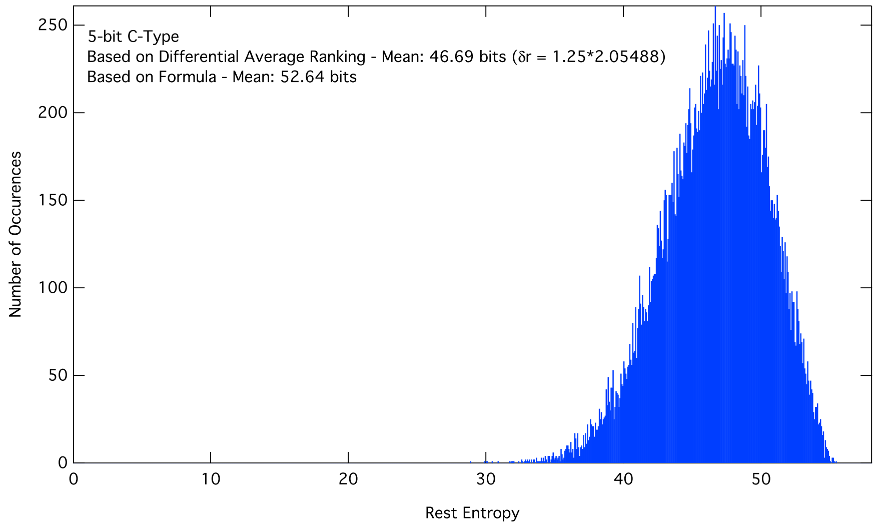


Fig. 45. Distributions for the rest entropy when sorting for Differential Average Ranking with $\delta r = 1.25 \times 2.05488$ for a given fixed key.

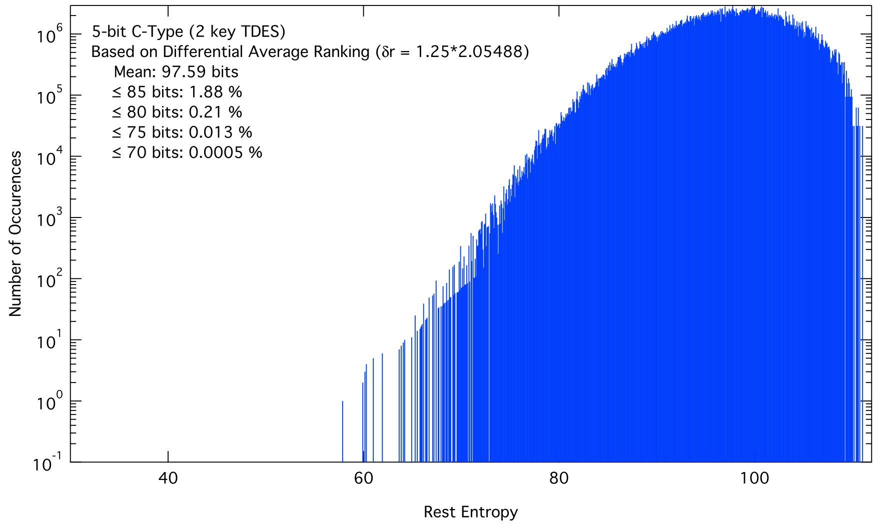
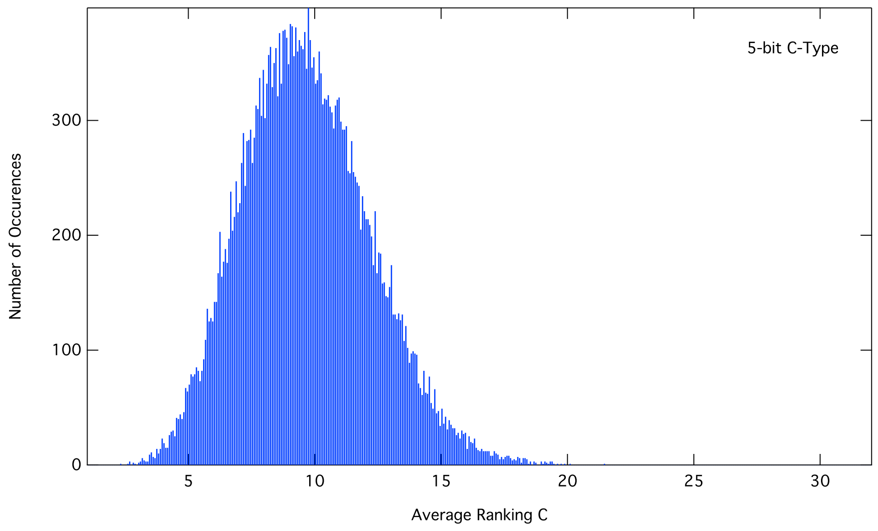


Fig. 46. Distribution of the rest entropy for a 2–key TDES when sorting according to Differential Average Ranking with $\delta r = 1.25 \times 2.05488$ for a given fixed key.

6.2 Brute-Force Results for a Given Fixed Key using a Few Traces in the Exploitation Phase

In this Section we analyse how the results improve further when we let go of the requirement of a *single* trace, and instead allow more than one trace (i.e. N traces) to be used in the Exploitation Phase. There are different approaches



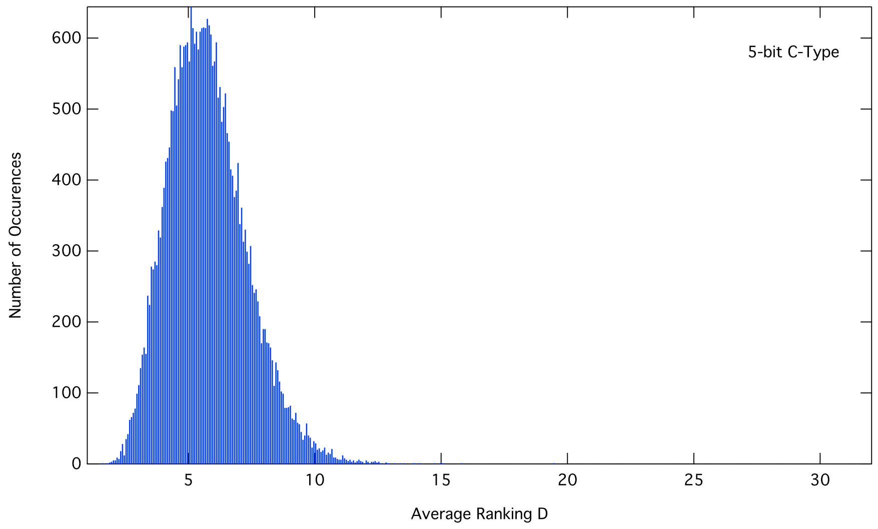


Fig. 47. Distributions found for the Average Ranking of the correct key for C and D Registers for a given fixed key, but using $N = 2$ traces in Exploitation, rather than a single trace. Clearly, the distributions have shifted to the left and are sharper, indicating that the attack effort will be less than for a single–trace attack.

possible how to perform a template attack when more than one trace is available in the Exploitation Phase. In this paper we have opted to perform a simple average over all N traces first and then subsequently use this average trace as a pattern in the pattern–template matching. An alternative approach would be to perform the pattern–template matching for each trace individually, and then average over the results. Either way, since we are averaging over a number of traces

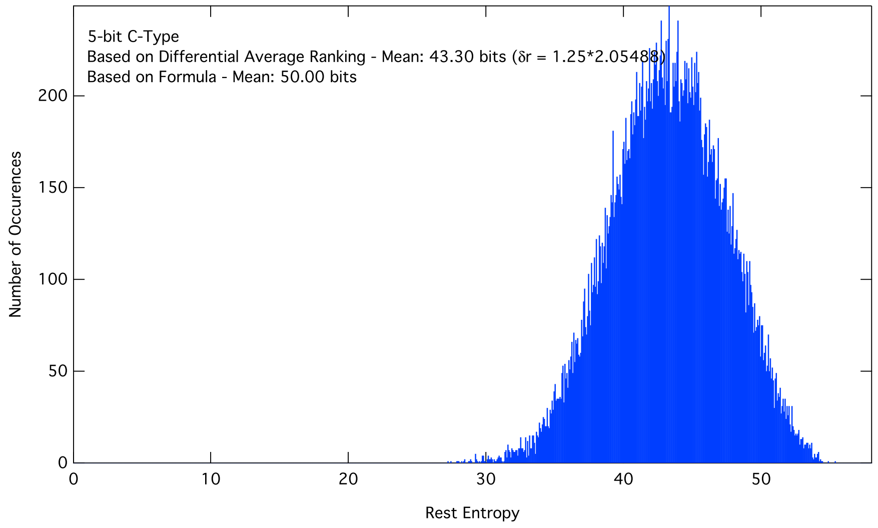


Fig. 48. Distributions for the rest entropy when sorting for Differential Average Ranking with $\delta r = 1.25 \times 2.05488$ for a given fixed key, but using $N = 2$ traces in Exploitation, rather than a single trace.

for each individual brute–force effort, the detrimental effect of outliers should be much weaker. In all scenarios analysed in this Section we have performed 32 k attacks, except for $N \geq 64$, where not enough traces were available.

In Fig. 47 we have plotted the distributions of the Average Rankings of the C and D Registers when $N = 2$ traces are used in the Exploitation Phase. Now the respective average values are 9.73 and 5.80, with their standard deviations being 2.52 and 1.56, respectively. So, in particular the attack on the D Register has improved compared to the single–trace attack.

In Fig. 48 the corresponding distribution of the rest entropy is shown for a single–key DES, using $N = 2$ traces in the Exploitation Phase, with an average rest entropy of 43.30 bit. It is apparent that this distribution is much more symmetric than those seen for single–trace attacks. This distribution then translates to a distribution of the rest entropy for a two–key TDES, as shown in Fig. 49, where the average rest entropy has come down to 91.29 bit, compared to 97.54 bit for the single–trace attack of Fig. 46.

As N increases in Figs. 50 through to 73, all these statistical values keep improving. A summary is given in Table 3. For $N = 16$, regardless of the particular threat metric chosen, the two–key TDES attack is clearly below 60 bits of effort if further contributions are all properly accounted for as well — i.e., total Hamming distance leakage, and key k_1 leaking twice as much as key k_2 in two–key TDES. With reference to the latter: Whilst for small values of N the average rest entropy for a single–key DES, $\bar{E}_{1D}$, decreases roughly by 3 bits when doubling N, this effect becomes smaller for larger N. Finally, please note that all the results in this Section are based on 5–bit templates. As shown in Sec. 4.4,

Table 3. Based on 5–bit templates, various statistics parameters for the random key ensemble (R) of Sec. 4.1, and the $N = 1, 2, 4, 8, 16, 32...$ fixed key ensemble. The results for $\bar{E}_{1D}$ and $\bar{E}_{2D}$ do not yet include the effect of the additional Hamming distance leakage, nor the fact that for a two–key TDES the outer key k_1 leaks twice as much. For $\bar{E}_{2D}$ this means a further reduction of the brute–force effort of about 6–10 bits, depending on N. When using 11–bit templates it is estimated that a further reduction by 2 bits will occur. $N = 1...32$ are all based on 32 k attacks with $\delta r = 1.25 \times 2.05488$; $N = 64$ is based on 24 k attacks, $N = 128$ is based on 12 k attacks, $N = 256$ is based on 6 k attacks, and $N = 512$ is based on 3 k attacks, all with $\delta r = 2.05488$.

N	$\bar{r}_C$	$\bar{r}_D$	σ_C	σ_D	Corr_{r_C,r_D}	$\bar{E}_{1D}$	σ_{1D}	$\bar{E}_{2D}$	≤ 70		≤ 65		≤ 60		≤ 55	
						[bits]	[bits]	[bits]	%	[bits]	%	[bits]	%	[bits]	%	[bits]
R	9.86	7.78	3.01	2.67	-0.041	45.66	4.63	96.48	5.26e-2	80.89	8.04e-3	78.60	1.10e-3	76.48	1.21e-4	74.66
1	11.25	7.43	2.85	2.15	-0.14	46.69	3.73	97.59	5.19e-4	87.56	1.34e-5	87.83	6.09e-7	87.29		
2	9.73	5.80	2.52	1.56	-0.21	43.30	4.14	91.29	5.18e-2	80.92	2.64e-3	80.21	1.11e-4	79.78		
4	8.20	4.63	2.10	1.10	-0.28	39.37	4.21	83.51	2.20	75.51	1.92e-1	74.02	5.43e-3	74.17		
8	6.89	3.92	1.66	0.75	-0.33	35.45	3.80	75.17	24.17	72.05	5.00	69.32	3.27e-1	68.26		
16	5.94	3.50	1.25	0.52	-0.37	32.17	2.99	67.65	70.54	70.50	35.60	66.49	5.62	64.15		
32	5.29	3.25	0.92	0.37	-0.39	29.80	2.12	61.93	96.09	70.06	81.34	65.30	33.18	61.59	1.45	61.11
64	4.95	3.13	0.71	0.28	-0.37	28.22	1.85	58.48	99.51	70.01	94.65	65.08	73.07	60.45	12.97	57.95
128	4.75	3.10	0.55	0.22	-0.29	27.49	1.34	56.46	100	70	99.54	65.01	92.77	60.11	26.44	56.92
256	4.63	3.11	0.43	0.18	-0.21	27.20	1.03	55.54	100	70	99.93	65.00	98.96	60.02	36.22	56.47
512	4.55	3.12	0.36	0.15	-0.15	27.17	0.83	55.27	100	70	100	65	99.93	60.00	41.00	56.29

results improve a little when using larger template sizes. For 11–bit templates, the rest entropies are reduced by about 2 bits compared to 5–bit templates.

Furthermore, in Sec. 3 we had established that the total Hamming distance leakage will on average reduce the brute–force effort by roughly 2.5 bits per DES key. This was based on single–trace attacks, though. The question now is whether this rule is also applicable for $N > 1$, as it is in principle conceivable that in such a case there is a different bias in the Hamming distances at the top ranks in the ranking lists, which could have an effect on this rule. However, it turns out this is not the case. In Table 4 we provide results like we did in Table 1 for $N = 1$, but now for $N = 512$, and we still find that the brute–force effort goes down by about 2.5 bits per DES key when applying a boundary condition for the total Hamming distance with a maximal error of ± 7. So, this rule of thumb still holds. Even better, as N increases the accuracy will improve with which the total Hamming distance can be determined, and hence the maximal error gets reduced. Consequently, for $N \gg 1$ much more than 2.5 bits per DES key can be recovered this way. If the total Hamming distance is perfectly known, as shown earlier, this gain can be as high as 6–7 bits per DES key, or about 13 bits for a two–key TDES, but we did not analyse this additional contribution yet.

These results clearly demonstrate that as expected the template attack becomes considerably stronger when abandoning the single–trace approach and instead using a few 10 to a 100 traces in the Exploitation Phase. In this context it should be noted that countermeasures against fault attacks often require mul-

Table 4. Same as Table 1, but for $N = 512$ and a few selected traces. Taking advantage of the total Hamming distance leakage with an accuracy of ± 7 yields roughly 2.5 bits, as before, and thus this gain does not appear to depend on N.

Trace ID	T'pl. Size	C– and D–Register			Exact HW		Exact HW ± 7	
	[bits]	$E^C_{r_{\max}}$	$E_{r_{\text{average}}}$	$E_{r_{\max}}$	$E_{r_{\text{average}}}$	$E_{r_{\max}}$	$E_{r_{\text{average}}}$	$E_{r_{\max}}$
1455104	5	**34.90**	**35.73**	**34.90**	**29.05**	**28.17**	**32.90**	**32.06**
95603	5	**33.13**	**31.03**	**32.95**	**24.66**	**26.52**	**28.57**	**30.43**
1302016	5	**27.31**	**28.77**	**27.43**	**22.24**	**20.88**	**26.09**	**24.63**

tiple calculations of the DES with the same arguments, and thus a few of these "additional" traces may be readily available.

On the other hand, we also find that these improvements seem to level off and the distributions get somehow "stuck" at some fixed values when N increases further. Becoming sharper, yes, but not moving all the way to the left. The reason for this is precisely the collisions of keys discussed in Sec. 5, where the key chosen in this Section gives a lowest bound to the rest entropy of 15.79 bits.[20] This is still substantially lower than what was achieved in Fig. 72 for 512 traces used for the Exploitation Phase, where the average is still as high as 27.17 bits. Even when using as many as 1.5 M traces in Exploitation phase, the resulting rest entropy is as high as 28.65 bits for Average Ranking, using an offset of $\delta r = 1.25 \times 2.05488$ in the search depths of the C and D Register. A smaller value of $\delta r = 0.5 \times 2.05488$ yields 25.92 bits, which is still much higher, coming down to 21.05 bits if we assume to know the total Hamming distance exactly. For such many traces in the Exploitation Phase it cannot be argued anymore that there is still noise present that degrades the results. Hence, we need to conclude that the search strategies chosen for the brute–force search are not optimal yet.[21]

Obviously, some further improvements are possible here by finding a more optimal value for δr, which among others will depend on N. This should have a noticeable impact as the distributions for the Average Rankings of the C and D Register become sharper and shaper, and less overlapping. It is also apparent from Table 3 that the (negative) correlation between the Average Rankings of the C and D Register seems to increase strongly with N, at least for moderate values of N. This can also be used to improve the key–enumeration scheme further.

And finally, it should be noted that neither the Average Ranking nor the Maximum Ranking is the optimal search strategy. This becomes apparent when studying Fig. 74, where we have plotted the results of these two search strategies

[20] When allowing an error in one of the entries of the 15–tuples of Sec. 5 by ± 1, then this lowest bound to the rest entropy is increased to 20.71 bits.

[21] Note that this difference cannot be explained by arguing that it is not correct to assume perfect Hamming distance leakage in the leakage model assumed in the 15–tuple analysis of Sec. 5. If more information leaks, the lowest rest entropy will be even lower, not higher.

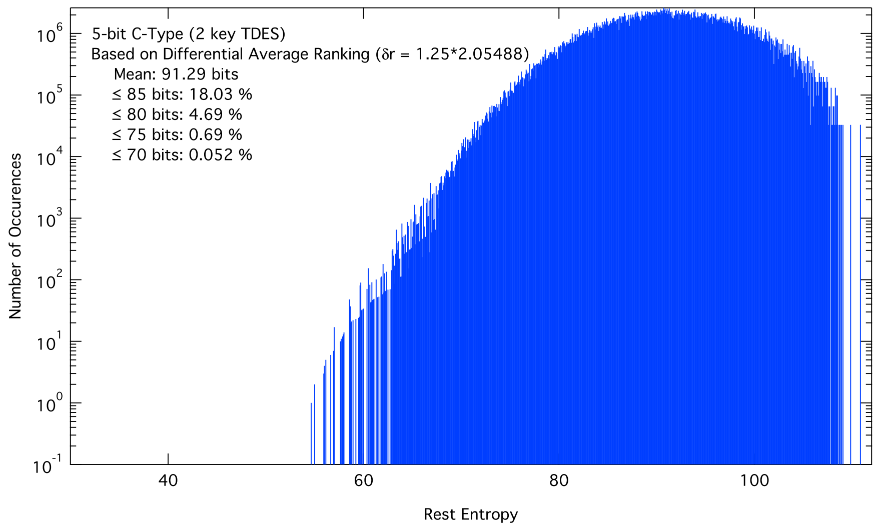
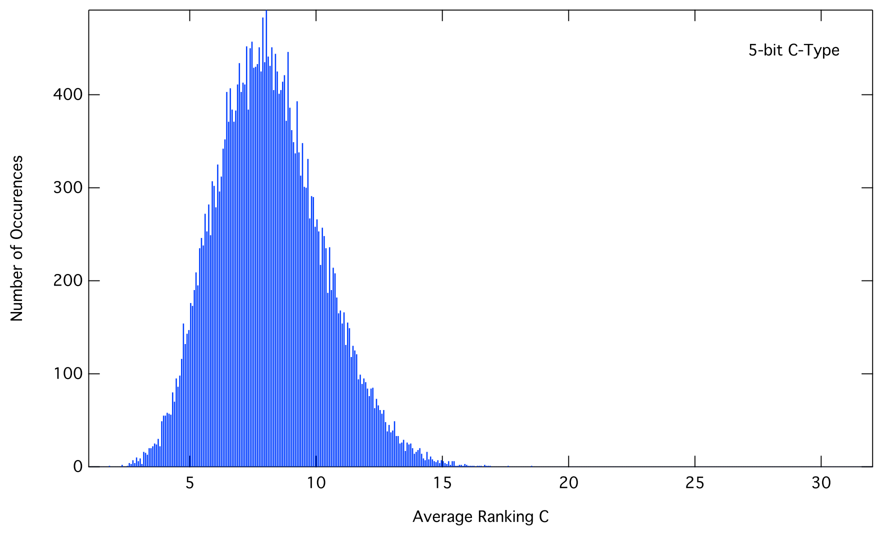


Fig. 49. Distribution of the rest entropy for a 2–key TDES when sorting according to Differential Average Ranking with $\delta r = 1.25 \times 2.05488$ for a given fixed key, but using $N = 2$ traces in Exploitation, rather than a single trace.

against each other. There are regions where one strategy clearly outperforms the other, and the other way round. So, a hybrid between these two strategies seems to be a better choice.



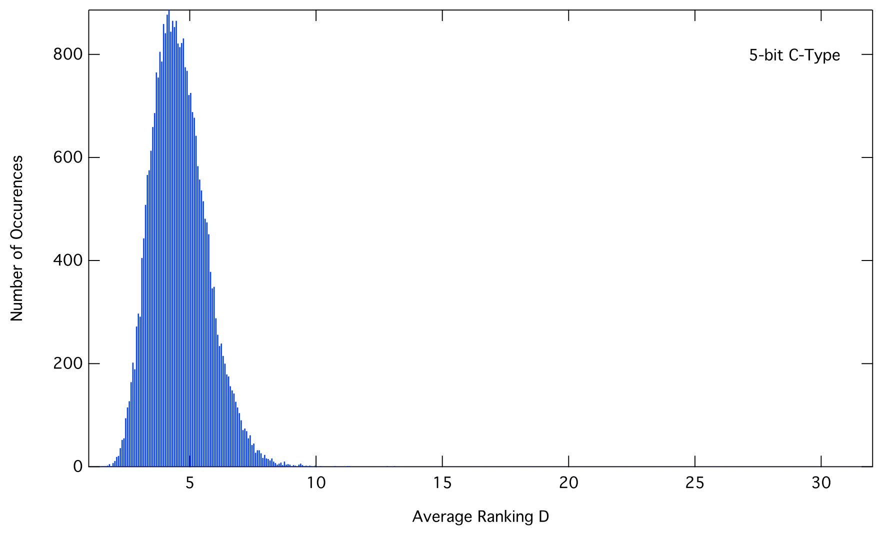


Fig. 50. Distributions found for the Average Ranking of the correct key for C and D Registers for a given fixed key, but using $N = 4$ traces in Exploitation, rather than a single trace. Clearly, the distributions have shifted to the left and are sharper, indicating that the attack effort will be less than for a single–trace attack.

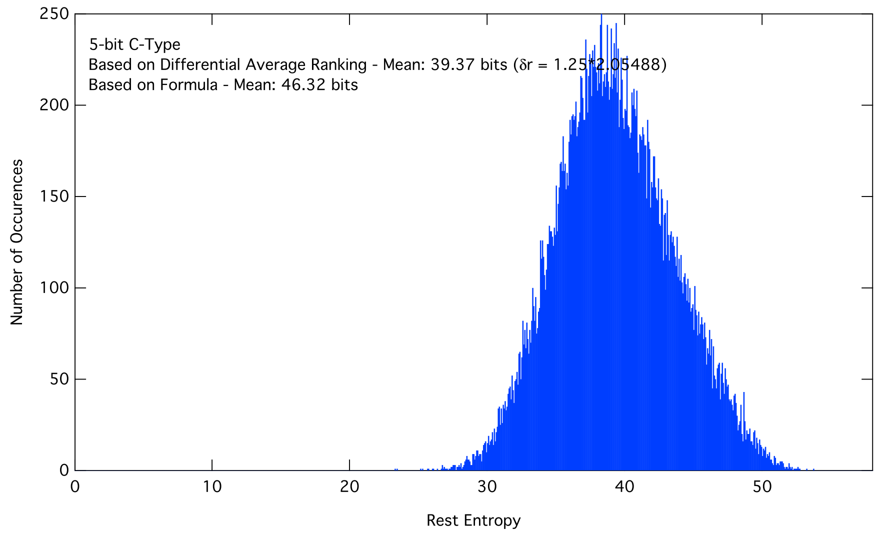


Fig. 51. Distributions for the rest entropy when sorting for Differential Average Ranking with $\delta r = 1.25 \times 2.05488$ for a given fixed key, but using $N = 4$ traces in Exploitation, rather than a single trace.

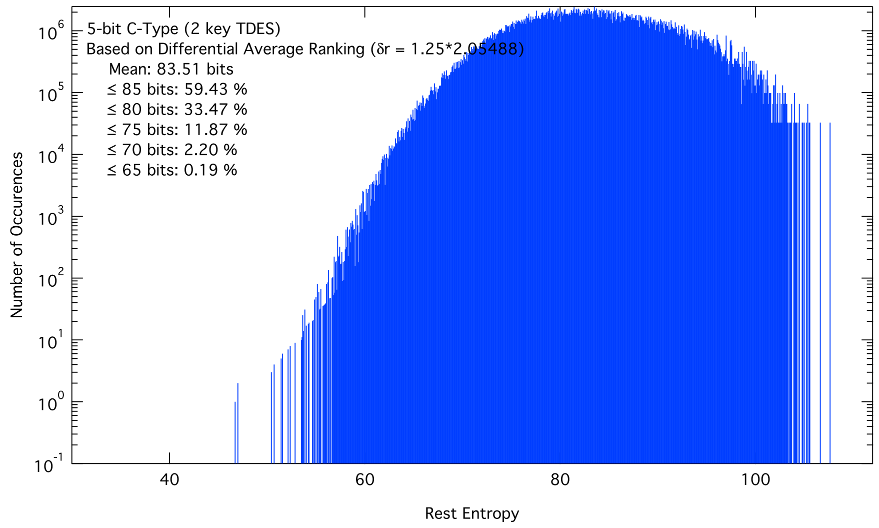
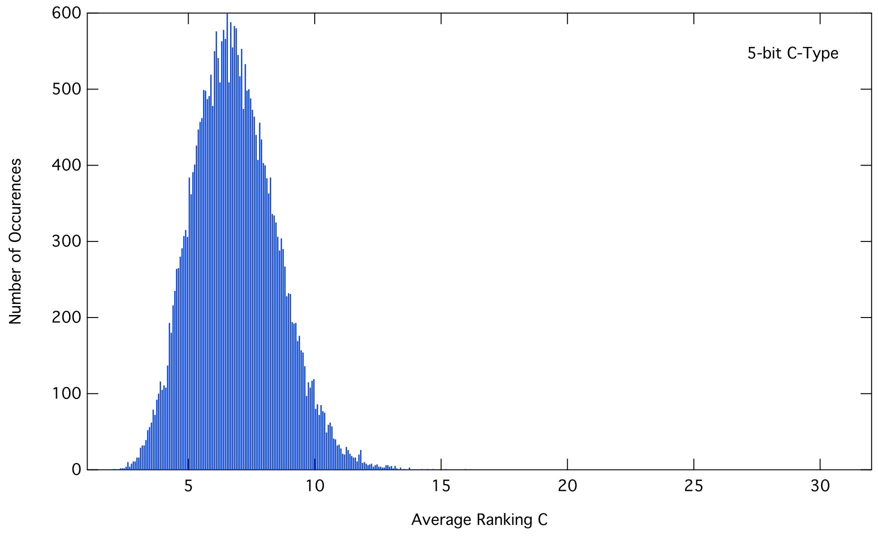


Fig. 52. Distribution of the rest entropy for a 2–key TDES when sorting according to Differential Average Ranking with $\delta r = 1.25 \times 2.05488$ for a given fixed key, but using $N = 4$ traces in Exploitation, rather than a single trace.



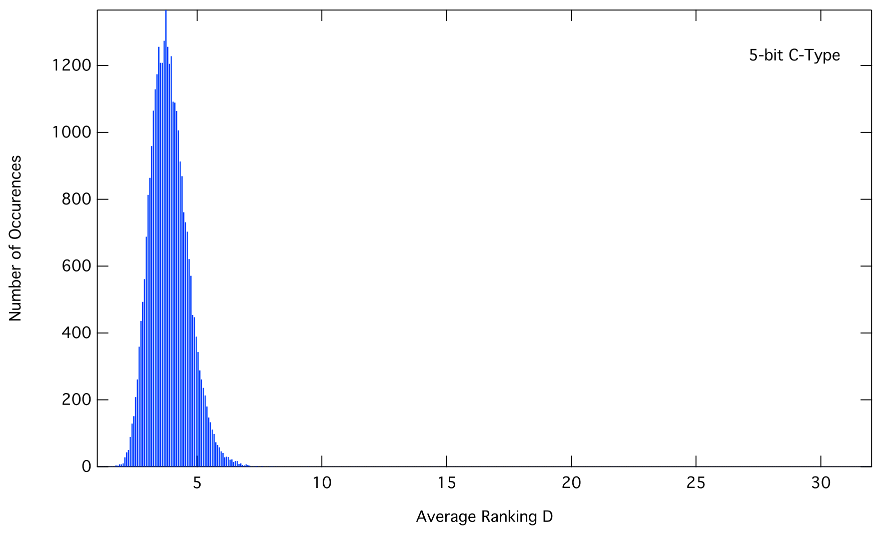


Fig. 53. Distributions found for the Average Ranking of the correct key for C and D Registers for a given fixed key, but using $N = 8$ traces in Exploitation, rather than a single trace. Clearly, the distributions have shifted to the left and are sharper, indicating that the attack effort will be less than for a single–trace attack.

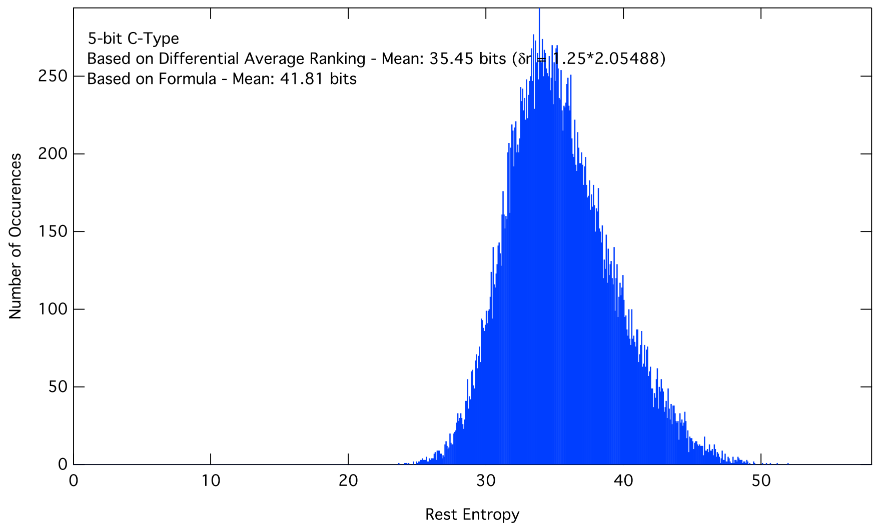


Fig. 54. Distributions for the rest entropy when sorting for Differential Average Ranking with $\delta r = 1.25 \times 2.05488$ for a given fixed key, but using $N = 8$ traces in Exploitation, rather than a single trace.

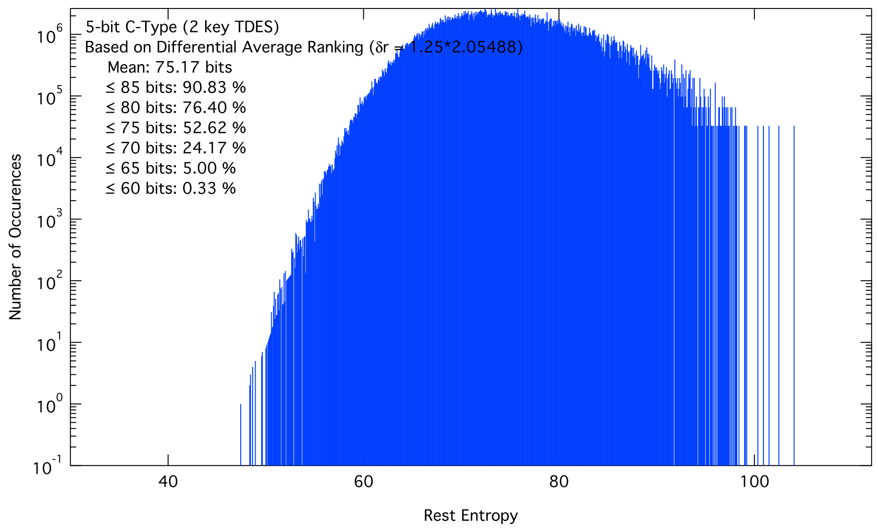


Fig. 55. Distribution of the rest entropy for a 2–key TDES when sorting according to Differential Average Ranking with $\delta r = 1.25 \times 2.05488$ for a given fixed key, but using $N = 8$ traces in Exploitation, rather than a single trace.

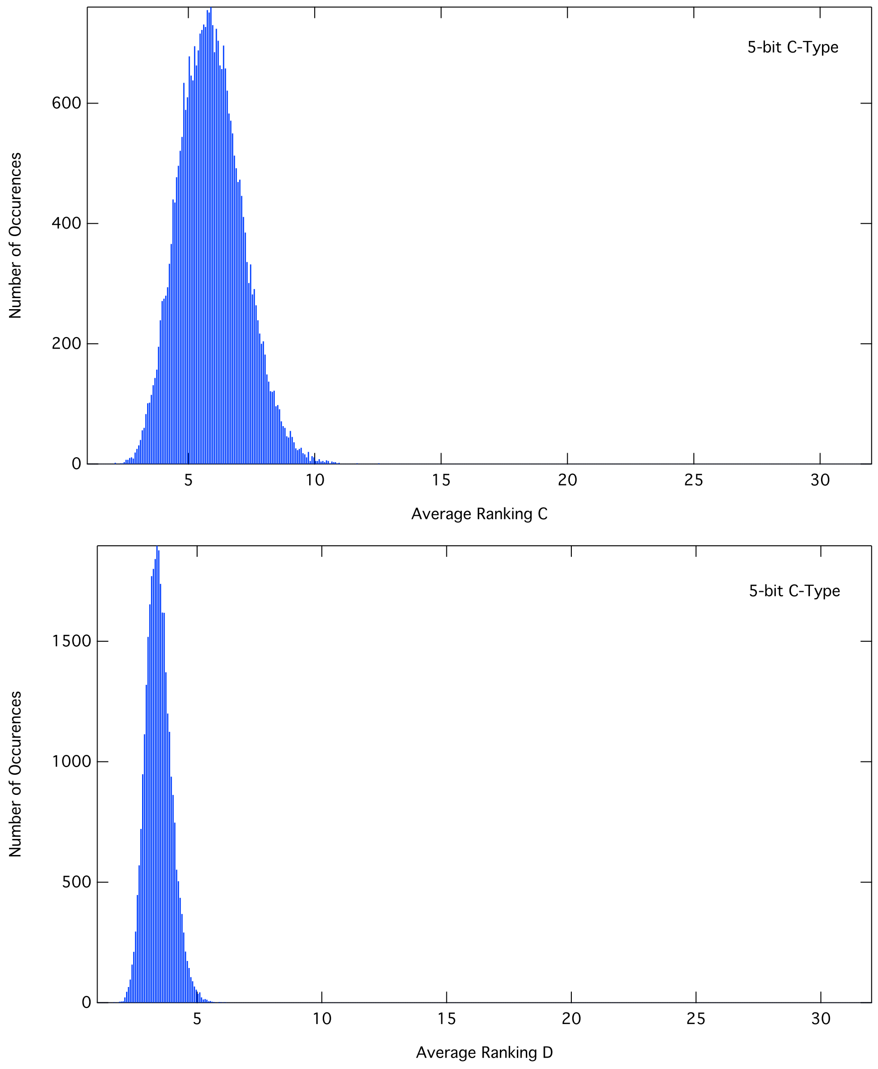


Fig. 56. Distributions found for the Average Ranking of the correct key for C and D Registers for a given fixed key, but using $N = 16$ traces in Exploitation, rather than a single trace. Clearly, the distributions have shifted to the left and are sharper, indicating that the attack effort will be less than for a single–trace attack.

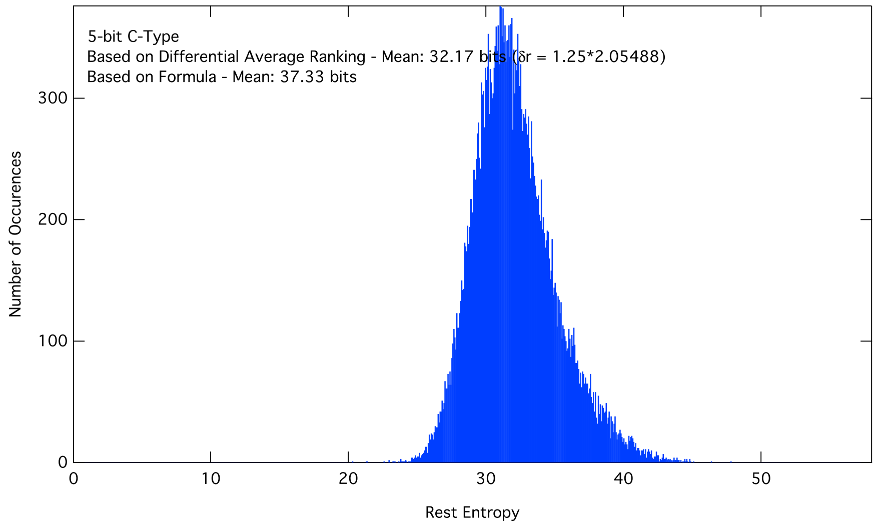


Fig. 57. Distributions for the rest entropy when sorting for Differential Average Ranking with $\delta r = 1.25 \times 2.05488$ for a given fixed key, but using $N = 16$ traces in Exploitation, rather than a single trace.

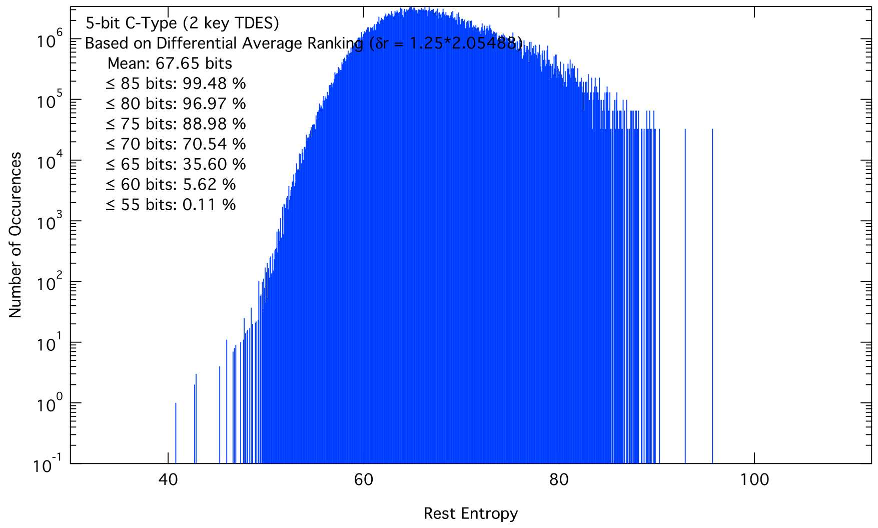


Fig. 58. Distribution of the rest entropy for a 2–key TDES when sorting according to Differential Average Ranking with $\delta r = 1.25 \times 2.05488$ for a given fixed key, but using $N = 16$ traces in Exploitation, rather than a single trace.

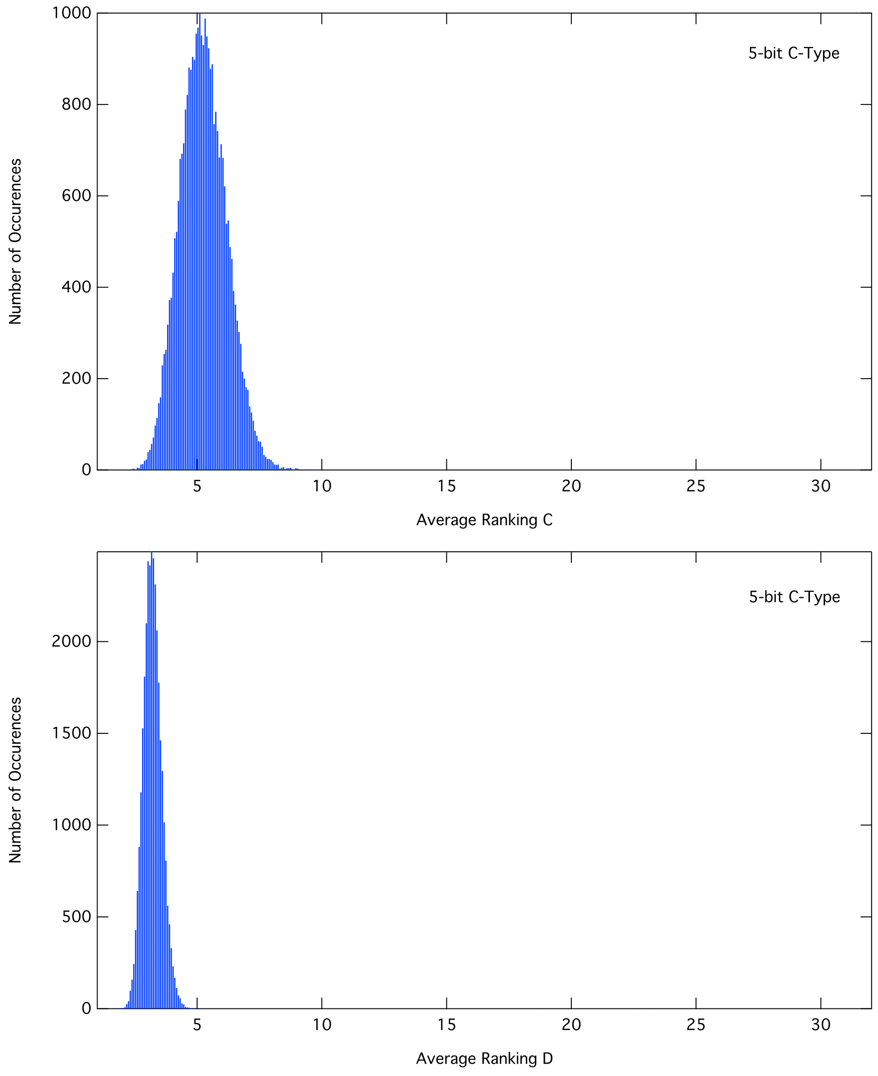


Fig. 59. Distributions found for the Average Ranking of the correct key for C and D Registers for a given fixed key, but using $N = 32$ traces in Exploitation, rather than a single trace. Clearly, the distributions have shifted to the left and are sharper, indicating that the attack effort will be less than for a single–trace attack.

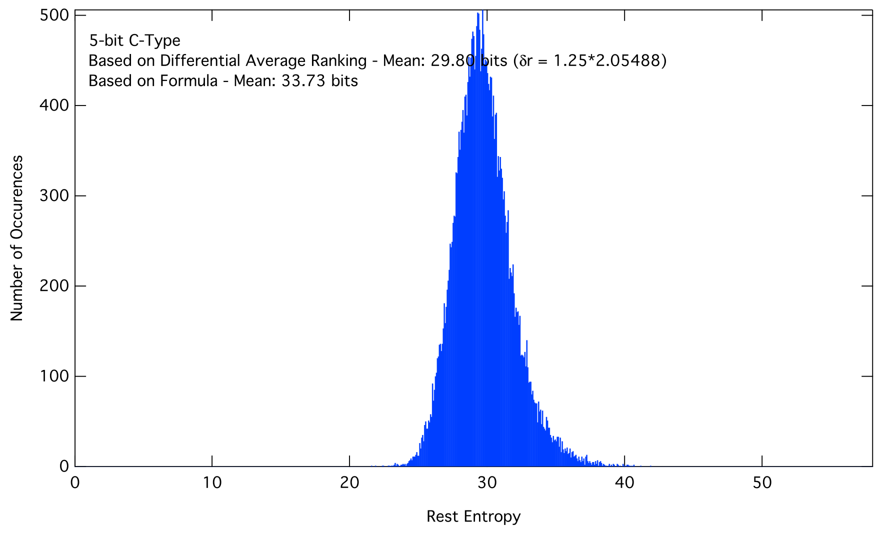


Fig. 60. Distributions for the rest entropy when sorting for Differential Average Ranking with $\delta r = 1.25 \times 2.05488$ for a given fixed key, but using $N = 32$ traces in Exploitation, rather than a single trace.

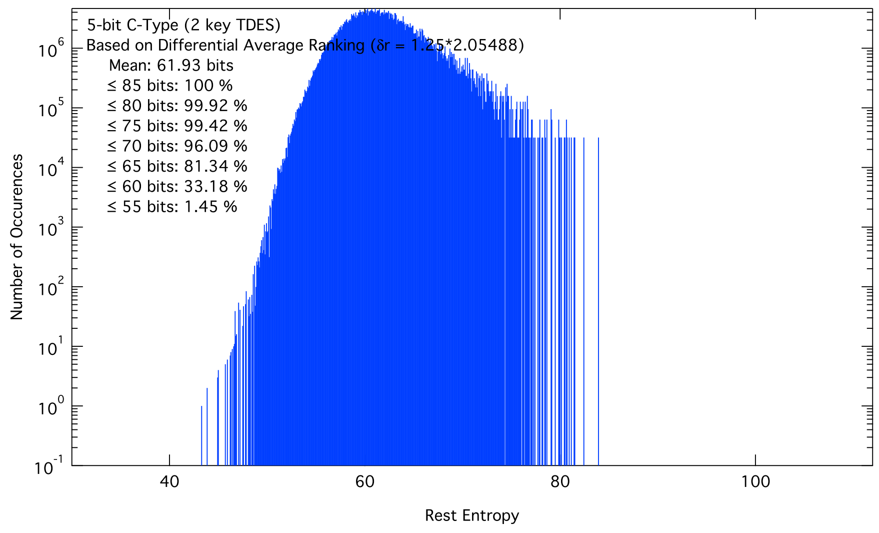


Fig. 61. Distribution of the rest entropy for a 2–key TDES when sorting according to Differential Average Ranking with $\delta r = 1.25 \times 2.05488$ for a given fixed key, but using $N = 32$ traces in Exploitation, rather than a single trace.

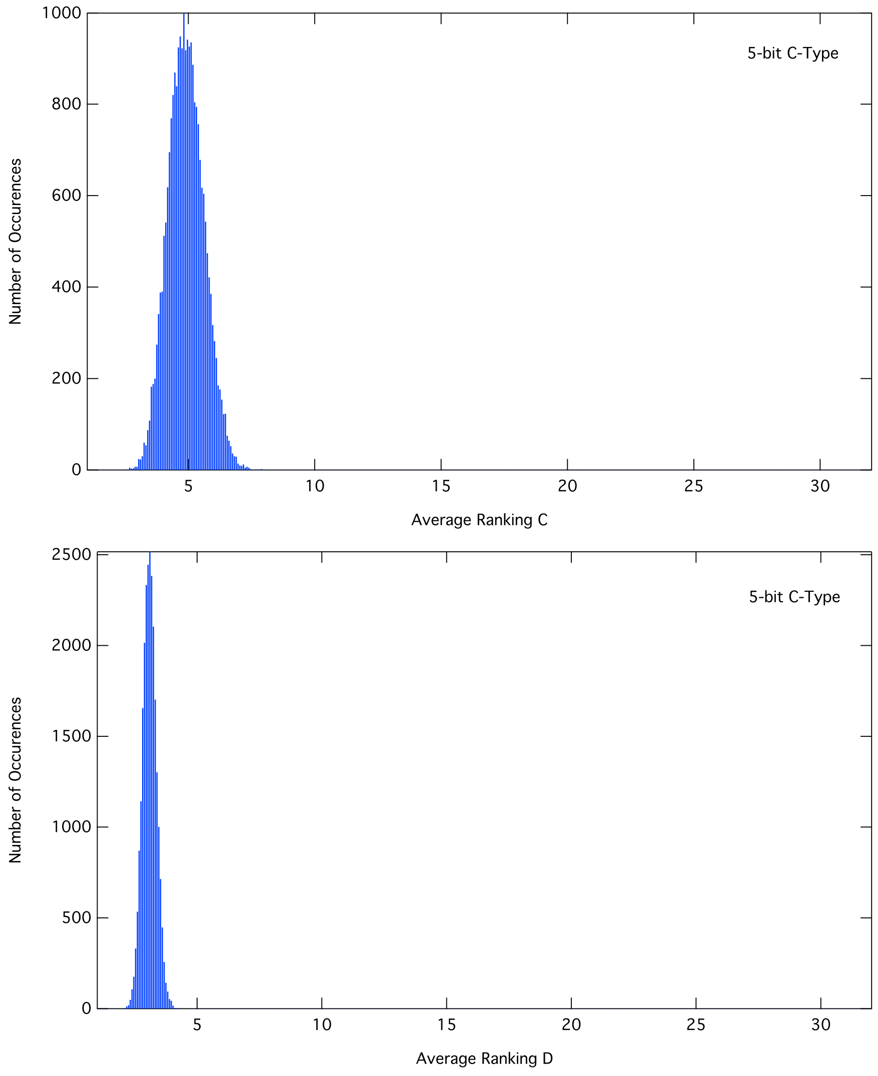


Fig. 62. Distributions found for the Average Ranking of the correct key for C and D Registers for a given fixed key, but using $N = 64$ traces in Exploitation, rather than a single trace. Clearly, the distributions have shifted to the left and are sharper, indicating that the attack effort will be less than for a single–trace attack.

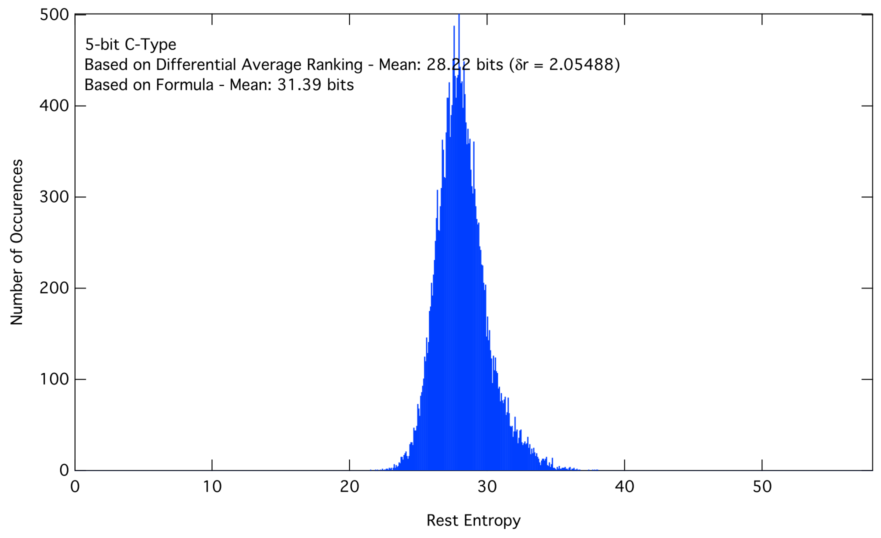


Fig. 63. Distributions for the rest entropy when sorting for Differential Average Ranking with $\delta r = 2.05488$ for a given fixed key, but using $N = 64$ traces in Exploitation, rather than a single trace.

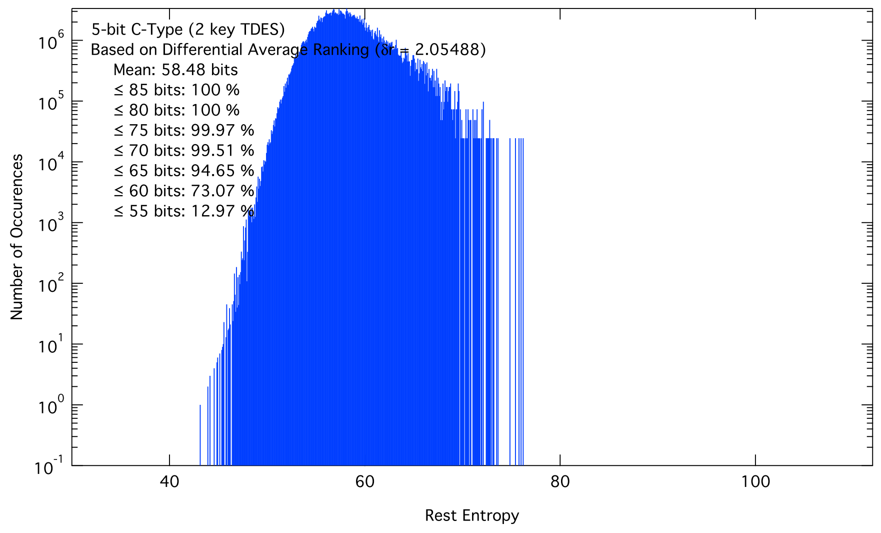
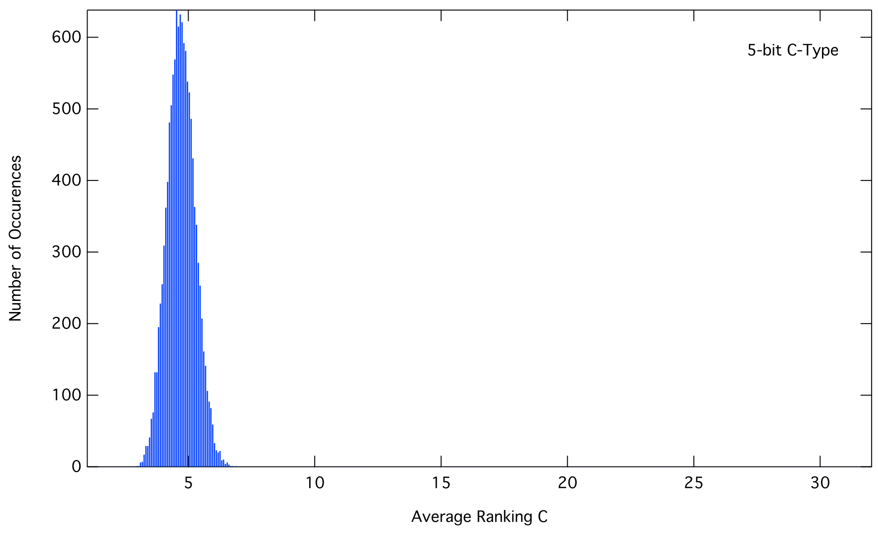


Fig. 64. Distribution of the rest entropy for a 2–key TDES when sorting according to Differential Average Ranking with $\delta r = 2.05488$ for a given fixed key, but using $N = 64$ traces in Exploitation, rather than a single trace.



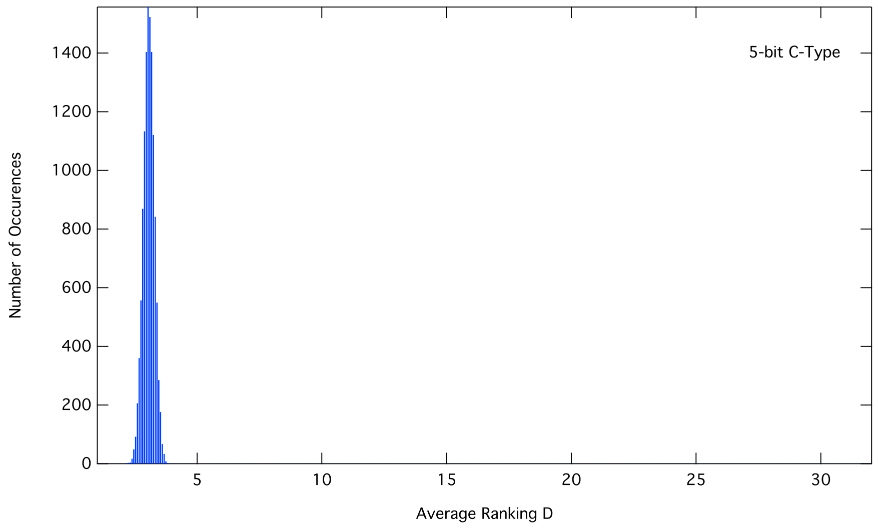


Fig. 65. Distributions found for the Average Ranking of the correct key for C and D Registers for a given fixed key, but using $N = 128$ traces in Exploitation, rather than a single trace. Clearly, the distributions have shifted to the left and are sharper, indicating that the attack effort will be less than for a single–trace attack.

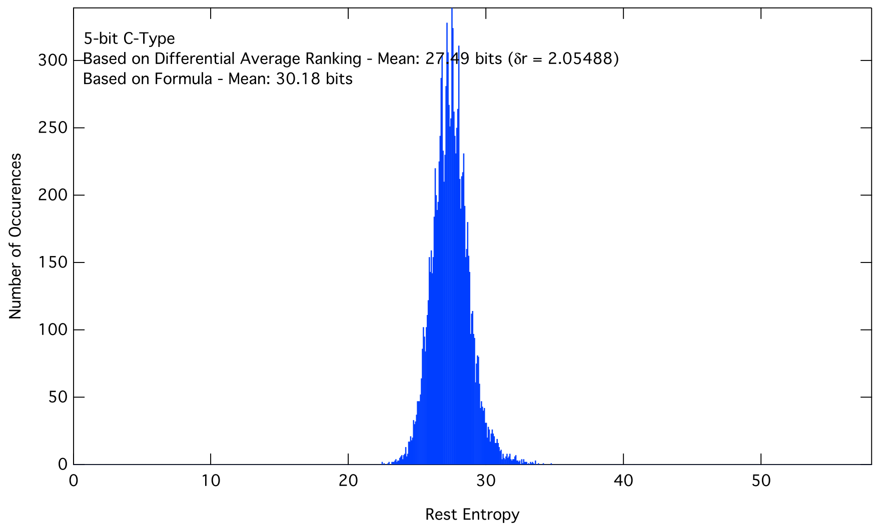


Fig. 66. Distributions for the rest entropy when sorting for Differential Average Ranking with $\delta r = 2.05488$ for a given fixed key, but using $N = 128$ traces in Exploitation, rather than a single trace.

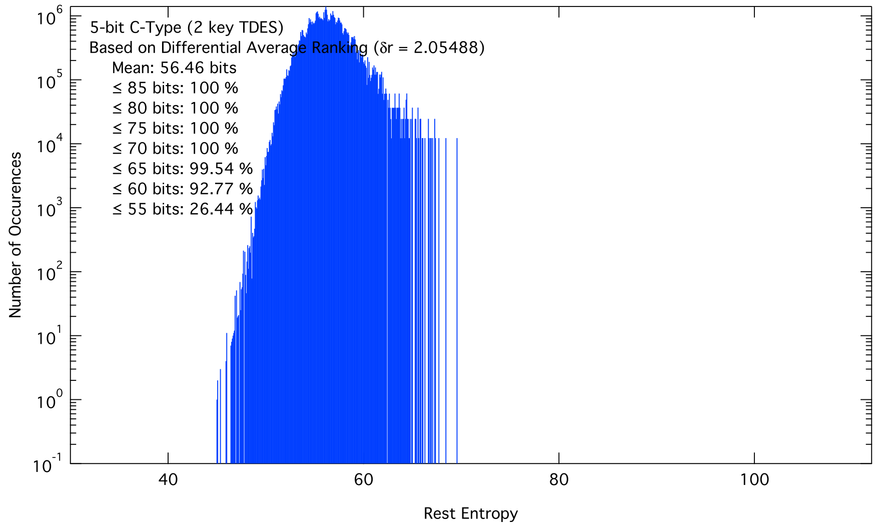


Fig. 67. Distribution of the rest entropy for a 2–key TDES when sorting according to Differential Average Ranking with $\delta r = 2.05488$ for a given fixed key, but using $N = 128$ traces in Exploitation, rather than a single trace.

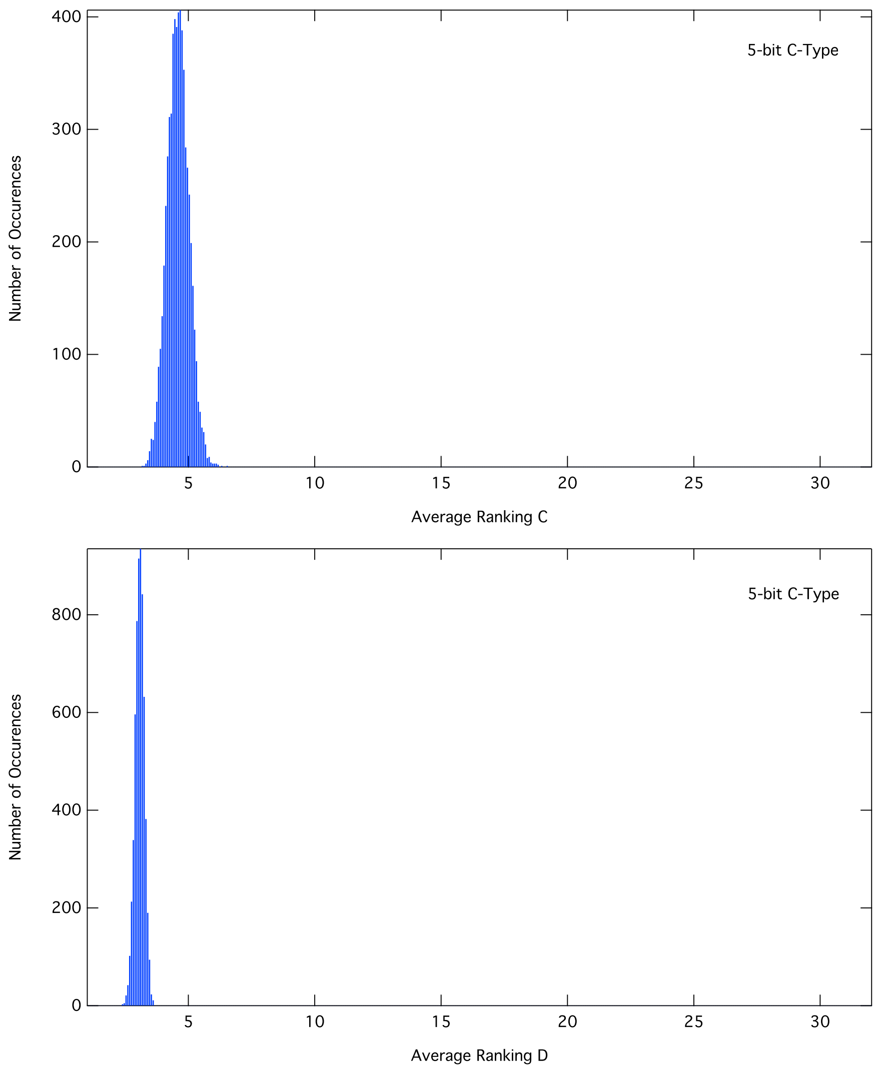


Fig. 68. Distributions found for the Average Ranking of the correct key for C and D Registers for a given fixed key, but using $N = 256$ traces in Exploitation, rather than a single trace. Clearly, the distributions have shifted to the left and are sharper, indicating that the attack effort will be less than for a single–trace attack.

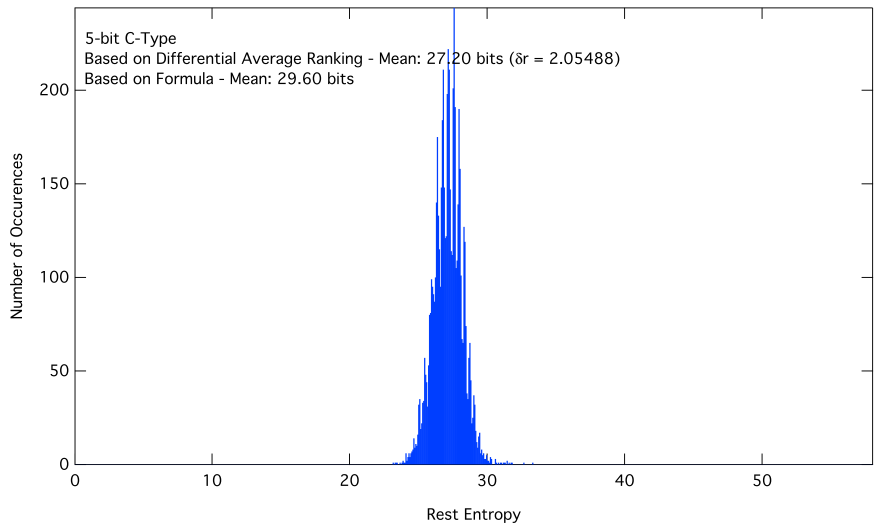


Fig. 69. Distributions for the rest entropy when sorting for Differential Average Ranking with $\delta r = 2.05488$ for a given fixed key, but using $N = 256$ traces in Exploitation, rather than a single trace.

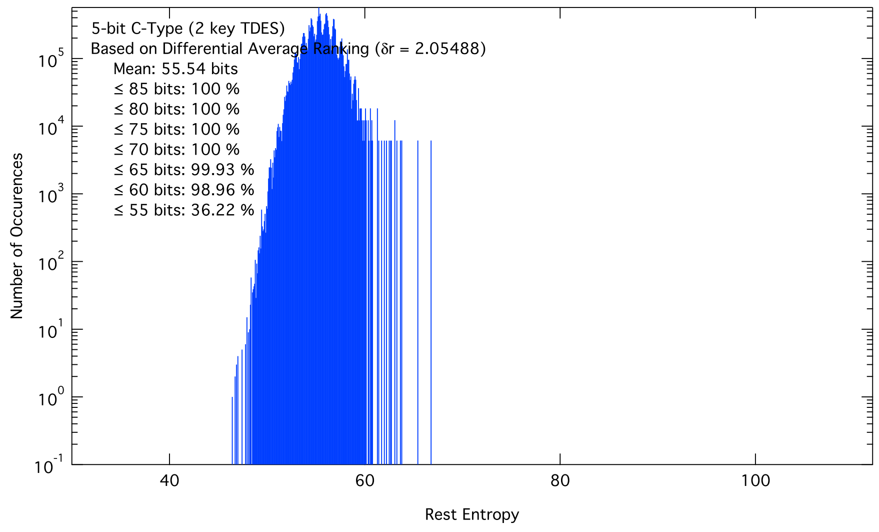
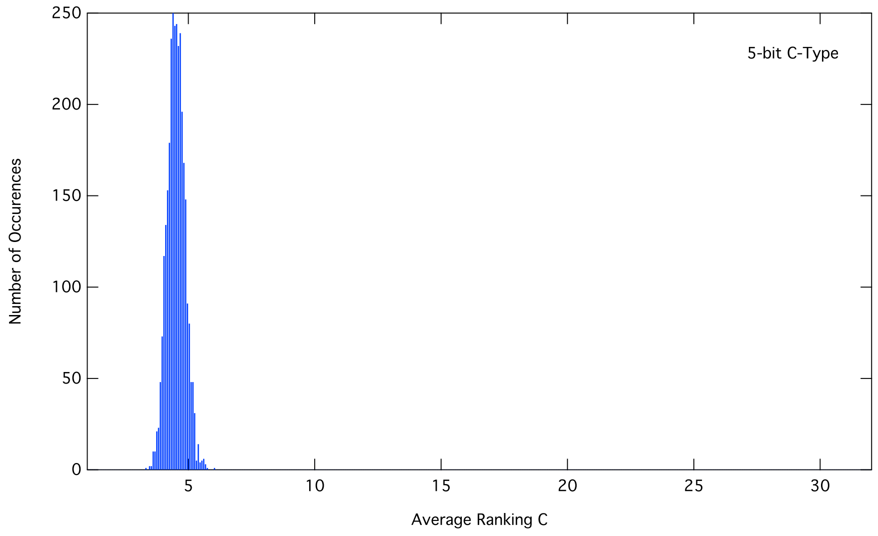


Fig. 70. Distribution of the rest entropy for a 2–key TDES when sorting according to Differential Average Ranking with $\delta r = 2.05488$ for a given fixed key, but using $N = 256$ traces in Exploitation, rather than a single trace.



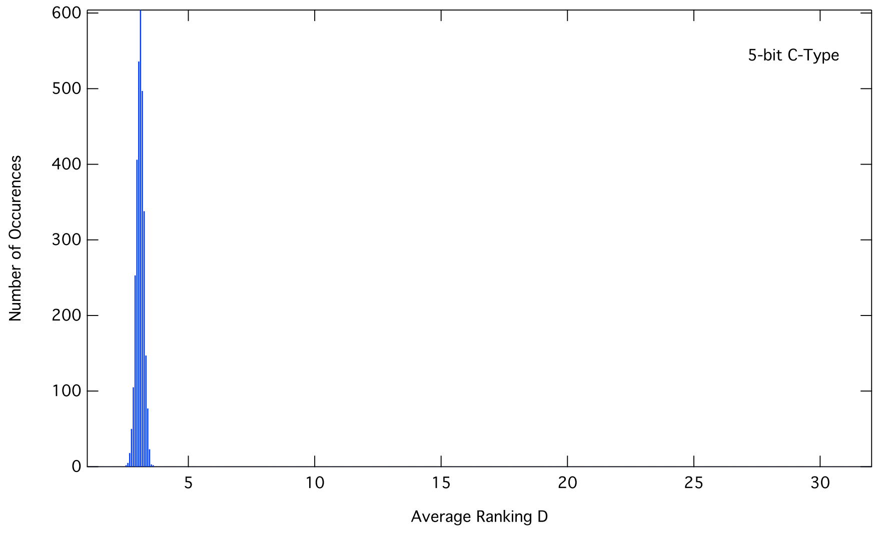


Fig. 71. Distributions found for the Average Ranking of the correct key for C and D Registers for a given fixed key, but using $N = 512$ traces in Exploitation, rather than a single trace. Clearly, the distributions have shifted to the left and are sharper, indicating that the attack effort will be less than for a single–trace attack.

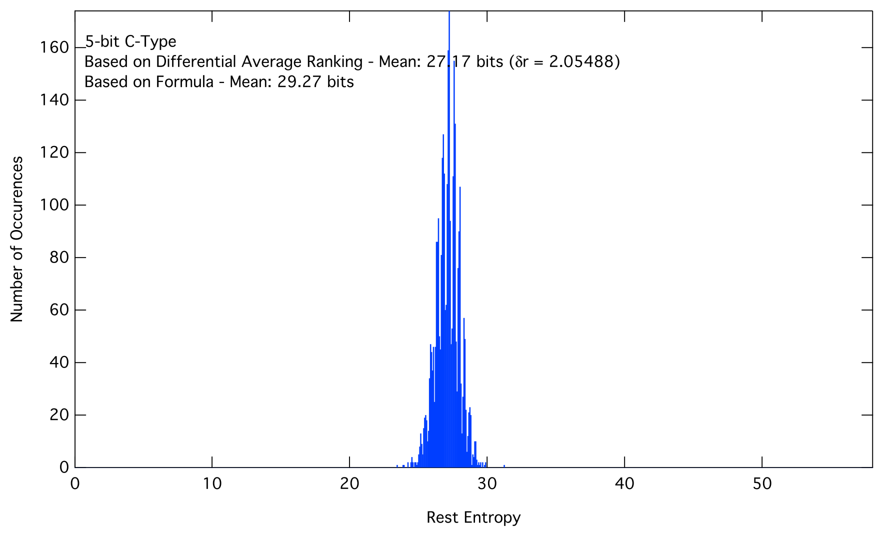


Fig. 72. Distributions for the rest entropy when sorting for Differential Average Ranking with $\delta r = 2.05488$ for a given fixed key, but using $N = 512$ traces in Exploitation, rather than a single trace.

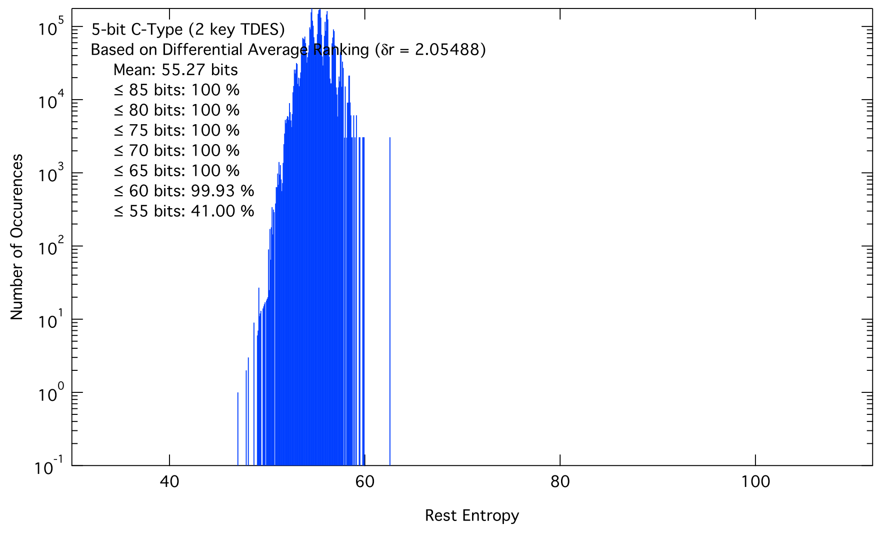


Fig. 73. Distribution of the rest entropy for a 2–key TDES when sorting according to Differential Average Ranking with $\delta r = 2.05488$ for a given fixed key, but using $N = 512$ traces in Exploitation, rather than a single trace.

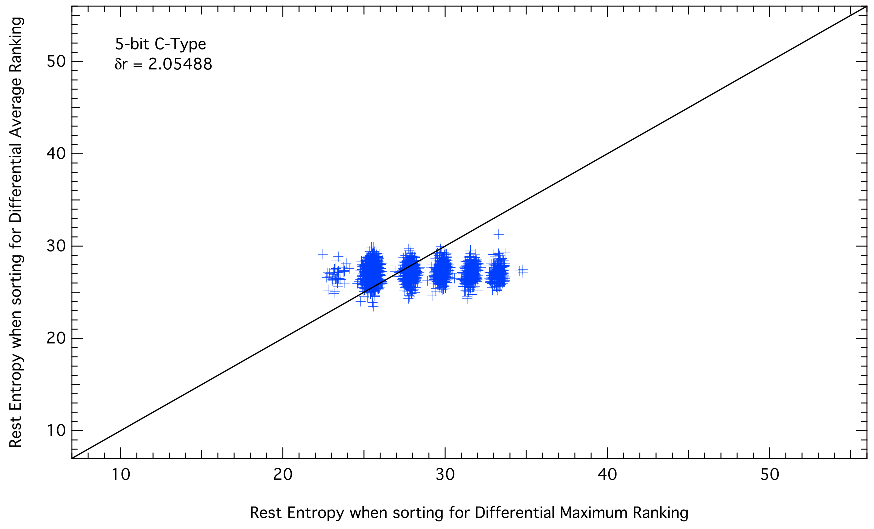


Fig. 74. Differential Average Ranking versus Differential Maximum Ranking with $\delta r = 2.05488$ for a given fixed key, but using $N = 512$ traces in Exploitation, rather than a single trace. There is virtually no correlation between the two sorting methods — only 0.4%. Clearly, for lower values of the rest entropy, Differential Maximum Ranking outperforms the Differential Average Ranking, whilst this is not the case for larger values thereof. This is an indication that neither of them is an optimal search strategy.

7 Conclusions

In conclusion, we have presented an analysis of the entire attack path, including the remaining brute–force effort, when exploiting the key scheduling leakage found in the DES hardware coprocessor of a well–known and widely deployed smart card chip. We used a template attack involving 28 overlapping templates distributed across two so–called C rings, which in turn relate to the C and D Register of the DES key schedule. The leakage occurs when updating the round key registers. Although this leakage is very strong, there is still a final brute–force step required to recover the DES key fully, and the usage of overlapping templates calls for new approaches to finding efficient key enumeration schemes for this step. The purpose of this paper was to establish how much of a brute–force effort is in fact required. In order to have more meaningful statistical results, we have performed the entire attack a couple of 100 k times on the same target device, but using different (EM) traces and different search parameters.

This attack does not work equally well for all DES keys and there is, in fact, a reason why a final brute–force step is required for this attack. An analysis of the key collisions in the key scheduling reveals that on average a brute–force effort of some 15 bits remains if only the Hamming distance leaks (but perfectly so), yet for some *weak keys* it goes down to a very few bits only. In this idealised, conservative leakage model the average brute–force effort to break a single DES is as low as 15 bits. The results shown in this paper indicate, though, that even within this conservative model there is still room for substantial improvements to the key search strategy.

A number of factors affect the strength of the attack. The attack improves with increasing template size, as expected, but not dramatically so (of the order of 2 bits in the rest entropy). More important is the choice of the key enumeration scheme, i.e., in which order to search through the key ranking lists provided by the template attack. Two main schemes have been analysed: Ordering the ranking lists by Maximum Ranking (i.e., the maximum ranking found on a given C ring), and ordering the lists by Average Ranking (i.e., the average ranking for a given C ring). The latter yields much better results on average, but clearly it is not the best search strategy yet, as there appears to be a cross over between these two schemes for very leaky traces. By and large these results are consistent with the more predictive analysis made in [2], except for very large template sizes.

When combining the original single–trace template attack of [2] with a template attack on the total Hamming distance itself, the single–trace attack can be improved by roughly 2.5 bits per DES key, or 5 bits in the case of two–key TDES.

Most work has been done using *single* traces during the Exploitation Phase, where we found a particularly leaky single–key DES trace that had only a little over 13 bits rest entropy. More representatively, for 9–bit templates, the *average* rest entropy for a single–key DES was found to be $45.5 - 2.5 = 43$ bits, and for a two–key TDES $95.9 - 2 \times 2.5 - 3.4 = 87.5$ bits. However, these distributions of rest entropy show a long tail towards smaller values, which may be exploitable for an attack with even less effort. These long tails are due to weak keys.

To be more precise, the *average* rest entropies are the relevant parameters characterising the brute–force effort, when the attack focuses on a single target device and only a single trace is available. However, in an attack scenario where a couple of target devices are available and all can be attacked simultaneously, and when it does not matter which target device will yield in the end to gain profit, then the relevant parameter is not the average rest entropy, anymore, but the long tail of the rest–entropy distribution to the left side, i.e., to smaller values — the weak keys. It turns out that in this case it is advantageous for the attacker to start many attacks in parallel and to stop when the first attack is successful. It is harder to create a proper statistics for this case, but the effective rest entropy when adding up the efforts of all parallel attacks is of the order of 68 to 80 bits only, depending on how many target devices are being attacked in parallel. For instance, 6.4% of all traces require an effort of up to $85 - 2 \times 2.5 - 3.4 = 76.6$ bits. Moreover, it should be noted that an alternative attack scenario of the second type is to use only *one* target device, and then to attack *many* single–trace measurements thereof in parallel. In essence, this approach is a tactics to take advantage of the existence of weak keys.

Single–trace attacks such as those described so far are particularly devastating as they cannot be protected against with SW countermeasures.

Finally, it is possible to make a traditional trade–off between the number of traces used for the attack during the Exploitation Phase, and the remaining brute–force effort required. When using, e.g., $N = 8$ traces in the Exploitation Phase instead of $N = 1$, the average rest entropy for a two–key TDES comes down to $75.2 - 2 \times 2.5 - 3.3 = 66.9$ bits for a 5–bit template. For $N = 16$ it is only $67.7 - 2 \times 2.5 - 2.4 = 60.3$ bits, for $N = 32$ it is $61.9 - 2 \times 2.5 - 1.6 = 55.3$ bits, for $N = 64$ it is $58.5 - 2 \times 2.5 - 0.7 = 52.8$ bits, for $N = 128$ it is $56.5 - 2 \times 2.5 - 0.3 = 51.2$ bits, and for $N = 256$ it is $55.5 - 2 \times 2.5 - 0.2 = 50.3$ bits,[22] which is perfectly possible to brute force with reasonable effort. When using 11–bit templates, these numbers should be reduced by an estimated further 2 bits, meaning that this attack can be done with $N \approx 100$ traces in the Exploitation Phase, and less than 50 bits of brute–force effort.

Such a strategy of using a few traces in the Exploitation Phase will not only improve the original single–trace template attack based on the 28 overlapping templates, but it will also improve the template attack based on the total Hamming distance of Sec. 3, and thus it is expected that the contribution of the latter of $\approx 2 \times 2.5 = 5$ bits for a two–key TDES will go up by a couple of bits as N increases, the maximum being ≈ 13 bits. And as with single–trace attacks, for some attack scenarios it makes sense to focus on the left tail of the distributions, the weak keys, and work out the effective rest entropy, which may be lower than the average one by a couple of bits.

In any case, as N increases, the remaining brute–force effort to break a two–key TDES becomes less and less, although for large N it is levelling off, the reason of which is the existence of key collisions in the key scheduling.

A couple of further improvements to the attack are possible, such as:

- Using better measurement equipment such as, e.g., 10 or 12 bit oscilloscopes, higher–resolution traces, and more Points Of Interest (POIs).
- Proper alignment of traces is crucial for single–trace or few–traces attacks. Since the number of traces used in the Exploitation Phase is very small, it is possible to improve their alignment manually, which is often superior to automated alignment. It is hard to predict, though, how significant such an improvement will be.
- In the same spirit, using Principal Component Analysis and Whitening Techniques is expected to improve the results further.
- Rather than using these more classical approaches, it may be more efficient to use supervised machine learning.
- Using larger template sizes (i.e., 11–bit, 13–bit, or even 15–bit). However, this should be only a minor effect of the order of 1 bit for TDES.
- The choice of δr can be improved, and possibly made dependent on N, $\bar{r}_C$ and $\bar{r}_D$. Again, this will yield only a few bits for TDES.
- In the same spirit, for single–trace attacks there is some weak (negative) correlation between the Average Ranking of the C and the D register, as shown in Fig. 13, and it is getting smaller with the size of the templates, which can be exploited to improve the key enumeration. This may yield just 1 bit for TDES for a single–trace attack, but with Table 3 it is clear that

[22] In this case N is so large already that the difference between $\bar{E}_{1D}(N)$ and $\bar{E}_{1D}(2N)$ is not ≈ 3 bits anymore, but rather only ≈ 1 bits.

this correlation is getting much stronger when using more than one trace in the Exploitation Phase.

- It should be possible to find better strategies for the key enumeration. For instance, there are indications that whilst sorting by Average Ranking is in general the better choice as key enumeration scheme, it appears to be outperformed for very leaky traces when sorting by Maximum Ranking. This particular improvement will only yield a few bits for TDES, but there may well be other, better search strategies that will only be found with further research. For instance, if the outliers in the rankings were to share certain properties, this may be factored into an improved search algorithm.
- In any case, it seems prudent to work with proper probabilities in the key enumeration scheme, rather than averages or maximum values as was done in this work.
- If search strategies are used that terminate at a certain predefined threshold in the key enumeration, regardless of whether the key has been found or not, then some prunings in the key enumeration list can be made prior to starting the search to reduce the brute–force effort by a couple of bits for TDES. Such an approach would be particularly beneficial when restricting the search to the left tail of the distribution, anyway.
- The 15–tuple approach suggests that the key enumeration strategy can be improved upon when better exploiting the dependencies between the leakages of the two C rings. After all, the total Hamming distance leakage yielded already an improvement of some 2.5 bits.
- Finally, it is possible to combine this key–schedule leakage with the key–loading leakage reported in [1] for the same target device to create better key enumeration lists. This is expected to improve the results very significantly and will be subject of future work. In Sec. 5 we show that in the case of a perfect Hamming leakage model this can yield a reduction of close to 7 bits for a two–key TDES.

Most importantly, the existence of weak keys means that any vulnerability analysis performed to assess the severity of this attack should be done using a weak key, of which there exist plenty — see footnote 15. Likewise, the attacker will know whether (s)he is targeting a weak key or not, simply by analysing the characteristics of the Hamming distances obtained in the side–channel analysis, and thus (s)he will know whether or not to proceed with the second and likely more expensive step of a brute–force attack.

Whilst for a single–trace attack it is impossible to find effective SW countermeasures as elaborated upon in [2], it is in principle at least possible to defend against this attack, if many traces were required in the Exploitation Phase, like 10 k - 100 k traces. However, we find that using a very few traces in Exploitation Phase is already enough to reduce the brute–force effort significantly, and in such a scenario SW countermeasures are not effective anymore — and in any case they will degrade the performance of the device massively by orders of magnitudes.

This device and other devices belonging to the same family of devices that are sharing by and large the same hardware DES coprocessor are currently deployed in TPMs, passports, ID cards, and in banking, to name but a few applications, with the total number of devices being in the billions. Given the results presented in this paper,[23] it seems prudent to assess the risk in continuing their usage for security applications. As part of a Responsible Disclosure Policy, the relevant Common Criteria certification body has been informed already in April 2016, who in turn has informed the manufacturer and the security evaluation laboratory.

References

1. Wagner, M., Hu, Y., Zhang, C., Zheng, Y.: Comparative Study of Various Approximations to the Covariance Matrix in Template Attacks. Cryptology ePrint Archive, Report 2016/1155, 2016
2. Wagner, M., Heyse. S.: Single–Trace Template Attack on the DES Round Keys of a Recent Smart Card, Cryptology ePrint Archive, Report 2017/057, 2017
3. https://en.wikipedia.org/wiki/Quicksort

[23] After all, we performed some 2*(297 k + 64 k + 32 k + 6 * 32 k + 24 k + 12 k + 6 k + 3 k) = 1260 k brute–force attacks on this device, thus clearly demonstrating that this attack is entirely feasible.