

# Improved Brute-Force Search Strategies for Single-Trace and Few-Traces Template Attacks on the DES Round Keys

Mathias Wagner, Stefan Heyse

`mathias.wagner@nxp.com`

**Abstract.** We present an improved search strategy for a template attack on the secret DES key of a widely-used smart card, which is based on a Common-Criteria certified chip. We use the logarithm of the probability function as returned by the template attack itself, averaged over all 28 template positions along the rings representing the C and D Registers of the DES key schedule, as the sorting criteria for the key candidates. For weak keys — which in this attack model have a minimal rest entropy of only two bits — we find that on average only 37.75 bits need to be recovered by brute force when using only a single trace in the Exploitation Phase. This effort goes down to just a few bits for a single DES key when using only a few traces in Exploitation Phase.

## 1 Introduction

Template attacks on the DES key loading [1] and the DES key schedule [2, 3] have been the topic of a number of previous publications, where we have established that such attacks can be successfully used to break the DES key of a smart card based on a recently certified chip.

In this paper we want to report on an improved search strategy for these attacks, using the same traces as in the previous works. The improvement comes about mostly by using a better sorting criteria for the ordering of the ranking of key candidates in the final brute-force step of the attack.

In a nutshell, this attack takes advantage of a weakness in the key schedule of the DES HW coprocessor of the smart card in question — certified by the German BSI — where the DES round keys of any two consecutive rounds in the key schedule leak their Hamming distances, with correlation function amplitudes being as large as 70%. Since the DES key schedule decomposes into two disjunct subkey schedules for the so-called C and the D Register, respectively, the complexity of the attack is drastically reduced — in essence it boils down to having to sort two lists — each comprising  $2^{27}$  subkey candidates — according to a yet to be defined sorting criteria, and then search for the correct key across these two ordered lists.

As in the previous papers, we will perform the template attack on the DES keys based on templates created on so-called rings for the C and D Registers. An alternative approach based on work presented in Sec. 5 of [3], taking more



will come back to the task of finding consistent key candidates across these 28 lists further below, but first let us specify what approximations we use when calculating the templates for this attack.

According to [1], various approximations are possible to evaluate the templates in the Profiling Phase. As in the previous papers [2, 3], we chose the most simple approximation by calculating a simple average  $\bar{C}$  over the covariances  $C_j$  of each template class  $j$  as

$$\bar{C} = \frac{1}{M} \sum_j C_j, \quad (4)$$

where  $M$  is the number of possible template values in each list, so  $M = 2^{11}$  in our case. This then leads to an approximate probability function  $P$ , the logarithm of which is called `LnP_AvC` in Table 1 of paper [1]. This is the most simple average one can use for the covariance, but other definitions proposed in [1] work equally well, and thus we believe our results are not sensitive to the precise method how to average over  $C_j$ .

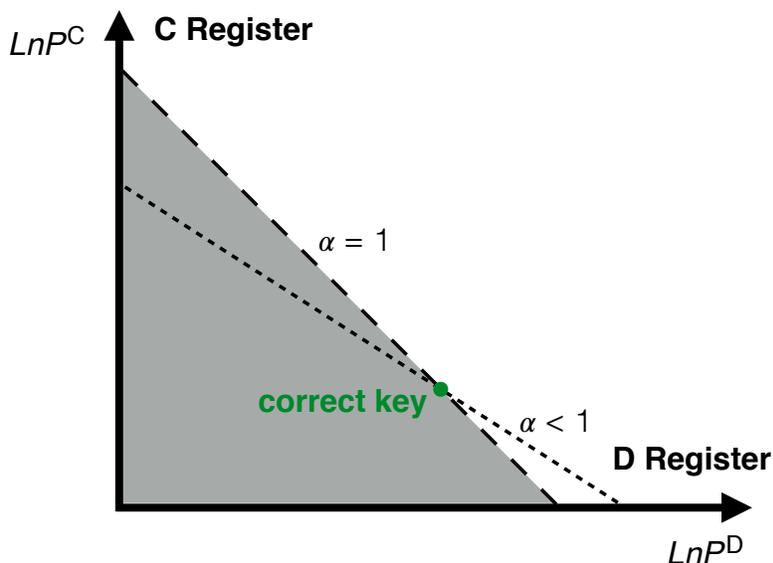
The next task then is to combine the 14 individual ranking lists for the C respectively D Register to a single ranking list for that register. Without loss of generality, let us look at the 14 ranking lists for the C Register. As elaborated in more detail in [2, 3], the first step is to weed out these 14 ranking lists to use only subkeys and corresponding ranking entries on each list that are compatible with the subkeys on neighbouring ranking lists. This is simply a requirement stemming from the fact that the template positions along the ring were chosen to overlap with each other, and hence the templates are not independent. Not to be forgotten is the overlap of the “first” and the “last” template along the ring, which reduces the possible choices by another factor of 2. Eventually, a total of  $2^{27}$  possible entries remain that are consistent across all 14 lists. This makes a lot of sense, since the C and the D Register control 28 subkey bits each, and one bit is consumed by the  $\oplus$  operation already, leaving 27 bits. With this we arrive at two (not yet) ordered lists of  $2^{27}$  entries each, one for the C Register, the other for the D Register.

What is new in this paper compared to our previous work is how we order these two ranking lists, each containing  $2^{27}$  subkey candidates, to reduce the effort of brute-force searching further. As a first step we calculate for each entry in these two lists the sum over the logarithm of the approximate probability function  $P$  (as returned by the pattern-template matching step) as

$$\text{Ln}P^{C,D} = \sum_{i=0}^{13} \ln(P_i^{C,D}), \quad (5)$$

where the superscripts C and D refer to the C and D Register, respectively, and  $i$  simply runs over all 14 template positions of the respective ring.<sup>2</sup> In essence, the underlying assumption with this metric is that the probability of a given entry

<sup>2</sup> Using QuickSort [4] the task of sorting a list of  $2^{27}$  subkey candidates according to the metric given by Eq. (5) is done in less than a minute on a standard PC.



**Fig. 1.** Search strategy dealing with the two subkey ranking lists of the C and D Registers of the DES key schedule. For each list the value of  $LnP^{C,D}$  is calculated according to Eq. (5), and then the sum  $LnP$  is taken according to Eq. (6), initially with  $\alpha = 1$ . All key candidates in the grey area are being successively tested as the threshold  $LnP$  increases, until eventually the correct key is found. (This threshold is called “Frontier” in [5, 6].) The grey triangle then represents the number of key candidates that had to be tried in this brute-force search. Suitably chosen values of  $\alpha < 1$  can reduce the brute-force search effort on average, reflecting the fact that on average the D Register leaks more than the C Register.

on the list to be the correct subkey is given by the product of the probabilities of all its contributing 14 templates. Clearly, this assumption is in contradiction to the very fact that these 14 templates overlap as shown, e.g., in Eqs. (2) and (3), and hence *cannot* be independent. However, the results we obtain with this approach are encouraging. Perhaps there is still room for further improvement when accounting for these overlap-induced dependencies as well.

To arrive at a single list of key candidates to be searched through, we can combine the C and D Register candidates and order the new list according to

$$LnP(\alpha) = LnP^C + \alpha LnP^D, \quad (6)$$

where the weighing parameter  $\alpha$  can be used to take into account that the D Register leaks more than the C Register, c.f. Fig. 16 in [2], in which case  $\alpha < 1$  holds. Searching is now done by successively increasing the value of the threshold  $LnP(\alpha)$  until the correct key is eventually found. The key space to be searched

through this way is indicated, for  $\alpha = 1$ , by the grey area in Fig. 1.<sup>3</sup> This approach is very similar to the optimal search strategy proposed earlier in [5] (see, e.g., Fig. 1 therein) or, more recently, in [6].

As always when performing the risk analysis of a vulnerability found, be it within the Common Criteria or any other security assessment scheme, one needs to consider the worst-case scenario. For the attack in question the worst case arises when the secret DES key is a so-called weak key, which in the perfect leakage model has a very low rest entropy in the final brute-force step of the attack. Weak keys as pertinent to the current attack have been discussed in Sec. 5 of [3], and in contrast to the 4 weak keys known from a cryptanalysis of the DES algorithm, plenty of weak keys exist for the present attack scenario. In the worst case, it turns out that weak keys have a rest entropy as low as 2 bits only for this attack, and some 0.135% of all DES keys fall in this class. In the trace set used in the Exploitation Phase of [2, 3] there are 378 traces and associated keys fulfilling this condition, and these will be the traces we will focus on in the subsequent analysis to perform 378 single-trace attacks.

For  $\alpha = 1$  we find — averaged over those 378 attacks — a remaining rest entropy of  $E(\alpha = 1) = 39.09$  bits. This can be optimised slightly when accounting for the fact that the D Register leaks more strongly than the C Register, by optimising the parameter  $\alpha$ , yielding  $E(\alpha = 0.68) = 38.77$  bits.<sup>4</sup> In contrast, the approach proposed in [2] based on taking simple averages of the rankings of the C and D Register lists yields a rest entropy of  $E_{\text{av}} = 41.73$  bits.<sup>5</sup> The new search strategy is thus some 3 bits better than the one originally proposed in [2].

This result can be further improved upon by taking into account the leakage of the Total Hamming Distance (THD)  $\|\text{dist}_{15}\|_1$  as elaborated in more detail in Secs. 3 and 5 of [3]. This leakage is different from that of the C rings discussed so far, as it also accounts for how often two key bits are actually connected xor-wise anywhere in the 16 rounds of the DES key schedule. In contrast, the two C rings only record whether two bits are connected via an  $\oplus$  at all, but not how often. According to Table 5 of [2] some bit pairs are connected only at 2 positions in the key schedule, whilst others are connected in as many as 12 different positions. The construction of the corresponding templates has been discussed in [3] and will not be repeated here. It shall suffice to say that all traces need to be preprocessed with one more step by averaging over the 16 DES key-schedule rounds.

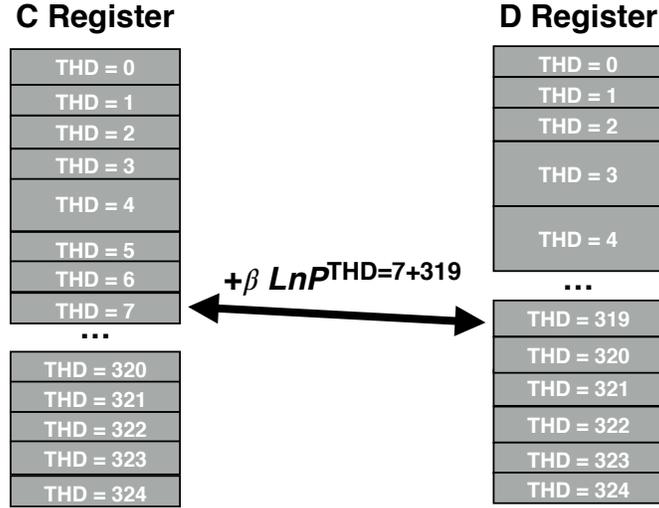
We deal with this additional leakage in the same way as we did when combining the leakages of the C and the D Registers — we add it to Eq. (6) using yet another weighting factor  $\beta$  as

$$\text{Ln}P(\alpha, \beta) = \text{Ln}P^{\text{C}} + \alpha \text{Ln}P^{\text{D}} + \beta \text{Ln}P^{\text{THD}} . \quad (7)$$

<sup>3</sup> It turns out that one can further improve the results slightly when using a non-linear function instead of the linear one used in Eq. (6). However, this becomes very cumbersome and the benefit is rather small. Thus, here we only optimise  $\alpha$ .

<sup>4</sup> Any value  $0 < \alpha < 1$  means that the D Register leaks more than the C Register.

<sup>5</sup> The difference to the numerical value reported in [2, 3] is due to the fact that in the current paper we restrict the analysis to weak keys only.



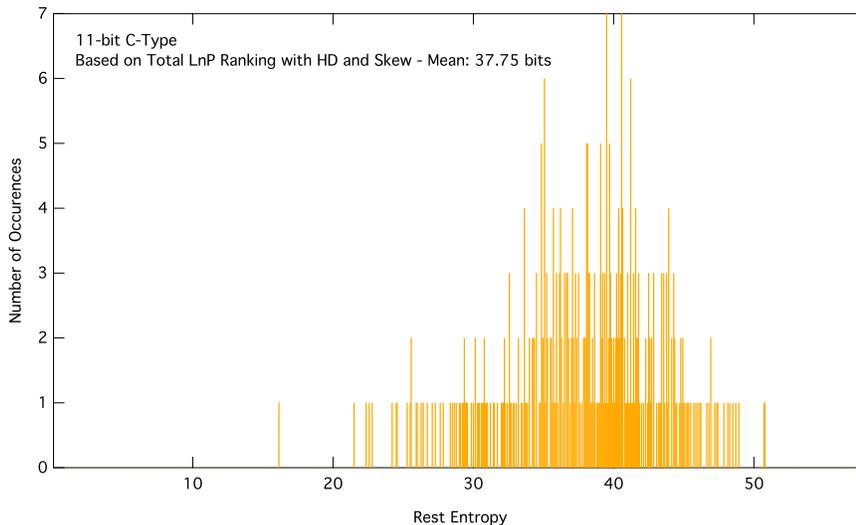
**Fig. 2.** Search strategy according to Eq. (7): For each register we first order the list according to the Total Hamming Distance (THD) of its entries. This gives 235 subsets for each register. Within each such subset we then order as before according to Eq. (6). Each combination of one subset from the left with one subset from the right has then a constant additional offset  $\beta \text{Ln}P^{\text{THD}}$  according to Eq. (7). In essence, we then need to keep track of and simultaneously advance  $325 \times 325$  thresholds or “Frontiers” connecting all subsets of the left with all subsets on the right, until the correct key is found.

However, as it stands, at first sight Eq. (7) seems to destroy our approach to have two disjoint ordered sets of only  $2^{27}$  subkeys, one belonging to the C Register, and the other belonging to the D Register. The term  $\beta \text{Ln}P^{\text{THD}}$  seems to imply that we now need to sort all  $2^{54}$  DES keys in one go, which is much more work than sorting two sets of only  $2^{27}$  subkeys each. Fortunately, this is not the case.

Firstly, we note that the Total Hamming Distance (THD)  $\|\text{dist}_{15}(k)\|_1$  of a DES key  $k$  decomposes into its contributions from the C and the D Register as  $\|\text{dist}_{15}(k)\|_1 = \|\text{dist}_{15}(k_C)\|_1 + \|\text{dist}_{15}(k_D)\|_1$ , where  $k_C$  and  $k_D$  are the respective subkeys belonging to these two registers. By definition, the term  $\beta \text{Ln}P^{\text{THD}}$  only depends on the sum  $\|\text{dist}_{15}(k)\|_1$ .

Secondly, as illustrated in Fig. 2 we can reorder the  $2^{27}$  subkeys of, e.g., Register C such that all subkeys with the same value of  $\|\text{dist}_{15}(k_C)\|_1$  are gathered within one contiguous subset, labelled with  $\text{THD} = 0, 1, 2, \dots, 324$ . There are a total of 325 such subsets, as the largest THD possible for a 27-bit subkey “scattered” and repeated across 16 DES rounds turns out to be 324, whilst the smallest is 0.<sup>6</sup> Within each such subset we order the subkeys in the same way as

<sup>6</sup> A few extreme values are actually not possible, like a Total Hamming Distance of 1, and so the number of subsets is actually slightly less than 325, but this does not matter for this analysis.



**Fig. 3.** Distribution for the rest entropy  $E_T$  based on 378 single-trace attacks and using Eq. (7) with  $\alpha = 0.675$ ,  $\beta = 0.252$ .

before according to increasing values of  $LnP^C$ . The same is then done with the subkeys belonging to Register D. The effort of sorting these two lists this way is only marginally larger than it was before. In essence, we are still sorting two lists of  $2^{27}$  subkeys each, and the problem remains easy to tackle.

With these newly ordered two lists we can still use the same search strategy as in Fig. 1, but now we apply it to all  $325 \times 325$  possible combinations of these C and D Register subsets. Each such combination has a constant value of  $\beta LnP^{THD}$ , which simply offsets this combination as a whole with regards to all other combinations. In the end, we can search over these  $325 \times 325$  combinations as before using a single threshold value for  $LnP$ .

For  $\alpha = 1$  we find a rest entropy of  $E_T(\alpha = 1, \beta = 0.253) = 38.06$  bits when averaged over the ensemble of 378 single-trace attacks targeting a weak key. If we optimise both parameters simultaneously, we find  $E_T(\alpha = 0.675, \beta = 0.252) = 37.75$  bits. The fact that the values of  $\alpha$  and  $\beta$  do not change much compared to the previous results is an indication that these results are fairly robust. The distribution over these 378 single-trace attacks is shown in Fig. 3. This result is about 4 bits better than the first results shown in [2]. Thus, accounting for the Total Hamming Distance leakage in the analysis yields one further bit reduction of the brute-force effort. This is less than what had been estimated in [3], though.<sup>7</sup>

Clearly, we can still improve these results in a number of ways. Firstly, we can account for yet more leakage mechanisms by adding more terms in Eq. (7). An

<sup>7</sup> It should be noted that optimizing the parameter  $\alpha$  alone yielded on average about a third of a bit only. Given that this is a fairly expensive operation, it may not be worth doing.

example would be the leakage during the key-loading stage of the same smart card as presented in [1]. Comparing Figs. 42 and 43 of [3], this can potentially yield an average additional reduction of up to 3.34 bits in the brute-force effort. With this the average rest entropy for a single-trace attack could come down to roughly 34.4 bits.

Secondly, Eq. (5) assumes that the 14 templates on a C ring are completely independent of each other, which is not the case, as they overlap. It would be interesting to find a way to take this property into account.

Finally, if we allow more traces to be used in the Exploitation Phase, results can be dramatically improved as already shown in Sec. 6.2 of [3] for a randomly chosen but fixed key with an average rest entropy. However, for the smart card in question we simply did not measure multiple traces for the same weak key at the time, and hence we cannot perform a similar analysis here. Yet, with the results of [3] it is more than reasonable to expect that it does not take more than a few traces to push the remaining average rest entropy below 20 bits for a weak key. If we keep increasing the number of traces used during the Exploitation Phase further, eventually the ultimate rest entropy of these weak keys should be reached, namely only 2 bits.

### 3 Conclusions

In conclusion, we have presented an improved search strategy for the remaining brute-force search in the template attack on the DES key schedule of a recent smart card based on a widely-used chip certified by the BSI. It is some 4 bits better than previous results, resulting in an average rest entropy of 37.75 bits for weak keys, using a single trace in the Exploitation Phase. This result can be further improved upon by combining it with other DES key leakage found in the same smart card (up to 3.34 bits less rest entropy), and/or using more than one trace during the Exploitation Phase. In the latter case the rest entropy is eventually expected to go down to values as low as 2 bits.

### 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. Wagner, M., Heyse, S., Guillemet, C: Brute-Force Search Strategies for Single-Trace and Few-Traces Template Attacks on the DES Round Keys of a Recent Smart Card, Cryptology ePrint Archive, Report 2017/614, 2017 (rev. in Dec 2017)
4. <https://en.wikipedia.org/wiki/Quicksort>
5. Veyrat-Charvillon, N., Gérard, B., Renauld, M., Standaert, F-X.: An Optimal Key Enumeration Algorithm and its Application to Side-Channel Attacks, Cryptology ePrint Archive, Report 2011/610, 2011
6. Li, Y., Wang, S., Wang, Z., Wang, J.: A Strict Key Enumeration Algorithm for Dependent Score Lists of Side-Channel Attacks, CARDIS, 2017