Comparative Study of Various Approximations to the Covariance Matrix in Template Attacks

Mathias Wagner, Yongbo Hu, Chen Zhang, Yeyang Zheng

NXP Semiconductors
mathias.wagner@nxp.com

Abstract. Template attacks have been shown to be among the strongest attacks when it comes to side–channel attacks. An essential ingredient there is to calculate the inverse of a covariance matrix. In this paper we make a comparative study of the effectiveness of some 24 different variants of template attacks based on different approximations of this covariance matrix. As an example, we have chosen a recent smart card where the plain text, cipher text, as well as the key leak strongly. Some of the more commonly chosen approximations to the covariance matrix turn out to be not very effective, whilst others are.

1 Introduction

Side–channel attacks have by now a very long history [1–3], and also the particular variant of template attacks [5–9] has been studied for quite some years. A critical piece in all these analysis is the calculation of a covariance matrix, and possibly approximations to it. The majority of the work published so far either uses the original definition, a simple average over all covariances, or some linearization thereof.

Template attacks take place in two phases — a first Profiling Phase, where templates are generated of the target function when all parameters and data are known. These templates are stored in a data base and are used in the second phase — the Exploitation Phase — to be matched against measurements of the target system when not all data are known anymore. The target system may be a physically different one than the one the templates were created with. The attack is successful, when the patterns found in Exploitation Phase can be matched with the correct templates generated during Profiling Phase. It often does not matter how many measurements (traces) are used in the Profiling Phase, since there it is assumed that the attacker has full control over the device and can take as many measurements as are required. It is even conceivable that more than one device is used in the Profiling Phase and this may in fact improve the robustness of the attack when applied to many target devices, due to the averaging effect. On the other hand, usually, the number of traces available for attacking the target device in the Exploitation Phase may well be limited due to countermeasures built into the target device, or the eco system it is operating in.

Table 1. Non–linear, non–differential template attack variants with generic names. Variables t_j and p_i refer to the templates / profiles created in the profiling respectively the Exploitation Phase, and m is the number of Points of Interest (POI). Averages over the entire respective data sets are denoted as $\bar{t}$ and $\bar{p}$, respectively. The various averages of the covariance matrix, like $\bar{C}$, $\tilde{C}$, and $\hat{C}$ are defined in Eqs. (4–6).

`individual` C ($\equiv$ `LnP`)	$-\ln(\det(C_j)) - \left\langle p_i - t_j \left\vert C_j^{-1} \right\vert p_i - t_j \right\rangle$
`normalised` C ($\equiv$ `LnP_NormC`)	$-\sqrt[m]{\det(C_j)} \left\langle p_i - t_j \left\vert C_j^{-1} \right\vert p_i - t_j \right\rangle$
`NoLnDet` C ($\equiv$ `LnP_NoDetC`)	$-\left\langle p_i - t_j \left\vert C_j^{-1} \right\vert p_i - t_j \right\rangle$
`average` C ($\equiv$ `LnP_AvC`)	$-\left\langle p_i - t_j \left\vert \bar{C}^{-1} \right\vert p_i - t_j \right\rangle$
`weighted average` C ($\equiv$ `LnP_wAvC`)	$-\left\langle p_i - t_j \left\vert \tilde{C}^{-1} \right\vert p_i - t_j \right\rangle$
`global average` C ($\equiv$ `LnP_gAvC`)	$-\left\langle p_i - t_j \left\vert \check{C}^{-1} \right\vert p_i - t_j \right\rangle$
`normalised average` C ($\equiv$ `LnP_nAvC`)	$-\left\langle p_i - t_j \left\vert \hat{C}^{-1} \right\vert p_i - t_j \right\rangle$
`least square` $C \equiv I_m$ ($\equiv$ `LnP_LSqr`)	$-\left\langle p_i - t_j \,\vert\, p_i - t_j \right\rangle$

In this paper we will address an improvement over what is currently state of the art for template attacks with regards to the evaluation and use of the covariance matrix.

In Sec. 2 we will discuss various approximations to the probability density function of the multivariate normal distribution typically used in template attacks. It will be seen that it is worthwhile evaluating all these approximations, since they can yield widely different results, depending on the situation, but every time consistently so. Since the attacker can study the strengths of these approximations in the Profiling Phase, we have to assume that the attacker will pick the strongest one for the actual attack.

Following a recent finding [10], in Sec. 3 these approximations are applied to the analysis of the DES hardware coprocessor of a recent smart card, where leakage is found for plain text, cipher text, as well as key in the form of $P[i] \oplus P[i+4]$, where $P[i]$ refers to the i–th byte of the plain text, and $C[i] \oplus C[i+4]$, as well as $K[i] \oplus K[i+4]$, $i = 0...3$.

2 Approximations to the Probability Function

The starting point is the standard definition of the template–matching probability P_{ij}, which except for constant terms that neither depend on the template j nor the pattern i is given by the "exact" formula [5] — using Dirac's Bra, $\langle .|$,

Table 2. Differential non–linear template attack variants.

`diff. individual` C (≡ `LnP_d`)	$-\ln(\det(C_j)) - \left\langle p_i - \bar{p} + \bar{t} - t_j \left\vert C_j^{-1} \right\vert p_i - \bar{p} + \bar{t} - t_j \right\rangle$
`diff. normalised` C (≡ `LnP_dNormC`)	$-\sqrt[m]{\det(C_j)} \left\langle p_i - \bar{p} + \bar{t} - t_j \left\vert C_j^{-1} \right\vert p_i - \bar{p} + \bar{t} - t_j \right\rangle$
`diff. NoLnDet` C (≡ `LnP_dNoDetC`)	$-\left\langle p_i - \bar{p} + \bar{t} - t_j \left\vert C_j^{-1} \right\vert p_i - \bar{p} + \bar{t} - t_j \right\rangle$
`diff. average` C (≡ `LnP_dAvC`)	$-\left\langle p_i - \bar{p} + \bar{t} - t_j \left\vert \bar{C}^{-1} \right\vert p_i - \bar{p} + \bar{t} - t_j \right\rangle$
`diff. weighted average` C (≡ `LnP_dwAvC`)	$-\left\langle p_i - \bar{p} + \bar{t} - t_j \left\vert \tilde{C}^{-1} \right\vert p_i - \bar{p} + \bar{t} - t_j \right\rangle$
`diff. global average` C (≡ `LnP_dgAvC`)	$-\left\langle p_i - \bar{p} + \bar{t} - t_j \left\vert \check{C}^{-1} \right\vert p_i - \bar{p} + \bar{t} - t_j \right\rangle$
`diff. normalised average` C (≡ `LnP_dnAvC`)	$-\left\langle p_i - \bar{p} + \bar{t} - t_j \left\vert \hat{C}^{-1} \right\vert p_i - \bar{p} + \bar{t} - t_j \right\rangle$
`diff. least square` $C \equiv I_m$ (≡ `LnP_dLSqr`)	$-\left\langle p_i - \bar{p} + \bar{t} - t_j \,\vert\, p_i - \bar{p} + \bar{t} - t_j \right\rangle$

and Ket, $|.\rangle$, vector notation [11] —

$$\ln(P_{ij}) \approx -\ln(\det(C_j)) - \left\langle p_i - t_j \left\vert C_j^{-1} \right\vert p_i - t_j \right\rangle \tag{1}$$

for its logarithm. Here p_i is the ith pattern of the Exploitation Phase, obtained by taking the mean over all traces found in class i during exploitation, whilst $\{t_j, C_j^{-1}\}$ denote the jth template (mean and inverse covariance) of the Profiling Phase. The ranking of each pattern i is derived by calculating $\ln(P_{ij})$ for all templates j and then ordering this list for the largest value.

Using Eq. (1) as a starting point, various approximations to it can be derived. First, one often finds that the $\ln(\det(C_j))$ term is actually more a nuisance than a help — possibly because in "reality" the determinant of the covariance might not depend on the template j at all, and all we see here are numerical and statistical errors due to the discreetness and finite size of C. Then the maximal value of $\ln(P_{ij})$ is trivially obtained in the limit $p_i \to t_i$, so by way of pattern matching, and not due to some fancy structure of C_j^{-1}. In such a case it may be better to simply drop this term,

$$\ln(P_{ij}) \approx -\left\langle p_i - t_j \left\vert C_j^{-1} \right\vert p_i - t_j \right\rangle , \tag{2}$$

or, alternatively, one can renormalise all the covariances C_j such that their determinants are all the same. This latter approach leads to

$$\ln(P_{ij}) \approx -\sqrt[m]{\det(C_j)} \left\langle p_i - t_j \left\vert C_j^{-1} \right\vert p_i - t_j \right\rangle , \tag{3}$$

where m is the number of Points of Interest (POI).

Another way of getting rid of the variable determinant term is to average over all covariances in a suitable form. We will consider four different ways of

Table 3. Linear (differential) template attack variants.

`lin. average` C ($\equiv$ `LnP_lAvC`)	$\left\langle p_i - \bar{t} \left\vert \bar{C}^{-1} \right\vert t_j - \bar{t} \right\rangle$
`lin. weighted average` C ($\equiv$ `LnP_lwAvC`)	$\left\langle p_i - \bar{t} \left\vert \tilde{C}^{-1} \right\vert t_j - \bar{t} \right\rangle$
`lin. global average` C ($\equiv$ `LnP_lgAvC`)	$\left\langle p_i - \bar{t} \left\vert \check{C}^{-1} \right\vert t_j - \bar{t} \right\rangle$
`lin. least square` $C \equiv I_m$ ($\equiv$ `LnP_lLSqr`)	$\left\langle p_i - \bar{t} \,\middle\vert\, t_j - \bar{t} \right\rangle$
`diff. lin. average` C ($\equiv$ `LnP_dlAvC`)	$\left\langle p_i - \bar{p} \left\vert \bar{C}^{-1} \right\vert t_j - \bar{t} \right\rangle$
`diff. lin. weighted average` C ($\equiv$ `LnP_dlwAvC`)	$\left\langle p_i - \bar{p} \left\vert \tilde{C}^{-1} \right\vert t_j - \bar{t} \right\rangle$
`diff. lin. global average` C ($\equiv$ `LnP_dlgAvC`)	$\left\langle p_i - \bar{p} \left\vert \check{C}^{-1} \right\vert t_j - \bar{t} \right\rangle$
`diff. lin. least square` $C \equiv I_m$ ($\equiv$ `LnP_dlLSqr`)	$\left\langle p_i - \bar{p} \,\middle\vert\, t_j - \bar{t} \right\rangle$

averaging, two of which should converge in the limit of many traces taken. The standard average $\bar{C}$ is simply given by

$$\bar{C} = \frac{1}{M} \sum_j C_j \ , \tag{4}$$

where M is the number of templates, whilst $\tilde{C}$ is a weighted average,

$$\tilde{C} = \frac{1}{M'} \sum_j n_j C_j \ , \tag{5}$$

where as weights the number of traces n_j pertinent to template j have been used, and $M' = \sum_j n_j$. Clearly, these two definitions are identical when n_j does not depend on j, or at least very nearly so, which is often the case for large data sets created with random data. However, when the target is not the value itself, but rather the Hamming weight, then this is not normally the case, as the Hamming weight distribution is highly uneven. Small and very large Hamming weights are much less frequently seen than average Hamming weights.

Yet another way to arrive at a normalised average covariance $\hat{C}$ is to expand on the idea underlying Eq. (3), and renormalise all covariances C_j prior to averaging such that all have the same determinant,

$$\hat{C} = \frac{1}{M} \sum_j \sqrt[-m]{\det(C_j)}\, C_j \ . \tag{6}$$

Still another idea is to view all traces of all templates belonging to a single super template and then work out the covariance $\check{C}$ of this data set.

But the most aggressive level of approximation is to assume $C \equiv I_m$, leading to

$$\ln(P_{ij}) \approx -\Big\langle p_i - t_j \,|\, p_i - t_j \Big\rangle \;, \tag{7}$$

which for obvious reasons is also referred to as the least–square approximation. If this variant is any good, then clearly it is the trivial mechanism of pattern matching, $p_j \to t_j$ for large n_j, which is the dominant effect, and not a fancy structure of C_j^{-1}.

Finally, when we assume C to be the same across all templates, anyway, then with a few more approximations it is possible to linearise all these expressions, where for instance Eq. (1) then becomes

$$\ln(P_{ij}) \approx \Big\langle p_i - \bar{t} \,\Big|C^{-1}\Big|\, t_j - \bar{t} \Big\rangle \;, \tag{8}$$

where $\bar{t} = (1/M') \sum_j n_j t_j$, and C being defined by either Eq. (4), or Eq. (5), or again by simply setting $C \equiv I_m$. Please heed the opposite sign in this equation.

On a finer point, it should be noted that the calculation of $\bar{t}$ is done during the Profiling Phase, which in practice can introduce an unwanted offset into the difference $p_i - \bar{t}$ needed in the term multiplied to C^{-1} from the left in Eq. (8), with a negative impact on the accuracy of the resulting matching probability. It may thus be better to replace $\bar{t}$ by the weighted average over all profiles, $\bar{p} = (1/N') \sum_i n'_i p_i$, obtained during the Exploitation Phase. This approximation tends to yield very poor results when the patterns are based on very few traces only. In such a situation the original linearisation is generally better, since the templates have been based on more traces, usually.

This trick of replacing $\bar{t}$ by $\bar{p}$ can also be applied to all the non–linear variants of $\ln(P)$, by simply inserting the difference $\bar{p} - \bar{t}$ on both sides of the bilinear form. These variants of $\ln(P)$ may perform better when the traces in the Exploitation Phase have a systematic offset compared to those of the Profiling Phase.

All these approximations give a grand total of 24 different definitions for $\ln(P_{ij})$ — one original, one without the $\ln(\det(C_j))$ term, one with renormalised C_j, four with different definitions of an average C, one least–square variant, then eight variants of these first eight, where $\bar{p} - \bar{t}$ got inserted on both sides, and another eight different linearisations of some of the average variants of above. Tables 1, 2 and 3 provide an overview, including the naming convention used in all subsequent plots. The strength of these template attack variants may be vastly different and thus one should always inspect all of them, since the attacker can do so during the Profiling Phase and determine which one works best.

Finally, it is also worthwhile to study the case where all the values of the pattern–template matchings relate to each other by way of having been derived from a single common (secret) source via xoring. So, they all relate to each other in a particular way as $I \oplus K$, where K is the common secret source — like a round key for DES and AES, or the secret cipher text of a previous DES call in a chained DES when attacking the input of the subsequent DES computation (as in the combined side–channel / brute–force attack on ISO9797-1 MAC [12]) — and I is the random, but known input. In such a case, it would be clearly wrong if

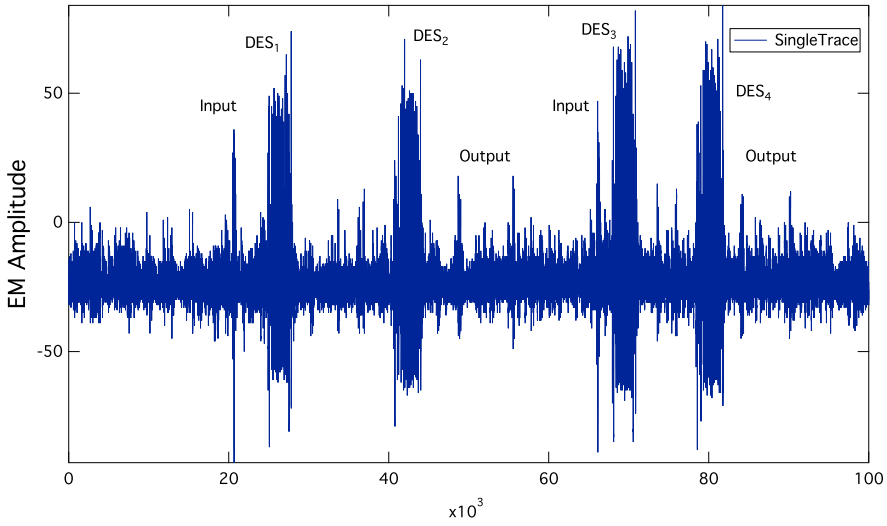

Fig. 1. A typical single EM trace of the TOE showing 4 calls to the DES hardware engine. It was obtained by placing a commercial Langer EM probe on top of what had previously been identified as a DES coprocessor hard macro in the chip layout. Sampling rate: 5 GS/s.

any two patterns i and i' were to point to the same template j as the most likely matching candidate. Consequently, a way needs to be found to eliminate pattern–template matchings that are not compatible with each other, and weeding out such inconsistencies should improve the ranking lists considerably. When all patterns relate to each other via a bijective xor mapping such that $\ln P_{i,i\oplus j}$ is the correct matching for pattern i, then the joint probability distribution is given by the sum over all [13],

$$\ln \hat{P}_j = \sum_i n_i \ln P_{i,i\oplus j} \quad , \tag{9}$$

where the sum runs over all patterns i. When the template attack is successful, j will point to the unknown common secret K. Another way of looking at Eq. (9) is to regard it as the product over all probabilities compatible with a particular value j for the unknown common secret.

3 Results for a Contemporary Smart Card

Target of Evaluation (TOE) was a smart card from 2012 containing a JAVA OS where we could freely call the DES function — although likely via a wrapper in an underlying crypto library. A typical single EM trace is given in Fig. 1, showing essentially 4 calls to the DES hardware engine. First we remove the strong timing jitter between these blocks using a simple rigid–pattern filter that locks into these four DES blocks. In a next step we refine the alignment by applying an

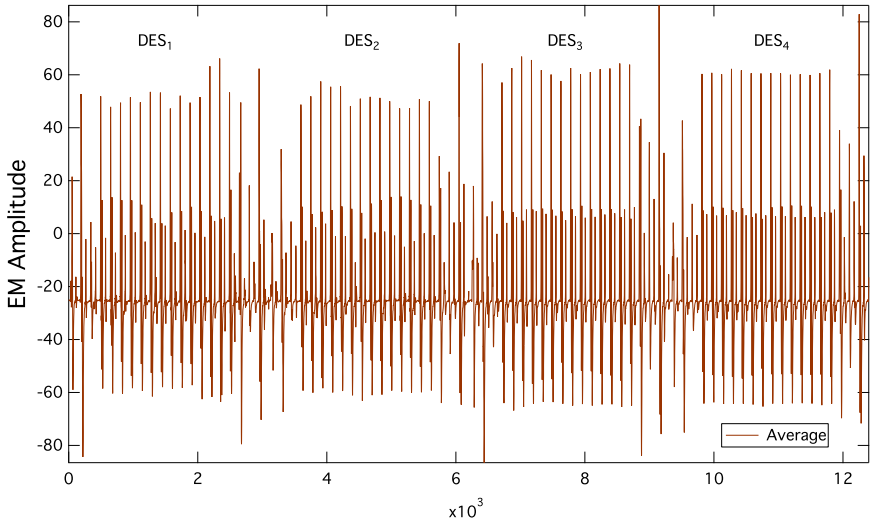

Fig. 2. Average trace with all four DES blocks aligned.

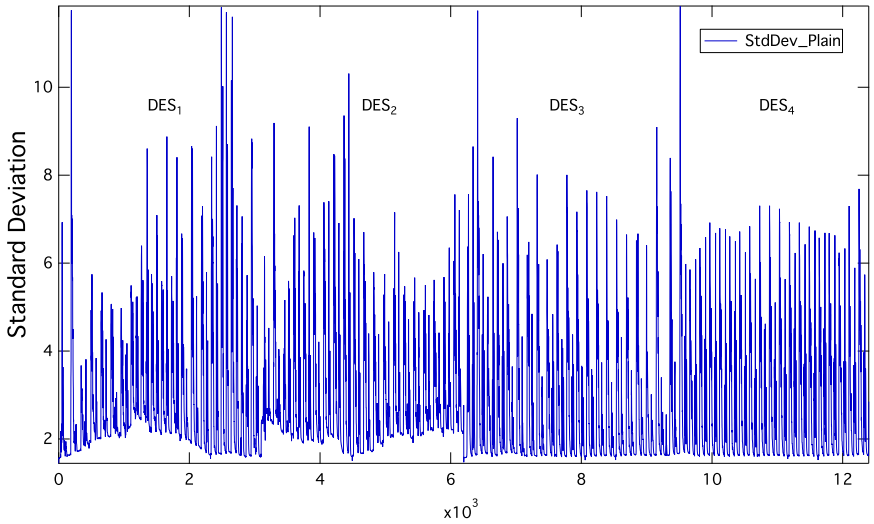

Fig. 3. Standard deviation with all four DES blocks aligned. The first DES block is the most difficult to align and far from perfect yet.

elastic alignment filter based on segment–wise parametrizing the internal clock / time as $T = a + bt + ct^2 + dt^3 + et^4 + ft^5$, where t corresponds to the external clock / time, and then finding the best set of coefficients (a, b, c, d, e, f) for each DES segment of each trace.[1] Starting with a little over 7M raw traces with

[1] The raw traces show two different types of timing jitter: Firstly, there are random delays inserted between the four DES blocks. These are removed by taking one trace

random plain text and random key, this procedure yielded just over 5M aligned traces, with their average trace given in Fig. 2, and its standard deviation shown in Fig. 3. Likely, the multiple calls to the DES hardware engine are due to countermeasures against fault attacks. We do not need to know whether these DES calls are "forward" or "backward" calculations — it does not matter, anyway, for launching a template attack, as long as they are static and do not change from one call to another. However, subsequent analysis reveals that these are forward – backward – forward – backward DES calls. Interestingly, they do not have precisely the same run–time behaviour, and also their standard deviations are slightly different as per Fig. 3. The last two DES blocks are much easier to align. Other than this alignment, no further preprocessing has been performed for calculating the templates later on.

The same procedure was then also applied to a further set of 2M raw traces, yielding 1.5M aligned traces in the end, where again the plain text was chosen randomly, but this time the key was kept fixed. This second set of traces will be used for the Exploitation Phase.

Fig. 4 shows strong leakage of the plain text, cipher text, as well as the key in the form of $P[i] \oplus P[i+4]$, $C[i] \oplus C[i+4]$, and $K[i] \oplus K[i+4]$, where $i = 0...3$, and i refers to the i–th byte. This leakage is most likely due to the internal architecture of this DES hardware engine, where data are moved into the frontend of the coprocessor in portions of 32 bits, and then put into place inside the coprocessor in the next clock cycle. Without any blinding countermeasures this will lead to a Hamming distance leakage of the left half of the plain text with its right half, and similarly for cipher text and key. The timing of leakage seen in the trace indicates that except for the very first DES call, where the plain text does not seem to leak at all, plain text and key are handled in parallel, whilst the cipher text is naturally being handled at a later point in time. Because of this larger algorithmic noise, attacking the plain text and the key should be somewhat harder than attacking the cipher text. Naturally, at most half of the plain text, cipher text, and key bits are possible to recover with this attack.

In Figs. (5–6) we show the results of a template attack in Exploitation Phase using templates generated with 5M traces as input and 576 points of interest (POI), chosen based on a χ^2 analysis[2] — and this for all 24 approximations to

as a reference trace and then for each trace perform a least–square search for each of the 4 DES blocks separately to get a rough alignment of those 4 DES blocks. Secondly, the internal clock does not appear to be very stable. In the most simple approximation we assume the internal clock frequency to vary smoothly over time and model this with a polynomial fit. Since polynomial fits only work well over rather short time intervals, it then becomes necessary to split each DES block into a couple of time segments and apply the polynomial fit to each of them separately, using continuity equations as needed. Again, for each trace and each time segment, the best set of coefficients is determined using a least–square algorithm.

[2] With canonical meanings of all variables, the definition of the χ^2 function is given as $\chi^2(t) = \frac{1}{n} \frac{1}{m_{\mathrm{HW}}-1} \sum_{i_{\mathrm{HW}}=0}^{m_{\mathrm{HW}}} n_{i_{\mathrm{HW}}} \frac{\left(\mu_{i_{\mathrm{HW}}}(t) - \mu(t)\right)^2}{\sigma^2(t)}$. With slight adaptations, this formula can be used for each byte separately, as well as for all 4 bytes together.

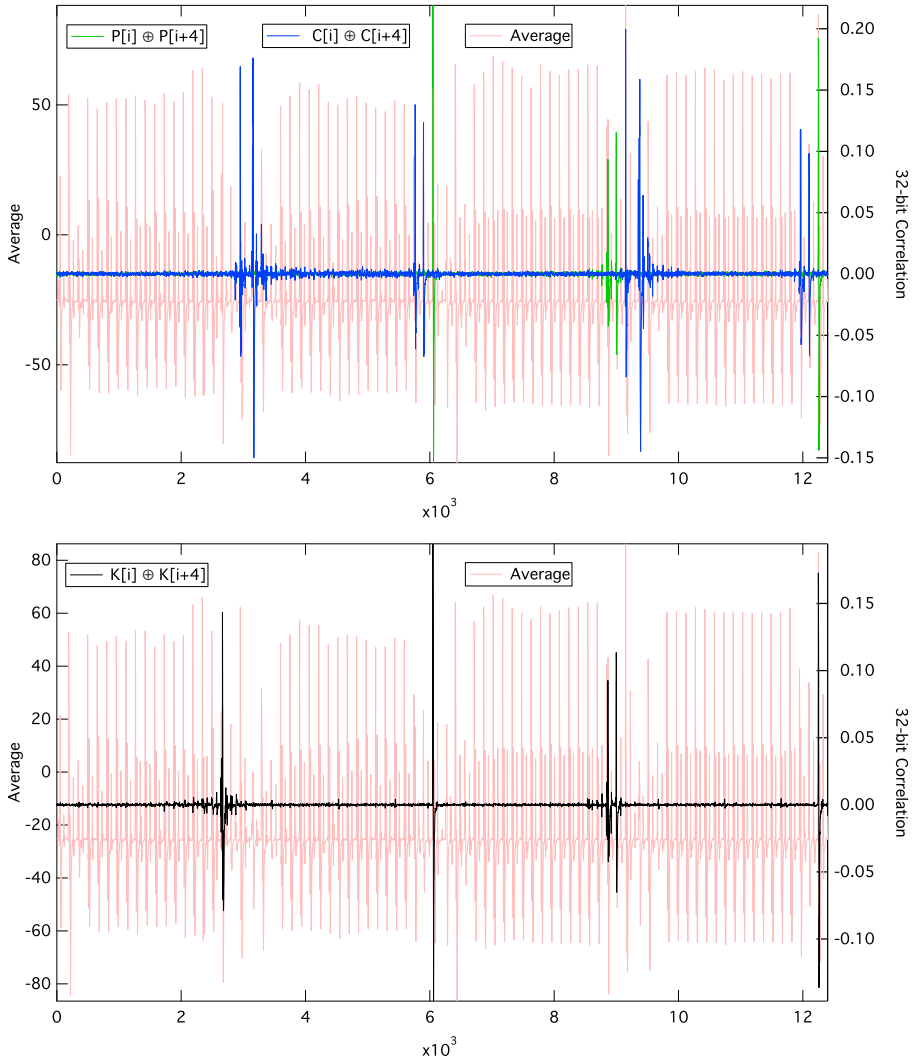

Fig. 4. Top: The 32–bit correlations of $P[i] \oplus P[i+4]$ and $C[i] \oplus C[i+4]$, $i = 0...3$, show strong peaks up to $\approx 20\%$. Bottom: The 32–bit correlation of $K[i] \oplus K[i+4]$, $i = 0...3$, shows also strong peaks but to a large extent at the same positions and similar strength as $P[i] \oplus P[i+4]$.

Eq. (1) as discussed in Sec. 2. Plotted is the average ranking as a function of the number of traces $\tilde{n}$ used. Here the average is always taken over all possible 256 values of the patterns. To be more precise, e.g., $\tilde{n} = 1024$ in Fig. 5, top part, means that $C[0] \oplus C[4]$ is kept fixed for 1024 traces, whilst $C[i] \oplus C[i+4]$, $i = 1...3$, is random because of the randomly chosen plain text, and this for

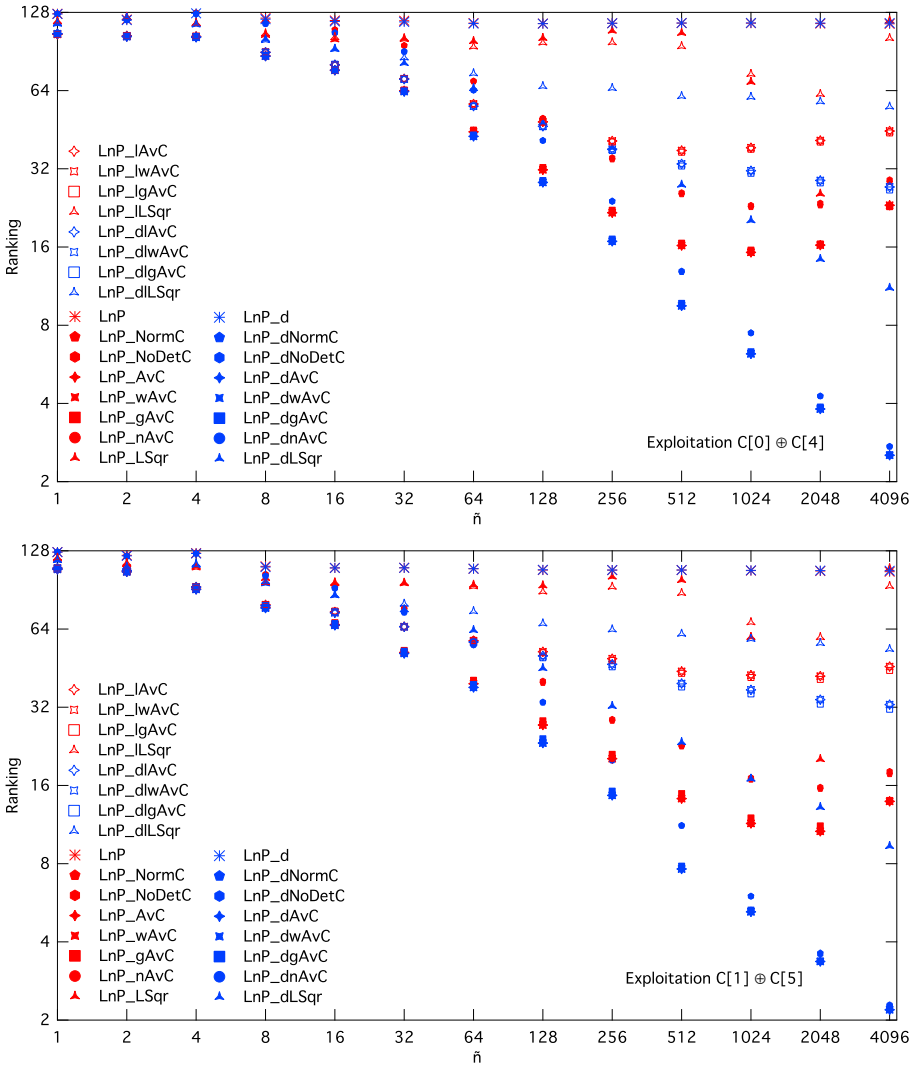

Fig. 5. Average rankings in Exploitation Phase for $C[i] \oplus C[i+4]$, $i = 0...1$, as a function of the number of traces used.

all 256 values of $C[0] \oplus C[4]$. In the end, an average is taken over all those 256 ranking results.

The traces used in Exploitation Phase had random plain text as input, and hence these results cannot be quite straight–forwardly applied to a situation where the entire plain text — and hence cipher text — is constant. But still, the results indicate strong leakage. The best performing approximations to Eq. (1) are `LnP_dAvC`, `LnP_dwAvC`, `LnP_dnAvC` and `LnP_dgAvC`, i.e., all approximations where some form of averaging has been performed over all covariances before

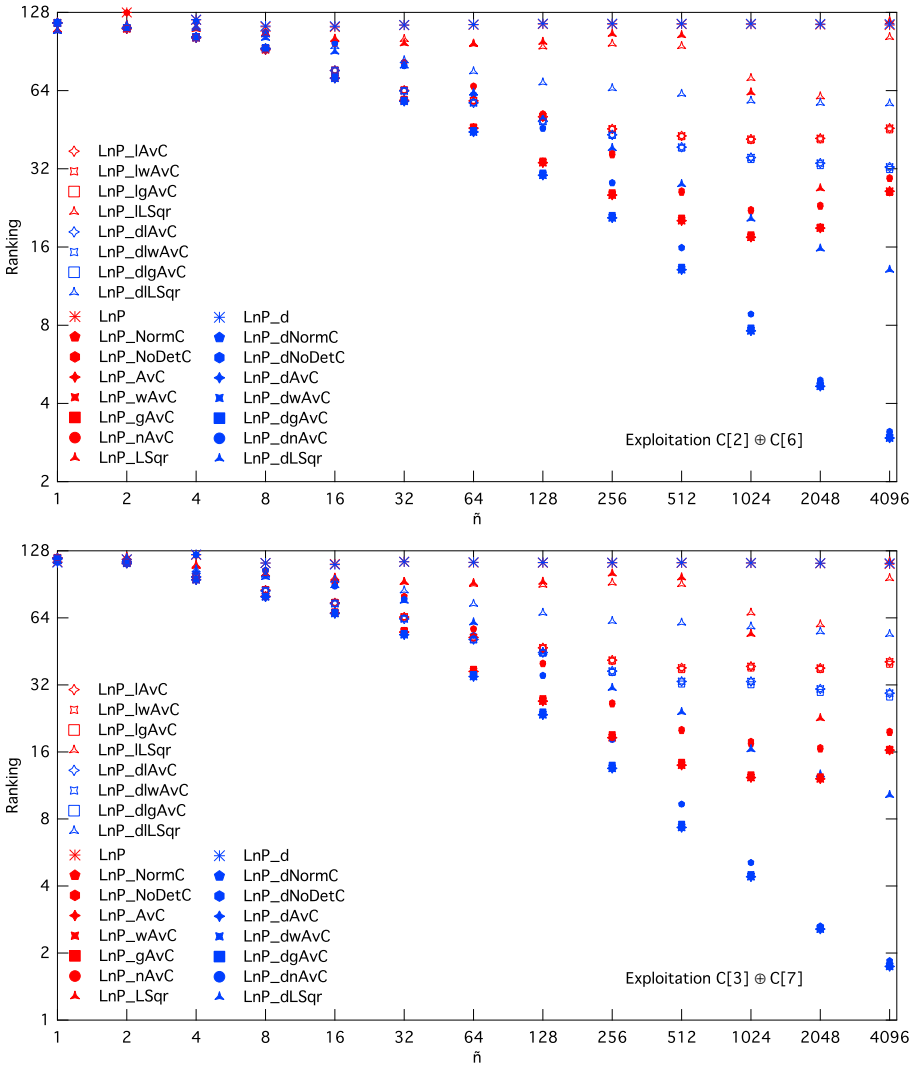

Fig. 6. Average rankings in Exploitation Phase for $C[i] \oplus C[i+4]$, $i = 2...3$, as a function of the number of traces used.

inversion, followed by a treatment for reducing the effect of constant offsets between Profiling and Exploitation Phase. Close runners up are `LnP_dNormC` and `LnP_dNoDetC`. In contrast, the original definition Eq. (1), i.e. `LnP`, is not effective at all.

In Figs. (7–8) we perform the same analysis for the plain text, based on 193 POI. Again, the same approximations as for the cipher text perform best. As expected, the results are not as good as for the attack on the cipher text, most likely because there is less leakage to begin with (number of POI is substantially

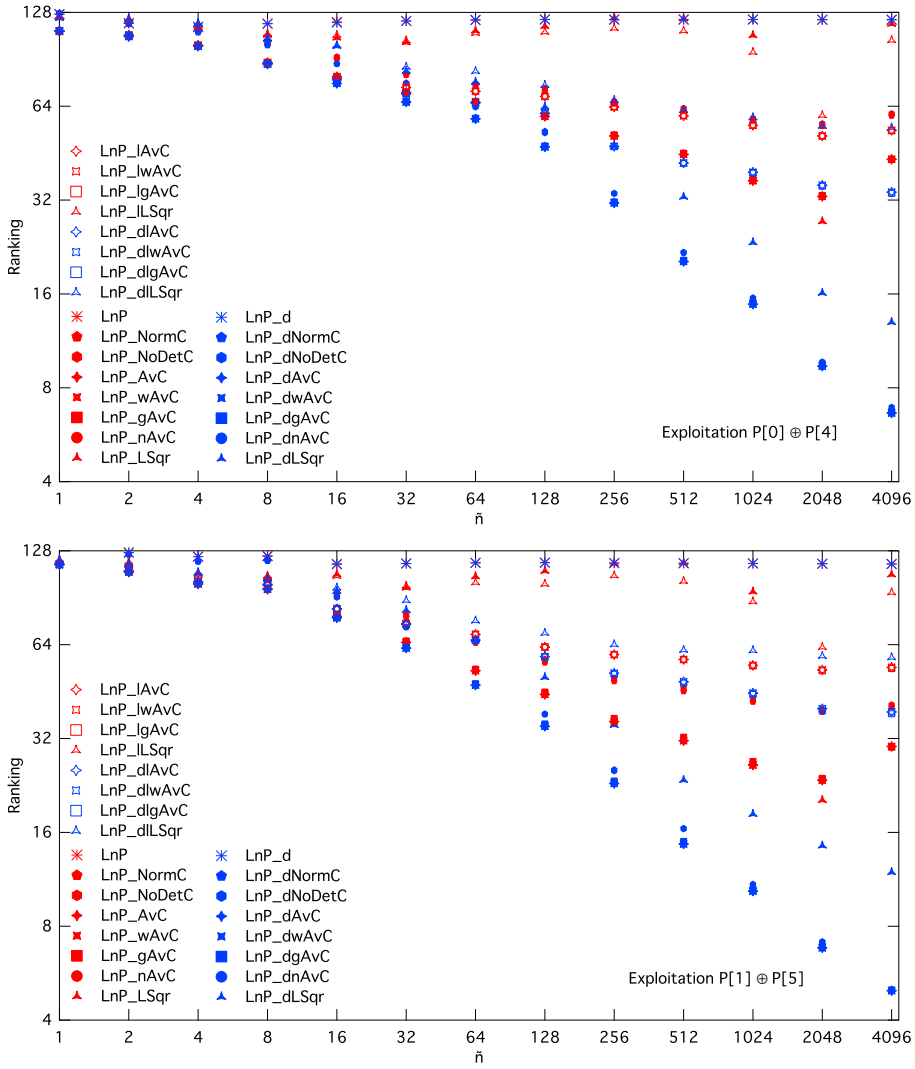

Fig. 7. Average rankings in Exploitation Phase for $P[i] \oplus P[i+4]$, $i = 0...1$, as a function of the number of traces used.

smaller) and, secondly, because the plain text leakage happens at the same time as the key leakage, which is a smart design choice.

An obvious use case for attacking the plain text is the attack on ISO9797-1 MAC [12] as scetched in Fig. 9. In fact, in this case one can do better than what is shown in Figs. (7–8), since for this chosen–plain–text attack one is actually trying to retrieve the output of the previous (i.e., first) DES calcuation, C1, which is added with $\oplus$ to the input P2 of the second stage, and hence the plain

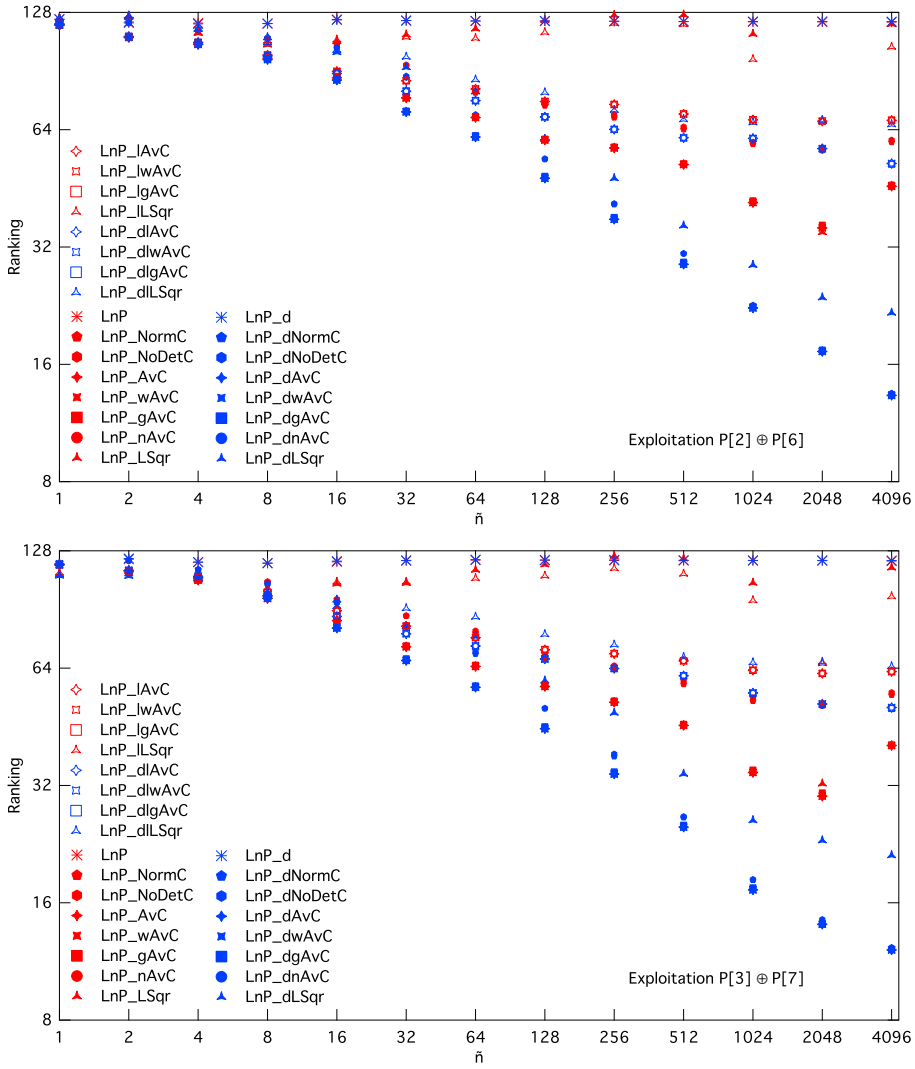

Fig. 8. Average rankings in Exploitation Phase for $P[i] \oplus P[i+4]$, $i = 2...3$, as a function of the number of traces used.

text input of the second DES call is interrelated via a simple $\oplus$, and thus Eq. (9) applies.

The fact that only half of the plain text can be recovered with the current attack does not matter, since for the attack on ISO9797-1 MAC one then simply takes twice as many traces eventually, using two different but fixed inputs IV and P1 to the first DES call. This will then mean twice as much effort in the remaining brute–force attack, but this is negligible.

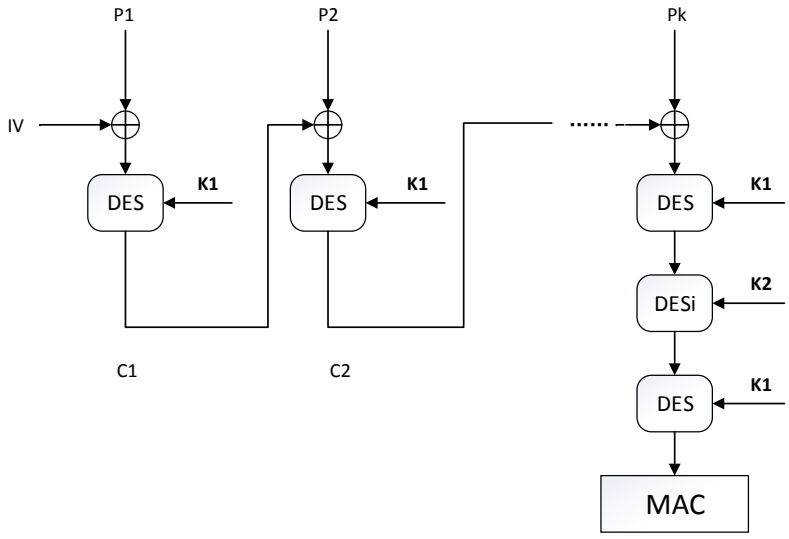

Fig. 9. ISO9797-1 MAC: IV, P1, P2 are all known and can be freely chosen. The attack according to [12] consists of a first step in which a side–channel attack in performed on the input of the second DES, which can be mapped back to the cipher text of the first DES, C1, and knowing IV and P1 a final brute–force attack is required to obtain the secret key K1 of the first DES.

Tables 4 and 5 show the results based on Eq. (9) and a chosen–plain–text attack. Here, $\tilde{n} = 256$ means that each of the 256 possible pattern values has been selected precisely once for a given target byte, whilst $\tilde{n} = 1024$ means each pattern value has been selected precisely four times.

For all four targets $P[i] \oplus P[i+4]$, $i = 0...3$, the two approximations performing consistently best are `LnP_lLSqr` and `LnP_dlLSqr`, which require between 256 and 2048 traces to achieve top rankings. It should be noted that although all the plain text bytes need to be chosen independently for each trace and each target byte anew, at least the choices for the *first* 256 traces can be the same for all four target ($\oplus$) bytes. Hence, those 256 traces can be reused and do not have to be counted four times when working out the total number of traces required to perform this attack.

In principle, it is even possible to use the cipher text results shown in Figs. (5–6), when applied to the first DES in the attack on ISO9797-1 MAC, as shown in Fig. 9, to reduce the number of required traces even further. Hence, in the end roughly 4K traces are required to have enough reliable plain text / cipher text relations to perform a brute–force attack on the first (single) DES. This remaining brute–force attack requires twice the normal effort because of the fact that, eventually, only $\oplus$ relations of the cipher text of the first DES are known via this template attack, but not the values themselves.

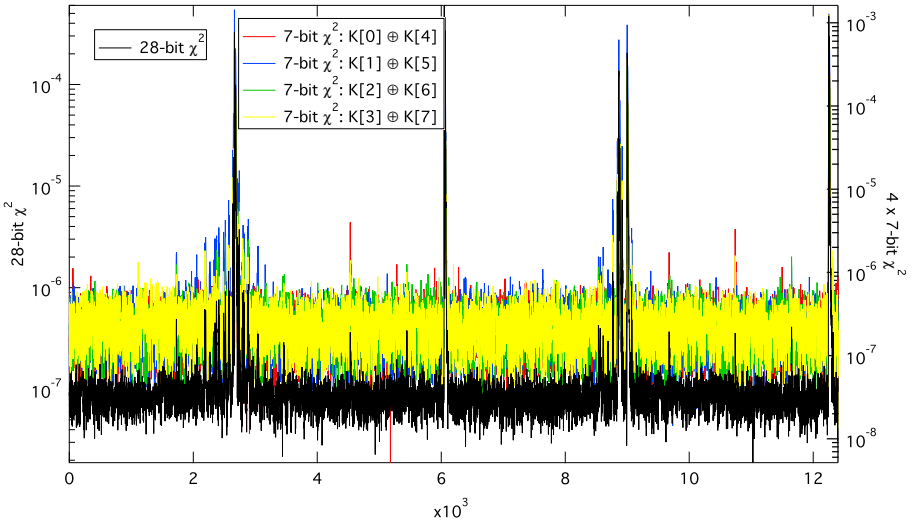

Fig. 10. χ^2 leakage for $K[i] \oplus K[i+4]$, $i = 0...3$, individually, as well as aggregated over all 28 bits.

Next, we investigate the leakage of the key more closely. In Fig. 10 we have plotted the χ^2 leakage for each 7-bit target $K[i] \oplus K[i+4]$, $i = 0...3$, separately, as well as the χ^2 leakage aggregated over all 28 bits. Four regions leak particularly, but there are some smaller peaks also seen inbetween, where, e.g., $K[0] \oplus K[4]$ leaks, but none of the other three does.

Figs. 11 and 12 show the rankings obtained for $K[i] \oplus K[i+4]$, $i = 0...3$, for a fixed but randomly chosen key, using 1696 POI that had been derived from a χ^2 analysis of Fig. 10. The fact that the key is fixed and only the plain text is randomly chosen does have a detrimental effect on the strength of the attack. Only some of the 24 approximations to Eq. (1) seem to converge to better ranking values, but none achieves top rank 1. For instance, `LnP_d` stays fairly constant at rankings 5, 15, 36, and 16, respectively. On the other hand, the approximations `LnP_lAvC`, `LnP_lwAvC`, and `LnP_lgAvC` start out somewhat poorly, but then converge to values $\approx 11...32$. This means a little over 2 bits leak per 7 bits, or some 8 – 10 bits of the 56–bit key, and this for as little as a few hundred traces (or even a single trace[3] in the case of `LnP_d`).

Presumably, this lack of convergence has to do with the fact that the templates chosen are smaller than the number of fixed bits (i.e., bits that are kept constant for all traces) that leak at the same time, and thus can "interfere" with each other (cross–talk). For instance, whilst according to Fig. 4 the leakage correlation amplitudes for plain text, cipher text as well as key are all of similar amplitude, the plain and cipher text turn out to be much more vulnerable

[3] If an attack is successful in a single trace during Exploitation Phase, there are no software countermeasures possible since, one way or another, those all rely on averaging over many traces. The only solution then is to fix the hardware.

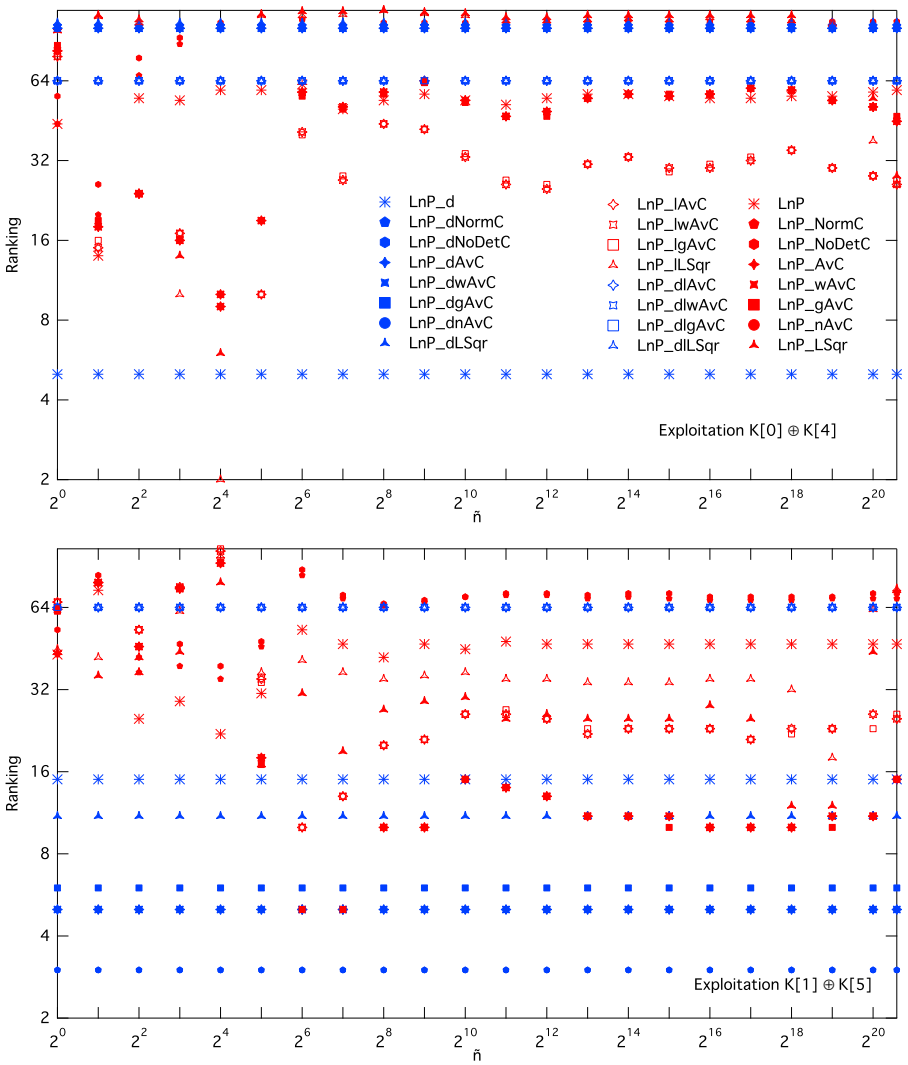

Fig. 11. Average rankings in Exploitation Phase for $K[i] \oplus K[i+4]$, $i = 0...1$, as a function of the number of traces used.

to template attacks than the key when comparing Figs. 5 – 8 with Figs. 11 – 12. The difference is that for the attack on the key 28 bits needed to be kept constant, whilst for plain and cipher text only 8 bits were kept constant at a time. Yet, the size of the templates was nearly the same — 7 respectively 8 bits. If this hypothesis is true, then the results for the attack on the key should improve when increasing the template size relatively to those fixed but leaky bits.

For instance, one could attack the Hamming weight of the entire 28–bit vector[4], or use larger template sizes, e.g., 14 bits per template. In any case, since the key is attacked directly, these results are straight–forwardly applicable to TDES without the need to invoke meet–in–the–middle attacks and the like.

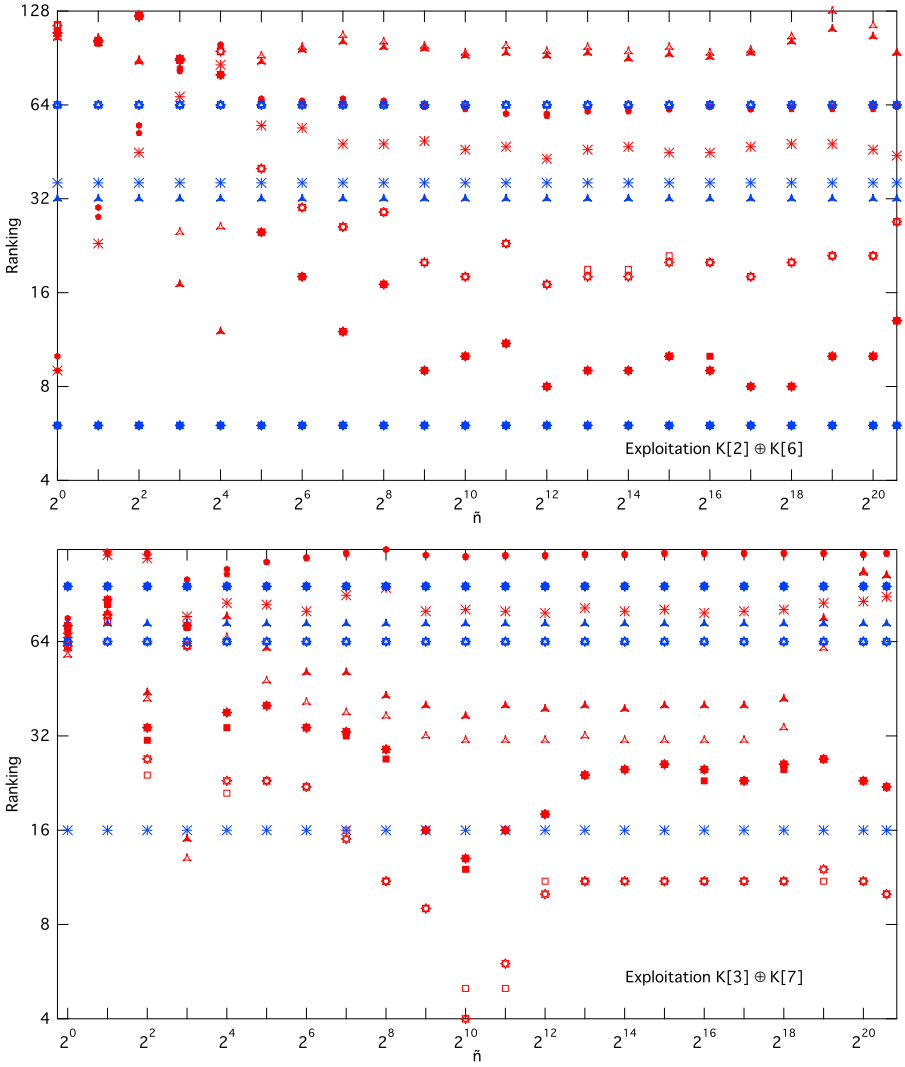

Fig. 12. Average rankings in Exploitation Phase for $K[i] \oplus K[i+4]$, $i = 2...3$, as a function of the number of traces used.

[4] Preliminary results indicate, though, that also when attacking the Hamming weight of all 28 bits, rankings do not seem to improve when increasing the number of traces used in the Exploitation Phase.

It is, of course, possible to combine this leakage of the key with the leakage of the plain and cipher text to create an even stronger attack on ISO9797-1 MAC, since it would help to reduce the remaining brute–force attack on the key by performing the brute–force attack on the ranked lists.

4 Conclusions

In conclusion we have shown that it is worth spending some effort on analysing a variety of approximations to the standard probability density function of the multivariate normal distribution, Eq. (1), when applying this to template attacks. Some approximations yield substantially and consistently so better results than others. Generally, approximations based on averaged covariance matrices seem to perform consistently better, and measures to reduce the effects of constant offsets between templates and patterns are also beneficial. In any case, the attacker can always study during the Profiling Phase which approximation is likely to yield the best results in the Exploitation Phase.

As an example, we have studied the attack on the ISO9797-1 MAC, applied to a fairly recent smart card, which exhibits leakage of plain text, cipher text, as well as the secret key itself during a DES calculation. Only some 4K traces are required to perform the attack, and this does not yet include the possibility of making trade-offs between the number of measurement traces taken on the one hand, and the remaining brute–force effort required on the other hand.

The analysis of the key leakage requires more work. Preliminary results indicate leakage of about $6 - 10$ bits per DES, and thus $18 - 30$ bits per TDES.

For this particular platform the issue becomes even more severe as it is using a DES hardware coprocessor built in what is called a hard–macro design technology, meaning that it is a design block with fixed geometry all the way down to the gate level. This hard macro is thus visible in the layout, and it is used in many other products in precisely the same shape over and over again. Hence, once a weakness such as this one has been found in one family member, it can be found in all other family members as well. Worse still, one has to assume that templates generated for one family member will be applicable for all other family members as well.

References

1. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M.J. (ed.) Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings. Lecture Notes in Computer Science, vol. 1666, pp. 388–397. Springer (1999), http://dx.doi.org/10.1007/3-540-48405-1_25
2. Messerges, T.S.: Using second-order power analysis to attack DPA resistant software. In: Koç, Ç.K., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings. Lecture Notes in Computer Science, vol. 1965, pp. 238–251. Springer (2000), http://dx.doi.org/10.1007/3-540-44499-8_19

3. Gierlichs, B., Batina, L., Preneel, B., Verbauwhede, I.: Revisiting higher-order DPA attacks:. In: Pieprzyk, J. (ed.) Topics in Cryptology - CT-RSA 2010, The Cryptographers' Track at the RSA Conference 2010, San Francisco, CA, USA, March 1-5, 2010. Proceedings. Lecture Notes in Computer Science, vol. 5985, pp. 221–234. Springer (2010), http://dx.doi.org/10.1007/978-3-642-11925-5_16
4. Xiao, L., Heys, H.M.: A simple power analysis attack against the key schedule of the camellia block cipher. Inf. Process. Lett. 95(3), 409–412 (2005), http://dx.doi.org/10.1016/j.ipl.2005.03.013
5. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Jr., B.S.K., Koç, Ç.K., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers. Lecture Notes in Computer Science, vol. 2523, pp. 13–28. Springer (2002), http://dx.doi.org/10.1007/3-540-36400-5_3
6. Archambeau, C., Peeters, E., Standaert, F., Quisquater, J.: Template attacks in principal subspaces. In: Goubin, L., Matsui, M. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4249, pp. 1–14. Springer (2006), http://dx.doi.org/10.1007/11894063_1
7. Medwed, M., Oswald, E.: Template attacks on ECDSA. In: Chung, K., Sohn, K., Yung, M. (eds.) Information Security Applications, 9th International Workshop, WISA 2008, Jeju Island, Korea, September 23-25, 2008, Revised Selected Papers. Lecture Notes in Computer Science, vol. 5379, pp. 14–27. Springer (2008), http://dx.doi.org/10.1007/978-3-642-00306-6_2
8. Fouque, P., Leurent, G., Réal, D., Valette, F.: Practical electromagnetic template attack on HMAC. In: Clavier, C., Gaj, K. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings. Lecture Notes in Computer Science, vol. 5747, pp. 66–80. Springer (2009), http://dx.doi.org/10.1007/978-3-642-04138-9_6
9. Choudary, O., Kuhn, M. G., Efficient Template Attacks in Smart Card Research and Advanced Applications. Springer International Publishing, 2013:253-270
10. Hu, Y.B., Zhang, C., Zheng, Y.Y., Wagner, M.: Ciphertext and Plaintext Leakage Reveals the Entire TDES Key. Cryptology ePrint Archive, Report 2016/1143, 2016
11. http://en.wikipedia.org/wiki/Bra-ket_notation
12. Feix, B., Thiebeauld, H.: Defeating ISO9797-1 MAC algo 3 by combining side-channel and brute force techniques. Cryptology ePrint Archive, Report 2014/702, 2014
13. Bayes, M., Price, M.: An Essay towards Solving a Problem in the Doctrine of Chances. By the Late Rev. Mr. Bayes, Communicated by Mr. Price, in a Letter to John Canton, Resonance, 2003, 95(4), pp 11–60.

Table 4. Rankings for $P[0] \oplus P[4]$ and $P[1] \oplus P[5]$ based on Eq. (9).

$\tilde{n}$	LnP	LnP_NormC	LnP_NoDetC	LnP_AvC	LnP_wAvC	LnP_gAvC	LnP_nAvC	LnP_LSqr
256	101	108	101	1	1	1	1	5
512	3	3	3	1	1	1	1	1
1024	1	1	1	1	1	1	1	1
2048	1	1	1	1	1	1	1	1
4096	2	2	2	1	1	1	1	1
8192	1	1	1	1	1	1	1	1
16384	1	1	1	1	1	1	1	1
$\tilde{n}$	LnP_d	LnP_dNormC	LnP_dNoDetC	LnP_dAvC	LnP_dwAvC	LnP_dgAvC	LnP_dnAvC	LnP_dLSqr
256	100	106	100	1	1	1	1	5
512	3	3	3	1	1	1	1	1
1024	1	1	1	1	1	1	1	1
2048	1	1	1	1	1	1	1	1
4096	2	2	2	1	1	1	1	1
8192	1	1	1	1	1	1	1	1
16384	1	1	1	1	1	1	1	1
$\tilde{n}$	LnP_lAvC	LnP_lwAvC	LnP_lgAvC	LnP_lLSqr	LnP_dlAvC	LnP_dlwAvC	LnP_dlgAvC	LnP_dlLSqr
256	1	1	1	5	1	1	1	5
512	1	1	1	1	1	1	1	1
1024	1	1	1	1	1	1	1	1
2048	1	1	1	1	1	1	1	1
4096	1	1	1	1	1	1	1	1

$\tilde{n}$	LnP	LnP_NormC	LnP_NoDetC	LnP_AvC	LnP_wAvC	LnP_gAvC	LnP_nAvC	LnP_LSqr
256	196	188	196	3	3	3	3	7
512	79	76	79	1	1	1	1	5
1024	23	22	23	1	1	1	1	1
2048	17	17	17	1	1	1	1	1
4096	1	1	1	1	1	1	1	1
8192	1	1	1	1	1	1	1	1
16384	1	1	1	1	1	1	1	1
$\tilde{n}$	LnP_d	LnP_dNormC	LnP_dNoDetC	LnP_dAvC	LnP_dwAvC	LnP_dgAvC	LnP_dnAvC	LnP_dLSqr
256	195	182	195	3	3	3	3	7
512	67	62	67	1	1	1	1	5
1024	21	20	21	1	1	1	1	1
2048	15	15	15	1	1	1	1	1
4096	1	1	1	1	1	1	1	1
8192	1	1	1	1	1	1	1	1
16384	1	1	1	1	1	1	1	1
$\tilde{n}$	LnP_lAvC	LnP_lwAvC	LnP_lgAvC	LnP_lLSqr	LnP_dlAvC	LnP_dlwAvC	LnP_dlgAvC	LnP_dlLSqr
256	3	3	3	7	3	3	3	7
512	1	1	1	5	1	1	1	5
1024	1	1	1	1	1	1	1	1
2048	1	1	1	1	1	1	1	1
4096	1	1	1	1	1	1	1	1

Table 5. Rankings for $P[2] \oplus P[6]$ and $P[3] \oplus P[7]$ based on Eq. (9).

$\tilde{n}$	LnP	LnP_NormC	LnP_NoDetC	LnP_AvC	LnP_wAvC	LnP_gAvC	LnP_nAvC	LnP_LSqr
256	34	36	34	2	2	2	2	51
512	12	10	12	1	1	1	1	4
1024	4	4	4	1	1	1	1	2
2048	1	1	1	1	1	1	1	1
4096	72	73	72	1	1	1	1	1
8192	6	6	6	1	1	1	1	1
16384	1	1	1	1	1	1	1	1
$\tilde{n}$	LnP_d	LnP_dNormC	LnP_dNoDetC	LnP_dAvC	LnP_dwAvC	LnP_dgAvC	LnP_dnAvC	LnP_dLSqr
256	38	37	38	2	2	2	2	51
512	10	10	10	1	1	1	1	4
1024	4	4	4	1	1	1	1	2
2048	1	1	1	1	1	1	1	1
4096	63	63	63	1	1	1	1	1
8192	3	3	3	1	1	1	1	1
16384	1	1	1	1	1	1	1	1
$\tilde{n}$	LnP_lAvC	LnP_lwAvC	LnP_lgAvC	LnP_lLSqr	LnP_dlAvC	LnP_dlwAvC	LnP_dlgAvC	LnP_dlLSqr
256	2	2	2	51	2	2	2	51
512	1	1	1	4	1	1	1	4
1024	1	1	1	2	1	1	1	2
2048	1	1	1	1	1	1	1	1
4096	1	1	1	1	1	1	1	1

$\tilde{n}$	LnP	LnP_NormC	LnP_NoDetC	LnP_AvC	LnP_wAvC	LnP_gAvC	LnP_nAvC	LnP_LSqr
256	129	121	129	5	5	5	5	1
512	21	15	21	1	1	1	1	1
1024	1	1	1	3	3	3	3	1
2048	2	2	2	2	2	2	2	1
4096	1	1	1	1	1	1	1	1
8192	4	4	4	1	1	1	1	1
16384	1	1	1	1	1	1	1	1
$\tilde{n}$	LnP_d	LnP_dNormC	LnP_dNoDetC	LnP_dAvC	LnP_dwAvC	LnP_dgAvC	LnP_dnAvC	LnP_dLSqr
256	134	130	134	5	5	5	5	1
512	29	26	29	1	1	1	1	1
1024	1	1	1	3	3	3	3	1
2048	2	2	2	2	2	2	2	1
4096	1	1	1	1	1	1	1	1
8192	3	3	3	1	1	1	1	1
16384	1	1	1	1	1	1	1	1
$\tilde{n}$	LnP_lAvC	LnP_lwAvC	LnP_lgAvC	LnP_lLSqr	LnP_dlAvC	LnP_dlwAvC	LnP_dlgAvC	LnP_dlLSqr
256	5	5	5	1	5	5	5	1
512	1	1	1	1	1	1	1	1
1024	3	3	3	1	3	3	3	1
2048	2	2	2	1	2	2	2	1
4096	1	1	1	1	1	1	1	1