Multi-target DPA attacks: Pushing DPA beyond the limits of a desktop computer

Luke Mather, Elisabeth Oswald, and Carolyn Whitnall

Department of Computer Science, University of Bristol,
Merchant Venturers Building, Woodland Road,
Bristol, BS8 1UB, United Kingdom.
{luke.mather, elisabeth.oswald, carolyn.whitnall}@bris.ac.uk

Abstract. Following the pioneering CRYPTO '99 paper by Kocher et al. differential power analysis (DPA) was initially geared around low-cost computations performed using standard desktop equipment with minimal reliance on device-specific assumptions. In subsequent years, the scope was broadened by, e.g., making explicit use of (approximate) power models. An important practical incentive of so-doing is to reduce the data complexity of attacks, usually at the cost of increased computational complexity. It is this trade-off which we seek to explore in this paper. We draw together emerging ideas from several strands of the literature—high performance computing, post-side-channel global key enumeration, and effective combination of separate information sources—by way of advancing (non-profiled) 'standard DPA' towards a more realistic threat model in which trace acquisitions are scarce but adversaries are well resourced. Using our specially designed computing platform (including our parallel and scalable DPA implementation, which allows us to work efficiently with as many as 2^{32} key hypotheses), we demonstrate some dramatic improvements that are possible for 'standard DPA' when combining DPA outcomes for several intermediate targets. Unlike most previous 'information combining' attempts, we are able to evidence the fact that the improvements apply even when the exact trace locations of the relevant information (i.e. the 'interesting points') are not known *a priori* but must be searched simultaneously with the correct subkey.

1 Introduction

Differential power analysis (DPA) was initially conceived as a computationally 'cheap' way to recover secret information from side-channel leakages, under the assumption that trace measurements could be easily acquired [14]. Over time, the emphasis has changed and several directions have been pursued in the literature, e.g. attacks using power models [6] and attacks using several trace points [7] ([15] surveys the many variations of DPA style attacks). Across all these directions, one 'measure' of attack success has emerged and now dominates the scientific

discourse with regards to attack efficiency. This measure is the number of power traces needed to identify the correct (sub)key[1].

What is the purpose of considering (sub)key recovery attacks? From a practical perspective any strategy is considered successful if it reveals 'enough' information about the (global) key to enable a brute force search. It is crucial that side-channel resistance, like other aspects of security, be considered with respect to realistic threat models. Real world adversaries are then (arguably) mostly interested in exploring trade-offs between the number of leakage traces available and the computational resources dedicated to extracting as much information as possible from those traces. Recent work by Veyrat-Charvillon et al. [24,25] present an algorithm for searching the candidate space containing the key and a means to estimate its size if the enumeration capabilities of the analyst are below those of a better-resourced adversary.

Resources, from the point of view of a contemporary DPA adversary, include not only sophisticated measurement equipment but crucially also processing capabilities that directly map to the *time* necessary to mount and complete attacks [8]. Moradi et al.'s recent work [17] demonstrates how the use of a handful of modern graphics cards allows for dramatic increase in processing capabilities, enabling an attack on 32-bit key hypotheses in a known point scenario (the leakage point corresponding to the attacked operation was determined *a priori* via a known key attack).

In this submission we explore the possibilities for sophisticated use of modern processing capabilities (such as those associated with high performance computing (HPC), albeit restricted to the setting of a few machines or a 'small' cluster) to facilitate 'multi-target' DPA attacks. Multi-target DPA consists in amalgamating outcomes from multiple single-target attacks with the aim of reducing global key entropy more quickly than an individual single-target attack. For example, against a sequential AES implementation, multi-target DPA could amalgamate the outcomes of standard attacks on the AddRoundKey, SubBytes, and MixColumns operations. We will show later that we can do this meaningfully, and also efficiently, for correlation-based DPA attacks—even in realistic scenarios where the exact leakage points for those target functions are not known and must each be searched within windows of the trace. Most importantly, we show that such attacks can dramatically out-perform single-target attacks and are by far the best strategy to minimise the number of leakage traces required.

1.1 Our contribution

An adversary who is capable of attacking large numbers of key hypotheses has a greater choice of intermediate target functions to attack. For instance, possible AES targets include the output of AES MixColumns (involving four bytes of the secret key) as well as the (implementation-dependent) intermediate computations for MixColumns (involving two or three key bytes at once). Given

[1] The overall key recovery works according to a divide-and-conquer strategy; each (for example) byte of the key is attacked and recovered individually.

the potential plethora of intermediate value combinations for a sequential AES implementation (as typically found on micro-processors) we investigate the effectiveness of some of the possible combinations with respect to the reduction on key guessing entropy. We also touch on the possibility of combining different distinguisher outputs and explain when this is (or is not) going to be helpful.

We also take inspiration from the suggestion of Veyrat-Charvillon et al. [24,25] (originally for the purposes of a key enumeration algorithm) that *probability distributions* on the subkeys can be derived from the outcome of a DPA attack. We propose an alternative (more conservative) heuristic for assigning 'probability' scores to subkeys, and show how these can be used to simply and usefully combine information from multiple standard univariate DPA attacks in a strategy inspired by Bayesian updating.

This research is rooted in our developed capability to efficiently process large numbers of key hypotheses over many repeat experiments; our architecture (which we sketch out) is influenced by the design of modern HPC platforms.

We structure our contribution as follows. We briefly provide the relevant preliminaries and then discuss prior literature (Section 2). We then introduce our specialised attack framework and explain our attack strategy, including our method of assigning and updating 'probability' scores, in Section 3. Section 4 reports the results of our experiments with simulated leakage data, exploring what can be achieved by combining the outcomes of attacks against different target functions, as well as investigating the potential to combine different DPA *strategies*. In Section 5 we report the outcomes of some practical attacks against traces measured from an ARM 7 microcontroller, including scenarios in which the precise locations of the intermediate targets in the traces are unknown.

1.2 Preliminaries: Differential power analysis

We consider a 'standard DPA attack' scenario as defined in [16], and briefly explain the underlying idea as well as introduce the necessary terminology here. We assume that the power consumption P of a cryptographic device depends on some internal value (or state) $F_{k^*}(X)$ which we call the *target*: a function $F_{k^*} : \mathcal{X} \to \mathcal{Z}$ of some part of the known plaintext—a random variable $X \overset{R}{\in} \mathcal{X}$—which is dependent on some part of the secret key $k^* \in \mathcal{K}$. Consequently, we have that $P = L \circ F_{k^*}(X) + \varepsilon$, where $L : \mathcal{Z} \to \mathbb{R}$ describes the data-dependent component and ε comprises the remaining power consumption which can be modeled as independent random noise (this simplifying assumption is common in the literature—see, again, [16]). The attacker has N power measurements corresponding to encryptions of N known plaintexts $x_i \in \mathcal{X}$, $i = 1, \ldots, N$ and wishes to recover the secret key k^*. The attacker can accurately compute the internal values as they would be under each key hypothesis $\{F_k(x_i)\}_{i=1}^N$, $k \in \mathcal{K}$ and uses whatever information he possesses about the true leakage function L to construct a prediction model $M : \mathcal{Z} \to \mathcal{M}$.

DPA is motivated by the intuition that the model predictions under the correct key hypothesis should give more information about the true trace mea-

surements than the model predictions under an incorrect key hypothesis. A distinguisher D is some function which can be applied to the measurements and the hypothesis-dependent predictions in order to quantify the correspondence between them. For a given such comparison statistic, D, the *estimated* vector from a practical instantiation of the attack is $\hat{\mathbf{D}}_N = \{\hat{D}_N(L \circ F_{k^*}(\mathbf{x}) + \mathbf{e}, M \circ F_k(\mathbf{x}))\}_{k \in \mathcal{K}}$ (where $\mathbf{x} = \{x_i\}_{i=1}^N$ are the known inputs and $\mathbf{e} = \{e_i\}_{i=1}^N$ is the observed noise). Then the attack is *o-th order successful* if $\#\{k \in \mathcal{K} : \hat{\mathbf{D}}_N[k^*] \leq \hat{\mathbf{D}}_N[k]\} \leq o$.

The *success rate* of a DPA attack is the probability that the correct key is ranked first by the distinguisher (the *o-th order success rate* is the probability it is ranked among the o first candidates); the *guessing entropy* is the expected number of candidates to test before reaching the correct one [23]. These metrics are often associated with the *subkeys* targeted in the 'divide-and-conquer' paradigm rather than with the global key when the partial outcomes are finally combined; we use the terms accordingly, unless explicitly stated.

Unless stated otherwise, we use the (estimate of the) Pearson correlation coefficient as distinguisher, in combination with a Hamming weight power model.

2 Related literature

Our work unites and advances three broad areas of the literature: resource-intensive side-channel strategies, post-SCA global key enumeration, and optimal combination of multiple sources of exploitable information.

Resource-intensive strategies. Such strategies have for a long time been considered mainly relevant in single-trace settings (e.g. SPA attacks using algebraic methods [18,19]); this has only lately begun to change, with a few recent studies making use of modern graphics cards to speed up DPA attacks [3,17]. These articles essentially use GPUs within a single machine to speed up the processing of standard correlation DPA attacks. Our more ambitious approach is to distribute all the different components of a DPA attack (*including* workloads related to combination functionality) across several cards and several machines.

Post-SCA global key enumeration. Recent work by Veyrat-Charvillon et al. [24,25] focuses on the opportunity for a well-resourced adversary to view side-channel analysis as an auxiliary phase in an enhanced global key search, rather than a stand-alone 'win-or-lose' attack. They present an algorithm for searching, based on probability distributions for each of the subkeys (derived from DPA outcomes) [24]. In the case of profiling DPA with Gaussian templates, the true leakage distributions conditioned on each subkey hypothesis are known, and the probabilities are naturally produced in the Bayesian template matching. In the case of non-profiling DPA, these conditional leakage distributions are *not* known; an attack does *not* produce a probability distribution on the subkey candidates but a set of distinguisher scores (for example, correlations) associated with each candidate. Deriving probabilities from these scores is tricky; the method suggested in [24] is to use the hypothesis-dependent fitted leakage models after a

non-profiled linear regression ('stochastic') attack as estimates on the 'true' conditional distributions. However, non-profiled linear regression-based DPA specifically relies on the fact that the models built under incorrect key hypotheses are *invalid*. Consequently, the hypothesised functions do *not* describe the true data-dependent deterministic behaviour of the trace measurements, and so they are useless for (statistical) inference. For this reason, we opt for a different ('safer') heuristic for assigning 'probability' scores, as explained in Section 3.1.

Combining multiple sources of information. Whilst profiling attacks with multivariate Gaussian templates [7] naturally exploit multiple trace points, notions of 'multivariate' non-profiled DPA are varied in nature and intention. In particular, techniques designed to defeat protected implementations are best considered separately from attempts to enhance trace efficiency, and we now focus on the latter. Already in an unprotected implementation, information on a given subkey generally leaks via more than one target function (AddRoundKey, SubBytes and MixColumns, for example, in the case of AES) and moreover each of those target functions can be seen to leak at more than one trace point. In some cases, an adversary may even have opportunity to observe multiple side-channels simultaneously (timing, power consumption, electromagnetic radiation...).

In the realms of both profiled and non-profiled DPA, several efforts have been made to combine information from multiple trace points in such a way as to optimise the (trace) efficiency of an attack. Dimensionality reduction techniques such as principal component analysis or linear discriminant analysis can be used to transform the (often collinear) trace measurements into a reduced number of linearly uncorrelated variables, together accounting for the important variation in the original data [1,4,21]. In this way it is even possible to combine information from different side-channels, such as power and electro-magnetic radiation [21]. Such methods can be very effective if the leakage associated with a particular intermediate value is concentrated into a single component giving rise to a stronger attack outcome than the 'best' of any individual point in the raw dataset. A recent work by Hajra et al. [12] achieves a similar end via signal processing techniques. They show how to maximise the signal-to-noise ratio (SNR) (and consequently demonstrate the success rate of a univariate correlation DPA) by finding the linear Finite Impulse Response (FIR) filter coefficients for the leakage signal. Hutter et al. [13] also seek to enhance DPA efficiency by incorporating multiple sources of information, but take an entirely different approach in which the combination is instead made at the trace acquisition stage. They measure the *difference* in consumption between two identical devices operating on different data, which they reason has a higher data-dependent signal because all environmental and operation-dependent noise is cancelled out.

Other suggestions involve performing separate attacks (against different targets, power models or using different distinguishers) and then attempting to combine the *distinguishing vectors* themselves in a meaningful way. Doget et al. [9] present options for combining difference-of-means (DoM) style outcomes in order to avoid the 'suboptimality' associated with attacks exploiting only one or a few of the bits at a time. Whitnall et al. [27] try applying a multivariate

extension of the mutual information to the AddRoundKey and an S-box jointly, but find that it is *less* efficient than the corresponding attack against the S-box alone, and moreover would not scale easily beyond a two-target scenario due to the complex nature of the statistic. Souissi et al. [20] suggest to combine different *distinguishers* (namely, Pearson's and Spearman's correlation) applied against the same or different leakage points by taking either the sum or the maximum of the two, and show that the former works better, and is most effective if the trace points contain non-equivalent information. Most directly related to our study is a paper by Elaabid et al. [10] which suggests to (pointwise) *multiply* correlation distinguishing vectors together in order to enhance distinguishing outcomes. They do this for four *known* leakage points, all relating to the same target function and power model, and find that it substantially improves over the outcomes achieved for any one of those leakage points taken individually. Our own combining approach is different: we first convert distinguishing vectors to 'probability' scores and view the multiplication as a Bayesian updating-like procedure. Moreover, we focus on combinations between *different target values* (rather than different leakage points for the same value) with potentially different-sized subkey hypotheses.

3 Methodology

3.1 Assigning probabilities

The attempt of [24] to estimate 'genuine' probabilities on the subkey hypotheses in the non-profiled setting (see Section 2), by using the recovered models derived from a linear regression based attacks, is expensive as well as unsuitable for our purpose. Ignoring the fact that the incorrect key hypotheses (using their approach) recover invalid models, the method of [24,25] may be viewed as one possible heuristic to assign probabilities to key guesses. It preserves the ranking of the keys as they appear in the distinguishing vector produced by a non-profiled linear regression-based DPA. However, because of the nature of the formula used it dramatically exaggerates the apparent distance between the high- and low-ranked key candidates. If the implied key is the right one it reinforces this 'correct' result. But if it is not the right one it reinforces the misleading result. In their application (i.e. key enumeration) this may cause a less efficient key search. However, we are aiming to combine distinguisher results, and hence key rankings, and mixing in a grossly exaggerated incorrect key ranking may destroy the effectiveness of the method.

Embracing the heuristic nature of the task of obtaining (from distinguishing vectors) scores which may be handled as though probabilities, we suggest the conversion be kept simple and conservative. Our approach firstly transforms the distinguishing vector to be positive-valued with a baseline of zero (in a manner appropriate to the statistic—e.g. the absolute value for correlation, subtraction of the minimum for the mutual information) before secondly normalising the scores to sum to one. We draw analogy between this idea and the notion of *subjective*

probability basic to a Bayesian view of statistics: both involve human-allocated scores derived from one's current best knowledge about reality.

3.2 Combining probabilities

A Bayesian interpretation views probabilities as measures of uncertainty on hypotheses. Each time new information becomes available, the current state of knowledge can be *updated* via Bayes' theorem:

$$\mathbb{P}(H|B) = \frac{\mathbb{P}(B|H)\mathbb{P}(H)}{\mathbb{P}(B)},$$

where H is some hypothesis (for example, a guess on the key, "$K = k$"), and B is some data (for example, a set of trace measurements $\mathbf{l} = L \circ F_{k^*}(\mathbf{x}) + \mathbf{e}$).

Suppose that we have probabilities for $(K = k)$ conditioned on two sources of data $\mathbf{l}_1, \mathbf{l}_2$, which are conditionally independent given $K = k$ so that $\mathbb{P}(\mathbf{l}_1, \mathbf{l}_2|K = k) = \mathbb{P}(\mathbf{l}_1|K = k)\mathbb{P}(\mathbf{l}_2|K = k)$. This is a natural assumption for the leakages of two target intermediate values: they are related via their shared dependence on the underlying key, but as long as they are separated in the trace, we would not expect any dependency in the residual variances after the key is taken into account. In this case, the task of combining the conditional probabilities is straightforward (see [2]):

$$\begin{aligned}
\mathbb{P}(K = k|\mathbf{l}_1, \mathbf{l}_2) &= \frac{\mathbb{P}(\mathbf{l}_1, \mathbf{l}_2|K = k)\mathbb{P}(K = k)}{\mathbb{P}(\mathbf{l}_1, \mathbf{l}_2)} \\
&= \frac{\mathbb{P}(\mathbf{l}_1|K = k)\mathbb{P}(\mathbf{l}_2|K = k)\mathbb{P}(K = k)}{\mathbb{P}(\mathbf{l}_1, \mathbf{l}_2)} \\
&= \frac{\mathbb{P}(\mathbf{l}_1)\mathbb{P}(\mathbf{l}_2)}{\mathbb{P}(\mathbf{l}_1, \mathbf{l}_2)} \times \frac{\mathbb{P}(K = k|\mathbf{l}_1)\mathbb{P}(K = k|\mathbf{l}_2)}{\mathbb{P}(K = k)},
\end{aligned}$$

(via Bayes' theorem again, since $\mathbb{P}(\mathbf{l}_i|K = k) = \mathbb{P}(K = k|\mathbf{l}_i)\mathbb{P}(\mathbf{l}_i)/\mathbb{P}(K = k)$). Since $a = \mathbb{P}(\mathbf{l}_1)\mathbb{P}(\mathbf{l}_2)/\mathbb{P}(\mathbf{l}_1, \mathbf{l}_2)$ does not depend on the key hypothesis we can treat it as a normalisation constant which just needs to be computed so as to satisfy $\sum_{k \in \mathcal{K}} \mathbb{P}(K = k|\mathbf{l}_1, \mathbf{l}_2) = 1$. In the typical case that all keys are *a priori* equally likely, the denominator in the second product term is $\frac{1}{|\mathcal{K}|}$ (constant for all key hypotheses) and simply gets absorbed into the normalising constant. Thus, conditional probabilities on the key candidates can be updated with the introduction of any new, independent information via a simple multiplication-and-normalisation step.

3.3 Parallelised attack architecture

Combining multiple distinguishing vectors and attacking target functions involving 24 or more bits of the key are both computationally demanding tasks, and necessitate the use of parallelised computation. We elected to use the OpenCL

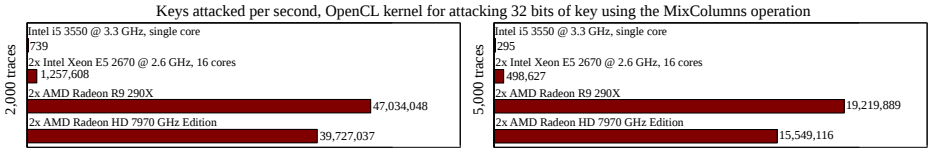


Fig. 1. Average keys per second recorded during DPA attacks on 32-bits of the input to the MixColumns operation for a variety of different sample sizes. Implementations are a 'naive' single-threaded CPU implementation, a parallelised OpenCL CPU-based implementation, and the two fastest OpenCL GPU implementations.

language and a set of graphics cards to parallelise the computation needed to attack up to 32-bits of a key, the combination and normalisation of distinguishing vectors, and finally the statistics necessary for evaluating the effectiveness of each combined attack.

We took inspiration from modern HPC facilities, in which a significant amount of the computing power is delivered by GPUs. Hence our experimental setup consists of several (up to 6) workstations, each containing two discrete GPUs (the cost per machine is approximately 2000 GBP). These were various pairs of high-end AMD and Nvidia cards, installed in our own workstations or within the Bluecrystal Phase 3 supercomputing facility[2]. In total, including all the functionality used to fully produce and analyse our experimental results, we were able to complete at least 2^{50} operations on combined distinguishing vectors, in very roughly a couple of weeks of computation time.

The most computationally demanding function was performing a 32-bit DPA attack on the MixColumns operation. Here we decided to share the cost over multiple GPUs, with each work group inside a single card computing a partial piece of the distinguishing vector using a portion of the traces and a subset of the key hypotheses, followed by a global reduction to compute the final vector. Fig. 1 shows the performance of our OpenCL attack implementation for a variety of devices, in terms of the number of key hypotheses tested per second.

We note that these benchmark timings are not likely to be optimal. We did not try to improve the memory coalescence of our kernels, nor did we try to perform any other non-trivial optimisation beyond maximising kernel occupancy, and so there may be *considerable* headroom in key-search throughput still to be gained. It is clear from the extremely cheap price for a dual GPU setup, coupled with the considerable performance increases observed with the introduction of new GPU architectures, that an adversary can acquire very large side-channel key-search capabilities at minimal financial cost.

Bartkewitz et al. [3] use Nvidia's CUDA technology and a Tesla C2070 to parallelise 8-bit CPA attacks on the SubBytes operation, and focus on maximising trace data throughput in an 8-bit setting. Our more ambitious goal is to

[2] Bluecrystal is managed by the Advanced Computing Research Centre at the University of Bristol—see `http://www.bris.ac.uk/acrc/`.

optimise for large key-search problems as well as for trace data throughput. In this context Moradi et al. [17] utilise 4 Nvidia Tesla GPUs to attack 32-bits of key using $60,000$ traces, and are able to attack a single time-point every 33 minutes. A direct comparison is not possible as we are using slightly more modern hardware and the exact computational costs included in the benchmarking are not clear—however we might expect to be able to perform a similar attack in approximately 20 minutes.

4 Experiments with simulated data

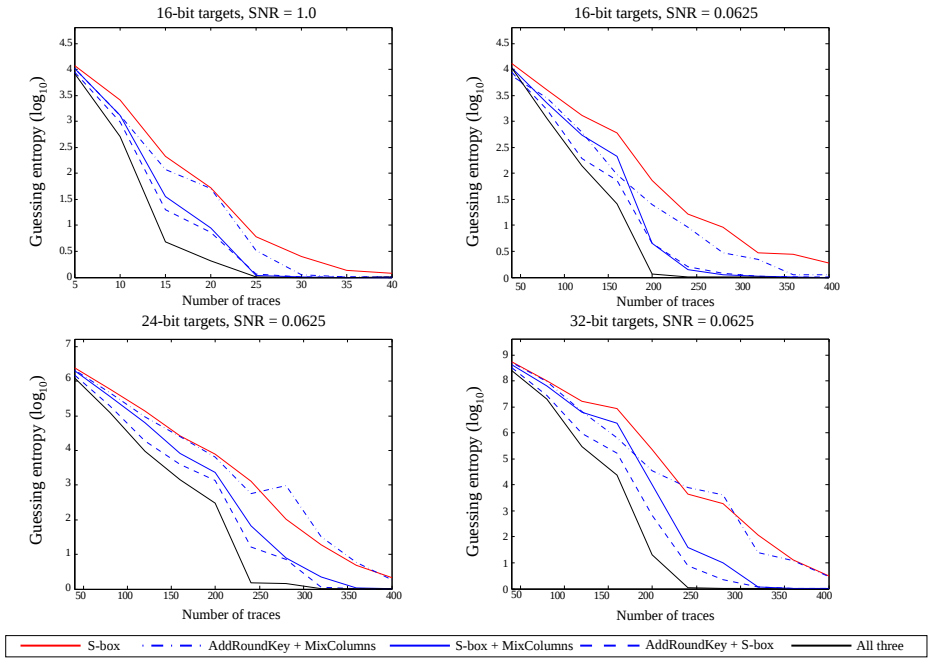
The goal of our combining strategy is to reduce (relative to 'standard univariate DPA') the guessing entropy on the subkeys (and consequently on the global key). Many types of combination are possible. We study the effect of combining outcomes from different targets as well as, secondarily, the effect of combining outcomes from different distinguishers applied to the same target. We do this initially for simulated trace measurements so that we can take into account different noise levels (i.e. by varying the SNR) as well as the impact of using an imperfect power model. Both aspects have practical relevance.

4.1 Combining outcomes from different targets

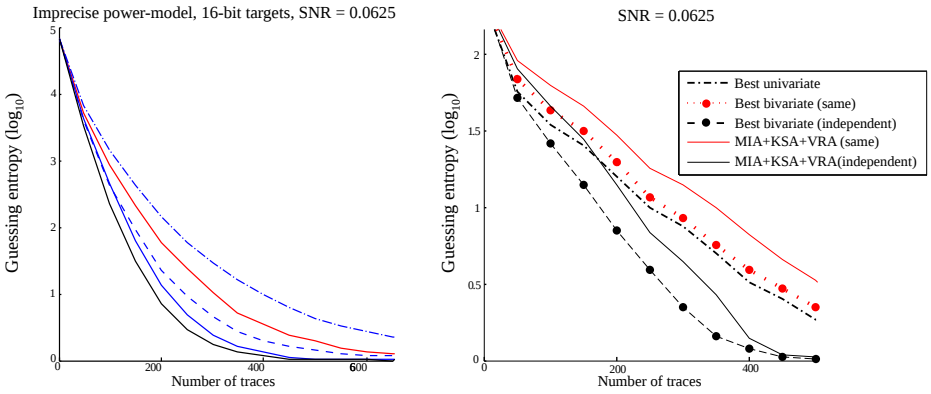
We simulated leakages of AES AddRoundKey, SubBytes, and three 8-bit interim values in the computation of MixColumns: one involving two key bytes (namely $GFm2(state_i \oplus state_{i+1})$ where $state_i$ is the i^{th} state byte after the SubBytes operation, and $GFm2$ denotes doubling in Rijndael's finite field), one involving three key bytes (namely $GFm2(state_i \oplus state_{i+1}) \oplus state_{i+1} \oplus state_{i+2}$), and one involving four key bytes (namely $GFm2(state_i \oplus state_{i+1}) \oplus state_{i+1} \oplus state_{i+2} \oplus state_{i+3}$) [3].

In the case of the 16-bit multi-target attack we necessarily hypothesise over two key bytes (in order to incorporate the MixColumns leakage). The experiments each involve two AddRoundKey correlation-based DPA attacks (which are then combined into probabilities on the full 16-bit subkeys via multiplication), two S-box attacks (combined likewise), and the one MixColumns attack, before multiplying each possible target function pair together, as well as multiplying all three together. Similarly, for the 24-bit multi-target attack we hypothesise over three key bytes. The experiments in the 24-bit attack then involve three AddRoundKey attacks, three S-box attacks, and the one attack on an interim MixColumns value. We amalgamate probabilities by multiplication as in the 16-bit case. The 32-bit multi-target attack proceeds in the same fashion: we combine four AddRoundkey attack results and four SubBytes results into the MixColumns attack result. The graphs in Fig. 2(a) show these different scenarios for a single column of the AES state.

[3] This targets a single intermediate byte. The relative effectiveness of combining all four attacks on all the possible intermediate bytes would also be interesting to investigate, but generating results requires time and so is left as future work.



(a) Outcomes for attacks combining several targets using up to 32-bit key hypotheses.



(b) Outcomes for attacks combining several distinguishers for the same target (the S-box output).

Fig. 2. Simulation results

In the following paragraphs we analyse these graphs with respect to three questions that are relevant for practice. Firstly, what is the impact of a (low) SNR with regards to our multi-target strategy? As we base our DPA attacks on correlation distinguishers, we would hope that, similarly to single-target attacks, multi-target attacks will 'scale' alongside the SNR. Secondly, we are interested in how the size of the key hypotheses impacts on the guessing entropy, and lastly, in how multi-target attacks behave when the attacker's power model is imprecise.

Impact of SNR. The top two graphs in Fig. 2(a) show the subkey guessing entropies (for a 16-bit key guess) as the number of traces increases, for the attacks against simulated Hamming weight leakages with two SNR levels. Aside from the fact that all attacks require increased numbers of traces as the SNR decreases (as we would expect) the scenarios exhibit similar outcomes. The attacks on S-boxes are effective at reducing uncertainty on the key (the results for these are printed in red), but are clearly outperformed by all three 'bivariate' combinations—even the one between the MixColumns sub-computation and AddRoundKey. The combination between all three further reduces the enumeration work required.

Impact of larger distinguishing vectors. The top right and the bottom graphs in Fig. 2(a) show the subkey guessing entropies for increasing subkey sizes (16-bit in the top right, 24-bit in the bottom left, and 32-bit in the bottom right). In all three experiments the multi-target attacks outperform the single target attacks. Note that the guessing entropy range naturally increases with the size of the key hypothesis and is in no way an indicator of attack degradation. For the 16-bit attack the guessing entropy is out of 2^{16} and eight such guesses need to be combined to get a global key with guessing entropy between 1 and 2^{128}. For the 32-bit attack the guessing entropy is out of 2^{32} but only four such guesses need to be combined. It is the global guessing entropy which ultimately matters and the subkeys always need to be combined at some point – incorporating information at (e.g.) the 32-bit level simply increases the scope of intermediate targets exploitable by the attacker. For both hypothesis sizes, the outcomes suggest that we are able to succeed with roughly half the number of leakage traces when using the best multi-target attack (for a fixed subkey guessing entropy, the best multi-target attacks require roughly half of the traces required by the best single-target attack). It is possible to estimate global key guessing entropies based on these results by assuming that the attacks on the other 'chunks' of the key would behave identically. For instance, in the 16-bit case, if all eight 16-bit attacks give identical outcomes, we could estimate global key entropies by raising the results of a single 16-bit attack to the power eight. However, this does not necessarily translate into practice, so we will instead show actual global key guessing entropies when we come to discuss attacks on real data.

Impact of imperfect power model. The left picture of Fig. 2(b) shows the outcomes (against a 16-bit subkey target: the legend from Fig. 2(a) applies) in the case where the Hamming weight is *not* a perfect match to the leakage, because

of the presence of a constant reference state (representing an address, for example) of Hamming weight 1. The most striking impact of this distortion occurs for attacks that include AddRoundKey as a target, which are no longer able to identify the correct key as a likely candidate. This is because the Hamming distance of the AddRoundKey from the reference state when the correct key is guessed is the same as the Hamming weight of the AddRoundKey when the key guess is the correct key XORed with the reference state. In effect, an incorrect key is masquerading as the correct one, and the correlation DPA against AddRoundKey will naturally preference this. (The same cannot happen for the S-box, for example, because the key XOR is *inside* the highly nonlinear transformation, with the Hamming distance being taken *afterwards*).

Nonetheless, in this case where the reference state is itself of low weight, incorporating AddRoundKey information still produces marginal reductions on the guessing entropies after S-box and MixColumns (separately, and combined). Greater imprecision of the power model will more strongly impact on AddRoundKey attacks; it may be advisable to exclude it as a target in such cases.

4.2 Combining outputs from the same target

One might ask whether or not the outcomes of different *distinguishing statistics* or *power models* can likewise be combined to some advantage.

Using different distinguishing statistics. Suppose we run three different attacks against the leakage of an AES S-box, e.g.: mutual information [11], Kolmogorov–Smirnov [26], and the variance ratio [22], all using a Hamming weight power model. The distinguishing vectors are transformed to have a baseline of zero and to sum to one, for use as heuristic 'probability' scores. We would then like to know whether the combined outcomes improve upon the individual ones.

The right picture in Fig. 2(b) shows what happens when we attempt this in the example scenario of Hamming weight leakage with SNR 0.0625. When the same measurements are used for all of the attacks, combining the outcomes actually *increases* the guessing entropy. By contrast, when independent measurements are used in each case (i.e., each distinguisher has been applied against a different point in the trace leaking the same information but with independent noise), there is some scope to refine the information on the key by combining outcomes—although all three outcomes together on average produce worse results than the best combination of two. We found that it was generally the addition of mutual information which degraded the outcome, as it required substantially more data to estimate to an equivalent degree of precision.

This is very much in line with what we might expect, and acts as a noteworthy warning: it is the addition of *new information* which improves attack outcomes—exploiting the same measurements using the same power models but with different distinguishers does *not* contribute anything further. In the context of our heuristic 'probability' distributions such a practice could be particularly dangerous, as it still serves to exaggerate the magnitude of the peaks, thus giving a false sense of increased certainty. Note that the multiplication step implicitly

assumes *independence* of the separate score vectors, which is clearly violated in the case that they are all based on the same leakage information.

Using different power models. In the light of the ineffectiveness of combining information about the same target, we briefly revisit previous work by Bevan and Knudsen [5]. They suggest to combine eight difference-of-means attacks, each targeting a distinct bit of the intermediate value, by 'summing over the distinguisher results' (in our approach we convert them into 'probability' distributions on the set of 2^8 subkeys, as per Section 3.1). Since each attack exploits a *separated portion* of the overall leaked value we may expect that each new bit attacked helps to further reduce the candidate search space—and, indeed, our experiments confirm this (see Appendix A). Such a technique is hence very useful in leakage scenarios which are unfamiliar to an attacker, which is often the case when attacking dedicated hardware.

5 Practical attacks

We tested our strategy in practice using a dataset of 10,000 traces from an ARM7 microcontroller running an unprotected implementation of AES. The 10,000 traces were divided up in 200 sets of 50 traces each to conduct sufficient repeat experiments to report reasonably precise estimates for the guessing entropies in the same vein as our simulated attacks. Multi-target attacks, similar to multivariate attacks, are greatly helped by knowledge about where the attacked intermediate values leak in the traces. Consider for instance a (multivariate) template attack: it is much harder for an adversary to conduct such an attack when in the profiling phase a similar device is available but not the exact implementation (of, say, AES). In such a case an adversary could still build templates for microprocessor instructions during profiling, but in the attack phase the adversary would need to find the specific trace points at which to apply the templates. Similarly, knowing precisely where the single-target leakages occur is helpful for a multi-target attack. We consequently focus initially on a 'known point' scenario and then make a first attempt at relaxing this assumption.

5.1 Practical attacks against known interesting points

We applied two multi-target attacks (one involving 16-bit, one 32-bit key hypotheses) under the assumption that interesting trace points are known, running 200 repeat experiments for increasing samples of up to size 50. For each 16-bit subkey, correlation DPA attacks were performed against the two corresponding AddRoundKey operations, the S-boxes and the MixColumns sub-computation $GFm2(state_i \oplus state_{i+1})$, where $state_i$ is the state byte corresponding to the i^{th} key byte after the S-box substitutions and (in this implementation) the ShiftRows operation. For each 32-bit subkey, correlation DPA attacks are performed against four AddRoundKey operations, four S-boxes, and the 32-bit MixColumns computation $GFm2(state_i \oplus state_{i+1}) \oplus state_{i+1} \oplus state_{i+2} \oplus state_{i+3}$.

The first two graphs in Fig. 3(a) show the guessing entropies on the first key-byte pair and the final global guessing entropies, estimated by multiplying the eight subkey guessing entropies together (the outcomes for the other seven subkey guesses can be found in Appendix B—see Fig. 5).[4] They largely, but not perfectly, match up with our observations for simulated traces. This is an important point: theory and practice rarely perfectly align, even in the case of a relatively 'simple' platform like the ARM7. In the practical experiments, AddRoundKey and the MixColumns sub-value are consistently unable to identify the correct key alone (at least, not within 50 traces). However, the two together produce guessing entropies to rival the effectiveness of the S-box attack, and both produce improvements in combination with the S-box. All three together produce the best guessing entropies for many of the 16-bit subkeys, although they are sometimes outperformed by the two-target S-box+AddRoundKey attacks, which achieve a marginal advantage overall (see Table 1 in Appendix B).

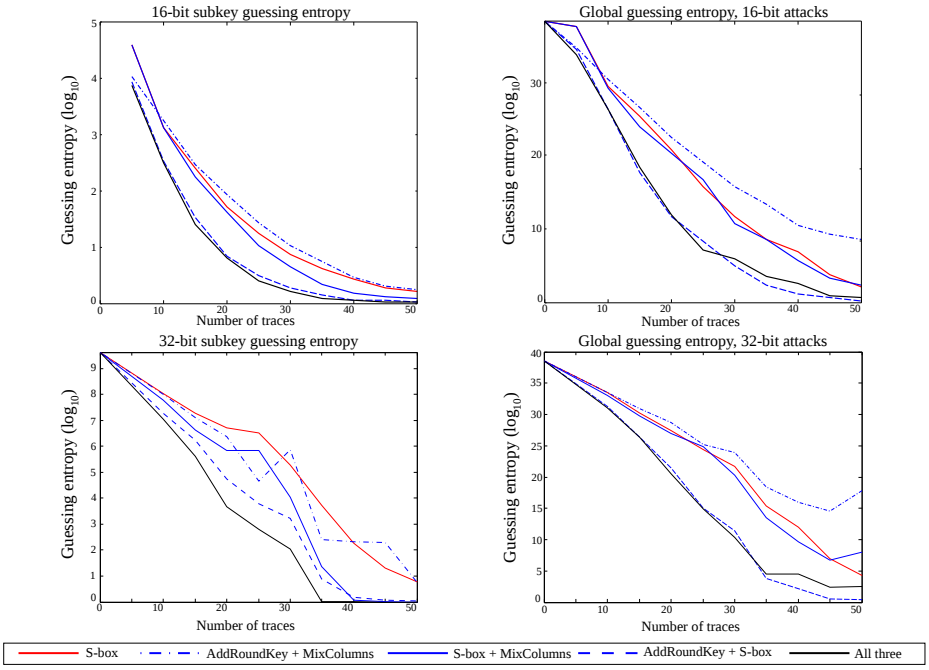
The second two graphs in Fig. 3(a) show the guessing entropies for the first 32-bit subkey and the final global guessing entropies. The global entropies were estimated by multiplying the four subkey entropies together. We observed varying behaviour for our combined attacks on different subkeys; our targeted MixColumns computation does not leak nearly as much information in the middle 8 bytes of the state as it does in the first and final 4 bytes. Consequently, despite (as suggested by our simulated attacks) observing strong performance of the combined three-target attacks in the latter two cases, in the global setting this advantage is diminished, and the 'trivariate' attack produces similar performance to the combined four-byte S-box+AddRoundKey attacks. It is noteworthy that even in the presence of this variable leakage, most combined attacks outperform the S-box attack. Graphs and data for each of the four separate subkey attacks can be found in Appendix B (see Fig. 6 and Table 2).

5.2 Practical attacks where interesting points are *a priori* unknown

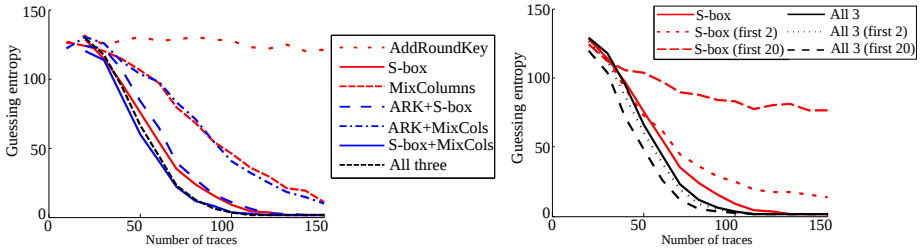
The natural next question to ask is whether we can relax the assumption that the leakage points are precisely known. We made some preliminary inroads using 'desktop-level' resources (whilst our GPU machines were occupied with other experiments), focusing, for computational feasibility, on 8-bit key hypotheses. The three targets we selected to combine were AddRoundKey, the S-box outputs, and the interim MixColumns value $GFm2(state_i \oplus state_{i+1})$ with the assumption that the second involved key byte of the two is known.

We relaxed the 'known point' assumption by visually inspecting the AES traces in order to identify the intervals in which each of the three target functions are contained. The first round takes about 1,400 clock cycles in total and the (non-overlapping) windows we selected for experimentation were of widths 240, 230, and 180 for AddRoundKey, SubBytes and MixColumns respectively. Within

[4] The more refined rank estimation methodology of [25] indicates that this simple method of approximating global guessing entropies underestimates the rank by 20 to 40 binary orders of magnitude.



(a) Outcomes for multi-target attacks in a known points scenario.



(b) Outcomes for multi-target attacks in a known interval scenario.

Fig. 3. Practical results

these windows we took an 'exhaustive search' approach. First, we subjected each point to a standard DPA attack against the associated target function, and computed the 'probability' scores. We then pairwise combined them in each of the three possible configurations, and finally we combined all three. We tried two strategies: in the first, we took (for each configuration) the combined vector with the largest peak as the one most likely to correspond to the correct key and pair/triple of leakage points, and in the second we took the N_t combined vectors with the largest peaks and multiplied these together (for different values of N_t), so achieving a sort of 'majority vote'.

The left side of Fig. 3(b) shows the average guessing entropy for each of the attacks using the first 'maximum peak' strategy. The AddRoundKey attack in an unknown point scenario performs very badly. Further analysis of the trace window reveals that there are other points exhibiting strong correlations with AddRoundKey $\oplus R$, for R some other (possibly address?) value in $\{0, 255\}$ (see Fig. 7 in Appendix C).[5] Moreover, at these points the *correct* key correlations are *low*, so that the contribution to the combined leakage is highly distorting (as opposed to when an 'imperfect but close' leakage prediction is made, in which case the combination can still improve distinguishability). In the presence of such misleading leakage information, it is reassuring that the attack outcomes are *robust* to the combining step.

The combined MixColumns and S-box attack exhibits lower guessing entropies than either of the two taken individually. The trivariate attack (as expected from the above) does not really add much to this, but again we reflect that the inclusion of AddRoundKey at least does not seem to harm the outcome.

The right side of Fig. 3(b) shows the advantage gained by multiplying the top-ranked few 'probability' vectors for the trivariate attack, as well as (for comparison) for the S-box attack on its own. Interestingly, even the addition of the second ranked vector degrades the S-box attack, whereas the product combining for the top-ranked triples reduces the guessing entropy at least up to $N_t = 20$. The subsequent total improvement over the S-box outcome on its own indicates this as a potentially worthwhile strategy for key recovery in an unknown point scenario.

From a practical perspective, a useful forward approach for multi-target attacks would be to 'try out' (for a concrete device and implementation) different combinations of targets, and different point selection strategies, to see which give the best results. We want to caution against drawing too many conclusions from these last experiments: they clearly represent a first step only!

6 Conclusion

We have shown how to amalgamate single-target 'standard' DPA attacks (using a correlation distinguisher and a Hamming weight power model) into multi-

[5] Note that the leakage of the S-box is less vulnerable to such distortions: a non-zero reference state will not masquerade as an alternative key hypothesis, as the key addition happens *inside* the S-box.

target attacks capable of increasing information on the correct key by combining DPA outcomes that are treated as heuristic probabilities. Leveraging our modern HPC-inspired computing platform, we are able to efficiently handle key hypotheses of up to 32 bits using a small cluster of simple workstations containing consumer graphics cards. Such a capability allows us to combine many intermediate targets; in this work we made the first serious attempt to explore the characteristics of successful combinations. Our results indicate that combining S-box+AddRoundKey or additionally including an intermediate MixColumns computation typically produces the strongest results. Multi-target attacks scale predictably with noise and are robust with regards to imprecise power models. Our primary investigative effort is mainly on 'known' (leakage) point attacks, in line with assumptions generally made for multivariate attacks. When leakage points are not known, an exhaustive search in suitable visually-identified trace windows, together with a 'majority vote'-style approach to decide on 'peaks', leads to improved practical attacks even in this challenging scenario.

Our definition of multi-target attacks and intuitive and efficient combination technique opens up many interesting new research questions: e.g. is there any single best combination of intermediate values for a given cipher? How effectively can we combine power and EM attack results in this way? Could we even move further on and include results from the second encryption round? What other strategies for combining in unknown point scenarios exist? How could we use this against implementations when masking and hiding are used? For better or worse, these are "interesting times"—to call to mind the fabled Chinese curse.

Acknowledgements. This work has been supported in part by EPSRC via grant EP/I005226/1. This work was carried out using the computational facilities of the Advanced Computing Research Centre, University of Bristol—`http://www.bris.ac.uk/acrc/`.

References

1. C. Archambeau, E. Peeters, F.-X. Standaert, and J.-J. Quisquater. Template Attacks in Principal Subspaces. In L. Goubin and M. Matsui, editors, *CHES 2006*, volume 4249 of *LNCS*, pages 1–14. Springer, 2006.
2. C. Bailer-Jones and K. Smith. Combining probabilities. Technical Report GAIA-C8-TN-MPIA-CBJ-053, Max Planck Institute for Astronomy, Heidelberg, January 2010.
3. T. Bartkewitz and K. Lemke-Rust. A high-performance implementation of differential power analysis on graphics cards. In *CARDIS*, volume 7079 of *LNCS*, pages 252–265. Springer, 2011.
4. L. Batina, J. Hogenboom, and J. van Woudenberg. Getting More from PCA: First Results of Using Principal Component Analysis for Extensive Power Analysis. In O. Dunkelman, editor, *Topics in Cryptology – CT-RSA '12*, volume 7178 of *LNCS*, pages 383–397. Springer Berlin / Heidelberg, 2012.
5. R. Bevan and E. Knudsen. Ways to Enhance Differential Power Analysis. In P. J. Lee and C. H. Lim, editors, *ICISC*, volume 2587 of *LNCS*, pages 327–342. Springer, 2002.

6. E. Brier, C. Clavier, and F. Olivier. Correlation Power Analysis with a Leakage Model. In M. Joye and J.-J. Quisquater, editors, *CHES 2004*, volume 3156 of *LNCS*, pages 135–152. Springer Berlin / Heidelberg, 2004.
7. S. Chari, J. Rao, and P. Rohatgi. Template Attacks. In B. Kaliski, Ç. Koç, and C. Paar, editors, *CHES 2002*, volume 2523 of *LNCS*, pages 51–62. Springer Berlin / Heidelberg, 2003.
8. Common Criteria, Techical editor: BSI. Application of Attack Potential to Smart Cards. http://www.commoncriteriaportal.org/files/supdocs/CCDB-2009-03-001.pdf, 2009.
9. J. Doget, E. Prouff, M. Rivain, and F.-X. Standaert. Univariate side channel attacks and leakage modeling. *J. Cryptographic Engineering*, 1(2):123–144, 2011.
10. M. Elaabid, O. Meynard, S. Guilley, and J.-L. Danger. Combined Side-Channel Attacks. In Y. Chung and M. Yung, editors, *Information Security Applications*, volume 6513 of *LNCS*, pages 175–190. Springer Berlin Heidelberg, 2011.
11. B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel. Mutual Information Analysis: A Generic Side-Channel Distinguisher. In E. Oswald and P. Rohatgi, editors, *CHES 2008*, volume 5154 of *LNCS*, pages 426–442. Springer-Verlag Berlin, 2008.
12. S. Hajra and D. Mukhopadhyay. SNR to Success Rate: Reaching the Limit of Non-Profiling DPA. Cryptology ePrint Archive, Report 2013/865, 2013. `http://eprint.iacr.org/`.
13. M. Hutter, M. Kirschbaum, T. Plos, J.-M. Schmidt, and S. Mangard. Exploiting the Difference of Side-Channel Leakages. In W. Schindler and S. A. Huss, editors, *COSADE*, volume 7275 of *LNCS*, pages 1–16. Springer, 2012.
14. P. C. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *Proceedings of CRYPTO 1999*, pages 388–397, London, UK, 1999. Springer-Verlag.
15. S. Mangard, E. Oswald, and T. Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*.
16. S. Mangard, E. Oswald, and F.-X. Standaert. One for All – All for One: Unifying Standard DPA Attacks. *IET Information Security*, 5(2):100–110, 2011.
17. A. Moradi, M. Kasper, and C. Paar. Black-Box Side-Channel Attacks Highlight the Importance of Countermeasures - An Analysis of the Xilinx Virtex-4 and Virtex-5 Bitstream Encryption Mechanism. In O. Dunkelman, editor, *CT-RSA*, volume 7178 of *LNCS*, pages 1–18. Springer, 2012.
18. M. Renauld and F.-X. Standaert. Combining Algebraic and Side-Channel Cryptanalysis against Block Ciphers. In *30^{th} Symposium on Information Theory in the Benelux*, 2009.
19. M. Renauld, F.-X. Standaert, and N. Veyrat-Charvillon. Algebraic Side-Channel Attacks on the AES: Why Time also Matters in DPA. In *Workshop on Cryptographic Hardware and Embedded Systems (CHES 2009)*, volume 5747 of *Lecture Notes in Computer Science*, pages 97–111. Springer, 2009.
20. Y. Souissi, S. Bhasin, S. Guilley, M. Nassar, and J.-L. Danger. Towards Different Flavors of Combined Side Channel Attacks. In O. Dunkelman, editor, *CT-RSA*, volume 7178 of *LNCS*, pages 245–259. Springer, 2012.
21. F.-X. Standaert and C. Archambeau. Using Subspace-Based Template Attacks to Compare and Combine Power and Electromagnetic Information Leakages. In E. Oswald and P. Rohatgi, editors, *CHES 2008*, volume 5154 of *LNCS*, pages 411–425. Springer Berlin Heidelberg, 2008.
22. F.-X. Standaert, B. Gierlichs, and I. Verbauwhede. Partition vs. Comparison Side-Channel Distinguishers: An Empirical Evaluation of Statistical Tests for Univariate Side-Channel Attacks against Two Unprotected CMOS Devices. In P. Lee and

J. Cheon, editors, *ICISC 2008*, volume 5461 of *LNCS*, pages 253–267. Springer Berlin / Heidelberg, 2009.
23. F.-X. Standaert, T. G. Malkin, and M. Yung. A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In *EUROCRYPT '09*, pages 443–461, Berlin, Heidelberg, 2009. Springer-Verlag.
24. N. Veyrat-Charvillon, B. Gérard, M. Renauld, and F.-X. Standaert. An Optimal Key Enumeration Algorithm and Its Application to Side-Channel Attacks. In L. R. Knudsen and H. Wu, editors, *Selected Areas in Cryptography*, volume 7707 of *LNCS*, pages 390–406. Springer, 2012.
25. N. Veyrat-Charvillon, B. Gérard, and F.-X. Standaert. Security Evaluations beyond Computing Power. In T. Johansson and P. Q. Nguyen, editors, *EUROCRYPT*, volume 7881 of *LNCS*, pages 126–141. Springer, 2013.
26. N. Veyrat-Charvillon and F.-X. Standaert. Mutual Information Analysis: How, When and Why? In C. Clavier and K. Gaj, editors, *Proceedings of CHES 2009*, volume 5747 of *LNCS*, pages 429–443. Springer Berlin / Heidelberg, 2009.
27. C. Whitnall and E. Oswald. A Comprehensive Evaluation of Mutual Information Analysis Using a Fair Evaluation Framework. In P. Rogaway, editor, *CRYPTO*, volume 6841 of *LNCS*, pages 316–334. Springer, 2011.

A Combining difference-of-means outcomes

Fig. 4 shows the reduction in subkey guessing entropy as an increasing number of difference-of-means (against different individual bits) are combined via our strategy.

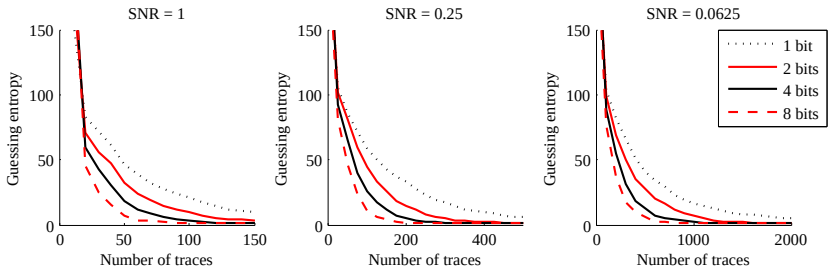


Fig. 4. Combining the outcomes of up to eight difference-of-means attacks against Hamming weight leakage of the AES S-box.

B Practical attacks against known leakage points

Fig. 5 shows all eight subkey attacks against the ARM7 data, combining up to three 16-bit target functions.

Fig. 6 shows all four subkey attacks against the ARM7 data, combining up to three 32-bit target functions.

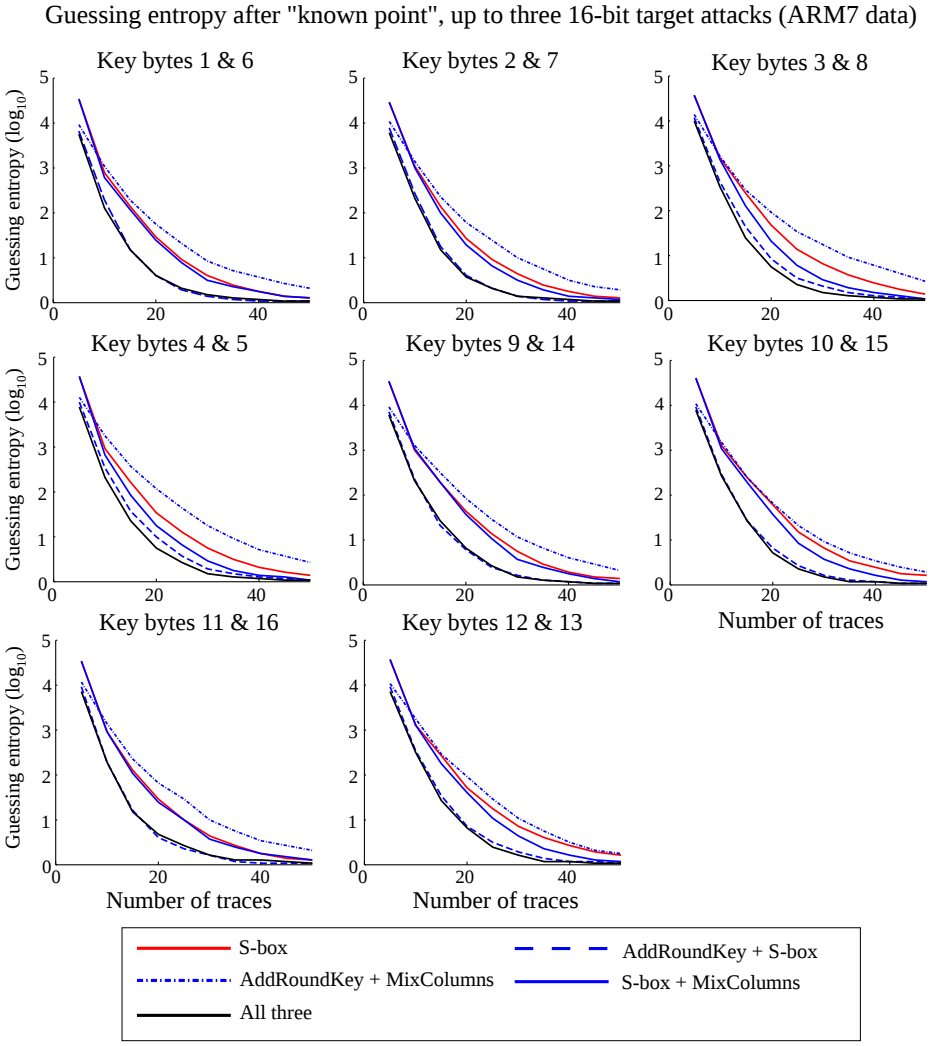


Fig. 5. Guessing entropies for key byte pairs using up to three targets, where the interesting points are known.

C Unknown point attacks: problem of rival peaks

Fig. 7 illustrates the difficulty of separating the true key from strong rival candidates when the relevant 'interesting points' in the trace are not known. As described in Section 5, this introduces distorting information into the point search, which reduces the ability to increase an attack's effectiveness by the addition of AddRoundKey outcomes.

Table 1. Global key guessing entropies after DPA against up to three 16-bit intermediate targets (ARM7 data).

	Number of traces				
	10	20	30	40	50
AddRoundKey	7.1e+30	7.3e+21	1e+18	2.8e+14	1.4e+11
S-boxes	3.5e+29	5.9e+20	5.9e+11	6.1e+6	1e+02
MixColumns	2.3e+35	8.2e+31	1.4e+29	3.4e+26	5.7e+24
ARK+S-boxes	2.1e+26	5e+11	8.3e+04	14	1.7
ARK+MixCols	2.7e+30	3.1e+22	6.2e+15	3.2e+10	4.1e+08
S-boxes+MixCols	1.5e+29	2.5e+20	5e+10	5.6e+05	1.5e+02
All three	2.4e+26	7.2e+11	6.9e+05	2.6e+02	4

Table 2. Global key guessing entropies after DPA against up to three 32-bit intermediate targets (ARM7 data).

	Number of traces				
	10	20	30	40	50
AddRoundKey	5.4e+33	1e+29	1.6e+24	1.8e+17	1.5e+14
S-boxes	3.5e+33	2.7e+27	5.4e+21	8.7e+11	1.9e+04
MixColumns	1e+35	1.2e+34	9.8e+31	1.2e+30	6.1e+21
ARK+S-boxes	2.1e+31	2.8e+21	2.7e+11	1.4e+02	2.4
ARK+MixCols	2.9e+33	5.2e+28	8.5e+23	1e+16	6.9e+17
S-boxes+MixCols	1e+33	8.3e+26	1.8e+20	4.5e+09	9e+07
All three	1.1e+31	3.5e+20	2.4e+10	3.1+e04	3e+02

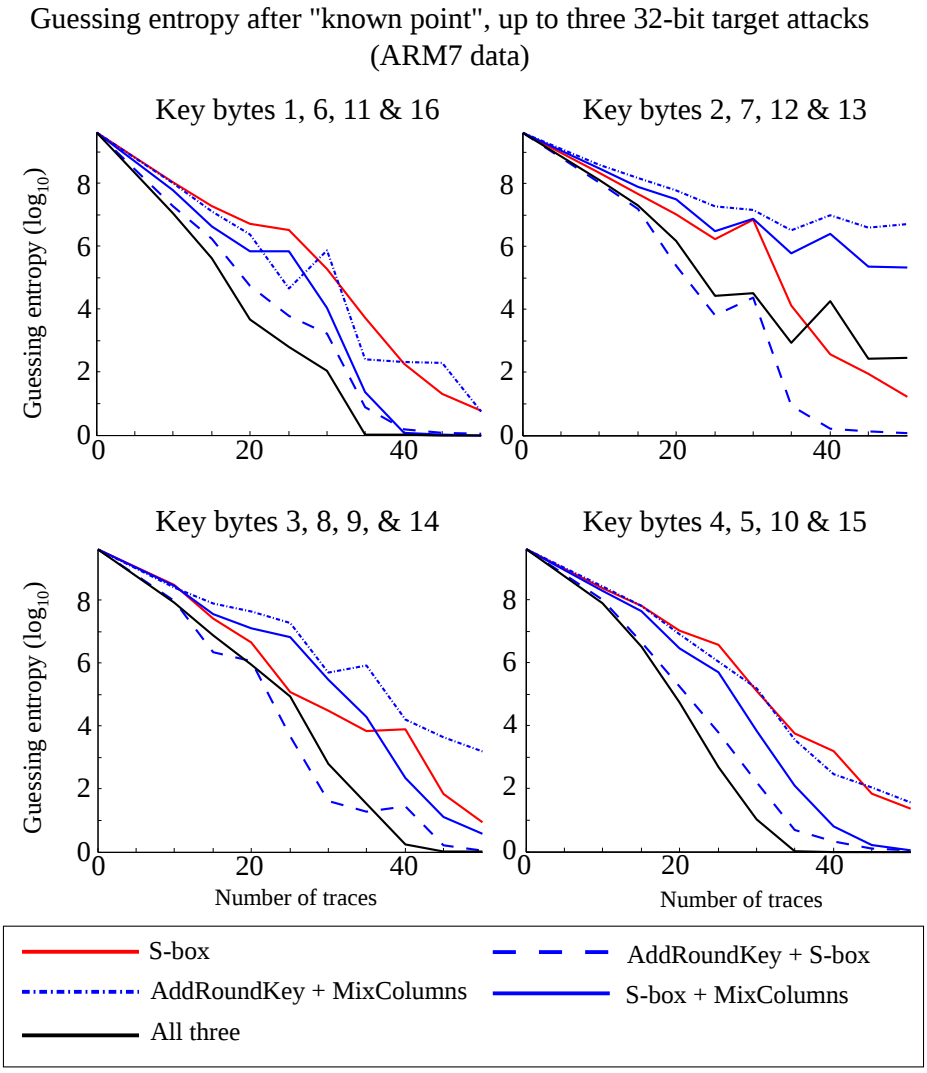


Fig. 6. Guessing entropies for chunks of four key bytes, using up to three targets, where the interesting points are known.

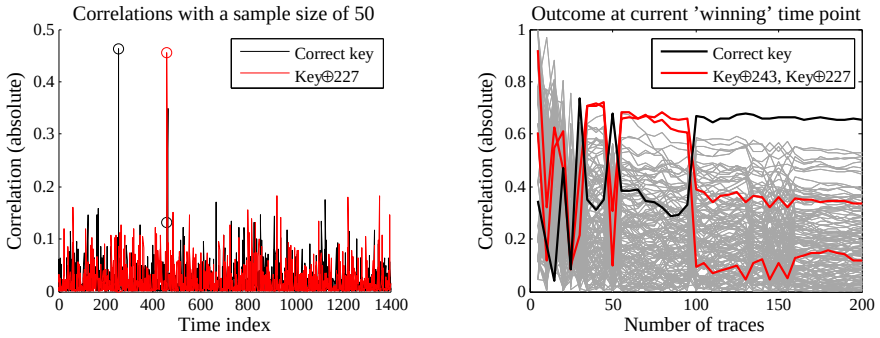


Fig. 7. Left: Example of a fixed XOR offset from the key producing a rival peak in the AddRoundKey correlation attack against the ARM7 traces. Right: The evolution of an AddRoundKey correlation attack against the ARM7 traces, showing the confounding effect of strong rival candidates.