# Characterisation and Estimation of the Key Rank Distribution in the Context of Side Channel Evaluations

Daniel P. Martin, Luke Mather, Elisabeth Oswald, and Martijn Stam

University of Bristol, Department of Computer Science,
Merchant Venturers Building, Woodland Road, BS8 1UB, Bristol, UK.
{dan.martin, luke.mather, elisabeth.oswald, martijn.stam}@bris.ac.uk

**Abstract.** Quantifying the side channel security of implementations has been a significant research question for several years in academia but also among real world side channel practitioners. As part of security evaluations, efficient key rank estimation algorithms were devised, which in contrast to analyses based on subkey recovery, give a holistic picture of the security level after a side channel attack. However, it has been observed that outcomes of rank estimations show a huge spread in precisely the range of key ranks where enumeration could lead to key recovery. These observations raise the question whether this is because of insufficient rank estimation procedures, or, if this is an inherent property of the key rank. Furthermore, if this was inherent, how could key rank outcomes be translated into practically meaningful figures, suitable to analysing the risk that real world side channel attacks pose? This paper is a direct response to these questions. We experimentally identify the key rank distribution and show that it is independent of different distinguishers and signal-to-noise ratios. Then we offer a theoretical explanation for the observed key rank distribution and determine how many samples thereof are required for a robust estimation of some key parameters. We discuss how this can be naturally integrated into real world side channel evaluation practices. We conclude our research by connecting non-parametric order statistics, in particular percentiles, in a practically meaningful way with business goals.

## 1 Introduction

To assess the outcome of an attack, researchers traditionally sought to determine the attack's success rate (SR). Standaert et al. [20] provided a formal definition for the SR and hypothesised that there is a link between attack outcomes (the success rate, assuming a single targeted intermediate value) and the leakage (measured in information theoretic terms in the same intermediate value). Further research aimed at characterising the SR, e.g. [18,21], or finding alternative ways to predict differential power analysis (DPA) outcomes, e.g. [8]. These contributions brought much needed clarity about some aspects of the (interactions) between target functions and leakage models, but (necessarily) had to restrict themselves to considering attack outcomes for *a single subkey* only.

In practice however, the effort to reveal the *entire* secret key is the concern of most primacy: given a number of traces, and a computational budget for key enumeration, what is the likelihood to reveal the secret key? This question can be answered both by a generalised SR (which is closely connected to the key guessing entropy (GE, see [20]), this line of research has recently been developed further by Duc et al. [6].) or by computing the *rank* of the secret key. Consequently, fast methods to compute the rank of the secret key have become a hot topic [2,3,10,14,25,24].

It is noteworthy that the first computationally efficient and accurate key estimation algorithm originated from an evaluation lab [10]. Their interest in the topic explains itself easily: assuming a sufficiently accurate method to estimate the true rank of the secret key, decisive leakage evaluations could be performed. However, the existing research brought to light an (unexpected) difficulty along the way: even though the aforementioned previous works sought to minimise the estimation error in key rank algorithms, the derived key ranks show a huge spread in exactly the range of ranks where enumeration is of practical importance. This opens up the question whether these ranks actually give meaningful information? And if so how would key rank computations be integrated in standardised security evaluations? The potential implication of these recent research results have prompted JHAS (JIL Hardware-related Attacks Subgroup, this industry led group essentially defines Common Criteria security evaluation practises for smart card products) to set up a specific working group that deals with the topic.

Our research offers answers to these questions: after introducing some background (Sect.2) we improve the key rank algorithm of Martin et al. [14] to produce the (to date) most precise key ranking algorithm (Sect. 3). Using this high-precision ranking algorithm, we focus on the properties of the key rank distribution: we begin by an experimental exploration of the key rank, which we accompany and strengthen by a theoretical analysis.Then, drawing from carefully designed simulations, we justify some general observations about the key rank such as the independence of side channel distinguisher and trace characteristics. We evaluate statistical metrics for the purpose of quantifying the risk from side-channel attacks through an "evaluation through rank estimation" approach and relate it to (potential) business goals.

## 2 Side-channel evaluations and key rank

This section covers some basic notation related to differential power analysis (DPA) style attacks on modern blockciphers, as well as surveying the recent works on computing fast and accurate estimates for the key rank.

We use a bold type face to denote multi-dimensional variables. A key $\mathbf{k}$ can be partitioned into $m$ subkeys, which we denote as $\mathbf{k} = (k^0, \dots, k^{m-1})$. We assume that all subkeys are of the same size (which holds in most scenarios in practice) and that each subkey can take one of $n$ possible values. As an example, for AES-128 typically the 128-bit key is subdivided in $m = 16$ subkeys of a byte

($n = 256$) each. The key to be recovered by the DPA attack is called the secret key and is denoted $\mathbf{sk} = (sk^0, \ldots, sk^{m-1})$.

### 2.1 Standard DPA model

In this paper we consider a standard DPA scenario as in Mangard et al. [13], which implies the attacks are single order and univariate. (Note that in higher order attacks the univariate targets still fit a standard DPA attack). An attacker has $N$ power measurements or traces $T_i$ corresponding to encryptions of $N$ known plaintexts $x_i \in \mathcal{X}$, $i = 1, \ldots, N$ and wishes to recover the secret key $\mathbf{sk}$.

For each subkey (so $j = 0, \ldots, m-1$) we assume that each trace $T_i$ is condensed to a single point of interest $P_{i,j}$ and that this value $P_{i,j}$ decomposes additively as $P_j = P_{\text{exp}} + P_{\text{noise}}$. Here $P_{\text{exp}}$, called the *signal*, is a deterministic function of the value of the subkey $sk^j$ and the relevant input $x_i$, whereas $P_{\text{noise}}$, called the *noise*, is drawn at random according to some distribution that does not depend on any of the input values (including the secret key $\mathbf{sk}$). The signal-to-noise ratio (SNR) is then defined as the ratio of the variance in the signal (when ranging over secret keys and plaintexts) divided by the variance in the noise:[1]

$$\text{SNR} = \frac{Var(P_{\text{exp}})}{Var(P_{\text{noise}})}.$$

The SNR is used to quantify the amount of leakage within a given measurement: the higher the SNR, the more information within the trace that can be exploited.

A distinguisher $D^j$ against the $j^{\text{th}}$ subkey takes as input the vector of condensed traces and corresponding plaintexts $(P_{i,j}, x_i)_{i=1,\ldots,N}$ and outputs a distinguishing vector $\mathbf{D}^j \in \mathbb{R}^n$, which assigns a score for each possible hypothesis of the subkey under consideration. Without loss of generality, we will assume that the higher the score for a subkey hypothesis $k^j$, the more likely the distinguisher deems that secret key equals $k^j$. The distinguisher $D$ on the complete key, simply runs the subkey distinguishers $D^j$ for each subkey and outputs a list of distinguishing vectors $\mathbf{D} \in \mathbb{R}^{n \times m}$ (namely a distinguishing vector for each subkey).

### 2.2 Key rank

The result of a side channel attack is a set of distinguishing vectors, which hold the information about subkeys (when studied individually) and the entire key (when studied jointly). To judge the potency of an attack, we need suitable metrics to express how well the distinguishing vectors enable key recovery.

Even though the ultimate goal is full key recovery, historically the emphasis has been on subkey recovery. The only relevant information in a subkey distinguishing vector $\mathbf{D}^j$ is the order it induces on possible subkey hypothesis, as a clever adversary would test the subkeys in order of likelihood (ignoring for a

[1] Strictly speaking the SNR is defined relative to a subkey and should be indexed by $j$; however when we later refer to the SNR it will be the same for all subkeys.

moment how one would test an individual subkey). The only information needed to identify the true subkey $sk^j$ in this ordering is its distinguishing score $d_{sk^j,j}$, leading to the following definition of subkey rank.

**Definition 1 (Subkey rank).** *Given the distinguishing vector* $\mathbf{D}^j$*, and the distinguishing score* $d_{sk^j,j}$ *for subkey* $sk^j$*, count the number of subkeys with score strictly larger than* $d_{sk^j,j}$*. We denote this* $\mathrm{rank}^j_{sk^j}(\mathbf{D}^j)$*.*

Extending subkey rank to a full key is based on the assumption that the distinguishing scores for individual subkeys can be added to give a meaningful score for the full score. For instance, given the distinguishing table $\mathbf{D} = (\mathbf{D}^0, \ldots, \mathbf{D}^{m-1})$ for the entire key, the score of secret key $\mathbf{sk}$ is computed as $W = \sum_{j=0}^{m-1} d_{sk^j,j}$ (where the notation $d_{sk^j,j}$ identifies the score corresponding to $sk^j$ in the distinguishing vector $\mathbf{D}^j$). In this case the actual values in the (subkey) distinguishing vectors becomes relevant.

**Definition 2 (Key rank).** *Given the distinguishing table* $\mathbf{D}$*, and the score* $W$ *of the secret key* $\mathbf{sk}$*, count the number of keys with score strictly larger than* $W$*. This is denoted* $\mathrm{rank}_{\mathbf{sk}}(\mathbf{D})$*.*

*Remark 1.* If multiple keys have the same scores as the secret key, we assume that the latter is ranked first. This gives a conservative rank for a given distinguishing vector, as it will be the earliest an adversary would enumerate the key. For distinguishers that don't actually distinguish that well (e.g. because they do not exploit any leakage) this can lead to key ranks that significantly underestimate the remaining effort to recover the full key.

The key rank $\mathrm{rank}_{\mathbf{sk}}(\mathbf{D})$ of a single secret key given a specific distinguishing table is not particularly interesting on its own. To say something meaningful, we will consider the key rank as a random variable that is the outcome of the experiment in Fig. 1. Here a random key, random plaintexts, and (implicitly) random noise in the measurements $T_i$ are chosen, as a result of which the output of the experiment is a random variable. We will denote this random variable $\mathrm{keyrank}_D(N)$, which highlights the dependency on the number of traces $N$ and the distinguisher $D$ being used; obviously the experiment depends on the primitive under attack, and how it leaks, as well. The random variable $\mathrm{keyrank}^j_D(N)$ denotes the rank of the $j$th subkey (that is, the experiment returns $\mathrm{rank}^j_{sk^j}(\mathbf{D}^j)$ instead).

**Definition 3 (Success rate).** *The success rate of a distinguisher* $D$ *as a function of the number of traces* $N$ *is defined as* $SR_D(N) = \Pr[\mathrm{keyrank}_D(N) = 0]$*, where the random variable* $\mathrm{keyrank}_D(N)$ *is defined by Fig. 1.*

The success rate, or first-order success rate, captures how frequently the secret key $\mathbf{sk}$ is deemed (among) the most likely by the distinguisher. Given that the score for a full key is computed as the sum of its constituent subkey scores, a full key is deemed the most likely iff all its constituent subkeys are

```
**experiment** $\mathbf{Keyrank}_D(N)$:
$\mathbf{sk} \overset{\$}{\leftarrow} \mathcal{K}$
$x_1, \ldots, x_N \overset{\$}{\leftarrow} \mathcal{X}$
For each $x_i$ capture trace $T_i$ leading to points
of interest $P_{i,j}$
For each subkey: $\mathbf{D}^j \leftarrow D^j((P_{i,j}, x_i)_{i=1,\ldots,N})$
$\mathbf{D} \leftarrow \{\mathbf{D}^j\}_{j=1}^m$
**return** $\mathrm{rank}_{\mathbf{sk}}(\mathbf{D})$
```

**Fig. 1.** The key rank experiment leading to random variable $\mathrm{keyrank}_D(N)$.

the most likely. Thus, when focusing on success rate, it suffices to look at the (first-order) *subkey* success rate.

Unfortunately, judging a distinguisher by its success rate only ignores key recovery attacks that include key enumeration as part of their strategy. One could look at higher-order success rates, where for the $M$-th order key recovery, the $M$ highest ranked key guesses are tested using a known plaintext–ciphertext pair, though this raises the question for which $M$ (and for realistic but large $M$, say $M = 2^{50}$ computing the $M$-th order success rate is a challenge on its own). Instead, we suggest to maintain the notion of $\mathrm{keyrank}_D(N)$ as a random variable and we will investigate its distribution as a whole. This allows us to identify those properties of the distribution crucial to a holistic assessment of the potency of a side channel attack.

*Remark 2.* While the random variable $\mathrm{keyrank}_D(N)$ is defined over the randomness of key, plaintexts, and the noise in the measurement, we emphasize that it is really the latter that matters. Indeed, one could equally consider key rank in a (non-adaptive) chosen plaintext setting and later on we will make the assumption that the randomness of the key is irrelevant (namely that conditioning the random variable $\mathrm{keyrank}_D(N)$ on the key $\mathbf{sk}$ makes no difference). This does mean that if the leakage is noise-free, looking at key rank as a random variable is not that meaningful anymore. Instead, the leakage will allow an adversary to determine a set (containing $\mathbf{sk}$) of most likely keys it considers equiprobable; the relevant metric in this case is the size of this set, not the rank as we defined it (which will default to 0).

## 2.3 Theoretical characterization of the key rank distribution

When comparing DPA distinguishers, it is customary to assume a specific leakage model (e.g. the Hamming weight of some intermediate value with Gaussian noise added). When the subkey distinguishing vector is considered as a random variable (cf. Fig. 1), its distribution is known [18]: it takes the shape of a multivariate normal distribution. Using order statistics, this leads to a characterization of the subkey rank distribution. This distribution is not particularly insightful in its algebraic form, but it can be numerically evaluated in time proportional to the $n$ (the size of the subkey space). However, extending this characterization into one for the full rank is not possible as the subkey rank distribution does not uniquely determine the full key rank distribution. One could attempt to use order statistics directly on the full key distinguishing table. However, even if this were possible, the resulting formulae are likely unwieldy in their algebraic form; moreover, numerical evaluation would this time be proportional in $n^m$ (i.e. the size of the full key space) which will be infeasible for any cryptosystem of relevance. This renders a full theoretical derivation of the key rank distribution moot. Instead, let us concentrate on typical statistics used to describe distributions, starting with the expected value, or the guessing entropy. Later we will hypothesise a candidate distribution.

**Guessing entropy.** First defined by Massey [15], the guessing entropy captures the expected number of guesses (with an optimum strategy) to correctly guess the value of a random variable (in our scenario the secret key). This can be linked to the key rank by observing that the key rank is the number of guesses an optimal adversary would take to guess the secret key. Standaert et al. [20] first made this connection. We use the definition as given by Rivain [18].

**Definition 4 (Subkey guessing entropy).** *The subkey guessing entropy is defined as the expected value of the subkey rank, namely*

$$GE_D^j(N) = \mathbb{E}(\mathrm{keyrank}_D^j(N)) \ .$$

A key observation is that the guessing entropy is the *expected value* of the distribution of the subkey rank. Rivain found that the distribution of a *distinguishing vector* tends to a multivariate Gaussian [18], but the general distribution of the subkey rank itself has not been thoroughly explored.

Extending the guessing entropy metric into the context of a full key is simple—we now are required to find the expected value of the key rank.

**Definition 5 (Key guessing entropy).** *The key guessing entropy is defined as:*

$$GE_D(N) = \mathbb{E}(\mathrm{keyrank}_D(N)) \ .$$

**Ranking entropy.** In this work we consider adversaries that would employ key enumeration as part of their attack strategy. This raises the question of how best to consider the relative strength of two adversaries that have different sized key enumeration budgets. Most differential attacks are chosen plaintext attacks, and thus the cost of checking the validity of a single key hypothesis is almost zero—a single call to an encryption or decryption. Thus, as in classical cryptanalysis, it is perhaps more useful to compare enumeration budgets in terms of *orders of magnitude*, i.e. consider the logarithm of (a function of) the key rank outcomes.

Recall that the guessing entropy $GE_D(N)$ is defined as $\mathbb{E}(\mathrm{keyrank}_D(N))$ To consider the orders of magnitude in relation to the guessing entropy, the obvious approach would be to consider $\log(GE_D(N)) = \log(\mathbb{E}(\text{keyrank}_D(N)))$. We will later show that this approach is not satisfactory. For that reason, we introduce here an alternative, which we call the *ranking* entropy. The ranking entropy is defined as the expectation of the logarithm of the rank, that is it equals $\mathbb{E}(\mathfrak{R})$, where $\mathfrak{R} = \log(\text{keyrank}_D(N))$ (for brevity, we will henceforth refer to $\text{keyrank}_D(N)$ simply by $R$). Note that taking logarithms and expectation do not commute, so in general the ranking entropy will not equal the log of the guessing entropy.

Calculating either the guessing entropy or the ranking entropy directly appears to be a hard problem. Instead, for this and other statistics we will resort to sampling from the distribution by repeatedly running the experiment of Fig. 1 instead. This requires an algorithm to calculate the key rank.

## 2.4 Key rank estimation

We want to understand the distributional properties of the key rank for different distinguishers and leakage scenarios. A key tool for our empirical investigation is an efficient and highly accurate rank estimation algorithm. Finding the rank of a subkey is trivial after sorting the distinguishing vector. Unfortunately, for the full key this approach no longer works as sorting the complete distinguishing vector for the full key is at least as expensive as exhaustive search on the full key. For instance, in case of a typical attack on AES, the distinguishing table consists of 16 distinguishing vectors of dimension 256 each. A naive (but accurate) algorithm would be to compute the product distribution (i.e. list all combinations of all subkeys) in order to compute the rank of the secret key.

There have been a host of more advanced key rank *estimation* algorithms that return either an interval containing the actual rank or a point estimate of the rank. When comparing such algorithms, both the efficiency and the accuracy are relevant. Accuracy is measured in bits, where $b$ bits of accuracy means that if an algorithm says the key has rank $2^x$, the actual rank is in the range $2^{x \pm b}$. Below we give a brief overview of existing key rank estimation algorithms.[2]

Veyrat-Charvillon et al. [24] proposed the first non-trivial key rank algorithm. They represent the distinguishing scores in a multi-dimensional space, where each dimension represents an individual distinguishing vector (sorted in descending order). This space can naturally be divided into two parts; those keys with rank higher than the target key and those with a rank lower. Using the property that the 'frontier' between these two halves is convex, the rank of the key can be estimated to within 10 bits by repeatedly pruning the space.

Glowacz et al. [10] construct an efficient rank algorithm based on the convolution of histograms. They utilise the property that if $H_1$ is a histogram of $\mathcal{S}_1$ and $H_2$ is a histogram of $\mathcal{S}_2$ then the convolution of $H_1$ and $H_2$ is a suitable approximation of $\mathcal{S}_1 + \mathcal{S}_2 = \{x_1 + x_2 | x_1 \in \mathcal{S}_1, x_2 \in \mathcal{S}_2\}$. By representing the

[2] A small technical caveat: we do not make a distinction between worst-case accuracy and the more fuzzy typical-case accuracy.

distinguishing vectors as histograms and using this property they are able to estimate the rank of the key to within one bit of accuracy.

Duc et al. [6] propose a similar solution to that of Glowacz et al. [10]. They repeatedly 'merge' each set of data in (similar to the histogram convolution) and then down-sample the resulting data (this can be seen as the binning step in creating histograms). Additionally, they down-sample to a fixed number of samples after each 'merge', instead of just on the original data. While Duc et al. do not explicitly give a bound on the estimation error, the additional down-sampling implies it will be worse than that of Glowacz et al.'s algorithm.

Bernstein et al. [2] propose two key rank algorithms. The first adds a post-processing phase to the algorithm by Veyrat-Charvillon et al. [24], which tightens the accuracy to 5 bits. The second algorithm uses techniques similar to counting all $y$-smooth numbers less than $x$. By having an accuracy parameter they are able to get the bound arbitrarily tight, at the expense of runtime.

Martin et al. [14] propose a key rank algorithm based on the pseudo-polynomial time algorithm for the knapsack problem. After mapping the distinguishing scores to integer weights (such that larger distinguishing scores give smaller integers), they are able to efficiently count the number of keys with a weight less than the target key which directly corresponds to the rank of the key. Varying the size of the resulting integers allows them to make a trade-off between accuracy and runtime.

All-but-one of these algorithms are essentially interval estimates of the key rank; the only exception being the algorithm by Martin et al., which provides a point estimate. Clearly all works emphasised the need of an accurate rank estimation to ensure that the resulting key ranks are practically meaningful. In some of these papers, as well as in related work on key enumeration [25], some observations were made about the seemingly large variation of the key rank. Poussier et al. [16] compared a number of the interval-based algorithms to determine to what extent this variation was due to the algorithm being used (despite the researchers' best efforts to improve the accuracy of their algorithms, estimation introduces an error and with it variation). Our interest is not in the 'algorithmic' noise, but rather in the intrinsic distributional properties of the key rank itself.

For our empirical investigation into the key rank distribution, we opted for Martin et al.'s approach, as we found that it provides the best efficiency/accuracy tradeoff (it gives better accuracy than the algorithm by Glowacz et al. and is more efficient than the second algorithm by Bernstein et al.).

### 2.5 Summary statistics

To be able to explore the characteristics of the key rank distribution further, we must sample from $R$ and estimate it—samples for $\mathfrak{R}$ are calculated by applying the logarithm to each sample from $R$. A first concern is to try to find the most appropriate estimators for the expected values of $R$ and $\mathfrak{R}$. However, these random variables have characteristics other than their mean (e.g. variance). To explore these, additional summary statistics—measures of location and spread—are necessary and we review the potential choices in the following.

*Estimates of the mean* To compute the ranking entropy and the log of the guessing entropy (Sect. 2.3) we must estimate $\mathbb{E}(R)$ and $\mathbb{E}(\mathfrak{R})$. The arithmetic (or sample) mean of $N$ samples $x_1, x_2, \ldots, x_N$ is $\bar{x} = (x_1 + x_2 + \ldots + x_N)/N$. The law of large numbers states that the arithmetic mean over a large number of trials should be close to the expected value, and thus is the correct estimator for $\mathbb{E}(R)$.

When orders of magnitude are of concern, the arithmetic mean may not be suitable—consider a hypothetical scenario in which a DPA attack is evaluated 1024 times. In 1023 of the occasions, the rank of the key is 1, and in the one remaining occasion the rank of the key is $2^{32}$. The arithmetic mean in this case is (just over) $2^{22}$, which clearly misrepresents the strength of the attack. In this case, the geometric mean of $N$ samples $x_1, x_2, \ldots, x_N$ may be more appropriate. It is defined as:

$$\tilde{x} = \left( \prod_{i=1}^{N} x_i \right)^{\frac{1}{N}}$$

The logarithm of the geometric mean of $R$ is the arithmetic mean taken on $\mathfrak{R}$ ($\log \bar{R} = \tilde{\mathfrak{R}}$). Consequently, the geometric mean is a suitable estimator for the ranking entropy $\mathbb{E}(\mathfrak{R})$. With reference to our prior 'extreme example' the geometric mean would deliver a rank of (just over) 1—a better judgement on an adversary's "order of magnitude" ability.

*Standard deviation* The estimated standard deviation

$$\hat{s}_X = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})^2}$$

captures the degree of variation in a distribution. From the side-channel evaluation perspective, this will be of concern—if the standard deviation of $R$ is large, then the adversary has a higher probability of being "lucky" (or "unlucky"). A similar geometric standard deviation exists, such that the geometric standard deviation of $R$ is equivalent to the arithmetic standard deviation of $\mathfrak{R}$.

*Order statistics* An alternative, non-parametric set of order statistics are the estimated percentiles of the distribution. The nearest rank $P$-th percentile is the smallest value in an ordered sample such that $P$ percent of the data set is less than or equal to that value. More formally, the index $i$ in the ordered list of $N$ samples is

$$i = \left\lceil \frac{P}{100} N \right\rceil ,$$

with the $P$-th percentile taken to be sample $x_i$.

The median (or 50th percentile) is a non-parametric measure of central tendency. In the case of our previous hypothetical scenario, the arithmetic mean was $2^{22}$, despite 99.9% of the ranks being 1. In the same scenario, the median would report 1, a much more representative value for the strength of the adversary. The median (and percentiles in general) have already seen use as descriptive statistics in the context of key rankings in Veyrat-Charvillion et al. [23].

Finally, the minimum and maximum values observed within a sample may be important. In the side-channel context, these essentially correspond to estimates for the best and worst case scenario for the adversary (and vice-versa for the evaluator). The minimum value could also be associated with an indication of the min-entropy of the distribution (although we leave this as an avenue for future exploration).

The order of a set of samples from $R$ is invariant under logarithms, and thus the minimum, maximum and percentile values from $\mathfrak{R}$ can be computed by taking the logarithm of the values for the equivalent samples from $R$.

## 3 Accurate estimation of the rank distribution

The ability to characterise the distribution of $R$ hinges on whether a sufficiently accurate estimation of an individual rank can be achieved. As previously established, the rank estimation algorithm of Martin et al. (hereafter, "KRE") is the optimal choice from the candidate set of algorithms for our experiments.

### 3.1 KRE improvements

The KRE algorithm can be seen as having two components or steps: the first is a lossy conversation from floating point distinguishing scores to integer weights, and the second is an accurate counting method.

For the first step, the KRE algorithm takes a precision parameter, which is a number of bits $p$. Each of the distinguishing scores produced by a side-channel attack are then converted to positive integers of size at most $2^p$. A typical side-channel attack produces floating-point distinguishing scores which, assuming the use of a modern CPU, are highly likely to be computed using 64-bit floating point arithmetic. Thus for any $p < 64$, the conversion from raw distinguishing scores to integer values is lossy, and can theoretically 'collide' two different distinguishing scores together into the same integer value, losing some of the information produced by the side-channel attack.

The runtime of KRE is effectively exponential in $p$; for the same set of distinguishing vectors, ranking at precision $p + 1$ will take approximately twice as long as ranking at precision $p$. Using a variety of algorithmic and implementation improvements, we were able to accommodate a large increase in the precision retained by the algorithm. These improvements enabled us to perform rank calculations approximately 16 times faster than the previous work, allowing us to run experiments at a precision of up to $p = 23$ in the order of 1-2 minutes (depending on the 'true' rank being estimated).

These improvements included a modification to the first step (the "MapToWeight" function as described in [14]). We applied a linear shift to the integer weights such that the subkey with the smallest distinguishing score has an integer weight of 1, and thus typically lowers the integer weight of the correct key (which affects the run-time linearly). In addition to some optimisations at the level of the implementation, we also modified the recurrence relation to avoid all calls to the "left child" function. With these modifications, we were able to push our implementation to retain up to 23 bits of precision. Full details can be found in Appendix A.

### 3.2 KRE precision

To provide a sanity check of how many bits of precision suffice for computing an 'exact' rank (similarly to the brief evaluation in [14]), we simulated a large number of DPA attacks and used the key rank estimation algorithm to estimate the rank of each attack using 8 to 23 bits of precision. Figure 2 and Table 1 illustrate the average error between our best guess at the true key rank (which is obtained by taking the estimate at 23 bits) and the rank estimates at each level of precision. Each additional bit of precision used in the rank estimation algorithm can only increase accuracy (increasing the number of bits by one approximately doubles the weight of the target key; this will reduce the number of collisions when converting the distinguishing scores to integers and can not introduce new collisions).

As can be observed in the figure and table, the average error rapidly decreases between 8 and 14 bits of estimation precision. From 17 bits of precision onwards, the average error is within 3 decimal places, dropping as low as 4 decimal places at 20 bits of precision, and with each additional bit approximately halving the average error. Given our available computational budget for all our experiments, we selected 20 to be the precision used for the KRE algorithm. This allows us to both very accurately estimate ranks and to run a large amount of experiments.

| Precision | Av. error (bits) | Precision | Av. error (bits) | Precision | Av. error (bits) |
|---|---|---|---|---|---|
| 8 | 0.302619 | 13 | 0.010231 | 18 | 0.000330 |
| 9 | 0.158402 | 14 | 0.005343 | 19 | 0.000154 |
| 10 | 0.082911 | 15 | 0.002756 | 20 | 0.000074 |
| 11 | 0.041216 | 16 | 0.001473 | 21 | 0.000033 |
| 12 | 0.020488 | 17 | 0.000641 | 22 | 0.000015 |

**Table 1.** The average error, in bits, for increasing increments of precision used in the rank estimation algorithm. Average taken using 1091 DPA attacks with ranks spread across the range $2^0$ to $2^{128}$, using the geometric mean.

## 4 Initial exploratory study

Now we have shown that we can estimate values from $R$ with a high degree of accuracy, we shift focus to exploring its distribution.

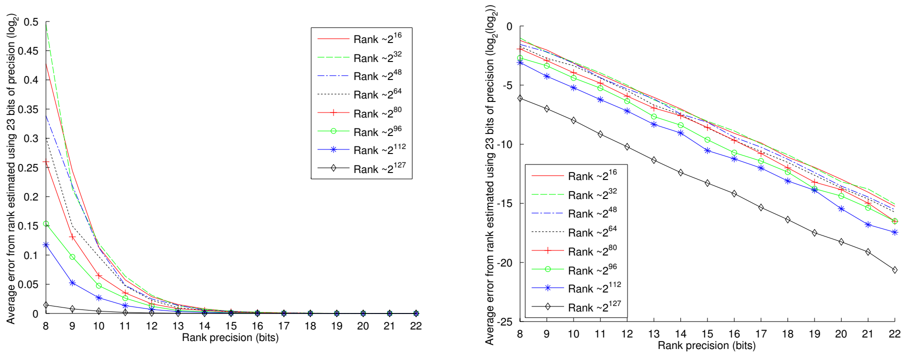

**Fig. 2.** (Left) Average error, in bits, from a 'true' rank taken to be the estimate as evaluated by the rank precision algorithm using 23 bits of precision. Rank estimates were evaluated using 8 to 22 bits of precision. Repeated DPA attacks were simulated using a random SNR, and placed into buckets if the estimated rank at 23 bits of precision was within 1 bit of $2^8, 2^{16}, \ldots, 2^{128}$. (Right) the same data, with the logarithm of the log-ranks applied.

## 4.1 Visualising the key rank distribution

As a first step we proceed to visualise the distribution of repeated key rank experiments at various depths. Histograms are an ideal tool for doing this; we hence run simulated experiments, using correlation power analysis (CPA, see [5]) as a distinguisher for attacking simulated Hamming-weight leakage with additive Gaussian noise with a low SNR of $2^{-7}$.

Figure 3 plots histograms of samples from $R$ across a range of different average rank values. In the middle range of rank values, the distribution appears to be appreciably normally distributed. However, we can observe non-normal behaviour at either end of the possible rank values, as can be seen in the top-left and bottom-right histograms. The bottom-right exhibits a much higher frequency of attacks of rank 0, producing a small additional peak at the left-tail of the distribution. Similarly, when the average rank is close to the maximum of $2^{128}$, the distribution is no longer symmetric, but is also without the additional peak. A review of statistical literature suggests that distributions that are 'clipped' in this way are defined as *truncated* distributions [9].

The x-axis of the histograms is log-scale: if the distribution of the logarithm of the ranks was indeed normal, then the distribution of the rank values themselves would be a log-normal distribution. The large skewness of a log-normal distribution would support our hypothesis that the arithmetic mean is not a suitable average, and rather the geometric mean is better suited. Our prediction of a log-normal distribution is supported by the central limit theorem, which implies that the product of positive random variables produces a log-normal distribution.

Given this information we conjecture that we have a delta-log-normal [1] distribution with truncation [9]. A delta-log-normal distribution is a distribution on

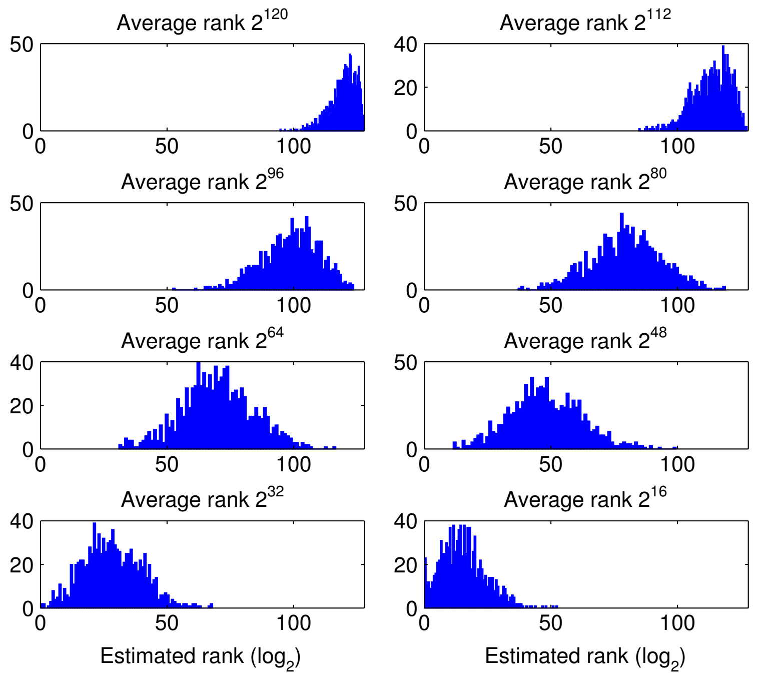

**Fig. 3.** Histograms for attacks with a (geometric) mean rank close to one of several values. Here the leakage is simulated Hamming-weight with Gaussian noise at an SNR of $2^{-7}$, with the attacker using CPA as a distinguisher.

a random variable $X$ such that $X$ is assigned value 0 with probability $\theta$ and follows the log-normal distribution with probability $1-\theta$. In this particular context the value of $\theta$ would directly correspond to the success rate of an adversary for full key recovery (without enumeration). The log-normal distribution could then be parameterised separately using standard methods. Truncation corresponds to when a random variable can not be assigned a value passed a certain threshold. It is clear that the rank can only be assigned a value between 0 and $2^{128}-1$, and thus must be truncated.

Whilst further research into this characterisation is a promising next-step, for the purposes of this work we instead pursue two questions of immediate importance: firstly, whether this shape and scale of distribution is consistent across the various contributory factors influencing the outcomes of side-channel attacks, and secondly whether the *non-parametric* order statistics outlined in Sect. 2.5 can be used as a simple and efficient method for extracting meaningful conclusions without making any assumptions about the underlying distribution.

## 4.2 Is an accurate rank distribution estimation viable?

Before further exploration of the candidacy of the summary statistics outlined in Sec 2.5, we devised an experiment to determine how many repeat experiments are necessary to reliably estimate them. We kept the leakage model and SNR, as well as the distinguisher used by the adversary, constant but used randomly generated plaintexts, keys and Gaussian noise. We assumed a CPA attack using the Hamming-weight power model, and the leakage was simulated on the AES SubBytes target function, using the Hamming-weight leakage function and Gaussian noise. In the experiment, each statistic was estimated using increasing amounts of repeat experiments on simulated data.

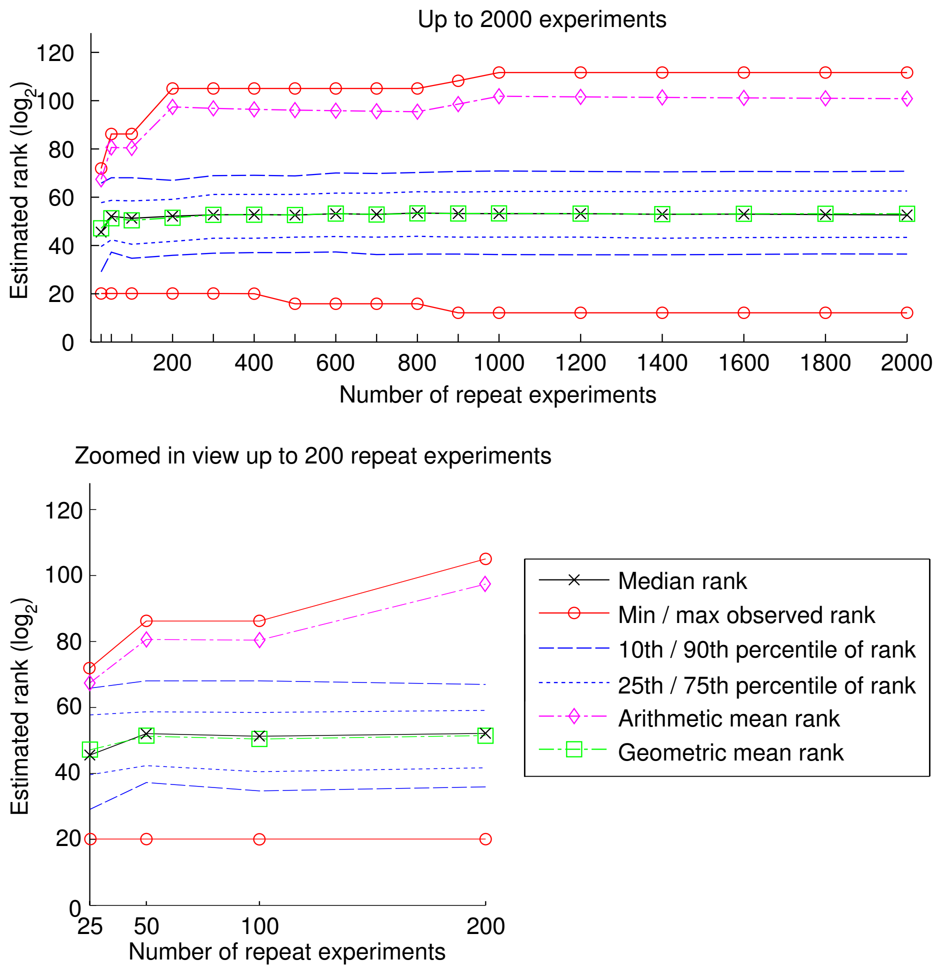

**Fig. 4.** Estimated summary statistics using increasing amounts of DPA attacks. Each DPA attack used a CPA-HW distinguisher on simulated AES SubBytes leakage using the Hamming-weight power model and Gaussian noise of an SNR $2^{-3}$.

The results in Fig. 4 exhibit the behaviour of the statistics. The maximum key rank values unsurprisingly exhibit the most variability—for key ranks above 80 we observe that the estimated values 'jump' at 50, 100, and 200 repeats where they stabilise. The other key ranks, hence those in the ranges were enumeration is within practical reach behave much more stable—from 25 repeats on they produce stable estimates, from 100 repeat experiments onwards the estimates have converged to the true value. The intuition behind the geometric mean being a sensible choice is sound, producing a line that is almost identical to that of the median, as expected under the assumption of a log-normal distribution. In fact, for all the experiments we pursued in this study, the geometric mean and median were nearly identical, and for simplicity we do not display it in future graphs. The unsuitability of the arithmetic mean (given orders of magnitude are a concern) is clear and consequently from here onwards we no longer calculate it.

*Resampling methods.* In the previous experiment, which was based on simulations, we were able to efficiently sample independent and mutually exclusive sets of key rank data. In practice this might not be possible as a single, large data-set might be available only. This situation is not uncommon and methods such as bootstrapping, jackknifing and k-fold cross validation are well understood [22,4,11] and therefore get employed in a variety of contexts. An important guideline though, irrespective of which resampling approach one chooses, is to pay attention to *randomly* selecting subsamples to avoid introducing a bias.

## 5 Characterising rank distributions

To understand whether the properties observed in the exploratory studies of Sect. 4 are common (or specific to the combination of distinguisher, leakage model and SNR), and to further explore characteristics of the distribution of $R$, we perform further simulated DPA attacks and vary the interesting parameters.

### 5.1 SNR and measurement counts

The starting point of our simulated experiments was to consider whether both the measurement SNR and the quantity of trace measurements available affects the rank distribution. These two variables are clearly dependent; a very low SNR can be overcome by using more measurements, and at high SNR levels a successful attack can be created using fewer trace measurements. As a consequence we devised a set of experiments for which the rank distribution can be analysed as both these variables change. We assumed an 'optimal' adversary operating under commonly considered leakage conditions—namely, Hamming-weight leakage on the AES SubBytes operation with additive Gaussian noise, and where the adversary launches a CPA attack using the Hamming-weight as a power model.

We simulated data under a variety of SNRs, beginning with a low-noise scenario of SNR $2^{-1}$, up to a high-noise scenario with SNR $2^{-7}$. For each unique

SNR, we simulated DPA attacks using increasing amounts of traces, beginning with a quantity for which the rank was approximately $2^{128}$, and increasing the number of traces until the vast majority of attacks produced a rank of 0. For each unique number of traces, we ran 1000 repeat attacks, and for each repeat generated the keys, plaintexts and additive noise at random.

Figure 5 visualises the summary statistics for attacks under the SNRs $2^{-7}$, $2^{-5}$ and $2^{-3}$. The general trends appear similar to those observed in our real world example. The variance observed is of most interest, both in terms of its magnitude and its consistency across multiple pairs of SNR and trace quantities.

Three main observations can be made:

1. The distribution appears to be at its widest in the middle range of ranks (e.g when the rank is between $2^{40}$ and $2^{80}$), and variance minimises for very poor attacks (rank $\approx 2^{128}$) and very good attacks (rank $\approx 2^{0}$).
2. The maximum variance appears to be very large, with the difference between (for example) the 10th and 90th percentiles being in the order of up to 40 bits in some cases.
3. The exact level of SNR does not appear to affect the variance or shape of the distribution in any independent way—assuming the same distinguisher is used, at any given SNR, given sufficient traces to establish an average rank of $x$, the dispersion of the distribution will be very similar to that produced by attacks at any other SNR that have an average rank close to $x$.

To confirm these three intuitions, we plotted the estimated geometric standard deviation against the (geometric) mean rank (or equivalently the arithmetic mean and standard deviation of samples from $\mathfrak{R}$). The results can be seen in Fig. 6, where each line corresponds to results obtained for seven different SNRs. The shape and magnitude of each line very closely match, indicating that the behaviour is indeed consistent across all SNRs. The curves peak at an average rank of approximately $2^{64}$, suggesting that it is the 'true' *rank of the attack* that affects the variance, and *not* any characteristic of the leakage noise or quantity of data available (for a fixed key rank).

These three characteristics in tandem present an unfortunate problem for an evaluator and for the viability of the guessing entropy as a stand-alone metric. Not only is the variance very large, and thus an adversary may with non-negligible probability produce an attack far out-performing the average attack, but also the variance is largest in the range of key ranks that are of most interest to an evaluator. There is a threshold at which an adversary may be considered unrealistic (e.g we might be confident that an adversary can enumerate $2^{54}$ keys, but not $2^{57}$), and unfortunately the distribution has the most variance here.

## 5.2 Distinguishers and higher-order attacks

A second consideration is whether the choice of distinguisher used by the adversary can change the characteristics of the rank distribution. Our previous experiments used CPA as the distinguisher, and so to compare, we launched

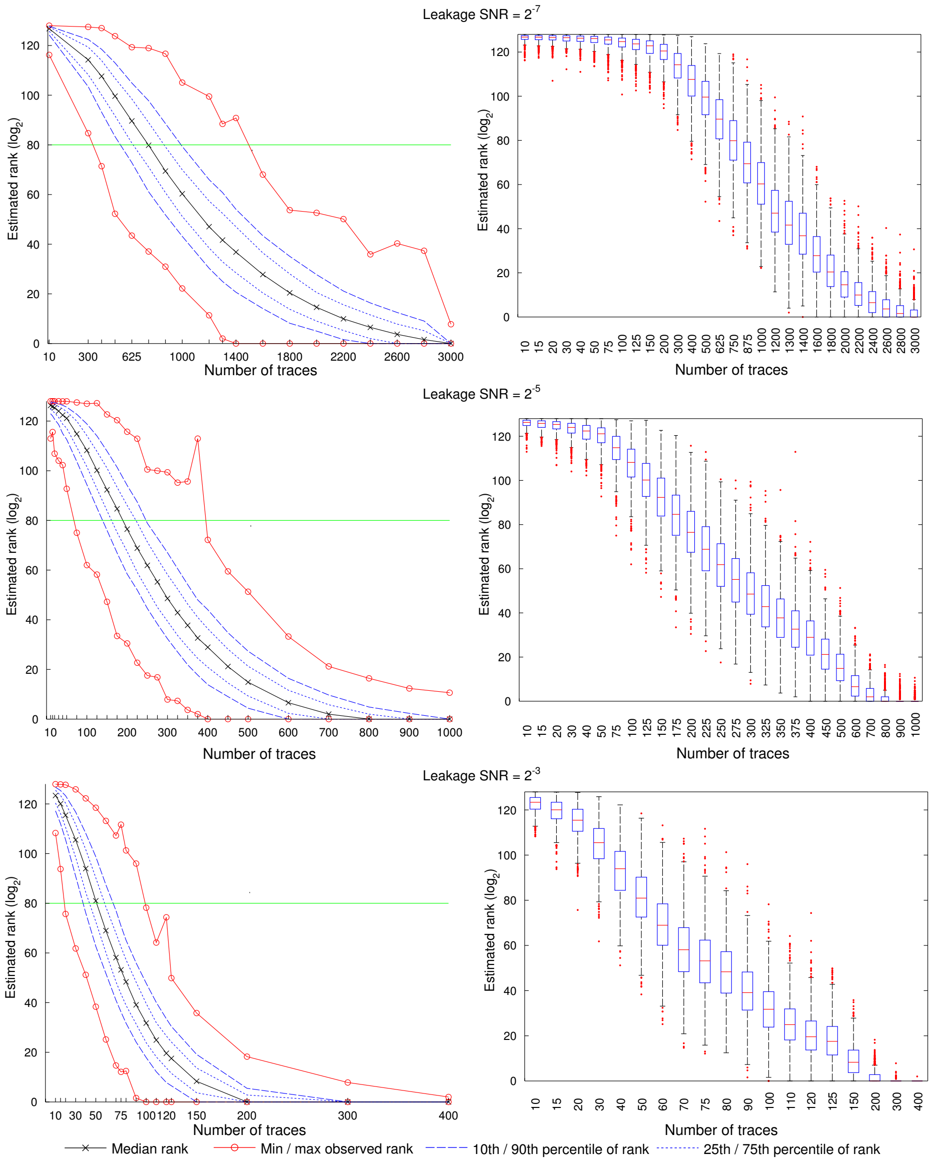

**Fig. 5.** (Left) Estimated ranks after 1,000 DPA attacks at SNRs $2^{-7}$, $2^{-5}$ and $2^{-3}$, using Hamming-weight CPA targeting simulated leakage on the AES SubBytes operation. (Right) Equivalent box-plots for using the same data as on the left. The central line in each box is the median, the box defines the inter-quartile range, the whiskers cover all samples not considered to be outlier values, and outliers are plotted individually.

two additional types of attacks. Firstly, we tried reduced[3] template attacks on

[3] Reduced in that we did not use multiple points or estimate a covariance matrix.

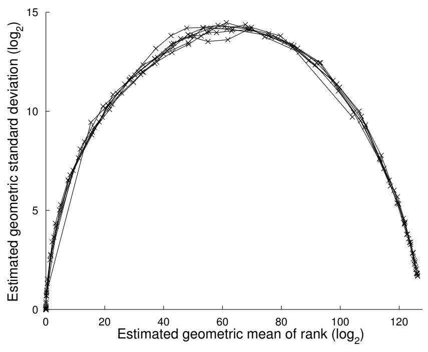

**Fig. 6.** A plot of the estimated geometric standard deviation against the geometric mean of the key rank, taken using 207,000 DPA attacks. Each line represents the standard deviation for attacks at the seven SNR values $2^{-7}, \ldots, 2^{-1}$. Each attack used simulated Hamming-weight leakage with CPA used as the distinguisher.

the simulated leakage. Secondly, we launched second-order attacks on a binary masked implementation of AES: the leakage sample corresponding to the mask value and the sample corresponding to the masked SubBytes operation were combined using the 'centre and multiply' method (see e.g [17]), and then a standard Hamming-weight CPA was launched. To enable a direct comparison with the standard CPA attacks, we ran the template attacks using data with an SNR of $2^{-7}$. For the second order attack, we reduced the SNR to $2^{-3}$ to alleviate the burden of having to use too many traces in the attack.

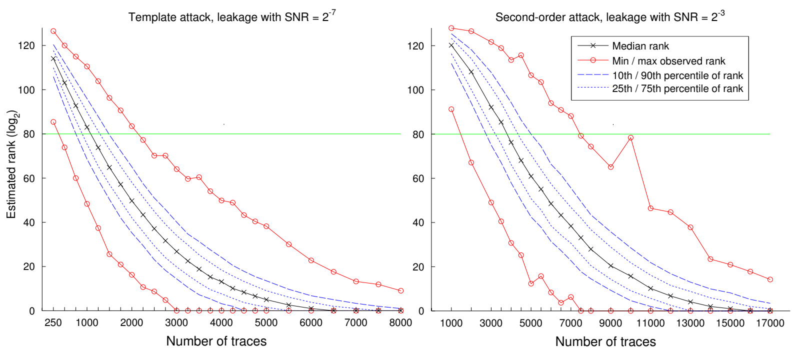

**Fig. 7.** Estimated ranks after template and second-order attacks, using simulated leakage on the AES SubBytes operation.

Figure 7 shows the results of our attacks. Again, we observe consistent behaviour as seen previously; the shape and trend for the percentiles is remarkably similar. We can observe one interesting discrepancy: the variance of the distribution produced by the template attacks, whilst still very large in the middle of the distribution and with a consistent shape, does appear to be *smaller* than that produced by any attack using correlation as a distinguisher (including the second-order attacks, which produce very similar rank variance to first-order ones). The reason for this distinguisher-specific dependency is unclear, and we leave this observation as an interesting starting-point for future research.

## 6 Embedding Rank Estimations into Real World Security Evaluations

In the previous sections we provided conclusive evidence that the key rank is random variable with inherently large variation. We showed that it is possible to meaningfully characterise average behaviour and spread using repeat experiments. A crucial questions remains though: how can this be integrated into practical side channel evaluations? In this section we discuss two radically different propositions for a solution. The first proposition is to employ some recent suggestions for short-cuts in evaluations; we find that these have shortcomings which limit their practical use. The second proposition is a practical re-use of measurements for repeat experiments, which leads to practically meaningful results.

### 6.1 Bounding the success rate of an adversary with enumeration

In some recent work, Duc et al. [6] provided some bounds that relate the mutual information between a subkey and the leakage traces, as a function of the adversary's success rate, the number of shares (if used within a masking scheme) and the number of traces used within a side channel attack. They also present a construction relating the success rate, enumeration effort and number of traces (for a fixed SNR and number of masks), in the best case for the adversary in an extended version of their paper [7, Sect. 4.3c, eq 24, Alg. 2]. We can interpret this as a lower bound on the key rank of the secret key at a given number of messages, by looking when the success rate first becomes non-zero.

Using code supplied by the authors of [6], we were able to evaluate this success rate bound in the context of idealised Hamming-weight information leakage. This data is shown in Fig. 8, re-using the simulated Hamming-weight CPA attack data under first-order and second-order attack conditions. As can be immediately seen from the large margin between the SR bound and the estimated ranks, this is a very loose bound—in the right hand graph, the SR bound is almost on top of the x- and y-axes. This supports their intuition (hinted at in [6]) that the theoretical bounds only tighten for a large number of masks, but cannot realistically approximate the performance of an adversary in the single or zero mask situation our work explores. Consequently this recent work, which hopes to 'short-cut' the effort in evaluations, seems to inaccurate for the kind of implementations that are of immediate real world interest.

Using a different technique to Duc et al., but with the same intention to short-cut evaluation efforts somewhat,Ye et al. argue for an algorithm that allows

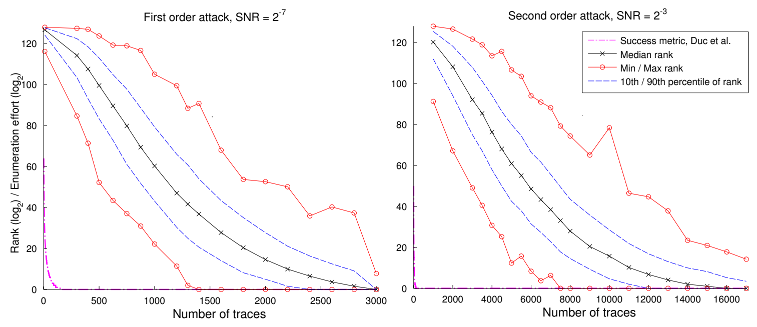

**Fig. 8.** Estimated key ranks compared to the success rate metric of Duc et al. [6] using simulated first and second order CPA attacks on Hamming-weight leakage.

to estimate the remaining effort of an adversary regarding enumeration and simultaneously provides the optimal guessing strategy. They suggest that their algorithm could be run once on a dataset. However, running their algorithm once can only deliver a single interval estimate of the key rank: repeat experiments would still produce a large variance which implies that any statement based on a single run is insufficient to determine the spread.

## 6.2 Real world evaluation of a challenging target

We utilise a interesting real-world data set provided by the authors of Longo et al. at CHES 2015 [12] to illustrate how to integrate key rank into practical evaluations. We re-implemented and re-ran one of the attacks described by Longo et al. at CHES 2015 [12]. They illustrated several standard DPA attacks on a complex device, and we selected their most challenging one: an attack on a hardware AES implementation, utilising EM measurements. We refer the reader to the attack paper for full details, but note that we use an improved attack strategy communicated to use after correspondence with the authors [12].

The available dataset consisted of approximately one million EM traces. These were acquired in line of their 'standard' assessment approach for cryptographic devices, which as some of the authors are from a well known expert company, can be regarded as being in line with industry best practice: after initially identifying the source of the leakage, they gathered as many traces as they could afford (given some allotted time budget) for a given unknown secret key.

In the previous section of this paper we highlight the fact that estimations of key rank properties need repeat experiments. However, due to the EIS property that typical block ciphers have [19], it is not necessary to run these on different keys. Instead we can divide up any set of experiments into smaller subset to run repeat experiments, which is how we proceed. To analyse the distribution of $R$ produced by the attacker[4] we ran multiple DPA attacks using increasing amounts

[4] The attack is a Hamming-distance correlation power analysis on the input and output of the final round of encryption.

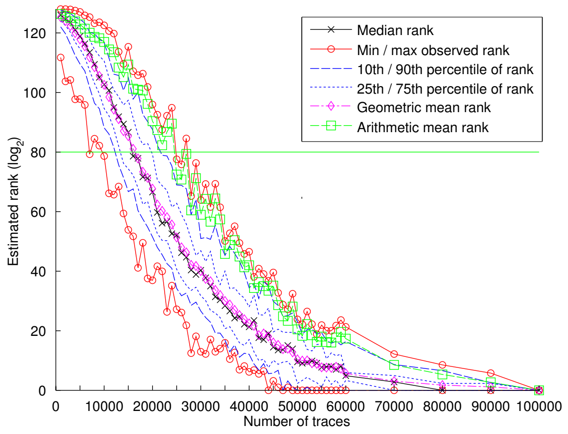

**Fig. 9.** Estimated key ranks after repeated DPA attacks on a set of 997,500 traces acquired from a BeagleBone Black device running AES-128 in hardware.

of traces from the data set. Figure 9 plots the trends of the minimum, maximum, various percentiles and geometric and arithmetic means for the estimated ranks as the number of traces available to the attacker increases.

The first attack reporting full key recovery uses approximately 45,000 traces, and we can immediately see from the graph that this should perhaps be considered a fortunate result for the attacker—at this number of traces, we observed some attacks of rank up to $\approx 2^{20}$. Also of interest is that the classical 80-bit security margin is first broken somewhere between 10,000 and 20,000 traces, and expected, considerable variance in the rank distribution, with a very large margin between the minimum and maximum values observed. The line for the arithmetic mean is again evidence that our intuition of computing statistics on $\mathfrak{R}$ is more meaningful—the line corresponding to the geometric mean (of $R$) is very close to the median.

**The power of percentiles** Percentiles are particularly informative statistics in the evaluation context. Recall that percentiles give the value below which a specific percentage of observations (among the sampled observations) fall. We can relate this to business goals such as having no more than a certain percentage of devices be susceptible to a particular side channel adversary, as we show in the example below.

Consider our evaluation of the real world data-set before: we sampled from the rank distribution using repeated attacks for an increasing amount of traces. Risk can be assessed using these key rank samples. As an example, assume that the 10th percentile of the estimated rank distribution is $2^{44}$ (in the Fig.9) — this indicates that of all the devices of that type sold into a market, 10% would succumb to a full key recovery attack by an adversary using around 23,000 traces and with $2^{44}$ as an enumeration budget. An alternative but equivalent interpretation would be that 90% of the devices are only vulnerable once the adversary's enumeration budget increases beyond $2^{44}$ (at 23,000 traces). Instead of phrasing attack scenarios around an how many devices are vulnerable, one can focus on a single devices but many adversaries. For instance, if a series of fixed adversaries attacked the same device (using 23,000 traces and enumerating up to 244 keys, then 10% would succeed).

These examples demonstrate that percentiles are a very efficient and simple way to assess the spread of the rank distribution and report it in a meaningful way in business terms. The use of different percentiles then allows the evaluator to fine-tune these security margins.

## 7 Conclusion

One of our key findings is that the shape of the distribution of the key rank is *consistent*; these observations hold irrespective of the type of differential attack used (with the small but interesting observation that template attacks seem to produce key rank distributions with a slightly smaller variance than similar correlation based attacks) and SNR. We thereby confirm that it has a large variance in exactly the range in which the assumed enumeration capability of an adversary transitions from realistic to unrealistic.

In our efforts to explore suitable statistical measures to capture the practically important key rank characteristics we observe that the guessing entropy, defined as the *expected value* of the key rank, is not always meaningful. As an average, the guessing entropy cannot quantify *any* of the very large amounts of variance we observe. Consequently, additional metrics must be used, and a natural step is to instead consider non-parametric order statistics, which brings us to consider the usage of percentiles to connect side channel outcomes with business goals.

We additionally observe that the rank distribution $R$ follows some flavour of a truncated delta-log-normal distribution. However, in practice we typically are concerned with behaviour of adversaries in the log-domain (in $\mathfrak{R}$)—an evaluator tends to be more interested in the *magnitude* of an adversary's capabilities and not the exact value. Whilst the logarithm of the guessing entropy can be appropriately estimated using the geometric mean, it is perhaps easier to switch to considering guessing entropy defined using the logarithm of the ranks, and estimated using the arithmetic mean.

With regards to practical impacts, we observed that at least some repeat experiments are necessary for stable estimates of the geometric mean, median and percentiles. Whilst this appears to incur an overhead at first glance in terms of trace measurements, we explain that it is sound to simply 'split' any existing data set into smaller subsets with which the repeat runs can be conducted. Finally we show that caution is needed with regards to using short-cut formulas, and end by illustrating an approach to evaluating the security of a real world device using repeat rank experiments.

## References

1. J. Aitchison. On the distribution of a positive random variable having a discrete probability mass at the origin. *Journal of the American Statistical Association*, 50(271):901–908, 1955.
2. D. J. Bernstein, T. Lange, and C. van Vredendaal. Tighter, faster, simpler side-channel security evaluations beyond computing power. *IACR Cryptology ePrint Archive*, 2015:221, 2015.
3. A. Bogdanov, I. Kizhvatov, K. Manzoor, E. Tischhauser, and M. Witteman. Fast and memory-efficient key recovery in side-channel attacks. *IACR Cryptology ePrint Archive*, 2015:795, 2015.
4. E. Bradley. *The jackknife, the bootstrap, and other resampling plans*. Philadelphia, Pa: Society for Industrial and Applied Mathematics, 1982.
5. 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.
6. A. Duc, S. Faust, and F. Standaert. Making masking security proofs concrete - or how to evaluate the security of any leaking device. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, pages 401–429, 2015.
7. A. Duc, S. Faust, and F. Standaert. Making masking security proofs concrete - or how to evaluate the security of any leaking device (extended version). Cryptology ePrint Archive, Report 2014/119, 2015. `http://eprint.iacr.org/`.
8. Y. Fei, Q. Luo, and A. A. Ding. A statistical model for DPA with novel algorithmic confusion analysis. In *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, pages 233–250, 2012.
9. D. J. Finney. The truncated binomial distribution. *Annals of Eugenics*, 14(1):319–328, 1947.
10. C. Glowacz, V. Grosso, R. Poussier, J. Schüth, and F. Standaert. Simpler and more efficient rank estimation for side-channel security assessment. In *Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers*, pages 117–129, 2015.
11. P. Good. *Practitioner's Guide to Resampling Methods.* CRC Press, 2012.
12. J. Longo, E. D. Mulder, D. Page, and M. Tunstall. Soc it to EM: electromagnetic side-channel attacks on a complex system-on-chip. In *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, pages 620–640, 2015.
13. 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.
14. D. P. Martin, J. F. O'Connell, E. Oswald, and M. Stam. Counting keys in parallel after a side channel attack. In *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, pages 313–337, 2015.
15. J. L. Massey. Guessing and Entropy. *IEEE International Symposium on Information Theory*, page 204, 1994.
16. R. Poussier, V. Grosso, and F. Standaert. Comparing approaches to rank estimation for side-channel security evaluations. In *CARDIS 2015*, pages 125–142, 2015.

17. E. Prouff, M. Rivain, and R. Bevan. Statistical analysis of second order differential power analysis. *IEEE Trans. Computers*, 58(6):799–811, 2009.
18. M. Rivain. On the exact success rate of side channel analysis in the gaussian model. In *Selected Areas in Cryptography, 15th International Workshop, SAC 2008, Sackville, New Brunswick, Canada, August 14-15, Revised Selected Papers*, pages 165–183, 2008.
19. W. Schindler, K. Lemke, and C. Paar. A stochastic model for differential side channel cryptanalysis. In *CHES 2005*, pages 30–46, 2005.
20. 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.
21. A. Thillard, E. Prouff, and T. Roche. Success through Confidence: Evaluating the Effectiveness of a Side-Channel Attack. In G. Bertoni and J.-S. Coron, editors, *CHES*, volume 8086 of *LNCS*, pages 21–36. Springer, 2013.
22. J. Tukey. Bias and confidence in not quite large samples. *The Annals of Mathematical Statistics*, 29:614–623, 1958.
23. 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.
24. 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.
25. X. Ye, T. Eisenbarth, and W. Martin. Bounded, yet Sufficient? How to Determine Whether Limited Side Channel Information Enables Key Recovery. In *CARDIS 2014*, volume 7707 of *LNCS*. Springer, 2014. to appear.

## A KRE optimisation

To allow the key rank algorithm of Martin et al. to run with a precision of up to 23 bits we had to include several implementation and algorithmic tricks to bring down the runtime of the algorithm.

### A.1 Distinguishing score to integer weight conversion

When the distinguishing scores are converted to integer weights they are done in such a way that the largest distinguishing score results in value $2^p$. However it is possible that this leads to scenarios where the distinguishing scores are unnecessarily large—for example, if all the distinguishing scores have value 1 they will end up with value $2^p$. To counter this we subtract the minimum integer score from all scores to scale them back. This increases the efficiency of the algorithm since the runtime is linear in the weight of the key.

### A.2 Recurrence relation

One of the major changes to the algorithm was adjust the recurrence relation. The first step was to use the 'wide sort' given in the original paper as it had the smallest memory footprint. Using a three-dimensional coordinate system to index the graph, the single loop over the graph was replaced with three for loops, one for each integer in the representation. Using the combination of the wide sort and the triple index system, it can be noted given $(x, y, z)$, such that the left child is not reject, it will always return $(x, y+1, z)$. This can be used to remove the majority of the memory copies and access by computing an entire partial weight within a subkey at once without having to work at an index at a time. The resulting algorithm is given in Figure A.2.

```
**Algorithm** KeyRank($m, n, W$):
**for** $j$ **from** $m-1$ **down to** $0$ **do**
  **for** $w$ **from** $0$ **up to** $W-1$ **do**
    **for** $i$ **from** $n-1$ **down to** $0$ **do**
      $Rank[w] \leftarrow Rank[w] + OldRank[RightChild(j, w, i)]$
    **end for**
  **end for**
  $OldRank \leftarrow Rank$
**end for**
**return** $Rank[0]$
```