IACR Communications in Cryptology
ISSN 3006-5496, Vol. 1, No. 3, 22 pages.

https://doi.org/10.62056/aytxl86bm

Key Rank Estimation Methods: Comparisons and Practical Considerations

Rebecca Hay[1] and Elisabeth Oswald[2,3]

[1] University of Bristol, Computer Science, Bristol, United Kingdom
[2] University of Klagenfurt, Digital Age Research Centre, Klagenfurt, Austria
[3] University of Birmingham, Computer Science, Birmingham, United Kingdom

Abstract. New proposals for scalable key rank estimation methods have appeared recently, in particular the sampling based approach MCRank. The idea is that one can consistently estimate the key rank by sampling only a small portion of the key space as a "proxy", leading to both an accurate and scalable approach, at least in comparison with another approach based on histograms. We show that the (earlier) GEEA algorithm is in fact a sampling based algorithm, and provide an in-depth comparison between GEEA (when adapted to produce rank estimates rather than guessing entropy estimates), GM bounds, MCRank and the currently most performant counting based rank estimation as implemented in the Labynkyr library. We find that although MCRank does live up to the promised accuracy and scalability for probability-based distinguishers, it fails to handle cases with unusual distinguisher distributions.
Furthermore, we put forward a novel proposal for a highly scalable key rank estimation method by introducing the notion of an "attacker budget". Our proposal is based on the idea that, in particular for very long keys, the exact key rank is less important than the knowledge whether a key is within a certain bound. Thus our "budget approach" is based on efficiently checking if the result of an attack is such that the attacker's budget suffices for successful enumeration. Our budget approach scales linearly with the key size and thus enables security estimations even for post-quantum key lengths.

Keywords: Key Rank Estimation · Large Keys · Key Security Assessments

1 Introduction

Side channel attacks reveal information about secret keys in the form of either scores or probabilities (i.e. normalised scores). Adversaries can trade off the effort put into an attack and the effort put into a post side-channel key enumeration. Consequently, it is relevant from designers' and evaluators' points of view to be able to quickly estimate the enumeration effort given the scores resulting from a side channel attack. This explains the interest in so-called key rank algorithms. Key ranking algorithms take the scores from an attack and the secret key as input, and return the number of keys that have a score higher than the secret key.

Key rank estimation is used routinely in the context of evaluating symmetric schemes, but they are also useful when considering asymmetric schemes. For instance, one may be interested in the information leakage when a secret key is loaded, or in the context of cold boot attacks [Pol19]. The versatility of key ranking to estimate the remaining effort of an adversary has lead to a considerable interest in *scalable* key rank estimation methods.

E-mail: rebecca.hay@bristol.ac.uk (Rebecca Hay), m.e.oswald@bham.ac.uk (Elisabeth Oswald)

Received: 2024-07-08 Accepted: 2024-09-02

After the initial publications by Veyrat-Charvillion et al. [VGRS12, VGS13], several faster and tighter key rank and key enumeration algorithms have been developed, by Glowacz et al. [GGP+15], Martin et al. [MOOS15], Bernstein et al. [BLvV15], Longo et al. [LMM+16] and Grosso [Gro18].

The approaches by Glowacz et al. [GGP+15] and Martin et al. [MOOS15] support key enumeration as well as key ranking, and it was later shown by Martin et al. [MMO18] that they are mathematically equivalent. Improvements to these methods appeared in work by Longo et al. [LMM+16],Martin et al. [MMOS16a] and Poussier et al. [PSG16]. An independent line of work by David and Wool [DW19b] (with further work [DW19a, DW22]), which is of comparable speed as the work of Glowacz et al. [GGP+15], but enables better bounds. Because single key rank estimations are of limited meaning, more ideas for the estimation of the expected key rank (aka "Guessing Entropy", short GE) were proposed by Zhang et al. [ZDF20], culminating in the GEEA algorithm.

In work by Young et al. [YMO22] GEEA was scrutinised and it was found that it delivers unreliable estimates, and its performance highly depends on the number of sampled keys. In the same paper, [YMO22] showed the path counting approach to estimate key ranks presented by Martin et al. [MOOS15], which is mathematically equivalent to histogram convolution [MMO18], leads to a very performant implementation that can scale up to key lengths of up to 4096 bits. Most recently "MCRank" by Camurati et al. [CDS23] appeared, which claims to be based on a novel idea of using sampling, and promises superior scalability and speed over the "state of the art".

1.1 Challenge: accuracy and scalability

The accuracy or tightness of a key rank algorithm is determined by how close the returned rank is to the true rank. There are two factors that impact the accuracy of key ranks. The first factor is the accuracy of the ranking algorithm itself, which may be determined by how much information is retained about the scores, or by the sample size that the estimate considers.

The second factor is due to variation of the distribution of the key rank. The key rank itself is a random variable, which depends on the score vectors. If the same attack is carried out multiple times (using exactly the same parameters, and the same key), each time a different set of scores will be produced, resulting in different key ranks. An early investigation into the rank distribution [MMOS16a] revealed that the distribution has a large variance, and thus to provide an accurate estimate of the key rank for a given attack vector, one must repeat the attack, derive the resulting key ranks and then consider how to best represent them (e.g by some average measure, or by some representative statistical plot).

Quantities like the average rank and the guessing entropy are the most popular methods to represent the outcomes of multiple key rank experiments. However, outputting averages can conceal the large variation in rank estimates, and thereby give a misleading representation of the quality of an approach. The use of descriptive statistics such as box plots can give a much more informative assessment of the accuracy of a ranking algorithm, as demonstrated in [YMO22].

The scalability of algorithms relates to their ability to deal with very long keys, and therefore very large key ranks. It was recently explored in [YMO22], and it was also the motivation for MCRank [CDS23]. Most existing key rank estimation methods are unable to scale for larger key sizes due to time or memory constraints. Assessing the leakage of very long keys, e.g. due to initial key loading operations (as pointed out in [CDS23]), will become increasingly problematic as the world shifts to dealing with post-quantum cryptographic keys where we must be able to assess the information loss in the context of very long keys.

1.2 Motivation

Many papers have appeared over the years claiming scalability to "very large keys", with the MCRank [CDS23] being the most recent proposal. The paper claims that it is based on a novel idea ("Monte Carlo sampling"), and that is better than the state of the art algorithms both in terms of scalability, accuracy and handling various types of distinguisher outcomes. Examining the paper closely reveals that neither the approach of the GEEA algorithm nor the path counting approach were considered for comparison. This leaves questions about the claimed novelty and superiority of the described approach. We conduct comparisons of various algorithms, not previously considered, where we find not all the claims hold. We focus on analysing how the algorithms handle score-based distributions in contrast to probability-based distributions, as well as situations where the subkeys are not independent. We propose a novel approach to vulnerability assessment using an attacker budget metric to work with very long keys and demonstrate how scalable some approaches might be when considering post-quantum key structures.

1.3 Outline of our contributions

Our baseline for comparison is the efficient key rank estimation approach that is available via the open source library Labynkyr [MMOS16b]. Labynkyr implements an optimised version of the path counting approach [MOOS15,LMM+16], which we will briefly summarise in Section 2. We also use this section to provide summaries of other counting and sampling based estimation approaches.

In Section 2.3, we show as our first contribution, that GEEA [ZDF20] and MCRank [CDS23] are in fact related: both papers describe a Monte Carlo sampling strategy to approximate the key rank (albeit GEEA estimates the guessing entropy as an average key rank metric). Thereafter in Section 3, we modify GEEA to estimate the rank rather than the guessing entropy (we call this algorithm REA) and show that the only difference between REA and MCRank is the sampling distribution. In Section 3, we also introduce our novel approach to vulnerability assessment using attacker budgets. In Section 4.1, we run experiments comparing path counting (using the Labynkyr implementation), MCRank (using their accompanying implementation that can be found on Github), REA and GM bounds as well as analysis of the accuracy of the Budget Approach. In Sections 4.2 and 4.3, we explore various types of distinguishing outcomes, looking at score-based distributions and dependency amongst subkeys. Finally, we investigate the scalability of these different approaches, firstly extending number of subkeys then moving to post-quantum key structures.

All experiments in this paper were conducted on a standard laptop with Intel I9-9880H (2.3GHz) mobile CPU with single threaded implementations.

2 Background

2.1 Counting based key rank estimation

In this section, we define the distinguishing vectors we obtain following a side channel attack and then how we can use these scores to calculate the rank of the secret key sk. An adversary with enumeration capability must make an additional effort proportional to the rank of sk when attempting to determine sk given likelihoods (or scores) from a side channel attack.

Definition 1 (Distinguishing Vector)**.** For each subkey i, the vector D_i is made up of scores $D_{k_i,i}$ corresponding to how likely subkey value k_i is to be the correct subkey value.

Definition 2 (Key Rank)**.** Given a matrix of distinguishing vectors $\mathbf{D} = (D_0, ..., D_{m-1})$ and secret key sk, the rank of sk is the number of keys k with a greater distinguishing

score (assuming in the following formula that we have additive scores).

$$rank_{sk}(\mathbf{D}) = |\{k = (k_0, .., k_{m-1}) : \sum_{i=0}^{m-1} D_{k_i,i} > \sum_{i=0}^{m-1} D_{sk_i,i}\}| \tag{1}$$

Thus a natural approach to calculating the rank of sk is to count all keys that are more likely to be the correct key than sk: which appears to be a challenging problem given typical sizes for the key space in contemporary cryptography. However, two algorithms emerged that were eventually shown to be mathematically equivalent solutions [MMO18] for counting possible key values: path counting [MOOS15] and histogram counting [GGP+15].

The idea behind the path counting approach is to count solutions to a knapsack problem, by creating a graph of potential solutions. Let W be the likelihood of the full key sk (based on the outcome of a side channel attack), and let m be the number of subkeys that an adversary has recovered. The path counting method constructs a graph with at most $m * W + 2$ nodes, and counts very efficiently how many paths lead from a designated start to a designated accept node of the graph. The paths corresponds to keys that are more likely than sk. The complexity of the resulting algorithm is independent of the distribution of the likelihoods, and the depth of the key; it depends on the number of subkeys, and the number of values of the subkeys. As a consequence it is possible to utilise various compiler optimisations by fixing these parameters at compile time.

The histogram counting method organises subkeys scores into histograms with a defined numbers of bins. Then the histograms are convolved, leading to a single large histogram representing the distribution of the full key scores. It is possible to locate the bin associated with the score of sk. Hence by counting the number of elements in all bins that are more likely than the bin associated with sk gives the rank of sk. The complexity of this counting method depends on the number of bins and the distribution of scores in the bins. Thus it is not independent of the distribution of the score and no easy complexity estimate can be given. Furthermore, because no general assumptions can be made about this distribution, a comparison between the efficiency of path counting and histogram counting must be based on experiments. This was done in [MMO18] focussing on the associated enumeration methods. Their comparison showed that for typical symmetric key sizes, both methods, when configured equivalently, were equally performant and mathematically equivalent for independent subkey scores that can be combined additively. But when the estimation precision was increased (both algorithms allow you to configure precision), or when key length was increased, then path counting was more performant.

2.2 GM Bounds

An alternative approach presented in [CP17] used formulae to calculate lower and upper bounds for the Massey Guessing Entropy metric, originally proposed in [Mas94]. We have translated the formulae to use same notation as seen throughout this work. Following a side channel attack, we assume the scores we obtain are probabilities. The approach requires these probabilities p_k, $k = 1, ..., n$ to be sorted from highest to lowest and then GM is defined by equation 2.2. In [CP17], it was highlighted that some distinguishers such as correlation power analysis return scores but it is possible to use statistical techniques to obtain pseudo-probabilities for use with these formulae.

$$GM = \sum_{k=1}^{n} k \cdot p_k \tag{2}$$

The following formulae can be used to calculate the full key lower and upper bounds for GM, for a key of m subkeys, where each subkey can take one of n possible values. In

this formula, p_k^i is the probability of the i-th subkey being value k and ln represents the natural log function.

$$\frac{1}{1+\ln n^m}\prod_{i=1}^{m}\left[\sum_{k=1}^{n}\sqrt{p_k^i}\right]^2 \leq GM^f \leq \frac{1}{2}\prod_{i=1}^{m}\left[\sum_{k=1}^{n}\sqrt{p_k^i}\right]^2 + \frac{1}{2} \tag{3}$$

Unlike the other approaches we are considering in this work, GM bounds do not require prior knowledge of the key, only the probabilities from the distinguisher. This approach could be used in situations when the key is unknown, such as by an attacker.

In [CP17], a Matlab implementation is provided (`https://gitlab.cs.pub.ro/marios.choudary/gmbounds`). This implementation provides both a direct computation for the formulae and symbolic execution alternative. We will use the Python wrapper for the Matlab implementation provided with the MCRank implementation [CDS23].

2.3 Sampling based key rank estimation

A common feature across several existing approaches is to start with the distribution of scores returned by a side channel attack. The score distributions for each (independent) subkey can be used to calculate the full distribution for distinguishing vectors via a convolution as observed by Glowacz et al. [GGP+15].

In 2020, the authors of [ZDF20] used the observation that the distributions for each individual subkey can be well described by a (multivariate) normal distribution. They also observe that rather than work with all scores for all subkey values, it is possible to *estimate the key rank by sampling from a subspace of the key space*. Both observations jointly then lead to the GEEA, which is based on computing the distribution of so-called comparison scores for each subkey, then sampling a uniformly random subset of the entire key space, and evaluating the distributions for the sampled keys in order to estimate multiple key ranks. This then turns into an estimate of the guessing entropy. Their method is a very classical application of "Monte Carlo sampling": they use a sample uniformly from the key space, test each of the sampled keys whether it is more likely than sk, and then use the proportion of more likely keys in their sample to estimate the number of more likely keys in the overall population.

Recently, the authors of [CDS23] explain the same principle. They also observe that it is possible to derive a distribution for the subkey scores. In contrast to [ZDF20] they don't estimate the parameters for probability density, but they compute the empirical cumulative distribution from a single experiment. Then, rather than working with the entire key space, they suggest to take a sample of the key space and estimate the rank by calculating the proportion of keys in the sample that are more likely than sk. Their sampling strategy picks keys from the score distribution (i.e. the distribution resulting from the side channel attack). In other words, they also use Monte Carlo sampling, but they do not draw keys from a uniform distribution, instead they draw it from the score distribution. They explain that this sampling strategy leads to a quicker convergence than uniform sample, and that on top of that, one could, or should, "calibrate" the score distributions, to improve the estimation accuracy further, and they provide a way of doing so. Their calibration method is supposed to be rank preserving and a closer look shows that their method is indeed one of the rank preserving score transformations that were proven in [MM18].

Clearly, both [CDS23] and [ZDF20] are based on the idea of Monte Carlo sampling, and the difference between them is how keys are sampled.

It should be noted that earlier work has also explored using sampling as part of the rank estimation process. This includes work by Bernstein et al. [BLvV15], who used sampling as a post-processing technique to extend the estimation algorithm provided

by Veyrat-Charvillon et al. [VGS13]. Like many other algorithms, the extended rank estimation algorithm also scales poorly to increasing key sizes.

3 Further sampling based approaches for key rank

We now illustrate our contributions by adapting GEEA for comprehensive comparison with other techniques and introducing a new approach for vulnerability assessments. This aims to provide a solution for key ranks greater than 2^{4096}, the point up to which the Labynkr implemenation of the path counting approach was identified as the fastest estimator [YMO22].

3.1 From GEEA to REA

To reasonably compare the techniques used in GEEA with MCRank we need to convert GEEA to return key rank estimates. We do this based on the following observations. GEEA works on comparison scores rather than the actual scores, and derives these for all key bytes independently. Comparison scores are calculated as the difference between a key byte score and the score for the corresponding correct key byte. We can then calculate the mean and variance (a Gaussian distribution is assumed) for the key byte score distribution. Comparison scores are defined in equation 4, whereby $s(\cdot)$ is a function returning the score vector, sk is the secret key and k is a key guess. We use $k_{i,j}$ to represent the jth possible value for subkey k_i.

$$\Delta_{i,j} = s(k_{i,j}) - s(sk_i) \text{ for } i \in \{1, \ldots, m\}, j \in \{1, \ldots, n\} \tag{4}$$

To derive an overall comparison score for a key guess k, the mean and variance μ_k and σ_k^2 for the score distribution respectively across the subkeys are simply added up. Because the secret key has a comparison score of 0, the probability that a key guess k returns a score greater than the secret key is given by equation 5 where $s(\cdot)$ returns the score vector and Φ_{μ_k,σ_k} is the normal distribution CDF with mean μ_k and standard deviation σ_k (Φ being the standard CDF with mean 0 and standard deviation 1). This equation gives the probability that the key guess k would be ranked higher than the secret key and should be counted towards the key rank estimate. GEEA calculates this probability for each key guess in a sample set M and rescales to the size of the key space $\mathbb{K}$ to estimate a key rank value, as shown by equation 6.

$$\mathbb{P}(s(k) > s(sk)) = 1 - \Phi_{\mu_k,\sigma_k}(0) = 1 - \Phi(\frac{-\mu_k}{\sigma_k}) \tag{5}$$

$$R = \frac{|\mathbb{K}|}{|M|} \sum_{i=0}^{|M|} 1 - \Phi(\frac{-\mu_k}{\sigma_k}) \tag{6}$$

Our rank adaptation of GEEA follows naturally from the observed working principle of GEEA, and we provide Algorithm 1 to show the resulting pseudo-code. We note that we provide x trace sets to allow for the estimation of mean and variance values for the comparative scores (denoted by $\widetilde{\mu}_j^i$ and $(\widetilde{\sigma}_{j,j}^i)^2$ for $i = 1, .., m$ and $j = 1, ..., n$) and we use notation s_l to indicate scores calculated using the trace set $l = 1, .., x$ and sk for the known secret key.

3.2 The Budget Approach

We now offer a new idea for vulnerability assessment in the context of very long keys by the following observation: the key rank relates to the precise position of the secret key

Algorithm 1: REA- (GEEA) based Rank Estimation Algorithm

Data: Score vectors $s_l(\cdot)$ for trace sets $l = 1, .., x$, Secret key sk, Secret key scores $s_l(sk_i)$ for $i = 1, ..m$ (sk_i can take values from $j = 1, ..., n$)
Result: Key rank estimate R

▷ Comparison Score Distributions

for $i \leftarrow 1$ **to** m **do**
  for $j \leftarrow 1$ **to** n **do**
    $\widetilde{\mu}^i_j = \frac{1}{x}\sum_{l=1}^{x}(s_l(k_{i,j}) - s_l(sk_i))$;
    $(\widetilde{\sigma}^i_{j,j})^2 = \frac{1}{x}\sum_{l=1}^{x}(s_l(k_{i,j}) - s_l(sk_i) - \widetilde{\mu}^i_j)^2$;
  end
end

▷ Estimate Rank

$M = (k_1, ..., k_{|M|}) \in \mathbb{K} \setminus \{sk\}$;
for $k \in M$ **do**
  $\widetilde{\mu}_k = \sum_{i=1}^{m} \widetilde{\mu}^i_{k_{i,j}}$;
  $(\widetilde{\sigma}_k)^2 = \sum_{j=1}^{x}(\widetilde{\sigma}^i_{k_{i,j}})^2$;
end
$R = \frac{|\mathbb{K}|}{|M|}\sum_{k=0}^{|M|} 1 - \Phi(\frac{\widetilde{\mu}_k}{\widetilde{\sigma}_k})$;
return R;

among all keys ordered by their likelihood. But what if rather than asking for the exact position of a specific key, we ask if a specific key is likely to be below a given "budget"? In other words, we suggest to ask the fundamental question: "Is sk vulnerable to an attacker using biased brute force search with a budget of B?" We now go on to formalise this idea, and design a corresponding algorithm.

Definition 3 (Attacker Budget)**.** The attacker budget, denoted B, is the number of keys an attacker can realistically enumerate.

We have chosen to keep the definition of the attacker budget broad on purpose. It is designed to be defined by the user and be a parameter that can incorporate various different measures affecting an attacker's abilities, for example run time or memory capacity.

Definition 4 (Key Vulnerability Condition)**.** We say that a key k is vulnerable to attacker with a budget B if k falls within the first B keys and secure from this attacker otherwise.

The attacker budget B by definition is the number of keys our attacker can enumerate. It can be directly mapped to a proportion of the key space that is vulnerable. The distribution of the key rank is thought to be a truncated normal distribution with density $F_X(x)$ [MMOS16a]. The truncation is due to the fact there are only a finite number of keys and there is at least one lowest score and one highest score. There may be multiple keys with the same score but there remains a limited number of possible score values.

Thus we can define the budget problem in terms of the cumulative distribution function (CDF) of a truncated normal distribution $F_X(x)$, truncated by the minimum and maximum possible score values. The CDF function $\Phi_{\widetilde{\mu},\widetilde{\sigma^2}}$ for estimated mean $\widetilde{\mu}$ and variance $\widetilde{\sigma}^2$ gives us $\mathbb{P}(X \leq x)$ for random variable X and we need to calculate the minimum score b for a key guess that would fall within this vulnerable space, $\mathbb{P}(X > b) = p$ where p is the probability of a key being in the budget ($p = 1 - \frac{B}{|\mathbb{K}|}$ for key space $\mathbb{K}$). We can use the truncated inverse CDF $\Phi^{-1}_{\widetilde{\mu},\widetilde{\sigma^2}}$ [Bur14] shown in equation 7, with l and u representing the lower and upper limits for truncation, to find b. The value b corresponds to the score for the last key found using biased brute force search and we can use this position to indicate

which category our secret key falls in. If this value is negative then the secret key must fall to the right of this key (a lower score) and therefore be in the vulnerable key space. If b is positive, the secret key falls to the left and is secure from this attacker.

$$b = \Phi^{-1}_{\widetilde{\mu},\widetilde{\sigma^2}}(\Phi_{\widetilde{\mu},\widetilde{\sigma^2}}(l) + p \cdot (\Phi_{\widetilde{\mu},\widetilde{\sigma^2}}(u) - \Phi_{\widetilde{\mu},\widetilde{\sigma^2}}(l))) \tag{7}$$

How to choose B? In 2016, [LMM+16] demonstrated 2^{40} keys could be easily enumerated in three days on some University computing infrastructure, and they conclude that 2^{50} would not be infeasible for some attackers with increased computational capability. From a cryptographic point of view, symmetric primitives are often required to at least need 2^{80} or 2^{100} in terms of brute force effort.

3.2.1 Budget Assessments

We now explain how to translate the budget idea into an efficient algorithm.

Firstly, recall that by repeatedly convolving subkey distributions, the resulting sum of distributions tends towards a normal distribution. Therefore we can estimate the parameters of the full score distribution in terms of the sum of the empirical means and variances of the subkey distributions. For our approach it will also be useful to include two additional values, the minimum and maximum scores.

Secondly, we use comparison scores (from [ZDF20]) as a basis for estimating the full key distribution. The formula for comparison scores was defined earlier in equation 4 with $s(\cdot)$ as the function returning the score vector, sk as the secret key and k as a key guess.

Combing these two ideas, we calculate the histograms from the comparison scores for subkey candidates, and then we generate an approximate normal distribution reflective of the full key space by estimating the mean and variance of the subkey distributions denoted as $\widetilde{\mu_i}$ and $(\widetilde{\sigma_i})^2$ for $i = 1, .., m$. By using comparison scores (relative to the secret key scores), the secret key is always at position 0 and this will be important for comparison with our attacker budget. We calculate the score corresponding the position of the budget as this corresponds to the minimum score for a key to fall in the vulnerable range, and then compare this score with the one linked to our secret key. It is important to note this approach differs from other key rank methods using distributions as we are not sampling and instead use characteristics of the entire key space.

Due to the precision required for calculations of b using the truncated inverse CDF, we need to make use of high-precision libraries (like MCRank does). In Python, we are limited to the IEEE 754 standard doubles of 53 bit precision and this is not sufficient for our implementation of Algorithm 2. We therefore used the mpmath library [mdt23], which has built-in functions for some statistical calculations. However, there is no built-in function in the library for inverse CDF so we are required to use the relationship between the normal distribution CDF and the Error function erf.

$$\Phi(z) = \frac{1}{2}(1 + erf(\frac{z}{\sqrt{2}})) \tag{8}$$

This can be adapted for normal distributions with mean μ and variance σ^2 using the formula $z = \frac{x-\mu}{\sigma}$:

$$\Phi(x, \mu, \sigma) = \frac{1}{2}(1 + erf(\frac{x-\mu}{\sigma\sqrt{2}})) \tag{9}$$

The idea of introducing a "budget" and asking whether or not a given secret key is within this budget bears resemblance to the previous work by Ye et al. [YEM14], in which a key enumeration method, known as the Weak Maximum Likelihood (ML) approach, was introduced. This previous work introduced an "effort vector" to determine the amount of

Algorithm 2: Budget Assessment Algorithm

Data: Score vector $s(\cdot)$, Secret key sk, Secret key scores $s(sk_i)$ for $i = 1, .., m$ (sk_i can take values from $j = 1, ..., n$)
Result: "Vulnerable" or "Secure" assessment of sk

```
for i ← 1 to m do
    ▷ Comparison Scores
    for j ← 1 to n do
        Δ_{i,j} = s(k_{i,j}) − s(sk_i);
    end
end
for i ← 1 to m do
    ▷ Keyspace Distribution
    μ̃_i = (1/n) Σ_{j=1}^{n} Δ_{i,j};
    (σ̃_i)^2 = (1/n) Σ_{j=1}^{n} (Δ_{i,j} − μ̃_i)^2;
end
μ̃ = Σ_{i=1}^{m} μ̃_i;
(σ̃)^2 = Σ_{i=1}^{m} (σ̃_i)^2;
l = Σ_{i=1}^{m} min(Δ_i);
u = Σ_{i=1}^{m} max(Δ_i);
p = 1 − B/|𝕂|;
    ▷ Calculate b
b = Φ^{-1}_{μ̃,σ̃^2}(Φ_{μ̃,σ̃^2}(l) + p · (Φ_{μ̃,σ̃^2}(u) − Φ_{μ̃,σ̃^2}(l)));
if b ≤ 0 then
    ▷ Final Assessment
    return "Vulnerable";
else
    return "Secure";
end
```

key space considered in relation to a maximum "cost". This idea here sounds similar to introducing a "budget", but it is fundamentally very different to our work.

The weak ML-algorithm of [YEM14] sets a cost, which can be considered similar to our budget, and then finds the optimal effort distribution vector in order to enumerate keys within the cost. The effort distribution vector decides how many values of each subkey should be considered for enumeration (= counting) to maximise the chance the correct key is found. This idea has merit if and only if traditional ranking is not possible, and if the distribution of the subkey scores is very different from subkey to subkey.

Our budget algorithm does not require discarding of subkey scores to facilitate rank estimation; the budget algorithm still takes the entire score distribution into account.

4 Experiments and Analysis

4.1 Comparing Estimation Accuracy and Speed

We investigated how recent approaches such as MCRank (with and without rescaling) compare with other techniques, including path counting, GM bounds and GEEA (converted to REA) in a typical key rank scenario. Hence we conduct simulation experiments: we use AES SubBytes as the target function, Hamming weight leakage as a leakage model and conduct template attacks to obtain scores that we can make additive to feed into the various algorithms. We use 10,000 traces to build our templates and 100 traces for the

attack then vary the signal-to-noise ratio (SNR) implying that the secret key is ranked at different depths for the different SNRs. To produce ranks that go across an entire 128 bit key space, we use SNR values from 0.001 to 0.5. Each experiment is repeated 100 times.

MCRank and REA both require us to choose the sample size: for a fair comparison we use the recommended 50,000 samples suggested by Camurati et al. [CDS23]. We note at this point that when introducing GEEA [ZDF20], Zhang et al. noticed the critical role that this sample size plays, and recommended a much larger sample size. However, in their experiments for GEEA they used 10k samples for performance reasons, thus 50k is generous in comparison. We show the results of this experiment in Figures 1a (estimation accuracy) and 1b (run time). We have also plotted the results for the lower and upper bounds found using Massey Guessing Entropy formulae [CP17]. These bounds are calculated by using the GM equations on the ordered probabilities from the distinguisher. Throughout our experiments, we represent path counting in blue (always using the Labynkyr implementation), MCRank in shades of green (with rescaling in light green), GEEA/REA in pink and GM bounds, shown in shades of orange.

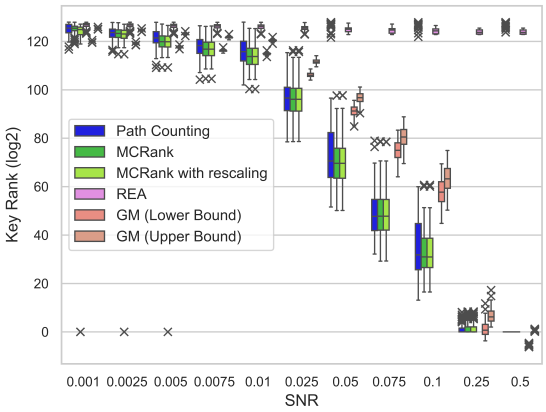
We shall discuss the estimation accuracy first. Rather than plotting average ranks we use box plots, initially introduced and defined in [Tuk77, HS16] to show the outcomes of the 100 experiments (per SNR). A box plot, informally speaking, consists of a "five number summary". It is shown as a box and "whiskers". The box shows where most of the data sits. The middle line of the box is the median, and the boundaries of the box are the first and third quartile[1]. The whiskers show the first and third quartile plus 1.5 of the inter-quartile range. Any additional points are outliers. A box plot thus is a very comprehensive visual summary of the outcomes of repeat experiments. The larger the box, the more variable the outcomes are. Given that all rank estimation algorithms get the same score vectors as inputs, a larger box represents greater variation in estimations.

We can see that the Labynkyr implementation of path counting and MCRank (with and without rescaling) produce similar results across all SNR values, in fact we see that rescaling had no impact on the key rank, producing almost identical box plots. We observe that the GM bounds frequently overestimates the key rank, producing results much higher than path counting (using Labynkyr) or the MCRank algorithms. Using GM bounds during security evaluations could result in systems being deemed secure when they are likely vulnerable to an attacker. It can also be seen that the GM bound calculations can produce results outside the range of possible values, i.e. above 2^{128} and below 0.

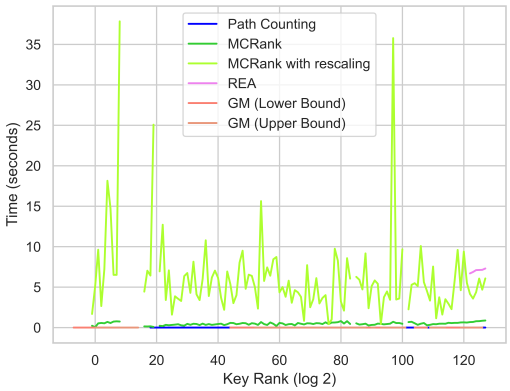
REA only produces results over 2^{120} for each SNR value, which does not reflect the expected trend of key ranks seen with the other estimation algorithms. It is worth highlighting that REA requires more than one set of traces and subsequently more than one set of scores for the algorithm. To compare the results from REA with the other techniques, we have assumed we also only have 100 traces for the attack stage, and these must be split between a minimum of 2 sets (with 50 traces in each). Therefore, it is not unexpected that the results are not comparable as we are working with considerably less information. In order to demonstrate the impact of the multiple score sets, we conducted experiments using 100 traces for each score set (taking a total of 200 traces) for comparison. These results are shown in Figure 2 and highlight that increasing the number of traces per set will produce key ranks that follow the expected trend. We see that for 100 trace sets, the key rank estimation drops to 0 for high SNR values and we struggle to estimate mid-range key rank values. Assuming this trend continues, to achieve estimates of the same quality as Labynkyr or MCRank we would need considerably more traces per set. We would argue that REA can not be safely used to estimate key ranks in this "standard scenario".

This brings us to examining the run time graph shown in Figure 1b. It shows the average execution time for the ranking experiments where the algorithms produce meaningful

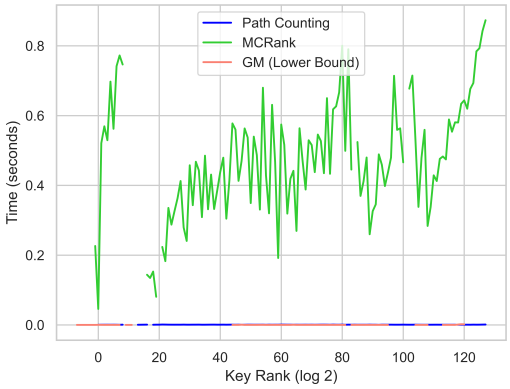
[1] Quartiles divide the data set into four equal portions.



(a) Estimation accuracy



(b) Average run times



(c) Average run times (zoomed to 0-1 seconds)

Figure 1: Estimations for AES-128 bit keys

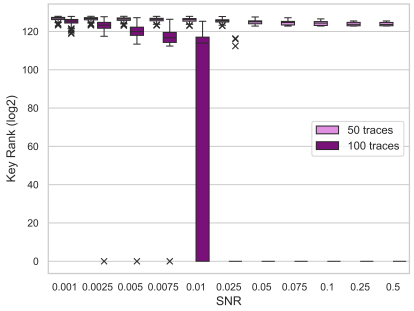


Figure 2: REA: 50 traces vs 100 traces

results (not all 0s or covering the full key space). In terms of average times across our experiments, we can see that path counting completes in around 0.01 seconds, MCRank without rescaling takes 0.05 seconds and with rescaling this average time is around 20 seconds. The MCRank approach with rescaling has a wide range of run times even at similar key rank values as the extent of resampling required is dependent on the randomly selected samples. As seen in Figure 1a, MCRank produces similar results to path counting (using Labynkyr) but the time difference is clear in Figure 1c, a zoomed in window of results under 1 second. We remark at this point that, as visible from the box plots, REA only produces estimates at the higher end of the rank spectrum and therefore we only have average times in this range (around 6-7 seconds). This is why we do not include performance results in the lower end of the key space for REA in Figure 1b, as for low key ranks we do not have accurate results for this approach. As the GM bounds are calculated using a fixed equation, it can be calculated in constant time. From the figure, we can see that this is equivalent to the time for path counting using Labynkyr.

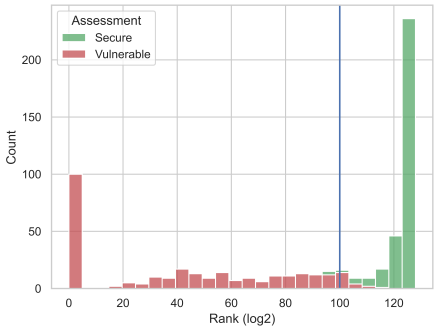
As this point, we can conclude that path counting and MCRank (with or without resampling) are the best approaches to consider further, based on accuracy and efficiency.

4.1.1 Comparison of budget approach

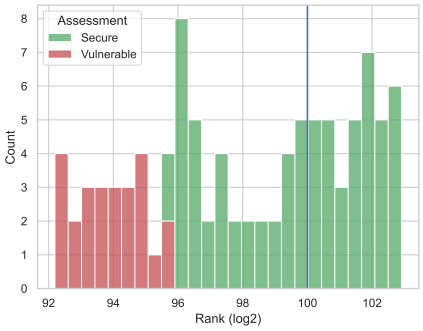
We compare our new approach with ranks estimated using Labynkyr implementation of path counting. As B is a user-defined parameter, it can be set to any value. For the following experiments we are using the biggest feasible enumeration range suggested by cryptographic intuition and set $B = 2^{100}$. We utilise the same simulation setup as before.

First, we check the accuracy of the budget approach by checking how often it produces a correct assessment for the canonical 128 bit key space situation: this means that the algorithm should return "Vulnerable" for keys with rank lower than 2^{100} and "Secure" for keys beyond this bound. Figure 3a shows the outcome, and we can see that only for keys just above the bound, the algorithm does return "Secure" instead of "Vulnerable" in a small number of cases. Running experiments focussed on keys around the budget (2^{100}), we see the cases that return "Secure" instead of "Vulnerable" are within 5 bits of the budget mark, as shown in Figure 3b.

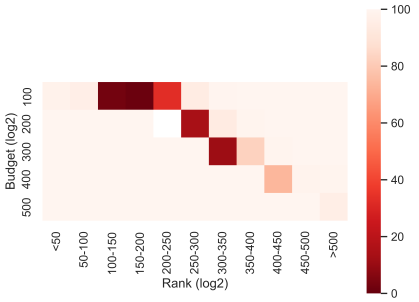
To assess the accuracy of the budget on larger keys size, we have run experiments on 512 bit keys looking at a range of possible budget values from 2^{100} to 2^{500}. The heatmap in Figure 3c shows the accuracy of the assessments, with lighter colours indicated greater success rates. The accuracy for each square is calculated based on experiments on 200 keys within a 50 bit rank range for a particular budget. The greatest number of errors in the experiments occur just above the budget value and due to the precision constraints in the Python implementation, we observe a greater number of errors for smaller budgets.



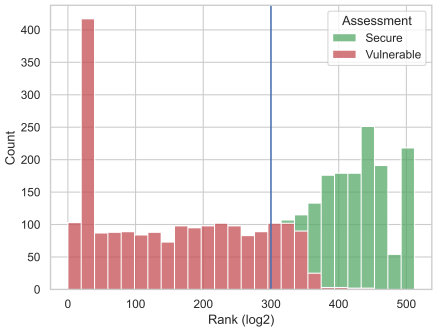
(a) 128 bit key assessments - Budget = 2^{100}



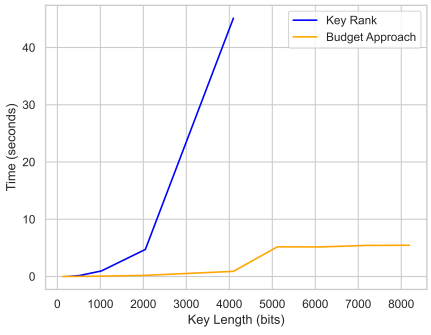
(b) 128 bit key assessments - ranks 2^{90} to 2^{110}



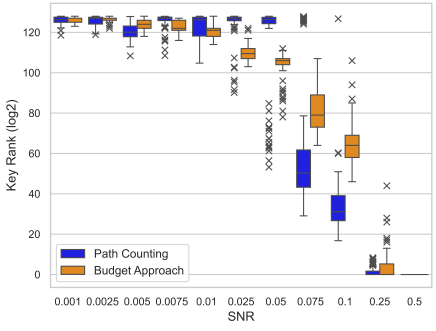
(c) 512 bit key assessments - Accuracy



(d) 512 bit key assessments - Budget = 2^{300}



(e) Average time



(f) Comparison of Path Counting and Budget Approach

Figure 3: Assessing the budget approach

Figure 3d shows the assessments for 512 bit keys with a budget set to 2^{300}. As with 128 bit keys, we get some inaccurate results around the budget mark.

Finally we consider the run time of the budget approach in comparison with path counting using Labynkyr. The budget algorithm calculations are not computationally linked to the position of the key and we therefore see consistent times per assessment regardless of key rank value. By design, the budget algorithm only factors in key size when generating the subkey histograms and finding the full key space normal distribution. This means as the key size grows there is a limited impact on the time per assessment. Figure 3e shows the average time per assessment of the budget approach compared to each estimation results from Labynkyr implementation for various key lengths. As key rank increases, the cost of the budget approach remains minimal. As a result, for longer keys, we can make security assessments much quicker than by using Labynkyr. This quick approach allows for the possibility of running repeated assessments with different budget parameters to identify bounds for the key rank, without needing to keep regenerating the initial distribution.

The main problem is the proportion of the key space considered vulnerable in relation to the overall key space. With the mpmath library, we can continually increase the value of the mpmath.dps parameter as needed to make assessments (i.e. we can adjust this to find small budgets even in very large key spaces if necessary).

It should be noted that we can use the budget approach to estimate key rank values using binary search but its primary function is for efficient vulnerability assessments against user-defined attackers by evaluation labs. As we are asking a slightly different question, it is difficult to make a direct comparison with estimation algorithms. Figure 3f shows box plots for path counting using Labynkyr and the Budget approach at the point our assessments change from secure to vulnerable. To calculate this, we deployed the budget approach for multiple budget values, starting at our path counting result (taken log 2) - 20 and increasing until our assessment no longer reads secure (within 5 bits). This means that our key must then be within the capability of the attacker.

Although in Figure 3b, we observe an overestimation of the key security, Figure 3d we see the opposite. Figure 3f shows cases of both over- and underestimation depending on the SNR value. It is worth noting that the budget approach is based on estimating the distribution parameters and we are using unbiased estimators for both the mean and variance.

4.2 Arbitrary Scores

As part of our comparison of the various key rank estimation algorithms, we investigated the handling of different distinguishing vector distributions. In [CPS16], the authors highlighted that when the distinguisher returns probabilities, we have known combination rules (in this case, summation) that can be used to build full key distributions but this is not the case for heuristic scores, such as those obtained from correlation power analysis. In their work, they make use of statistical techniques such as Bayesian extensions or linear regression with aim of transforming the score distribution into a probability-based one.

It is easily proven that for independent random variables with normal distributions, the distribution of the summation will also be normal [HMC05] so for our probability distinguishers that follow a normal distribution, we can easily demonstrate that summation over subkey distributions will return another normal distribution for the full key space. Analysis of the distribution for correlation coefficients found that it can be expressed by hypergeometric distribution [KK51] but there is no evidence to suggest we can easily apply the same summation rules in this case.

Figure 4 shows the results of running path counting using Labynkyr and MCRank (with and without rescaling) after using correlation power analysis attacks to generate our scores. We can see that path counting is unaffected by the different distinguisher distribution

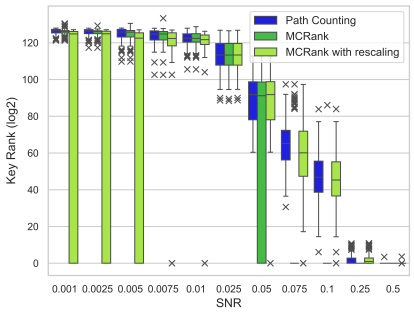


Figure 4: Correlation Power Analysis

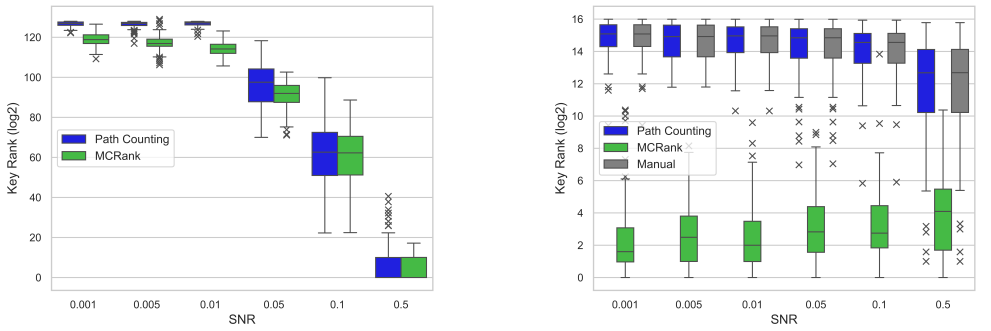
and produces similar results to those seen from the template attacks in Figure 1a. The path counting approach does not use any knowledge of the distribution and scores can be supplied from any distribution for the conversion to weights. The knapsack problem that fundamentally underpins the algorithm is deployed over the converted weights rather than the original distinguishing vectors. MCRank on the other hand uses the distribution of the distinguishing vectors and the combination rules between the subkey distribution play a crucial role. In the cases where these rules are not easily definable, we see greater variance in the accuracy of our results. In Figure 4, we see that at SNR value of 0.05, MCRank (without rescaling) returns potential rank values covering the majority of the key space and higher SNR values result in rank values of 0 with high probability. At low SNRs, using rescaling produces results across the entire keyspace but for the higher SNR values, where original MCRank struggles, rescaling greatly improves the accuracy. As seen in Section 4.1, however this comes at considerable cost.

4.3 Dependent Distinguisher Outcomes

As mentioned in the previous section, we have a statistical understanding of how to combine independent random variables with normal distributions. Therefore, assuming independent distinguishing vectors producing probabilities for our subkeys, we can easily map out the distribution for the full key space and subsequently use the various algorithms for rank estimation. It is possible, however, that our subkey distributions may not be entirely independent. When collecting trace readings from our target device, there may be a processing phase to smooth traces and remove unnecessary noise. This could include some level of filtering defined over a small interval of the x-axis across the trace. The resulting trace would then have some dependency between the subkey distributions and as with the score-based distinguisher case, we have no knowledge of how to successfully combine these distributions.

We conducted experiments to highlight the impact this dependency can have on estimation algorithms, particularly MCRank that relies on the successful combination of subkey distributions. Figure 5a shows the accuracy results for 128 bit keys where we have assumed dependency within the trace across rolling windows of 4 subkeys. It is clear that both path counting and MCRank can handle a small level of dependency within the traces.

If we increase the level of dependency between the subkeys, we question whether the algorithms can still produce reasonable estimations. In order to have a ground truth measurement, we have restricted our experiments to only 2 subkeys (16 bit keys), allowing us to calculate a rank value manually. Figure 5b shows the results from experiments where the scores are generated relying on trace information from across both subkey values, creating some level of dependency between the subkey distributions. As there is some



(a) 4 subkey distributions (128 bit keys)

(b) 2 subkey distributions (16 bit keys)

Figure 5: Dependency between subkeys

discrepancy in the results between the two approaches, we use the box plots for our manual calculations as the ground truth and can clearly see that path counting returns almost the same results across all SNR values. MCRank, on the hand, struggles in this more extreme case, producing very low rank values.

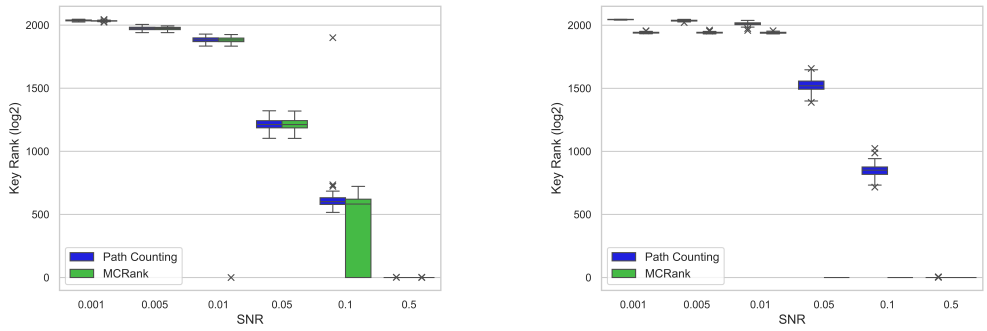
4.4 Going Post-Quantum

To further compare path counting and MCRank, we now investigate how well the two approaches perform in the case of longer keys. To do so we can only run MCRank without rescaling: with rescaling, as we have seen in previously, its run time is simply too slow. REA is also too slow for long keys and GM does not give us the same level of accuracy.

We conducted 100 independent experiments for different 2048 bit keys with varying SNR values to produce data that puts the correct key at different depths of the 2048 bit key space. Figure 6a and 6b show the accuracy results following template and CPA attacks respectively. As in previous sections, we utilise box plots to visualise the results, and we consider the path counting results as the best available ground truth. For template attacks, we can see that the two approaches produce very similar results across the SNR values. We see that for SNR 0.1, MCRank does have a greater variation in results, mostly overestimating the ability of an adversary. In the case of correlation scores, with just 50,000 samples, MCRank appears to struggle estimating mid-range key ranks. For instance, at SNR 0.05 it consistently returns as key rank estimate the rank 0 instead of around 2^{1500}. Similar to what we observed before, MCRank's overestimates are more drastic when applied to correlation scores. We can see that the impact of score-based distributions, seen in Section 4.2, only increases with longer key sizes.

All of our experiments up until this point have been conducted as a proof-of-concept that we could extend key rank estimation algorithms to handle longer key sizes and post-quantum keys. We have simulated longer keys by increasing the number of subkeys but assuming a continuation of the AES key structure, with byte-length subkeys. To fully demonstrate the potential of these algorithms for post-quantum key security evaluation, we need to adapt them to handle different key structures.

To compare path counting with the other approaches, we will need to adapt the Labynkyr implementation. It is worth highlighting that the path counting algorithm can be easily generalised as fundamentally we are applying the knapsack problem across a weight table of size m subkeys by n potential values, with the maximum weight determined by the sum of weights for the correct secret subkey values. We can easily change the values



(a) Template Attacks

(b) Correlation Power Analysis

Figure 6: Estimations for 2048 bit keys

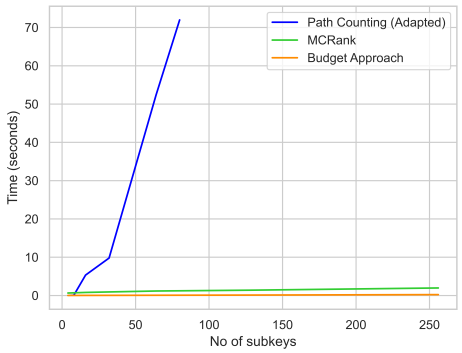


Figure 7: Average time for key rank estimations for Kyber keys

of m and n, and the size of the subkeys does not impact the logistics of the algorithm. Likewise, neither of the distribution based approaches used for MCRank or the Budget approach make assumptions about the subkey sizes and can easily be applied to any key structure as long as we can divide into subkey chunks and combine scores in a defined way.

We will use the lattice-based cryptosystem Kyber-768 as an example. The shared secret keys can be broken down into 256 subkeys with values taken mod q=3329. This means that each subkey can no longer be expressed by a single byte, we need 12 bits. We adapted the Labynkyr implementation so the user can set the size of each subkey and the functions to read from files will process the keys accordingly. As we are assuming 12 bits for our example key, we set n to be 4096, the maximum value we can represent using 12 bits but also ensure our scores distribution eliminates values $> q$ by making these weights much greater than the secret key total weight (and thus they will never be considered).

For MCRank and the Budget approach, the only adaptations needed are setting the user provided parameters for the number of subkeys ($m = 256$) and possible values each subkey can take ($sk_i \in \{0, .., q-1\}, q = 3329$).

Figure 7 shows the average time for estimations in our proof-of-concept experiments, conducted on keys of increasing size, with subkeys taking values mod q. Our experiments focus on the computational aspect of the different approaches. As an initial proof-of-concept, we chose probabilities at random to generate the distinguisher vectors. Although this does not reflect the expected distribution for Kyber keys, it shows a worst-case

scenario.[2] We can see that MCRank can estimate rank values for keys up to 256 subkeys in length within a couple of seconds, while the path counting approach (using adapted Labynkyr) takes considerably longer and in fact due to memory constraints is unable to handle more than 80 subkeys. We can also see that we can also apply the Budget approach to Kyber keys of 256 subkeys in seconds.

5 Application to Practice

In practice, the best method to use for key rank estimation depends on four factors: the length of the key (or size of the key space), the size of the subkey chunks, the amount of dependency between these subkey chunks and the type of scores resulting from the side channel attack. We now discuss these factors in turn.

For shorter keys (i.e. less than 2048 bits in length), split into independent chunks, our results show that path counting using Labynkyr and MCRank both achieve accurate and efficient results assuming that they distinguishing scores resemble probabilities. If the side-channel attack returns score-based distinguishing vectors (e.g. as a result of a correlation attack), path counting (which does not require probabilities) performed consistently in our experiments, whereas MCRank did not.

When considering dependent subkeys, there are two situations: either there is dependency as a result of the key recovery process or as a result of the leakage characteristics of the device. The former case was explored by Camurati et al. [CDS23] when considering how to approach key rank for AES-256. The latter case, which we explore in this work, demonstrated that path counting implementation appeared unaffected by processing dependencies and although MCRank was able to handle small levels of dependencies, increasing the levels had an impact on the accuracy of estimations. We consider this as an interesting avenue for further work.

As the key space grows towards 2^{4096}, path counting via Labynkyr remains efficient for byte-sized subkeys. However, in the context of post-quantum cryptography, subkeys are larger and represent modular numbers. Our modifications to Labynkyr enable processing such types of keys, however at suboptimal performance, and further work is necessary to implement a version of path counting that can tackle such keys. MCRank is unaffected by changing the key structure, thus at the moment, would be the better option, assuming subkeys are independent. For even longer keys, the Budget approach introduced in this work becomes more applicable and provides efficient key security assessments.

6 Conclusion

We compared existing approaches for key rank estimation designed for use with large key sizes. By adapting the GEEA algorithm to return key rank estimates rather than guessing entropy, we were able to compare it to the most recent work, MCRank, as well as path counting, using the fast Labynkyr implementation, and GM bounds. We showed that path counting and MCRank produce very similar results when using distinguishing probabilities from template attacks. The other approaches do not have the same level of accuracy in comparison, despite the efficiency of the GM bounds calculation. Using resampling with MCRank provided little to no benefit, despite the increase in computational time and we chose to focus on the original version of the algorithm.

We presented a novel proposal for security assessments of keys by rephrasing the question away from key rank estimates to whether the key falls within an attacker's budget.

[2] The efficiency of the rank estimation algorithm is impacted by the number of subkeys and the size of the subkey chunks, rather than the distribution of the subkeys. In our case, we ensure we use the correct subkey parameters for Kyber.

In our formal definition of this new metric, we allowed the budget to be determined by the user and therefore it could be defined in relation to various factors that could affect an attacker. We designed an algorithm that can efficiently provide assessments for keys and could scale to handle keys of at least 8192 bits in length, much longer than the previous top limit of the path counting approach. Although the budget approach has a margin of error around the chosen budget bound, we found that a certain "safety margin" can be set. For instance, in our experiments we found that if the actual budget is 2^{80}, then selecting 2^{100} as a bound implies that keys are classified correctly with near certainty.

We analysed the impact on accuracy of using score-based distributions and distributions with some level of dependency. In these cases, we have no concrete knowledge on how to successfully combine distributions and this affects the accuracy of results obtained from approaches that rely on such knowledge, such as MCRank. The results demonstrated that path counting is the best algorithm to use in situations where the distribution is not ideal, although resampling does provide some improvement when using correlation scores. We also explored using path counting and MCRank for longer key sizes, firstly considering 2048 bit keys following an AES subkey structure and then using Kyber-768 as an example to explore post-quantum cryptography cases. We adapted the Labynkyr implementation for use on key structures with larger than byte-sized subkeys and then demonstrated it was possible to estimate Kyber keys up to 80 subkeys in length before encountering memory constraints. MCRank allowed us to surpass this and reach 256 subkeys more efficiently, indicated this would be the better algorithm for longer key sizes, but only when the distinguishing vector distributions meet the ideal criteria.

Acknowledgments

Elisabeth Oswald has been supported in part by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement number 725042). Rebecca Hay has been funded by an NCSC studentship. Both authors would like to thank the anonymous reviewers for their constructive comments.

References

[BLvV15] Daniel J. Bernstein, Tanja Lange, and Christine van Vredendaal. Tighter, faster, simpler side-channel security evaluations beyond computing power. *IACR Cryptology ePrint Archive*, 2015:221, 2015. URL: http://eprint.iacr.org/2015/221.

[Bur14] John Burkardt. The truncated normal distribution, 2014.

[CDS23] Giovanni Camurati, Matteo Dell'Amico, and François-Xavier Standaert. Mcrank: Monte carlo key rank estimation for side-channel security evaluations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(1):277–300, 2023. URL: https://doi.org/10.46586/tches.v2023.i1.277-300, doi:10.46586/TCHES.V2023.I1.277-300.

[CP17] Marios O. Choudary and Pantelimon George Popescu. Back to massey: Impressively fast, scalable and tight security evaluation tools. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*, pages 367–386. Springer, 2017. doi:10.1007/978-3-319-66787-4_18.

[CPS16] Marios O. Choudary, Romain Poussier, and François-Xavier Standaert. Score-based vs. probability-based enumeration - A cautionary note. In Orr Dunkelman and Somitra Kumar Sanadhya, editors, *Progress in Cryptology - INDOCRYPT 2016 - 17th International Conference on Cryptology in India, Kolkata, India, December 11-14, 2016, Proceedings*, volume 10095 of *Lecture Notes in Computer Science*, pages 137–152, 2016. doi:10.1007/978-3-319-49890-4_8.

[DW19a] Liron David and Avishai Wool. Fast analytical rank estimation. In Ilia Polian and Marc Stöttinger, editors, *Constructive Side-Channel Analysis and Secure Design - 10th International Workshop, COSADE 2019, Darmstadt, Germany, April 3-5, 2019, Proceedings*, volume 11421 of *Lecture Notes in Computer Science*, pages 168–190. Springer, 2019. doi:10.1007/978-3-030-16350-1_10.

[DW19b] Liron David and Avishai Wool. Poly-logarithmic side channel rank estimation via exponential sampling. In Mitsuru Matsui, editor, *Topics in Cryptology - CT-RSA 2019 - The Cryptographers' Track at the RSA Conference 2019, San Francisco, CA, USA, March 4-8, 2019, Proceedings*, volume 11405 of *Lecture Notes in Computer Science*, pages 330–349. Springer, 2019. doi:10.1007/978-3-030-12612-4_17.

[DW22] Liron David and Avishai Wool. Rank estimation with bounded error via exponential sampling. *J. Cryptogr. Eng.*, 12(2):151–168, 2022. doi:10.1007/s13389-021-00269-4.

[GGP+15] Cezary Glowacz, Vincent Grosso, Romain Poussier, Joachim Schüth, and François-Xavier Standaert. Simpler and more efficient rank estimation for side-channel security assessment. In Gregor Leander, editor, *Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers*, volume 9054 of *Lecture Notes in Computer Science*, pages 117–129. Springer, 2015. doi:10.1007/978-3-662-48116-5_6.

[Gro18] Vincent Grosso. Scalable key rank estimation (and key enumeration) algorithm for large keys. In Begül Bilgin and Jean-Bernard Fischer, editors, *Smart Card Research and Advanced Applications, 17th International Conference, CARDIS 2018, Montpellier, France, November 12-14, 2018, Revised Selected Papers*, volume 11389 of *Lecture Notes in Computer Science*, pages 80–94. Springer, 2018. doi:10.1007/978-3-030-15462-2_6.

[HMC05] R.V. Hogg, J.W. McKean, and A.T. Craig. *Introduction to Mathematical Statistics*. Pearson education international. Pearson Education, 2005. URL: https://books.google.co.uk/books?id=vIEZAQAAIAAJ.

[HS16] Christian Heumann and Micheal Schomaker Shalabh. *Introduction to statistics and data analysis*. Springer, 2016. doi:10.1007/978-3-031-11833-3.

[KK51] J.F. Kenney and E.S. Keeping. *Mathematics of Statistics: Part two*. Number v. 1. D. Van Nostrand Company, Incorporated, 1951. URL: https://books.google.co.uk/books?id=Hkb_wwEACAAJ.

[LMM+16] Jake Longo, Daniel P. Martin, Luke Mather, Elisabeth Oswald, Benjamin Sach, and Martijn Stam. How low can you go? Using side-channel data to enhance brute-force key recovery. *IACR Cryptol. ePrint Arch.*, page 609, 2016. URL: http://eprint.iacr.org/2016/609.

[Mas94] James L Massey. Guessing and entropy. In *Proceedings of 1994 IEEE International Symposium on Information Theory*, page 204. IEEE, 1994. doi:10.1109/ISIT.1994.394764.

[mdt23] The mpmath development team. *mpmath: a Python library for arbitrary-precision floating-point arithmetic (version 1.3.0)*, 2023. http://mpmath.org/.

[MM18] Daniel P. Martin and Marco Martinoli. A note on key rank. Cryptology ePrint Archive, Paper 2018/614, 2018. https://eprint.iacr.org/2018/614. URL: https://eprint.iacr.org/2018/614.

[MMO18] Daniel P. Martin, Luke Mather, and Elisabeth Oswald. Two sides of the same coin: Counting and enumerating keys post side-channel attacks revisited. In Nigel P. Smart, editor, *Topics in Cryptology - CT-RSA 2018 - The Cryptographers' Track at the RSA Conference 2018, San Francisco, CA, USA, April 16-20, 2018, Proceedings*, volume 10808 of *Lecture Notes in Computer Science*, pages 394–412. Springer, 2018. doi:10.1007/978-3-319-76953-0_21.

[MMOS16a] Daniel P. Martin, Luke Mather, Elisabeth Oswald, and Martijn Stam. Characterisation and estimation of the key rank distribution in the context of side channel evaluations. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 548–572, 2016. doi:10.1007/978-3-662-53887-6_20.

[MMOS16b] Daniel P. Martin, Luke Mather, Elisabeth Oswald, and Martijn Stam. Labynkyr. https://github.com/sca-research/labynkyr, 2016.

[MOOS15] Daniel P. Martin, Jonathan F. O'Connell, Elisabeth Oswald, and Martijn Stam. Counting keys in parallel after a side channel attack. In Tetsu Iwata and Jung Hee Cheon, editors, *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*, volume 9453 of *Lecture Notes in Computer Science*, pages 313–337. Springer, 2015. doi:10.1007/978-3-662-48800-3_13.

[Pol19] Ricardo Villanueva Polanco. *Cold boot attacks on post-quantum schemes*. PhD thesis, Royal Holloway, University of London, 2019.

[PSG16] Romain Poussier, François-Xavier Standaert, and Vincent Grosso. Simple key enumeration (and rank estimation) using histograms: An integrated approach. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, volume 9813 of *Lecture Notes in Computer Science*, pages 61–81. Springer, 2016. doi:10.1007/978-3-662-53140-2_4.

[Tuk77] John Wilder Tukey. Exploratory data analysis. *Reading/Addison-Wesley*, 1977.

[VGRS12] Nicolas Veyrat-Charvillon, Benoît Gérard, Mathieu Renauld, and François-Xavier Standaert. An optimal key enumeration algorithm and its application

to side-channel attacks. In Lars R. Knudsen and Huapeng Wu, editors, *Selected Areas in Cryptography, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers*, volume 7707 of *Lecture Notes in Computer Science*, pages 390–406. Springer, 2012. doi:10.1007/978-3-642-35999-6_25.

[VGS13] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Security evaluations beyond computing power. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 126–141. Springer, 2013. doi:10.1007/978-3-642-38348-9_8.

[YEM14] Xin Ye, Thomas Eisenbarth, and William Martin. Bounded, yet sufficient? how to determine whether limited side channel information enables key recovery. In Marc Joye and Amir Moradi, editors, *Smart Card Research and Advanced Applications - 13th International Conference, CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers*, volume 8968 of *Lecture Notes in Computer Science*, pages 215–232. Springer, 2014. doi:10.1007/978-3-319-16763-3_13.

[YMO22] Rebecca Young, Luke Mather, and Elisabeth Oswald. Comparing key rank estimation methods. In Ileana Buhan and Tobias Schneider, editors, *Smart Card Research and Advanced Applications - 21st International Conference, CARDIS 2022, Birmingham, UK, November 7-9, 2022, Revised Selected Papers*, volume 13820 of *Lecture Notes in Computer Science*, pages 188–204. Springer, 2022. doi:10.1007/978-3-031-25319-5_10.

[ZDF20] Ziyue Zhang, A. Adam Ding, and Yunsi Fei. A fast and accurate guessing entropy estimation algorithm for full-key recovery. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(2):26–48, 2020. URL: https://doi.org/10.13154/tches.v2020.i2.26-48, doi:10.13154/TCHES.V2020.I2.26-48.