Poly-Logarithmic Side Channel Rank Estimation via Exponential Sampling

Liron David[1], Avishai Wool[2]

School of Electrical Engineering, Tel Aviv University, Ramat Aviv 69978, Israel
[1]lirondavid@gmail.com, [2]yash@eng.tau.ac.il

Abstract. Rank estimation is an important tool for a side-channel evaluations laboratories. It allows estimating the remaining security after an attack has been performed, quantified as the time complexity and the memory consumption required to brute force the key given the leakages as probability distributions over d subkeys (usually key bytes). These estimations are particularly useful where the key is not reachable with exhaustive search.
We propose ESrank, the first rank estimation algorithm that enjoys provable poly-logarithmic time- and space-complexity, which also achieves excellent practical performance. Our main idea is to use exponential sampling to drastically reduce the algorithm's complexity. Importantly, ESrank is simple to build from scratch, and requires no algorithmic tools beyond a sorting function. After rigorously bounding the accuracy, time and space complexities, we evaluated the performance of ESrank on a real SCA data corpus, and compared it to the currently-best histogram-based algorithm. We show that ESrank gives excellent rank estimation (with roughly a 1-bit margin between lower and upper bounds), with a performance that is on-par with the Histogram algorithm: a run-time of under 1 second on a standard laptop using 6.5 MB RAM.

1 Introduction

1.1 Background

Side-channel attacks (SCA) represent a serious threat to the security of cryptographic hardware products. As such, they reveal the secret key of a cryptosystem based on leakage information gained from physical implementation of the cryptosystem on different devices. Information provided by sources such as timing [12], power consumption [11], electromagnetic emulation [21], electromagnetic radiation [2, 9] and other sources, can be exploited by SCA to break cryptosystems.

A security evaluation of a cryptographic device should determine whether an implementation is secure against such an attack. To do so, the evaluator needs to determine how much time, what kind of computing power and how much storage a malicious attacker would need to recover the key given the side-channel leakages. The leakage of cryptographic implementations is highly device-specific, therefore the usual strategy for an evaluation laboratory is to launch a

set of popular attacks, and to determine whether the adversary can break the implementation (i.e., recover the key) using "reasonable" efforts.

Most of the attacks that have been published in the literature are based on a "divide-and-conquer" strategy. In the first "divide" part, the cryptanalyst recovers multi-dimensional information about different parts of the key, usually called subkeys (e.g., each of the $d = 16$ AES key bytes can be a subkey). In the "conquer" part the cryptanalyst combines the information all together in an efficient way via key enumeration [19, 23, 6]. In the attacks we consider in this paper, the information that the SCA provides for each subkey is a probability distribution over the N candidate values for that subkey, and the SCA probability of a full key is the product of the SCA probabilities of its d subkeys.

A security evaluator knows the secret key and aims to estimate the number of decryption attempts the attacker needs to do before he reaches to the correct key, assuming the attacker uses the SCA's probability distribution. Clearly enumerating the keys in the optimal SCA-predicted order is the best strategy the evaluator can follow. However, this is limited by the computational power of the evaluator. This is a worrying situation because it is hard to decide whether an implementation is "practically secure". For example, one could enumerate the 2^{50} first keys for an AES implementation (in the optimal order) without finding the correct key, and then conclude that the implementation is practically secure because the attacker needs to enumerate beyond 2^{50} number of keys. But, this does not provide any hint whether the concrete security level is 2^{51} or 2^{120}. This makes a significant difference in practice, especially in view of the possibility of improved measurement setups, signal processing, information extraction, etc., that should be taken into account for any physical security evaluation, e.g., via larger security margins.

In this paper, we introduce a new method to estimate the rank of a given secret key in the optimal SCA-predicted order. Our algorithm enjoys simplicity, accuracy and provable poly-logarithmic time and memory efficiency and excellent practical performance.

The rank estimation problem: Given d independent subkey spaces each of size N with their corresponding probability distributions $P_1, ..., P_d$ such that P_i is sorted in decreasing order of probabilities, and given a key k^* indexed by $(k_1, ..., k_d)$, let $p^* = P_1(k_1) \cdot P_2(k_2) \cdot ... \cdot P_d(k_d)$ be the probability of k^* to be the correct key. The evaluator would like to estimate the number of full keys with probability higher than p^*, when the probability of a full key is defined as the product of its subkey's probabilities.

In other words, the evaluator would like to estimate k^*'s rank: the position of the key k^* in the sorted list of N^d possible keys when the list is sorted in decreasing probability order, from the most likely key to the least. If the dimensions, or k^*'s rank are small, one can easily compute the rank of the correct key by a straightforward key enumeration. However, for a key with a high rank r, any optimal-order key enumeration requires $\Omega(r)$ time — which may be prohibitive, and the currently-best optimal-order key enumeration algorithm [23] requires $\Omega(N^{d/2})$ space, which again may be prohibitive. Hence developing

fast and low-memory algorithms to estimate the rank without enumeration is of great interest.

1.2 Related work

The best key enumeration algorithm so far, in terms of optimal-order, was presented by Veyrat-Charvillon, Gérard, Renauld and Standaert in [23]. However, its worst case space complexity is $\Omega(N^{d/2})$ when d is the number of subkey dimensions and N is the number of candidates per subkey - and its space complexity is $\Omega(r)$ when enumerating up to a key at rank $r \leq N^{d/2}$. Thus its space complexity becomes a bottleneck on real computers with bounded RAM in realistic SCA attacks.

Since then several near-optimal key enumeration were proposed [4, 18, 20, 26, 3, 10, 6, 15, 13, 14, 17, 22]. However, none of these key enumeration algorithms enumerate the whole key space within a realistic amount of time and with a realistic amount of computational power: enumerating an exponential key space will always come at an exponential cost. Hence the need for efficient and accurate rank estimation for keys that have a high rank.

The first rank estimation algorithm was proposed by Veyrat-Charvillon et al. [24]. They suggested to organize the keys by sorting their subkeys according to the a-posteriori probabilities provided, and to represent them as a high-dimensional dataspace. The full key space can then be partitioned in two volumes: one defined by the key candidates with probability higher than the correct key, one defined by the key candidates with probability lower than the correct key. Using this geometrical representation, the rank estimation problem can be stated as the one of finding bounds on these "higher" and "lower" volumes. It essentially works by carving volumes representing key candidates on each side of their boundary, progressively refining the lower and upper bounds on the key rank. Refining the bounds becomes exponentially difficult at some point.

A number of works have investigated solutions to improve upon [24]. In particular, Glowacz et al. [10] presented a rank estimation algorithm that is based on a convolution of histograms and allows obtaining tight bounds for the key rank of (even large) keys. This Histogram algorithm is currently the best rank estimation algorithm we are aware of. The space complexity of this algorithm is $O(dB)$ where d is the number of dimensions and B is a design parameter controlling the number of the histogram bins. A comparable result was developed independently by Bernstein et al. [3].

Martin et al. [18] used a score-based rank enumeration, rather than a probability based rank estimation. They mapped the rank estimation to a knapsack problem, which can be simplified and expressed as path counting. Subsequently, in [16] Martin et al. show that their algorithm [18] is mathematically equivalent to the Histogram algorithm [10] for a suitable choice of their respective discretization parameter, thus they can both be equally accurate. Since the two algorithms are equivalent we compared our algorithm's performance only to that of the Histogram algorithm [10].

Ye et al. investigated an alternative solution based on a weak Maximum Likelihood (wML) approach [26], rather than a Maximum Likelihood (ML) one for the previous examples. They additionally combined this wML approach with the possibility to approximate the security of an implementation based on "easier to sample" metrics, e.g., starting from the subkey Success Rates (SR) rather than their likelihoods. Later Duc et al. [7] described a simple alternative to the algorithm of Ye et al. and provided an "even easier to sample" bound on the subkey SR, by exploiting their formal connection with a Mutual Information metric. Recently, Wang at al. [25] presented a rank estimation for at dependent score lists.

Choudary et al. [5] presented a method for estimating Massey's guessing entropy (GM) which is the statistical expectation of the position of the correct key in the sorted distribution. Their method allows to estimate the GM within a few bits. However, the *actual* guessing entropy (GE), i.e., the *rank* of the correct key, is sometimes quite different from the expectation. In contrast, our algorithm focuses on the real GE.

1.3 Contribution

In this paper we propose a simple and effective new rank estimation method called ESrank, that is fundamentally different from previous approaches. We have rigorously analyzed its accuracy, time and space complexities. Our main idea is to use exponential sampling to drastically reduce the algorithm's complexity. We prove ESrank has a poly-logarithmic time- and space-complexity: for a design parameter $1 < \gamma < 2$ ESrank has $O(\frac{d^2}{4}(\log_\gamma N)^2 \log(\log_\gamma N))$ time and $O(d \log_\gamma N + \frac{d^2}{16}(\log_\gamma N)^2)$ space, and it can be driven to any desired level of accuracy (trading off time and space against accuracy). Importantly, ESrank is simple to build from scratch, and requires no algorithmic tools beyond a sorting function.

Beyond asymptotic analysis, we evaluated the performance of ESrank through extensive simulations based on a real SCA data corpus, and compared it to the currently-best histogram-based algorithm. We showed that ESrank gives excellent rank estimation (with roughly a 1-bit margin between lower and upper bounds), with a performance that is on-par with the Histogram algorithm: a run-time of under 1 second, for all ranks up to 2^{128}, on a standard laptop using at most 6.5 MB RAM. Hence ESrank is a useful addition to the SCA evaluator's toolbox.

2 The ESrank Algorithm for the case $d = 2$

We start with describing the idea of our algorithm in case $d = 2$, then we shall extend this idea for the general case $d \geq 2$.

Algorithm 1: Exact rank.

Input: Two non-decreasing probability distributions P_1, P_2 of size N each, the correct key $k^* = (k_1, k_2)$ and its probability $p^* = P_1[k_1] \cdot P_2[k_2]$.

Output: $Rank(k^*)$.

```
1 i = 1; j = N; rank = 0;
2 while i ≤ N and j ≥ 1 do
3 |   p = P1[i] · P2[j];
4 |   if p ≥ p* then
5 |   |   rank = rank + j;
6 |   |   i = i + 1;
7 |   else
8 |   |   j = j − 1;
9 return rank;
```

2.1 An exact rank estimation for $d = 2$

Definition 1 ($Rank(k^*)$)**.** *Let d non-increasing subkey probability distributions P_i for $1 \leq i \leq d$ and the correct key $k^* = (k_1, ..., k_d)$ be given. Let $p^* = P_1[k_1] \cdot ... \cdot P_d[k_d]$ be the probability of the correct key. Then, define $Rank(k^*)$ to be the number of keys $(x_1, ..., x_d)$ s.t. $P_1[x_1] \cdot ... \cdot P_d[x_d] \geq p^*$.*

Definition 2. *Let 2 non-increasing subkey probability distributions P_1 and P_2, each of size N, the correct key $k^* = (k_1, k_2)$ and an index $1 \leq i \leq N$ be given. Let $p^* = P_1[k_1] \cdot P_2[k_2]$ be the probability of the correct key. Then define H_i to be the number of points (i, j) such that $P_1[i] \cdot P_2[j] \geq p^*$, i.e.,*

$$H_i(k^*) = |\{(i, j) | P_1[i] \cdot P_2[j] \geq p^*\}|.$$

The idea of the algorithm is to find $H_i(k^*)$ for each i. The rank of the correct key k^* is the sum of $H_i(k^*)$ over $1 \leq i \leq N$, i.e.,

$$Rank(k^*) = \sum_{i=1}^{N} H_i(k^*).$$

The pseudo code is described in Algorithm 1. The correctness of Algorithm 1 stems from the observation that $H_i \geq H_{i+1}$ for all $1 \leq i \leq N - 1$. Therefore, to find H_{i+1}, j starts from H_i and it is decreased until H_{i+1} is found.

Proposition 1. *The running time of Algorithm 1 is $\Theta(N)$.*

Proof: At the beginning $i = 1$ and $j = N$. In each iteration either i is increased by 1 or j is decreased by 1 until either $i = N + 1$ or $j = 0$. Therefore, the number of steps is at most $2 \cdot N$ in case both i and j reach their limits, and is at least N in case only one of them reaches it limit. Therefore, the running time is $\Theta(N)$. □

2.2 Exponential Sampling with $d = 2$

To make this algorithm faster, we use sampling. Intuitively, we sample a set of indices SI and run Algorithm 1 on the $SI \times SI$ grid. On the sampled indices Algorithm 1 is no longer exact, but we can modify it to produce lower and upper bounds on $Rank(k^*)$. As we shall see, if we use exponential sampling, defined in Section 2.3, we can bound the inaccuracy introduced by the sampling.

Given a non-increasing subkey probability distribution P of size N, the sampling process returns a sampled probability distribution (SI, SP) of size N_s where $N_s = O(\log N)$. SI contains the sampled indices and SP contains their corresponding probabilities such that $SP[i] = P[SI[i]]$ for all $i \leq N_s$.

The goal of the exponential sampling is to maintain an invariant on the ratio between sampled indices. Let $1 < \gamma \leq 2$ be given and let b be the smallest i such that $i/(i-1) \leq \gamma$. The first b sampled indices are the first b indices of P. The rest of the sampled indices are sampled from P at powers of γ. More formally, for all $i \leq N_s - 1$ the sampled indices obey:

$$\begin{cases} SI[i] = i & \text{if } i \leq b \\ SI[i]/SI[i-1] \leq \gamma \text{ and } SI[i+1]/SI[i-1] > \gamma & \text{otherwise.} \end{cases} \tag{1}$$

E.g., if $\gamma = 2$ then $b = 2$, and for $SI = \{1, 2, 4, 8, \ldots, N\}$ invariant (1) holds. The pseudo code of this sampling is described in Algorithm 2. Note that the indices 1 and N are always included in SI.

Lemma 1. *If SI is the output of Algorithm 2 then for any index $i \geq b + 1$ in SI it holds that*

$$SI[i] = \lfloor \gamma \cdot SI[i-1] \rfloor.$$

and

$$SI[i] - SI[i-1] = \lfloor (\gamma - 1) \cdot SI[i-1] \rfloor.$$

Proof: According to Algorithm 2, it holds that

$$\frac{SI[i]}{SI[i-1]} \leq \gamma$$

and

$$\frac{SI[i]+1}{SI[i-1]} > \gamma.$$

Therefore,

$$\gamma \cdot SI[i-1] - 1 < SI[i] \leq \gamma \cdot SI[i-1].$$

Since $SI[i]$ is an integer it holds

$$SI[i] = \lfloor \gamma \cdot SI[i-1] \rfloor.$$

The difference $SI[i] - SI[i-1]$ obeys

$$(\gamma - 1) \cdot SI[i-1] - 1 < SI[i] - SI[i-1] \leq (\gamma - 1) \cdot SI[i-1].$$

Algorithm 2: Sampling Process.

Input: A probability distribution P of size N, b, γ.
Output: A sampled probability distribution (SI, SP).

```
for i = 1 to b do
    SI[i] = i; SP[i] = P[i];
j = b; i = j + 1; c = j + 1;
while i < N do
    if i/j ≤ γ and (i + 1)/j > γ then
        SI[c] = i; SP[c] = P[i];
        c = c + 1; j = i; i = i + 1;
SI[c] = N; SP[c] = P[N];
return (SI, SP);
```

The indices of SI are integers, therefore we get

$$SI[i] - SI[i-1] = \lfloor (\gamma - 1) \cdot SI[i-1] \rfloor.$$

□

Proposition 2. *Let $N_s = |SI|$ be the size the sample returned by Algorithm 2. Then $b + \log_\gamma(N/b) \leq N_s < b + \log_\gamma(N/(b-1))$.*

Proof: Since $b \cdot r^{Ns} \geq N$ and $b \cdot r^{Ns-1} < N$.

Definition 3. *Let two sampled probability distributions $(SI, SP_1), (SI, SP_2)$, each of size N_s, the correct key $k^* = (k_1, k_2)$, its probability p^* and an index $1 \leq i \leq N_s$ be given. Then define H_i^S to be the number of points (i, j) s.t. $1 \leq j \leq N$ and $SP_1[i] \cdot P_2[j] \geq p^*$, i.e.,*

$$H_i^S(k^*) = |\{(i,j) | 1 \leq j \leq N \text{ and } SP_1[i] \cdot P_2[j] \geq p^*\}|.$$

The difference between every two successive indices in the sampled probability distributions might be bigger than 1, i.e., $SI[i+1] - SI[i] > 1$ therefore, besides counting H_i^S for each $i \leq N_s$ we also need to add the number of points (i, j) such that $SI[i] \leq i \leq SI[i+1]$. Recall that Algorithm 2 always includes $i = N$ in SI.

Definition 4. *Let two sampled probability distributions $(SI, SP_1), (SI, SP_2)$, each of size N_s, the correct key $k^* = (k_1, k_2)$, its probability p^* and an index $1 \leq i \leq N_s$ be given. Then define $H_{a,b}^S$ be the number of (i, j) s.t. $1 \leq j \leq N$ and $SI[a] < i < SI[b]$ and $SP_1[i] \cdot P_2[j] \geq p^*$, i.e.,*

$$H_{a,b}^S(k^*) = |\{(i,j) | 1 \leq j \leq N \text{ and } SP_1[i] \cdot P_2[j] \geq p^* \text{ and } SI[a] < i < SI[b]\}|.$$

The idea of Algorithm 3 is to find $H_i^S(k^*)$ for each $i \in \{1, ..., N_s\}$ and $H_{i,i+1}^S(k^*)$ for each $i \in \{1, ..., N_s - 1\}$. The rank of the correct key k^* is the following sum:

$$Rank(k^*) = \sum_{i=1}^{N_s} H_i^S(k^*) + \sum_{i=1}^{N_s - 1} H_{i,i+1}^S(k^*).$$

Algorithm 3: Calculating Upper and Lower bounds.

Input: Sampled probability distributions SP_1, SP_2 each of size N_s, b, the correct key $k^* = (k_1, k_2)$ and it probability p^*.

Output: Upper and lower bounds on $Rank(k^*)$.

```
1  iLast = N_s; jLast = N_s;
2  if k_1 == 1 then jLast = k_2;
3  if k_2 == 1 then iLast = k_1;
4  i = 1; j = jLast; ub = 0; lb = 0;
5  while i ≤ iLast and j ≥ 1 do
6  |  pCurr = SP_1[i] · SP_2[j];
7  |  if pCurr ≥ p* then
8  |  |  u = l = SI_2[j]; uPrev = u;
9  |  |  ub = ub + u; lb = lb + l;
10 |  |  if i ≥ b + 1 then
11 |  |  |  ub = ub + uPrev · (SI[i] − SI[i − 1] − 1);
12 |  |  |  lb = lb + l · (SI[i] − SI[i − 1] − 1);
13 |  |  i = i + 1;
14 |  else if j > 1 then
15 |  |  pNext = SP_1[i] · SP_2[j − 1];
16 |  |  if pNext < p* < pCurr then
17 |  |  |  u = SI[j] − 1; l = SI[j − 1]; uPrev = u;
18 |  |  |  ub = ub + u; lb = lb + l;
19 |  |  |  if i ≥ b + 1 then
20 |  |  |  |  ub = ub + uPrev · (SI[i] − SI[i − 1] − 1);
21 |  |  |  |  lb = lb + l · (SI[i] − SI[i − 1] − 1);
22 |  |  |  i = i + 1;
23 |  |  else
24 |  |  |  j = j − 1;
25 |  else
26 |  |  j = j − 1;
27 if j < 1 and i ≤ iLast then
28 |  ub = ub + uPrev · (SI[i] − SI[i − 1] − 1);
29 return (lb, ub);
```

Since we are given sampled distributions, we cannot calculate the exact values of $H_i^S(k^*)$ and $H_{i,i+1}^S(k^*)$. Instead we calculate upper and lower bounds for each $H_i^S(k^*)$ and $H_{i,i+1}^S(k^*)$ as illustrated in Figure 1.

Definition 5. *Let $up(H_i^S(k^*))$ be an upper bound of $H_i^S(k^*)$ and let $up(H_{i,i+1}^S(k^*))$ be an upper bound of $H_{i,i+1}^S(k^*)$, i.e.,*

$$H_i^S(k^*) \leq up(H_i^S(k^*)) \text{ and } H_{i,i+1}^S(k^*) \leq up(H_{i,i+1}^S(k^*)).$$

Definition 6. *Let $low(H_i^S(k^*))$ be a lower bound of $H_i^S(k^*)$ and let $low(H_{i,i+1}^S(k^*))$ be a lower bound of $H_{i,i+1}^S(k^*)$, i.e.,*

$$H_i^S(k^*) \geq low(H_i^S(k^*)) \text{ and } H_{i,i+1}^S(k^*) \geq low(H_i^S(k^*)).$$

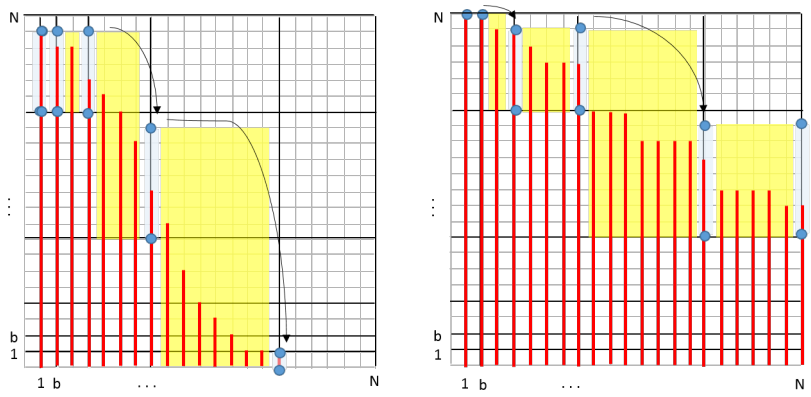

Fig. 1. The red bars represent the un-sampled H_i's, and the black grid represents the sampled indices in SI. For each sampled index $1 \leq i \leq N_s$ the blue circles are upper and lower bounds on $H_i^S(k^*)$. The yellow-shaded rectangles represent $H_{i,i+1}^S(k^*)$ for each $b \leq i \leq N_s - 1$, for two different keys. Note that the yellow-shaded rectangles stop exactly one index before the sampled indices, in both dimensions.

Therefore, it holds

$$\sum_{i=1}^{N_s} low(H_i(k^*)) + \sum_{i=1}^{N_s-1} low(H_{i,i+1}(k^*)) \leq Rank(k^*) \leq \\ \leq \sum_{i=1}^{N_s} up(H_i(k^*)) + \sum_{i=1}^{N_s-1} up(H_{i,i+1}(k^*)). \tag{2}$$

2.3 Bounding the Sampled Distributions

Given two probability distributions P_1 and P_2, each of size N, we first sample the indices using Algorithm 2. We get sampled probability distributions (SI, SP_1) and (SI, SP_2) each of size N_s when SI is the set of sampled indices and SP_1, SP_2 are the corresponding sampled probabilities. Given these sampled probability distributions, the next step is to calculate an upper bound and a lower bound for $Rank(k^*)$. This is done in Algorithm 3.

To do this, it keeps two variables: ub for the upper bound and lb for the lower bound. At the beginning, both ub and lb are initialized to 0.

Definition 7. *Given a key k^*, and given $1 \leq i \leq N_s$, let u_i be the value of u at iteration i in Algorithm 3 and let l_i be the value of l at iteration i in Algorithm 3.*

Algorithm 3 starts with $i = 1$ and $j = N_s$. It decreases j until one of the two options happens:
(a) (line 16) We reach the highest j such that

$$SP_1[i] \cdot SP_2[j] < p^* < SP_1[i] \cdot SP_2[j-1].$$

In this case $(i,j) \in H_i^S(k^*)$ but $(i,j-1) \notin H_i^S(k^*)$, therefore

$$SI[j-1] \leq H_i^S(k^*) \leq SI[j]-1.$$

Therefore the values of l_i and u_i become

$$l_i = SI[j-1] \text{ and } u_i = SI[j]-1, \tag{3}$$

and the running totals ub and lb are updated (line 9).
(b) (line 7) We reach the highest j such that

$$SP_1[i] \cdot SP_2[j] \geq p^*.$$

In this case we have the exact value of $H_i^S(k^*)$ which is

$$H_i^S(k^*) = SI[j].$$

Therefore the values of l_i and u_i become

$$l_i = u_i = SI[j], \tag{4}$$

and the running totals ub and lb are updated (line 18).

In the next step, after finding bounds on H_i^S, the algorithm moves to $i+1$ and finds bounds on H_{i+1}^S. Since $H_i^S \geq H_{i+1}^S$ we start from j of the previous iteration i.e., j s.t. $SI[j-1] \leq H_i^S \leq SI[j]$ and decrease it to get the corresponding bounds on H_{i+1}^S.

Once $i \geq b+1$ (lines 10, 19) the difference $SI[i]-SI[i-1] \geq 1$ therefore $H_{i-1,i}^S(k^*) \geq 1$ and it should be added. To upper bound this number we multiply the upper bound of H_{i-1}^S, which is $uPrev_i = u_{i-1}$, by the width of $H_{i-1,i}^S(k^*)$, which is $(SI[i]-SI[i-1]-1)$ (lines 11, 20). To lower bound $H_{i-1,i}^S(k^*)$ we multiply the lower bound of H_i^S, which is l_i by the width of $H_{i-1,i}^S(k^*)$, (lines 12, 21); see Figure 1.

Theorem 1. *Let two sampled probability distributions SP_1 and SP_2, which are sampled from the probability distributions P_1 and P_2 respectively using Algorithm 2 with $\gamma > 1$ be given and let b be the smallest i such that $i/(i-1) \leq \gamma$. For a key k^*, let ub and lb be the outputs of Algorithm 3. Then $ub/lb \leq \gamma^2$.*

Proof: From Equation (2) it holds that

$$ub = \sum_{i=1}^{N_s} up(H_i(k^*)) + \sum_{i=1}^{N_s-1} up(H_{i,i+1}(k^*)).$$

Since

$$up(H_i^S(k^*)) = u_i \text{ and } up(H_{i,i+1}(k^*)) = u_i \cdot (SI[i+1]-SI[i]-1)$$

we get

$$ub = \sum_{i=1}^{N_s} u_i + \sum_{i=1}^{N_s-1} u_i \cdot (SI[i+1]-SI[i]-1).$$

Since $SI[i+1] - SI[i] = 1$ for all $1 \leq i \leq b-1$, the first $b-1$ elements of the second sum are 0.

$$\begin{aligned} ub = &\sum_{i=1}^{N_s} u_i + \sum_{i=b}^{N_s-1} u_i \cdot (SI[i+1] - SI[i]) - 1) \\ &\sum_{i=1}^{b-1} u_i + \sum_{i=b}^{N_s-1} \big(u_i + u_i \cdot (SI[i+1] - SI[i] - 1)\big) + u_{N_s} \\ &\sum_{i=1}^{b-1} u_i + \sum_{i=b}^{N_s-1} u_i \cdot (SI[i+1] - SI[i])) + u_{N_s} \end{aligned}$$

Separating the b'th term from the second sum we get

$$ub \leq \left(\sum_{i=1}^{b-1} u_i\right) + u_b \cdot (SI[b+1] - SI[b]) + u_{N_s} + \sum_{i=b+1}^{N_s-1} u_i \cdot (SI[i+1] - SI[i]). \quad (5)$$

Similarly from Equation (2) it holds that

$$lb = \sum_{i=1}^{N_s} low(H_i(k^*)) + \sum_{i=1}^{N_s-1} low(H_{i,i+1}(k^*)).$$

Since

$$low(H_i^S(k^*)) = l_i \text{ and } low(H_{i,i+1}(k^*)) = l_{i+1} \cdot (SI[i+1] - SI[i] - 1)$$

(Note the shift in indices where the multiplication is by the lower bound of $i+1$) we get

$$lb = \sum_{i=1}^{N_s} l_i + \sum_{i=1}^{N_s-1} l_{i+1} \cdot (SI[i+1] - SI[i] - 1).$$

Again the first $b-1$ elements of the second sum are 0, therefore

$$lb = \sum_{i=1}^{N_s} l_i + \sum_{i=b}^{N_s-1} l_{i+1} \cdot (SI[i+1] - SI[i] - 1).$$

By shifting index i by 1 in the second sum, we get

$$\begin{aligned} lb &= \sum_{i=1}^{N_s} l_i + \sum_{i=b+1}^{N_s} l_i \cdot (SI[i] - SI[i-1] - 1) \\ &= \sum_{i=1}^{b} l_i + \sum_{i=b+1}^{N_s} \big(l_i + l_i \cdot (SI[i] - SI[i-1] - 1)\big) \\ &= \sum_{i=1}^{b} l_i + \sum_{i=b+1}^{N_s} l_i \cdot (SI[i] - SI[i-1]) \\ &\geq \sum_{i=1}^{b} l_i + \sum_{i=b+1}^{N_s-1} l_i \cdot (SI_1[i] - SI_1[i-1]). \end{aligned} \quad (6)$$

In order to show $ub/lb \leq \gamma^2$, we prove the following two Lemmas (in the Appendix):

Lemma 2.

$$\Big(\sum_{i=b+1}^{N_s-1} u_i \cdot (SI[i+1] - SI[i])\Big) / \Big(\sum_{i=b+1}^{N_s-1} l_i \cdot (SI[i] - SI[i-1])\Big) \leq \gamma^2$$

Lemma 3.

$$\Big((\sum_{i=1}^{b-1} u_i) + u_b \cdot (SI[b+1] - SI[b]) + u_{N_s}\Big) / \Big(\sum_{i=1}^{b} l_i\Big) \leq \gamma^2.$$

3 The general case $d > 2$

Given $d > 2$ sampled probability distributions $(SI_1, SP_1), ..., (SI_d, SP_d)$, and the correct key $k^* = (k_1, ..., k_d)$, we now follow the intuition of the $d = 2$ case to solve the general case. To do so, we organize the d distributions into pairs, merge the pairs into $d/2$ joint distributions, sub-sample the joint distributions, and continue in the same way until we get to a single pair of distributions sampled from the $N^{d/2}$-dimensioned half-keys. We achieve this via a sequence of algorithms described below.

3.1 Merging two sampled distributions into a joint distribution

Given two sampled non-increasing probability distributions $(SI_1, SP_1), (SI_2, SP_2)$, each of size N_s, we wish to merge them into one non-increasing distribution, and compute lower and upper bounds on the ranks of the points. Algorithm 4 implements this task.

First, the algorithm goes over the grid of N_s^2 points (i, j) such that $1 \leq i \leq N_s$ and $1 \leq j \leq N_s$. For each point (i, j) it calculates the point's probability $SP_1[i] \cdot SP_2[j]$. Then, we sort these points in decreasing order of their probabilities.

Given two consecutive points (i_1, j_1) and (i_2, j_2) in the sorted order such that $Prob(i_1, j_1) \geq Prob(i_2, j_2)$, all the points whose probability is greater than $Prob(i_1, j_1)$ are also greater than $Prob(i_2, j_2)$, therefore, all the points in the rank of (i_1, j_1) are contained in the rank of (i_2, j_2). Relying on this observation, if we know the order of the N_s^2 points according to their probabilities, we can bound the accumulative rank of these points while going over them from the most likely point to the least. In this way, the upper-bound of the rank of the current point (i_c, j_c) is the upper bound of the previous point (i_p, j_p) plus the following expressions:

$$\begin{aligned} &+ (SI[j_p + 1] - SI[j_p]) \cdot (SI[i_p + 1] - SI[i_p] - 1) \\ &+ SI[j_p + 1] - SI[j_p] - 1 \\ &+ 1 \\ &= (SI[j_p + 1] - SI[j_p]) \cdot (SI[i_p + 1] - SI[i_p]). \end{aligned} \tag{7}$$

Algorithm 4: Calculating the joint probability distribution.

Input: Sampled probability distributions SP_1, SP_2 each of size N_s.
Output: Joint probability distribution.

1 $r = 1$;
2 **for** $i = 1$ *to* N_s **do**
3 | **for** $j = 1$ *to* N_s **do**
4 | | $Y(r,1) = SP_1[i] \cdot SP_2[j]$; $Y(r,2) = (i,j)$;
5 | | $r = r + 1$;
6 $Y = Sort(Y)$ in decreasing order of $Y(r,1)$;
7 $ub(1,1) = 1$; $ub(1,2) = SP_1[1] \cdot SP_2[1]$;
8 $lb(1,1) = 1$; $lb(1,2) = SP_1[1] \cdot SP_2[1]$;
9 **for** $r = 2$ *to* N_s^2 **do**
10 | $(i_c, j_c) = Y(r,2)$; $(i_p, j_p) = Y(r-1,2)$;
11 | $ub(r,1) = ub(r-1,1) + (SI(j_p+1) - SI(j_p)) \cdot (SI(i_p+1) - SI(i_p)$;
12 | $lb(r,1) = lb(r-1,1) + (SI(j_c) - SI(j_c-1)) \cdot (SI(i_c) - SI_1(i_c-1)$;
13 | $ub(r,2) = lb(r,2) = Y(r,2)$;
14 return (ub, lb);

The first term in (7), $(SI[j_p+1] - SI[j_p]) \cdot (SI[i_p+1] - SI[i_p] - 1)$, represents the number of points that might come after the previous point and before the current point, which are not on the SI grid. I.e., these are the points (i,j) s.t.

$$SI[i_p] < i < SI[i_p+1] \text{ and } SI[j_p] \leq j < SI[j_p+1].$$

$SI[j_p+1]$ is not included since we haven't reached that point yet.

The second term in (7), $SI[j_p+1] - SI[j_p] - 1$, represents the number of points that might come after the previous point and before the current point which *are* on the SI grid. I.e., these are the points (i,j) s.t.

$$i = SI[i_p] \text{ and } SI[j_p] < j < SI[j_p+1].$$

$SI[j_p]$ is not included since the point $(SI[i_p], SI[j_p])$ is the previous point and it was already included and $SI[j_p+1]$ is not included since we haven't reached that point yet.

The last addition in (7) is 1, accounting for the current point itself.

The resulting expression can be seen in Algorithm 4 (line 11). A similar derivation can be done for the lower bound (omitted).

Note that Algorithm 4 does not require the one-dimensional ranks or even knowing upper or lower bounds on the one-dimensional ranks.

3.2 Sampling the joint probability distribution

The output of Algorithm 4 is a distribution over N_s^2 elements. We now show that we can sub-sample this distribution, via exponential sampling, using the *same* parameters b and γ used to create the one-dimension samples. Theorem 2 below shows that a sub-sampling with the same b and γ always exists.

Algorithm 5: Sub-Sampling the joint distribution.

Input: A joint probability distribution $(inSI, inSP)$ of size N_s^2, b, γ.
Output: A sampled probability distribution (SI, SP).

```
for i = 1 to b do
    SI[i] = inSI[i]; SP[i] = inSP[i];
j = b; i = j + 1; c = i + 1;
while i < N_s^2 do
    if inSI[i]/inSI[j] ≤ γ and inSI[i + 1]/inSI[j] > γ then
        SI[c] = inSI[i]; SP[c] = inSP[i];
        c = c + 1; j = i;
SI[c] = inSI[N_s^2]; SP[c] = inSP[N_s^2];
return (SI, SP);
```

We would like to sample this joint probability distribution using Algorithm 2, using b and γ, except now instead of the 1-dimensional ranks we sample using the rank-upper/lower-bounds, See Algorithm 5.

For this, we shall prove in Lemma 5 that the first b indices of the joint probability distribution are $1, ..., b$ and we shall prove in Theorem 2 that the ratio between any two successive ranks is at most γ.

Lemma 4. *For any index $i \geq b+1$ in SI it holds that*

$$SI[i] - SI[i-1] \leq (\gamma - 1) \cdot SI[i-1].$$

Lemma 5. *Given two sampled probability distributions (SI_1, SP_1) and (SI_2, SP_2) that are sampled by Algorithm 2 merged by Algorithm 4. The first b upper ranks in the upper joint probability distribution are the integers $1, .., b$ and the first b lower ranks in the lower joint probability distribution are the integers $1, .., b$.*

Proof: According to the sampling process in Algorithm 2 it holds: $\forall i \leq b$ $SI_1[i] = i$ and $SI_2[i] = i$. Therefore, the joint probability contains the indices of $(i, j) \in \{1, ..., b\} \times \{1, ..., b\}$. Since the first b points with the highest probabilities are somewhere in the square: $\{1, ..., b\} \times \{1, ..., b\}$. The rank of the first b composed only from points in this square, therefore for $i \leq b$, the upper bound and lower bound of the i'th element in the joint distribution are equal to each other and equal to i.

Theorem 2. *Given the joint probability distribution of the sampled probability distributions $(SI_1, SP_1), (SI_2, SP_2)$, The ratio between any two consecutive upper (lower) ranks is at most γ, where $1 < \gamma \leq 2$.*

Proof: Let $up(i_c, j_c)$ be the upper bound on the rank of point (i_c, j_c) as in Algorithm 4. As can be seen in Algorithm 4 the difference between the upper ranks of any two consecutive points (i_c, j_c) and (i_p, j_p) is

$$up(i_c, j_c) - up(i_p, j_p) = (SI_2(j_p + 1) - SI_2(j_p)) \cdot (SI_1(i_p + 1) - SI_1(i_p)).$$

By Lemma 4 it holds that

$$SI_2(j_p+1) - SI_2(j_p) \leq SI_2(j_p) \cdot (\gamma - 1)$$
$$SI_1(i_p+1) - SI_1(i_p) \leq SI_1(i_p) \cdot (\gamma - 1).$$

Therefore,

$$up(i_c, j_c) - up(i_p, j_p) \leq SI_2(j_p) \cdot SI_1(i_p) \cdot (\gamma - 1)^2$$

The trivial lower bound of $rank(i_p, j_p)$ is the multiplication of its index, therefore

$$up(i_n, j_n) - up(i_p, j_p) \leq up(i_p, j_p) \cdot (\gamma - 1)^2$$

Since $1 < \gamma \leq 2$, it holds $(\gamma - 1)^2 \leq (\gamma - 1)$, therefore

$$up(i_c, j_c) - up(i_p, j_p) \leq up(i_p, j_p) \cdot (\gamma - 1)$$

and we get

$$up(i_c, j_c) \leq up(i_p, j_p) \cdot \gamma.$$

Similarly, as can be seen in Algorithm 4 the difference between the lower ranks of any two successive points (i_n, j_n) and (i_p, j_p) is

$$up(i_c, j_c) - up(i_p, j_p) = (SI_2(j_c) - SI_2(j_c - 1)) \cdot (SI_1(i_c) - SI_1(i_c - 1)).$$

A similar argument shows that $low(i_c, j_c) \leq low(i_p, j_p) \cdot \gamma$.

Theorem 2 shows that in the joint $N_s \times N_s$ distribution, the upper (lower) bounds of every two consecutive points (in sorted order) obey the invariant $ub(i_c, j_c) \leq \gamma \cdot ub(i_p, j_p)$.

Corollary 1. *The sample produced by Algorithm 5 on an input distribution of size N_s^2 consists of $O(N_s)$ ranks.*

3.3 The ESrank Algorithm: Putting it all together

Given $d > 2$ sampled probability distributions $(SI_1, SP_1), ..., (SI_d, SP_d)$, and the correct key $k^* = (k_1, ..., k_d)$, we first merge the d sampled probability distributions into $d/2$ sampled joint distributions, so that we get $d/2$ sampled upper- and lower-bounded distributions. Now, We take the $d/2$ upper-bounded distributions and merge them into $d/4$ sampled upper-bounded distributions, and similarly for the lower bounded distribution. We continue in the same way until we get two pairs of joint distributions: one pair of upper sampled joint distributions and one pair of lower sampled joint distributions. Now, we apply Algorithm 3 on the upper pair sampled joint distribution to get the upper bound of $Rank(k^*)$ and again, we apply Algorithm 3 on the lower pair sampled joint distribution to get the lower bound of $Rank(k^*)$. Algorithm 6 shows the complete pseudo-code for ESrank.

Algorithm 6: ESrank: Calculating the upper and lower bounds for $d > 2$.

Input: The probability distributions $P_1, ..., P_d$, the correct key $k^* = (k_1, ..., k_d)$, b and γ.

Output: Upper and lower bounds of $rank(k^*)$.

```
for i = 1 to d do
    (SI_i, SP_i) = Alg2(P_i, b, γ) ;          // Sample the input distributions
dim = d;
while dim ≠ 2 do
    for i = 1 to dim/2 do
        (ub_i, lb_i) = Alg4((SI_{2i-1}, SP_{2i-1}), (SI_{2i}, SP_{2i})) ;   // Merge
        (SI_i, SP_i) = Alg5(ub_i) ;                                         // Sub-Sample
    dim = dim/2;
(ub', lb') = Alg3((SI_1, SP_1), (SI_2, SP_2)) ;          // Calculate upper bound
dim = d;
while dim ≠ 2 do
    for i = 1 to dim/2 do
        (ub_i, lb_i) = Alg4((SI_{2i-1}, SP_{2i-1}), (SI_{2i}, SP_{2i})) ;   // Merge
        (SI_i, SP_i) = Alg5(lb_i) ;                                         // Sub-Sample
    dim = dim/2;
(ub'', lb'') = Alg3((SI_1, SP_1), (SI_2, SP_2)) ;          // Calculate lower bound
return (ub', lb'');
```

3.4 Theoretical Performance

Time complexity At each level of Algorithm 6 it uses Algorithm 4 to merge the sampled distributions received from the previous level. Algorithm 4 goes over N_s^2 pairs, calculates their probabilities using $\Theta(N_s^2)$ time, and sorts them using $\Theta(N_s^2 \cdot \log N_s)$ time. Let $T(d, \gamma)$ be the total the running time. Then

$$\begin{aligned} T(d,\gamma) &\leq \sum_{i=1}^{\log d - 2} \frac{d}{2^i} (2^{i-1} \log_\gamma N)^2 \log (2^{i-1} \log_\gamma N)^2 \\ &\leq \frac{d^2}{4} (\log_\gamma N)^2 \log(\log_\gamma N). \end{aligned}$$

I.e., we see that ESrank has a poly-logarithmic time complexity (in N).

Accuracy Assume the correct key is $k^* = (k_1, ..., k_d)$. For a key (k_i, k_{i+1}) and $(SI_i, SP_i), (SI_{i+1}, SP_{i+1})$ let $k_{i,i+1}$ be the real rank of (k_i, k_{i+1}). At the lowest level Theorem 1 and Algorithm 3 give that $up(k_i, k_{i+1}) \leq \gamma^2 k_{i,i+1}$. In the next level, each rank in the sampled joint distribution is multiplied by at most γ^2, therefore each term in the sum that composes $up(\gamma^2 k_{i,i+1}, \gamma^2 k_{i+2,i+3})$ is multiplied by at most γ^4. Hence $up(\gamma^2 k_{i,i+1}, \gamma^2 k_{i+2,i+3}) \leq \gamma^4 up(k_{i,i+1}, k_{i+2,i+3}) \leq \gamma^4 \gamma^2 k_{i,i+1,i+2,i+3} = \gamma^6 k_{i,i+1,i+2,i+3}$. We continue in the same way, and get

$$up(Rank(k^*))/Rank(k^*) \leq \gamma^{\sum_{i=1}^{\log d} 2^i} = \gamma^{2d-2}.$$

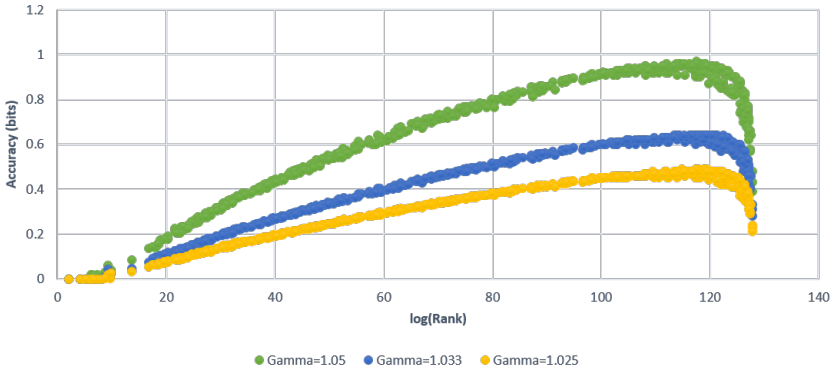

Fig. 2. The accuracy ($\log_2$ of the ratio between the upper- and lower-bounds) for the ESrank algorithm as a function of $\log_2(Rank(k^*))$ for different parameter settings: $\gamma = 1.05$ (green), $\gamma = 1.033$ (blue), $\gamma = 1.025$ (yellow).

Since $rank(k^*)$ might be any value in $[low(Rank(k^*), up(Rank(k^*)]$, we get

$$accuracy(d, \gamma) = up(Rank(k^*))/low(Rank(k^*)) \leq \gamma^{2d-2}.$$

E.g., for AES-128 with a preprocessing step of merging the 16 8-bit distributions into $d = 8$ 16-bit distributions we get $2d - 2 = 14$.

Space complexity In the first step we need to store d distributions of size $\log_\gamma N$ from Algorithm 4. In order to merge each pair of distributions into one, we need addition memory of $(\log_\gamma N)^2$. After merging 2 distributions each of size $(\log_\gamma N)$, we get one sampled distribution of size $(\log_\gamma N^2)$ which is $2(\log_\gamma N)$. Since we do not need the original pair any more, we can overwrite this space of size $2(\log_\gamma N)$ and store the new distribution into it. In the same way, in order to merge two distributions of size $\log_\gamma N^2$ we need additional space of $(\log_\gamma N^2)^2$, and the merged distribution will overwrite the original pair. In the last step, we need to merge 4 distributions, each of size $N^{d/4}$, therefore the maximum additional space we need is $(\log_\gamma N^{d/4})^2$. In total we get $d \log_\gamma N + (\log_\gamma N^{d/4})^2$ which is

$$space(d, \gamma) = d \log_\gamma N + \frac{d^2}{16} (\log_\gamma N)^2.$$

4 Empirical Evaluation

We evaluated the performance of the ESrank algorithm through an extensive simulation study. We compared the our algorithm to the currently best rank estimation algorithm: the Histogram algorithm of [10]. We implemented both in Matlab. We ran both algorithms on a 2.80GHz i7 PC with 8GB RAM running Microsoft windows 7, 64bit.

	Time (Seconds)	Space (MB)	Accuracy < 1 bit (%)
$\gamma = 1.025$	0.59	6.48	100
$\gamma = 1.033$	0.3	3.68	100
$\gamma = 1.05$	0.16	1.60	99.83
$\gamma = 1.065$	0.05	0.96	56.95
$B = 50K$	0.62	3.20	100
$B = 35K$	0.29	2.24	100
$B = 20K$	0.12	1.28	100
$B = 5K$	0.01	0.32	99.83

Table 1. Performance summary of the ESrank and Histogram algorithms. The Accuracy column indicates the percentage of traces for which the difference between the upper- and lower-bounds of the estimated ranks was below 1 bit.

For the performance evaluation we used the data of [8]. Within this data corpus there are 611 probability distribution sets gathered from a specific SCA. The SCA of [8] was against AES [1] with 128-bits keys running on an embedded processor with an unstable clock. Each set represents a particular setting of the SCA: number of traces used, whether the clock was jittered, and the values of tunable attack parameters. The attack grouped the key bits into 16 8-bit subkeys, and hence its output probability distributions are over these byte values. Each set in the corpus consists of the correct secret key and 16 distributions, one per subkey. The distributions are sorted in non-increasing order of probability, each of length 2^8. We used the same technique suggested in [10]: merge the $d = 16$ probability lists of size $N = 2^8$ into $d = 8$ lists of size $N = 2^{16}$. We measured the upper bound, lower bound, time and space for each trace using ESrank and the Histogram rank estimation.

Bound Tightness Figure 2 shows that the analytical performance of section 3.4 indeed agrees with the empirical results. For different values of γ we get accuracy which corresponds to at most γ^{14}: e.g., when $\gamma = 1.05$ Figure 2 shows a margin of at most 0.9 bits. We can see that as γ becomes closer to 1, the accuracy becomes closer to 0. As we expected, the maximum gap between the upper bound and the lower bound happens for ranks around $100-120$ since the difference between any two successive indices in the sampled set becomes greater when the indices becomes greater.

Time and Space Analysis Table 1 shows the time, the space and the percentage of the traces for which the accuracy is better than 1 bit, for ESrank with $\gamma = 1.025, 1.033, 1.05, 1.065$ and for Histogram [10] with B =50,000, 35,000, 20,000, 5,000. As we can see, the two algorithms, using the described parameters - all take less than 0.6 seconds and use under 6.5 MB of memory. In a practical sense ESrank is on-par with the Histogram algorithm: both exhibit a

run-time of under 1 second using less than 6.5 MB, to get a 1-bit margin of uncertainty in the rank for all ranks up to 2^{128}.

5 Conclusion

In this paper we proposed a simple and effective new rank estimation method. We have rigorously analyzed its accuracy, and its time and space complexities. Our main idea is to use exponential sampling to drastically reduce the algorithm's complexity. We proved ESrank has a poly-logarithmic time- and space-complexity, and it can be driven to any desired level of accuracy (trading off time and space against accuracy). Importantly, ESrank is simple to build from scratch, and requires no algorithmic tools beyond a sorting function.

We evaluated the performance of ESrank through extensive simulations based on a real SCA data corpus, and compared it to the currently-best histogram-based algorithm. We showed that ESrank gives excellent rank estimation (with roughly a 1-bit margin between lower and upper bounds), with a performance that is practically on-par with the Histogram algorithm: a run-time of under 1 second, for all ranks up to 2^{128}, on a standard laptop. Hence ESrank is a useful addition to the SCA evaluator's toolbox.

Acknowledgement Liron David was partially supported by The Yitzhak and Chaya Weinstein Research Institute for Signal Processing.

References

1. FIPS PUB 197, advanced encryption standard (AES), 2001. U.S. Department of Commerce/National Institute of Standards and Technology (NIST).
2. D. Agrawal, B. Archambeault, J.R. Rao, and P. Rohatgi. The EM side-channel(s). In *Cryptographic Hardware and Embedded Systems-CHES 2002*, pages 29–45. 2003.
3. 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.
4. Andrey Bogdanov, Ilya Kizhvatov, Kamran Manzoor, Elmar Tischhauser, and Marc Witteman. Fast and memory-efficient key recovery in side-channel attacks. In *Selected Areas in Cryptography (SAC)*, 2015.
5. Marios O Choudary and PG Popescu. Back to massey: Impressively fast, scalable and tight security evaluation tools. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 367–386. Springer, 2017.
6. L. David and A. Wool. A bounded-space near-optimal key enumeration algorithm for multi-subkey side-channel attacks. In *Proc. RSA Conference Cryptographers' Track (CT-RSA'17), LNCS 10159*, pages 311–327, San Francisco, February 2017. Springer Verlag.
7. Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. Making masking security proofs concrete. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 401–429. Springer, 2015.

8. D. Fledel and A. Wool. Sliding-window correlation attacks against encryption devices with an unstable clock. In *Proc. 25th Conference on Selected Areas in Cryptography (SAC)*, Calgary, August 2018.
9. Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In *Cryptographic Hardware and Embedded Systems—CHES 2001*, pages 251–261. Springer, 2001.
10. Cezary Glowacz, Vincent Grosso, Romain Poussier, Joachim Schueth, and François-Xavier Standaert. Simpler and more efficient rank estimation for side-channel security assessment. In *Fast Software Encryption*, pages 117–129, 2015.
11. Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Advances in Cryptology—CRYPTO'99*, pages 388–397. Springer, 1999.
12. Paul C Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Advances in Cryptology—CRYPTO'96*, pages 104–113, 1996.
13. Yang Li, Xiaohan Meng, Shuang Wang, and Jian Wang. Weighted key enumeration for em-based side-channel attacks. In *2018 IEEE International Symposium on Electromagnetic Compatibility and 2018 IEEE Asia-Pacific Symposium on Electromagnetic Compatibility (EMC/APEMC)*, pages 749–752. IEEE, 2018.
14. Yang Li, Shuang Wang, Zhibin Wang, and Jian Wang. A strict key enumeration algorithm for dependent score lists of side-channel attacks. In *International Conference on Smart Card Research and Advanced Applications*, pages 51–69. Springer, 2017.
15. 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. Cryptology ePrint Archive, Report 2016/609, 2016. `https://eprint.iacr.org/2016/609`.
16. Daniel P Martin, Luke Mather, and Elisabeth Oswald. Two sides of the same coin: counting and enumerating keys post side-channel attacks revisited. In *Cryptographers Track at the RSA Conference*, pages 394–412. Springer, 2018.
17. 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 *International Conference on the Theory and Application of Cryptology and Information Security*, pages 548–572. Springer, 2016.
18. Daniel P. Martin, Jonathan F. O'Connell, Elisabeth Oswald, and Martijn Stam. Counting keys in parallel after a side channel attack. In *Advances in Cryptology–ASIACRYPT 2015*, pages 313–337. Springer, 2015.
19. Jing Pan, Jasper GJ Van Woudenberg, Jerry I Den Hartog, and Marc F Witteman. Improving dpa by peak distribution analysis. In *International Workshop on Selected Areas in Cryptography*, pages 241–261. Springer, 2010.
20. Romain Poussier, François-Xavier Standaert, and Vincent Grosso. Simple key enumeration (and rank estimation) using histograms: an integrated approach. In *Proc. 18th Cryptographic Hardware and Embedded Systems–CHES 2016*, pages 61–81. Springer, 2016.
21. Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (EMA): Measures and counter-measures for smart cards. In *Smart Card Programming and Security*, pages 200–210. Springer, 2001.
22. Dan Shepherd. Quantum key search with side channel advice. In *Selected Areas in Cryptography–SAC 2017: 24th International Conference, Ottawa, ON, Canada, August 16-18, 2017, Revised Selected Papers*, volume 10719, page 407. Springer, 2018.

23. Nicolas Veyrat-Charvillon, Benoît Gérard, Mathieu Renauld, and François-Xavier Standaert. An optimal key enumeration algorithm and its application to side-channel attacks. In *International Conference on Selected Areas in Cryptography*, pages 390–406. Springer, 2012.
24. Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Security evaluations beyond computing power. In *Advances in Cryptology–EUROCRYPT 2013*, pages 126–141. Springer, 2013.
25. Shuang Wang, Yang Li, and Jian Wang. A new key rank estimation method to investigate dependent key lists of side channel attacks. In *2017 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, pages 19–24. IEEE, 2017.
26. Xin Ye, Thomas Eisenbarth, and William Martin. Bounded, yet sufficient? how to determine whether limited side channel information enables key recovery. In *Smart Card Research and Advanced Applications (CARDIS)*, pages 215–232. 2014.

Appendix: Proofs

Proof of Lemma 2:

$$\Big(\sum_{i=b+1}^{N_s-1} u_i \cdot (SI[i+1]-SI[i])\Big)/\Big(\sum_{i=b+1}^{N_s-1} l_i \cdot (SI[i]-SI[i-1])\Big) \leq \gamma^2 \qquad (8)$$

We shall prove that for all $b+1 \leq i \leq N_s-1$

$$u_i \cdot (SI[i+1]-SI[i])/l_i \cdot (SI[i]-SI[i-1]) \leq \gamma^2$$

and that will prove Equation (8). From Equation (3) and Equation (4) either $l_i = SI[j-1], u_i = SI[j]-1$ or $l_1 = u_i = SI[j]$, therefore

$$u_i/l_i \leq \gamma,$$

and we only need to prove

$$(SI[i+1]-SI[i])/(SI[i]-SI[i-1]) \leq \gamma. \qquad (9)$$

From Lemma 1, it holds that

$$SI[i+1]-SI[i] = \lfloor(\gamma-1)\cdot SI[i]\rfloor$$

and

$$SI[i]-SI[i-1] = \lfloor(\gamma-1)\cdot SI[i-1]\rfloor$$

therefore Equation (9) is

$$\frac{SI[i+1]-SI[i]}{SI[i]-SI[i-1]} = \frac{\lfloor(\gamma-1)\cdot SI[i]\rfloor}{\lfloor(\gamma-1)\cdot SI[i-1]\rfloor} \leq \frac{(\gamma-1)\cdot SI[i]}{\lfloor(\gamma-1)\cdot SI[i-1]\rfloor}.$$

By Lemma 1 Equation (9) is

$$= \frac{(\gamma-1)\cdot\lfloor\gamma\cdot SI[i-1]\rfloor}{\lfloor(\gamma-1)\cdot SI[i-1]\rfloor} = \frac{(\gamma-1)\cdot\lfloor\gamma\cdot SI[i-1]\rfloor}{\lfloor\gamma\cdot SI[i-1]-SI[i-1]\rfloor}.$$

Since $SI[i-1]$ is an integer it holds

$$= \frac{(\gamma-1)\cdot\lfloor\gamma\cdot SI[i-1]\rfloor}{\lfloor\gamma\cdot SI[i-1]\rfloor - \lfloor SI[i-1]\rfloor} = \gamma - 1 + \frac{(\gamma-1)\cdot\lfloor SI[i-1]\rfloor}{\lfloor\gamma\cdot SI[i-1]\rfloor - \lfloor SI[i-1]\rfloor}$$
$$= \gamma - 1 + \frac{(\gamma-1)\cdot SI[i-1]}{\lfloor\gamma\cdot SI[i-1] - SI[i-1]\rfloor} = \gamma - 1 + \frac{(\gamma-1)\cdot SI[i-1]}{\lfloor(\gamma-1)\cdot SI[i-1]\rfloor} \leq \gamma.$$

□

Proof of Lemma 3:

$$\Big(\big(\sum_{i=1}^{b-1} u_i\big) + u_b\cdot(SI[b+1]-SI[b]) + u_{N_s}\Big) \Big/ \Big(\sum_{i=1}^{b} l_i\Big) \leq \gamma^2. \tag{10}$$

We can write this equation in the following way:

$$\frac{\big(\sum_{i=1}^{b-1} u_i\big) + u_b\cdot(SI[b+1]-SI[b]) + u_{N_s}}{\sum_{i=1}^{b} l_i} = \frac{\sum_{i=1}^{b} u_i}{\sum_{i=1}^{b} l_i} + \frac{u_b\cdot(SI[b+1]-SI[b]-1) + u_{N_s}}{\sum_{i=1}^{b} l_i}.$$

Since $u_i/l_i \leq \gamma$, Equation (10) is upper bounded by

$$\leq \gamma + \frac{u_b\cdot(SI[b+1]-SI[b]-1) + u_{N_s}}{\sum_{i=1}^{b} l_i}.$$

By Lemma 1 and since $l_1 \geq l_2 \geq ... \geq l_b$ and $u_b \geq u_{N_s}$, we get

$$\leq \gamma + \frac{u_b\cdot((\gamma-1)\cdot SI[b]-1) + u_b}{b\cdot l_b}.$$

Since $SI[b] = b$ and $u_i/l_i \leq \gamma$

$$\leq \gamma + \frac{\gamma\cdot l_b\cdot(\gamma-1)\cdot b}{b\cdot l_b} = \gamma + \gamma\cdot(\gamma-1) = \gamma^2.$$

□