Comparing Key Rank Estimation Methods

Rebecca Young[1], Luke Mather[2], and Elisabeth Oswald[1,3][0000−0001−7502−3184]

[1] Department of Computer Science, University of Bristol, UK
[2] PQShield, Bristol, UK
[3] Digital Age Research Center (D!ARC), University of Klagenfurt, Austria

Abstract. Recent works on key rank estimation methods claim that algorithmic key rank estimation is too slow, and suggest two new ideas: replacing repeat attacks with simulated attacks (PS-TH-GE rank estimation), and a shortcut rank estimation method that works directly on distinguishing vector distributions (GEEA). We take these ideas and provide a comprehensive comparison between them and a performant implementation of a classical, algorithmic ranking approach, as well as some earlier work on estimating distinguisher distributions. Our results show, in contrast to the recent work, that the algorithmic ranking approach outperforms GEEA, and that simulation based ranks are unreliable.

Keywords: Key rank, Estimation

1 Introduction

From a real-world adversary's point of view, a side channel attack is successful if it reveals enough information about the unknown secret key such that this key can be found via a biased brute-force key search. Such a biased brute-force key search works by ranking all keys according to their distinguishing scores. This pragmatic viewpoint is also taken in the context of formal evaluations where the actual demonstration of an attack may include the argument of how much effort remains for a biased brute-force search.

The computational effort for an adversary to perform a biased brute-force key search, which can be estimated by an evaluator by calculating the position of a known secret key within the (ranked) key space, were first addressed by the academic community in [15, 16]. Then, in quick succession, a number of better (faster and tighter) key rank and key enumeration algorithms appeared, e.g. [1,6,7,9,12] to name some approaches. The algorithms by [6] and [12] enable both key enumeration as well as key ranking, and it was later shown that they are mathematically equivalent [10]. There also exist several improvements to existing algorithms; the algorithm by [12] in particular was improved in [9] but also in [11], and there exists a fast implementation [6] by [13]. There is also [3], which is of comparable speed as [6], but enables better bounds. A short cut estimator for the average key rank called GM (for "Massey Guessing Entropy") was proposed in [2] to deal with long keys. This estimator turns out to be highly problematic in the experiments by [17].

1.1 The challenge in practice

A *single run* of any existing ranking algorithm is unproblematic: for a key space associated with a typical symmetric encryption algorithm, even a few minutes of computation time are of no concern in an evaluation. However memory and time requirements for the mentioned ranking algorithms don't scale linearly, and a single side channel experiment is insufficient to judge an implementation (unless it is trivially vulnerable). Consequently, the challenge is to have an approach that enables to estimate key ranks over many (repeat) experiments for potentially long keys very quickly.

Many Ranks: a single experiment can only give "circumstancial evidence", thus repeating experiments, and computing some (summary) statistics is necessary for a sound interpretation. Beside the question of *which statistic to use to report outcomes*, the challenge in practice is to produce *multiple experiments* within the time and financial constraints of a product evaluation cycle.

Long Keys: whilst a typical symmetric algorithm takes 128 bits of key material, long term security requires the use of 256 bits of key material in the symmetric setting. The idea of biased brute-force attacks also applies to asymmetric cryptography, which then leads to the requirement of dealing with keys that are considerably longer than 128 bits. Thus the challenge in a practical evaluation setting is to also *deal with long cryptographic keys*.

1.2 Recent conceptual advances

The latest work [17] proposes two ideas to make average key rank computations[4] faster, which they call PS-TH-GE (pseudo-theoretic GE) and GEEA (GE estimation algorithm).

The idea behind the PS-TH-GE goes back to [4, 5, 8, 14], which all observed that it is possible to statistically characterise the distribution of distinguishing vectors resulting from correlation, distance of means, and template attacks. Whilst the previous work [14] derived the distributions for specific distinguishers, [17] suggest to simply use the plug in estimator (empirical mean and covariance) based on repeat attack samples from the actual device. Thus, [17] can deal with any "additive" distinguisher. This is a potential solution for the problem of needing many experiments: instead of (re)sampling distinguishing vectors from attacks on real device data, the suggestion is to sample them from simulations that are based on the real device data.

The additional idea behind the GEEA is instead of estimating the full distribution of distinguishing vectors, it suffices to estimate the distribution related to "pair-wise" success rates. Then the distribution over arbitrarily many distinguishing vectors (representing a full key rather than a single subkey) can can be easily derived (i.e. long keys are easy to deal with), and from this distribution, a key rank estimate can be produced by sampling from this distribution.

[4] The average key rank is often also referred to as the key guessing entropy, or GE.

1.3 Assumptions, gaps in knowledge, and our contribution

The premise behind [17], but also other recent works like [2] is that even the fastest ranking algorithms [12] and [6] (despite the various optimisations) are still too slow to be of practical use in the context of long keys and also repeat experiments. In [17] the cost of running a histogram based ranking algorithm is given as roughly 17s for a 16-byte key. Many of the reported experiments in [17] are for either single byte keys or short keys (where calculating the key rank is trivial)[5]. Only a few experiments are for realistic key sizes, where notably a comparison with ranks derived (algorithmically via e.g. [6] or [12]) from actual attacks for long keys are missing. Furthermore, a direct comparison with the previous work of [14] is missing as well: the previous work [14] derived explict expressions for distinguishers like correlation and templates, whereas [17] base most experiments on deep learning distinguishers where we do not have explicit formulas for the distinguishing vectors. It is thus unclear how the use of the plug-in estimator (as suggested by [17]) compares to the carefully derived estimators from Rivain.

The lack of substantial large key experiments, and experiments with known distinguisher distributions creates multiple gaps and makes it impossible to understand how well the PS-TH-GE and GEEA perform in comparison to a well implemented classical ranking algorithm, and/or using estimators for known distinguisher distributions. It is also unclear how good simulation-based repeat ranks are in contrast to fresh-attack-based repeat ranks.

We believe that this creates an overall gap to the needs of real world evaluations where the rank of the full key matters, and guarantees are required about the behaviour of any ranks derived from estimated distinguisher distributions. Our contribution aims to narrow this gap. First we compare our customised implementation of an open-source algorithmic ranking algorithm (based on [9]) with an implementation of GEEA in Section 2. We then compare ranks based on Rivain's method as well as the methods by Zhang et al. (PS-TH-GE and GEEA) with each other as well as with ranks based on fresh data in Section 3 (we use simulations for this purpose). For completeness we also work with some real device data in Section 4.

Our results show that the (optimised) algorithmic ranking based on [9] is significantly faster than GEEA, but more importantly that it can cope with keys of up to 4096 bits. There is in fact a severe performance penalty "hidden" in the GEEA algorithm, which is how many keys need to be sampled (the parameter M), see Sect. 2. We then show that in the context of classical distinguishers (correlation and templates) both PS-TH-GE and GEEA offer considerably less accurate ranking results than using a performant ranking algorithm implementation. We also observe that Rivain's formulas for correlation and template based distinguishing vectors are delivering by far better results than just using the plug-in estimator i.e. using the PS-TH-GE. We derive these conclusions from

[5] The style of experiments in [17] is in line with the single byte experiments in previous work such as [14].

experiments based on simulated data, in Sect. 3. We demonstrate the same behaviour based on data from a real device in Sect. 4, and provide a more in-depth conclusion in Sect. 5.

2 Classical Key Rank Estimation vs. GEEA

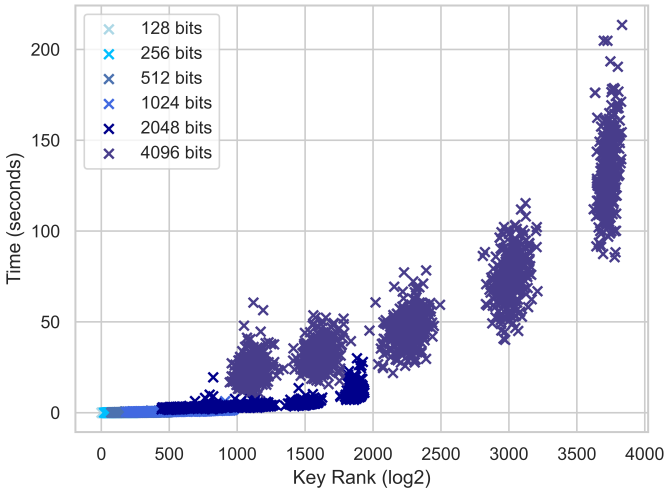
We want to challenge the premise of [17] and [2] that algorithmic key rank implementations are not performant enough to be used "at scale" during evaluations. To challenge this premise, we took the open source library https://github.com/sca-research/labynkyr that was released alongside [9]. In their work [9] examine a large number of key rank implementation options (for the purpose of enumeration) and provide optimised C++ libraries to support large scale, parallel key enumeration experiments. We extracted the key ranking part from it, and wrote our own interface for the library so that we could use it conveniently with both simulated and real data. In order to cater for very long keys, we use Boost's multi-precision library to efficiently accumulate the large rank values. Finally we took advantage of the concept of compile time evaluation, where one can supply the compiler with the values of certain variables and have it "pre-compute" the output of relevant functions, with the aim of achieving some (slight) performance advantage during run time (e.g. because we know the dimension of any specific key rank experiment a priori, we can compile for these specific dimensions.)

All our experiments were conducted on standard computing equipment, we use laptops (1.6 GHz and 2.3 GHz), and the implementations are all single threaded.

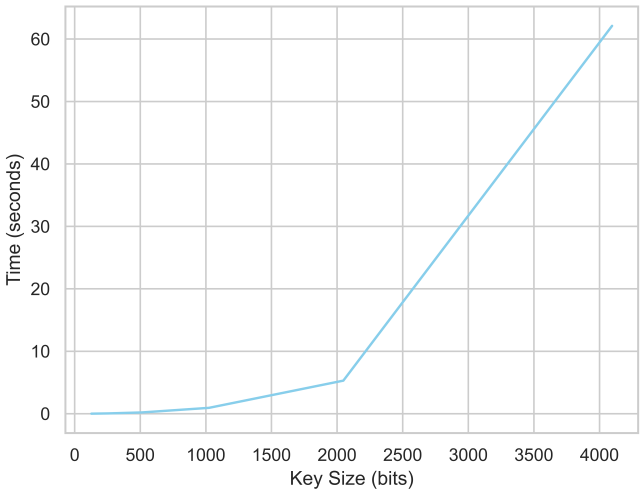
2.1 Evaluating a modern algorithmic ranking algorithm

We then ran a set of simulated attacks in order to quickly generate distinguishing scores to profile the performance of our ranking implementation. In the simulated attacks, all distinguishing scores are generated via standard DPA style attacks using correlation as a distinguisher. We fixed the simulated device leakage model as well as the adversarial prediction model to Hamming weight, used the usual AES SubBytes as a target function, and varied the additive Gaussian noise to "shift" keys (when fixing the number of attack traces). In this way we were able to generate a very large number of repeat AES attack simulations (for a single subkey), which we used to generate data for attacks with keys of increasing length (by simply grouping single key experiments). We went from the standard AES key size of 128 bits, up to 256 bits, and then further up using typical asymmetric key sizes 512, 1024, 2048 and 4096, by simply increasing the number of columns. For each of these key sizes we took three random keys, and went through simulations with added noise creating SNR values ranging from 0.1 up to 0.5. In total we performed 9000 key rank experiments on our simulated data.

Figures 1a and 1b show the results. In both plots, the x-axis represents the rank of a returned key, and the y-axis the time that it took to compute its rank. Each `x` represents on outcome, and we indicated groupings based on the size of



(a) Distribution of execution time across increasing key space size.



(b) Average execution time across increasing key space size.

Fig. 1: Key rank performance evaluation

the key space with different shades of blue. The plot in Fig. 1a shows the time that it takes for one key rank experiment to finish. Across all key space sizes, keys which have a low rank (thus are easy to find) are returned faster, and there is a small gradual increase for deeper keys. Obviously, for very long keys (4096 bits), there is a more marked time difference for key rank experiments returning likely vs unlikely keys. The slowest experiments are, obviously, the experiments that return unlikely keys from the 4096 bit case, where it took up to 220 seconds to complete a single run of key rank. Short keys, such as 128 bit keys, take in the worst case 0.01 seconds to complete. The plot in Fig. 1b shows a simple arithmetic average over the execution time for each key space size: the time does not scale linearly, but key lengths of interest are all within "easy reach".

Our experiments dispel the myth that algorithmic key ranking is too slow for practice. For short and medium keys (128-2024 bits) the time to return a single key rank (including deep keys) is extremely short. For very long keys (4096), the time to return a single (deep key) experiment is such that repetitions are practically possible.

2.2 Evaluating GEEA performance characteristics

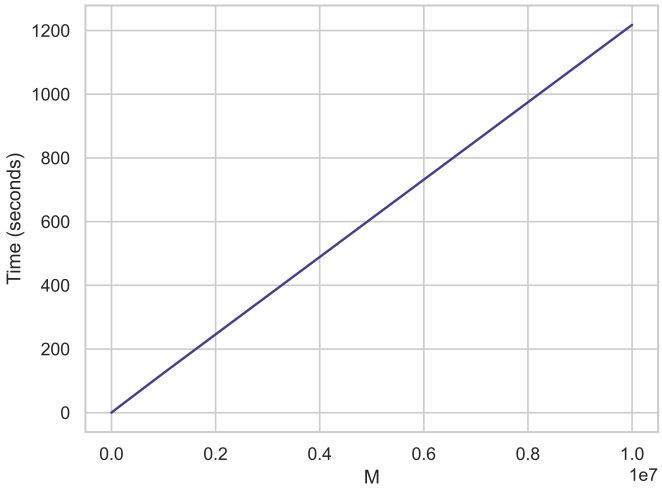
The trick behind GEEA is that one can compute the GE based on working with comparison vectors (i.e. the difference between the score of the correct key and the incorrect key candidates), which enables writing the GE as a sum of "pair-wise" comparison scores. This trick helps to reduce the number of parameters that need to be estimated (co-variances between key candidates are now ignored), and enables us to easily move from the distribution associated with a single subkey to the full key distribution. The resulting GE estimation algorithm is given as Alg. 1 in [17].

A crucial parameter of the GEEA algorithm is the factor M, which is the number of samples that one takes from the full key distribution. Each sample of the distribution gives an estimated value for the key rank. As in all key rank estimation approaches, a single sample only gives circumstantial evidence. Thus more than one sample is required, which leads to the immediate question of how many samples M are necessary to get an accurate key rank estimation?

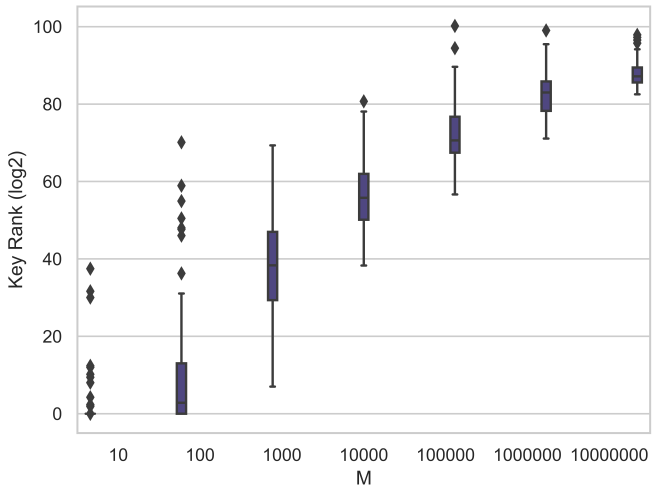
In their paper [17] say that choosing M in the order of 10^7 leads to good results, but in their experiments they set $M = N$ (N would be the number of repeat attacks in a conventional setting). In the context of their concrete experiments, this results in N being in the order of $10,000$, which is evidently much smaller than 10^7.

Our GEEA implementation runs in Python, and we ensured to match the speed of [17]: they report that GEEA for $M = 1$ takes about 10^{-4}s). The most costly part of computing GEEA is the CDF computation: for each sample, this look-up is required, thus the cost of GEEA is directly proportional to M.

Figures 2a and 2b show two representative results from running GEEA with an increasing value of M for a key with 16 bytes. Figure 2a shows the linear increase of the execution time as a function of M. Figure 2b side shows the



(a) GEEA performance cost with increasing M



(b) GEEA rank outcomes with increasing M.

Fig. 2: GEEA performance evaluation as M increases.

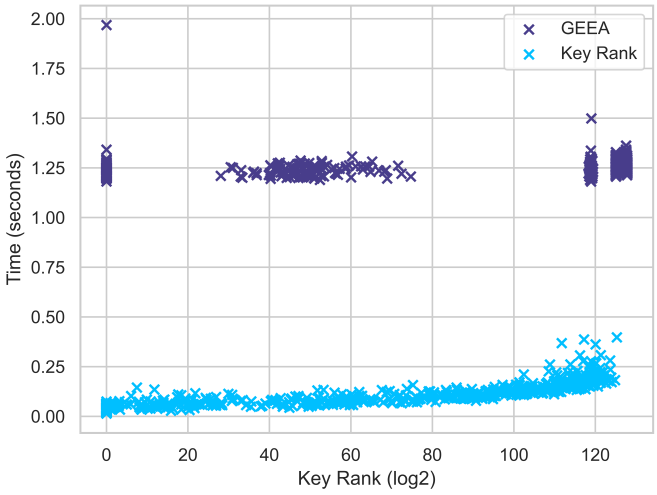


Fig. 3: GEEA vs Key Rank

distribution of the ranks for increasing values of M. We used a box plot to visualise the distribution. A box plot gives a "five number summary" and contains a box and "whiskers". The box represents where most of the ranks sit, with the middle line being the median (the boundaries are given by the first and third quartile). The whiskers represent the first/third quartile plus 1.5 of the interquartile range, and any additional points are the outliers. Figure 2b shows very clearly how the rank estimation quality of GEEA changes as M increases.

We argue that at the very least M needs to be chosen equal or larger than $10,000$ for any reasonable estimation quality, which would indicate that a single run of GEEA is considerably slower than an equivalent run of our good implementation of an algorithmic ranking algorithm. Figure 3 provides evidence for this claim: we compared the execution time for ranking keys (16 byte full key) at different depths of the key space. GEEA was always slower than algorithmic key rank. Perhaps with a more optimised implementation of GEEA (e.g. switching to C or C++, and using an optimised implementation for the CDF computation, which we do via a standard library call) one could bring a GEEA with a reasonable choice for M to the performance of key rank for a 16 byte key. Evidently though, we find no supporting evidence that GEEA would outperform algorithmic ranking.

3 Challenging Simulation based Key Ranks

The idea of using simulated distinguishing scores to speed up success rate estimations dates back to [14] who derived the distribution of the distinguishing

vectors of correlation and template based DPA style attacks. Using the distribution of such distinguishing vectors [14] demonstrate that drawing from these distributions, rather than conducting fresh experiments (aka experiments that require the measurement of new traces from a device), enables a very accurate prediction of success rates for the corresponding attacks. Three distributions were characterised: two modified correlation coefficients $\dot{\rho}$ and $\ddot{\rho}$, as well as a least squares estimate L_k.

The key insight by [17] was to notice that the distribution of any "additive distinguisher" [14] can be characterised by a multivariate normal distribution (MVN) and that the parameters of the MVN can be estimated by the plug-in estimators (i.e. by computing the average and the empirical co-variance matrix describing a distinguishing vector directly by observing multiple such vectors arising from attacks on real traces). Once the statistical characterisation has been obtained, further attack outcomes can be simulated without having to take new measurements from a device (the idea of simulating attacks from the estimated (aka theoretic) distributions gives rise to the name "pseudo-theoretic", leading to the name PS-TH-GE).

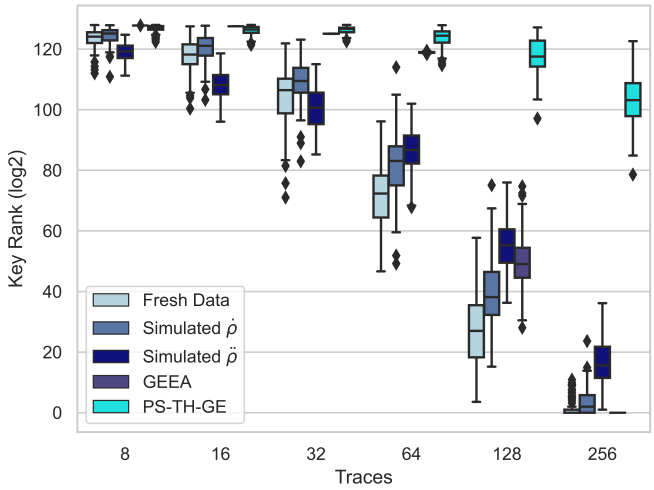
3.1 Theoretical differences between approaches

At face value, the PS-TH-GE seems like a natural extension of Rivain's approach, but there is a subtle yet important difference. The characterisation of the distinguisher distributions in Rivain's paper enables the estimation of the MVN characterising the distinguishing vectors directly from *trace* data. This means that in a profiling trace set with N traces, we use all N traces for estimating the parameters of the distinguishing vector resulting from an attack.

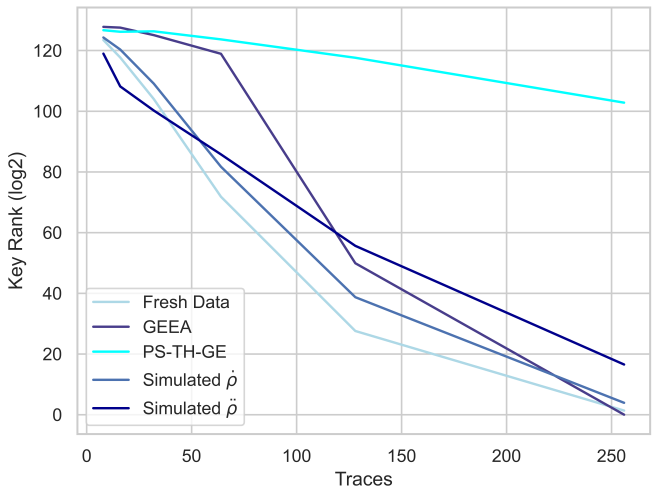
In the approach by [17] we must divide our profiling trace set by q to generate q independent experiments. Each experiment then generates a distinguishing vector for an attack with N/q traces, and we use these distinguishing vectors to estimate the parameters describing the distinguishing vector distribution. This holds for both the PS-TH-GE as well as for GEEA.

Thus the estimator of Rivain uses a single profiling set of size N, but the empirical estimator (i.e. in the context of the PS-TH-GE and GEEA) requires the use of multiple profiling sets of size N/q.

All of these discussed estimations are thus based on a set, i.e. a finite amount of available data for profiling. From a statistical perspective, one can thus use them to make predictions about attack outcomes using up to some number of traces: N for Rivain's approach and N/q for the PS-TH-GE and GEEA. It is not clear that it is possible to use the estimations for attacks with more than N, resp. N/q traces, and we will demonstrate in a practical experiment that the quality of the estimations gets poorer when trying to predict outcomes of attacks with more than N, and resp. N/q traces.



(a)



(b)

Fig. 4: Comparison between key rank estimation algorithms for correlation attacks.

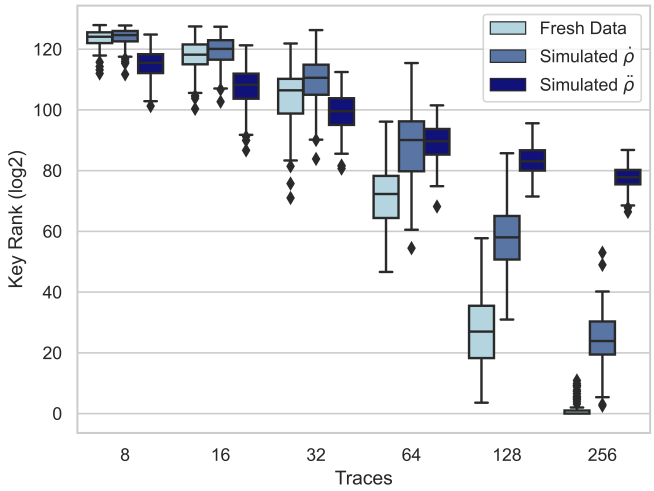


Fig. 5: Average key rank estimations using correlation based models, initial trace set of 32 traces

3.2 Practical differences between approaches: correlation

We first focus on a comparison between the approaches in the context of a typical correlation based attack. We based our experiments on simulations (16 key bytes), which enable us not only to fully control the attacks (and thereby key depths) but also to do many attacks extremely quickly for all rank estimation methods. We use the same type of simulations as described before (HW leakage model, Gaussian noise) and set the noise to achieve a signal to noise ratio of 0.1 (we performed experiments with different SNRs that all show the same outcomes, thus only include a subset of them). By varying the number of attack traces, we can then shift the rank of the correct key.

For the algorithmic ranking on the fresh data, as well as the simulation based experiments using Rivain's formulas for $\dot{\rho}$ and $\ddot{\rho}$ we use all N traces (given along the x-axis). For PS-TH-GE and GEEA, we took N/q with $q = 8$ if $N > 8$; for $N = 8$ we use $q = 4$.

Figures 4a and 4b show the outcomes of our comparison. Figure 4a shows box plots for all the ranking approaches, which makes it clear that none of the approaches based on simulated distinguishing vectors (i.e. all approaches bar classical ranking on fresh data) underestimate the average rank and also the "lucky adversary". Figure 4b only shows the average ranks: the picture here also makes clear that ranks based on simulations show a higher rank on average. However, the additional distributional information, which makes the shortcomings of the simulation based ranks perhaps even clearer is now not available.

We argue that these outcomes not only demonstrate that simulation based ranks underestimate the average case adversary, but they also demonstrate the need to report more statistical information than just the average, because lucky adversaries are important for practical security.

Next we want to know how much the number of initial profiling traces N impacts on the outcomes of simulated key ranks. We set up an experiments where we estimated $\dot{\rho}$ and $\ddot{\rho}$ based on $N = 32$ traces, and the performed key rank estimations using Rivain's formulas (N is a parameters in these formulas) for varying N. Figure 5 shows that simulated key ranking experiments with $N \leq 32$ well approximate the actual key rank (as per freshly sampled attack data), whereas key ranking experiments with $N > 32$ significantly diverge from the true key rank (they underestimate the vulnerability).

These experiments show clearly that if simulation based ranking would be considered in practical security evaluations, then it is clearly preferable to use the precisely characterised distributions from Rivain over a simple plug-in estimator based approach, which underpins the PS-TH-GE. They also demonstrate the need for ensuring plenty of profiling traces (or examples of attack outcomes when using the PS-TH-GE), because simulated ranks tend to underestimate the power of the adversary. Thus a recommendation would be to "overdimension" the profiling data set, which will enable to generate reasonably good simulations for attacks with a number of traces that is smaller than the number of traces used for profiling.

3.3 Practical differences between approaches: Gaussian templates

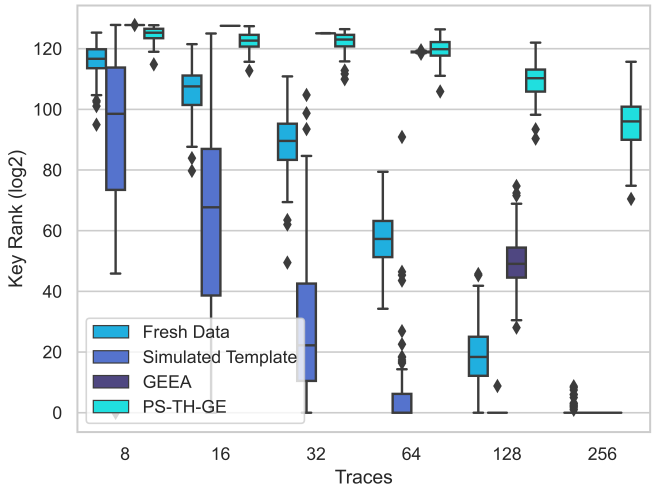
Rivain also gives the precise characterisation for a Gaussian template based distinguisher. Using the same approach as described before, we ran experiments to compare several key rank estimations. We note at this point that because full rank covariance matrices are often not-invertible, we only keep the diagonal of the covariance matrix.

Figures 6a and 6b show the outcomes of these experiments. It is clear from the outcomes that all simulation based key ranks are severely misestimating the key rank outcomes: both in terms of average ranks as well as the lucky adversary.

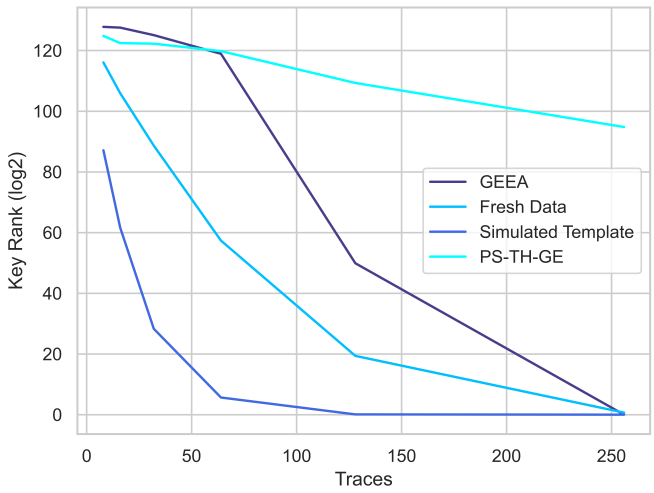
These outcomes are particularly interesting because templating is of importance in practical security evaluations, and it appears that whilst the PS-TH-GE is severely overestimating the key rank, using the characterisation by Rivain appears to severely underestimate the key rank.

4 Real trace experiments

The simulations in the previous sections are informative and representative for key rank outcomes. We add one more experiment, this time using a public trace set corresponding to an AES implementation on an ARM Cortex M3, which we took from the various available trace sets on the public repository Zenodo

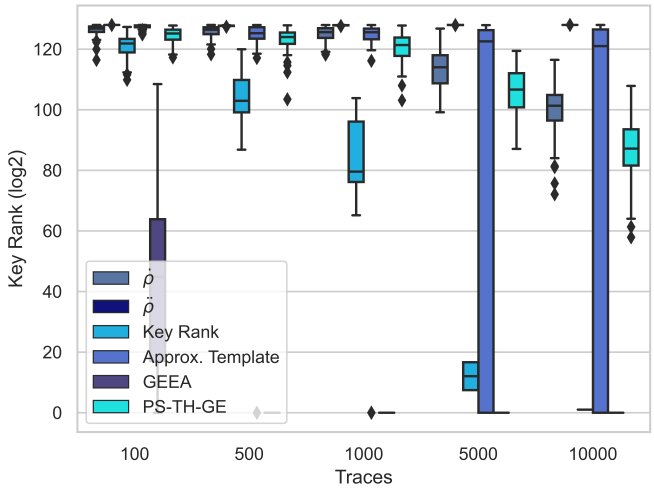


(a)

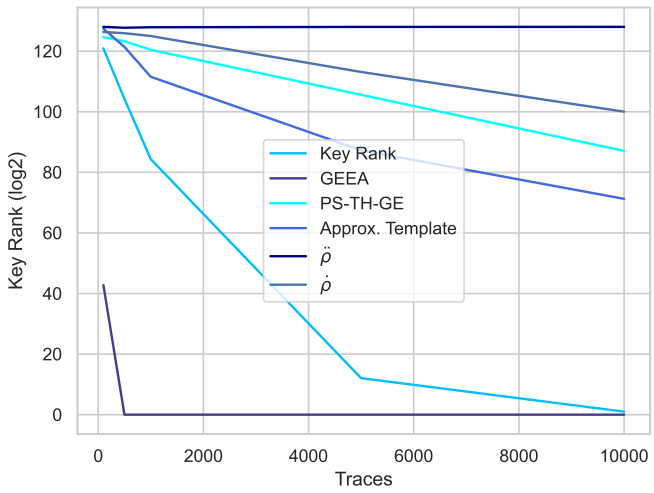


(b)

Fig. 6: Comparison between key rank estimation algorithms for Gaussian template attacks.



(a)



(b)

Fig. 7: Comparison between key rank estimation algorithms based on real data (8-bit AES implementation).

https://doi.org/10.5281/zenodo.5710205. The data features traces for multiple keys, and has two rounds of AES (8-bit, table look-up based implementation).

We conduct a classical differential style attack using the SubBytes output as a target. We compared the outcomes of GEEA and our algorithmic ranking implementation, over an increasing number of traces. The rank estimation outcomes from these attacks are show in Fig. 7a and 7b. In accordance to the experiments based on simulated data, GEEA misrepresents the strength of the adversary, which is problematic in an evaluation context. We see once more that reporting the distribution of the key rank estimations gives a more nuanced picture of the strength of attacks at certain levels of number of attack traces.

5 Conclusion

Our contribution analyses and challenges in some ways the premise of some of the most recent works on key rank estimation. Two assumptions were made in previous work: firstly, that algorithmic key ranking algorithms are too slow to deal with long keys or many experiments and secondly, that it is possible to create synthetic experiments based on characterising a single profiling dataset. Our own algorithmic ranking implementation, which is a more easily useable and customisable version of an open source library for parallel key enumeration, outperforms the supposedly best competitor GEEA, at all key lengths and depths. We also show that it is possible to rank keys with up to 4096 bits.

Using our performant algorithmic ranking algorithm, we then challenge the recent claims about the accuracy of GEEA and PS-TH-GE, as well as a much earlier approach by Rivain. We show that Rivain's approach, which is limited to classical distinguishers, is considerably better than PS-TH-GE and GEEA for these classical distinguishers. We find that care needs to be taken when simulating attack outcomes, no matter whether the simulations use synthetic vectors based on Rivain's approach or the more generic estimation underpinning the PS-TH-GE and GEEA. If synthetic outcomes are produced by an initial dataset containing N traces, then synthetic attacks for classical distinguishers, with a trace complexity of up to N are reasonably accurate at least in simulations.

There is an open question as to how big does the profiling dataset need to be to derive good enough estimations for the distinguisher distribution? In our experiments we used profiling sets of a size that would enable full key recovery with near certainty. However, the estimated distinguishing vector distributions from these datasets seemingly do not enable the creating of synthetic attack outcomes that match reality for *full keys* (i.e. outcomes that we observe in repeat experiments with fresh traces). There is an explanation for this: full key distributions arise from the joint distribution of *many* subkey distributions. Any small divergence between the estimations for a subkey thus multiply over the many subkey and turn into a significant divergence in terms of the full key distribution.

This indicates that the profiling dataset needs to be considerably larger than the dataset that we are interested in terms of attacks. It remains an open question how how much larger it would need to be. But our results indicate that

there is perhaps much less advantage in the idea of using synthetic attack outcomes because for the (preceding) estimation, considerably more data than for a successful attack has to be acquired. As a result one may as well stick to the current practice of aiming for several repeat experiments over which some key rank summary statistics are computed.

References

1. 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.
2. 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.
3. Liron David and Avishai Wool. Poly-logarithmic side channel rank estimation via exponential sampling. In Mitsuru Matsui, editor, *CT-RSA*, volume 11405 of *Lecture Notes in Computer Science*, pages 330–349. Springer, 2019.
4. Yunsi Fei, A. Adam Ding, Jian Lao, and Liwei Zhang. A statistics-based success rate model for DPA and CPA. *J. Cryptogr. Eng.*, 5(4):227–243, 2015.
5. Yunsi Fei, Qiasi Luo, and A. Adam Ding. A statistical model for DPA with novel algorithmic confusion analysis. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*, pages 233–250. Springer, 2012.
6. Cezary Glowacz, Vincent Grosso, Romain Poussier, Joachim Schüth, and François-Xavier Standaert. Simpler and more efficient rank estimation for side-channel security assessment. In *International Workshop on Fast Software Encryption*, pages 117–129. Springer, 2015.
7. Vincent Grosso. Scalable key rank estimation (and key enumeration) algorithm for large keys. In *International Conference on Smart Card Research and Advanced Applications*, pages 80–94. Springer, 2018.
8. Victor Lomné, Emmanuel Prouff, Matthieu Rivain, Thomas Roche, and Adrian Thillard. How to estimate the success rate of higher-order side-channel attacks. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, volume 8731 of *Lecture Notes in Computer Science*, pages 35–54. Springer, 2014.
9. 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 Cryptology ePrint Archive*, 2016:609, 2016.
10. 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.
11. 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.

12. Daniel P Martin, Jonathan F O'connell, Elisabeth Oswald, and Martijn Stam. Counting keys in parallel after a side channel attack. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 313–337. Springer, 2015.
13. 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.
14. Matthieu Rivain. On the exact success rate of side channel analysis in the gaussian model. In *International Workshop on Selected Areas in Cryptography*, pages 165–183. Springer, 2008.
15. 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*, volume 7707 of *LNCS*, pages 390–406. Springer, 2012.
16. Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Security Evaluations beyond Computing Power. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT*, volume 7881 of *LNCS*, pages 126–141. Springer, 2013.
17. Ziyue Zhang, A Adam Ding, and Yunsi Fei. A fast and accurate guessing entropy estimation algorithm for full-key recovery. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 26–48, 2020.