IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2022, No. 4, pp. 886–905. DOI:10.46586/tches.v2022.i4.886-905

GE vs GM: Efficient side-channel security evaluations on full cryptographic keys

Anca Rădulescu, Pantelimon G. Popescu and Marios O. Choudary [*]

University Politehnica of Bucharest, Bucharest, Romania
anca_radulescu96@ymail.com,pgpopescu@yahoo.com,marios.choudary@cs.pub.ro

Abstract.
Security evaluations for full cryptographic keys is a very important research topic since the past decade. An efficient rank estimation algorithm was proposed at FSE 2015 to approximate the empirical guessing entropy remaining after a side-channel attack on a full AES key, by combining information from attacks on each byte of the key independently. However, these could not easily scale to very large keys over 1024 bits. Hence, at CHES 2017, it was proposed a new approach for scalable security evaluations based on Massey's guessing entropy, which was shown tight and scalable to very large keys, even beyond 8192 bits. Then, at CHES 2020, it was proposed a new method for estimating the empirical guessing entropy for the case of full-key evaluations, showing also important divergences between the empirical guessing entropy and Massey's guessing entropy. However, there has been some confusion in recent publications of side-channel evaluation methods relying on these two variants of the guessing entropy. Furthermore, it remained an open problem to decide which of these methods should be used and in which context, particularly given the wide acceptance of the empirical guessing entropy in the side-channel community and the relatively little use of the other.
In this paper, we tackle this open problem through several contributions. First of all, we provide an unitary presentation of both versions of the guessing entropy, allowing an easy comparison of the two metrics. Secondly, we compare the two metrics using a set of common and relevant indicators, as well as three different datasets for side-channel evaluations (simulated, AVR XMEGA 8-bit microcontroller and a 32-bit device). We used these indicators and datasets also to compare the three full-key evaluation methods from FSE 2015, CHES 2017 and CHES 2020, allowing us to provide a clear overview of the usefulness and limitations of each method. Furthermore, our analysis has enabled us to find a new method for verifying the soundness of a leakage model, by comparing both versions of the guessing entropy. This method can be easily extended to full-key evaluations, hence leading to a new useful method for side-channel evaluations.

Keywords: side-channel attacks · security evaluations · guessing entropy · bounds · scalability

1 Introduction

Power and electromagnetic analysis attacks are powerful tools to extract secret information from hardware devices, such as modern System-on-Chip processors used in smpartphones or the cryptographic microcontrollers used in banking smartcards. These side-channel attacks apply a divide-and-conquer strategy, such that they are able to target each subkey byte of a cryptographic algorithm independently. This may allow an attacker to mount a

[*]We thank Christ our God for this work.

Received: 2022-04-15 Accepted: 2022-06-15 Published: 2022-08-31

practical attack on a block cipher such as AES, when using a key of 128 or 256 bits (16 or 32 bytes, respectively), by targeting each of the 16 or 32 key bytes independently, whereas a purely brute-force search attack on the full key is computationally infeasible.

While classic side-channel attacks focused on attacking a single key byte, recent advances in side-channel attacks have focused on the problem of estimating the rank[1] of the full key of a cryptographic algorithm, after obtaining sorted lists of probabilities for the different subkeys that compose the full key (e.g. lists for the 16 subkey bytes of AES, when used with a 128-bit key). These algorithms represent very useful tools for security evaluators that need to estimate the security of a given device.

One of the first approaches towards estimating the rank of a full 128-bit key was proposed by Veyrat-Charvillon et al. [CNG+13], albeit with a considerable error margin. Afterwards, other algorithms [GCG+15, MPO+15, BJL+15] have reduced the bounds of this estimation to within one bit for 128-bit keys and can run within seconds of computation, after being given with a list of sorted probabilities for the individual subkeys. One of the most efficient such algorithms is the one by Glowacz et al. presented at FSE 2015 [GCG+15], which allows estimating the empirical guessing entropy introduced by Standaert et al. [SMY09] in the full-key scenario. However, these algorithms cannot easily scale for large keys composed of more than 128 bytes (e.g. an RSA 2048 or 4096 bit key), while at the same time providing tight bounds.

In this context, Choudary and Popescu [CP17] presented at CHES 2017 a new approach, based on mathematical bounds of Massey's Guessing Entropy [Mas94], to bound the guessing entropy remaining after a side-channel attack for very large cryptographic keys (or other secret data). They showed that their method works for keys of 8192 bits (1024 bytes) and beyond, in almost constant time and memory, which none of the other methods could do.

Then, at CHES 2020, Zhang et al. [ZZD+20] presented a new approach for estimating the empirical guessing entropy in full-key evaluations, that may provide advantages over previous approaches such as the FSE 2015 method.

Concurrent to these publications and also within the paper of Zhang et al., there have been several observations regarding the limitations of each method and in particular the observation that the empirical guessing entropy and Massey's guessing entropy may diverge considerably, leading to substantially different results [GV18, DW19, AMP+19, ZZD+20]. Nevertheless, it remained an open problem to decide which of the two guessing entropies that are estimated by the full-key evaluation methods cited above (FSE 2015, CHES 2017, CHES 2020) is better suited for side-channel evaluations.

In this paper, we give an answer to this open problem, by attentively presenting and comparing the two guessing entropy metrics as well as these three full-key estimation algorithms. We show both analytically and experimentally the advantages and weaknesses of each method, by using a set of common indicators and three side-channel datasets: a simulated dataset, traces from the hardware AES engine of the AVR XMEGA microcontroller and traces from a software bitsliced AES implementation on a 32-bit device.

This analysis enables us to provide a guide that can help us decide which method to use for various side-channel security evaluations. Furthermore, our analysis has enabled us to develop a new method for deciding whether a leakage model is sound, by actually combining, rather than choosing, the two variants of the guessing entropy.

In the next section we provide some background on full-key side-channel evaluations. Then, in Section 3 we provide a detailed presentation of the two guessing entropy variants used in the side-channel literature: Massey's guessing entropy and the empirical guessing entropy. Then, in Section 4 we present the three full-key evaluation methods analysed in this paper (FSE 2015, CHES 2017, CHES 2020). In Section 5 we present our evaluation

[1]The position of the correct key in a list sorted by the side-channel attack algorithm. The higher the correct key in the list, the better the side-channel algorithm performs and hence the worst the security protection level.

context: the key indicators used for comparing implementations as well as our datasets and the method of Template Attacks used to implement our side-channel attacks. Sections 6 and 7 present our analysis for the guessing entropy metrics as well as for the full-key evaluation methods, respectively.

2 Background: security evaluations for side-channel attacks on full cryptographic keys

Given a physical device (e.g. a smartcard) that implements a cryptographic algorithm, such as AES, we may record side-channel traces (power consumption or electromagnetic emissions) using an oscilloscope. In this case, for each encryption of a plaintext p_i with a key $k\star$, we can obtain a leakage trace $\mathbf{x}_i$ that contains some information about the encryption operation.

For the particular case of AES and other similar block ciphers that use a substitution box (S-Box), a common target for side-channel attacks is the S-box operation $v = \text{S-box}(k\star \oplus p)$ from the first round of the block cipher. Since this operation is done for each subkey $k\star$ in part (for AES each subkey only has 8 bits), we can attack each of the subkeys separately. And by using information from the leakage traces, a side-channel attack such as DPA [KJJ99], CPA [BEC+04] or Template Attacks [CRR02] can assign higher probabilities to the correct subkeys, leading to a very powerful brute-force search on each subkey.

After obtaining the lists of probabilities for each subkey, we may need to combine these lists in some way in order to determine what are the most likely values for the full cryptographic key. One important motivation for this is that secure devices, such as the microcontrollers used in EMV cards, need to obtain a Common Criteria [Com] or EMVCo [EMV] certification at some assurance level (e.g. *EAL4+*). To provide such certification, evaluation laboratories may need to verify the security of devices against side-channel attacks also for the case of full-key recovery attacks, in particular where some subkeys may leak considerably different than others.

For the particular case of AES, we need to combine from 16 bytes (128-bit key) to 32 bytes (256-bit key). If the target device leaks enough information and sufficient measurements are done, then the attack may provide a probability close to one for the correct subkey value, while assigning a very small probability to the other candidate subkey values. In this case, the combination is trivial, as we only need to use the most likely value for each subkey. However, in practice, due to noise in the measurements and various security measures in secured devices, the correct value of each subkey may be ranked anywhere between the first and the last position. In this case, a trivial direct combination of all the lists of probabilities is not computationally feasible. Note that this problem arises in any scenario where we need to combine multiple lists of probabilities.

To deal with this combination problem in the context of side-channel attacks, two kinds of algorithms have emerged in recent years: key enumeration and rank estimation algorithms. Key enumeration algorithms [VGR+12, PRS+16] provide a method to output full keys in decreasing order of likelihood, such that we can minimize the number of keys we try until finding the correct one (which is typically verified by comparing the encryption of a known plaintext/ciphertext pair).

The other kind of algorithms, which are the main focus of this paper, are guesswork estimation algorithms. These algorithms provide an estimate of the level of security remaining after a side-channel attack targeting a full cryptographic key, i.e. the estimated number of keys we should try (guess) until finding the correct one if we were to apply a similar approach to key enumeration. The great advantage of such estimation algorithms is that we can estimate the guesswork even if this is very high (e.g. 2^{80} or larger), whereas enumerating such large number of keys is computationally infeasible.

Veyrat-Charvillon et al. [CNG+13] proposed the first practical guesswork estimation algorithm for 128-bit keys, known as rank estimation algorithm because it estimates the key rank [SMY09] (empirical guessing entropy). This algorithm could run in between 5 and 900 seconds. The main drawbacks of this algorithm are that the bounds of the rank estimation can be up to 20-30 bits apart from the real key rank and the required time to tighten the bounds increases exponentially. Soon afterwards, many other algorithms, including those analysed in this paper, provided more efficient and scalable methods to perform guesswork estimation [GCG+15, MPO+15, BJL+15, CP17, GV18, DW19, ZZD+20, DW21].

Among these, we shall focus on three remarkable methods: a) the rank estimation from FSE 2015 [GCG+15] – one of the first efficient implementations for rank estimation, among the best performant methods to this day[2]; b) the scalable bounds on Massey's guessing entropy from CHES 2017 [CP17] – the most scalable solution to date; c) the rank estimation method from CHES 2020 [ZZD+20] – introducing a new approach to rank estimation: based on estimating the distribution of scores.

We start below with a presentation of the two guesswork metrics (*Massey's* guessing entropy and the *empirical* guessing entropy), on which the three full-key evaluation methods mentioned above are based on.

3 Security metrics based on expected guesswork

The methods analysed in this paper estimate or bound two main security metrics: the key rank (or *empirical* guessing entropy), which we shall refer to as GE in this paper, and *Massey's* guessing entropy, which we shall refer to as GM. Unfortunately, through the past decade these metrics have been mixed in several forms, causing some confusion. Hence, in this section we aim to clarify each metric in the context of side-channel attacks. This will be useful and necessary to understand better the differences between the three methods analysed afterwards, in the reminder of the paper. We start with a brief history overview of these metrics and then provide more details on each of them.

The first description of what we term as *Massey's guessing entropy* was most probably in the paper by James Massey in 1994 [Mas94], where he defined a value G as "the number of guesses used in the guessing strategy that minimizes $E[G]$". A few years later, under his supervision, Christian Cachin published his PhD thesis on "Entropy measures and unconditional security in cryptography" [Cac97]. In his thesis, Cachin termed the measure $E[G]$ (rewritten as $E[G(X)]$) as the "guessing entropy of X", where X is the random variable to be guessed (e.g. the key of a cryptographic algorithm). Also in this thesis, Cachin presented the "conditional guessing entropy of X given Y", $E[G(X|Y)]$, for "the case of guessing X with knowledge of a correlated random variable Y". This conditional guessing entropy was then used in the context of side-channel attacks probably for the first time by Köpf and Basin [KB07]. Shortly afterwards, Standaert et al. [SMY09] defined an empirical version of the guessing entropy that they termed as "guessing entropy". This definition and measure was then commonly used within the side-channel research community to evaluate the success of side-channel attacks. However, this also introduced some confusion for several reasons: (a) the measure is actually a conditional guessing entropy, as it depends on the side-channel observations; (b) they used the same "guessing entropy" term for a measure that is computed differently than the previously introduced conditional guessing entropy, defined by Cachin [Cac97] and used in the side-channel context by Köpf and

[2]The recent paper of David and Wool [DW21] also states that the FSE 2015 rank estimation algorithm is the best to date, their advantage over FSE 2015 being that they can prove a bound on the error for their method, while this is not certain for FSE 2015, particularly when dealing with strange distributions. However, for our datasets the FSE 2015 bounds are very tight, hence we can resort to use this method, which is very practical and easy to implement.

Basin [KB07]; (c) this measure has been called either "guessing entropy" or "rank" or "key rank" across different publications. As a result, it is not entirely clear what metric should be used: if we should prefer one metric over the other or in which context we should use each of them. Throughout this paper, we aim to bring some light into this issue, given the importance of these metrics for the scalable methods evaluated here.

3.1 Massey's guessing entropy (GM)

As detailed earlier, James L. Massey proposed in 1994 a metric used to capture the "number of guesses used in the guessing strategy that minimizes $E[G]$" [Mas94] (until we find the desired value, e.g. some cryptographic key). Cachin rewrote this value as $E[G(X)]$ and termed it as the "guessing entropy of X" [Cac97], specifying the random variable X to be guessed. This is computed as:

$$E[G(X)] = \sum_{i=1}^{N} i \cdot p_i, \tag{1}$$

where $p_1 \geq p_2 \geq \ldots \geq p_N$ are the probabilities for different values of the random variable X, in descending order, according to the guessing strategy.

Cachin then defined also $G(X|Y)$, for the case of guessing X with knowledge of a correlated random variable Y, as the "guessing function for X given Y when $G(X|y)$ is a guessing function for the probability distribution $P_{X|Y=y}$". In this case, we can compute the average number of guesses needed to determine X, with knowledge of a correlated random variable Y, obtaining the *conditional guessing entropy of X given Y* as:

$$E[G(X|Y)] = \sum_{y\in\mathcal{Y}} P_Y(y) E[G(X|y)]. \tag{2}$$

Köpf and Basin used this conditional guessing entropy in the context of side-channel attacks (where the correlated random variable Y is the side-channel leakage), stating that $E[G(X|Y)]$ is "a lower bound on the expected number of off-line guesses that an attacker must perform for key recovery after having carried out a side-channel attack" [KB07].

For our side-channel evaluation context, we are interested in the guessing entropy (or conditional guessing entropy) of a secret key K given the leakage X, i.e. $E[G(K|X)]$. For this, we can first compute $E[G(K|X = \mathbf{X})]$ as:

$$E[G(K|X = \mathbf{X})] = \sum_{i=1}^{|\mathcal{S}|} i \cdot P(k_i|X = \mathbf{X}), \tag{3}$$

where $|\mathcal{S}|$ represents the number of possible values of K and $P(k_1|X = \mathbf{X}) \geq P(k_2|X = \mathbf{X}) \geq \ldots \geq P(k_{|\mathcal{S}|}|X = \mathbf{X})$ represent the conditional probabilities obtained after a side-channel attack with traces $\mathbf{X}$ (see acquisition details in Section 5), sorted in descending order (according to the guessing strategy). Then, we can compute $E[G(K|X)]$ as:

$$E[G(K|X)] = \sum_{\mathbf{X}\in X} P(X = \mathbf{X}) E[G(K|X = \mathbf{X})]. \tag{4}$$

Since it is often not possible to iterate over all the possibilities of the leakage space, we often approximate the above expectation using enough experiments (N), obtaining the conditional guessing entropy as follows:

$$E[G(K|X)] \approx \frac{1}{N} \sum_{q=1}^{N} E[G(K|X = \mathbf{X}_q)]. \tag{5}$$

We shall refer to this value as *Massey's guessing entropy* (or GM for short) in the reminder of the paper. Hence, we have:

$$\mathrm{GM} = \frac{1}{N}\sum_{q=1}^{N} E[G(K|X = \mathbf{X}_q)]. \tag{6}$$

3.2 Empirical guessing entropy (GE)

From a more empirical perspective, Standaert et al. presented in 2009 another guessing entropy, based on the position of the correct key in the sorted vector of conditional probabilities [SMY09]. Let $P(k_1|X = \mathbf{X}) \geq P(k_2|X = \mathbf{X}) \geq \ldots \geq P(k_{|\mathcal{S}|}|X = \mathbf{X})$ be again the conditional probabilities obtained after a side-channel attack with traces $\mathbf{X}$, sorted in descending order (according to the guessing strategy), for each of the $|\mathcal{S}|$ possible values of the key. Then, we can define this guessing entropy for a single experiment (typically known as *rank*) as:

$$\mathrm{rank}(K|X = \mathbf{X}) = i, \quad P(k_1|\mathbf{X}) \geq \ldots \geq P(k_i|\mathbf{X}) = P(k\star|\mathbf{X}) \geq \ldots \geq P(k_{|\mathcal{S}|}|\mathbf{X}), \tag{7}$$

where $k\star$ represents the correct key and $\mathbf{X}$ the side-channel traces. That is, $\mathrm{rank}(K|X = \mathbf{X})$ provides the *actual* position of the correct key $k\star$ in the list of candidate values, sorted accordingly to the guessing strategy (typically after a side-channel attack, in our context).

Similarly to the case of *Massey's guessing entropy* GM (see above), we are also interested here in obtaining an expected or average value for this rank. We can compute this as:

$$\mathrm{rank}(K|X) = \sum_{\mathbf{X}\in X} P(X = \mathbf{X})\mathrm{rank}(K|X = \mathbf{X}) \approx \frac{1}{N}\sum_{q=1}^{N}\mathrm{rank}(K|X = \mathbf{X}_q), \tag{8}$$

where N is again the number of experiments used for a sufficient approximation of the rank. We shall refer to this last value as the *empirical guessing entropy* (or GE for short). Hence, we have:

$$\mathrm{GE} = \frac{1}{N}\sum_{q=1}^{N}\mathrm{rank}(K|X = \mathbf{X}_q). \tag{9}$$

We can also compute the expected value of the rank, $\mathrm{rank}(K|X)$, based on the probability that the correct key $k\star$ is ranked at a given position ($\mathrm{pos}_{k\star}$) in the descendingly sorted list of possible key values:

$$\mathrm{rank}(K|X) = \sum_{i=1}^{|\mathcal{S}|} i \cdot P[\mathrm{pos}_{k\star} = i|X], \tag{10}$$

although the probabilities $P[\mathrm{pos}_{k\star} = i|X]$ are difficult to use in scalable security estimation methods. However, there is another form of the above equation, easier to estimate and which is used by one of the methods analysed in this paper (see Section 4.3), based on the probabilities that other keys (k_i) are more (or less) likely than the correct one ($k\star$):

$$\mathrm{rank}(K|X) = 1 + \sum_{\{i|k_i\neq k\star\}} \mathbf{P}[\mathrm{pos}_{k_i} < \mathrm{pos}_{k\star}|X] = 1 + \sum_{\{i|k_i\neq k\star\}} \{1 - \mathbf{P}[\mathrm{pos}_{k\star} < \mathrm{pos}_{k_i}|X]\}. \tag{11}$$

This last equation can be more helpful than Equation 10, because the summation here can be estimated using the normal cumulative density function, as shown in Section 4.3.

4 Efficient full-key security evaluation methods

In this section we present the three main guesswork estimators analysed in this paper: the rank estimation from FSE 2015 [GCG+15], the scalable bounds from CHES 2017 [CP17] and the recent rank estimation method from CHES 2020 [ZZD+20]. We may rewrite slightly the original notations in order to harmonize the presentation in this paper.

4.1 FSE 2015 estimator for GE

The rank estimation method of Glowacz et al. [GCG+15], to which we refer as the FSE 2015 estimator in this paper, is one of the fastest such algorithms and scales well for keys up to 128 bytes. To use this algorithm, we must first compute the logarithms of the conditional probabilities, $\log P(k_i^j | X = \mathbf{X})$, for all the n_{s} chunks ($1 \leq i \leq |\mathcal{S}|$, $1 \leq j \leq n_{\mathrm{s}}$) of a full cryptographic key (e.g. $n_{\mathrm{s}} = 16$ byte chunks for a 16-byte AES key). Then, we can compute the histograms of the log-probabilities for each key chunk:

$$H_j = \mathrm{histogram}(\log P(k_i^j | X = \mathbf{X})), \tag{12}$$

and finally compute a large histogram for the entire key, by combining the individual histograms through convolution:

$$H_{\mathrm{full}} = H_1 * H_2 * \ldots * H_{n_{\mathrm{s}}}. \tag{13}$$

This is typically computed by first convolving the first two histograms, then the result with the third histogram and so on.

Having computed the convolution of all the histograms and with knowledge of the correct key (and hence of the bins containing the correct key chunk in each histogram), we can estimate the full key rank by adding the values of the bins in the full histogram, starting with the bin that should contain the correct full-key until the last one:

$$\mathrm{rank}_{\mathrm{FSE}}(K^{\mathrm{full}} | X = \mathbf{X}) = \sum_{t = \mathrm{bin}(\log P(k\star^{\mathrm{full}} | X = \mathbf{X}))}^{\#\mathrm{bins\ in\ } H_{\mathrm{full}}} H_{\mathrm{full}}(t). \tag{14}$$

This basically estimates the number of keys with a log-probability larger than or equal to the one of the correct key, which essentially estimates the rank of the correct key.

Similar to the empirical guessing entropy (see previous section), we can approximate the expectation of this rank of the full key using N experiments, obtaining:

$$\mathrm{GE}_{\mathrm{FSE}}^{\mathrm{full}} = \frac{1}{N} \sum_{q=1}^{N} \mathrm{rank}_{\mathrm{FSE}}(K^{\mathrm{full}} | X = \mathbf{X}_q). \tag{15}$$

There have been some speed improvements on this method, such as the work of Grosso from CARDIS 2018 [GV18], which also compares the execution time of the FSE 2015 estimator and the CHES 2017 bounds [GV18, Fig. 6, p.14]. However, as mentioned by the same author, the performance improvements come at the cost of tightness. Similarly, the recently published method of David and Wool [DW21] can improve slightly the performance at the cost of memory consumption, but is also more complicated to implement. Hence, for generality and simplicity, we shall use the FSE 2015 method in the analysis of this work, given that it remains one of the representative algorithms for estimating the empirical guessing entropy GE when targeting full cryptographic keys.

4.2 CHES 2017 bounds for GM

At CHES 2017, Choudary and Popescu [CP17] showed efficient and scalable bounds for Massey's guessing entropy as follows:

$$\frac{1}{1+\ln|\mathcal{S}|}\left[\sum_{i=1}^{|\mathcal{S}|} p_i^{1/2}\right]^2 \leq E[G(K|X=\mathbf{X})] \leq \frac{1}{2}\left[\sum_{i=1}^{|\mathcal{S}|} p_i^{1/2}\right]^2 + \frac{1}{2}, \tag{16}$$

where $p_i = P(k_i|X=\mathbf{X})$ are again the conditional probabilities obtained after a side-channel attack, but not necessarily sorted. This allows a faster evaluation, as we do not need to sort the probabilities, as it is necessary e.g. for computing Massey's guessing entropy or the empirical guessing entropy (see Section 3). By averaging the terms over many experiments, we obtain the bounds for the approximation of the expected value of Massey's guessing entropy, which was termed GM:

$$\frac{1}{N}\sum_{q=1}^{N}\frac{1}{1+\ln|\mathcal{S}|}\left[\sum_{i=1}^{|\mathcal{S}|} p_{i,q}^{1/2}\right]^2 \leq \mathrm{GM} \leq \frac{1}{N}\sum_{q=1}^{N}\left[\frac{1}{2}\left[\sum_{i=1}^{|\mathcal{S}|} p_{i,q}^{1/2}\right]^2 + \frac{1}{2}\right], \tag{17}$$

where $p_{i,q} = P(k_i|X=\mathbf{X}_q)$ is the conditional probability for key value k_i in experiment q.

In the same CHES 2017 paper, the authors also showed that these bounds can be used for full-key security evaluations [CP17, Theorem 1], providing the following bounds:

$$\frac{1}{1+\ln|\mathcal{S}|^{n_s}}\prod_{j=1}^{n_s}\left[\sum_{i=1}^{|\mathcal{S}|}\sqrt{p_i^j}\right]^2 \leq E[G(K^{\mathrm{full}}|X=\mathbf{X})] \leq \frac{1}{2}\prod_{j=1}^{n_s}\left[\sum_{i=1}^{|\mathcal{S}|}\sqrt{p_i^j}\right]^2 + \frac{1}{2}, \tag{18}$$

where $p_i^j = P(k_i^j|X=\mathbf{X})$ are the conditional probabilities for the j-th key chunk values k_i^j $(1 \leq j \leq n_s)$. We may again average each term over N experiments, to obtain bounds for Massey's guessing entropy GM on the full key ($\mathrm{GM}^{\mathrm{full}}$):

$$\frac{1}{N}\sum_{q=1}^{N}\frac{1}{1+\ln|\mathcal{S}|^{n_s}}\prod_{j=1}^{n_s}\left[\sum_{i=1}^{|\mathcal{S}|}\sqrt{p_{i,q}^j}\right]^2 \leq \mathrm{GM}^{\mathrm{full}} \leq \frac{1}{N}\sum_{q=1}^{N}\left[\frac{1}{2}\prod_{j=1}^{n_s}\left[\sum_{i=1}^{|\mathcal{S}|}\sqrt{p_{i,q}^j}\right]^2 + \frac{1}{2}\right], \tag{19}$$

where $p_{i,q}^j = P(k_i^j|X=\mathbf{X}_q)$ are the conditional probabilities for the j-th key chunk values k_i^j in experiment q.

This is the fastest and most scalable method for full-key evaluation known to date. We shall refer to these lower and upper bounds on GM as $\mathrm{LB}_{\mathrm{GM}}$ and $\mathrm{UB}_{\mathrm{GM}}$, respectively, regardless of whether we use them for single byte or full-key evaluations.

4.3 CHES 2020 estimator for GE

At CHES 2020, Zhang et al. [ZZD+20] proposed a new method, termed 'GEEA' (guessing entropy estimation algorithm) to estimate the empirical guessing entropy (GE). Their method relies on the observations made by Rivain et al. [Riv08, LPR+14], that the success rate can be computed from the multivariate Gaussian distribution of the ranking score vectors. Zhang et al. have used this distribution to produce the GEEA estimator, which is expected to approximate better the value rank$(K|X)$ than the average estimator GE (see Section 3.2 for details).

Given a list of scores $\mathbf{s} = \{s_1, s_2, \ldots, s_{|\mathcal{S}|}\}$, obtained after a side-channel attack (e.g. the probabilities $P(k_i|X=\mathbf{X})$ or their logarithm $\log P(k_i|X=\mathbf{X})$), we can compute a *comparison vector* $\mathbf{\Delta}(K|X = \mathbf{X})$ having $(|\mathcal{S}| - 1)$ elements:

$$\Delta_i(K|X = \mathbf{X}) = s_i - s_{k\star}, \tag{20}$$

where the values Δ_i are computed for all the scores values s_i, except for $s_{k\star}$, the score for the correct key. Then, Rivain et al. [Riv08, LPR+14] have shown that if the scores from multiple experiments can be combined through addition[3], then this comparison vector follows a multivariate distribution $\mathcal{N}(\mu_{\mathbf{\Delta}}, \frac{1}{n_\mathrm{a}}\mathbf{\Sigma}_{\mathbf{\Delta}})$, where $\mu_{\mathbf{\Delta}}$ and $\mathbf{\Sigma}_{\mathbf{\Delta}}$ are the mean vector and respectively covariance matrix of the comparison vector, and n_a represents the data available to an attacker, for which we wish to estimate the comparison vector. Note that we need to obtain the side-channel traces over a sufficient number N of experiments to estimate the mean ($\mu_{\mathbf{\Delta}}$) and covariance ($\mathbf{\Sigma}_{\mathbf{\Delta}}$) of the comparison vector. Then, we use these multivariate parameters to estimate the comparison vector for a given value n_a, with the goal of estimating the guessing entropy obtained by an adversary that has access to n_a attack traces.

From a particular observation of the comparison vector $\mathbf{\Delta}(K|X = \mathbf{X})$, we can compute the rank as:

$$\mathrm{rank}(K|X = \mathbf{X}) = N_{\mathrm{pos}}(\mathbf{\Delta}(K|X = \mathbf{X})) + 1, \tag{21}$$

where $N_{\mathrm{pos}}(\mathbf{\Delta}(K|X = \mathbf{X}))$ represents the number of positive components of $\mathbf{\Delta}(K|X = \mathbf{X})$, i.e. the number of components for which $s_i > s_{k\star}$. Thus, we can obtain the expected value of the rank as:

$$\mathrm{rank}(K|X) = 1 + \sum_{\{i|s_i \neq s_{k\star}\}} \mathbf{P}[s_i > s_{k\star}] = 1 + \sum_{\{i|s_i \neq s_{k\star}\}} \{1 - \mathbf{P}[s_{k\star} > s_i]\}. \tag{22}$$

Based on the above results, Zhang et al. provide the following formula for their GEEA estimator of the rank:

$$\mathrm{GEEA}_{n_\mathrm{a}} = 1 + \sum_{i=1}^{|\mathcal{S}|-1} \Phi\left(\frac{\sqrt{n_\mathrm{a}}\Delta_i(K|X)}{\sqrt{\mathbf{S}_{ii}(K|X)}}\right), \tag{23}$$

where

$$\begin{aligned} \Delta_i(K|X) &= \frac{1}{N}\sum_{q=1}^{N} \Delta_i(K|X = \mathbf{X}_q), \\ \mathbf{S}_{ii}(K|X) &= \frac{1}{N}\sum_{q=1}^{N} [\Delta_i(K|X = \mathbf{X}_q) - \Delta_i(K|X)]^2, \end{aligned} \tag{24}$$

are the estimated mean and variance of the component $\Delta_i(K|X = \mathbf{X})$ of $\mathbf{\Delta}(K|X = \mathbf{X})$, $\Phi(\cdot)$ represents the normal cumulative density function and n_a is the assumed number of attack traces available to an attacker for the estimation of the empirical guessing entropy using GEEA.

The authors of GEEA also provide an extension of their estimator for full-key evaluation. Given the set of comparison vectors $\{\mathbf{\Delta}^1, \mathbf{\Delta}^2, \ldots, \mathbf{\Delta}^{n_\mathrm{s}}\}$ for n_s target bytes (e.g. sub-key values), with their respective estimated mean and variance components $\Delta_i^j(K|X)$ and ${\mathbf{S}_{ii}}^j(K|X)$ for each $\mathbf{\Delta}^j$, Zhang et al. [ZZD+20] sum these values, assuming that the full key comparison score is the sum of byte comparison scores[4], to obtain the mean and variance

[3]They term a side-channel attack with this property an “additive distinguisher” and show that popular attack methods have this property, either directly (e.g. using the correlation coefficient) or indirectly (e.g. using a score derived from the logarithm of the multivariate Gaussian distribution in the case of Template Attacks).

[4]An assumption that may need more analysis in the context of side-channel attacks.

of the comparison score vector for each possible full-key candidate $k^f = (k^{i_1}|k^{i_2}|\dots|k^{i_{n_s}})$:

$$\Delta_{k^f}(K|X) = \sum_{j=1}^{n_\mathrm{s}} \Delta^j_{i_j}(K|X), \ \mathbf{S}_{k^f}(K|X) = \sum_{j=1}^{n_\mathrm{s}} \mathbf{S}_{i_j i_j}{}^j, \tag{25}$$

with the observation that each of the values $\Delta^j_{i_j}(K|X)$ and $\mathbf{S}_{i_j i_j}{}^j$ are particular to each subkey byte j ($1 \le j \le n_\mathrm{s}$). With these values, Zhang et al. compute the estimation of the full key rank from a (random) set[5] of M full-key values $S^f = \{k^f_1, k^f_2, \dots, k^f_M\}$ as:

$$\mathrm{GEEA}^{\mathrm{full}}_{n_\mathrm{a}} = 1 + \frac{n_\mathrm{s} * |\mathcal{S}| - 1}{M} \sum_{k^f_j \in S^f} \Phi\left(\frac{\sqrt{n_\mathrm{a}}\Delta_{k^f_j}(K|X)}{\sqrt{\mathbf{S}_{k^f_j}(K|X)}}\right). \tag{26}$$

We make again a note here that, while for the FSE 2015 and CHES 2017 methods the approximation of the expected value of the rank ($\mathrm{GE}^{\mathrm{full}}_{\mathrm{FSE}}$) or Massey's conditional guessing entropy ($\mathrm{GM}^{\mathrm{full}}$) were obtained through averaging over N individual experiments (see Equations 15 and 19), for GEEA the approximation of the expected value of the rank is obtained directly from the estimated mean and covariance parameters (see Equations 23 and 26), but in turn these parameters are also obtained by averaging over N individual experiments (see Equation 24). Hence, in all cases we need to iterate over N experiments to estimate acceptably the desired metrics.

5 Evaluation context

To analyse and compare the methods discussed in this paper, we have used the following indicators, that we think allow a fair and useful comparison of the methods from different perspectives:

- *Precision:* through this indicator, we want to check how well and how fast a given metric approximates the expected value that is trying to estimate. This will be typically done by measuring the standard deviation of the given method and by observing this deviation over a different number of experiments.
- *Resource complexity:* with this indicator we shall compare the time and memory complexity of the methods, either from the estimated theoretical bounds or from practical results.
- *Scalability:* this indicator is particularly useful for the full-key evaluations. Here, we aim to measure the possibility of each method to cope with evaluations on large cryptographic keys. In particular, we shall compare the rate of increasing time/memory complexity as a function of key length.
- *Relevance for side-channel evaluations:* this is a (possibly subjective) indicator, that observes the usefulness of a method for side-channel evaluation. This will be done through an iteration of possible scenarios where a method seems of interest for such security evaluations.

Furthermore, in order to provide a more comprehensive analysis of the evaluation methods, we used three distinct datasets: one from MATLAB simulated data (simulated dataset), one from the hardware AES co-processor of an AVR XMEGA device (XMEGA dataset) and one from a 32-bit ARM device (SoC dataset). We provide some more details below.

[5]Such random set can be obtained by selecting random values for each sub-key value and then concatenating these values in order to obtain each full-key candidate.

5.1 Simulated dataset

For this dataset, we simply implemented AES and added uniform noise to the output of the sub-bytes operation and then applied Template Attacks [CK13] to obtain lists of probabilities for each key byte of the AES key (16 key bytes in total).

The data contains unidimensional leakage samples $\mathbf{x}_i$ produced as the hamming weight of the AES S-box output value mixed with Gaussian noise, i.e.

$$\mathbf{x}_i = \text{HW}(\text{S-box}(k \oplus p_i)) + r_i, \quad 1 \leq i \leq \#\text{traces in dataset} \tag{27}$$

where p_i is the plaintext byte corresponding to this trace, and r_i represents the Gaussian noise (variance 10). We shall refer to this as the simulated dataset.

5.2 Xmega dataset

This dataset consists of $2^{20} \approx 1M$ power-supply traces of the AES engine inside an AVR XMEGA microcontroller, obtained while the cryptographic engine was encrypting different uniformly distributed plaintexts. The traces correspond to the S-box lookup from the first round key. Each trace contains $m = 5000$ oscilloscope samples recorded at 500MS/s, using a Tektronix TDS7054 oscilloscope, configured at 250 MHz bandwidth in HIRES mode with Fastframe and 10mV/div vertical resolution, using DC coupling. The XMEGA microcontroller was powered at 3.3 V from batteries and was run by a 2MHz sinewave clock. We shall refer to this as the XMEGA dataset.

5.3 SoC dataset

The third dataset consists of 100000 power traces acquired for a bitsliced variant of the AES-128 algorithm. The 32-bit unprotected implementation covered in [BJG+15] was used. The acquisition campaign was conducted entirely on a ChipWhisperer-Lite [Lit22], using the integrated STM32F303 32-bit ARM target architecture and the attached capture instrument. Each trace consists of 5000 samples recorded at $\approx$ 30MS/s (for a sampling clock of 29.48 MHz), covering also the processing of the first S-box operation. We shall refer to this as the SoC dataset.

5.4 Template attacks

To use our datasets with the methods evaluated in this paper, we need to obtain lists of probabilities for the possible values of the 16 subkeys used with our AES implementations. For this, we use Template Attacks (TA) [CRR02, CK13] on each subkey during the S-box lookup of the first AES round, thus obtaining the desired lists of probabilities. The Template Attacks work in two steps: a profiling and an attack step. In the profiling step, we first compute a set of profiling parameters that typically estimate a multivariate distribution for each possible candidate value (e.g. key byte). Then, during the attack step we compute the likelihood of the attack traces given the template parameters, hence obtaining (typically via Bayes) probabilities for each possible candidate value.

For each dataset we have between a few hundred thousands and one million traces. We split the data randomly into profiling and attack sets and we do this for many experiments (typically we create over 100 such sets). The traces in each pair of sets (profiling and attack) are randomised prior to the separation into profiling and attack set, so that we can remove unwanted effects such as temperature influencing consecutive traces. Typically we use many more traces for profiling, so the profiling parameters are well estimated (e.g. for the XMEGA dataset we used around 200 traces per byte value in each set). For the attack set we typically select between a few hundred and a few thousand traces per set.

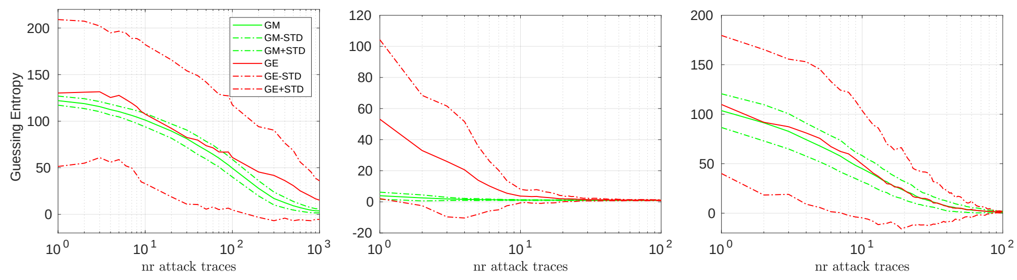

Figure 1: GM and GE with one standard deviation above and below across our datasets. Left: XMEGA; middle: SoC; right: simulated. Mean values and standard deviation obtained from 100 independent experiments.

In the case of the simulated dataset, the templates were obtained simply by computing the mean and covariance parameters for the simulated samples (one leakage sample per trace) and then using these template parameters on the set of attack traces. For the XMEGA dataset, we first applied a sample selection method equivalent to the SNR and SOST methods [CK18] in order to compress the traces down to only very few samples per trace and then applied the Template Attack on these compressed traces. Finally, for the SoC dataset, we combined stochastic models with Principal Component Analysis [COK14] in order to improve the profiling step and to reduce the size of the traces.

After executing a side-channel attack using a vector $\mathbf{X}$ of leakage traces (e.g. the *real* or *simulated* traces in our case), we obtain a vector of scores or probabilities $\mathrm{d}(k|\mathbf{X}) \in \mathbb{R}^{|\mathcal{S}|}$ for each possible key byte value $k \in \{1, \dots, |\mathcal{S}|\}$, where $|\mathcal{S}|$ is the number of possible values (typically $|\mathcal{S}| = 256$ for one AES subkey byte). In the case of Template Attacks, we obtain probabilities and we shall often write $P(k|\mathbf{X}) = \mathrm{d}(k|\mathbf{X})$.[6]

After obtaining the probabilities $P(k|\mathbf{X})$ for each subkey byte k, we can compute the security metrics and rank estimation methods presented in the paper.

6 Using GE or GM for security evaluations

Since the FSE 2015 and CHES 2020 methods approximate the expected value of the rank (i.e. they compute a variant of GE) and the CHES 2017 bounds are obtained for Massey's conditional guessing entropy (GM), we start by comparing these two variants of the guessing entropy. Then, in the next section we analyse the three methods used for full-key evaluation.

6.1 Precision of GE and GM

It was recently mentioned that the GM might have bad precision [AMP+19], requiring to average the results of many experiments to get good estimation results. Similarly, Zhang et al. [ZZD+20, Section 1,p.27] stated that the GM calculation suffers "the same practical problem of needing to average over many data sets" as the GE.

To verify the extent of this situation, we have used our datasets and computed both the GE and the GM metrics, with the formulas from Section 3, using $q = 100$ experiments and a varying amount of attack traces n_a ($1 \leq n_\mathrm{a} \leq 100$). Using this data we have also computed the standard deviation of each metric, hence comparing their accuracies. The

[6]Unprofiled side-channel attacks such as CPA often return a score vector, e.g. based on the correlation coefficient $\rho_k \in [-1, 1]$ for each possible candidate value k, which might not work very well with rank estimation methods. However, even in the unprofiled setting is possible to use other methods, such as linear regression on the fly [COP+16] to obtain pseudo-probabilities that work well with rank estimation algorithms.

Table 1: 5th, 50th and 95th percentile of standard deviation for GE and GM methods.

Method	XMEGA	SoC	simulated
GE	21.92, 65.8, 76.48	0.28, 0.47, 18.95	0.91, 10.79, 67.05
GM	2.26, 6.6, 9.79	0.02, 0.08, 0.5	0.81, 3.52, 16.02

results are shown in Figure 1. We also show in Table 1 the 5th, 50th (median) and 95th percentile of the standard deviation (across the different values of n_a attack traces) for each method in each experiment. From these results we observe that GE has indeed a large standard deviation, generally one order of magnitude larger than the standard deviation of the GM across our experiments. Therefore, methods based on GM may allow security evaluations even when having access to few attack traces.

The large differences between GE and GM can be explained by the fact that the calculation of the GE relies on the exact (actual) position of the correct key, which may fluctuate greatly, while the calculation of GM only depends on the relative magnitude of probabilities, regardless of the position of the correct key in the sorted vector of probabilities.

6.2 Resource complexity of GE and GM

As can be seen from their definitions (see Equations 6 and 9), both GE and GM require first sorting of the probabilities, for computation of the rank (Equation 7) and conditional guessing entropy (Equation 3), respectively. This has a computational complexity $O(|\mathcal{S}| \log |\mathcal{S}|)$, where $|\mathcal{S}|$ is again the number of possible values of the target candidate $k\star$ (e.g. a key byte or the entire key for full-key evaluations). If we consider also the number of experiments N required for a good estimation, then the complexity is $O(N|\mathcal{S}| \log |\mathcal{S}|)$, although in general the parameter N can be kept relatively small even when dealing with large key sizes, hence the important factor here is $|\mathcal{S}|$.

In any case, it is clear that both metrics have the same computational requirements and that it becomes impractical to use them for full cryptographic keys (e.g. 128-bit keys), due to the impossibility of performing the sorting in acceptable time. Hence the requirement for the full-key evaluation methods evaluated in next section.

In terms of memory, both methods need to store all the probabilities $P(k_i|X = \mathbf{X})$ in order to perform the sorting, hence they require memory linear with $|\mathcal{S}|$. This can again become impractical for large keys.

6.3 Relevance of GE and GM for side-channel security evaluations

After analysing the definitions, accuracies and performance of GM and GE, we need to evaluate the usefulness of each measure for security evaluation purposes. For this task, we shall resort to their original apparition and scope, as presented in Section 3.

As seen, GM approximates the conditional guessing entropy of the key K given the leakage X. Based on its original definition by Massey [Mas94] and then its further development by Cachin [Cac97], GM estimates the average number of guesses needed to determine the correct value of the key variable K when given the side-channel leakage X. This measure was then applied precisely in the side-channel context by Köpf and Basin [KB07], who gave the important statement that this conditional guessing entropy is *a lower bound on the expected number of off-line guesses that an attacker must perform for key recovery after having carried out a side-channel attack.* Hence, we can expect that the *actual* number of off-line guesses that an attacker must perform for key recovery after a side-channel attack is higher than the GM.

On the other hand, GE was introduced in the side-channel community by Standaert et al. [SMY09] precisely as an empirical measure, that would provide the exact number of guesses that an attacker would need to determine the correct value of the key variable K when given the leakage X.

Given this situation, we may expect that in general GM may provide a lower value than GE. Hence, an important question is which of them should be used and in which scenario, given that the GE has been used predominantly so far in the side-channel community, while GM has been reintroduced only recently, since the development of efficient bounds for full-key evaluations.

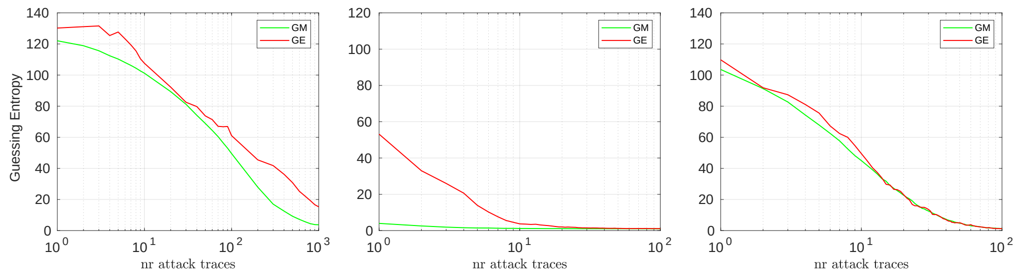

Figure 2: Results of GE and GM targeting one key byte across all our datasets: XMEGA (left), SoC (middle) and simulated (right).

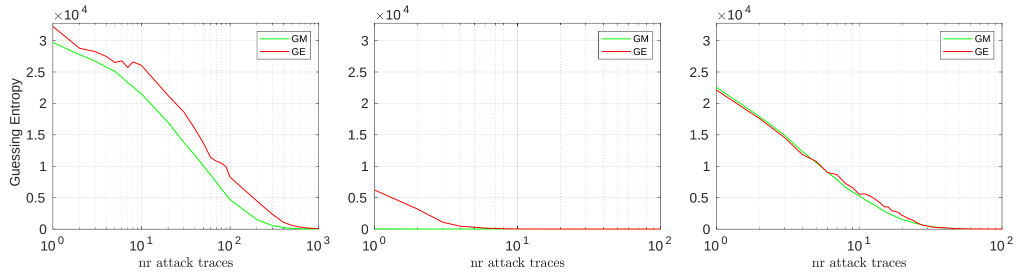

Figure 3: Results of GE and GM targeting two key bytes across all our datasets: XMEGA (left), SoC (middle) and simulated (right).

To give an answer to the question of which of them should be used and in which scenarios, we start by evaluating both metrics within our datasets, when targeting one and two bytes (for full-key evaluations see next section). Our results are shown in Figures 2 and 3 for evaluations on one and two bytes, respectively.

As we can see, indeed the GM is consistently below GE, as expected from the discussion above. Furthermore, Zhang et al. [ZZD+20, see Appendix] also showed that in general GM will be lower than GE and that values of GM above GE are generally not observable in numerical studies. Given these results and observations, we can state the following conclusions when targeting a small key chunk (one or two bytes):

- GM will likely provide a lower bound on the actual guessing entropy, i.e. an attacker will generally require more effort than shown by GM.
- GE provides a good approximation of the expected effort needed by a real attacker, at the cost of some more experiments to obtain a smooth approximation.

Therefore, we may select to use one or the other depending on our evaluation requirements. If we must provide an estimate of the difficulty in attacking a device that is as close as possible to the attacker's perspective, then we should use GE. However, if all we need is to check whether the security of a device is above a certain threshold, then GM may also be of use. This will become much more relevant when we focus on the full-key scenario, in the next section.

Besides the above conclusions regarding the usefulness of each metric individually, we can observe another very important aspect, based on the results shown above: that the difference between GE and GM is related to the quality of the leakage model, as also noticed by Zhang et al. [ZZD+20, Appendix] in their analysis. Examining the figures above for both, evaluations on a single byte as well as evaluations on two bytes, we can see that the two metrics almost overlap for the simulated dataset, they diverge slightly (GM below GE) for the XMEGA dataset and they diverge greatly for the SoC dataset.

Such variations in the difference between GE and GM can be explained as follows. For the simulated dataset, we have used the exact leakage model that was used to generate the traces (since they are simulated from the leakage model) and in this case, as noted also by Zhang et al., the correct key is very well distinguished and hence the order of probabilities generally matches well with the order of the correct key, leading to GE being very close to GM. For the XMEGA dataset, we have a leakage model that is not entirely accurate, due to the noise of the various components influencing the leakage traces used to attack the hardware AES implementation in the XMEGA device. Finally, for the SoC dataset, we have a very weak leakage model due to targeting one or two bytes at a time in the bitsliced AES implementation on a 32-bit device. In this case, only 8 or 16 bits from the 32 processed by the device are actually relevant (e.g. the first bit of the first 8 key bytes), while the remaining bits produce substantial noise, hence leading to the weak leakage model and to the large difference between GE and GM.

Hence, rather than using only one metric or the other, these results show that it can be very useful to compute both and use their difference as a method to verify the quality of the leakage model used during security evaluations: if the two metrics are close to each other, then the model is good, otherwise the model may suffer from estimation or assumption errors [DFS+14].

7 Comparative analysis of full-key evaluation methods

In this section, we analyse the three full-key security evaluation methods presented earlier: the rank estimation algorithm from FSE 2015 [GCG+15], the scalable bounds for GM from CHES 2017 [CP17] and the newer rank estimation algorithm from CHES 2020 [ZZD+20], using the indicators and datasets presented earlier (see Section 5).

7.1 Precision of full-key estimators: the case for low-data complexity tools.

As Azouaoui et al. [AMP+19] have mentioned, it is very useful to have a tool that can quickly (e.g. using very few attack traces) determine whether an attack has some chance of revealing the key with practical computation time and hence whether we should perform key enumeration (which is not trivial for full cryptographic keys). In this context, evaluating the precision of each full-key evaluation method seems very relevant.

For this task, we have applied the three full-key evaluation methods on our datasets, comparing their standard deviation (similarly to what we did in Section 6.1). Our results are shown in Figure 4. The 5th, 50th and 95th percentile of the standard deviation across the n_{a} attack traces in bits (i.e. by taking the difference of the logarithms of the measures with and without a standard deviation) for each method and dataset is shown in Table 2.

For the GEEA method, the computation of the mean and standard deviation need some explanation, since the GEEA method already produces an estimation based on a

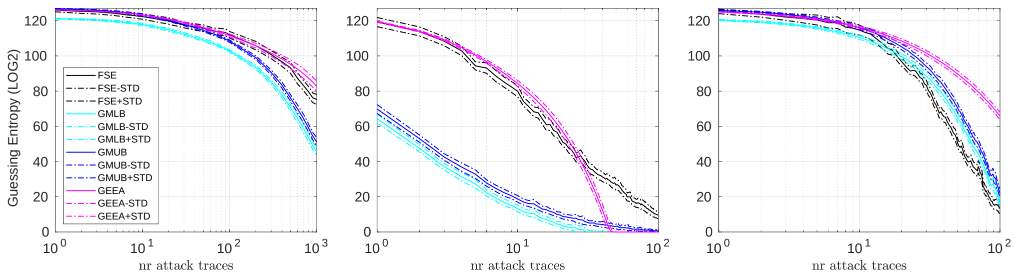

Figure 4: Precision results across the three full-key evaluations methods for all our datasets: XMEGA (left), SoC (middle) and simulated (right).

Table 2: 5th, 50th and 95th percentile of standard deviation (in bits) for the three full-key evaluations methods using an AES 128-bit key (16 key bytes).

Method	XMEGA	SoC	simulated
FSE 2015	1.12, 1.84, 3.14	2.11, 2.67, 3.20	2.11, 2.89, 3.41
CHES 2017 LB_{GM}	0.29, 0.74, 2.42	0.67, 1.36, 2.39	0.68, 2.22, 3.22
CHES 2017 UB_{GM}	0.29, 0.74, 2.42	0.5, 1.32, 2.34	0.68, 2.22, 3.22
CHES 2020	0.04, 0.56, 2.88	0, 0.81, 3.46	0.49, 1.77, 2.08

prior use of mean and standard deviation parameters. In order to provide a reasonable comparison between metrics, we have decided to allow the same amount of data to each metric. Hence, since the GEEA method estimates the guessing entropy of an attack with n_a traces based on the previously computed parameters, we have used a total of n_a traces per each computation of the GEEA method and then computed the mean and standard deviation of GEEA over N such computations. Furthermore, in order to provide a fair comparison also in terms of computing power, we have limited the number of random keys used by the full-key GEEA estimator (see Equation 26) to $M = 10^4$, which results in a similar computation time as for the computation of the mean and standard deviation values for the FSE 2015 and CHES 2017 methods [7].

These results show that both the CHES 2017 and CHES 2020 methods have good precision overall, while the FSE 2015 method has a somewhat larger standard deviation. However, we should also notice that unfortunately the CHES 2020 method provides results quite far from the FSE 2015 method, most likely due to the selected number of random keys used for its approximation, as detailed above.

7.2 Time/complexity analysis of full-key estimators

To continue our analysis, we show in Table 3 the time required to compute each method for 16-byte (128-bit), 128-byte (1024-bit) and 1024-byte (8192-bit) keys across all the datasets. The time includes the computation for all values of number of attack traces and all iterations. As mentioned before, for the CHES 2020 method it is difficult to set a clear point, as its computation time depends also on the number of random keys that we want to use for its approximation. The more we use, the better should be its estimation, but increasing it too much becomes impractical. For 16 bytes we have used $M = 10^4$ and for 128 bytes we used $M = 10^6$ values, resulting in the time shown. Nevertheless, the memory requirements remain the same, since we only need to keep the same list of score vectors, regardless of the number of random keys that we use.

These results confirm that the CHES 2017 method is faster than the others by one or

[7]Between 15-30 seconds on our machine: Intel(R) Core(TM) i5-4460 CPU @ 3.20GHz, 16GB RAM.

Table 3: Computation time (in seconds) for all the full-key evaluation methods on our XMEGA/SoC/simulated datasets with different number of key bytes.

Method	16 bytes	128 bytes	1024 bytes
FSE 2015	29/60/172	1027/5336/4689	N/A
CHES 2017	1/1/1	2/6/6	40
CHES 2020	17/18/26	432/415/473	N/A

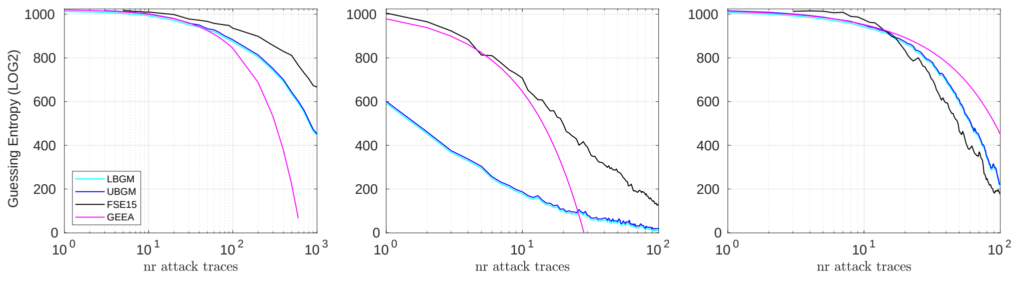

Figure 5: Results of all full-key evaluation methods for a 128-byte (1024-bit) key across all our datasets: XMEGA (left), SoC (middle) and simulated (right).

more orders of magnitude. Nevertheless, one may still prefer one of the other methods, if the requirements are to estimate precisely the empirical guessing entropy GE.

7.3 Scalability of full-key estimators

In this section we explore the scalability of each method by comparing them using 128 byte keys[8] across all our datasets. The results of these methods across our datasets are shown in Figure 5. We make here the following observations:

- All methods can be computed relatively well for 128 byte keys. However, when trying to compute the results for even larger keys, e.g. 1024 byte keys, we could not obtain this for the FSE 2015 and CHES 2020 methods, due to their computational and memory limitations. This was somewhat expected for the FSE 2015 method, but now it was also seen for the CHES 2020 method. Hence, for such larger keys, the CHES 2017 method might be the only viable solution.

- We see again the difference between the results of the FSE 2015 and CHES 2020 methods, confirming that they do not lead to same results when dealing with large keys and moderate amount of computation. From previous publications, such as our CHES 2017 paper [CP17, Figure 5], we observe that the FSE 2015 method follows closely the empirical guessing entropy GE. Hence, our results imply that the CHES 2020 method cannot reliably approximate the empirical guessing entropy, using moderate computation.

- These figures also show that the FSE 2015 and CHES 2017 are close for the simulated and XMEGA datasets, but differ substantially for the SoC dataset, i.e. where the leakage model is not accurate as described earlier. Hence, this confirms that we may combine the FSE 2015 and CHES 2017 methods to determine whether a leakage model is sound.

[8]For simplicity, we have just replicated the lists of probabilities from the 16 AES keys as many times as needed. The scalability conclusions would be the same with other lists.

Table 4: Properties of full-key evaluation methods.

Method	Precision	Speed	Scalability	Remarks
FSE 2015	Moderate	Slow	Up to 256-byte keys	Good estimate of GE.
CHES 2017	Good	Fast	1024-byte keys and beyond	Good estimate of GM. Generally a lower bound for GE. May be used together with FSE 2015 to assess the quality of a leakage model.
CHES 2020	Good	Slow	Up to 256-byte keys	Good precision. Can deviate from FSE 2015/GE. May be useful with a large amount of computation if we need high precision.

7.4 Relevance of full-key estimators for security evaluations

As seen by previous results, each of the full-key estimation methods explored in this paper has its advantages and limitations. We make a summary of our observations in Table 4.

8 Conclusion

In this paper we have explored the differences between two versions of the guessing entropy, as used for security evaluations of side-channel attacks: Massey's guessing entropy and the empirical guessing entropy. Our analysis has clarified previous confusion on the difference between these two measures when used for side-channel evaluations, allowing security evaluators to have a better understanding of each metric and its potential use.

Furthermore, we have analysed three representative full-key estimation methods for these security metrics, namely the methods presented at FSE 2015, CHES 2017 and CHES 2020. Our analysis presents a clear overview of the advantages and limitations of each method, which should be of great utility for any security evaluator.

In addition, we have discovered a new method for verifying the soundness of a leakage model used in a side-channel attack, by combining the empirical guessing entropy and Massey's guessing entropy. This may provide a useful tool for side-channel evaluations of both small and large keys.

Acknowledgement: We thank very much Professor Virgil Gligor for his profound and friendly dedication towards helping and improving this paper.

Funding: This research was supported by a grant from the Romanian Ministry of Education and Research, CNCS–UEFISCDI, project number PN-III-P1-1.1-TE-2019-2245.

References

[Mas94] J.L. Massey, *Guessing and Entropy*, IEEE ISIT, 1994, p. 204.

[Cac97] Christian Cachin, "Entropy measures and unconditional security in cryptography", PhD diss., ETH Zurich, 1997.

[KJJ99] Paul Kocher, Joshua Jaffe and Benjamin Jun, "Differential Power Analysis", CRYPTO 1999.

[CRR02] S. Chari, J. Rao, and P. Rohatgi, "Template Attacks", CHES 2002, Springer, 2003, LNCS 2523, pp 51–62.

[BEC+04] Brier, Eric, Christophe Clavier, and Francis Olivier. "Correlation power analysis with a leakage model.", Cryptographic Hardware and Embedded Systems-CHES 2004. Springer Berlin Heidelberg, 2004, pp. 16–29.

[KB07] B. Köpf and D. Basin, "An Information-theoretic Model for Adaptive Side-channel Attacks", in Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS), 2007, pp. 286–296.

[SMY09] F.-X. Standaert, T. G. Malkin, and M. Yung, "A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks", Eurocrypt 2009, LNCS 5479, pp 443–461.

[Riv08] Matthieu Rivain, "On the exact success rate of side channel analysis in the gaussian model", in International Workshop on Selected Areas in Cryptography, pp. 165-183. Springer, Berlin, Heidelberg, 2008.

[VGR+12] N. Veyrat-Charvillon, B. Gerard, M. Renauld and F.-X. Standaert, "An optimal Key Enumeration Algorithm and its Application to Side-Channel Attacks", SAC 2012.

[CNG+13] Veyrat-Charvillon, Nicolas, Benoît Gérard, and François-Xavier Standaert, "Security Evaluations beyond Computing Power", EUROCRYPT 2013, LNCS 7881, pp. 126–41.

[CK13] O. Choudary and M. G. Kuhn, "Efficient Template Attacks", CARDIS 2013, Berlin, 27–29 November 2013, LNCS 8419, pp. 253–270.

[COK14] Choudary, Marios O., and Markus G. Kuhn. "Efficient stochastic methods: profiled attacks beyond 8 bits". In International Conference on Smart Card Research and Advanced Applications, pp. 85-103. Springer, Cham, 2014.

[LPR+14] Victor Lomné, Emmanuel Prouff, Matthieu Rivain, Thomas Roche, and Adrian Thillard, "How to estimate the success rate of higher-order side-channel attacks", in International Workshop on Cryptographic Hardware and Embedded Systems (CHES) 2014, pp. 35–54.

[DFS+14] Durvaux, François, François-Xavier Standaert, and Nicolas Veyrat-Charvillon, "How to certify the leakage of a chip?", In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 459–476, 2014.

[GCG+15] Glowacz, Cezary, Vincent Grosso, Romain Poussier, Joachim Schüth, and François-Xavier Standaert, "Simpler and More Efficient Rank Estimation for Side-Channel Security Assessment". Fast Software Encryption 2015, LNCS 9054, pp. 117–29.

[BJL+15] Bernstein, Daniel J., Tanja Lange, and Christine van Vredendaal, "Tighter, Faster, Simpler Side-Channel Security Evaluations beyond Computing Power". `https://eprint.iacr.org/2015/221`.

[MPO+15] Martin, Daniel P., Jonathan F. O'Connell, Elisabeth Oswald, and Martijn Stam, "Counting Keys in Parallel After a Side Channel Attack". ASIACRYPT 2015, LNCS 9453, pp. 313–37.

[BJG+15] Balasch, Josep, Benedikt Gierlichs, Oscar Reparaz, and Ingrid Verbauwhede. "DPA, bitslicing and masking at 1 GHz". CHES 2015, pp. 599–619.

[PRS+16] Poussier, Romain, François-Xavier Standaert, and Vincent Grosso, "Simple Key Enumeration (and Rank Estimation) Using Histograms: An Integrated Approach", CHES 2016, to appear, `http://eprint.iacr.org/2016/571`.

[COP+16] Choudary, Marios O., Romain Poussier and François-Xavier Standaert, "Score-based vs. Probability-based Enumeration – a Cautionary Note –", Indocrypt 2016, to appear.

[CP17] Marios O. Choudary and P. G. Popescu, "Back to massey: impressively fast, scalable and tight security evaluation tools", CHES 2017, pp. 367–386.

[CK18] M. O. Choudary and M. G. Kuhn, "Efficient, Portable Template Attacks", in IEEE Transactions on Information Forensics and Security, vol. 13, no. 2, 2018, pp. 490–501.

[GV18] Grosso, Vincent. "Scalable key rank estimation (and key enumeration) algorithm for large keys", In CARDIS 2018, pp. 80–94.

[DW19] Liron David and Avishai Wool. "Poly-logarithmic side channel rank estimation via exponential sampling", In Cryptographers' Track at the RSA Conference, pp. 330-349.

[AMP+19] Azouaoui, Melissa, Romain Poussier, François-Xavier Standaert, and Vincent Verneuil, "Key Enumeration from the Adversarial Viewpoint", In CARDIS 2019, pp. 252–267.

[ZZD+20] Zhang, Ziyue, A. Adam Ding, and Yunsi Fei. "A Fast and Accurate Guessing Entropy Estimation Algorithm for Full-key Recovery", CHES 2020, pp. 26–48.

[DW21] L. David and A. Wool, "Rank estimation with bounded error via exponential sampling", J Cryptogr Eng, Aug. 2021, doi: 10.1007/s13389-021-00269-4.

[Lit22] ChipWhisperer Lite, `https://rtfm.newae.com/Starter%20Kits/ChipWhisperer-Lite/`.

[Com] The Common Criteria web site. `https://https://www.commoncriteriaportal.org/`

[EMV] The EMVCo Certification Process. `https://www.emvco.com/processes-forms/certification/`