Back to Massey: Impressively fast, scalable and tight security evaluation tools

Marios O. Choudary and P. G. Popescu *

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

Abstract. None of the existing rank estimation algorithms can scale to large cryptographic keys, such as 4096-bit (512 bytes) RSA keys. In this paper, we present the first solution to estimate the guessing entropy of arbitrarily large keys, based on mathematical bounds, resulting in the fastest and most scalable security evaluation tool to date. Our bounds can be computed within a fraction of a second, with no memory overhead, and provide a margin of only a few bits for a full 128-bit AES key.

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

1 Introduction

Side-channel attacks are powerful tools to extract secret information from hardware devices, such as the cryptographic microcontrollers used in banking smartcards. These 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 practical side-channel 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.

Recent advances in side-channel attacks have focused on the problem of estimating the rank 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 recent algorithms represent very useful tools for security evaluators that need to estimate the security of a given device. The algorithm proposed by Veyrat-Charvillon et al. [7] was the first method that could estimate the rank of a full 128-bit key, albeit with a considerable error margin. More recent algorithms [11,13,12] 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.

But none of these algorithms can scale for large keys composing of more than 256 bytes (e.g. an RSA 2048 or 4096 bit key), while at the same time providing tight bounds. Furthermore, even for smaller key sizes (e.g. 128-bit AES key), existing approaches can deviate from the actual security metric.

* We thank our God, the One God in Three Persons: Father, Son and Holy Spirit, for this work.

In this paper, we present sound mathematically-derived bounds for the guessing entropy, which allow us to evaluate the security of devices using arbitrary large keys (even 512 bytes or more). These have no memory requirements, can be computed instantaneously and provide bounds within a few bits.

2 Background: side-channel attacks and key enumeration

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 [1], CPA [3] or Template Attacks [2] 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 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, not just in the case of AES, as we shall show below.

To deal with this combination problem in the context of side-channel attacks, two kinds of combination algorithms have emerged in recent years: key enumeration and rank estimation algorithms. Key enumeration algorithms [5,14] 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 directly related to our paper, are the rank estimation algorithms. These algorithms provide an estimate of the full *key rank*, i.e. the number of keys we should try until finding the correct one if we were to apply a similar approach to key enumeration. The great advantage of rank estimation algorithms is that we can estimate the key rank even if this rank is very high (e.g. 2^{80} or larger), whereas enumerating such large number of keys is computationally infeasible. For security evaluations, this was until now probably the most convenient tool, since it can quickly estimate the security of a device. However, it is important that these rank estimation algorithms provide some guarantee of their bounds, since otherwise their output can be misleading.

Veyrat-Charvillon et al. [7] proposed the first efficient rank estimation algorithm for 128-bit keys, which 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. More recently, new algorithms [11,13,12] have improved the speed and tightness of the rank estimation. Among these, the histogram-based approach of Glowacz et al. [11] is probably the fastest and scales well even up to keys composed of 128 bytes (e.g. 1024-bit RSA key).

Nevertheless, none of these recent algorithms can scale efficiently to larger cryptographic keys, e.g. 2048-bit (256 bytes) or 4096-bit (512 bytes) keys, such as common RSA keys used for public key encryption. We have tested the C implementation of Glowacz et al. [11] on 256 subkey bytes and it took about 64 seconds per iteration (using the default N=2048 bins and the merge parameter set to two, i.e. doing a pre-computation step where lists of subkey probabilities are first combined two by two; for merge=3 the memory requirements killed the program), while for 512 subkey bytes (merge = 2) the memory requirements killed again the program[1]. The algorithm of Martin et al. [13] is also prohibitive for large keys, since it runs in $O(m^2 n \log n)$, where m is the number of subkeys and n is the number of possible values per subkey. Similarly, the PRO algorithm of Bernstein et al. [12] (which is the fastest of the two proposed by the authors) took about 5 hours for 256 subkey bytes and made the evaluation platform run out of swap memory (according to their results).

In contrast, our methods presented in the following sections allow us to obtain tight bounds instantaneously for arbitrarily large keys[2]. This is the first fully scalable security evaluation method proposed to date.

A possible scenario where such scalable methods are required, is the evaluation of side-channel attacks against the key loading operation. That is, side-channel attacks which target the transfer of keys from memory to registers, rather than the cryptographic algorithm itself. This was the case for example in the attacks of Oswald and Paar against the commercial Mifare DES-

[1] On a Intel i5 4-core CPU at 3.2 GHz, with 16 GB RAM.

[2] The only limitation being the numerical representation used by the computing machine.

Fire MF3ICD40 [6] or the attacks of Choudary and Kuhn [8] against the AVR XMEGA. Recent secure devices, such as the *A7101CGTK2: Secure authentication microcontroller* [23] support RSA encryptions with keys up to 4096 bits (512 bytes). Hence, in order to evaluate the security of these devices against full-key recovery side-channel attacks during the key loading operation, we need scalable rank estimation algorithms.

Furthermore, our methods are generally applicable, so they can be used in any other scenario where probability lists need to be combined to determine the approximate security of some system.

3 Experimental data

In order to present and demonstrate our results, we used two distinct datasets, one from a hardware AES implementation and the other from MATLAB simulated data. The first 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 *real* dataset.

The second dataset consists of simulated data, generated using MATLAB. 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.

$$\mathbf{x}_i = \mathrm{HW}(\text{S-box}(k \oplus p_i)) + r_i, \tag{1}$$

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.

3.1 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. To do this we use template attacks (TA) [2,8] on each subkey during the S-box lookup of the first AES round.[3]

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 $k \in \{1, \ldots, |\mathcal{S}|\}$, where $|\mathcal{S}|$ is the number of possible values (typically $|\mathcal{S}| = 256$ for one AES

[3] For the case of the real dataset, we first applied a Correlation Power Analysis (CPA) attack [3] to determine which is the leakage sample that leaks the most and then used this single sample in a template attack.

subkey byte). In the case of template attacks we obtain real probabilities and we shall often write $P(k|\mathbf{X}) = \mathrm{d}(k|\mathbf{X})$.[4]

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

4 Security metrics

To evaluate the security of a device against different side-channel attacks, an evaluator will typically use some evaluation metric. Standaert et al. [4] presented several such metrics for the case of attacks that target a single subkey at a time. Among these, we present below the guessing entropy and the conditional entropy. Afterwards we show how to derive scalable and tight bounds for these metrics. These allow us to obtain very efficient methods for estimating the security of devices against full-key recovery side-channel attacks.

4.1 Guessing entropy

In 1994, James L. Massey proposed a metric [16], known as the *Guessing Entropy*, to measure the average number of guesses that an attacker needs to find a secret after a cryptanalytic attack (such as our side-channel attacks).

Given the probability vectors $P(k|\mathbf{X})$ for each subkey obtained after a side channel attack, we can compute Massey's guessing entropy as follows. First, sort all the probability values $P(k|\mathbf{X})$, obtaining the sorted probability vector $\mathbf{p} = \{p_1, p_2, \ldots, p_{|\mathcal{S}|}\}$, where $p_1 = \max_k P(k|\mathbf{X})$, p_2 is the second largest probability and so on. Then, compute Massey's guessing entropy (GM) as:

$$\mathrm{GM} = \sum_{i=1}^{|\mathcal{S}|} i \cdot p_i. \tag{2}$$

Massey's guessing entropy represents the *statistical expectation* of the position of the correct key in the sorted vector of conditional probabilities. A similar measure is the *actual* guessing entropy (GE) [4], which provides the position of the correct key in the sorted vector of conditional probabilities. The GE is computed as follows: given the vector of sorted probabilities (or scores) $\mathbf{p} = \{p_1, p_2, \ldots, p_{|\mathcal{S}|}\}$, return the position of the probability corresponding to the correct key $k\star$[5]:

$$\mathrm{GE} = i, \quad p_i = P(k\star|\mathbf{X}). \tag{3}$$

As we can see from their definitions, both measures are computed from the posteriori probabilities of the keys given a set of leakage traces, but the GM

[4] 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 [15] to obtain pseudo-probabilities that work well with rank estimation algorithms.

[5] This measure assumes that an evaluator knows which is the correct key.

computes the *expected* position of the correct key, while the GE computes the *actual* position of the correct key. For this reason, the GE requires knowledge of the correct key, while the GM does not. Furthermore, we can see that averaging the GE over many experiments we approximate precisely the GM. Therefore, if we had exact probabilities, the GM would be the expected value of the GE.

In terms of usage, the GE is the most used measure in the side-channel evaluations published so far, mainly because it represents the actual position of the correct key and also because it can be computed even when using score-based attacks which do not output probabilities for each key (e.g. by sorting the keys according to their correlation after a correlation power attack and selecting the position of the correct key).

However, if we can obtain good probabilities for the key candidates (e.g. by using template attacks), then the GM can be a better evaluation tool, because as we said, the GM represents the expected value of the GE, but also because it is less affected by minor differences between probabilities. That is, when p_1 is much larger than the other probabilities, both measures will return 1 (or close to 1). On the other hand, for all scenarios in which the key is not easy to detect and the probabilities $p_1, p_2, \ldots, p_{|\mathcal{S}|}$ are very close to each other, any minor variation in the probabilties (e.g. due to measurement errors) will lead to possibly large variations of GE, while GM will provide the correct result, i.e. the expected value (which should be around $(|\mathcal{S}| + 1)/2$ if all the probabilities are very close).

Furthermore, the GM will allow us to derive the fast, scalable and tight bounds that we present in the following sections.

In our results, we shall show the logarithm (base 2) of the guessing entropy.

4.2 Conditional entropy

In information theory, the mutual information $I(X, Y)$ between two random variables X and Y is defined as:

$$I(X, Y) = H(X) - H(X|Y), \tag{4}$$

where

$$H(X) = -\mathbb{E} \log_2 P(X) = -\sum_{x \in \mathcal{X}} P(x) \cdot \log_2 P(x) \tag{5}$$

represents Shannon's entropy for the random variable X, and

$$H(X|Y) = \sum_{y \in \mathcal{Y}} P(y) H(X|Y = y) = -\sum_{y \in \mathcal{Y}} P(y) \sum_{x \in \mathcal{X}} P(x|y) \cdot \log_2 P(x|y) \tag{6}$$

represents the conditional entropy of X given Y. In short, the conditional entropy shows how much entropy (uncertainty) remains about the variable X when the variable Y is given.

As before, we are interested in knowing how much uncertainty (entropy) remains about the random variable K (representing the secret key byte k), when a set of leakage traces (represented by the variable L) is given; this can be quantified using the conditional entropy defined above. If K represents one key byte,

as in our setup, then $H(K) = 8$.[6] Using this notation we obtain the conditional entropy

$$H(K|L) = \sum_{\mathbf{X}\in\mathcal{L}} P(\mathbf{X}) H(K|L=\mathbf{X}) = -\sum_{\mathbf{X}\in\mathcal{L}} P(\mathbf{X}) \sum_{k\in\mathcal{K}} P(k|\mathbf{X}) \cdot \log_2 P(k|\mathbf{X}). \tag{7}$$

In practice, we can compute the conditional entropy from (7) using one of the following options[7]:

1. Compute an integral over the full leakage space, leading to the computationally-intensive form:

$$H(K|L) = -\int_{\mathbf{X}\in\mathcal{L}} P(\mathbf{X}) \sum_{k\in\mathcal{K}} P(k|\mathbf{X}) \cdot \log_2 P(k|\mathbf{X}) d\mathbf{X}. \tag{8}$$

2. Use Monte Carlo sampling from a limited subset of N traces:

$$H(K|L) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{k\in\mathcal{K}} P(k|\mathbf{X}_i) \cdot \log_2 P(k|\mathbf{X}_i). \tag{9}$$

The first form is computationally intensive, as for multi-dimensional leakage traces the integral in (8) needs to be computed over a multi-dimensional space. Therefore, in our experiments we used the second form, where N is the number of iterations (usually $N = 100$) over which we computed the second summation and the probabilities $P(k|\mathbf{X}_i)$ were obtained from template attacks on each iteration.

5 Tight bounds for guessing entropy

In this section, we explain how to adapt several known bounds (lower and upper) of the guessing entropy (GM) in the context of side-channel attacks, when we deal with a single list of probabilities (e.g. targeting a single subkey byte). These bounds can be used as a fast approximation of the GM (since they run in linear time, because they don't require the sorting operation that is necessary for the computation of GM), but their great advantage is in the context of multiple lists of probabilities (see next section).

5.1 Bounds for Massey's guessing entropy from probabilities

Arikan [19] presented a lower and an upper bound for GM. We can adapt these bounds to our side-channel context using the notation from previous sections, as follows:

$$\frac{1}{1+\ln|\mathcal{S}|} \left[\sum_{k=1}^{|\mathcal{S}|} p_k^{1/2}\right]^2 \leq \mathrm{GM} \leq \left[\sum_{k=1}^{|\mathcal{S}|} p_k^{1/2}\right]^2, \tag{10}$$

[6] We assume all key bytes are equally likely, in the absence of leakage information.

[7] There are other ways to estimate the conditional entropy, including several variants of the Monte Carlo method. Here we focused only on the two most popular such variants.

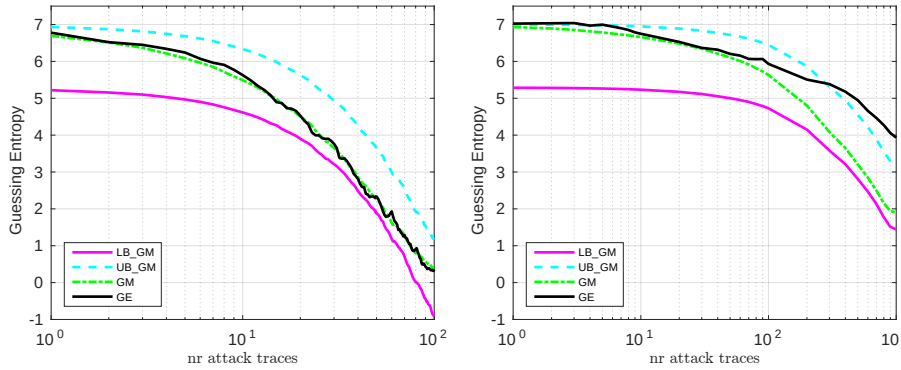

Fig. 1. GM, GE and GM bounds from probabilities for the simulated (left) and real (right) datasets, when targeting a single subkey byte. These are averaged results over 100 experiments.

with the important remark that in the lower and upper bounds the individual probabilities $p_k = P(k|X)$ do not need to be sorted. This means that both bounds can be computed in $O(|\mathcal{S}|)$. The upper bound of Arikan has been improved in [18], by Theorem 3, yielding:

$$\mathrm{GM} \leq \frac{1}{2}\left[\sum_{k=1}^{|\mathcal{S}|} p_k^{1/2}\right]^2 + \frac{1}{2} \leq \left[\sum_{k=1}^{|\mathcal{S}|} p_k^{1/2}\right]^2 .$$

Combining this with (10), we obtain the tighter relation:

$$\frac{1}{1+\ln|\mathcal{S}|}\left[\sum_{k=1}^{|\mathcal{S}|} p_k^{1/2}\right]^2 \leq \mathrm{GM} \leq \frac{1}{2}\left[\sum_{k=1}^{|\mathcal{S}|} p_k^{1/2}\right]^2 + \frac{1}{2}. \tag{11}$$

We shall refer to these lower and upper bounds as $\mathrm{LB}_{\mathrm{GM}}$ and $\mathrm{UB}_{\mathrm{GM}}$, respectively.

We show the results of using these bounds on the simulated (left) and real (right) datasets in Figure 1. We can make several observations. Firstly, the bounds are in both cases within 1-2 bits apart[8] for all values of the guessing entropy. Secondly, we see that for the simulated dataset the GM is very close to the GE, but for the real dataset the GE deviates considerably and even goes outside the upper bound of the GM. In all our experiments, we observed that the GM stays either close or below the GE. This can be explained by the fact that even if many probabilities are close to each other in value, small ordering

[8] While this is not as tight as other rank estimation algorithms, we shall see later that our bounds stay tight even when using a large number of target subkey bytes and that they are always sound (due to the mathematical demonstration), while existing rank estimations can provide estimation and calculation errors.

errors can have a higher impact on GE (which only depends on the order) than on GM (which only depends on the probability values).

As we shall show later, previous rank estimation algorithms, such as the one of Glowacz et al. [11], also tend to follow more the GM rather than the GE, because they also rely on the values of probabilities rather than the exact position of the correct key, even though such algorithms also use the value of the correct key in order to position their bounds closer to the actual position of the correct key. Nevertheless, both measures can be useful. If we need the exact position of the key for a particular set of measurements, then GE is the best tool. However, the GE cannot be computed for large number of target subkeys and is also subject to the particular measurements, i.e. noise can cause the correct subkey value to be ranked very bad, even though its probability is very close to those in the top, leading to a very high GE, while the GM will show a lower value. Hence, in such scenario the GM may actually provide a better intuition since with a new set of traces (e.g. the attacker), the correct subkey value could be ranked better, leading to a smaller GE. Furthermore, the fact that the GM will in general be below the GE (or very close to it in case it is slightly above) means that relying on the GM will provide a safer conclusion from a designer perspective. That is, if the resulting GM is above a desired security level for some scenario, then we can be confident enough that the GE will either be very close or above.

From the figure, we also see that GM always stays within the bounds. This is guaranteed, given the mathematical derivation. And as we shall see, the algorithmic approaches can introduce estimation errors and provide erroneous results that are neither between our bounds nor close to the expected GE.

Besides the above differences between GM and GE, what is most important in our context, is that we can obtain very fast and scalable bounds for GM.

Finally, we mention another important difference between GM and GE, namely for the computation of GE we need knowledge of the real key (so we can compute its position), while for the GM we do not. Hence, our GM bounds allow anyone to estimate the security of a device, while previous rank estimations could only be used by evaluators having access to the real key (target) values.

5.2 Bounds from conditional entropy

We now show how to bound Massey's guessing entropy as a function of the conditional entropy, using Massey's inequality [16] and McEliece and Yu inequality [17]. This allows us to obtain a general relation between the guessing entropy and the conditional entropy in the context of side-channel attacks.

Let $H(K|L = \mathbf{X})$ be the conditional entropy obtained for the set of leakage traces $\mathbf{X}$. Applying Massey's inequality [16] to GM and $H(K|L = \mathbf{X})$, we obtain the following upper bound for the conditional entropy:

$$2 + \log(\mathrm{GM} - 1) \geq H(K|L = \mathbf{X}). \tag{12}$$

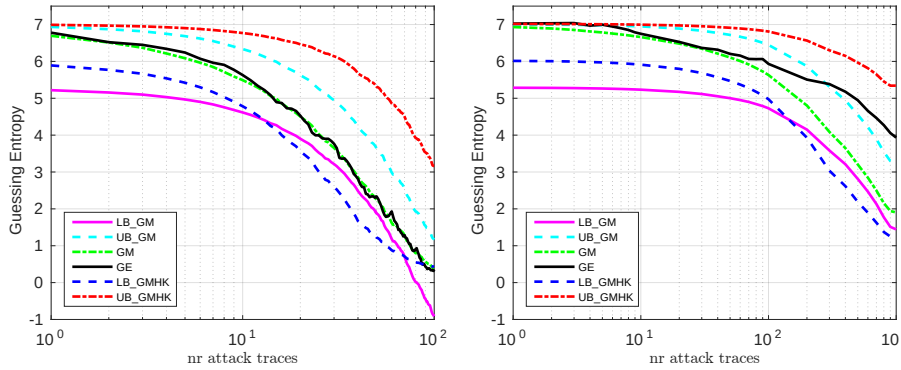

Fig. 2. GM, GE and GM bounds from probabilities and conditional entropy $H(K|L)$ for the simulated (left) and real (right) datasets, when targeting a single subkey byte. These are averaged results over 100 experiments.

Then, applying McEliece and Yu's inequality [17], we obtain a lower bound for the conditional entropy as:

$$H(K|L = \mathbf{X}) \geq \frac{2 \log |\mathcal{S}|}{|\mathcal{S}| - 1}(\mathrm{GM} - 1). \tag{13}$$

Using (12) and (13), we obtain lower and upper bounds for GM as a function of the conditional entropy:

$$2^{H(K|L=\mathbf{X})-2} + 1 \leq \mathrm{GM} \leq \frac{|\mathcal{S}| - 1}{2 \log |\mathcal{S}|} H(K|L = \mathbf{X}) + 1. \tag{14}$$

We refer to these as $\mathrm{LB}_{\mathrm{GMHK}}$ and $\mathrm{UB}_{\mathrm{GMHK}}$, respectively.

Remark 1. The left inequality in (14) is true when $H(K|L = \mathbf{X})$ is greater than 2 bits.

We show these bounds in Figure 2, along with the previous bounds, for both the simulated (left) and real (right) datasets. We can see that in both cases the lower bound $\mathrm{LB}_{\mathrm{GMHK}}$ stays within 1 bit of GM for all values of GM, while the upper bound $\mathrm{UB}_{\mathrm{GMHK}}$ deviates substantially, even more than 3 bits from the GM. Secondly, in both results we see that $\mathrm{UB}_{\mathrm{GM}}$ is a much better upper bound than $\mathrm{UB}_{\mathrm{GMHK}}$. We observed this in all our experiments. Combining the best of these bounds, we can say that for the lower bound we should use the maximum between $\mathrm{LB}_{\mathrm{GMHK}}$ and $\mathrm{LB}_{\mathrm{GM}}$, while for the upper bound we should use $\mathrm{UB}_{\mathrm{GM}}$.

6 Impressive scaling: scalable bounds for guessing entropy

We now show how to scale the bounds presented in the previous section to arbitrarily many lists of probabilities, so they can be used to estimate the security of a full AES key (16-32 subkey bytes) or even RSA key (128-512 subkey

bytes), while being computable in time that increases *linearly* with the number of subkeys targeted.

In the following, we shall use the notation GM^f to refer to the GM for the full key, n_s for the number of subkeys composing the full target key, and $|\mathcal{S}|^{n_\mathrm{s}}$ for the number of possible full key values (e.g. $n_\mathrm{s} = 16, |\mathcal{S}|^{n_\mathrm{s}} = 2^{128}$ for AES-128).

6.1 Using bounds of GM for evaluation of full key

In Section 5.1, we showed how to derive tight bounds for GM from probabilities in the case of a single subkey byte. Considering the shape of the summation involved in (11), we need a way to avoid the computation of all the possible probabilities in the set of cross-probabilities from the full key space. Splitting the full sum into groups of partial sums leads to our main result:

Theorem 1. *(LB_{GM} and UB_{GM} for full key) Let $p^i_1, p^i_2, ..., p^i_{|\mathcal{S}|}$ be the probabilities for the $i = 1, 2, ..., n_s$ target subkey. Then we have*

$$\frac{1}{1+\ln|\mathcal{S}|^{n_\mathrm{s}}}\prod_{i=1}^{n_\mathrm{s}}\left[\sum_{k=1}^{|\mathcal{S}|}\sqrt{p^i_k}\right]^2 \leq GM^f \leq \frac{1}{2}\prod_{i=1}^{n_\mathrm{s}}\left[\sum_{k=1}^{|\mathcal{S}|}\sqrt{p^i_k}\right]^2 + \frac{1}{2}.$$

Proof. Considering the $\mathrm{LB_{GM}}$ and $\mathrm{UB_{GM}}$ bounds for the full key, we have

$$\frac{1}{1+\ln|\mathcal{S}|^{n_\mathrm{s}}}\left[\sum_{k=1}^{|\mathcal{S}|^{n_\mathrm{s}}}\sqrt{p^f_k}\right]^2 \leq \mathrm{GM}^f \leq \frac{1}{2}\left[\sum_{k=1}^{|\mathcal{S}|^{n_\mathrm{s}}}\sqrt{p^f_k}\right]^2 + \frac{1}{2}.$$

Then, adding the fact that the new probabilities are combined as a product of n_s probabilities from target subkeys, i.e. $p^f_k = \prod_{i=1}^{n_\mathrm{s}} p^i_j$, with $j = j(k,i) \in \{1, 2, ..., |\mathcal{S}|\}$ and factoring accordingly, we obtain that

$$\sum_{k=1}^{|\mathcal{S}|^{n_\mathrm{s}}}\sqrt{p^f_k} = \left[\sum_{k=1}^{|\mathcal{S}|}\sqrt{p^1_k}\right]\cdot\left[\sum_{k=1}^{|\mathcal{S}|}\sqrt{p^2_k}\right]\cdot\ldots\cdot\left[\sum_{k=1}^{|\mathcal{S}|}\sqrt{p^{n_\mathrm{s}}_k}\right]$$

i.e.

$$\sum_{k=1}^{|\mathcal{S}|^{n_\mathrm{s}}}\sqrt{p^f_k} = \prod_{i=1}^{n_\mathrm{s}}\left[\sum_{k=1}^{|\mathcal{S}|}\sqrt{p^i_k}\right]$$

and we are done.

$\mathrm{UB_{GM}}$ runs in $O(|\mathcal{S}|)$ and for full key runs in $O(n_\mathrm{s}\cdot|\mathcal{S}|)$, i.e. the computation time only increases *linearly* with the number of subkey bytes.

Remark 2. We can estimate the number of bits δ between our $\mathrm{LB_{GM}}$ and $\mathrm{UB_{GM}}$ bounds for the full key as $\delta = \log 2(\mathrm{UB_{GM}}) - \log 2(\mathrm{LB_{GM}}) = \log 2(\mathrm{UB_{GM}}/\mathrm{LB_{GM}})$. Ignoring the $1/2$ factor (which is negligible as the number of subkeys increases), we obtain the following approximation:

$$\delta \approx \log 2\left(\frac{1+\ln|\mathcal{S}|^{n_\mathrm{s}}}{2}\right) = \log 2\left(\frac{1+n_\mathrm{s}\cdot\ln|\mathcal{S}|}{2}\right) \text{ bits.}$$

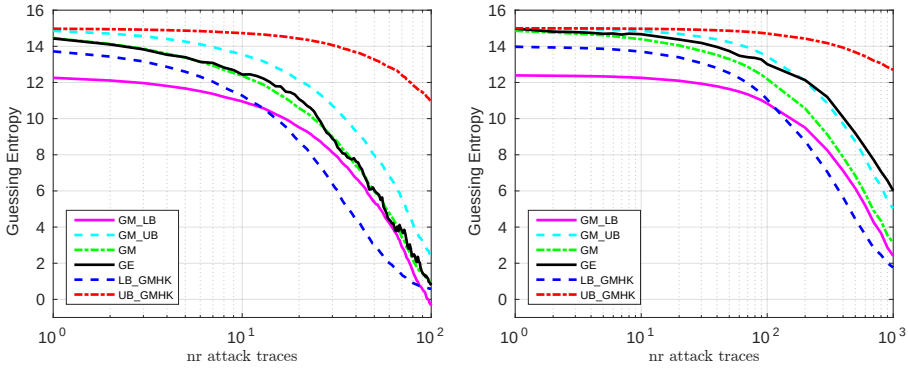

Fig. 3. GM, GE and GM bounds for the simulated (left) and real (right) datasets, when targeting two subkey bytes. These are averaged results over 100 experiments.

6.2 Using bounds of $H(K|L)$ for evaluation of full key

Assuming independence between target subkeys and considering the bounds presented into (14) applied for the full key space yields

Theorem 2. *(LB_{GMHK} and UB_{GMHK} for full key) Let $H(K|L = \mathbf{X}_i)$ be the conditional entropy for the $i = 1, 2, ..., n_s$ target subkey, then*

$$2^{\sum_{i=1}^{n_s} H(K|L=\mathbf{X}_i)-2} + 1 \leq GM^f \leq \frac{|\mathcal{S}|^{n_s} - 1}{2 \log |\mathcal{S}|^{n_s}} \sum_{i=1}^{n_s} H(K|L = \mathbf{X}_i) + 1.$$

Proof. Considering (14) applied for full key space yields

$$2^{H(K^f|L^f=\mathbf{X})-2} + 1 \leq \mathrm{GM}^f \leq \frac{|\mathcal{S}|^{n_s} - 1}{2 \log |\mathcal{S}|^{n_s}} H(K^f|L^f = \mathbf{X}) + 1,$$

where $H(K^f|L^f = \mathbf{X})$ is the joint conditional entropy for all n_s target subkeys. And because of the assumed independence between target subkeys, yields from [20, Theorem 2.6.6] that

$$H(K^f|L^f = \mathbf{X}) = \sum_{i=1}^{n_s} H(K|L = \mathbf{X}_i),$$

which gives us the wanted result.

Again, both bounds $\mathrm{LB}_{\mathrm{GMHK}}$ and $\mathrm{UB}_{\mathrm{GMHK}}$ run in $O(|\mathcal{S}|)$ and for full key in $O(n_s \cdot |\mathcal{S}|)$, i.e. they both scale *linearly* with the number of target subkeys.

In Figure 3, we show our scaled bounds for GM^f for the case of targeting two subkey bytes. We computed both GM^f and GE^f by first obtaining the cross-product of probabilities between the first two subkeys in the datasets. We see

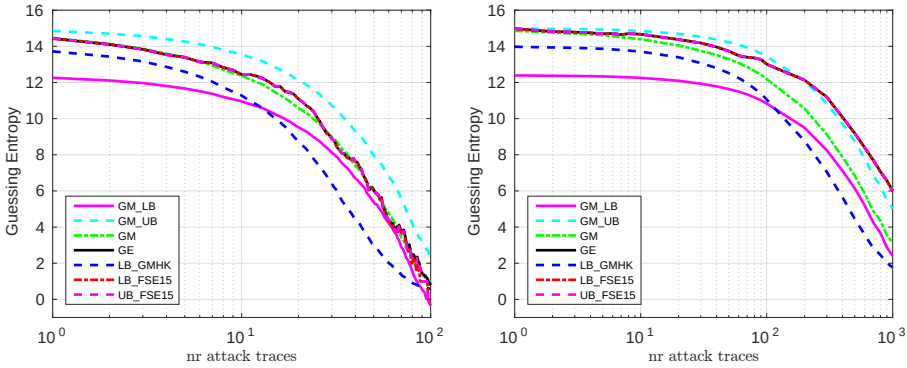

Fig. 4. GM, GE GM bounds and FSE15 bounds for the simulated (left) and real (right) datasets, when targeting two subkey bytes. These are averaged results over 100 experiments.

again that our bounds are correct for GM^f, while GE^f goes slightly outside the bounds for the real dataset, as observed also when targeting a single subkey byte (refer to Sections 4.1 and 5.1 for an explanation). $\mathrm{LB}_{\mathrm{GM}}$ and $\mathrm{UB}_{\mathrm{GM}}$ stay within about 2 bits in both the simulated and real experiments from Figure 3. $\mathrm{UB}_{\mathrm{GMHK}}$ stays again far from GM^f, but $\mathrm{LB}_{\mathrm{GMHK}}$ is tighter than $\mathrm{LB}_{\mathrm{GM}}$ for higher values of GM^f, as we saw also in the case of a single subkey byte. This confirms that for the lower bound we should use the maximum from $\mathrm{LB}_{\mathrm{GM}}$ and $\mathrm{LB}_{\mathrm{GMHK}}$, while for the upper bound we should use $\mathrm{UB}_{\mathrm{GM}}$.

6.3 GM bounds from element positioning

Considering the computational advantage of working with scalable bounds for GM, in [21,22], based on an inequality related to positioning an element into a sorted matrix, the authors present new scalable bounds for GM as follows:

$$\prod_{i=1}^{n_s} \mathrm{GM}_i \leq \mathrm{GM}^f \leq |\mathcal{S}|^{n_s} - \prod_{i=1}^{n_s} (|\mathcal{S}| - \mathrm{GM}_i)\,,$$

where GM_i is the guessing entropy of the $i = 1, 2, ..., n_s$ target subkey.

In order to answer the authors, which left the improvement of these bound as open question, we accepted the challenge and refined both bounds. But because we observed (see Figure 7 in Appendix) that these improved bounds are still much weaker than the $\mathrm{LB}_{\mathrm{GM}}$ and $\mathrm{UB}_{\mathrm{GM}}$ bounds, we leave the results and proofs of this part in Appendix.

6.4 GM bounds versus the FSE 2015 rank estimation

As mentioned in Section 2, several algorithms [7,11,13,12] have been proposed in recent years to estimate the rank of the correct full key. Among them, the

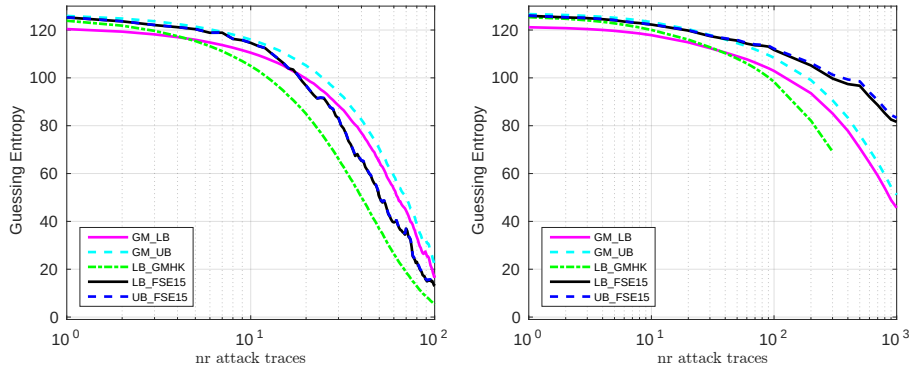

Fig. 5. GM bounds and FSE15 bounds for the simulated (left) and real (right) datasets, when targeting 16 subkey bytes. These are averaged results over 100 experiments.

rank estimation of Glowacz et al. [11], to which we shall refer as FSE15 from now on, is probably the fastest and scales well for keys up to 128 bytes (e.g. 1024-bit RSA key). For this reason, in Figure 4, we compare our GM bounds to the results of FSE15 (using their C implementation) for the case of two subkeys, for both the simulated (left) and real (right) datasets. The results show that the FSE15 bounds generally stay within our GM bounds in both data sets, but for the real data set they go slightly beyond our bounds, following the GE^f.

In Figure 5, we compare our GM bounds and the FSE15 bounds for the full 16-byte AES key, again for the simulated (left) and real (right) datasets. From this figure, we see that our GM bounds are tight even for the full 128-bit AES key (16 subkeys), $\mathrm{LB}_{\mathrm{GM}}$ and $\mathrm{UB}_{\mathrm{GM}}$ staying within 5 bits of each other in both experiments. From the experiments on the real dataset, we also see that $\mathrm{LB}_{\mathrm{GMHK}}$ fails once the guessing entropy decreases below 70 bits, due to numerical limitations[9] when computing the bound at this point. Comparing our bounds to the FSE15 bounds in the simulated data set, we can see that for higher values of GM^f, the FSE15 bounds stay within our GM bounds, but afterwards they start to deviate, due to the deviation of GE^f from GM^f. A similar pattern is observed with the real data set.

From these experiments, we can see that the FSE15 bounds follow the GE, while our GM bounds follow the GM, and in general the FSE15 bounds stay within our GM bounds, due to the GE being close to the GM. Depending on the requirements, one may prefer to use one tool or the other. However, while less tight than the FSE15 bounds, our GM bounds have the advantage of being scalable to arbitrarily large number of subkeys, while any of the previous rank estimation algorithms, including the FSE15 bounds are limited due to memory and computation time to some maximum size.

[9] We used MATLAB R2015b.

Table 1. Comparing GM bounds with rank estimation algorithms.

Method	Good	Bad
FSE '15 [11]	Very fast (< 1s) for up to $n_s = 128$. Very tight bounds.	Not scalable for $n_s \geq 256$ (slow).
Asiacrypt '15 [13]	Tight bounds (similar to FSE'15). Fast for $n_s = 16$ ($1 - 4$ s).	Memory can be prohibitive for large key sizes. Not scalable: $O({n_s}^2 \lvert\mathcal{S}\rvert \log \lvert\mathcal{S}\rvert)$ (very slow for large key size).
Eurocrypt '15 [10]	Success Rate (SR) for full key as function of time complexity. Time: $O(n_s \cdot Nmax^2)$	No method to go from SR to key rank for a given set of leakage traces. Not scalable for tighter bounds (would require large Nmax).
PRO [12]	Fast for $n_s = 16$ (about 7 s). Tight bounds as function of α (can be slow).	Can run out of RAM for large keys ($\alpha = 2^{13}$). Takes about 5 hours for large keys, not scalable.
Eurocrypt '13 [7]	Bounds within 6 bits for key ranks smaller than 2^{30}, when targetting a 128-bit key.	Run time: 5s–900s. Bound up to 20-30 bits for large key ranks ($2^{50} - 2^{100}$). Memory: 4k – 83 MB. Weak bounds (40 bit) for small key rank.
CARDIS '14 (Ye) [9]	Acceptable bound, unclear for 16-bit (close to Eurocrypt'13).	Computationally intensive. Scalability may be bad (not evaluated).
CT-RSA '17 [21]	Fast and scalable: $O(n_s \cdot (\lvert\mathcal{S}\rvert \log \lvert\mathcal{S}\rvert))$.	Weak lower bound. Very weak upper bound.
LB_{GM} and UB_{GM}	Guaranteed bounds for GM. Fastest method to date. Scales to arbitrarily large n_s: $O(n_s \cdot \lvert\mathcal{S}\rvert)$. Tight bounds (5 bits for 128-bit key). Constant (negligible) memory.	

6.5 GM bounds versus rank estimation algorithms

Given the development of several rank estimation algorithms in the recent years [11,13,10,12,7,9,21], we provide in Table 1 a comparison of these algorithms with our GM bounds in terms of computation time, memory requirements, tightness and accuracy for different key sizes.

7 Conclusion

In this paper we have presented the first fully scalable, tight, fast and sound method for estimating the guessing entropy from arbitrarily many lists of probabilities. This method, based on mathematically-derived bounds, allows us to

estimate within a few bits the guessing entropy for a 128-bit key, but can also be used to estimate the guessing entropy for cryptographic keys of 1024 bytes (8192 bits) and much larger, which is not possible with any of the previous rank estimation algorithms due to memory or running time limitations.

As an illustration of this capability, we show in Figure 6, the computation of our bounds for a 1024-byte (8192-bit) key[10]. For simplicity and easier reproducibility, we have used the simulated dataset, where we have replicated the 16 lists of probabilities (one for each target subkey byte) 64 times, so we get 1024 lists of probabilities[11]. The plot shows our $\mathrm{LB}_{\mathrm{GM}}$ and $\mathrm{UB}_{\mathrm{GM}}$ bounds for this case. Note in the right side, that the margin between our bounds is about 11.5 bits, which is expected from Remark 2. We leave this figure as a reference for future methods, as none of the previous ones could be used to obtain this plot.

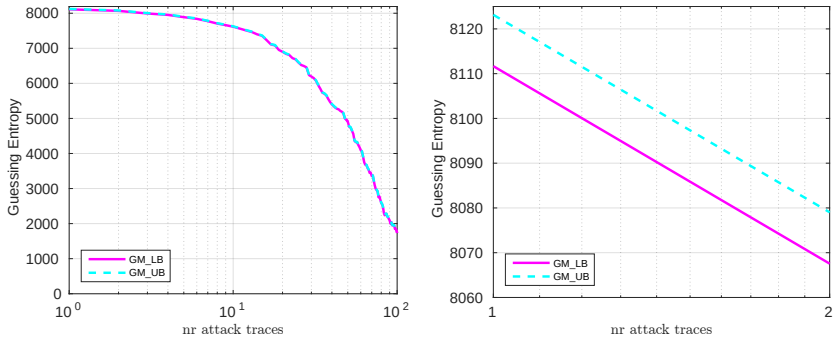

Fig. 6. $\mathrm{LB}_{\mathrm{GM}}$ and $\mathrm{UB}_{\mathrm{GM}}$ bounds for a 1024-byte (8192-bit) key, computed from 1024 lists of probabilities. We used a logarithmic Y-axis, as in the rest of the figures. On the right, we show a zoom for $n_{\mathrm{a}} = 1$ and $n_{\mathrm{a}} = 2$ attack traces only.

[10] Computed using symbolic variables and variable precision arithmetic features of MATLAB, within 13 seconds per iteration.

[11] However, the computation of our bounds would work equally well for any set of lists of probabilities.

Acknowledgement: This work has partially been funded by University Politehnica of Bucharest, through the Excellence Research Grants Program, UPB - GEX. Identiers: UPB - EXCELENŢĂ - 2016, *Noi metode pentru modelarea consumului de energie în dispozitivele electronice* and *Managementul Eficient al Datelor în Sisteme Distribuite Moderne bazat pe Noi Limite ale Entropiei (acronym:* **BigDataH***)*, Contract numbers: 17&18/26.09.2016.

References

1. Paul Kocher, Joshua Jaffe and Benjamin Jun, "Differential Power Analysis", CRYPTO 1999.
2. S. Chari, J. Rao, and P. Rohatgi, "Template Attacks", CHES 2002, Springer, 2003, LNCS 2523, pp 51–62.
3. 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.
4. 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.
5. 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.
6. Oswald, David, and Christof Paar, "Breaking Mifare DESFire MF3ICD40: Power Analysis and Templates in the Real World", In CHES 2011, LNCS 6917, pp. 207–222.
7. Veyrat-Charvillon, Nicolas, Benoît Gérard, and François-Xavier Standaert, "Security Evaluations beyond Computing Power", EUROCRYPT 2013, LNCS 7881, pp. 126–41.
8. O. Choudary and M. G. Kuhn, "Efficient Template Attacks", CARDIS 2013, Berlin, 27–29 November 2013, LNCS 8419, pp. 253–270.
9. Ye, Xin, Thomas Eisenbarth, and William Martin, "Bounded, yet sufficient? How to determine whether limited side channel information enables key recovery", CARDIS 2014.
10. Duc, Alexandre, Sebastian Faust, and F.-X. Standaert. "Making masking security proofs concrete". Eurocrypt 2015, pp. 401–429, 2015.
11. 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.
12. 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`.
13. Martin, Daniel P., Jonathan F. OConnell, Elisabeth Oswald, and Martijn Stam, "Counting Keys in Parallel After a Side Channel Attack". ASIACRYPT 2015, LNCS 9453, pp. 313–37.
14. 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`.

15. Choudary, Marios O., Romain Poussier and François-Xavier Standaert, "Score-based vs. Probability-based Enumeration – a Cautionary Note –", Indocrypt 2016, to appear.
16. J.L. Massey, *Guessing and Entropy*, IEEE ISIT, 1994, p. 204.
17. R.J. McEliece and Z. Yu, *An Inequality On Entropy*, IEEE ISIT 1995, p. 329, ISBN 0-7803-2453-6.
18. S. Boztaş, *Comments on "An Inequality on Guessing and Its Application to Sequential Decoding"*, IEEE Transactions on Information Theory 43(6) 1997.
19. E. Arikan, *An Inequality on Guessing and Its Application to Sequential Decoding*, IEEE Transactions on Information Theory 42(1) 1996.
20. T.M.Cover, J.A. Thomas, *Elements of Information Theory - second edition*, Wiley 2006 ISBN: 0-471-24195-4
21. L. David, A. Wool, *A Bounded-Space Near-Optimal Key Enumeration Algorithm for Multi-subkey Side-Channel Attacks*, CT-RSA 2017.
22. L. David, A. Wool, *A bounded-space near-optimal key enumeration algorithm for multi-dimensional side-channel attacks*, Cryptology ePrint Archive, Report 2015/1236 (2015). http://eprint.iacr.org/2015/1236
23. NXP A710x family: Secure authentication microcontroller. `http://www.nxp.com/products/identification-and-security/secure-authentication-and-anti-counterfeit-technology/secure-authentication-microcontroller:A710X_FAMILY`. Last visited: October 11th, 2016.

A GM bounds from element positioning

Considering the computational advantage of working with scalable bounds for GM, in [21,22], based on an inequality related to positioning an element into a sorted matrix, the authors present new scalable bounds for GMas follows:

$$\prod_{i=1}^{n_s} \mathrm{GM}_i \leq \mathrm{GM}^f \leq |\mathcal{S}|^{n_s} - \prod_{i=1}^{n_s} (|\mathcal{S}| - \mathrm{GM}_i)\,, \tag{15}$$

where GM_i is the guessing entropy of the $i = 1, 2, ..., n_s$ target subkey.

In order to answer the authors, which left the improvement of these bound as open question, we accept the challenge and refine both bounds as follows.

First, for the upper bound, we may observe that the base inequality involved here, representing the positioning of an element into a sorted matrix built from the combination of products of elements of two vectors x and y, see [22] Fig3, i.e.

$$ij \leq rank(x_i, y_j) \leq n^2 - (n-i)(n-j), \quad i, j = 1, 2, ...n$$

is weak and its meaning is unclear (the position of an element could not go so far). A more meaningful form of this inequality is a tighter one, like

$$ij \leq rank(x_i, y_j) \leq n^2 - (n-i+1)(n-j+1) + 1$$

and considering its right hand side applied for our context reveals an improved upper bound for GM. But this improvement is almost unnoticeable in our experiments, so we will not use it for the following discussions.

Now, for the lower bound we define as $I_1, I_2, ..., I_m$ nonempty, pairwise disjoint subsets of $\{1, 2, ..., n_s\}$, with $\cup_j I_j = \{1, 2, ..., n_s\}$, $j = 1, 2, ...m$, $m \leq n_s$ and with the same number of elements ($|I_1| = |I_2| = ... = |I_m| = n_s/m$).

We further define as GM_{I_j} the guessing entropy of the combined target subkeys from the subset I_j, $j = 1, 2..., m$. We have the following

Theorem 3. *(EP bounds for GM) Considering the above we have*

$$\prod_{i=1}^{n_s} GM_i \leq \prod_{j=1}^{m} GM_{I_j} \leq GM^f.$$

Proof. The left hand side of the inequality follows directly from the combination of the left hand sides of the inequality of [21, Theorem 2] particularized for $d = j$, with $j = 1, 2, ..., m$, i.e. from inequalities

$$\prod_{i \in I_j} \mathrm{GM}_i \leq \mathrm{GM}_{I_j}, \quad j = 1, 2, ...m.$$

For the right hand side of the inequality we notice that grouping the target subkeys into subgroups of the same size, yields lists of probabilities per the new target subkeys of the same size, so we can still apply the left hand side of the generalized base inequality form [21] in order to obtain the wanted result.

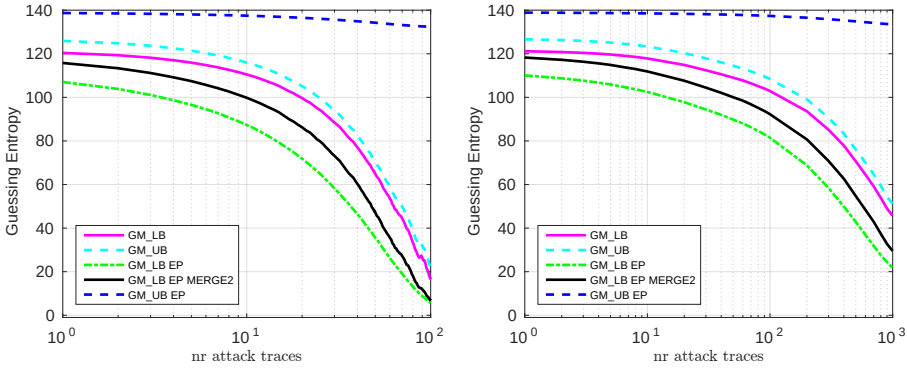

Fig. 7. GM bounds and EP bounds for the simulated (left) and real (right) datasets, when targeting 16 subkey bytes. These are averaged results over 100 experiments.

It is easy to observe that the previous result is a refinement of the lower bound presented into [21,22]. Also we can refine furthermore the result by using a good grouping strategy. For example in practice $n_s = 16$ so a lower bound will arrive from grouping into 2 groups of 8 elements, then a weaker lower bound will be derived by grouping into 4 groups of 4 elements, and so on until we group each individual element, which is the lowest bound, i.e.

Corollary 1.

$$\prod_{i=1}^{n_s} GM_i = \prod_{j=1}^{n_s} GM_{I_j} \leq \prod_{j=1}^{8} GM_{I_j} \leq \prod_{j=1}^{4} GM_{I_j} \leq \prod_{j=1}^{2} GM_{I_j} \leq GM^f.$$

Because of the computational limitations, in our experiments we have only considered grouping as much as two elements per group, i.e. two subkeys. In Figure 7, we compare these bounds, with with our GM bounds. We see that even after merging, the two EP bounds are weaker than our $\mathrm{LB}_{\mathrm{GM}}$ and $\mathrm{UB}_{\mathrm{GM}}$ bounds.