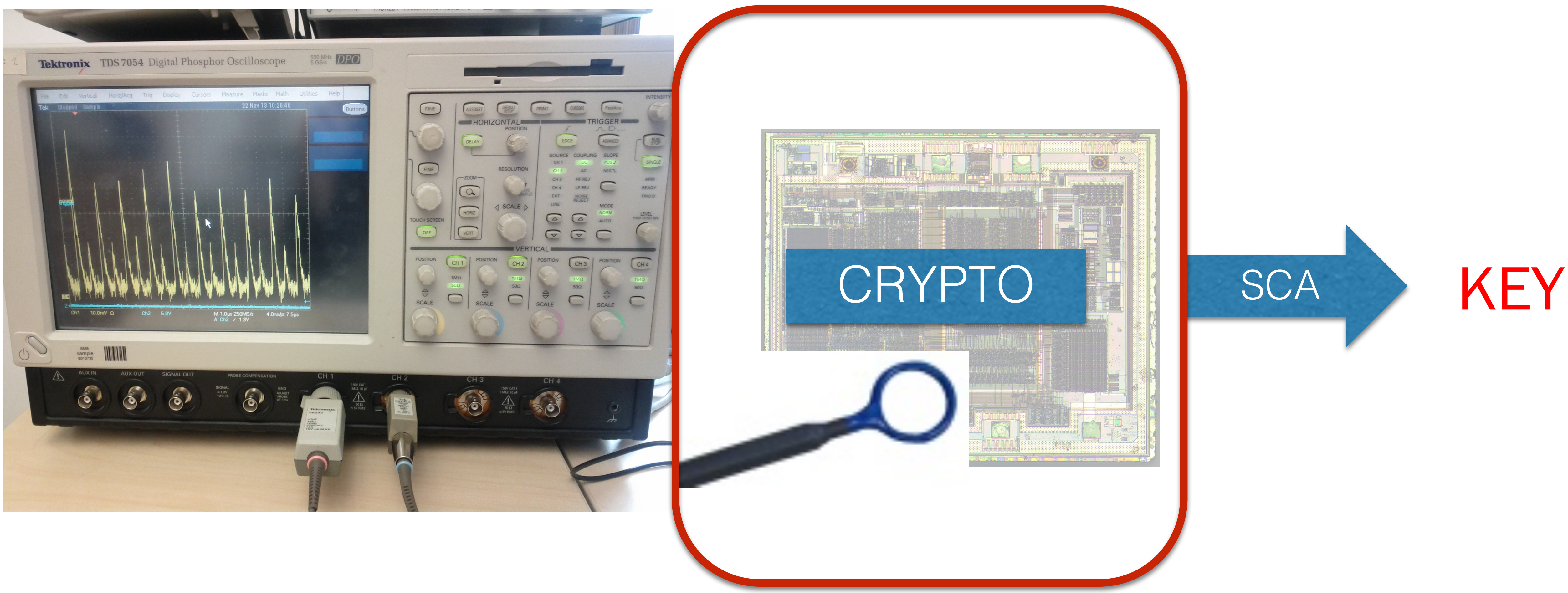
Back to Massey: Impressively fast, scalable and tight security evaluation tools

Marios O. Choudary and Pantelimon George Popescu
University Politehnica of Bucharest
CHES 2017, Taipei

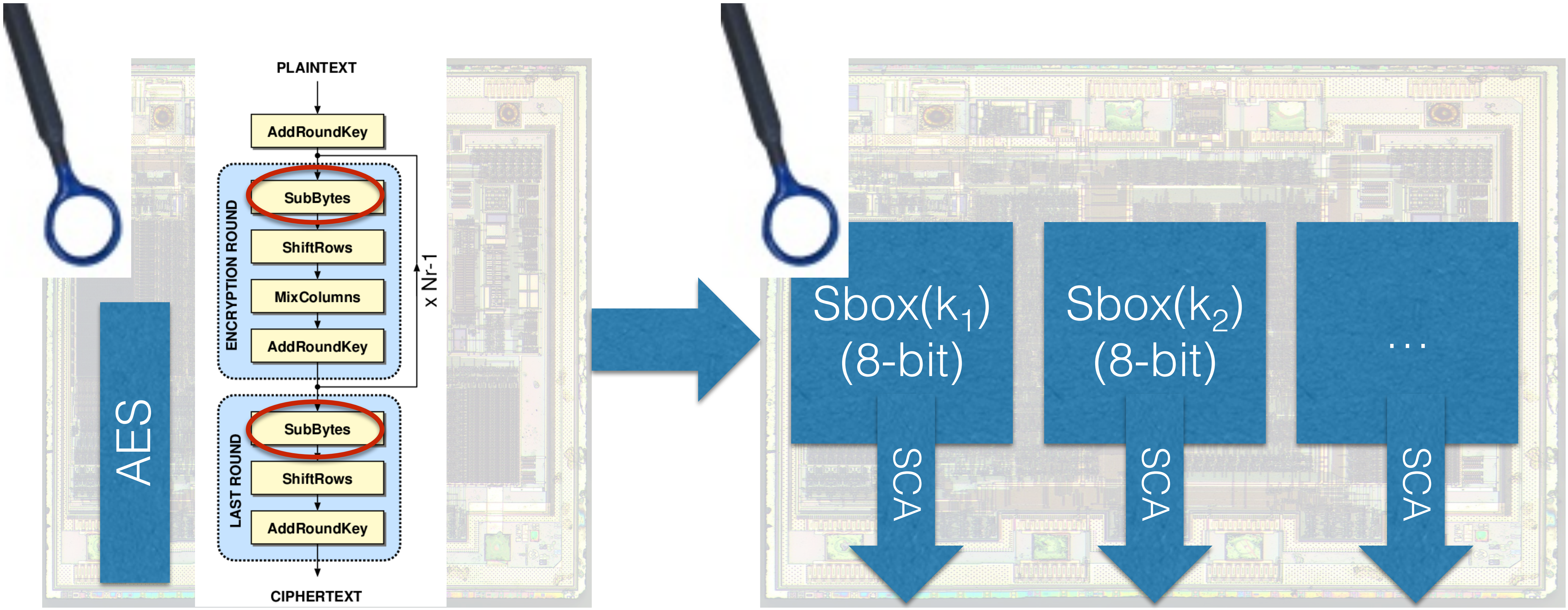
Side Channel Attacks (SCA)

- Are powerful tools to extract data (e.g. secret keys) used in cryptographic algorithms



SCA on crypto algorithms

- Improved brute-force attacks by Divide and Conquer strategy:



- Target 8-bit subkeys instead of full crypto key (e.g. 128-bit)

Security Evaluations

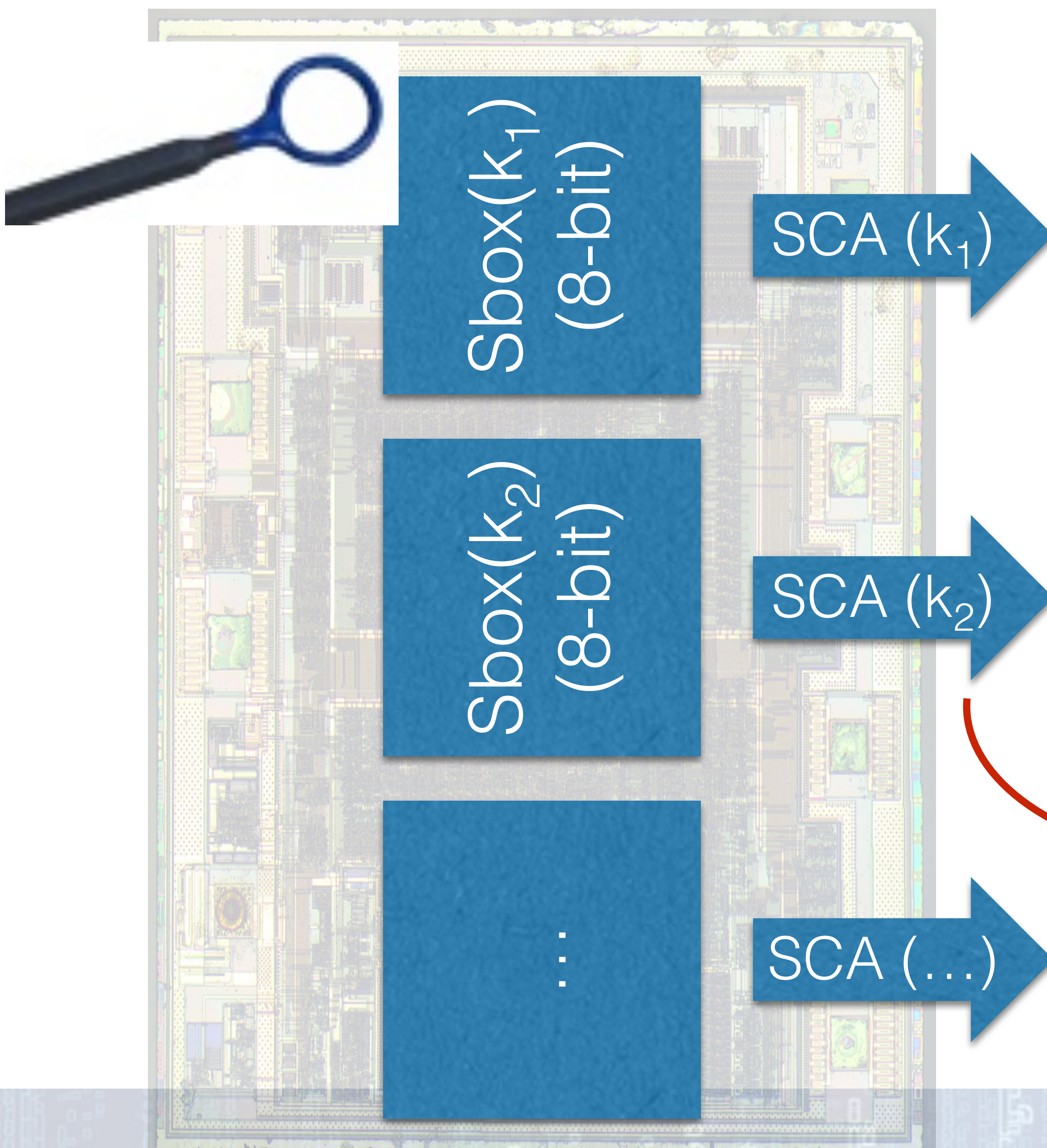
- Used to determine security of a device against side-channel attacks (as well as other attacks…)
- Performed by chip designers as well as specialised evaluation labs (for certification purposes)
- Certifications (e.g. Common Criteria, EMV) typically needed for commercial security-critical products (e.g. banking cards)

Evaluations on single subkeys

- Due to Divide and Conquer strategy, classic evaluation tools apply mostly to single subkeys (bytes, words):
  - **Guessing entropy** (our focus)
  - Success rate
  - Mutual information
  - …

Evaluations on single subkeys

- These tools require lists of probabilities (or scores) for each value of a subkey:



L_1 = [P(k_1=0), P(k_1=1), ..., P(k_1=255)]

L_2 = [P(k_2=0), P(k_2=1), ..., P(k_2=255)]

e.g. Template Attacks

...

Guessing entropy (GM)

- James L. Massey, '94 ('guess work')

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

|S| is the number of values per subkey
p_i are the sorted probabilities after the SCA:

$$p_1 = P(k = v_1) \geq p_2 = P(k = v_2) \geq \ldots \geq p_{|\mathcal{S}|} = P(k = v_{|\mathcal{S}|})$$

- Statistical expectation of position of correct key value in sorted list of probabilities

- Expected amount of work for optimised brute force attack

Empirical guessing entropy (GE) (aka key rank)

- Standaert et al., '06
  - GE = position of correct key (k_{good}) in the sorted list of probabilities:

$$p_1 = P(k = v_1) \geq p_2 = P(k = v_2) \geq \ldots \geq p_{|\mathcal{S}|} = P(k = v_{|\mathcal{S}|})$$

  - e. g. if $k_{good} = v_2$ => GE = 2

Guessing entropy

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

- Statistical expectation of the position of correct key
- Does not require knowledge of k_{good}
  => may be used with unknown key

GE = position of k_{good}

- Actual position of correct key for a set of samples
- Requires knowledge of k_{good}

Our claim: GM can bebetter than GE for security evaluations (e.g. if we have probabilities)

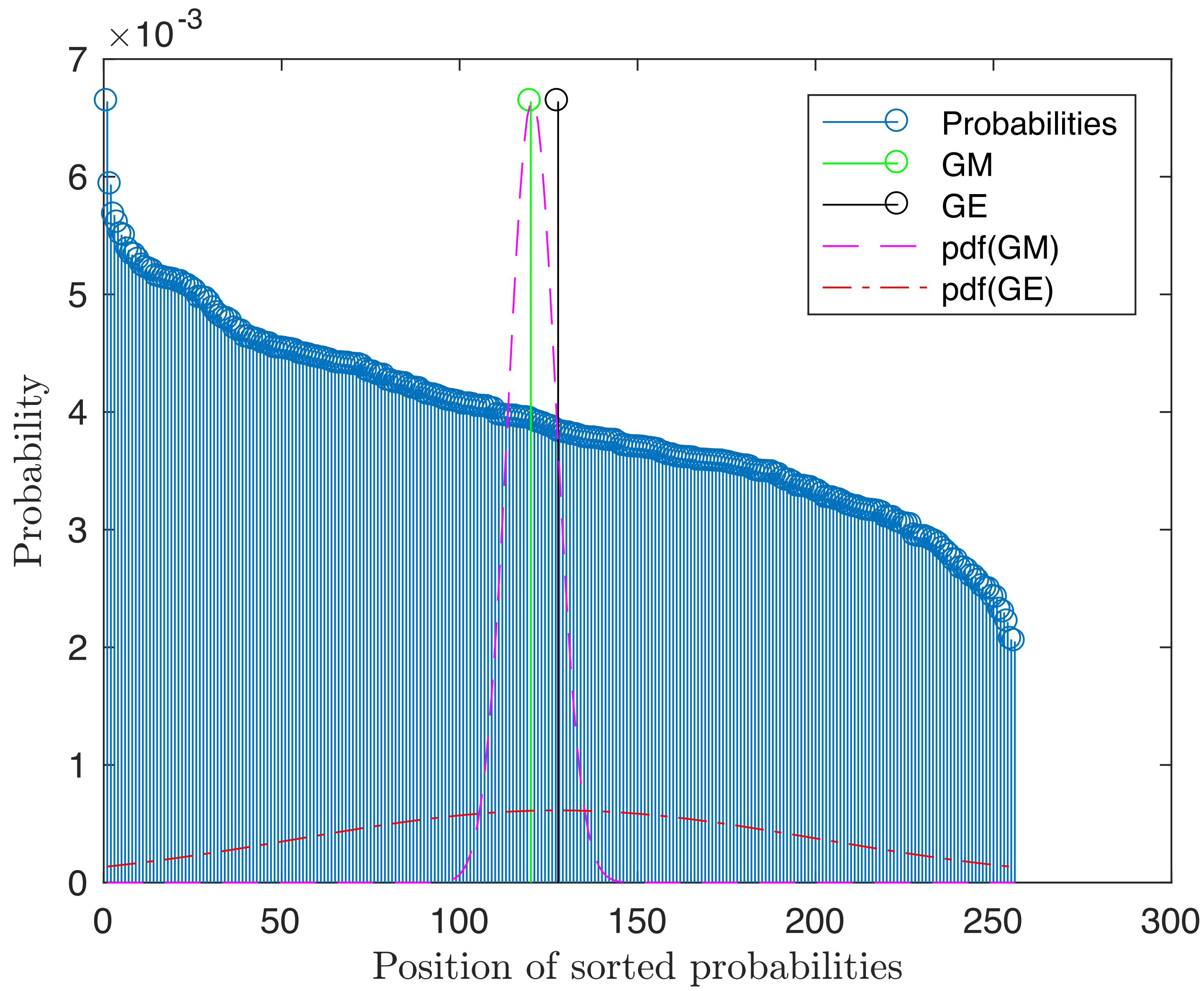
Experimental data sets

- Simulated data set
  - Target is AES S-box lookup
  - Hamming Weight leakage model
  - One sample

- Real data set:
  - Target is AES S-box lookup from AVR XMEGA AES crypto engine
  - Template Attack profiling
  - LDA compression

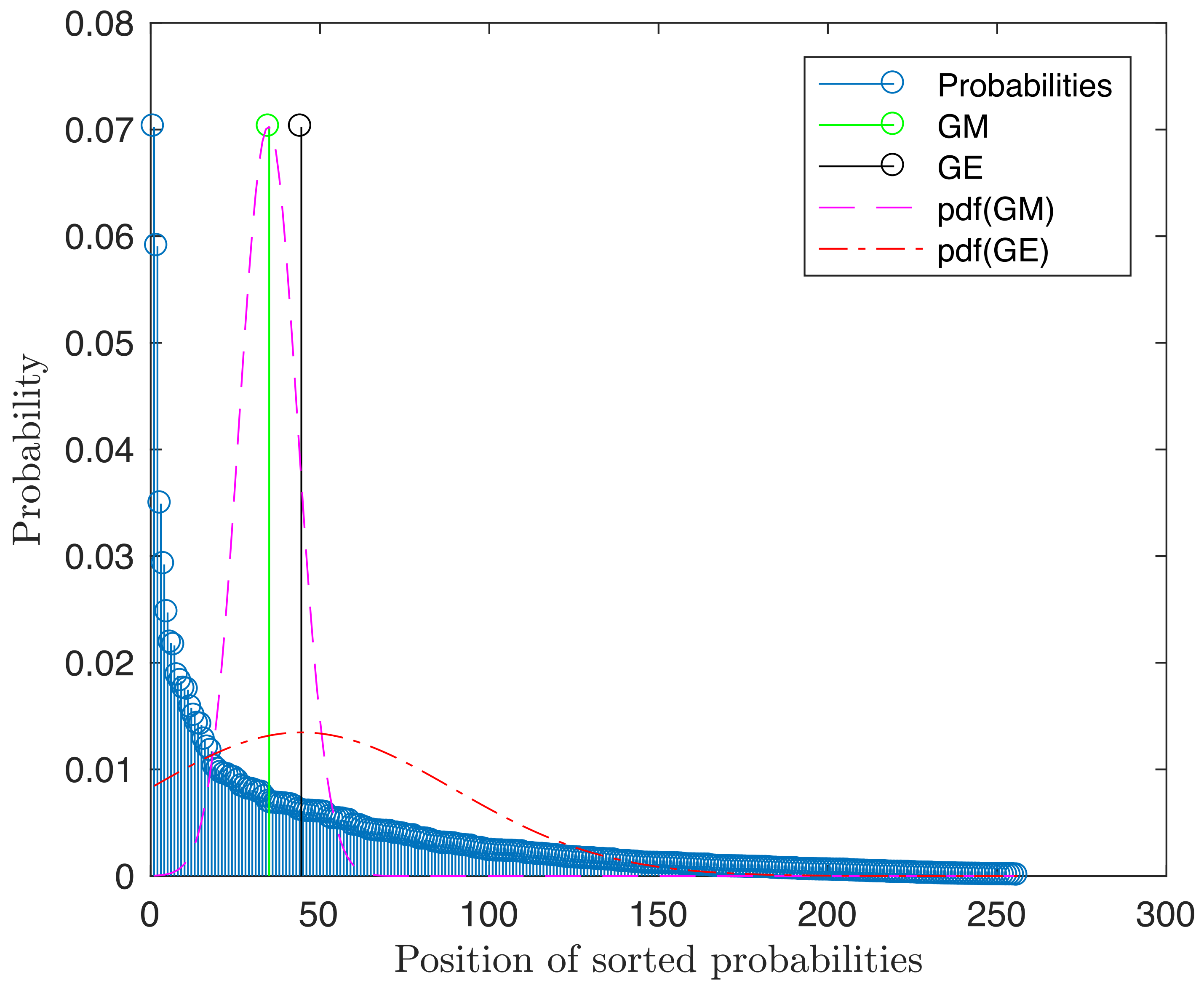
Guessing entropy

- Probabilities for real data with a single attack trace
  Very large standard deviation for GE (100 iterations)



Guessing entropy

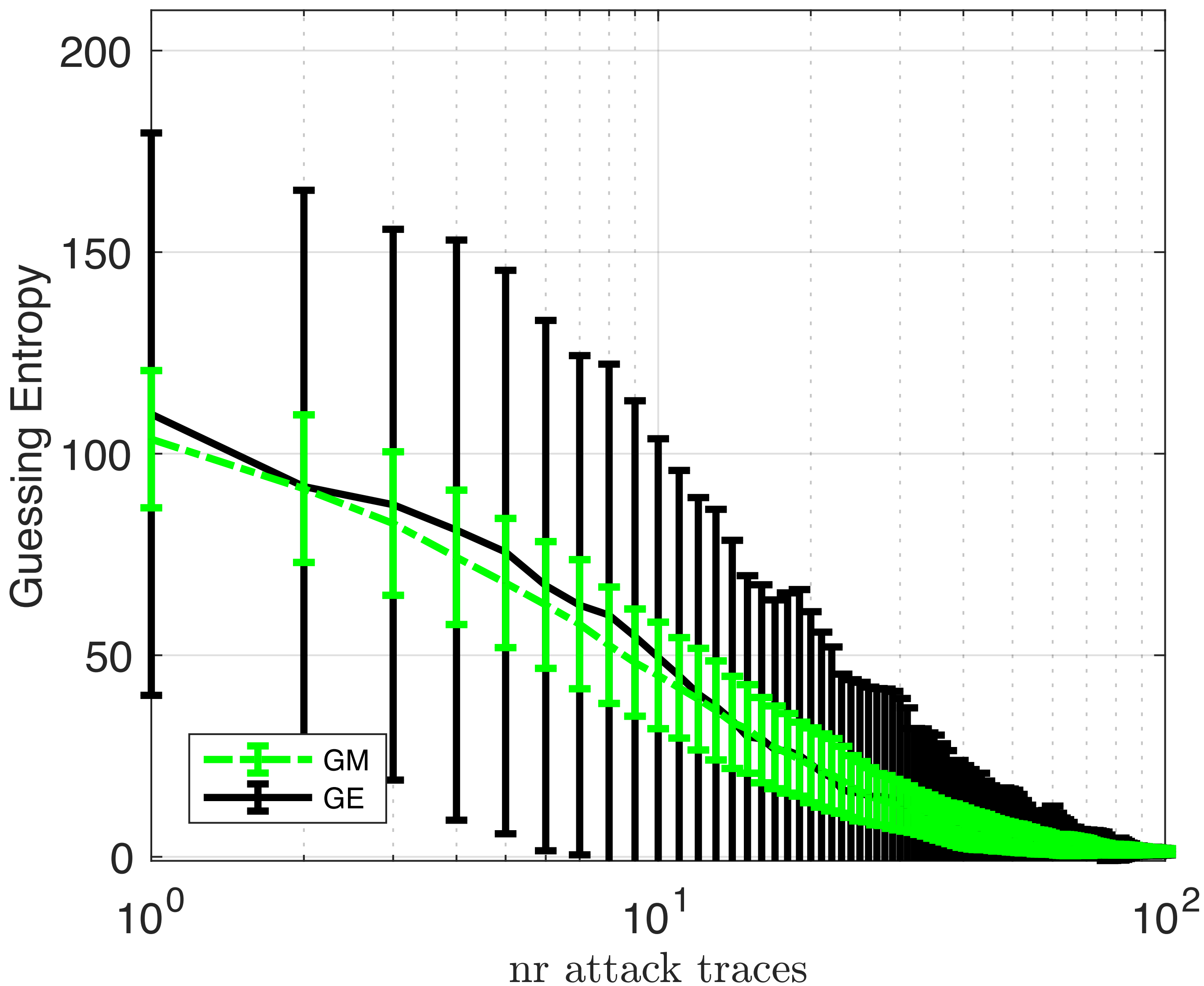
- Probabilities for real data with 100 attack traces
  Again large standard deviation for GE



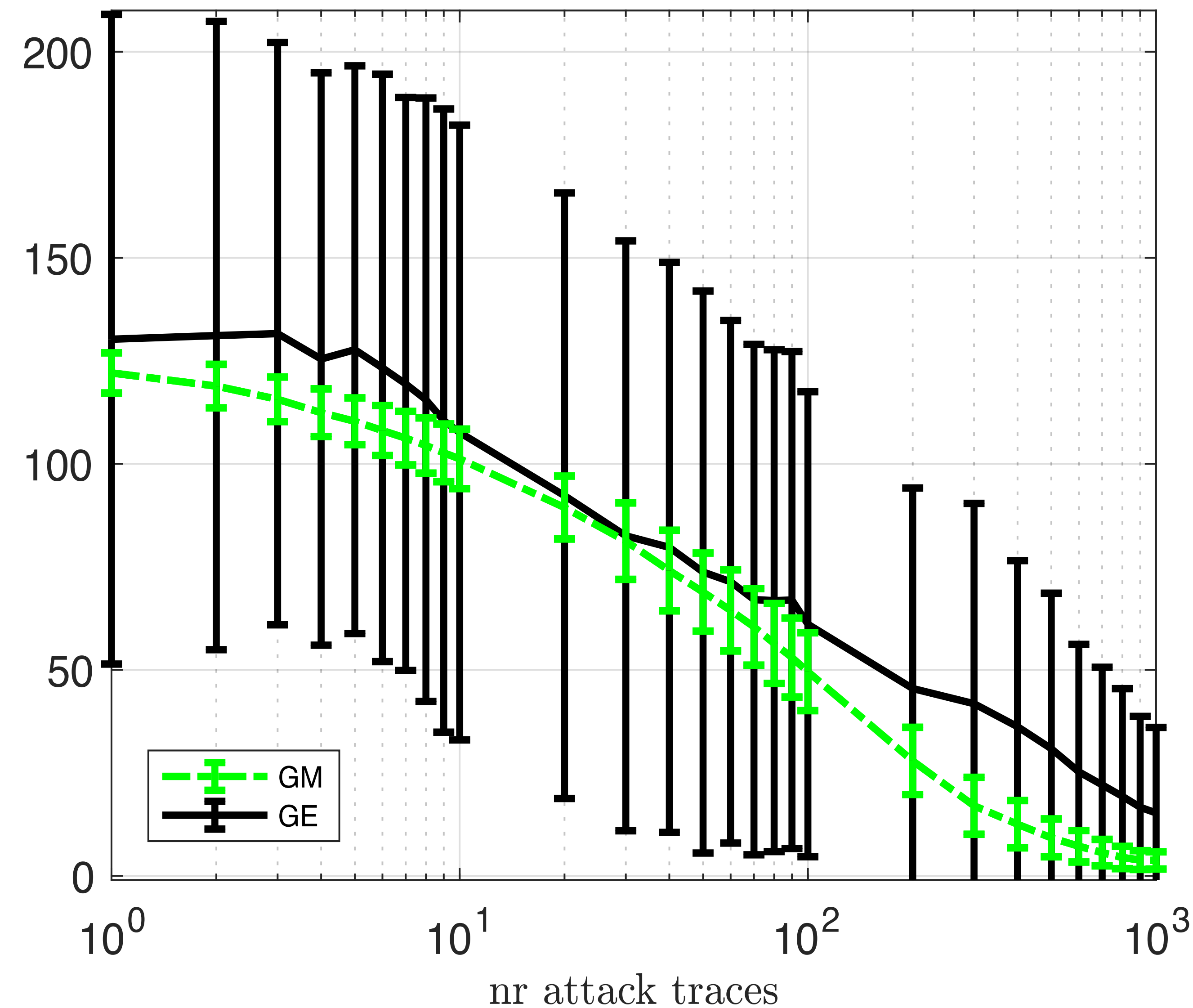
GM, GE on a single key byte

Large standard deviation for GE in both experiments

- Simulated data set



- Real data set

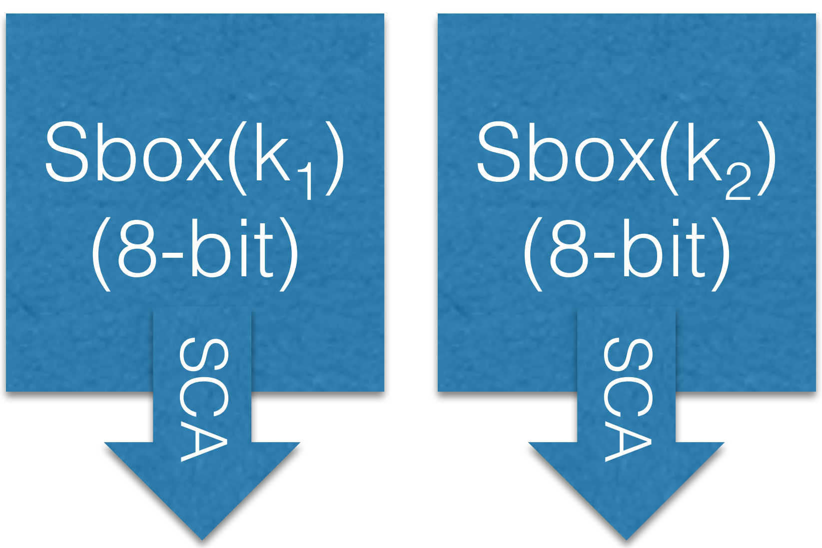


Problems for full-key evaluation:

GM, GE do not scale!

- **n_s** = 2 bytes => $|S|^{ns} = 256^2$ = 65536 probabilities to compute and sort

$$\mathrm{GM}^f = \sum_{i=1}^{|\mathcal{S}|^{n_s}} i \cdot p_i$$



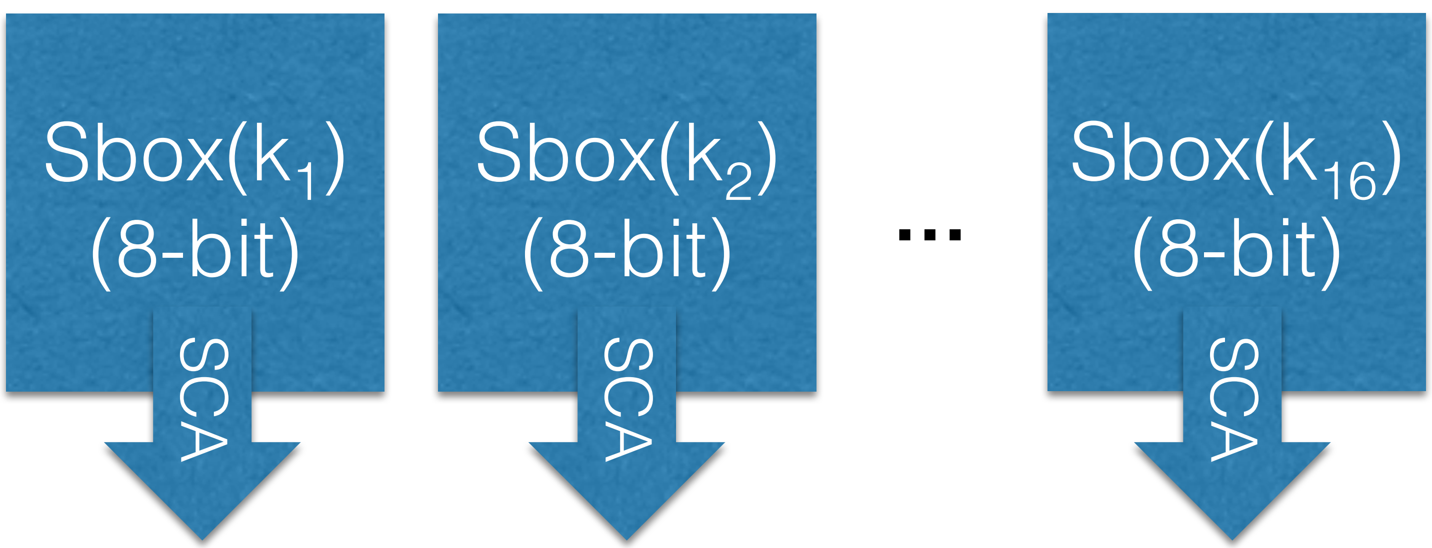
$$\begin{Bmatrix} p_1 \\ p_2 \\ \vdots \\ p_{256} \end{Bmatrix} \times \begin{Bmatrix} p_1 \\ p_2 \\ \vdots \\ p_{256} \end{Bmatrix} = \begin{Bmatrix} p_1 \\ p_2 \\ \vdots \\ p_{65536} \end{Bmatrix}$$

n_s = 2 bytes
=> we can still do it
(takes a few min)

Problems for full-key evaluation: GM, GE do not scale!

- n_s = 16 bytes => $|S|^{ns} = 256^{16}$ = 3.4... x 10^{38} probabilities to compute and sort

$$\mathrm{GM}^f = \sum_{i=1}^{|\mathcal{S}|^{n_s}} i \cdot p_i$$



n_s = 16 bytes

=> we can not do it

$$\left\{ \begin{array}{c} p_1 \\ p_2 \\ \vdots \\ p_{256} \end{array} \right\} \times \left\{ \begin{array}{c} p_1 \\ p_2 \\ \vdots \\ p_{256} \end{array} \right\} \times \cdots \times \left\{ \begin{array}{c} p_1 \\ p_2 \\ \vdots \\ p_{256} \end{array} \right\} = \left\{ \begin{array}{c} p_1 \\ p_2 \\ \vdots \\ p_{2^{128}} \end{array} \right\}$$

Full-key Evaluation tools

- **Key enumeration:** efficient algorithmic combination of lists of probabilities to output the most likely values of the full key (optimised brute force search attack) $f(k_{good}, L_1, L_2, \ldots) \Rightarrow P(k_{full} = v_1) > P(k_{full} = v_2) > \ldots$
- **Rank estimation:** algorithmic estimation (bound) of the key rank (empirical guessing entropy) $f(k_{good}, L_1, L_2, \ldots) \Rightarrow \{\text{lbound(GE)}, \text{ubound(GE)}\}$

Full-key Evaluation tools

- **Limitations:**

  - Existing key enumeration and rank estimation algorithms can only practically work with less than 256-byte (2048-bit) keys (i.e. 256 probability lists)

    (due to computation time and memory consumption)

    => existing tools we cannot evaluate the security of a device against a full-key SCA for keys of 512-byte (4096-bit) and larger
    (e.g. key-loading attack on large RSA keys)

Our main result:
scalable GM bounds for large keys

- Mathematical bounds from Massey's guessing entropy
  - **Fast**: a fraction of a second for a 128-byte key
  - **Tight**: a few bits margin for a 128-byte key
  - **Scalable**: we have computed the bounds for a full-key SCA on 1024-byte (8192-bit) and 8192-byte (65536-bit) keys
  - **With mathematical proofs**

Our main result:
scalable GM bounds for large keys

From math literature, we arrived at the following bounds:

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

(LB_GM) (UB_GM)

- n_s is number of subkeys (key bytes) in full key (e.g. n_s=16 for AES-128)
- |S| is number of possible values per subkey (e.g. 256 for 8-bit implementation of AES).

Our main result:

scalable GM bounds for large keys

From math literature, we arrived at the following result:

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

(LB_GM) (UB_GM)

- Complexity: $\boxed{O(n_\mathrm{s}\cdot|\mathcal{S}|)}$

  => computation increases linearly with number of subkeys

- We can compute distance between LB_GM-UB_GM:

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

Our main result: scalable GM bounds for large keys

From math literature, we arrived at the following result:

$$\frac{1}{1+\ln|\mathcal{S}|^{n_s}} \prod_{i=1}^{n_s} \left[\sum_{k=1}^{|\mathcal{S}|} \sqrt{p_{i,k}}\right]^2 \leq \mathrm{GM}^f \leq \frac{1}{2} \prod_{i=1}^{n_s} \left[\sum_{k=1}^{|\mathcal{S}|} \sqrt{p_{i,k}}\right]^2 + \frac{1}{2}$$

(LB_GM) (UB_GM)

- Complexity: $\boxed{O(n_s \cdot |\mathcal{S}|)}$

=> computation increases **linearly** with number of subkeys

VERY SCALABLE!

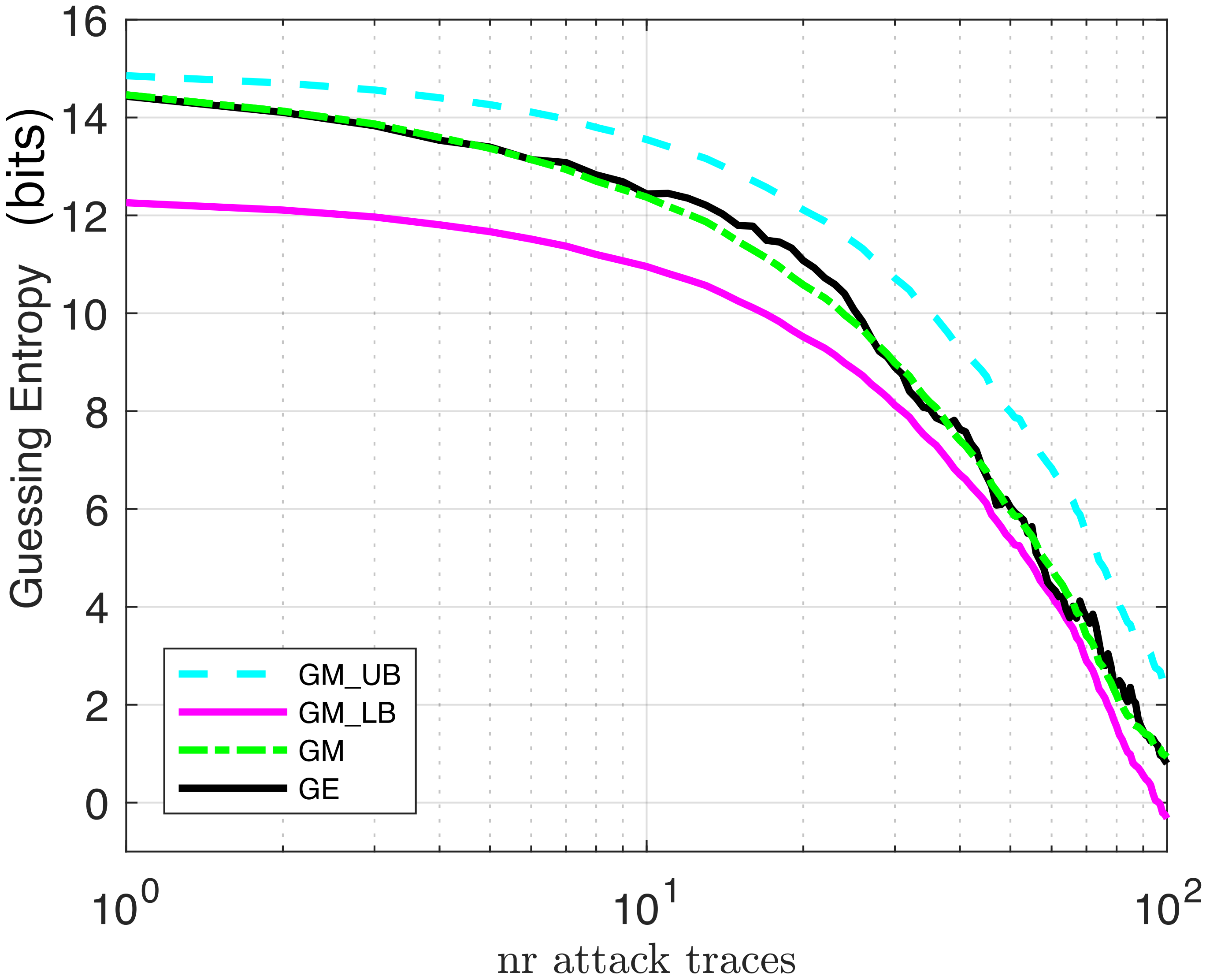
- We can compute distance between LB_GM-UB_GM.compute distance between

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

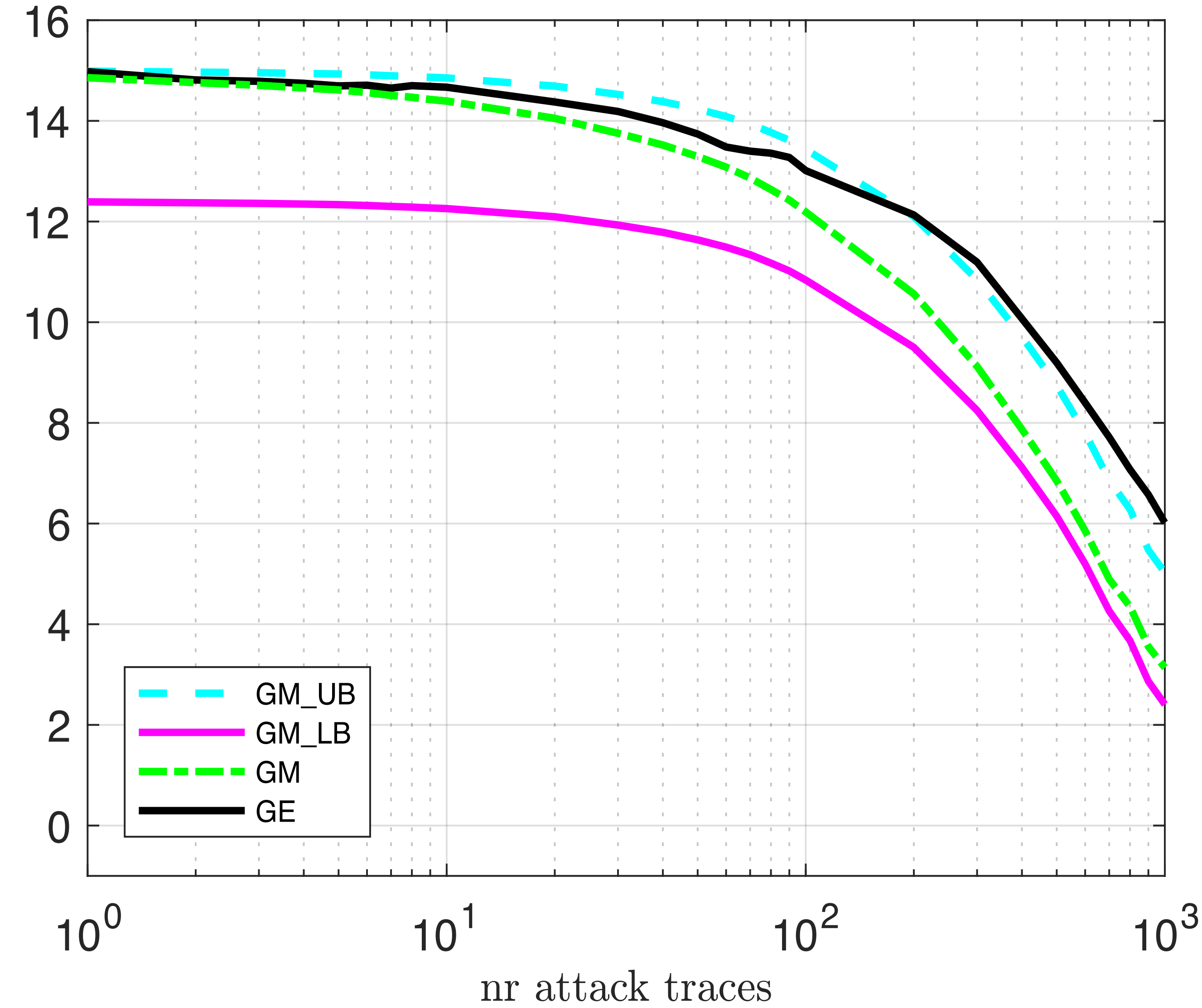
- LB_GM -- UB_GM:

GE, GM and GM bounds on two key bytes

- Simulated data set



- Real data set

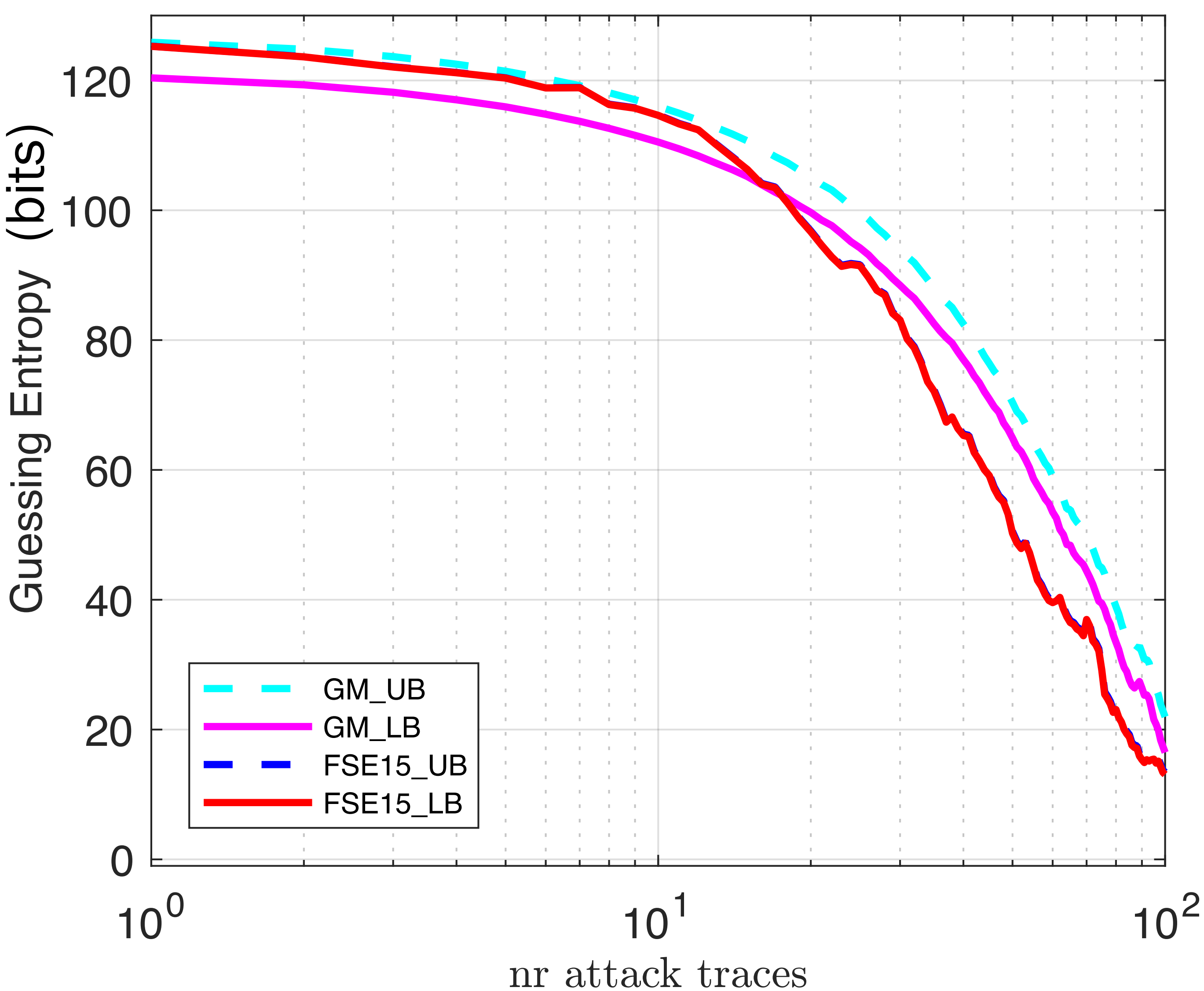


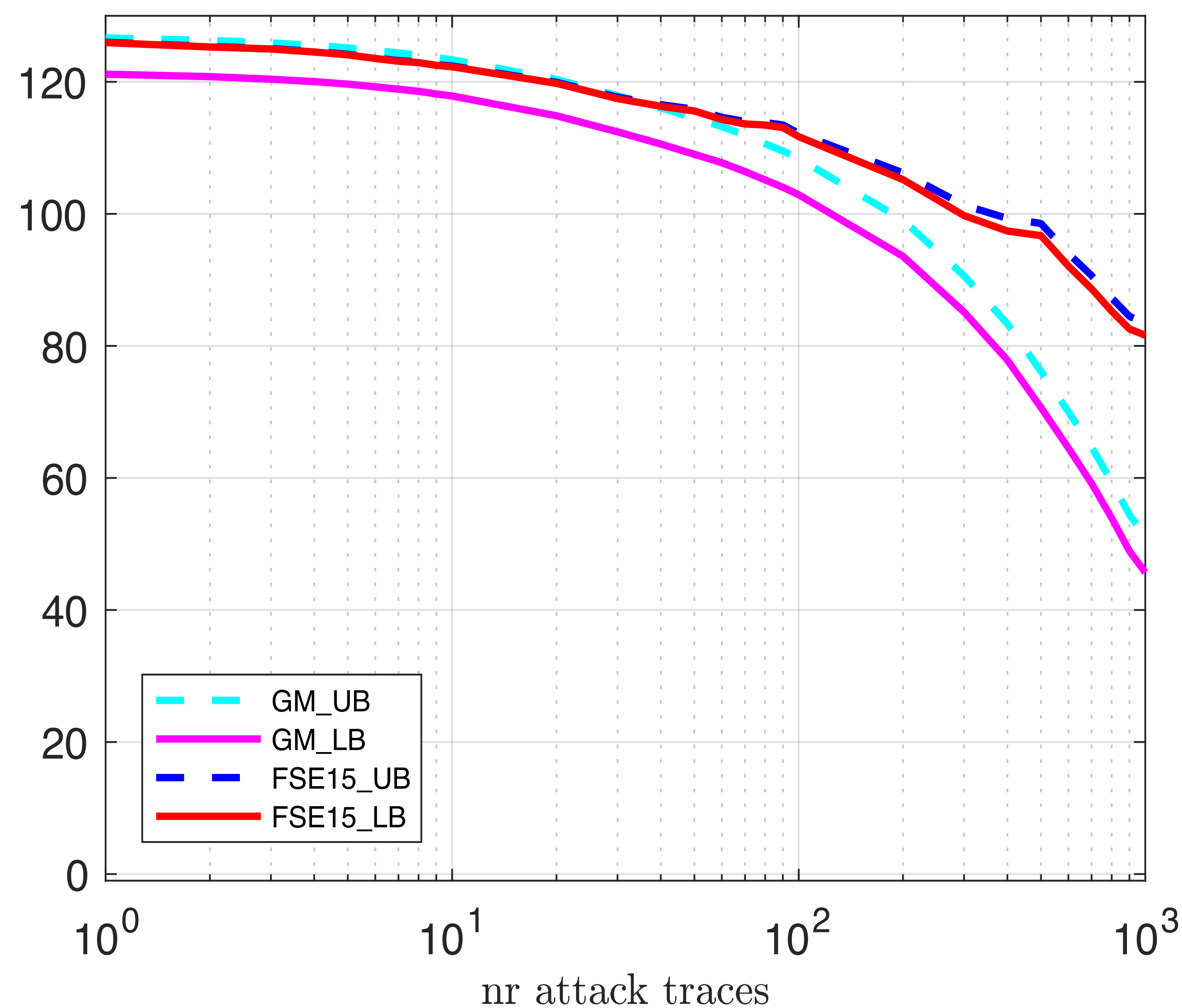
GM bounds vs rank estimation (FSE’15) on 16 key bytes

- Could not compare with GE or GM (not computable for full AES key)
- FSE’15 (Glowacz et al.) : probably the best (tightness + speed) rank estimation algorithm to date
  - Although still not scalable for keys larger than 256 bytes

GM bounds vs rank estimation (FSE'15) on 16 key bytes

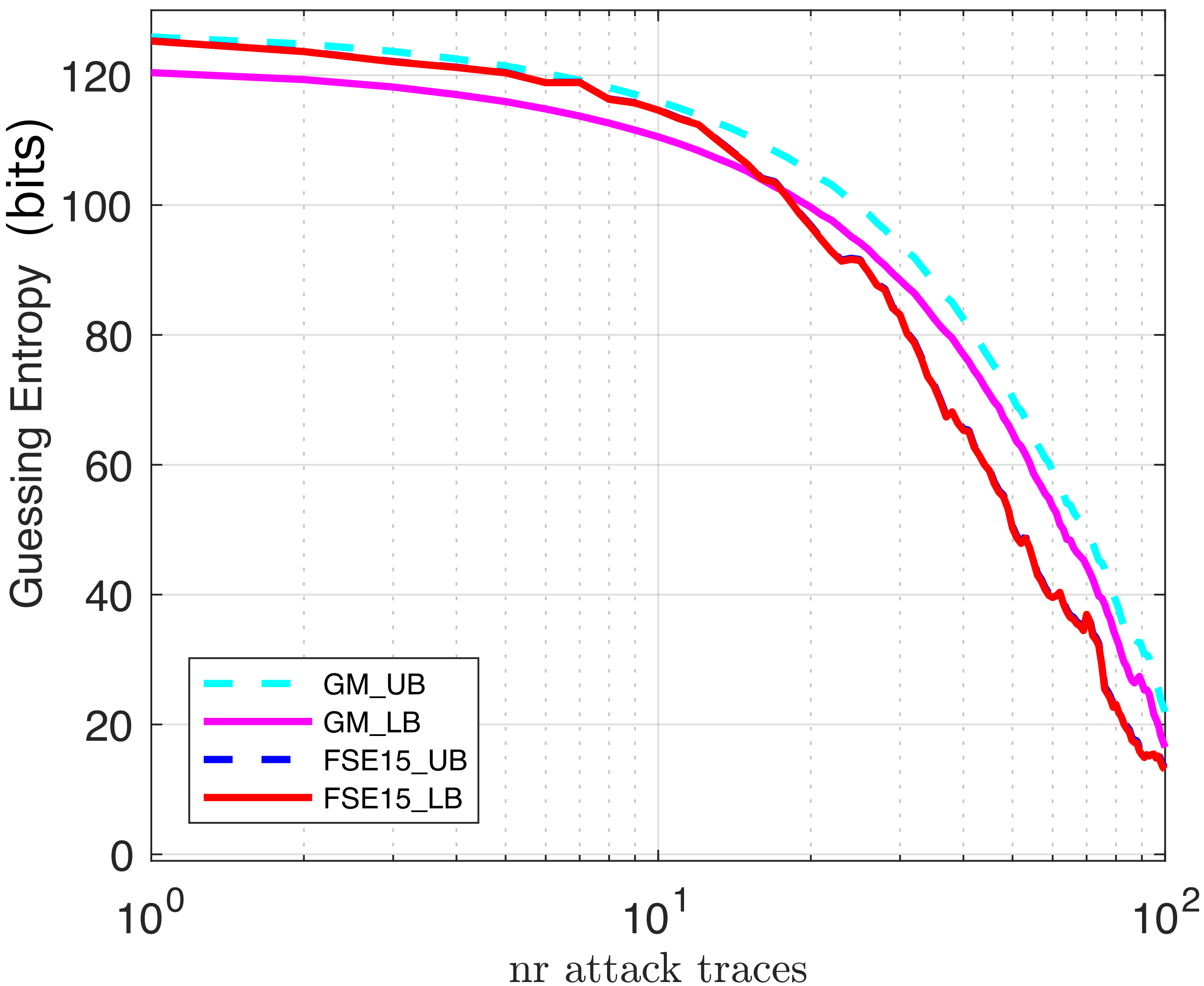
- Simulated data set
- Real data set





GM bounds vs rank estimation (FSE'15) on 16 key bytes

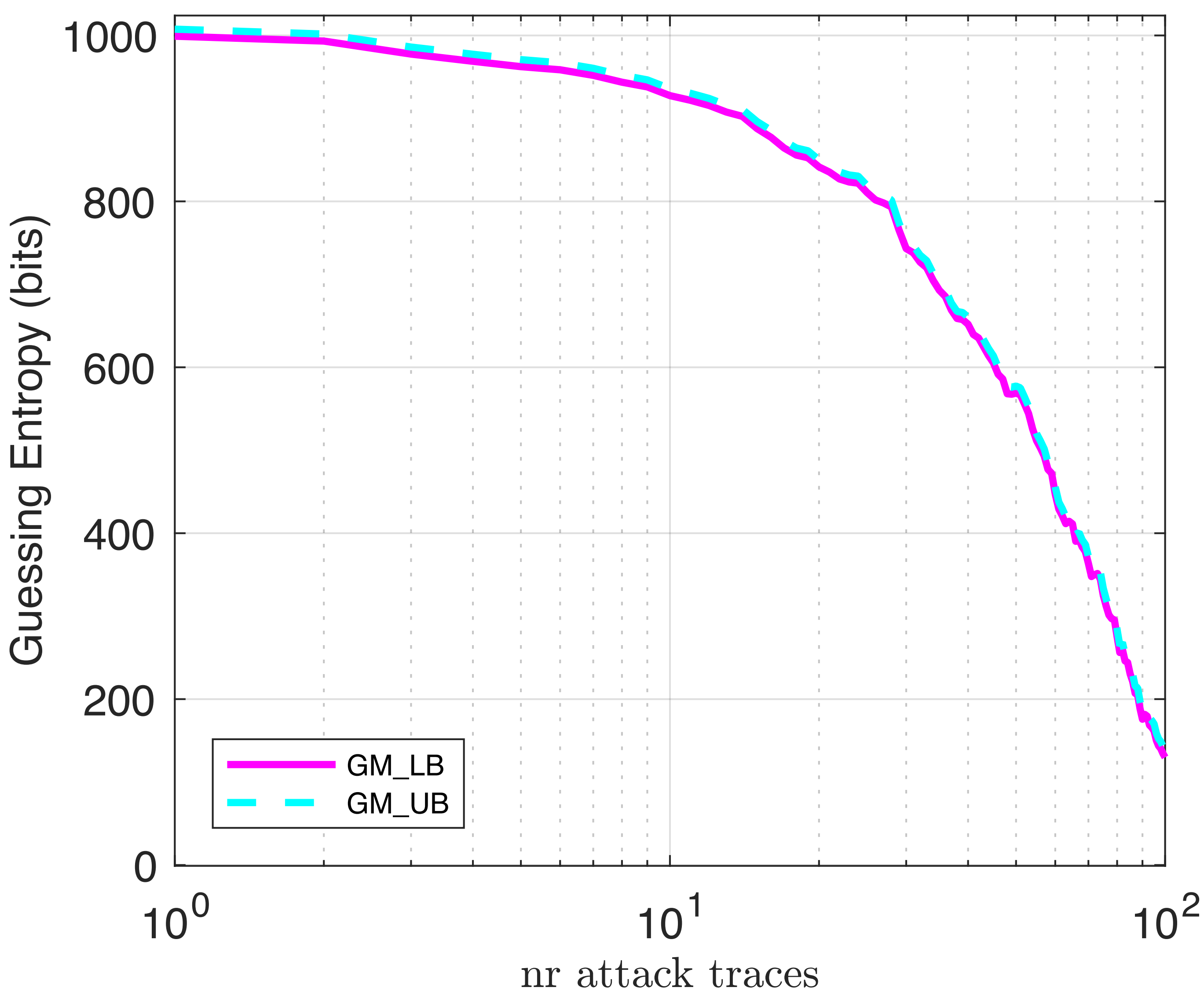
- Simulated data set



- Computation time (16 key bytes)
  - GM bounds: < 10 ms per iteration
  - FSE'15 bounds: ~300 ms per iteration

GM bounds on 128 key bytes

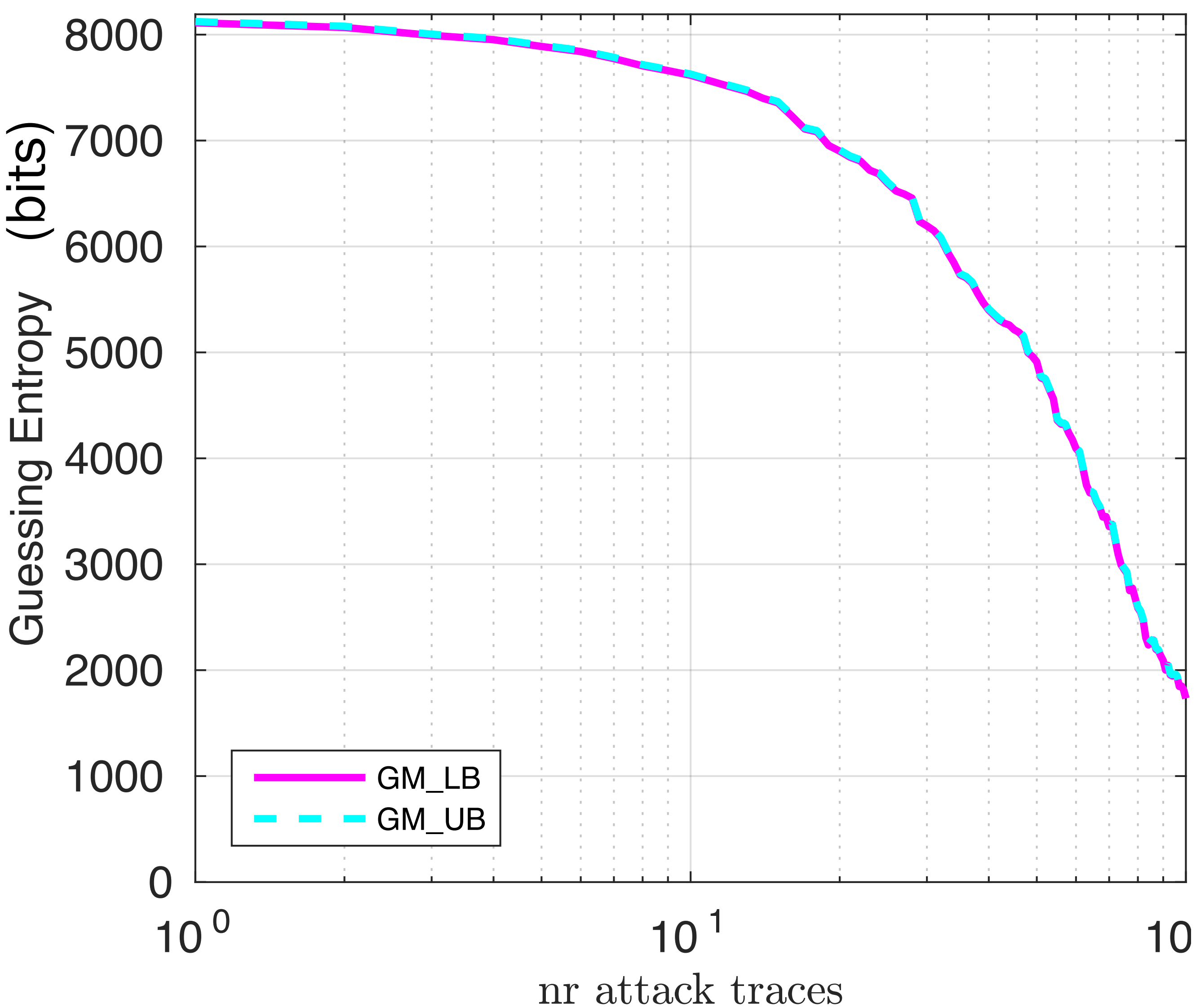
- Simulated data set

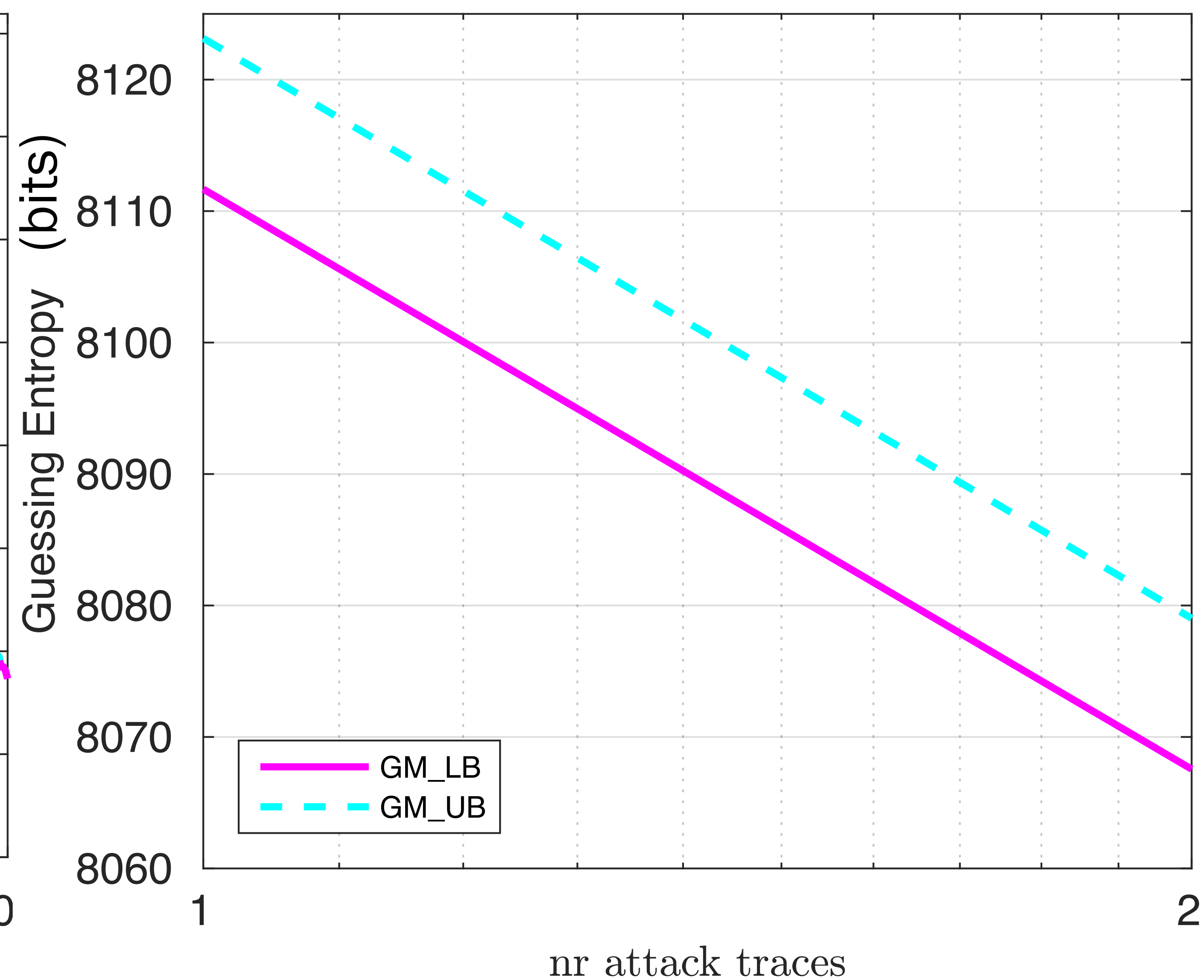


- Constant memory
- Computation time (128 key bytes)
  - 150 ms per iteration
  - FSE'15 requires a few seconds for similar tightness.

Our GM bounds for 1024 bytes (8192-bit key)

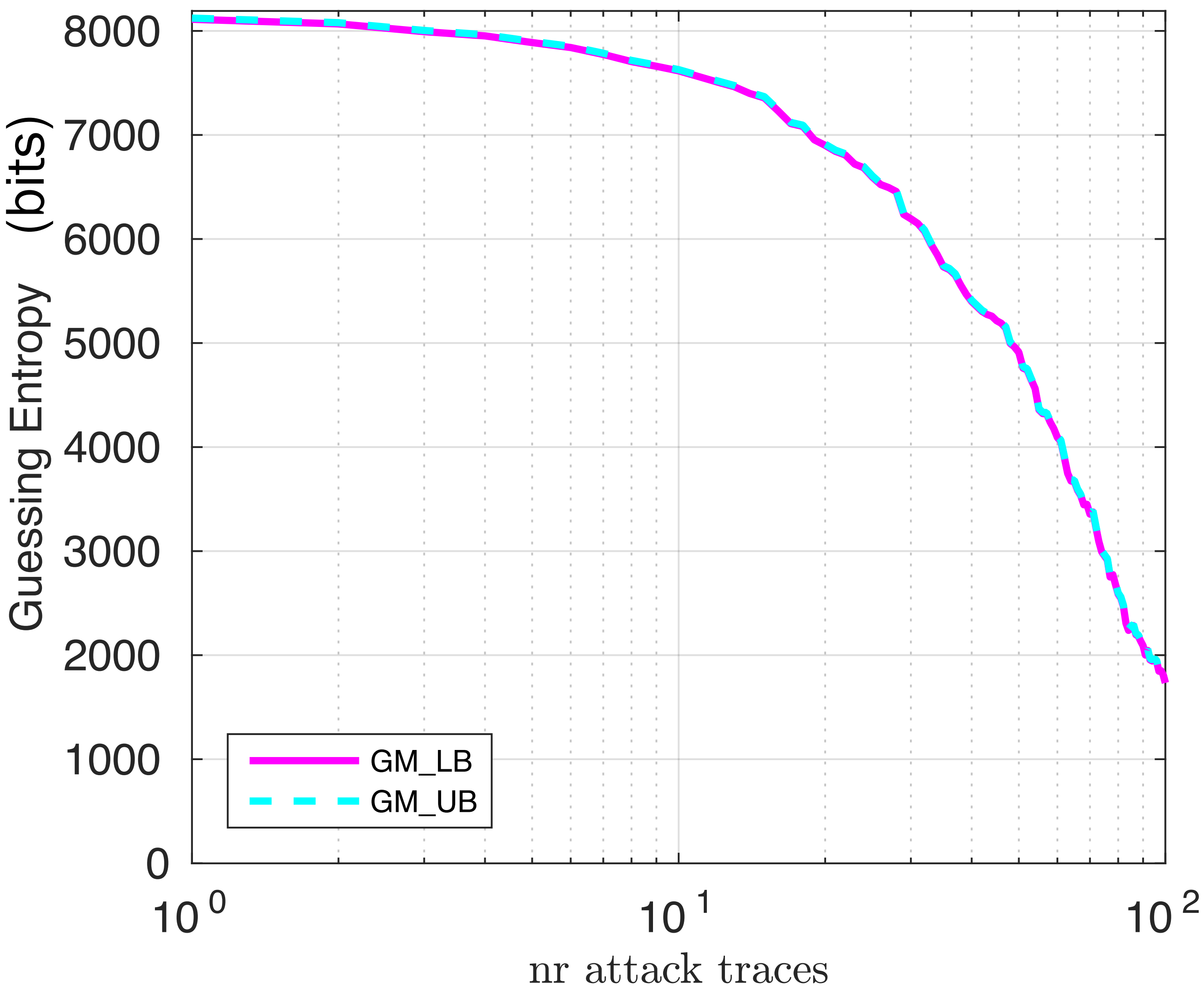
- Based on simulated data set, replicated to obtain 1024 subkeys





Our GM bounds for 1024 bytes (8192-bit key)

- Based on simulated data set, replicated to obtain 1024 subkeys

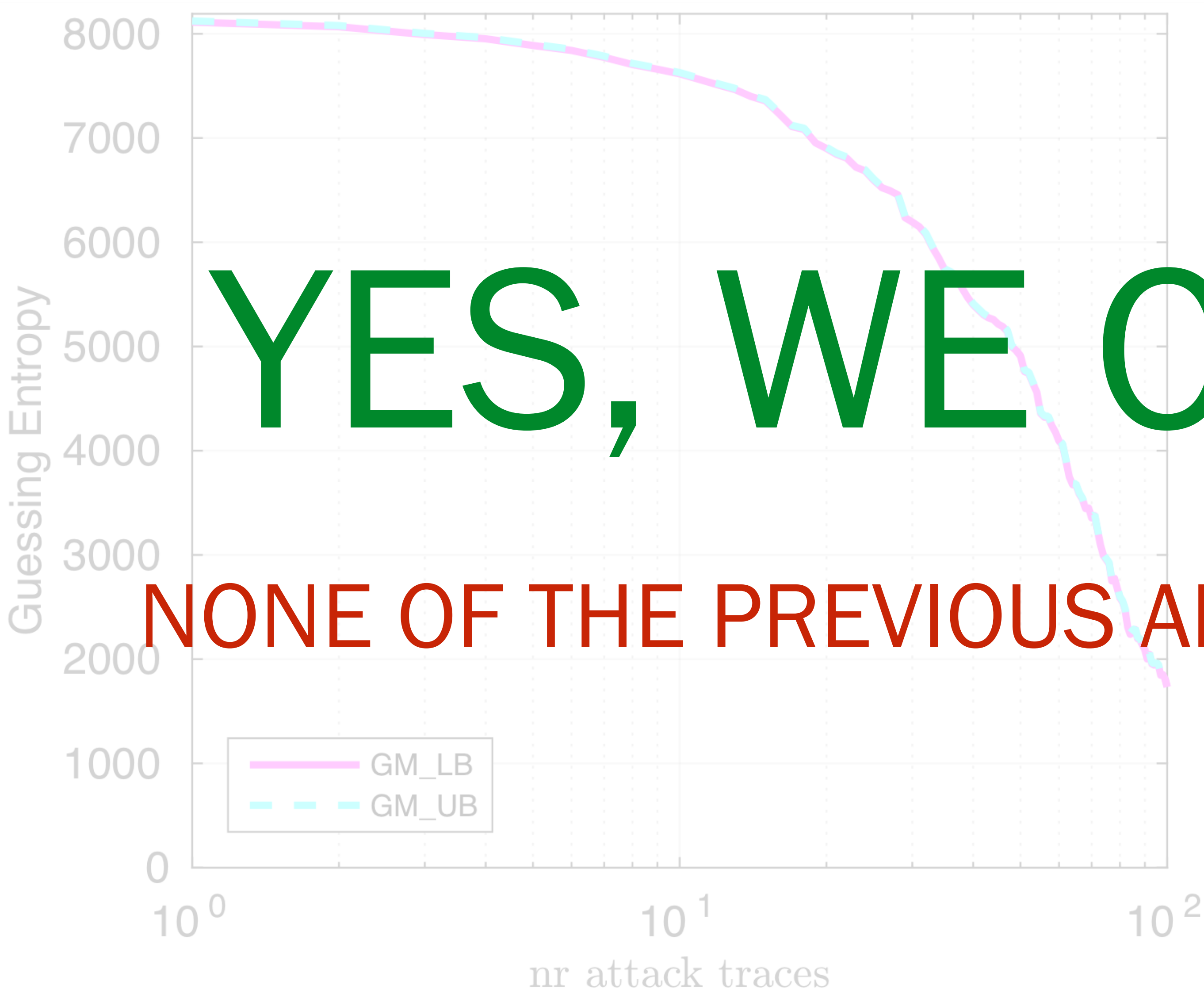


- Constant memory
- Computation time (1024 key bytes)
  - ~70s per iteration:

    - MATLAB VPA (very slow)
    - no optimisations

Our GM bounds for 1024 bytes (8192-bit key)

- Based on simulated data set, replicated to obtain 1024 subkeys



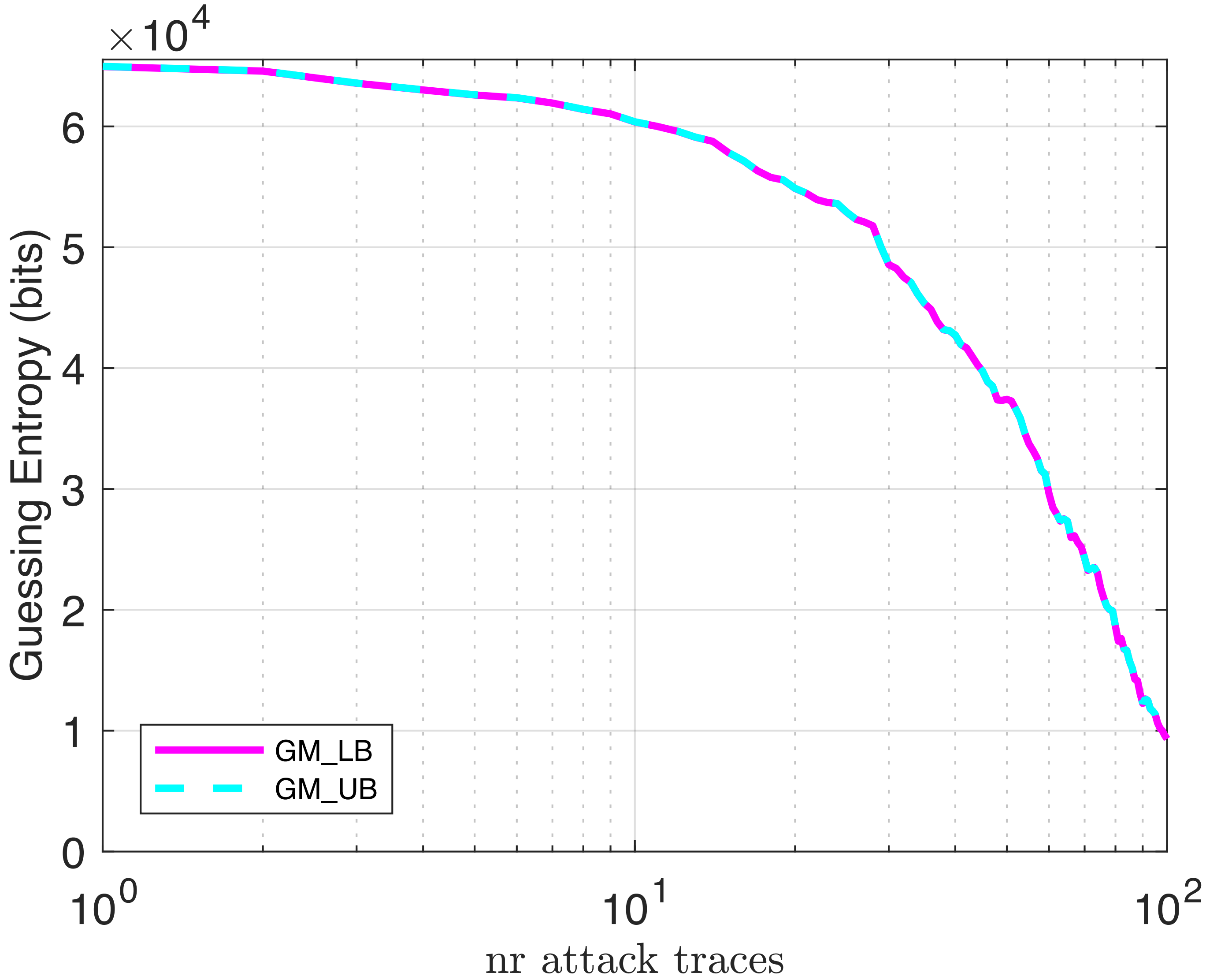
- Constant memory
- Computation time (1024 key bytes)
  - ~70s per iteration:
    - MATLAB VPA
    - no optimisations

YES, WE CAN DO IT!

NONE OF THE PREVIOUS ALGORITHMS COULD DO IT!

Our GM bounds for 1024 bytes (8192-bit key)

- We can even go further: 8192-byte (65536-bit) key



- Constant memory
- Computation time (8192 key bytes)
  - ~1000s per iteration:

    - MATLAB VPA (very slow)
    - no optimisations

Conclusions

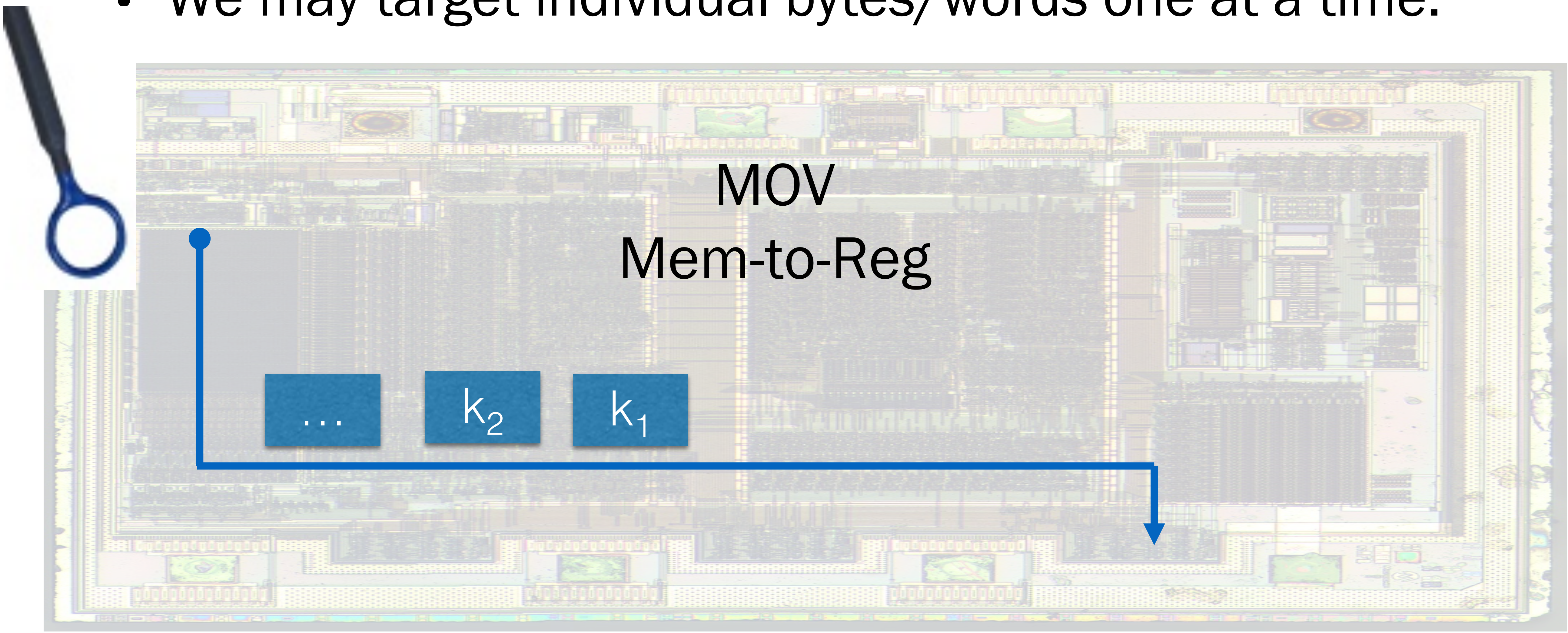
- GM can be a valuable evaluation tool
- Our GM bounds provide the fastest and most scalable full-key SCA evaluation tool to date
- We can evaluate very large keys
  - Results shown for 1024-byte (8192-bit) and 8192-byte (65536-bit) key
- Read the paper for more details and results
- Code available: https://gitlab.cs.pub.ro/marios.choudary/gmbounds

If you like this, please sponsor us ☺

marios.choudary@cs.pub.ro
pgpopescu@yahoo.com

SCA on key-loading operations

- We may target individual bytes/words one at a time:



GM bounds vs rank estimation methods

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: $O({n_s}^2 \|\mathcal{S}\| \log \|\mathcal{S}\|)$ (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: $O(n_s \cdot (\|\mathcal{S}\| \log \|\mathcal{S}\|))$.	Weak lower bound. Very weak upper bound.
$\mathbf{LB_{GM}}$ and $\mathbf{UB_{GM}}$	Guaranteed bounds for GM. Fastest method to date. Scales to arbitrarily large n_s: $O(n_s \cdot \|\mathcal{S}\|)$. Tight bounds (5 bits for 128-bit key). Constant (negligible) memory.	