riscure

Efficient practical key recovery for side channel attacks

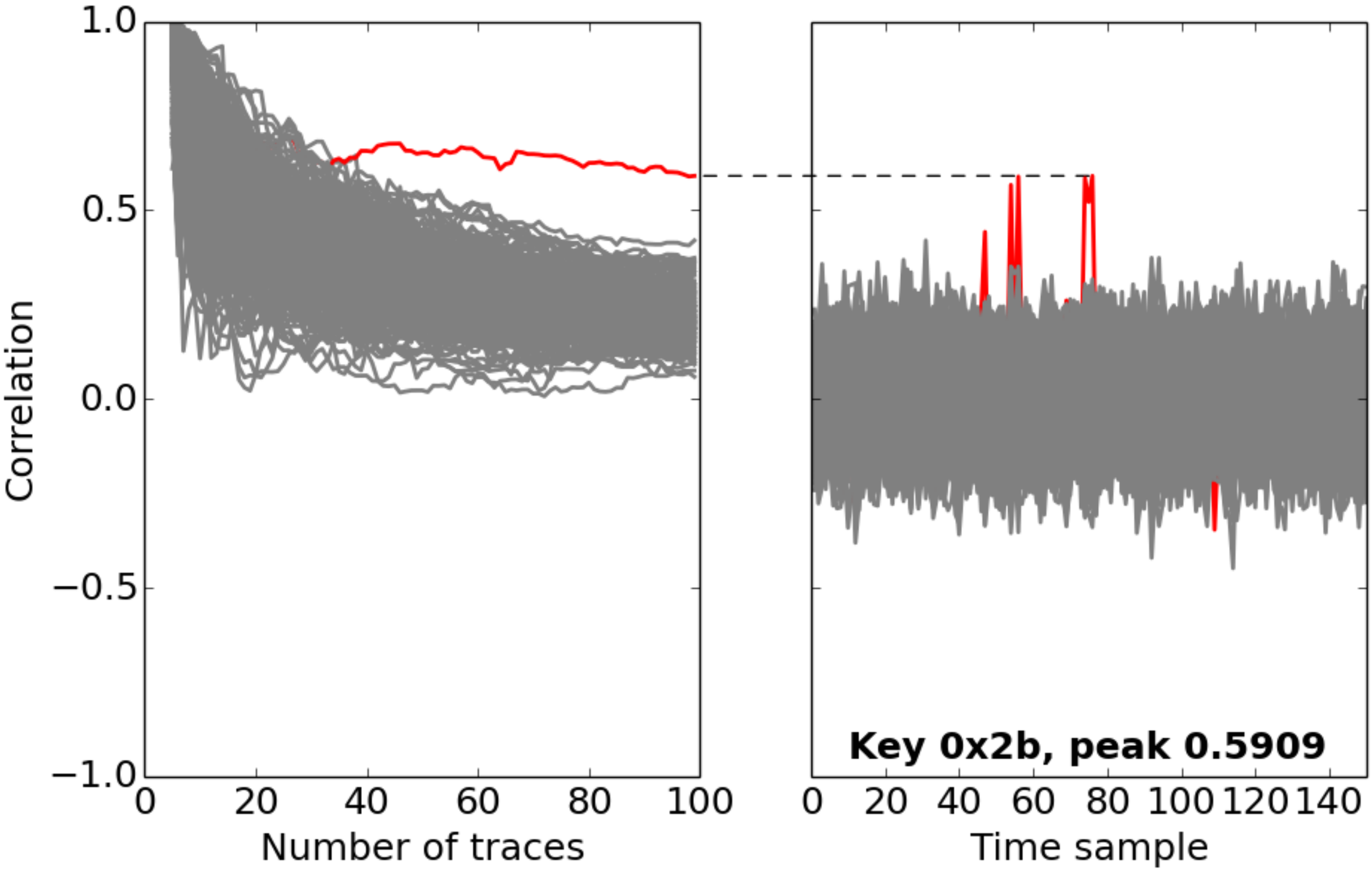
Ilya Kizhvatov

Joint work with Andrey Bogdanov and Kamran Manzoor (DTU), and Marc Witteman (Riscure)

MCrypt, Les Deux Alpes, 15 August 2014

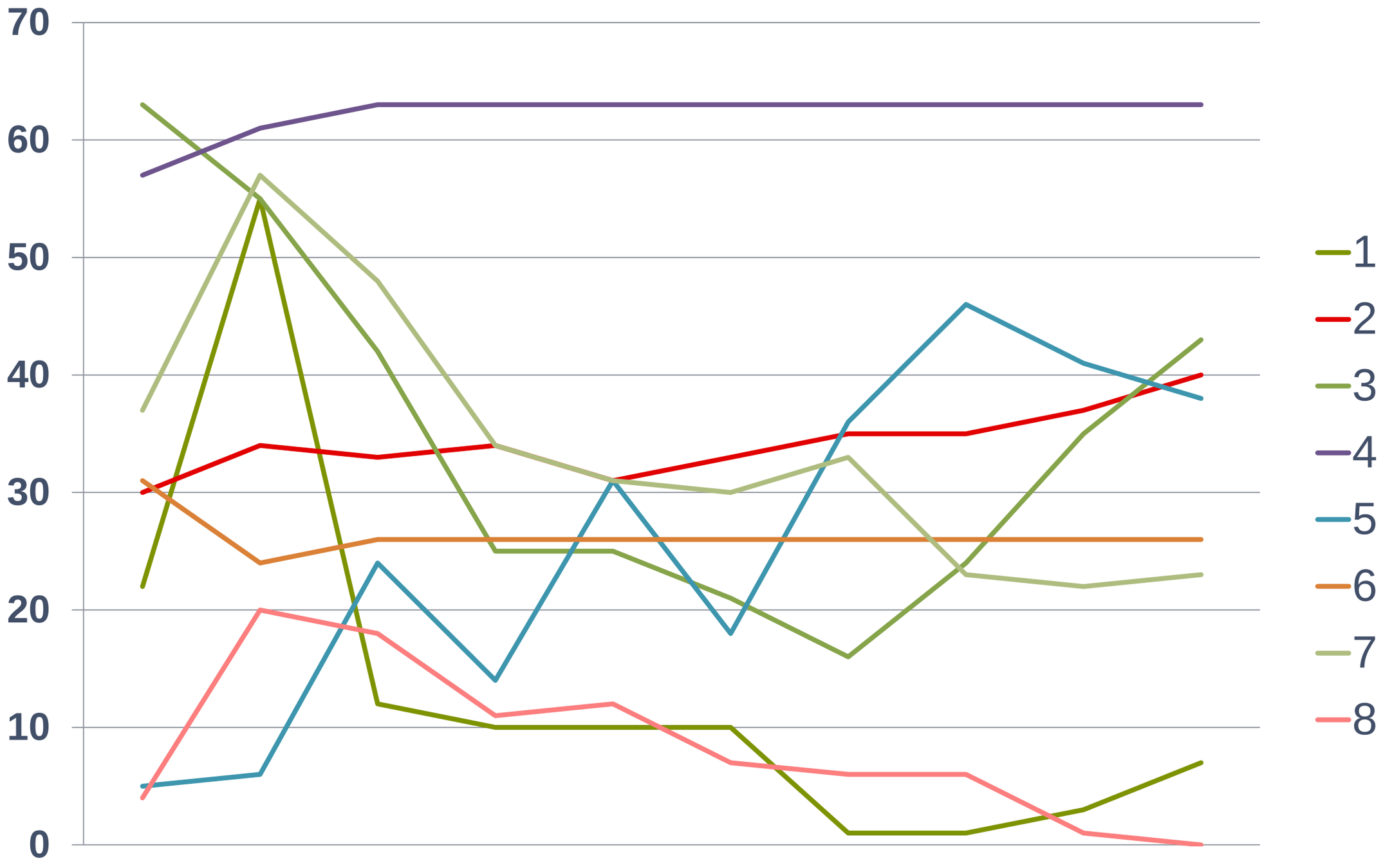
DPA recap

Distinguishing key chunks



DPA recap

Evolution of the rank of correct key chunk candidates



Problem statement

Conquer phase:

- The sub-keys are combined to generate the full-key.
- *Ideally* full-key = the most probable sub-key candidate from each list

k_0	0.0065*	k_0	0.0071	…….	k_0	0.0070
k_1	0.0063	k_1	0.0068		k_1	0.0067
⋮	⋮	⋮	⋮		⋮	⋮
k_n	0.0010	k_n	0.0011		k_n	0.0012

*Probability values for illustration

Problem statement

- How to choose the full key?

k_0	0.0065*
k_1	0.0063
⋮	⋮
k_n	0.0010

k_0	0.0071
k_1	0.0068
⋮	⋮
k_n	0.0011

… … …

k_0	0.0070
k_1	0.0067
⋮	⋮
k_n	0.0012

*Probability values for illustration

Problem statement

- How to choose the full key?

k_0	0.0065*
k_1	0.0063
⋮	⋮
k_n	0.0010

k_0	0.0071
k_1	0.0068
⋮	⋮
k_n	0.0011

… … …

k_0	0.0070
k_1	0.0067
⋮	⋮
k_n	0.0012

*Probability values for illustration

Problem statement

- How to choose the full key?

k_0	0.0065*
k_1	0.0063
⋮	⋮
k_n	0.0010

k_0	0.0071
k_1	0.0068
⋮	⋮
k_n	0.0011

… … …

k_0	0.0070
k_1	0.0067
⋮	⋮
k_n	0.0012

*Probability values for illustration

Problem statement

- How to choose the full key?

k_0	0.0065*
k_1	0.0063
⋮	⋮
k_n	0.0010

k_0	0.0071
k_1	0.0068
⋮	⋮
k_n	0.0011

… … …

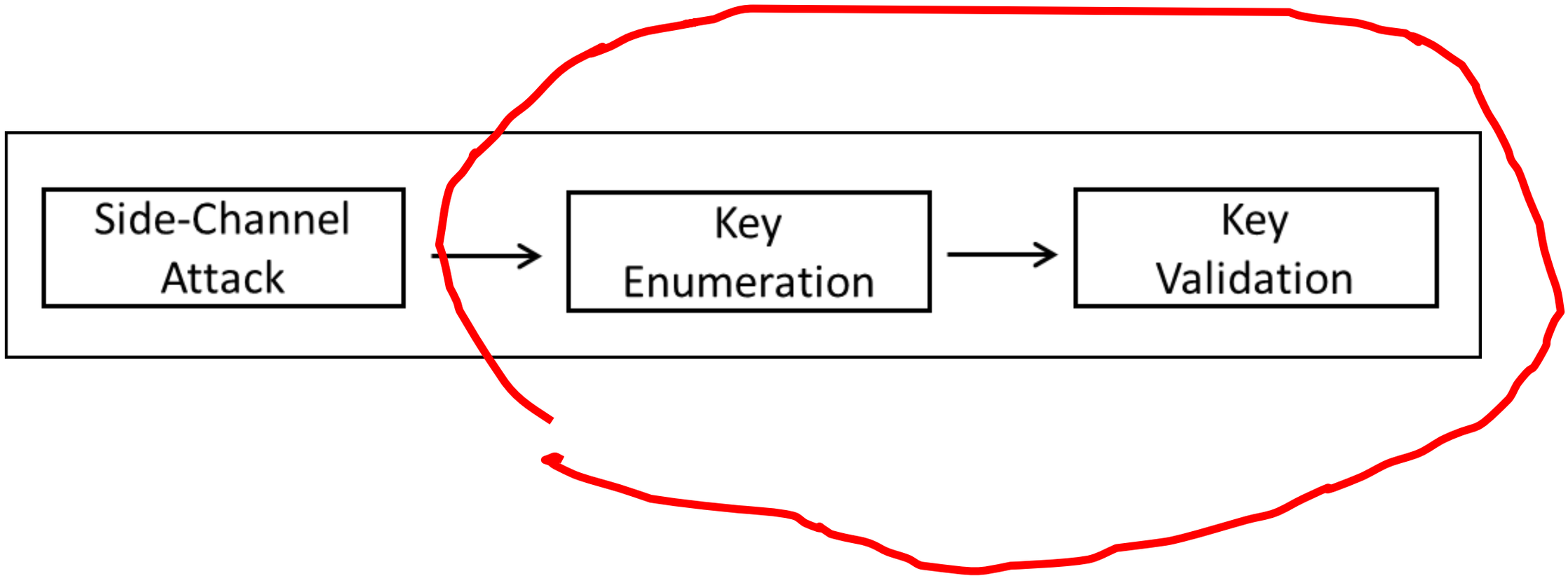
k_0	0.0070
k_1	0.0067
⋮	⋮
k_n	0.0012

- Solution:
  - Key enumeration
  - Concurrent validation of full keys

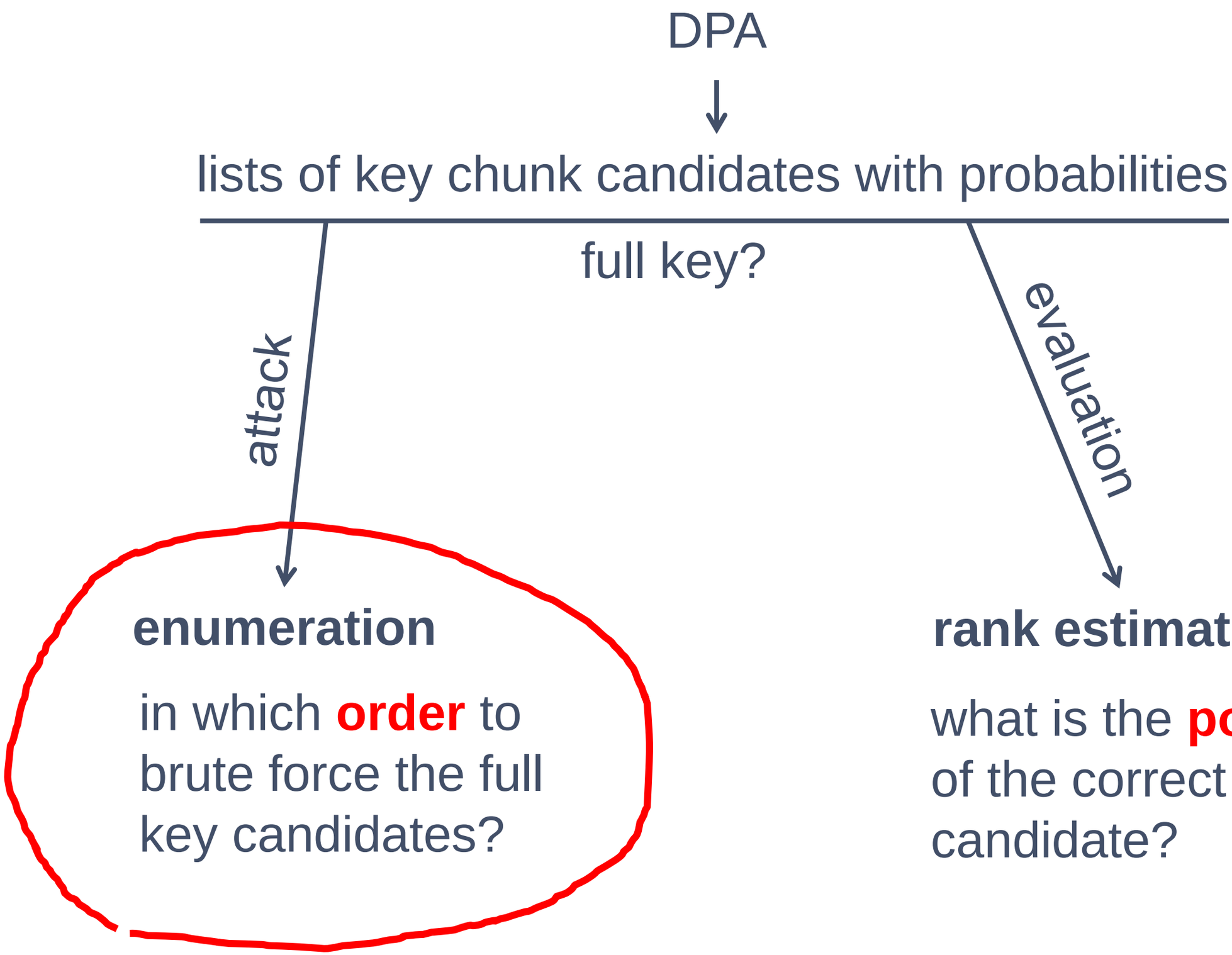
*Probability values for illustration

Problem statement

Find the key after DPA as fast as possible. On a desktop.



Context

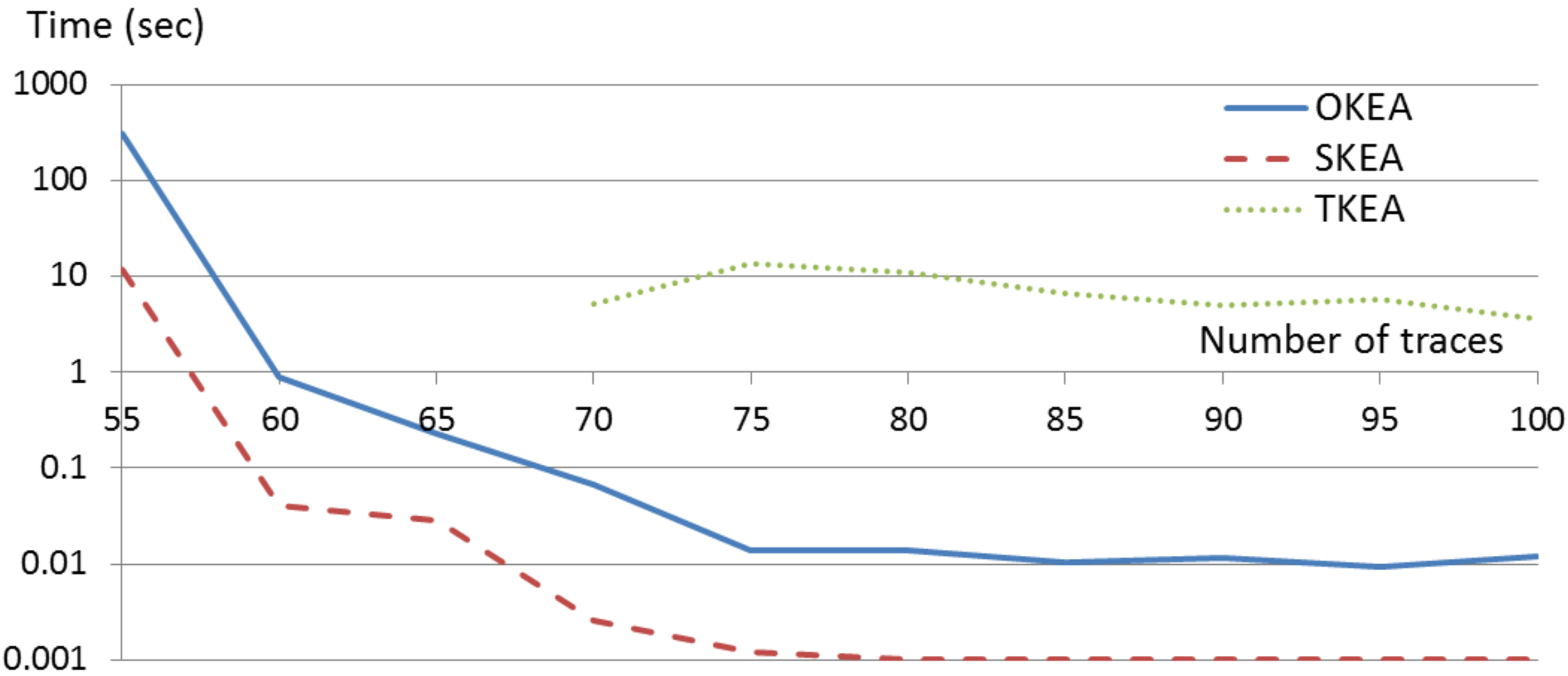

DPA
lists of key chunk candidates with probabilities
full key?
attack
evaluation
enumeration
in which order to brute force the full key candidates?
rank estimation
what is the position of the correct full key candidate?

In this work so far…

- Benchmarked 3 enumeration algorithms:
  - Trivial Key Enumeration Algorithm (TKEA)
  - Optimal Key Enumeration Algorithm (OKEA) [SAC 2012]
  - **Score based Key Enumeration Algorithm (SKEA) [Marc Witteman] – new solution**

- Combined with key validation to get the full solution

Time to find the key



Trivial Key Enumeration Algorithm (TKEA)

k_0	0.0065*
k_1	0.0063
⋮	⋮
k_n	0.0010

k_0	0.0071
k_1	0.0068
⋮	⋮
k_n	0.0011

… … …

k_0	0.0070
k_1	0.0067
⋮	⋮
k_n	0.0012

*Probability values for illustration

Trivial Key Enumeration Algorithm (TKEA)

k_0	0.0065*
k_1	0.0063
⋮	⋮
k_n	0.0010

k_0	0.0071
k_1	0.0068
⋮	⋮
k_n	0.0011

… … …

k_0	0.0070
k_1	0.0067
⋮	⋮
k_n	0.0012

*Probability values for illustration

Trivial Key Enumeration Algorithm (TKEA)

k_0	0.0065*	k_0	0.0071	... ...	k_0	0.0070
k_1	0.0063	k_1	0.0068		k_1	0.0067
⋮	⋮	⋮	⋮		⋮	⋮
k_n	0.0010	k_n	0.0011		k_n	0.0012

*Probability values for illustration

Trivial Key Enumeration Algorithm (TKEA)

k_0	0.0065*		k_0	0.0071	… … …		k_0	0.0070
k_1	0.0063		k_1	0.0068			k_1	0.0067
⋮	⋮		⋮	⋮			⋮	⋮
k_n	0.0010		k_n	0.0011			k_n	0.0012

*Probability values for illustration

Trivial Key Enumeration Algorithm (TKEA)

k_0	0.0065*
k_1	0.0063
⋮	⋮
k_n	0.0010

k_0	0.0071
k_1	0.0068
⋮	⋮
k_n	0.0011

… … …

k_0	0.0070
k_1	0.0067
⋮	⋮
k_n	0.0012

*Probability values for illustration

Trivial Key Enumeration Algorithm (TKEA)

k_0	0.0065*		k_0	0.0071	… … …		k_0	0.0070
k_1	0.0063		k_1	0.0068			k_1	0.0067
⋮	⋮		⋮	⋮			⋮	⋮
k_n	0.0010		k_n	0.0011			k_n	0.0012

*Probability values for illustration

Trivial Key Enumeration Algorithm (TKEA)

- Analysis:
  - Does not consume significant memory
  - Does not follow optimality at all
  - Has a very high throughput
  - Does not generate key within a feasible amount of time
- Above claims are based on our experimental results
- Not of our interest… 

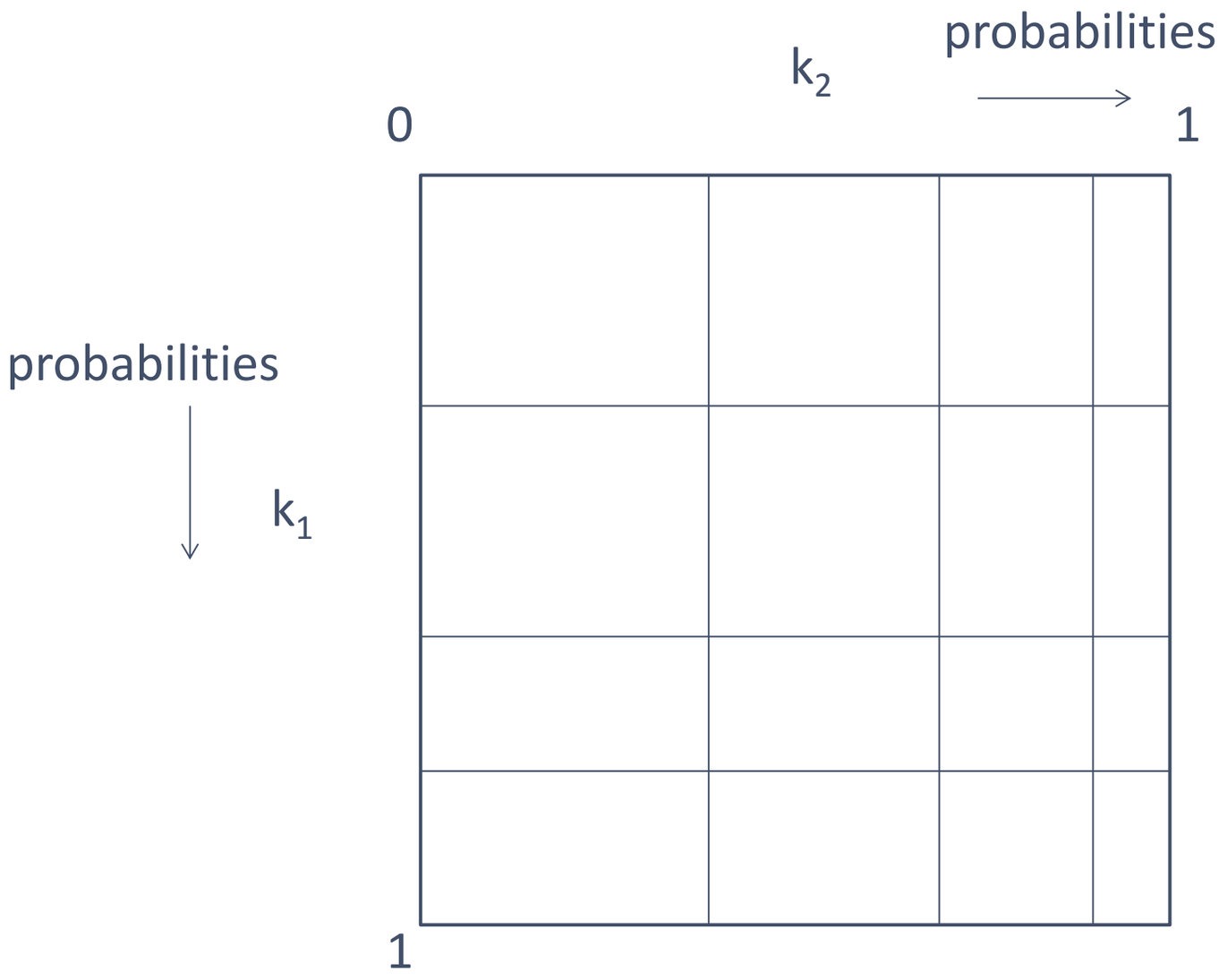
Optimal Key Enumeration Algorithm (OKEA) [SAC 2012]

- Proposed by Nicolas Veyrat-Charvillon, Benoît Gérard, Mathieu Renauld and François-Xavier Standaert in 2012

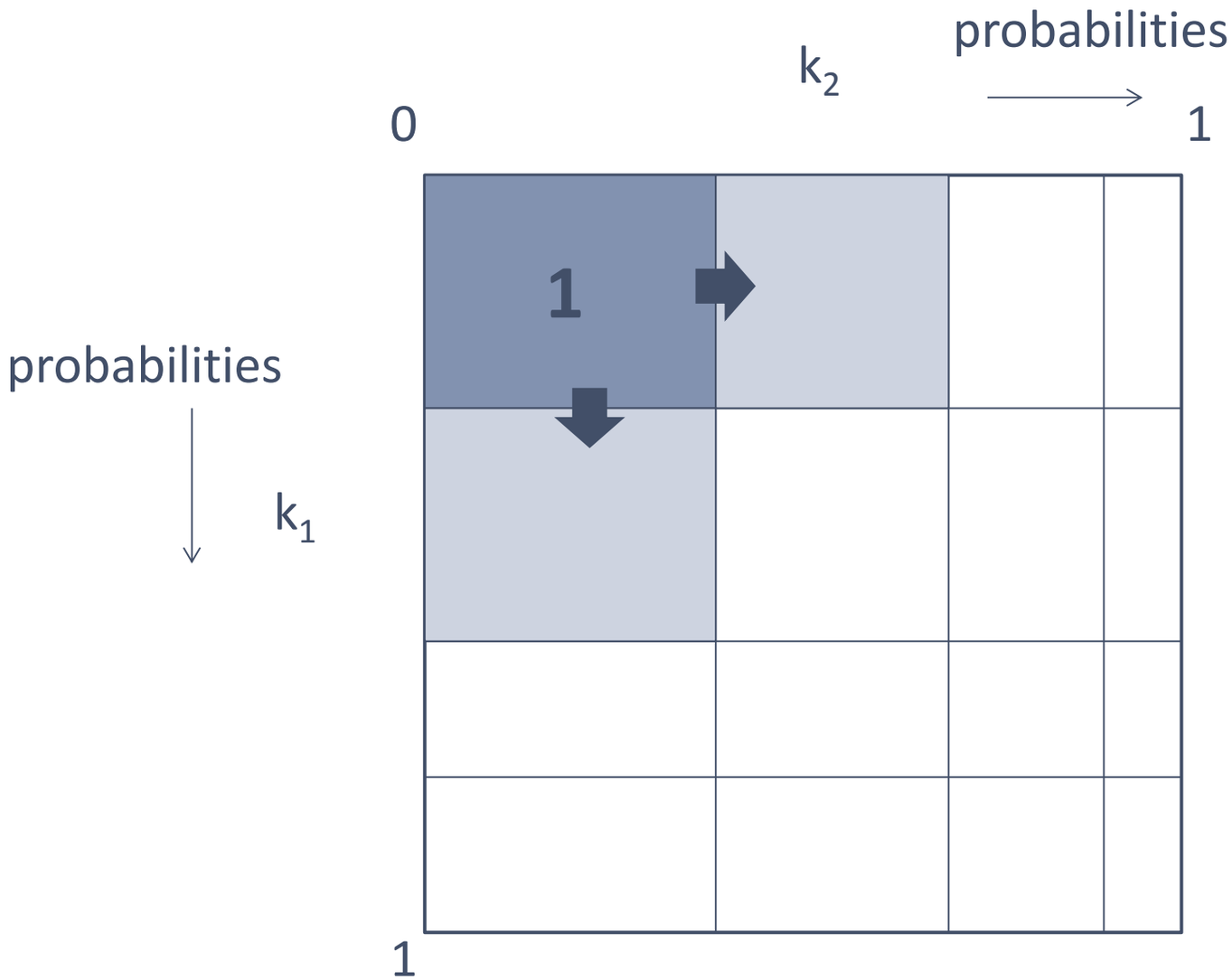
- Requires probability based ranking of sub-key lists

- Let us discuss a simple case of merging two lists

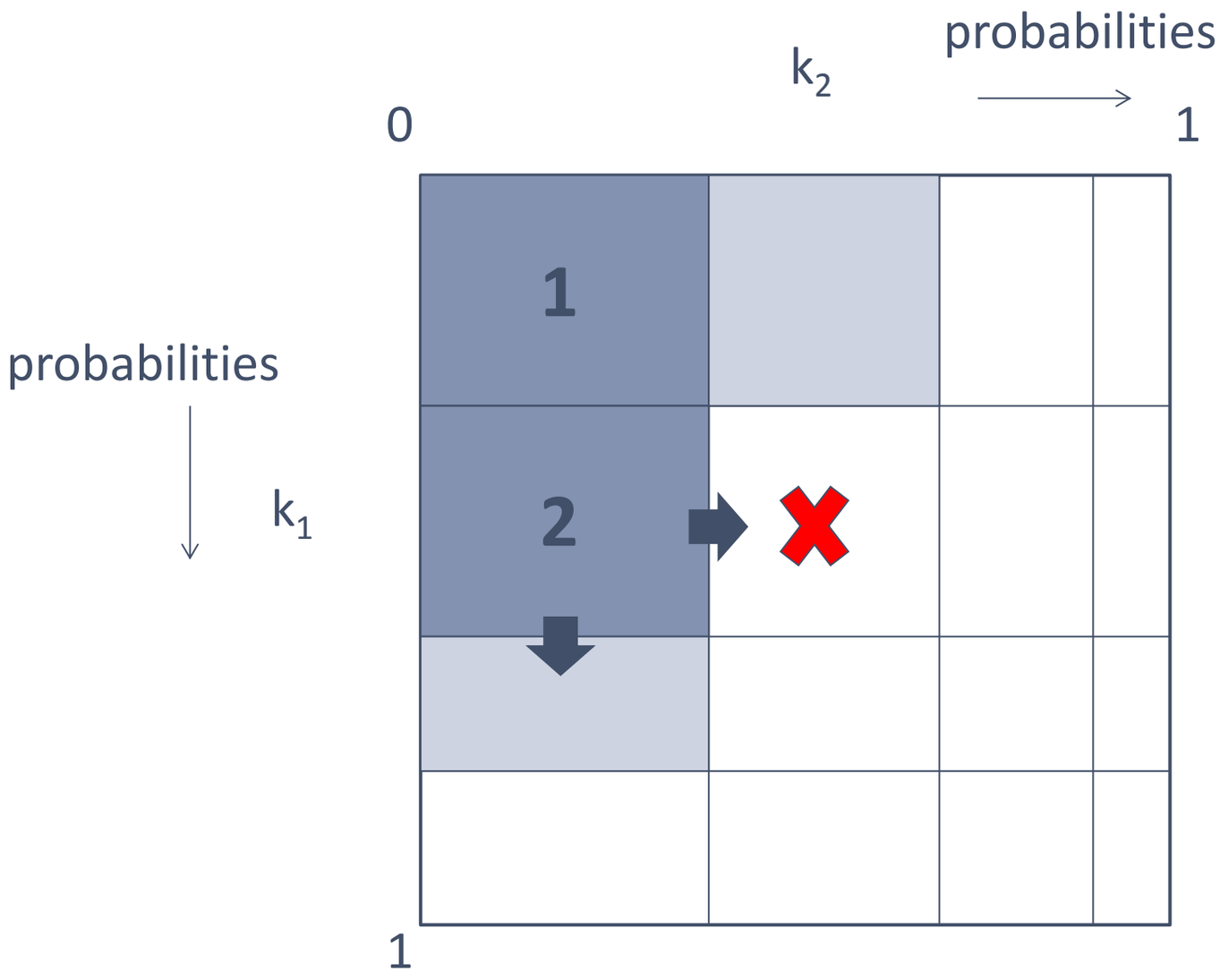
Optimal Key Enumeration Algorithm (OKEA) [SAC 2012]



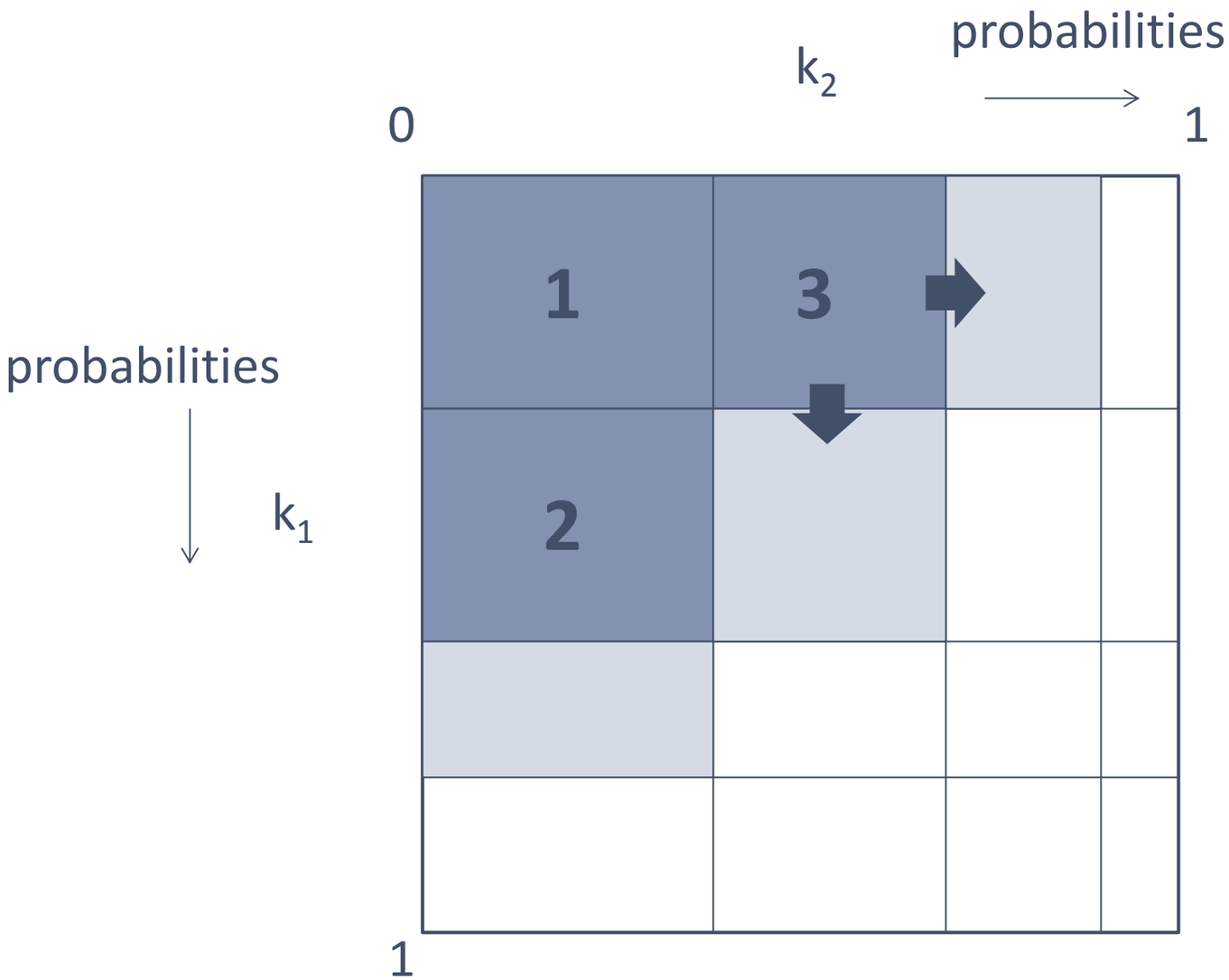
Optimal Key Enumeration Algorithm (OKEA) [SAC 2012]



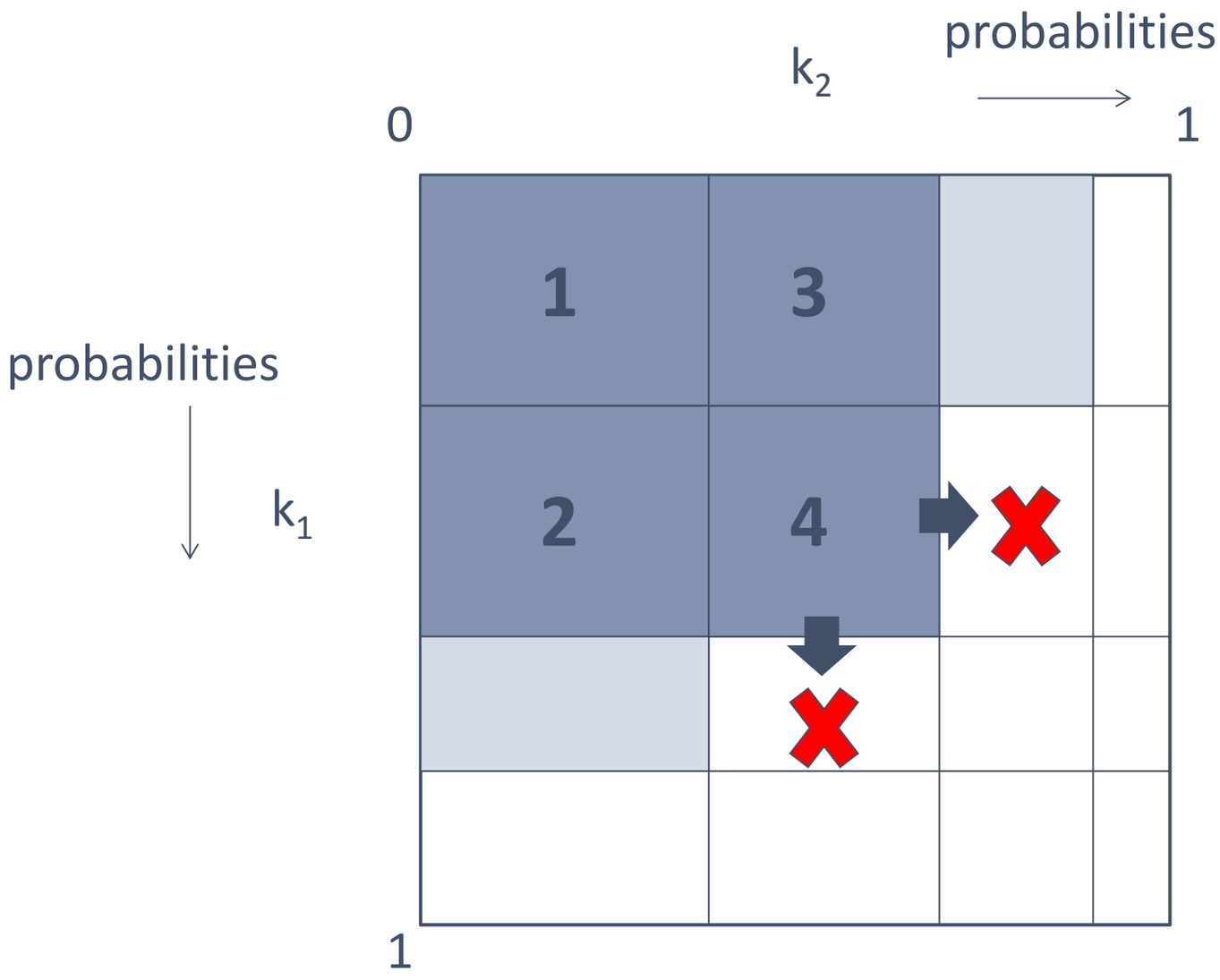
Optimal Key Enumeration Algorithm (OKEA) [SAC 2012]



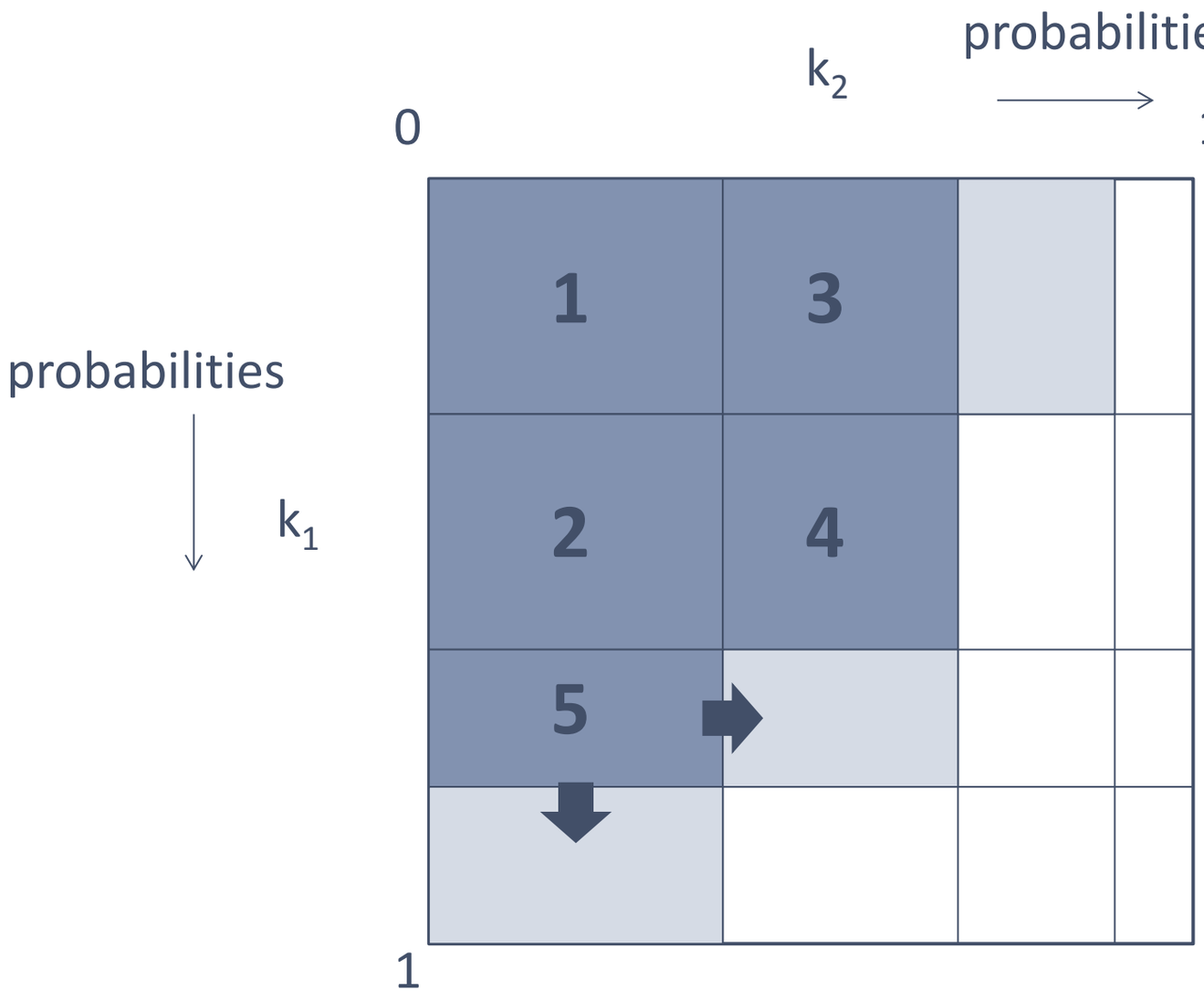
Optimal Key Enumeration Algorithm (OKEA) [SAC 2012]



Optimal Key Enumeration Algorithm (OKEA) [SAC 2012]

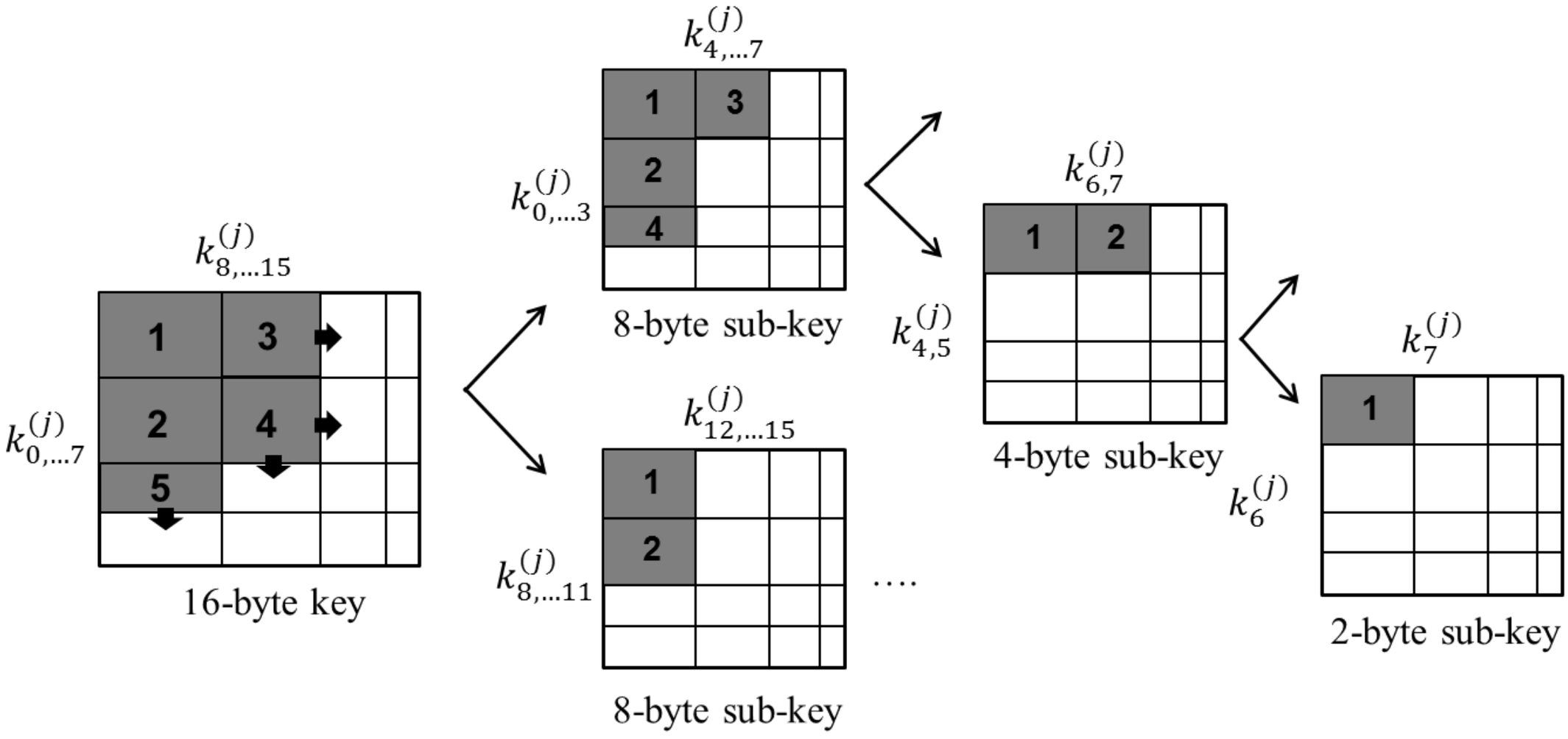


Optimal Key Enumeration Algorithm (OKEA) [SAC 2012]



Optimal Key Enumeration Algorithm (OKEA) [SAC 2012]

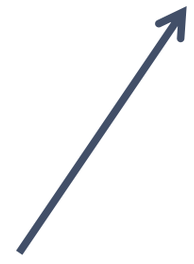
- Merging n lists:
  - merging two lists n-1 times
  - recursive decomposition



OKEA Performance on CPU [SAC 2012]

Figures from the paper

# key candidates	2^{12}	2^{16}	2^{20}	2^{24}	2^{28}	2^{32}	2^{36}	2^{40}
Time	0s	0.03s	0.55s	9.2s	163s	3130s	12h	221h
Memory	88KB	405KB	2.7MB	20MB	225MB	1.8GB	10GB	70GB
Throughput keys/sec	-	2^{21}	2^{21}	2^{21}	2^{21}	2^{20}	2^{21}	2^{20}



predictions!

Optimal Key Enumeration Algorithm (OKEA) [SAC 2012]

- Analysis:
  - Consumes large amount of memory
  - Strictly follows optimality
  - Has very low throughput
  - Takes large amount of time to generate the correct key
- Above claims are based on our experimental results
- Still not of our interest… 

Score based Key Enumeration Algorithm (SKEA) [Marc Witteman]

- Requires discrete score based ranking of sub-key lists (conversion e.g. by multiplication and rounding)

- Main contribution of this work

- Explained in the following slides at a toy example (4 key chunks)

Score based Key Enumeration Algorithm (SKEA) [Marc Witteman]

Key part ⟶

Score ↓

	0	1	2	3
0	5	5	4	5
1	2	3	2	4
2	1	1	0	3
3	0	1	0	2

Best cumulated score: 5+5+4+5=19

Next: 5+5+4+4=18

Enumeration is done by descending cumulated score

Score based Key Enumeration Algorithm (SKEA) [Marc Witteman]

19

5	5	4	5
2	3	2	4
1	1	0	3
0	1	0	2

18

5	5	4	5
2	3	2	4
1	1	0	3
0	1	0	2

17

5	5	4	5
2	3	2	4
1	1	0	3
0	1	0	2

5	5	4	5
2	3	2	4
1	1	0	3
0	1	0	2

5	5	4	5
2	3	2	4
1	1	0	3
0	1	0	2

16

5	5	4	5
2	3	2	4
1	1	0	3
0	1	0	2

5	5	4	5
2	3	2	4
1	1	0	3
0	1	0	2

5	5	4	5
2	3	2	4
1	1	0	3
0	1	0	2

5	5	4	5
2	3	2	4
1	1	0	3
0	1	0	2

Score based Key Enumeration Algorithm (SKEA) [Marc Witteman]

Key part ⟶

Score ↓

	0	1	2	3
0	5	5	4	5
1	2	3	2	4
2	1	1	0	3
3	0	1	0	2

Score based Key Enumeration Algorithm (SKEA) [Marc Witteman]

Key part →

Score ↓

	0	1	2	3
0	5	5	4	5
1	2	3	2	4
2	1	1	0	3
3	0	1	0	2

Cumulated Maximum

19	14	9	5

Score based Key Enumeration Algorithm (SKEA) [Marc Witteman]

riscure

Key part ⟶

Score ↓

	0	1	2	3
0	5	5	4	5
1	2	3	2	4
2	1	1	0	3
3	0	1	0	2

Cumulated Maximum

19	14	9	5

Cumulated Minimum

3	3	2	2

Score based Key Enumeration Algorithm (SKEA) [Marc Witteman]

riscure

Key part ⟶

Score ↓

	0	1	2	3
0	5	5	4	5
1	2	3	2	4
2	1	1	0	3
3	0	1	0	2

Cumulated Maximum

19	14	9	5

Cumulated Minimum

3	3	2	2

Cumulated score ↓

Key part ⟶

	0	1	2	3
19				
18				
17				
16				
15				
14				
13				
12				
11				
10				
9				
8				
7				
6				
5				
4				
3				
2				

Table shows for each key part for a given accumulated score

- Minimal index
- Maximal index

Score based Key Enumeration Algorithm (SKEA) [Marc Witteman]

Key part ⟶

Score ↓

	0	1	2	3
0	5	5	4	5
1	2	3	2	4
2	1	1	0	3
3	0	1	0	2

Cumulated Maximum

19	14	9	5

Cumulated Minimum

3	3	2	2

Table shows for each key part for a given accumulated score

- Minimal index
- Maximal index

Key part ⟶

Cumulated score ↓

	0	1	2	3
19	0 / 0			
18				
17				
16				
15				
14				
13				
12				
11				
10				
9				
8				
7				
6				
5				
4				
3				
2				

Score based Key Enumeration Algorithm (SKEA) [Marc Witteman]

riscure

Key part ⟶

Score ↓

	0	1	2	3
0	5	5	4	5
1	2	3	2	4
2	1	1	0	3
3	0	1	0	2

Cumulated Maximum

19	14	9	5

Cumulated Minimum

3	3	2	2

Cumulated score ↓

Key part ⟶

	0	1	2	3
19	0 / 0			
18	0 / 0			
17	0 / 0			
16	0 / 1			
15	0 / 2			
14	0 / 3	0 / 0		
13	0 / 3	0 / 0		
12	0 / 3	0 / 1		
11	0 / 3	0 / 1		
10	0 / 3	0 / 3		
9	0 / 3	0 / 3	0 / 0	
8	0 / 3	0 / 3	0 / 0	
7	1 / 3	0 / 3	0 / 1	
6	1 / 3	1 / 3	0 / 1	
5	2 / 3	2 / 3	1 / 3	0 / 0
4	3 / 3	2 / 3	1 / 3	1 / 1
3	3 / 3	2 / 3	2 / 3	2 / 2
2				3 / 3

Table shows for each key part for a given accumulated score

- Minimal index
- Maximal index

Score based Key Enumeration Algorithm (SKEA) [Marc Witteman]

- Analysis:
  - Consumes very limited constant amount of memory
  - Sub-optimal
  - Has very high throughput
  - Takes realtively less time to reach the correct key
- Above claims are based on our experimental results
- Satisfies most of the properties of our interest. 

Experimental comparison

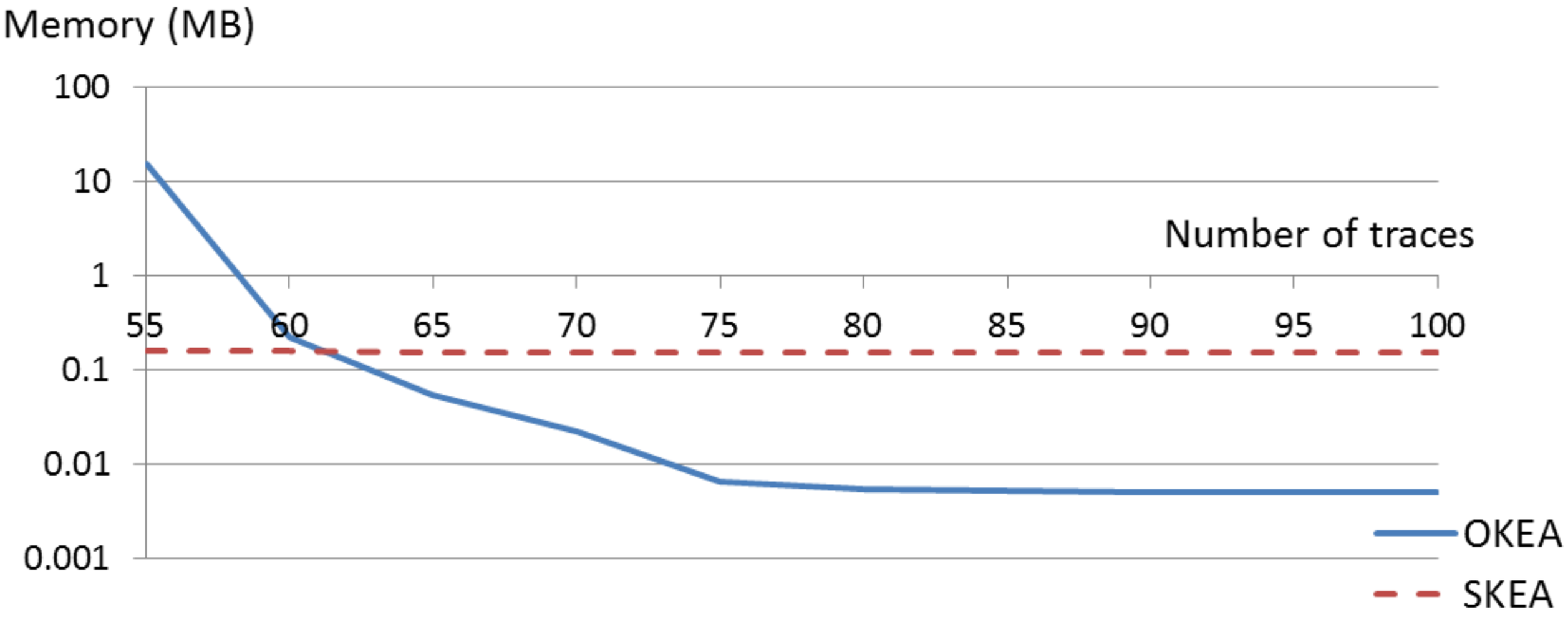
- Key enumeration algorithms under consideration work for all side-channel attacks that employ divide and conquer strategy.
- In order to evaluate their performance, we just need sorted sub-key lists.
- Executed a correlation coefficient based Differential Power Analysis (DPA) attack on ATmega implementation of AES-128.
- Yielded 16 sub-key lists sorted according to the correlation scores of the sub-key candidates.

Experimental Comparisons

- The attack was executed with varying the number of power traces 55, 60, …, 100.
- Collected 12 cases for each number of power traces. 12 cases for 100 traces, 12 cases for 95 and so on.
- Executed the key enumeration algorithms on a common laptop (Intel core i5-2450M with 8 GB RAM and running 64-bit Windows 7).
- 1 hour was the time limit to find the correct key.
- Both OKEA and SKEA were able to find the correct key within 1 hour. TKEA could only find the key till 70 traces.

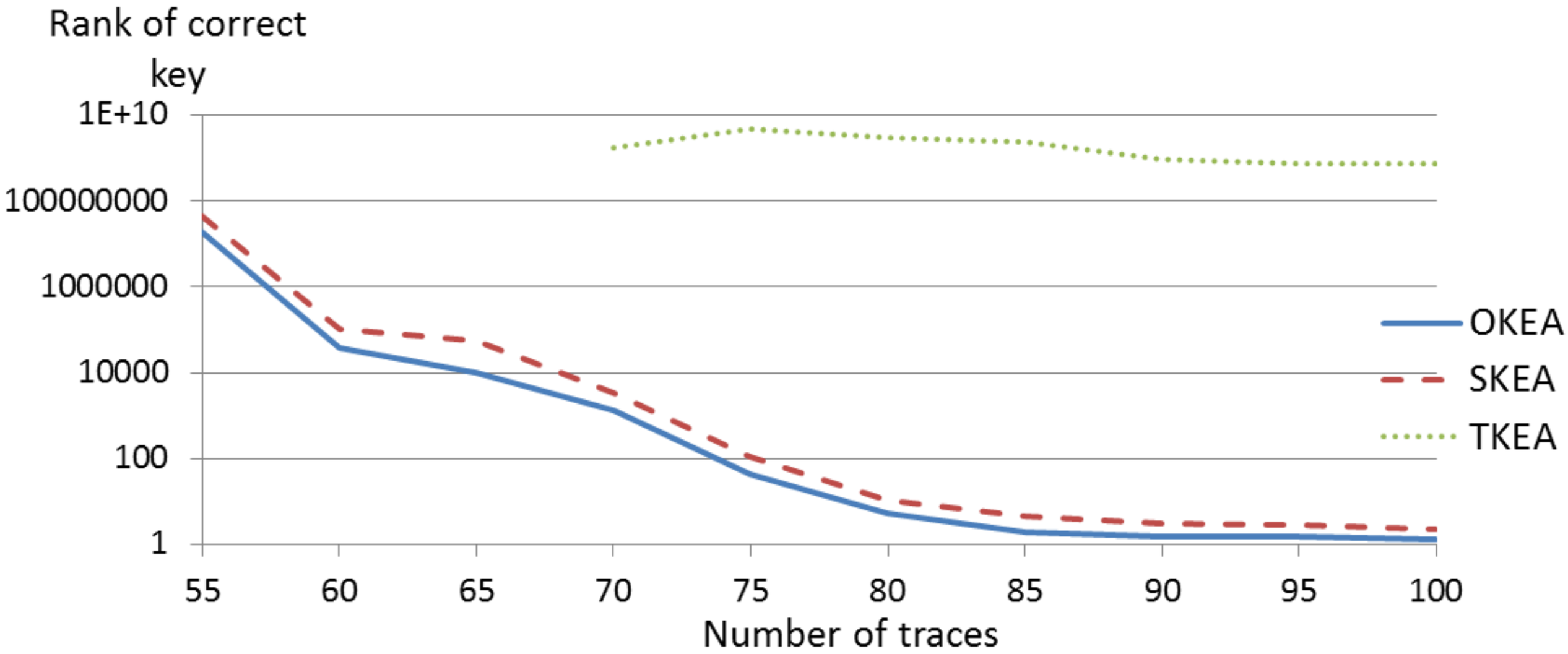
Experimental Comparison

Memory Consumption



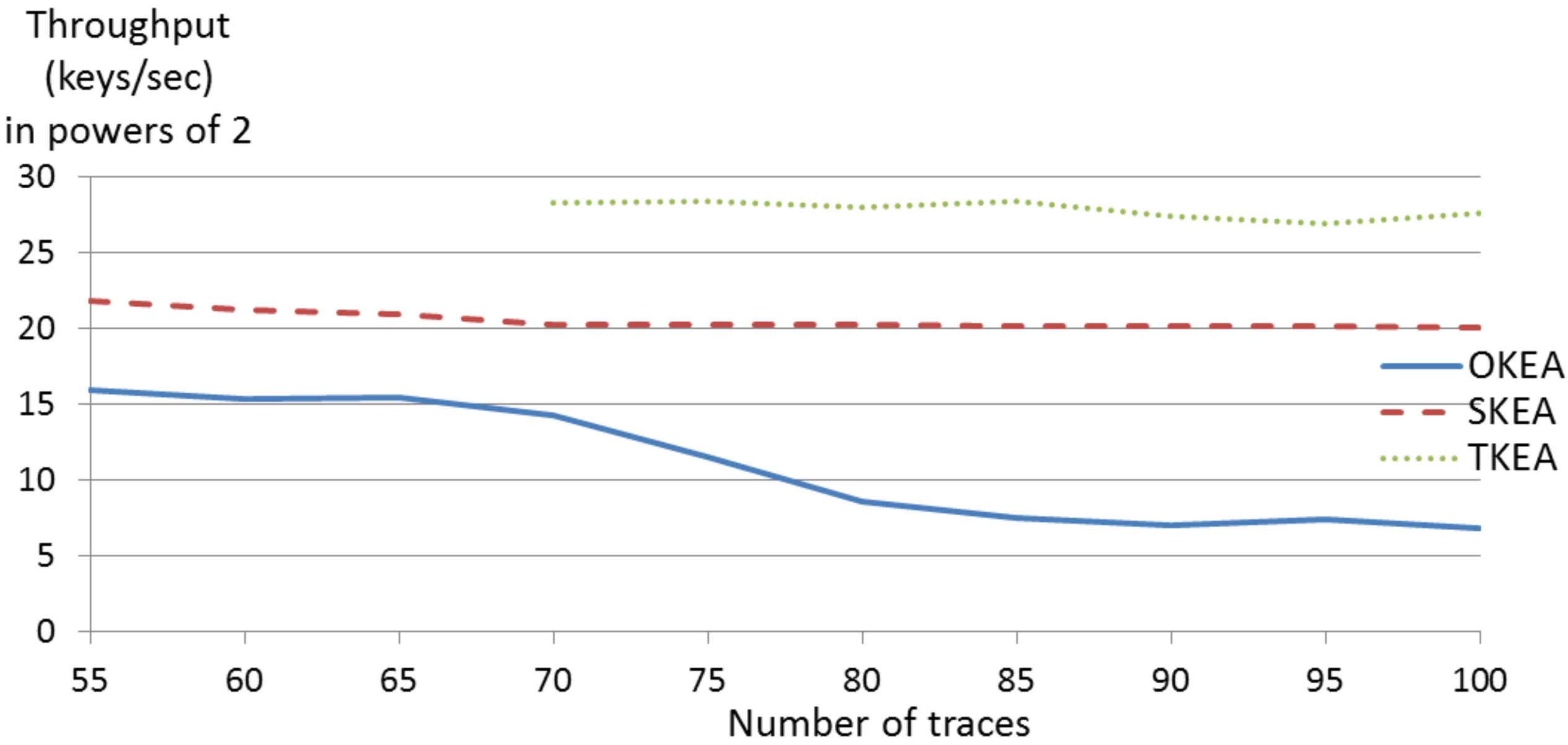
Experimental Comparison

Optimality



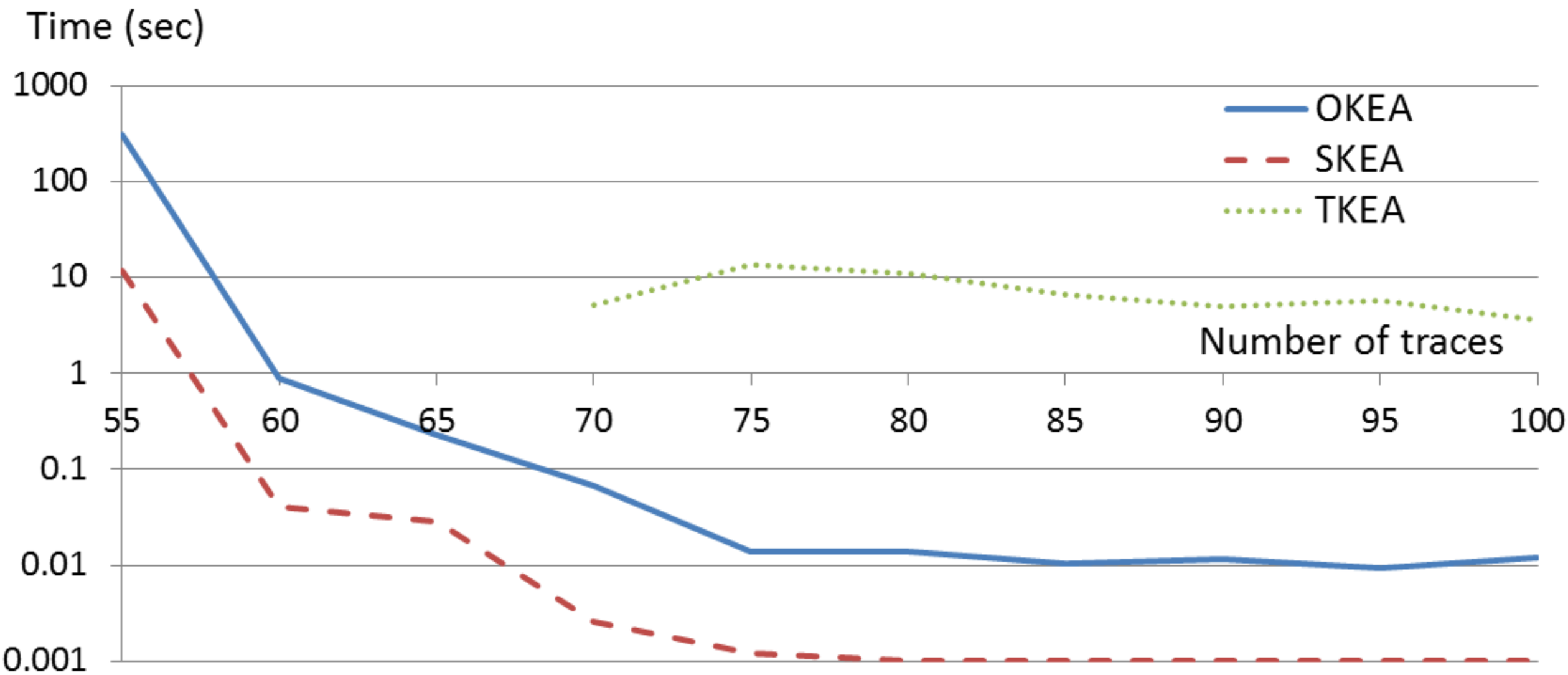
Experimental Comparisons

Throughput



Experimental Comparisons

Time to find the Correct Key



Discussion

- None of three key enumeration algorithms satisfy all the properties of our interest

- In contrast to both TKEA and OKEA, SKEA satisfies most of them

- **We propose to deploy SKEA as key enumeration algorithm**

Key Validation

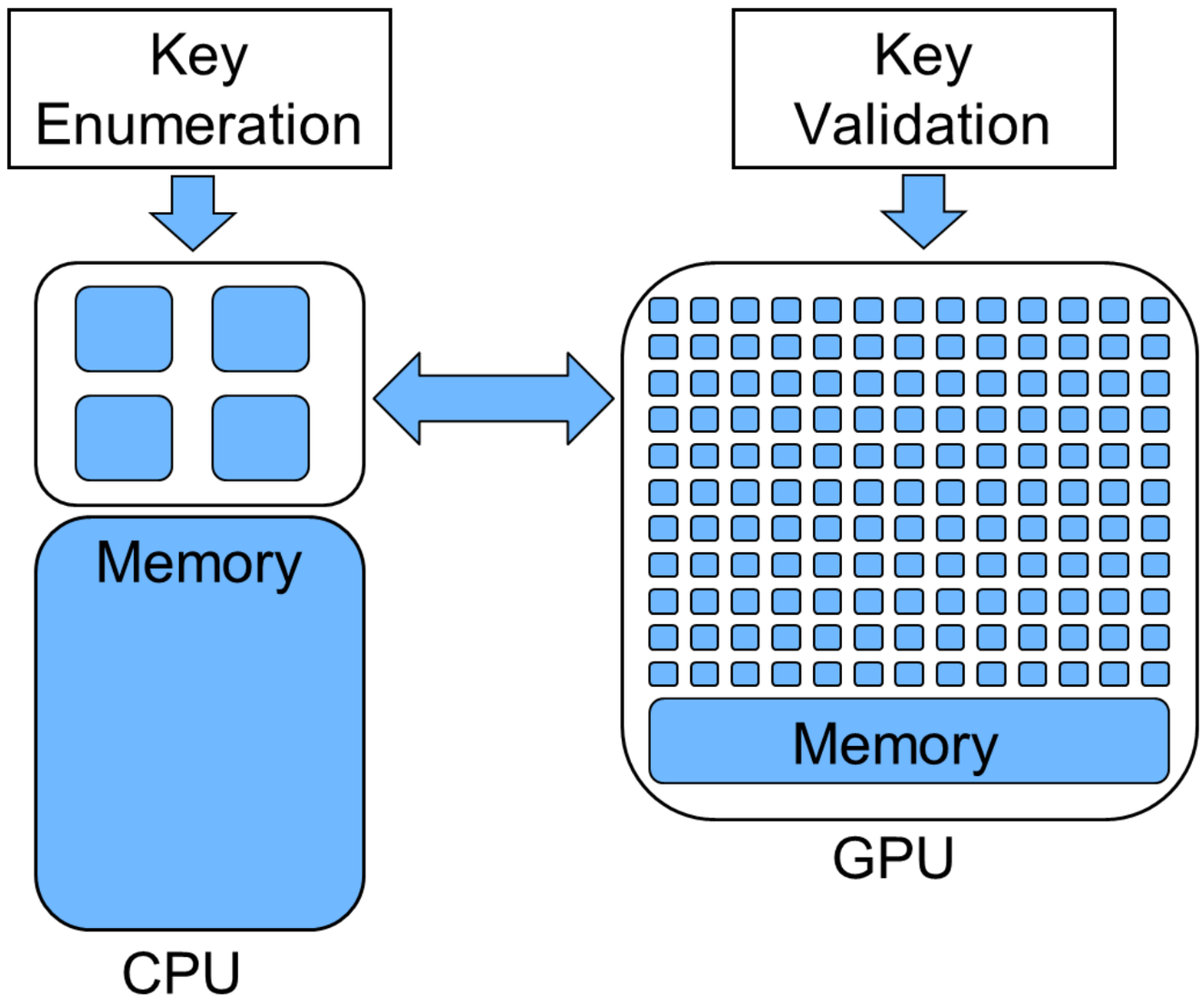
- Second step of key recovery mechanism
- Test whether the key generated by the key enumeration step is correct or not
- In order to validate a key, an attacker encrypts a particular plaintext with that key and matches the corresponding ciphertext with the one already computed with the stored secret key
- **Key validation rate should be compatible with key generation rate**

Key Validation

- We propose to deploy SKEA. Maximum throughput of SKEA $\cong 2^{23}\ keys/\text{sec}$
- Throughput of key validation should at least be $\cong 2^{23}\ keys/\text{sec}$
- Utilize the immense parallel processing power of a GPU
- Implemented AES on an NVIDIA GPU using CUDA platform
- Achieved key validation rate of more than $\cong 2^{23}\ \ keys/\text{sec}$

Proposed Solution

- Simultaneous execution of key enumeration (SKEA) on a CPU and key validation on a GPU.



riscure

Challenge your security

Contact: Ilya Kizhvatov
ilya@riscure.com

Riscure B.V.
Frontier Building, Delftechpark 49
2628 XJ Delft
The Netherlands
Phone: +31 15 251 40 90

www.riscure.com

Riscure North America
71 Stevenson Street, Suite 400
San Francisco, CA 94105
USA
Phone: +1 650 646 99 79

inforequest@riscure.com