A New Key Rank Estimation Method to Investigate Dependent Key Lists of Side-Channel Attacks

Shuang Wang, Yang Li*, Jian Wang

li.yang@nuaa.edu.cn, li.yangheu@gmail.com

Nanjing University of Aeronautics and Astronautics, Nanjing, China

Oct. 20th, 2017 @ AsianHost 2017

Outline

- 1. Background
  - Side-channel attacks, divide and conquer
  - Key enumeration and key rank estimation
- 2. Motivation & Problems Setting
  - Key recovery vs key difference recovery
  - Dependent key list
- 3. Our Solution
  - Basic idea
  - Dependent-Key List Rank Estimation Algorithm (DK-REA)
- 4. Experimental Verification
  - Setup & Results & Analysis
- 5. Conclusions and Future Work

Outline

- 1. Background
  - Side-channel attacks, divide and conquer
  - Key enumeration and key rank estimation
- 2. Motivation & Problems Setting
  - Key recovery vs key difference recovery
  - Dependent key list
- 3. Our Solution
  - Basic idea
  - Dependent-Key List Rank Estimation Algorithm (DK-REA)
- 4. Experimental Verification
  - Setup & Results & Analysis
- 5. Conclusions and Future Work

Side-channel attack and its evaluation

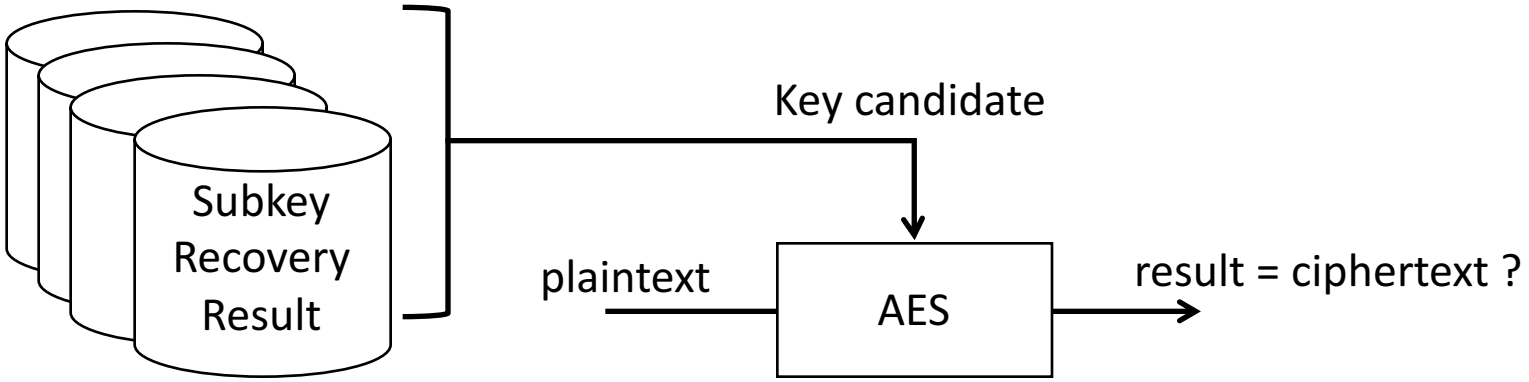
- Side-channel attack is powerful
  - Power analysis, time analysis, electromagnetic analysis etc.
  - Practical security threat to cryptographic devices
- SCA follows a divide-and-conquer approach
  - Key is divided into sub-keys to be recovered independently
  - When all sub-keys are successfully recovered, the whole key is recovered as well
  - Successful recovery means correct sub-key has “highest score” among all key candidates
- Security evaluation of cryptographic implementations
  - Data complexity (1k ~ 5M traces for key recovery)
  - Computational complexity (Usually not very high)

How to define a successful attack

- Full-key recovery
  - Each sub-key has the highest score
  - Useful for practical attackers
  - Focus of this paper
- Sub-key recovery
  - Recover some sub-keys
  - Useful for security evaluation
    - Global Success Rate > 80% used in DPA contest v2
- Existence of sensitive leakage
  - Detection of leakage of sensitive information
    - T-test etc.
  - Useful for strict security evaluation, countermeasure proposal

Trade-off between data and computations

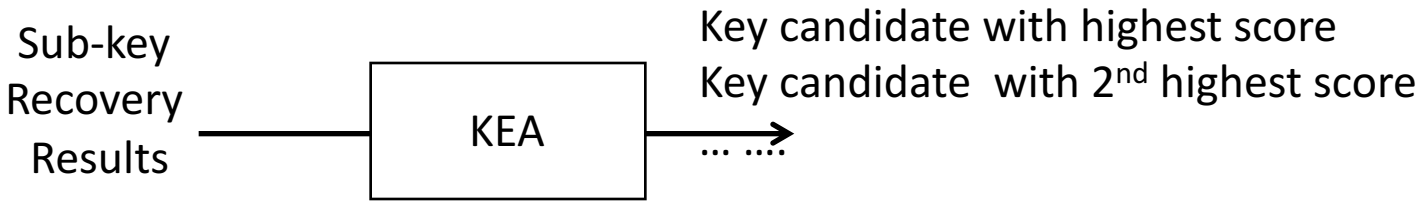
- In full-key recovery, attackers can always use plaintext-ciphertext test to verify the correctness of a key



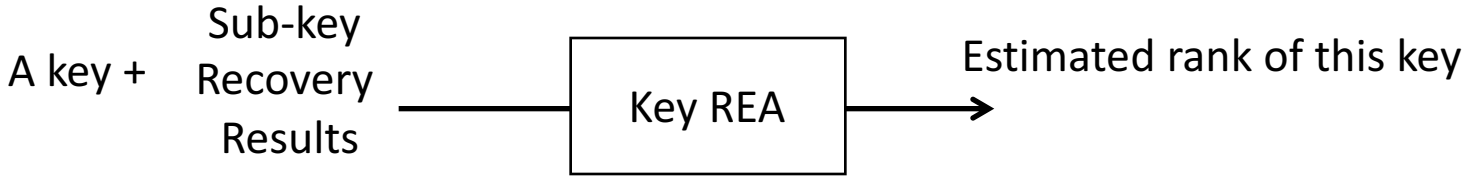
  - E.g, verify 2^{40} most likely key candidates to find the correct key
  - More computation to trade for less data complexity!
- How to enumerate the most likely keys from sub-key recovery results?
  - to minimize # of test trails.
- How to evaluate the rank of a certain key candidate?
  - to quickly understand required # of test trails.

Key Enumeration Algorithm (KEA) and Key Rank Estimation Algorithm (REA)

- Key Enumeration Algorithm
  - Enumerate key candidates in decreasing order of likelihood
  - Useful for practical attack



- Key Rank Estimation Algorithm
  - Estimate rank of a given key
  - Useful for known-key evaluation
    - Rank can beyond the KEA's computational power



Existing Key RE Algorithms

- First rank estimation algorithm proposed in Eurocrypt 2013
  - transfers search problem to a multidimensional problem
  - 10 bits of accuracy
- In 2015, Bernstein et al. introduced two better RE algorithms
  - First Alg. add post processing to increase accuracy from 10 bits to 5 bits
  - Second Alg. ranks the keys based on polynomial problem, flexible accuracy at the expense of run time.
- In 2015, Glowacz et al. proposed a rank estimation algorithm
  - Based on convolution of histograms.
  - Bounds with less than 1 bit of tightness within seconds of computation
  - Scales to larger key sizes
- In 2015, Martin et al. mapped the rank estimation problem
  - to a multi-dimensional knapsack problem
- All of them focused on merging independent key lists

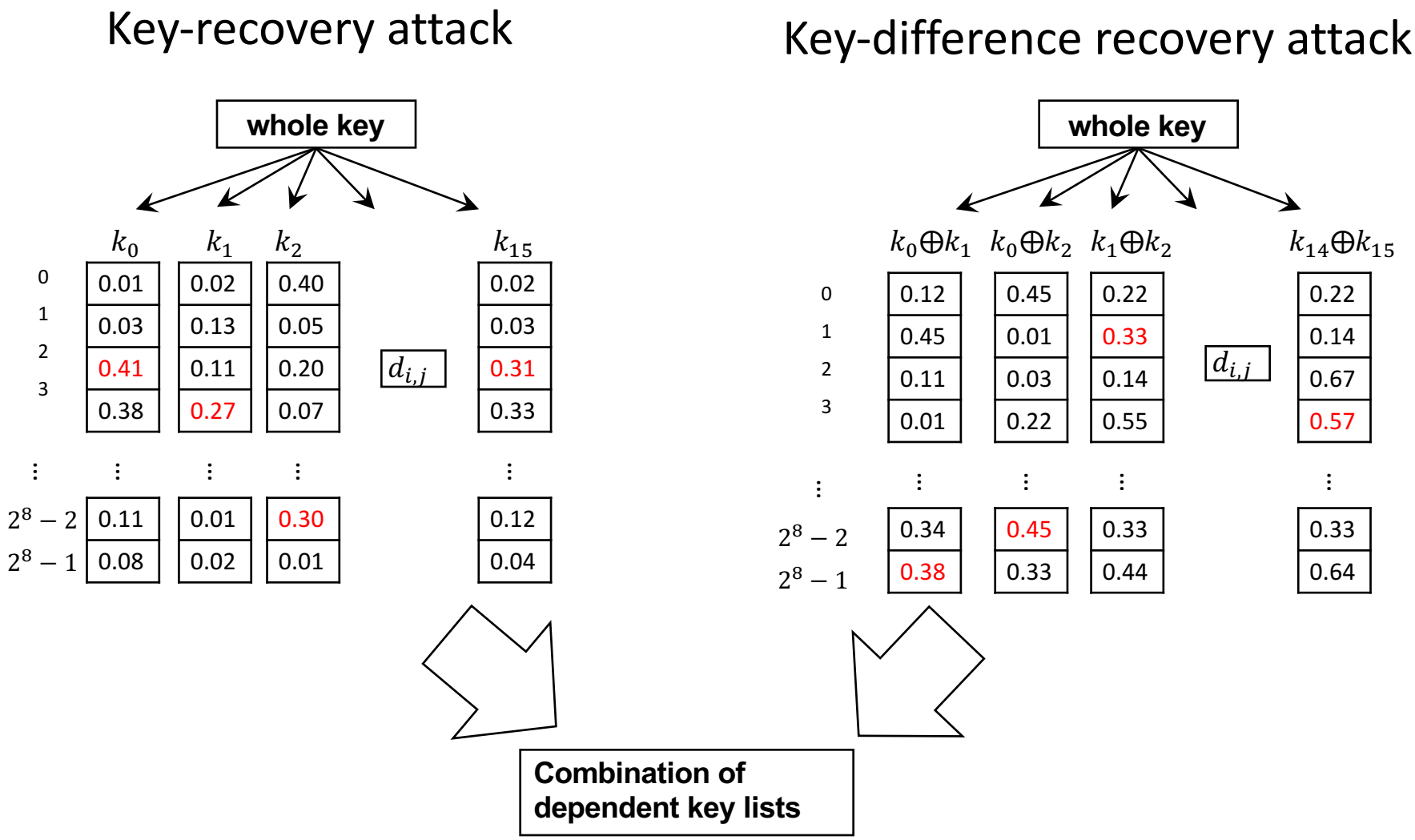
Outline

- 1. Background
  - Side-channel attacks, divide and conquer
  - Key enumeration and key rank estimation
- 2. Motivation & Problems Setting
  - Key recovery vs key difference recovery
  - Dependent key list
- 3. Our Solution:
  - Ideas
  - Dependent-Key List Rank Estimation Algorithm (DK-REA)
- 4.Experimental Verification
  - Setup & Results & Analysis
- 5. Conclusions and Future Work

Motivation of this work

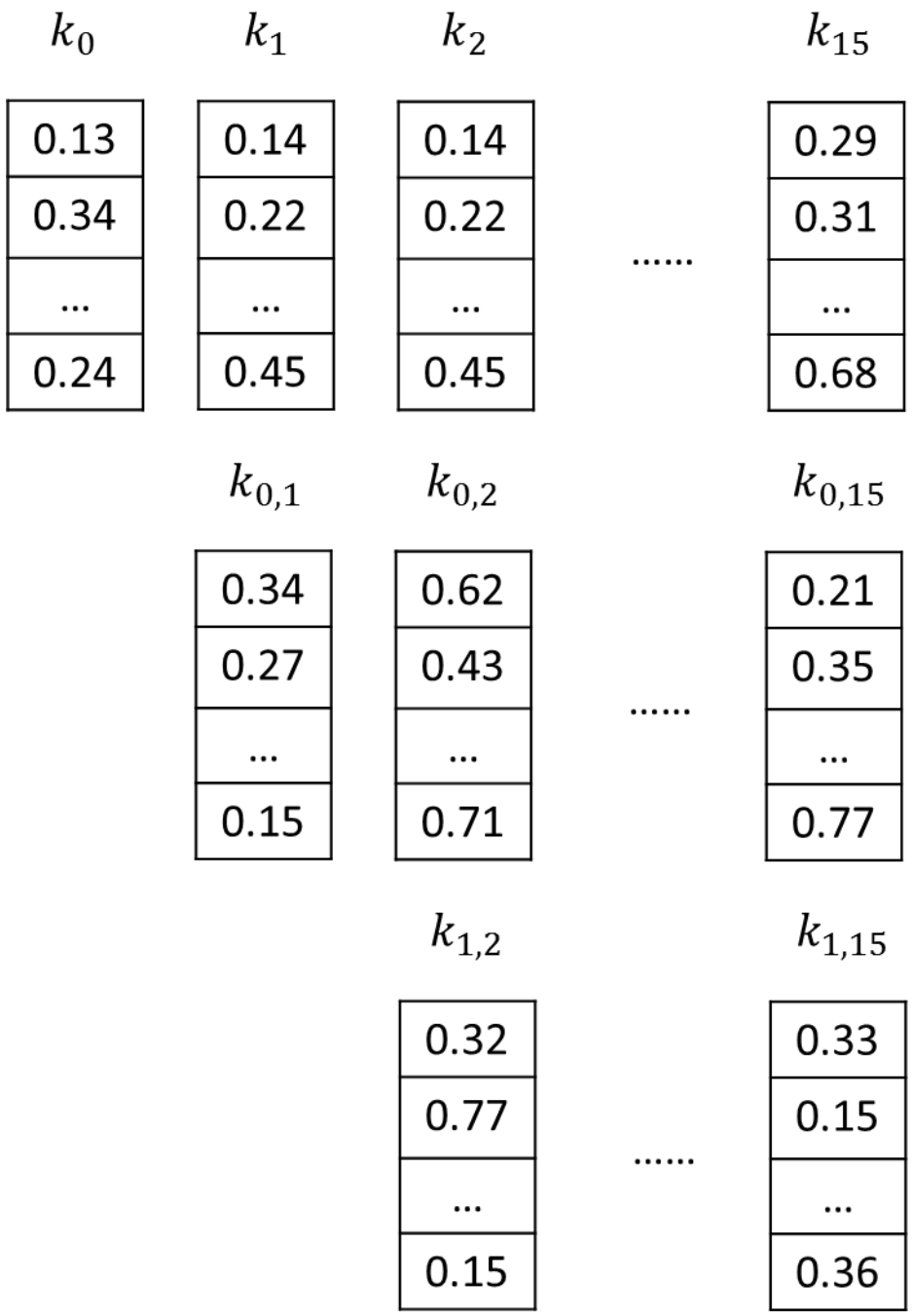
- Key is divided into sub-keys to be recovered independently
- But key can be divided in different formats, e.g. there are two well-known types of SCAs
- Key Recovery Attacks:
  - Target: $k_0, k_1, \dots, k_{15}$
  - Based on leakage model from small component (e.g. S-box)
  - Differential Power Analysis, Correlation Power Analysis, Mutual Information Analysis, etc
- Key Difference Recovery Attack
  - Target : $k_{0,1} = k_0 \oplus k_1, k_{0,2}, \dots, k_{0,15}, k_{1,2}, \dots, k_{1,15}, \dots, k_{14,15}$
  - Based on similarity of leakage from two small components
  - Collision attacks, Correlation-Enhanced Collision Attacks

Can we combine these attack results?



Motivation

- Why to do so?
  - Intuitively, more score lists should lead to better attack result!
  - Different attacks use different information to recover key
    - Leakage model
    - Collision model
- We want to
  - Propose a key RE algorithm for dependent key lists (DK-REA),
  - Verify whether using more key lists is helpful



Outline

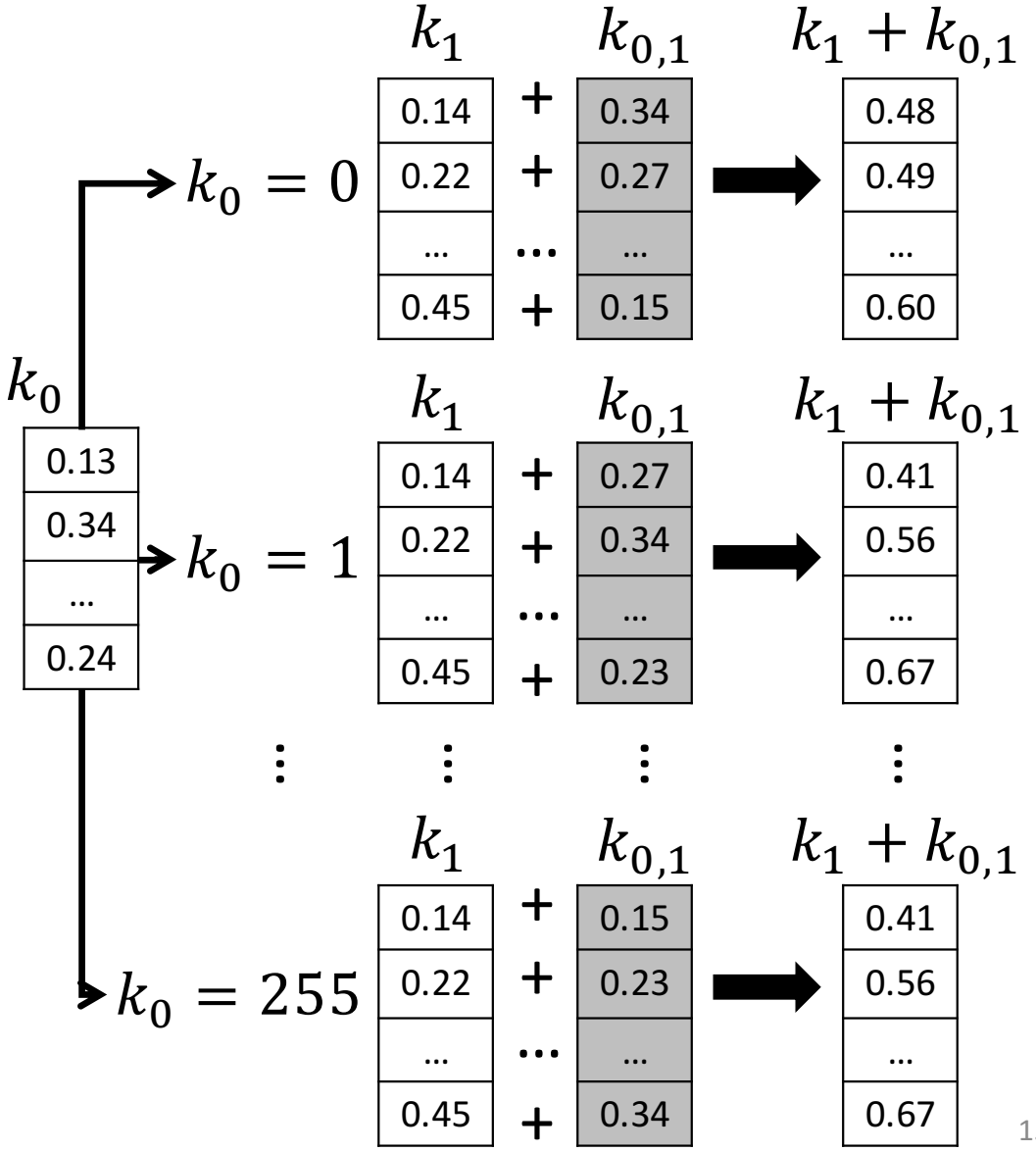
- 1. Background
  - Side-channel attacks, divide and conquer
  - Key enumeration and key rank estimation
- 2. Motivation & Problems Setting
  - Key recovery vs key difference recovery
  - Dependent key list
- 3. Our Solution
  - Basic idea
  - Dependent-Key List Rank Estimation Algorithm (DK-REA)
- 4. Experimental Verification
  - Setup & Results & Analysis
- 5. Conclusions and Future Work

Straightforward solution cannot work

- DK-KRA is not trivial due to dependency
- Many meaningless combinations don't satisfy dependency
  - E.g. $k_0 = 0, k_1 = 1, k_{0,1} = 2$ is invalid since $k_0 \oplus k_1 \neq k_{0,1}$
  - Actually, most of combinations are invalid consider 16 key bytes and 15 key byte differences
  - Straightforward combination is impracticable
- A new method to combine dependent key lists
  - Deal with dependency before combination

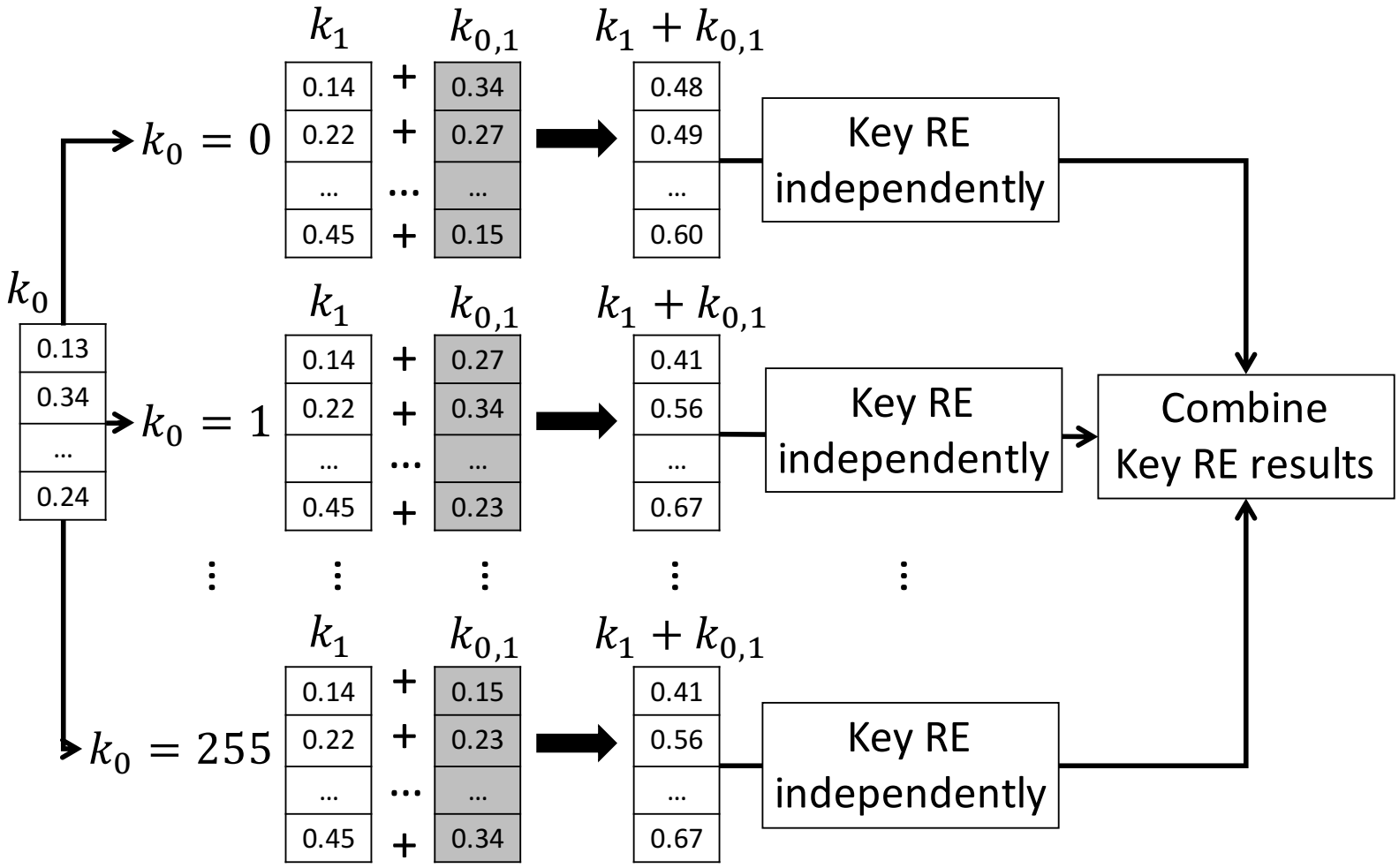
Our Solution: divide and conquer

- Take k_0, k_1 and $k_{0,1}$ as an example, there is an XOR relationship between them as $k_0 \oplus k_1 = k_{0,1}$
- When a subkey i.e. k_0 is fixed, there is a one-one correspondence between k_1 and $k_{0,1}$
- We combine key lists of k_1 and $k_{0,1}$ to obtain a new list as k_1+$k_{0,1}$, which is independent from $k_2, k_3, \ldots, \mathrm{k}_{15}$



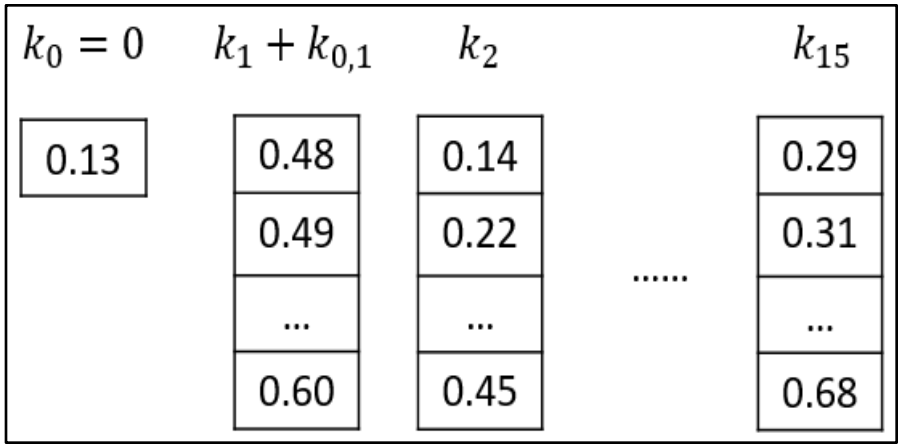
Our Solution: divide and conquer

- Key RE for dependent lists is reduced to 2^8=256 key RE problems for independent key lists

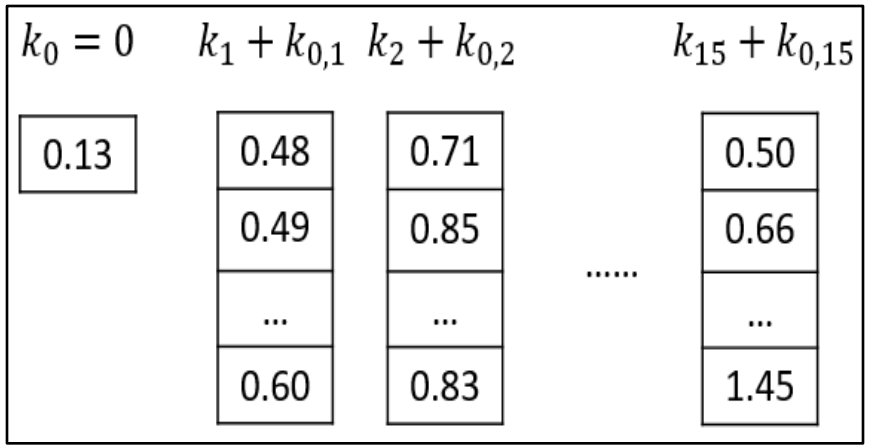


Our General Solution

- Fix 1 subkey k_i, there is correspondence between k_j and $k_{i,j}$, $j \in \{0,1,\dots,15\}$.
- Fix 1 key byte allows to add up to 15 subkey-differece score lists
- When $k_i = m$, $m \in \{0,1,\dots,255\}$, we combine $(k_j, k_{i,j}) = s(k_j) + s(k_{i,j}{=}k_j \oplus m)$, $s(k_j)$ represents the score of k_j
- $k_p + k_{i,p}$ and $k_q + k_{i,q}$ are independent from each other as well



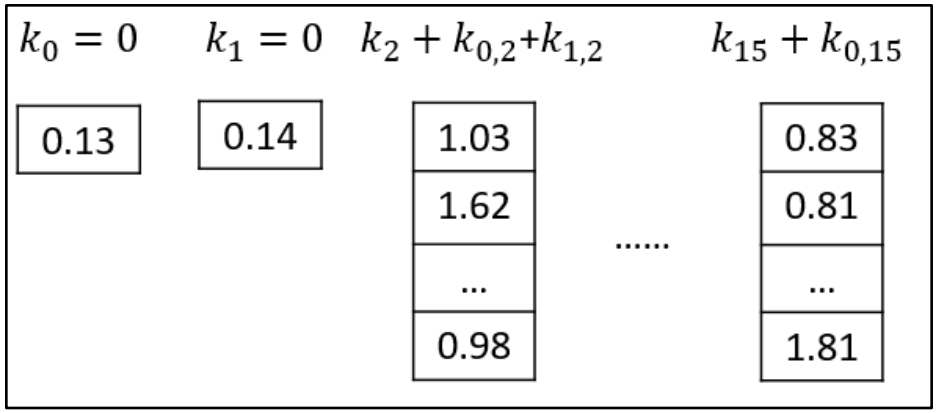
Adding 1 subkey-difference score list



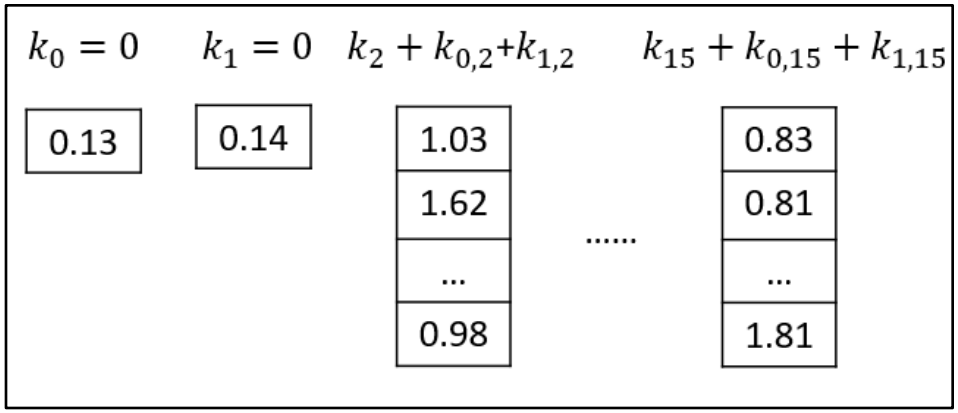
Adding 15 subkey-difference score lists

Our General Solution

- If we want to add more than 15 dependent score lists, fix 1 subkey is not enough.
- We can fix 2 subkeys, such as k_0, k_1, and add up to 29 lists
- 15+14=29 lists
- There is a correspondence between k_0, k_i and $k_{0,i}$ and $k_{1,i}$.



Adding 16 subkey-difference score lists



Adding 29 subkey-difference score lists

Limitations of DK-REA

- Computational overhead
  - Adding N subkey difference lists
  - If 1<N<16, computational complexity is increased by a factor of 2^8
  - If 15<N<30, computational complexity is increased by a factor of 2^{16}
- Added dependent score lists are fixed
  - Dependent score lists that can be added cannot be freely chosen
  - Depends on the selected fixed subkey
  - e.g., when we select k_0, 15 dependent score lists that can be added are $k_{0,1}$, $k_{0,2}$, $k_{0,3}$,…, $k_{0,15}$.

Outline

- 1. Background
  - Side-channel attacks, divide and conquer
  - Key enumeration and key rank estimation
- 2. Motivation & Problems Setting
  - Key recovery vs key difference recovery
  - Dependent key list
- 3. Our Solution
  - Basic idea
  - Dependent-Key List Rank Estimation Algorithm (DK-REA)
- 4. Experimental Verification
  - Setup & Results & Analysis
- 5. Conclusions and Future Work

Experimental Setup: simulated leakage

- Target: AES-128
  - 16 S-boxes in first AES round
  - Serial implementation
  - Leakage model: Hamming weight of the S-box output with noise
- We simulated two attacks
  - Correlation Power Analysis → key recovery
  - Correlation Enhanced Collision Attack → key difference recovery
- Two types of noises in the leakage measurement

$$LeakageModel(\alpha) = HW(\alpha) + \mathcal{N}(0, 4^2)$$

$$MesauredLeakage(\alpha) = LeakageModel(\alpha) + \mathcal{N}(0, 4^2)$$

Experimental Setup: REA with independent score lists

- DK-REA uses a REA for independent score lists. All previous REA algorithms can be used here.
- In this paper, we use the key REA based on histograms for its simplicity and efficiency.

Algorithm 2 Rank estimation algorithm based histograms [9]

Input: The score lists and the histograms H_i

Output: An approximation of the rank of the correct key.

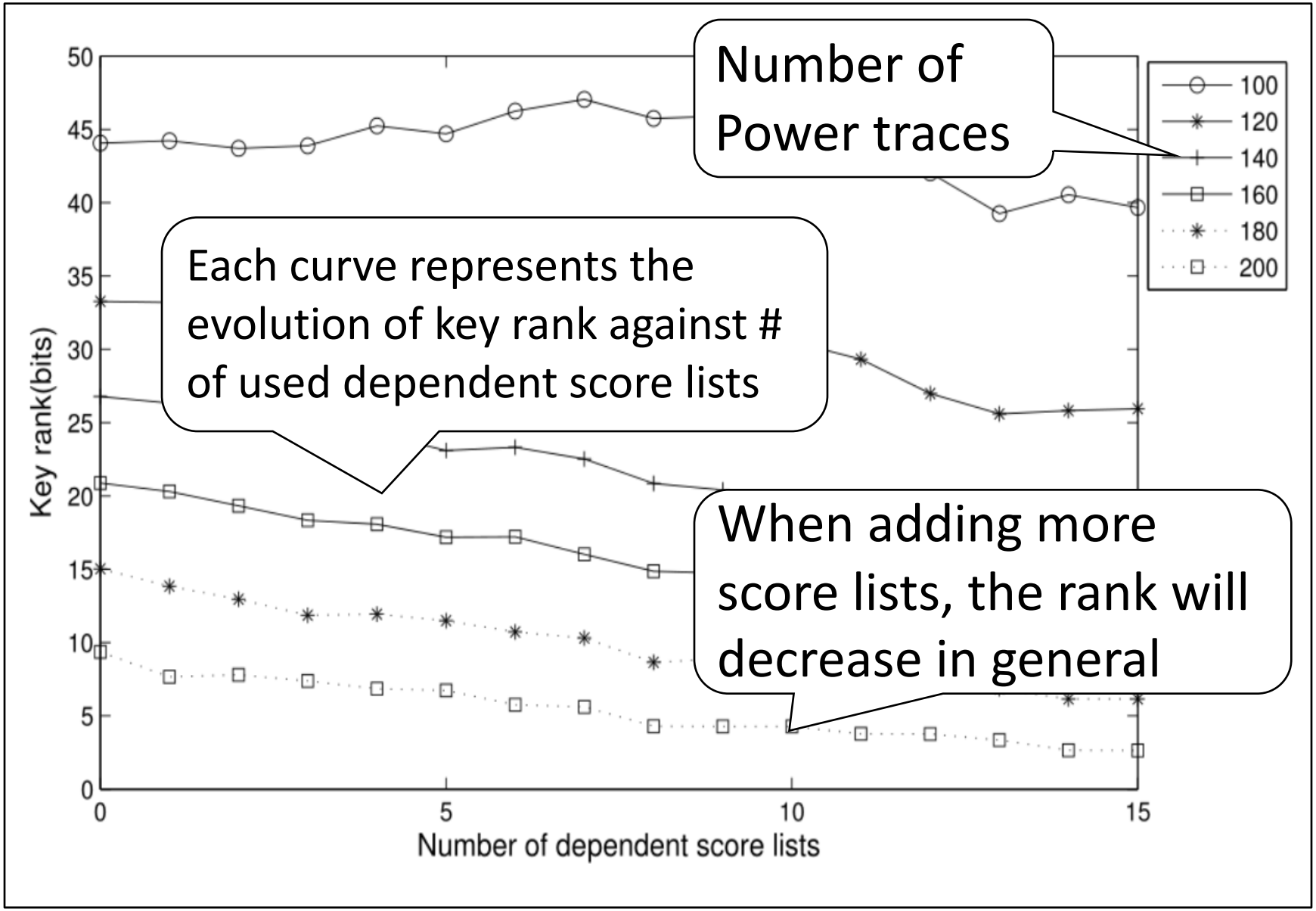
1: Initialization:$H_{curr} = H_1$;
2: Histograms convolution:
3: **for** $i = 2 : N_p$ **do**
4: $\quad H_{curr} = conv(H_{curr}, H_i)$;
5: **end for**
6: Rank estimation:
7: $$Estimated\_rank \approx \sum_{i=bins(k)}^{N_p \cdot N_{bin} - (N_p - 1)} H_{curr}(i)$$

C. Glowacz, V. Grosso, R. Poussier, J. Schth, and F. X. Standaert, *Simpler and More Efficient Rank Estimation for Side-Channel Security Assessment*. Springer Berlin Heidelberg, FSE 2015.

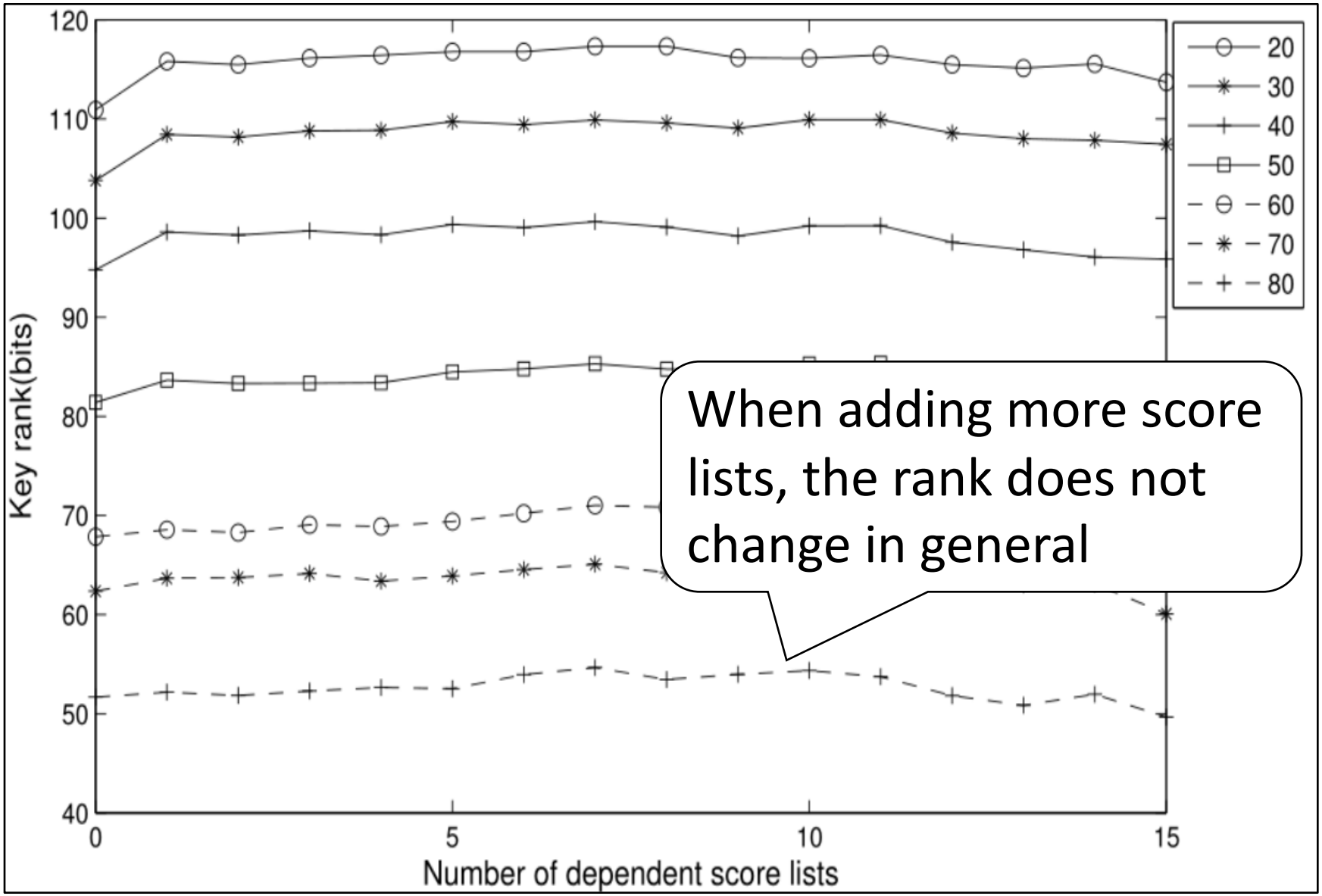
Experimental Setup: repetitions

- We performed
- 100 times experiments for adding 0 to 15 subkey-difference score lists
- 10 times experiments for adding 0 to 29 subkey-difference score lists.
- Taking into account the constraints of time and space, we do not consider the key rank estimation using more than 29 subkey-difference score lists

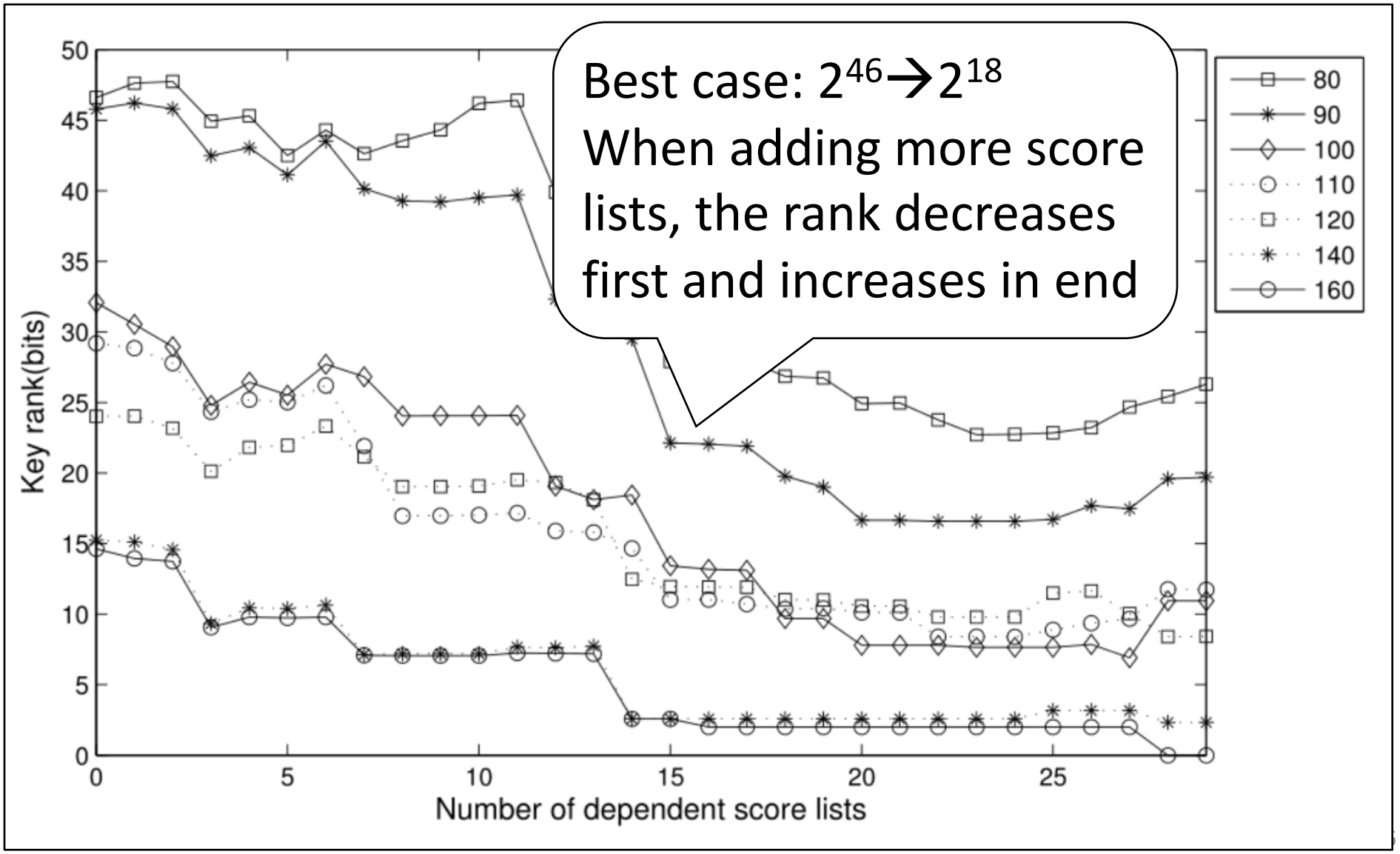
DK-REA up to 15 lists (100+ traces)



DK-REA up to 15 lists (<100 Traces)



DK-REA up to 29 lists



Outline

- 1. Background
  - Side-channel attacks, divide and conquer
  - Key enumeration and key rank estimation
- 2. Motivation & Problems Setting
  - Key recovery vs key difference recovery
  - Dependent key list
- 3. Our Solution
  - Basic idea
  - Dependent-Key List Rank Estimation Algorithm (DK-REA)
- 4. Experimental Verification
  - Setup & Results & Analysis
- 5. Conclusions and Future Work

Conclusions

- In this paper, we proposed DK-REA
  - Able to estimate key rank for dependent score lists
  - Generally the attack result is improved with more lists
    - Key rank reduced from 2^{46} to 2^{18} in the best case
  - Improvement also has limitations
    - When data is not enough
    - When too many dependent lists are added
- DK-REA is
  - A new tool in SCA to explore the possibility of merging various attack results for a more accurate security evaluation result

Future work

- More experiments to verify the effects of DK-KEA
  - With practical power traces
  - With different structures
    - e.g. serial and parallel implementations
  - With different leakage models
    - e.g. two attacks with large difference in key recovery efficiency
- DK-KEA for more complex dependency relations such as algebraic side-channel attack
- Reduce overhead!
- Rank Estimation Alg. → Key Enumeration Alg.
  - Our work to be presented at CARDIS 2017 in this Nov.

Thanks very much for your attentions!

Questions?

li.yang@nuaa.edu.cn
li.yangheu@gmail.com