# Multiple-Layer Candidate Sieves against Serial Cryptographic Implementations*

Changhai Ou[1,2], Fan Zhang[3(✉),4], Xinping Zhou[5], Kexin Qiao[6], and Renjun Zhang[7]

[1] School of Cyber Science & Engineering, Wuhan University, Wuhan, China
ouchanghai@whu.edu.cn
[2] State Key Laboratory of Cryptology, P.O.Box 5159, Beijing, China
[3] School of Cyber Science and Technology, College of Computer Science and Technology, Zhejiang University, Hangzhou, China
fanzhang@zju.edu.cn
[4] Key Laboratory of Blockchain and Cyberspace Governance of Zhejiang Province, Hangzhou, China
[5] Institute of Information Engineering, Chinese Academy of Sciences, China
zhouxinping@iie.ac.cn
[6] School of Cyber Science & Engineering, Beijing Institute of Technology, China
qiao.kexin@bit.edu.cn
[7] School of Mathematics, Hangzhou Normal University, China
zhangrenjun@hznu.edu.cn

**Abstract**

The existing multiple-layer candidate sieve exploits collisions to filter the candidates to achieve a much smaller space for easier key recovery, and tries to recover the key ranking at very deep candidate space. However, it leads to enormous computation yet achieves very low success probability. In this paper, we build a novel Simple Multiple-Layer Sieve (SMLS) from Correlation Power Analysis (CPA) and achieve better performance than the existing one. Furthermore, we build two combined sieves named Two-Layer Stacking Sieve (TLSS) and Full-Layer Stacking Sieve (FLSS) since same operations in serial cryptographic implementation generate similar leakage. The experimental results verify their superiority.

## 1 Introduction

Traditional side-channel attacks can be divided into divide-and-conquer and analytical. Divide-and-conquer attacks like Correlation Power Analysis (CPA) [3], are easy to perform. Analytical attacks such as Correlation-Enhanced Collision Attack (CECA) [5], are more complex but exploit more leaky information. The sub-keys are not always the best candidates in the actual attacks. Therefore, the divide-and-conquer distinguishers are usually combined with key enumeration [7]. However, due to the limited computing power, the enumerable candidate space is very small.

### 1.1 Related Works

Wiemers et al. built a multiple-layer candidate sieve from CECA against AES-128 in [10], and started a meaningful exploration of key recovery from very deep candidate space. They kept the current $w$ best combined candidates with the largest cumulative collision correlation coefficients and only considered them when considering the next sub-key. In this case, most

*Corresponding author: Fan Zhang.

of combined candidates were discarded, and they could recover the key from a much smaller remaining space. It is worth noting that this sieve is based on the assumption that they only have $120 \cdot 256$ collision correlation coefficients output by CECA, without any information from divide-and-conquer distinguishers. In this case, we cannot get sub-keys directly from CECA, but an XORed value $\delta_{i,j} = k_i \oplus k_j$ between two sub-keys $k_i$ and $k_j$. Taking the window size $w = 256$ as an example, only the $\delta_{1,2} = k_1 \oplus k_2$ with the largest collision correlation coefficient can be considered when we guess the first 2 sub-keys, since it includes 256 combined candidates of $k_1$ and $k_2$ satisfying the collision condition. The performance of CECA is also worse than CPA. However, this is not a surprise. Since the main purpose of CECA is to attack flawed masking implementations (e.g., DPA *contest* $v4.1$ [2]). In this case, we only require plaintexts rather than intermediate values (e.g., the outputs of S-boxes in AES-128).

Compared with collision attacks like CECA, the Double Sub-keys Recovery scheme proposed by Zhou et al. in [11] achieves better performance. For simplicity, we use DSR to denote it. It is based on the fact that the look-up table operations in the serially implemented cryptographic algorithms are the same and they generate similar leakage. It stacks the leaky samples of two S-boxes together, calculates the correlation coefficients under all possible combined candidates of these two sub-keys. Finally, it returns the best combined candidate with the highest correlation coefficient. On one hand, the stacked samples are equal to twice the number of samples we get, which greatly improves the Success Rate [9] of the attacks. On the other hand, DSR returns two specific values rather than a XORed value $\delta$, thus making its candidate space smaller than CECA. However, this also means that DSR needs to consider all possible combinations of them when merging $n$ sub-keys. Taking AES-128 as an example, the complexity of recovering 5 sub-keys simultaneously reaches to $2^{8 \cdot 5}$. Therefore, it is infeasible to conquer more sub-keys simultaneously by exploiting DSR.

## 1.2 Our Contributions

To improve the above mentioned disadvantages, we exploit the information between sub-keys by using DSR, and combine it with CPA to make it suitable for multiple-layer candidate sieve. We then build three novel multiple-layer candidate sieves named Simple Multiple-Layer Sieve (SMLS), Two-Layer Stacking Sieve (TLMS) and Full-Layer Stacking Sieve (FLSS) from them. They achieve significantly higher success rate than the existing scheme. Except for SMLS built from single distinguisher CPA, our TLSS and FLSS are combined attacks built from both DSR and CPA. Here DSR is not a traditional divide-and-conquer distinguisher, but is more like an analytical one. It makes TLSS and FLSS exploit information between sub-keys to reduce their number of possible combined candidates, so as to avoid exhaustion and obtain better key ranking than CPA. All of these bring TLSS and FLSS new significance, and make their further study important and essential.

## 1.3 Organization

The rest of this paper is organized as follows. Experimental setups, CPA, and CECA based Multiple-layer candidate sieve are given in Section 2. Pre-processing before stacking attacks, MLSS, and TLSS are introduced in Section 3. Our Stacking attack and optimized correlation computation are detailed in Section 4. The experimental results are given in Section 5 to show the superiority of our schemes. Finally, Section 6 concludes this paper.

# 2 Preliminaries

## 2.1 Experimental Setups

Our first experiment is performed on an ATMega328p micro-controller with clock operating frequency of 16 MHz. The unprotected AES-128 algorithm provided by [1] is implemented on it. We encrypt 1 000 000 plaintexts and exploit a WaveRunner 8104 oscilloscope to acquire the power traces. The sampling rate is set to 1 GS/s. We perform classic CPA on 10 000 power traces to extract the Points-Of-Interest (POIs) [4] in the first round and get 16 POIs with correlation coefficients between 0.35 and 0.59 for further analysis. These POIs are not strictly aligned since the implementation of AES-128 in [1] does not allow this.

Our second experiment is performed on an AT89S52 micro-controller specially designed for side-channel attacks having clock operating frequency of 12 MHz. The shortest instructions take 12 clock cycles for their execution. We implement the AES-128 algorithm exploiting assembly language, and use a *Tektronix DPO 7254* oscilloscope to capture leakage. All look-up table operations are finished by instruction "*MOVC A,@A+DPTR*", which takes 24 clock cycles. Here the register "DPTR" saves the starting address of S-box, another register "A" saves the offset and the result of look-up table is saved back to "A". The sampling rate of oscilloscope is set to 500 MS/s and 100 000 traces are acquired for attacks.

## 2.2 Correlation Power Analysis

The classic Correlation Power Analysis(CPA) obtains the key-related information by exploiting the correlation between the hypothesis power consumption of the intermediate values (e.g., Hamming weights) and the real leakage. We only consider AES-128 in this paper. Let $\mathcal{P}_i = (p_i[1], p_i[2], \ldots, p_i[n])$ denote the $n$ encrypted plaintext byte values corresponding to the $i$-th sub-key, $\mathcal{H}_i^{gk} = \left(h_i^{gk}[1], h_i^{gk}[2], \ldots, h_i^{gk}[n]\right)$ denote the $n$ Hamming weights of the $i$-th S-box outputs in the first round under a guess $gk$, and $\mathcal{T}_i = (t_i[1], t_i[2], \ldots, t_i[n])$ denote the corresponding leaky samples of a POI. They are all one-dimensional arrays. The corresponding correlation coefficient between them under a guess $gk$ can be expressed as:

$$\mathcal{R}_i^{gk} = \frac{\mathsf{Cov}\left(\mathcal{H}_i^{gk}, \mathcal{T}_i\right)}{\sigma\left(\mathcal{H}_i^{gk}\right) \cdot \sigma\left(\mathcal{T}_i\right)}. \tag{1}$$

Here $\mathsf{Cov}$ and $\sigma$ denote the covariance and standard deviation computation between two one-dimensional arrays, respectively. In this case, variance $\sigma^2(\mathcal{T}_i) = \frac{\sum_{j=1}^{n}(t_i[j]-\mu(\mathcal{T}_i))^2}{n-1}$ and mean $\mu(\mathcal{T}_i) = \sum_{j=1}^{n} t_i[j]$, so as to $\sigma\left(\mathcal{H}_i^{gk}\right)$. CPA returns the guessing value with the largest correlation coefficient and regards it as the best candidate of this sub-key.

## 2.3 Correlation-Enhanced Collision Attack

Correlation-Enhanced Collision Attack (CECA) [5] can be exploited to detect the collision values $\delta_{i,j}$ between any two sub-keys $k_i$ and $k_j$. Suppose that we perform CECA to detect the collision value between the first and second S-boxes in the first round of AES-128. CECA divides traces $\mathcal{T}_1$ and $\mathcal{T}_2$ with size of $n$ into 256 classes according to the values of their plaintext bytes, respectively. It then calculates the mean power consumption array $\mathcal{M}_j (j = 1, 2)$ of them,

and computes the correlation coefficient:

$$\mathcal{R}_{i,j}^{\alpha} = \rho\left\{\left(\mathcal{M}_1^{\alpha}, \mathcal{M}_2^{\alpha\oplus\delta}\right) \middle| \alpha = 0, 1, 2, \ldots, 255\right\}$$

between them under a guess $\delta = gk_1 \oplus gk_2$. Here $\alpha$ from 0 to 255 is the index of 256 means of 256 classes.

## 2.4 Double-Subkey Recovery

Zhou et al. gave the Double-Subkey Recovery scheme in [11]. For simplicity, we use DSR to denote it, which can be expressed as:

$$\mathcal{R}_{i,j}^{gk_1,gk_2} = \rho\left(\mathcal{H}_i^{gk_1} \cup \mathcal{H}_j^{gk_2}, \mathcal{T}_i \cup \mathcal{T}_j\right). \tag{2}$$

Here "$\cup$" denotes the stacking operation. For example, $\mathcal{T}_i \cup \mathcal{T}_j$ means we put the $n$ samples corresponding to $k_j$ on another $n$ samples corresponding to $k_i$, thus obtaining a new sample array with size of $2 \cdot n$. DSR travels all possible values of $k_i$ and $k_j$, and returns the best combined candidate of them.

We perform experiments on the power traces sampled from AT89S52 micro-controller, the Success Rates [9] of CPA and DSR are shown in Fig. 1. The success rate of DSR is significantly higher than that of CPA performed on both sub-keys, which verifies its effectiveness. The success rate of CPA performed on both two sub-keys simultaneously is about the product of the success rates of them. Although DSR is a single distinguisher improved from CPA, it is not strictly a divide-and-conquer one, but more similar to an analytical one. It can make better use of the information between sub-keys than CECA, which brings special significance to it.

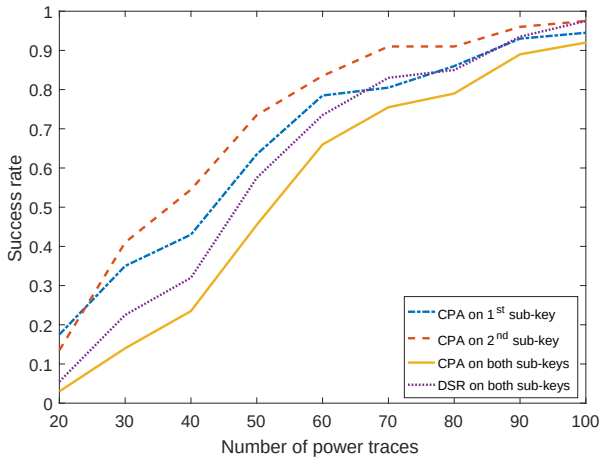


Figure 1: Success rate under different number of power traces.

## 2.5 CECA Based Multiple-Layer Sieve

Wiemers et al. built a multiple-layer candidate sieve from CECA in [10] based on the assumption that only $120 \cdot 256$ collision correlation coefficients of CECA are known. Specifically, they kept

a current window $B_{j-1}$ to save $w$ best combined candidates $(k_1, k_2, \ldots, k_{j-1})$ with the largest cumulative collision correlation coefficients. They computed

$$B_j = B_{j-1}\left((k_1, k_2, \ldots, k_{j-1})\right) + \sum_{i=1}^{j-1} \mathcal{R}_{i,j}^{k_i \oplus k_j} \tag{3}$$

for any combined candidate in window $B_{j-1}$ when considering the new sub-key $k_j$. The algorithm gets a temporary window with size of $256 \cdot w$ after traversing $k_j$ and $B_{j-1}$. It selects the new $w$ best combined candidates $(k_1, k_2, \ldots, k_j)$ for the next round of sieve.

# 3 New Multiple-Layer Candidate Sieves

## 3.1 Pre-processing

If the operations of S-boxes are the same, their power consumption will be similar. In this case, the linear correlation between the intermediate values of two S-boxes and their leakage can be combined directly. This corresponds to the power traces of our AT89S52 micro-controller. In this simple scenario, Eq. 2 can be directly exploited to perform CPA. However, serial AES implementations like [1], have different look-up table operations. It is difficult to align the power traces set of our ATmega328p micro-controller, and the power consumption of adjacent samples is also significantly different. Therefore, we standardize the samples:

$$\mathcal{T}_i' = \frac{\mathcal{T}_i - \mu\left(\mathcal{T}_i\right)}{\sigma\left(\mathcal{T}_i\right)} \tag{4}$$

before attacks. Here $\mu$ and $\sigma$ denote the mean and standard deviation computation on samples $\mathcal{T}_i$, and they satisfy the definitions given in Section 2.2. The experimental results show that we achieve better performance after the pre-processing.

## 3.2 Simple Multiple-Layer Sieve Built from CPA

We introduce CPA into multiple-layer sieve, thus obtaining the rank of each sub-key, avoiding inefficient random guessing and improving the success rate. We rank the candidates of sub-key $k_j$ $(1 \le j \le 16)$ according to their correlation coefficients in CPA and obtain $\mathcal{K}_j$. $\mathcal{R}_j^{\mathcal{K}_j^{gk}}$ is the correlation coefficient of its $gk$-th best candidate $(1 \le gk \le \tau_k)$. Here we calculate:

$$B_j = B_{j-1}\left((k_1, k_2, \ldots, k_{j-1})\right) + \mathcal{R}_j^{\mathcal{K}_j^{gk}} \tag{5}$$

for each of the $w$ optimal combined candidates $(k_1, k_2, \ldots, k_{j-1})$ with the largest cumulative correlation coefficients when considering the new sub-key $k_j$. Then, we select $w$ candidates from the current window $B_j$ having a total number of $w \cdot \tau_k$ candidates rather than $w \cdot 256$ when considering the next sub-key $k_{j+1}$. We name this multiple-layer candidate sieve as Simple Multiple-Layer Sieve (SMLS).

SMLS is similar to the key recovery algorithm given by Wiemers et al. (see Section 2.5). In fact, all candidate sieves can be expressed like Eqs. 3 and 5. In other words, the difference between them is, how much new cumulant (e.g. correlation coefficient) a combined candidate gets when a new sub-key is considered. The next round of sieve will be performed on the new cumulants, and the $w$ combined candidates with highest cumulants will be selected when considering the next round of sieve.

**Algorithm 1:** Simple Multiple-Layer Sieve(SMLS) built from CPA.

**Input:** Plaintexts $\mathcal{P}$, power traces $\mathcal{T}$ and thresholds $\tau_k$, $\tau_g$.
**Output:** The $\tau_g$ best combined candidates $\mathbb{K}$.

$(\mathcal{K}_1, \mathcal{R}_1) \leftarrow \mathsf{CPA}(\mathcal{P}_1, \mathcal{T}_1)$;
$\mathbb{K} = \mathcal{K}_1^{1\ldots\tau_k}; \mathbb{R} = \mathcal{R}_1^{1\ldots\tau_k}$;
**for** *i from 2 to 16* **do**
  $cn = 0$; $\mathbb{K}^{'} = \emptyset$; $\mathbb{R}^{'} = 0$;
  $(\mathcal{K}_i, \mathcal{R}_i) \leftarrow \mathsf{CPA}(\mathcal{P}_i, \mathcal{T}_i)$;
  **for** *j from 0 to* $\tau_k$ **do**
    **for** *t from 1 to* $\tau_g$ **do**
      $cn = cn + 1$;
      $\mathbb{R}^{'}_{cn} \leftarrow \mathbb{R}_t + \mathcal{R}_i^j$; $\mathbb{K}^{'}_{cn} \leftarrow \left[\mathbb{K}_t, \mathcal{K}_i^j\right]$;
    **end**
  **end**
  $(\mathbb{R}, \mathbb{K}) \leftarrow \mathsf{Select}\left(\mathbb{R}^{'}, \mathbb{K}^{'}, t_g\right)$;
**end**

### 3.3 Two-Layer Stacking Sieve

It is worth noting that SMLS is built from a single distinguisher CPA and not a stacking sieve, and the information between sub-keys is still "independent". This is similar to Template Attack (TA) [8], in which the probability that the guessing values of two sub-keys are correct is their probability product. We will combine two distinguishers DSR and CPA to perform more efficient attacks in this section. Due to the existence of DSR, the information between each two sub-keys is more well utilized.

For simplicity, stacking attack can be directly performed on the well computed Hamming weights $\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_{16}$. Stacking Hamming weights $\mathcal{H}_i^{k_i}$ of $k_i\,(1 \le i \le j-1)$ of a candidate $(k_1, k_2, \ldots, k_{j-1})$ in $B_{j-1}$ on Hamming weights $\mathcal{H}_j$ of $k_j$ under the $gk$-th guess ranked by CPA can be expressed as:

$$\mathcal{H}_{i,j}^{'} = \mathcal{H}_i^{k_i} \bigcup \mathcal{H}_j^{\mathcal{K}_j^{gk}}. \tag{6}$$

The corresponding cumulative correlation coefficient is:

$$\mathcal{R}_j^{\mathcal{K}_j^{gk}} = \sum_{i=1}^{j-1} \rho\left(\mathcal{H}_i^{k_i} \cup \mathcal{H}_j^{\mathcal{K}_j^{gk}}, \mathcal{T}_i \cup \mathcal{T}_j\right). \tag{7}$$

For simplicity, we name this MLS combining CPA with DSR built on each two sub-keys as Two-Layer Stacking Sieve (TLSS).

The results of TLSS on the first two and three sub-keys of AES-128 are given in Fig. 2. Here 50 power traces are randomly selected from power trace set of our ATmega328p micro-controller. We move all 30 candidates of $k_1$ within threshold $\tau_k = 30$ to $B_1$ to initialize the window, of which the size is $w = 30$ rather than $w = 4 \cdot 1 \cdot 30$. All the candidates of $k_2$ within $\tau_k$ are stacked with the candidates in this window. The stacking attacks often generate a region with high correlation coefficients if a candidate of them has a good linear relationship between its corresponding hypothesis Hamming weights and power consumption. The correct sub-key $k_{i+1}$ acts on $t_c = \tau_k + (\tau_k - 1) + \ldots + (\tau_k - i + 1)$ combined candidates, so the size of window $w$

**Algorithm 2:** Two-Layer Stacking(TLS) sieve.

**Input:** Assumed Hamming weights $\mathcal{H}_1, \ldots, \mathcal{H}_{16}$, power traces $\mathcal{T}$ and thresholds $\tau_k$, $\tau_g$.
**Output:** The best $\tau_g$ combined candidates $\mathbb{K}$.

$(\mathcal{K}_1, \mathcal{R}_1) \leftarrow \mathsf{CPA}(\mathcal{P}_1, \mathcal{T}_1)$;
$\mathbb{K} = \mathcal{K}_1; \mathbb{R} = \mathcal{R}_1$;
**for** *i from 2 to 16* **do**
  $cn = 0$; $\mathbb{K}^{'} = \emptyset$; $\mathbb{R}^{'} = 0$;
  **for** *gk from 0 to* $\tau_k$ **do**
    $\mathcal{H}^{'}[1 \ldots n] \leftarrow \mathcal{H}_i^{gk}$;
    **for** *gm from 1 to* $\tau_g$ **do**
      **for** *gh from 1 to* $i-1$ **do**
        $cn = cn + 1$;
        $\mathcal{H}^{'}[n+1 \ldots 2 \cdot n] \leftarrow \mathcal{H}_t^{\mathbb{K}_{gm}[gh]}$;
        $\rho \leftarrow \mathsf{Corr}\left(\mathcal{H}^{'}, \mathcal{T}^{'}\right)$;
        $\mathbb{R}_{cn}^{'} \leftarrow \mathbb{R}_t + \rho$; $\mathbb{K}_{cn}^{'} \leftarrow \left[\mathbb{K}_t, \mathcal{K}_i^{gk}\right]$;
      **end**
    **end**
  **end**
  $(\mathbb{R}, \mathbb{K}) \leftarrow \mathsf{Select}\left(\mathbb{R}^{'}, \mathbb{K}^{'}, t_g\right)$;
**end**

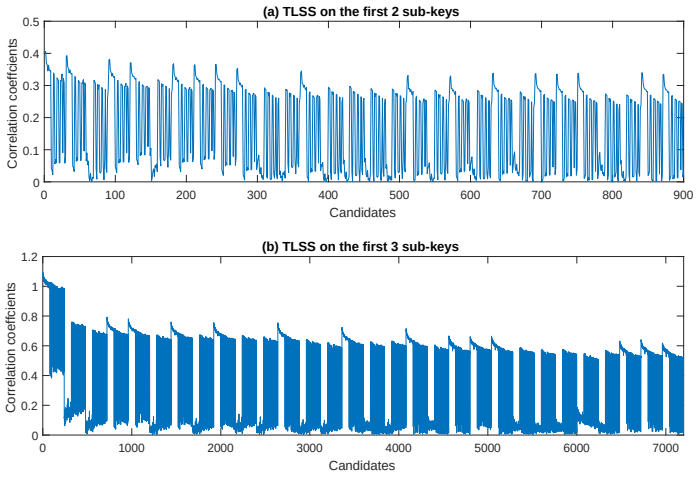


Figure 2: TLSS performed on the first 2 and 3 sub-keys with $\tau_k = 30$ and $w = 4 \cdot i \cdot \tau_k$ $(i \geq 2)$.

can be set close to this. However, our window works very well under $w = 4 \cdot i \cdot \tau_k$, and enlarging it to $w = 10 \cdot i \cdot \tau_k$ closer to $t_c$ will not significantly improve success rate. This illustrates that the correct combined candidate falls into the region with high correlation coefficients in

the window. In this case, to achieve higher success rate, we can set a larger $\tau_k$ to make more sub-keys fall within threshold.

# 4 Optimization

## 4.1 Full-Layer Stacking Sieve

Stacking attack has achieved good performance on DSR and TLSS. In this case, we can simply stack the power traces of all sub-keys together for attacks. This is obviously impossible for DSR when all key candidates are considered simultaneously. However, it is just feasible on the sieve which combines CPA and filtering operation. Moreover, the side-channel leakage provides us the key-relevant information. It can be recovered from the corresponding candidate space with high probability when $\tau_k$ is set reasonably.

Suppose that for a combined candidate $(k_1, k_2, \ldots, k_{j-1})$ with Hamming weights $\mathcal{H}_i^{k_i}$ $(1 \leq i \leq j-1)$ of their intermediate values, we can stack them together and obtain:

$$\mathcal{H}_{j-1}^{'} = \bigcup_{i=1}^{j-1} \mathcal{H}_i^{k_i}. \tag{8}$$

Then, we stack the current $(j-1) \cdot n$ Hamming weights on Hamming weights $\mathcal{H}_j^{\mathcal{K}_j^{gk}}$ corresponding to the $gk$-th most possible candidate of the $j$-th sub-key output by CPA, and obtain:

$$\mathcal{H}_j^{'} = \mathcal{H}_{j-1}^{'} \bigcup \mathcal{H}_j^{\mathcal{K}_j^{gk}}. \tag{9}$$

For power traces, we can simply stack them as:

$$\mathcal{T}_j^{'} = \bigcup_{i=1}^{j} \mathcal{T}_i. \tag{10}$$

The key recovery are then performed on $\mathcal{H}_j^{'}$ and $\mathcal{T}_j^{'}$. Compared with TLSS, this new sieve has extended DSR to all sub-keys together in a subtle way. Therefore, we name it as Full-Layer Stacking Sieve (FLSS). It is worth mentioning that this attack performance may be not good when there are too many sub-keys to stack. Since the wrongly guessed $j$-th sub-key will not significantly reduce the correlation coefficient when the former $j-1$ sub-keys are all correctly guessed. Fortunately, the number of sub-keys in ciphers is usually small.

We also perform FLSS on the first 2 and 3 sub-keys of the AES-128 implemented on our ATmega328p micro-controller, and 50 power traces are randomly selected from our power trace set. The results are shown in Fig. 3. Sieves FLSS and TLSS show a region with the highest correlation coefficients at the beginning of the graphs, and several peaks rather than only one peak appear on figures like CPA. This indicates that most of sub-keys become the best candidates in CPA attacks. Since the correlation coefficients of TLSS are cumulative, they can be larger than 1.0 in the third sub-key. However, the Hamming weights of a combined candidate are stacked together in FLSS, then Eq. 1 is performed between them and the stacked time samples. Therefore, the highest correlation coefficient is maintained at a certain level when the number of stacked samples is large enough (e.g., 0.48 as shown in Fig. 3(b)).

**Algorithm 3:** Full-Layer Stacking(FLS) sieve.

**Input:** Assumed Hamming weights $\mathcal{H}_1, \ldots, \mathcal{H}_{16}$, power traces $\mathcal{T}$ and thresholds $\tau_k$, $\tau_g$.
**Output:** The best $\tau_g$ combined candidates $\mathbb{K}$.

$\mathbb{K} = \mathcal{K}_1; \mathbb{R} = \mathcal{R}_1$;
$\mathcal{T}^{'}[1 \ldots n] \leftarrow \mathcal{T}_1$;
**for** *i from 2 to 16* **do**
  $cn = 0$; $\mathbb{K}^{'} = \emptyset$; $\mathbb{R}^{'} = 0$;
  $\mathcal{T}^{'}[n \cdot (i-1)+1 \ldots i \cdot n] \leftarrow \mathcal{T}_i$;
  **for** *gm from 0 to* $\tau_g$ **do**
    **for** *gh from 1 to* $i-1$ **do**
      $\mathcal{H}^{'}[n \cdot (gh-1)+1 \ldots gh \cdot n] \leftarrow \mathcal{H}_{gk}^{\mathbb{K}_{gm}[gh]}$;
    **end**
    **for** *gk from 1 to* $\tau_k$ **do**
      $\mathcal{H}^{'}[n \cdot (i-1)+1 \ldots i \cdot n] \leftarrow \mathcal{H}_i^{gk}$;
      $cn = cn + 1$;
      $\mathbb{R}_{cn}^{'} \leftarrow \mathsf{Corr}\left(\mathcal{H}^{'}, \mathcal{T}^{'}\right)$;
      $\mathbb{K}_{cn}^{'} \leftarrow \left[\mathbb{K}_t, \mathcal{K}_i^j\right]$;
    **end**
  **end**
  $(\mathbb{R}, \mathbb{K}) \leftarrow \mathsf{Select}\left(\mathbb{R}^{'}, \mathbb{K}^{'}, t_g\right)$;
**end**

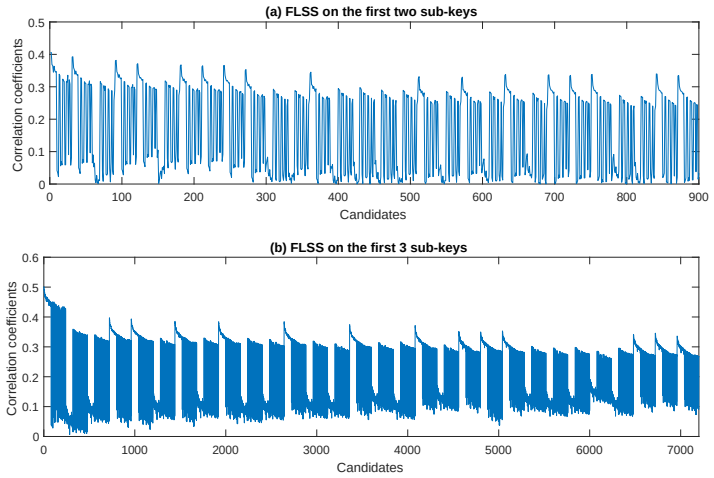


Figure 3: FLSS performed on the first 2 and 3 sub-keys with $\tau_k = 30$ and $w = 4 \cdot i \cdot \tau_k$ $(i \geq 2)$.

### 4.2 Optimization of Correlation Calculation

DSR and TLSS only stack power traces of two sub-keys in AES-128, while FLSS stacks power traces of all 16 sub-keys. It has a large amount of repeated calculation on correlation coefficients, and satisfies Eq. 1. The another expression of this equation is:

$$\mathcal{R} = \frac{\sum_{i=1}^{n} \left(h\left[i\right] - \bar{h}\right)\left(t\left[i\right] - \bar{t}\right)}{\sqrt{\sum_{i=1}^{n} \left(h\left[i\right] - \bar{h}\right)^2} \cdot \sqrt{\sum_{i=1}^{n} \left(t\left[i\right] - \bar{t}\right)^2}}. \tag{11}$$

Here $\bar{h} = \mu\left(\mathcal{H}\right)$ and $\bar{t} = \mu\left(\mathcal{T}\right)$ as given in Section 2.2. Actually, Eq. 11 can be further expressed as:

$$\rho = \frac{n\sum \mathcal{H} \cdot \mathcal{T} - \sum \mathcal{H} \cdot \sum \mathcal{T}}{\sqrt{n\sum \mathcal{H}^2 - \left(\sum \mathcal{H}\right)^2} \cdot \sqrt{n\sum \mathcal{T}^2 - \left(\sum \mathcal{T}\right)^2}}. \tag{12}$$

Here $\sum \mathcal{H} = \sum_{i=1}^{n} h\left[i\right]$ and $\sum \mathcal{T} = \sum_{i=1}^{n} t\left[i\right]$ are the sum of Hamming weights and samples respectively, $\sum \mathcal{H}^2 = \sum_{i=1}^{n} h^2\left[i\right]$ and $\sum \mathcal{T}^2 = \sum_{i=1}^{n} t^2\left[i\right]$ are the corresponding sum of squares, and $\sum \mathcal{H} \cdot \mathcal{T} = \sum_{i=1}^{n} h\left[i\right] \cdot t\left[i\right]$. In this case, we only need to define 5 arrays to save these 5 variables for all combined candidates in current window $B_{j-1}$ in our implementation, compute their new cumulants when considering a new sub-key and update them when computing the new correlation coefficient. This improves the speed of FLSS in our experiments by 6 to 10 times.

## 5 Experimental Results

### 5.1 Experimental Results on ATmega328p Micro-controller

Our first experiment is performed on ATmega328p micro-controller. Since the candidate space under $\tau_k$ is $\tau_k^i$ when considering a total of $i$ sub-keys, we set an independent window size $w = 4 \cdot i \cdot \tau_k$ for the $i$-th sub-key. The experimental results under different number of power traces are shown in Fig. 4. Each experiment is repeated 200 times. The success rate of the multiple-layer sieve built from CECA by Wiemers et al. drops to 0 after the 5-th sub-key even after the combination with CPA. It achieves good performance under the assumption that only $120 \cdot 256$ coefficients output by CECA are known, but not works very well in our attack scenario. FLSS gets higher success rate than SMLS and TLSS. The mean time consumption of CECA based MLS given by Wiemers et al. combined with CPA, SMLS, FLSS and TLSS is about 1.43, 1.47, 16.22 and 15.81 seconds, respectively. These fully illustrate the superiority of FLSS after optimization. Since enlarging $w$ rather than $\tau_k$ will not significantly improve success rate as explained in Section 3.3, we only consider $w = 4 \cdot i \cdot \tau_k$ in this paper. To improve the success rate, further improving $\tau_k$ to make more sub-keys fall into threshold is a good choice.

The success rates shown in Fig. 4 are zigzag, and they suddenly drop down, which vividly shows that the consideration of some sub-keys significantly affects the attack performance. However, the performance of CPA of SMLS also declines, which illustrates that the decline of success rates is not caused by stacking attacks, but by the noise of the selected POIs of the sub-keys. Moreover, the success rates of 4 schemes are very close when the number of stacked sub-keys is no more than 8 in Figs. 4(b), 4(c) and 4(d). This also gives us a hint: the first round-key of AES-128 can be divided into several big "blocks", of which each includes one or several sub-keys; stacking attacks can then be performed on them respectively to obtain a higher success rate.

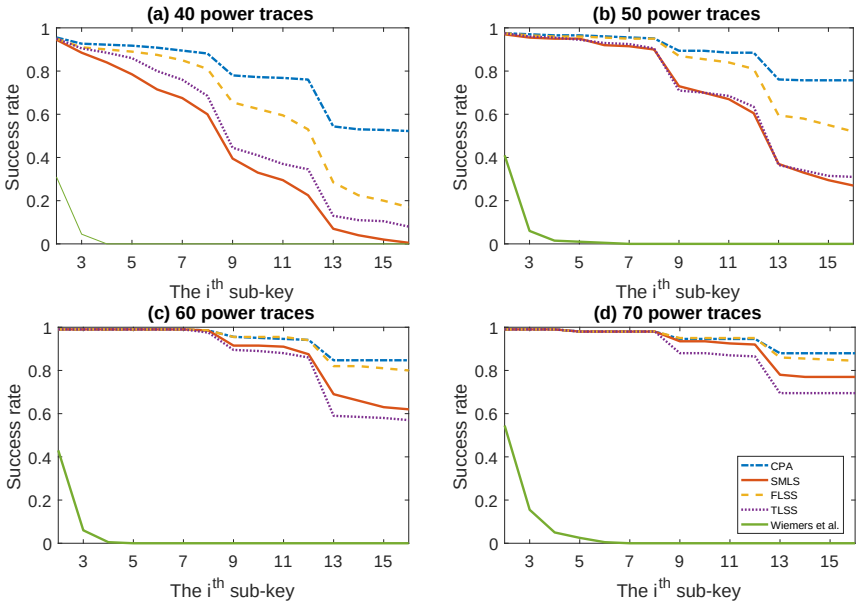


Figure 4: Success rate under different number of power traces.

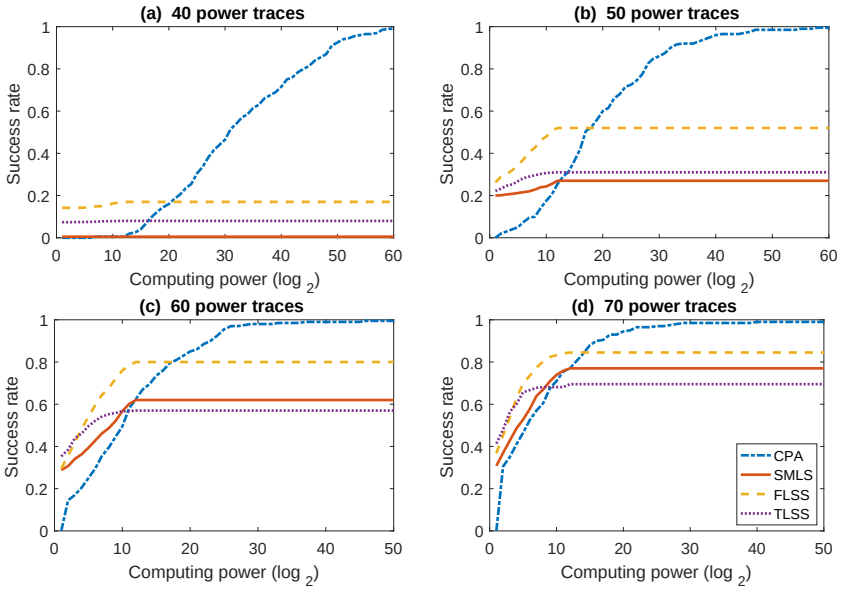


Figure 5: Success rate different under computing power.

The histogram based key rank estimation given in [7] is also exploited here, and the success rate of CPA under different computing power is given in Fig. 5. The corresponding evaluation was also exploited in [6]. For example, the value 30 on the horizontal ordinate indicates that the key is ranked at $2^{30}$, we have to enumerate $2^{30}-1$ candidates before recovering it. Since the outputs of SMLS, TLSS and FLSS are combined candidates including 16 sub-keys unlike CPA, we also exploit rank of the cumulative correlation coefficients to indicate the number of combined candidates we need to search before recovering the key. It can also be seen from Fig. 5 that the curves of TLSS and FLSS are farther from CPA than the one corresponding to SMLS, which indicates that DSR provides additional key-related information in stacking attacks. Moreover, TLSS and FLSS are combined attacks unlike SMLS, and still maintain high success rates under very small window size $w = 4 \cdot i \cdot \tau_k$, which fully illustrates their effectiveness.

## 5.2 Experimental Results on AT89S52 Micro-controller

The experimental results performed on the power trace set of AT89S52 micro-controller are given in Fig. 6. Since the AES-128 programmed by assembly language implemented on the chip operates exactly the same for each S-box, the power traces can be strictly aligned and the POIs of 16 S-boxes have the same leaky characteristics. This makes the success rate in Fig. 6 smooth, and there is no zigzag and sudden drop on them. The success rates of SMLS, TLSS and FLSS are close to that of CPA with the increasing number of power traces, and the performance of TLSS is almost better than that of SMLS. However, its performance is better than SMLS when $n$ is small, and worse than SMLS when $n$ is large in Fig. 4. This also highlights that the stacking attacks can better exploit the information between the sub-keys if the operations of all S-boxes are the same and their power traces can be strictly aligned.

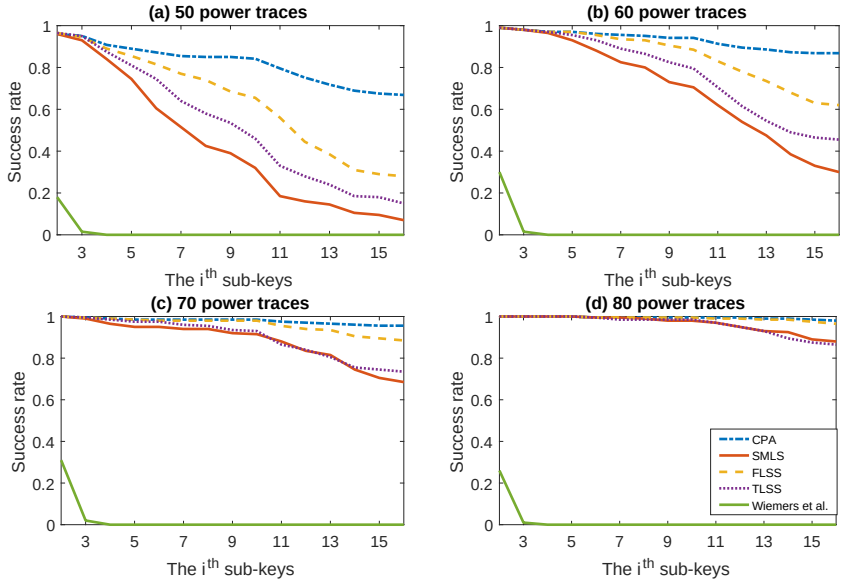


Figure 6: Success rate under different number of power traces.

Similar conclusion can be drawn from Fig. 6 that the efficiency of the MLS built from

CECA by Wiemers et al. is much lower than our SMLS, TLSS and FLSS even after combining with CPA. The mean time consumption of them is about 1.01, 1.04, 13.21 and 11.17 seconds respectively, similar to time consumed by the experiments given in Section 5.1. The success rates of SMLS, TLSS and FLSS decrease faster when the number of stacked sub-keys is more than 8. It is worth noting that, since the sieves SMLS, TLSS and FLSS only consider the candidates within threshold $\tau_k$ provided by CPA, their success rates in Figs. 4 and 6 are lower than that of CPA, unlike Fig. 1.

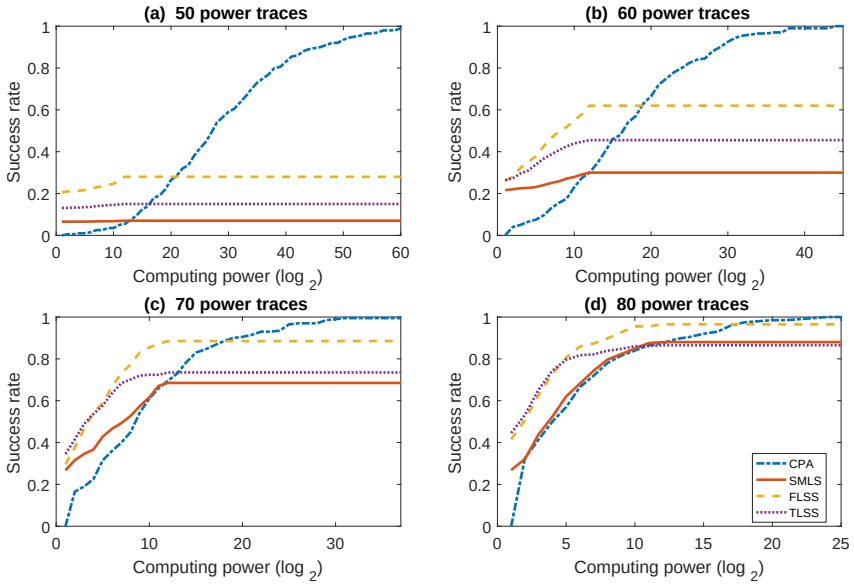


Figure 7: Success rate under different computing power.

The increasing number of power traces enables the attacks to recover the key with lower computing power (as shown in Fig. 7). The stacking sieves TLSS and FLSS achieve significantly higher success rates than non-stacking sieve SMLS, and overlap at the beginning of Figs. 7(b), 7(c) and 7(d). This indicates that they exploit DSR to get the information between sub-keys, so as to eliminate a large number of wrong combined candidates and reduce the number of possible combinations (i.e. guessing space). SMLS and CPA have partial overlap, and this grows when more power traces are used. This is because that SMLS is built from CPA, and they have the same key ranking. In this case, discarding combined candidates whose ranking is behind the correct key in SMLS will not affect the key ranking. This is especially when the number of power traces increases and $\tau_k = 30$ is not reduced correspondingly. That is to say, SMLS does not finally optimize the CPA when the threshold $\tau_k$ is large enough to make all sub-keys fall within it. This also illustrates the effectiveness of our TLSS and FLSS.

# 6 Conclusions

The existing multiple-layer candidate sieve exploits CECA to filter the candidates to achieve a much smaller space for easier key recovery. It is an important exploration to recover the key

ranked at very deep candidate space, and its research is of great significance. However, they lead to enormous computation yet achieve very low success probability as we have explained before. In this paper, we improve DSR and combine it with CPA, so as to make it exploitable for multiple-layer candidate sieves. We further propose a non-stacking sieve named SMLS and two stacking sieves named TLSS and FLSS. Experiments on AT89S52 and ATmega328p micro-controllers indicate that they achieve significantly better performance than the existing schemes.

Due to the introduction of information between sub-keys by DSR, a large number of wrong combined candidates are discarded by sieves. This makes the key rank more advanced and even significantly better than the one corresponding to CPA. Moreover, the success rates of our TLSS and FLSS are significantly improved. These two advantages bring new significance to our schemes. According to the principle of MLS, our SMLS, TLSS and FLSS can also be performed against parallel cryptographic implementations theoretically. We will take it and fault tolerance as our future works.

# 7 Acknowledgments

This work was supported in part by National Key R&D Program of China (2020AAA0107700), by National Science Foundation of China (62072398), by Alibaba-Zhejiang University Joint Institute of Frontier Technologies, by Zhejiang Key R&D Plan (2019C03133, 2021C01116).

# References

[1] Avr-crypto-lib. https://github.com/DavyLandman/AESLib.

[2] Dpa contest. http://www.dpacontest.org/home/.

[3] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation Power Analysis with a Leakage Model. In *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, pages 16–29, 2004.

[4] François Durvaux and François-Xavier Standaert. From Improved Leakage Detection to the Detection of Points of Interests in Leakage Traces. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, pages 240–262, 2016.

[5] Amir Moradi, Oliver Mischke, and Thomas Eisenbarth. Correlation-Enhanced Power Analysis Collision Attack. In *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, pages 125–139, 2010.

[6] Romain Poussier, Vincent Grosso, and François-Xavier Standaert. Comparing Approaches to Rank Estimation for Side-Channel Security Evaluations. In *Smart Card Research and Advanced Applications - 14th International Conference, CARDIS 2015, Bochum, Germany, November 4-6, 2015. Revised Selected Papers*, pages 125–142, 2015.

[7] Romain Poussier, François-Xavier Standaert, and Vincent Grosso. Simple Key Enumeration (and Rank Estimation) Using Histograms: An Integrated Approach. In *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, pages 61–81, 2016.

[8] Christian Rechberger and Elisabeth Oswald. Practical Template Attacks. In *Information Security Applications, 5th International Workshop, WISA 2004, Jeju Island, Korea, August 23-25, 2004, Revised Selected Papers*, pages 440–456, 2004.

[9] François-Xavier Standaert, Tal Malkin, and Moti Yung. A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, pages 443–461, 2009.

[10] Andreas Wiemers and Dominik Klein. Entropy Reduction for the Correlation-Enhanced Power Analysis Collision Attack. In *Advances in Information and Computer Security - 13th International Workshop on Security, IWSEC 2018, Sendai, Japan, September 3-5, 2018, Proceedings*, pages 51–67, 2018.

[11] Xinping Zhou, Degang Sun, Zhu Wang, Changhai Ou, and Juan Ai. Double-Key Recovery Based Correlation Power Analysis. In *2016 IEEE Trustcom/BigDataSE/ISPA, Tianjin, China, August 23-26, 2016*, pages 1016–1022, 2016.