My traces learn what you did in the dark: recovering secret signals without key guesses

Si Gao[1,2], Hua Chen[1]✉, Wenling Wu[1], Limin Fan[1], Weiqiong Cao[1,2], and Xiangliang Ma[1,2]

[1] Trusted Computing and Information Assurance Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing, P.R.China 100190
[2] University of Chinese Academy of Sciences, Beijing, P.R.China 100049
`{gaosi,chenhua,wwl,fanlimin,caowq,maxiangliang}@tca.iscas.ac.cn`

Abstract. In side channel attack (SCA) studies, it is widely believed that unprotected implementations leak information about the intermediate states of the internal cryptographic process. However, directly recovering the intermediate states is not common practice in today's S-CA study. Instead, most SCAs exploit the leakages in a "guess-and-determine" way, where they take a partial key guess, compute the corresponding intermediate states, then try to identify which one fits the observed leakages better. In this paper, we ask whether it is possible to take the other way around—directly learning the intermediate states from the side channel leakages. Under certain circumstances, we find that the intermediate states can be efficiently recovered with the well-studied Independent Component Analysis (ICA). Specifically, we propose several methods to convert the side channel leakages into effective ICA observations. For more robust recovery, we also present a specialized ICA algorithm which exploits the specific features of circuit signals. Experiments confirm the validity of our analysis in various circumstances, where most intermediate states can be correctly recovered with only a few hundred traces. To our knowledge, this is the first attempt to directly recover the intermediate states in a completely non-profiled setting. Our approach brings new possibilities to the current SCA study, including building an alternative SCA distinguisher, directly attacking the middle encryption rounds and reverse engineering with fewer restrictions. Considering its potential in more advanced applications, we believe our ICA-based SCA deserves more research attention in the future study.

Keywords: **Side Channel Analysis**, **Signal Recovery**, **Independent Component Analysis**

1 Introduction

Nowadays, Side Channel Attacks (SCA) pose a major threat for various cryptographic devices [1,2,3,4,5]. With some data-dependent leakage measurements (such as power supply currents or electromagnetic emissions), an SCA attacker can efficiently retrieve the secret key, even if the underlying cipher is cryptographically strong.

In a typical SCA context (illustrated in Figure 1), the attacker Eve encrypts T plaintexts and measures the corresponding leakages $\mathbf{L}$. In SCA, it is widely believed that $\mathbf{L}$ depends on some key-related intermediate states, denoted as $\mathbf{x}_k = \{x_k(1), ..., x_k(T)\}$. With certain key guess k_i, Eve computes the intermediate state sequence $\mathbf{x}_{k_i} = \{x_{k_i}(1), ..., x_{k_i}(T)\}$ according to the encryption algorithm. Throughout this paper, we denote this sequence as a *signal*. Since the leakages $\mathbf{L}$ only depend on the correct signal $\mathbf{x}_k$, Eve combines all possible $\mathbf{x}_{k_i}$ with $\mathbf{L}$ and learns the most likely key guess.

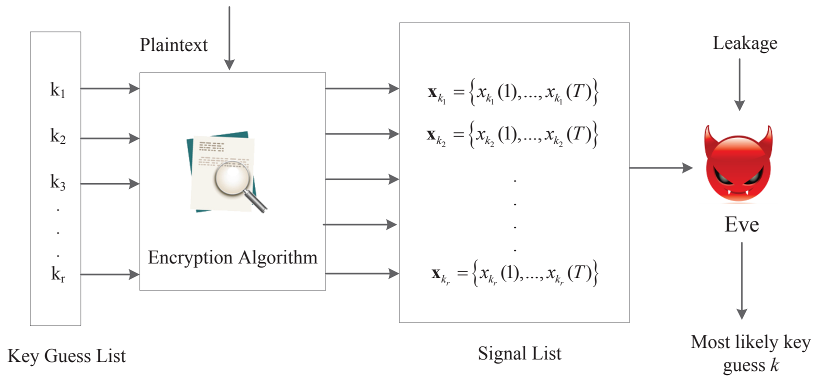
Plaintext
k_1
k_2
k_3
k_r
Encryption Algorithm
$\mathbf{x}_{k_1} = \{x_{k_1}(1), ..., x_{k_1}(T)\}$
$\mathbf{x}_{k_2} = \{x_{k_2}(1), ..., x_{k_2}(T)\}$
$\mathbf{x}_{k_r} = \{x_{k_r}(1), ..., x_{k_r}(T)\}$
Leakage
Eve
Key Guess List
Signal List
Most likely key guess k

Fig. 1. A typical SCA procedure

In Figure 1, Eve has a list of all possible intermediate state sequences (signals) and tries to find the correct one with the corresponding leakages. In SCA studies, such procedure is called a *side channel distinguisher*. The term *distinguisher* demonstrates its inherent limitation: such process only distinguishes the correct signal from the wrong ones, yet never directly retrieve any secret. A natural question to ask, is whether Eve can take one step further and directly learn the correct signal from the leakages, without an enumerative signal list.

It is not surprising that little SCA study answers this question. As the master key is the only secret in modern block ciphers, the "key-distingshers" above already present enough threat for unprotected chips. Nonetheless, recovering the intermediate states without a key guess may still be helpful in certain circumstances. For instance, in a typical SCA procedure, the computation cost strictly depends on the correlated key size. If the target intermediate state involves a large proportion of the secret key (> 32 bits), enumerating all $\mathbf{x}_{k_i}$ becomes infeasible. Other examples include Side Channel Analysis for Reverse Engineering (SCARE), where the secret components baffle the computation of $\mathbf{x}_{k_i}$. Since the signal list cannot be efficiently computed, SCA in these cases needs a more general secret recovery technique.

Related work. Despite a few *ad hoc* SPA-like attacks, in general, most previous non-profiled SCAs cannot directly recover the intermediate states. Collision attack may be the closest match in this direction. In a collision attack, the attacker collects a few collisions and solves the intermediate states from the collision equations [6]. As a prevalent tool in SCARE [7,8,9], collision attack exploits the similarity between the leakages from sequential computations of the same Sbox. If the measurements from two Sbox computations match, we can reasonably predict they share the same input (a.k.a a *collision*). Although marked as a non-profiled attack, collision attacks share exactly the same routine as *Template Attacks* (TA) [10]. The only difference lies in the profiling stage, where collision attacks use other sequentially-implemented Sbox computations as the profiling trace set [6]. Since the leakages of the exact same Sbox computation are not always available, such "online-profiling" stage imposes restrictions on the implementations as well as the target ciphers. Indeed, none of the previous collision-based SCAREs gave realistic experiments to validate their results [7,8,9].

Our Contribution. In this paper, we present the first attempt to recover the secret intermediate states directly from the observed leakages. Our analysis shows that, under certain circumstances, recovering the intermediate states can be regarded as a noisy Blind Source Separation (BSS) problem[1] [12]. Following the study of BSS, we introduce the well-studied Independent Component Analysis (ICA) [13] to SCA. In signal processing, ICA is a widely used tool for recovering unknown sources in a blind context. Considering ICA takes at least n observations to recover n sources, we propose several methods to construct multi-channel observations from the side channel leakages. Moreover, since typical ICA algorithms are sensitive to noise, we present a more robust ICA algorithm based on Belouchrani and Cardoso's work on discrete ICA [14]. By exploiting the specific features of circuit signals, our specialized ICA gives efficient recoveries in the SCA context. The overall analysis works well in our realistic experiments, recovering over 80% of the intermediate states correctly, with only a few hundred traces. Furthermore, our ICA-based SCA brings several new possibilities to the current non-profiled SCA studies. In our experiments, our ICA-based SCA helps to improve the key-recovery result in a limited trace set, extend SCA to the middle encryption rounds as well as loosen the restrictions of current SCAREs. As a potentially powerful tool, we believe our ICA-based SCA deserves more research attention in the future.

Paper Organization. In the next section, we present a brief introduction of Independent Component Analysis, discussing its assumptions as well as limitations. Section 3 shows how to convert the intermediate state recovery to an ICA problem. Specifically, we propose several methods to construct multi-channel observations from the side channel leakages and build a specialized ICA algorithm for circuit signals. Section 4 demonstrates several advanced applications

[1] Although our work uses the concept of *Blind Source Separation*, it has little connection with Merino Del Pozo and Standaert's work [11]. Their paper focuses on reducing noise in the preprocessing stage, rather than recovering any secret.

of our ICA-based SCA. With realistic measurements of an unprotected software implementation of DES, we confirm the validity of our approach. Further discussions about our ICA-based SCA and its promising prospects in the future are presented in Section 5.

2 Independent Component Analysis

2.1 Definition

Independent Component Analysis (ICA) [13] belongs to a border class of problems called *Blind Source Separation* (BSS) [12], which requires to separate a set of mixed signals, without the aid of information about the source signals or the mixing process. A common example is the *cocktail party problem*, which suggests a partygoer can focus on a single conversation in a noisy room.

Suppose we have n simultaneous conversations (sources) $\mathbf{S} = \{s_1, s_2, ..., s_n\}$ going on in the party room. Microphones are placed in different positions, recording m mixtures (observations) of the original sources $\mathbf{Y} = \{y_1, y_2, ..., y_m\}$[2]. Assuming the observation y_j is a linear mixture of all sources, we have

$$y_j = a_{j,1}s_1 + a_{j,2}s_2 + ... + a_{j,n}s_n$$

where $a_{j,i}$ stands for the real-valued coefficient. The overall mixing procedure can be written as

$$\mathbf{Y} = \mathbf{AS}$$

where $\mathbf{A}$ is called the *mixing matrix.* In signal processing, such statistical model is called *Independent Component Analysis* [13]. With additional multivariate Gaussian noise $\mathbf{N}$, the noisy ICA model is defined as

$$\mathbf{Y} = \mathbf{AS} + \mathbf{N}$$

2.2 Assumptions and Ambiguities

Since both $\mathbf{S}$ and $\mathbf{A}$ are not given, in general, the BSS problem is highly underdetermined. In order to find useful solutions, BSS usually requires some additional assumptions. Specifically, ICA relies on the following assumptions [13]:

- **Independence**: The source signals are independent of each other.
- **Non-Gaussianity**: The values in each source signal have non-Gaussian distributions.

In addition, typical ICA algorithms also assume the number of observations is no less than the number of sources ($m \geq n$) [13]. Like most noisy statistical models, noise significantly affects the effectiveness of ICA.

From the ICA model, it is not hard to see the following ambiguities hold [13]:

[2] Although ICA studies usually use $\mathbf{X}$ to denote the observations, considering $\mathbf{X}$ is already occupied by the intermediate state, here we use $\mathbf{Y}$ instead.

- **ICA cannot determine the amplitude[3] of the sources.** As both $\mathbf{S}$ and $\mathbf{A}$ are unknown, any scalar multiplier in s_i can always be cancelled by dividing the corresponding column of $\mathbf{A}$ with the same scalar.
- **ICA cannot determine the order of the sources.** As both $\mathbf{S}$ and $\mathbf{A}$ are unknown, we can freely choose a permutation matrix $\mathbf{P}$. In the ICA model, $\mathbf{Y} = \mathbf{AP^{-1}PS}$, where $\mathbf{A}' = \mathbf{AP^{-1}}$ and $\mathbf{S}' = \mathbf{PS}$ is also a valid solution.

In addition, in ICA, the input signal usually enters a *whitening transformation* before any further analysis. Theoretically speaking, a *whitening transformation* is a decorrelation transformation that transforms a set of random variables into a set of new random variables with zero means and identity covariance. As a result, ICA only returns the estimates of the "whitened sources". In other words, the means of the original sources cannot be determined through ICA. For typical ICA applications like separating independent speeches, none of these ambiguities presents an obstacle in practice.

2.3 Principles and Algorithms

Generally speaking, ICA aims to find the independent components by maximizing the statistical independence of the estimated components. According to the Central Limit Theorem, the distribution of the sum of independent random variables with finite variance tends towards a Gaussian distribution. For ICA, this means compared with the sources $\mathbf{S}$, each observation y_j tends closer to a Gaussian distribution. ICA searches for a linear transformation $\mathbf{W}$, which makes the distribution of $\mathbf{S}' = \mathbf{WY}$ as non-Gaussian as possible. Many ICA algorithms return such $\mathbf{S}'$ as an estimate of the original sources $\mathbf{S}$. This leads to the most prevalent strategy—maximization of non-Gaussianity. With kurtosis and negentropy, many successful ICA algorithms were built upon this measurement, including FastICA [15] and JADE [16]. The other approach minimizes the mutual information between each source, using measurements like Kullback-Leibler Divergence or maximum entropy. Well known algorithms like Infomax [17] and Kernel ICA [18] follow this approach. Although there exist many ICA algorithms, as Hyvärinen [13] and Bell's explanation [19], they are indeed not "that" different. For efficiency, we simply choose FastICA as our primary ICA technique in this paper. Due to the space limit, here we omit further technical details. Interested readers can learn more from Hyvärinen's tutorial [13]. As our work only focuses on applying ICA in SCA, technical details do not affect the comprehension of the following discussions.

3 ICA-based Signal Recovery

This section focuses on how to apply ICA in the side channel context. With certain leakage model, we first demonstrate that recovering secret signals in

[3] Amplitude is defined as the maximum absolute value of the difference from the reference value. If the reference value is 0, amplitude is the maximum absolute value of the signal.

SCA can be converted to an ICA problem. However, directly applying ICA brings several technical issues. We propose several solutions in this section, then discuss their pros and cons. Moreover, to better fit the SCA context, we further present a specialized ICA algorithm to exploit the features of circuit signals. In order to make our approach more intuitive, we also present a toy example in the following discussion.

3.1 ICA versus SCA: Similarities and Differences

Throughout this paper, we assume the leakage follows the weighted Hamming Weight model. For an n-bit intermediate state X, the corresponding data-dependent leakage can be written as

$$\mathrm{L}(x) = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + ... + \alpha_n x_n,\ \ \alpha_i \in \mathbb{R} \tag{1}$$

where x_i represents the i-th bit of x. With T times measurements, the sequence of the intermediate states $\mathbf{x} = \{\mathbf{x}(1), \mathbf{x}(2), ..., \mathbf{x}(T)\}$ forms a T-length signal. Apparently, $\mathbf{x}$ can also be regarded as a group of binary signals, where $\mathbf{x} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}^T$ and $\mathbf{x}_i = \{\mathbf{x}_i(1), ..., \mathbf{x}_i(T)\}$. As the observed leakages $\mathbf{y}$ capture the instantaneous mixtures of $\mathbf{x}$, SCA in this setting shares many similarities with ICA (Figure 2). Indeed, the basic assumptions of ICA—non-gaussianity and independence—are

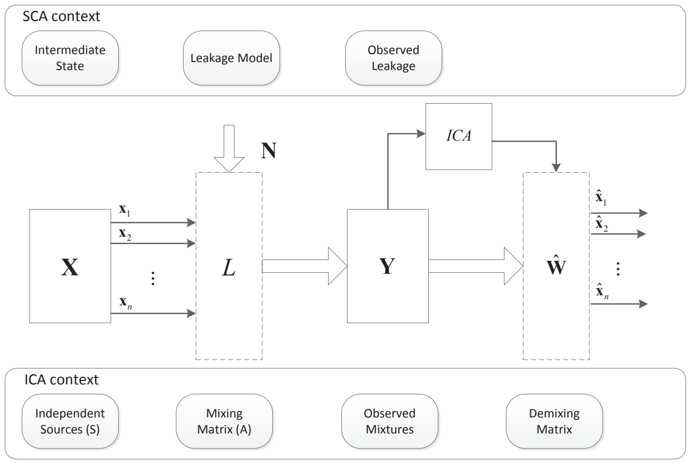

Fig. 2. A comparison between ICA and SCA

naturally satisfied: since all $\mathbf{x}_i$ are 1-bit 0-1 signals, the distribution of $\mathbf{x}_i$ is far from Gaussian. Considering the sources come from a cryptographic intermediate state, the cryptographic operation ensures each bit is statistically independent. The major difference is ICA requires at least n observations, whereas the SCA context provides only one leakage model. Although not explicitly stated, the additional noise may be another problem: since SCA exploits the unintended information leakage, the Signal-to-Noise-Ratio (SNR) in SCA is usually much lower than typical ICA contexts. For easy comparison, we list the similarities and differences between ICA and SCA in Table 1.

Table 1. Similarities and differences between ICA and SCA

	ICA	SCA
Sources	$\mathbf{s} = \{\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_n\}$	$\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$
Distribution	non-Gaussian	Bernoulli
Independence	independent	independent
Observation	$\mathbf{Y} = \mathbf{AS} + \mathbf{N}$	$\mathbf{l} = \alpha\mathbf{x} + \mathbf{N}$
Number of observations	m	1
Level of Noise	low	high

Toy example. Assume our intermediate state $\mathbf{x}$ has only 2 bits (n=2). The leakages follow the standard Hamming Weight model, where both α_1 and α_2 in Equation (1) equals to 1 and $\alpha_0 = 0$. If the attacker takes 4 leakage measurements (T=4) with $\{\mathbf{x}(1) = 0, \mathbf{x}(2) = 1, \mathbf{x}(3) = 2, \mathbf{x}(4) = 3\}$, the resultant leakage measurements $\mathbf{l} = \{0, 1, 1, 2\}$ can be regarded as an observation in ICA. $\mathbf{x}_1 = \{0, 0, 1, 1\}$ and $\mathbf{x}_2 = \{0, 1, 0, 1\}$ are two blind sources in ICA and the mixing procedure is $\mathbf{l} = \mathbf{x}_1 + \mathbf{x}_2$.

3.2 Applying ICA in SCA: Obstacles and Solutions

As demonstrated in Table 1, the number of observations and the level of noise are two major obstacles for applying ICA in the SCA context. In the following, we further discuss these two obstacles and propose several possible solutions.

Constructing multi-channel observations The first difficulty we have to overcome is to construct multi-channel observations from the side channel leakages. In the following, our discussion focuses on the best solution we found for power leakage. We also list a branch of other solutions in the end of this section, which may have better performance in other circumstances.

Our primary solution is based on a simple observation: if a binary source $\mathbf{s}$ is XORed with a constant k, the resultant source $\mathbf{s}'$ is

$$\mathbf{s}' = \begin{cases} \mathbf{s} & k = 0 \\ \mathbf{1} - \mathbf{s} & k = 1 \end{cases}$$

In ICA, since the *whitening transformation* removes all constant terms in $\mathbf{s}'$, XORing $k = 1$ has the same effect as flipping the sign of the whitened source $\mathbf{s}$. According to our discussion in Section 2.2, we can move the flipping sign to the corresponding coefficients in the mixing matrix $\mathbf{A}$. In SCA, suppose we can measure the leakages of $X = S \oplus k$. With m different k_i, the overall model can be regarded as

$$\mathbf{x}' = \mathbf{s}$$

$$\mathbf{A}' = \begin{pmatrix} \alpha_1 & \cdots & \alpha_n \\ \alpha_1 & \cdots & \alpha_\mathrm{n} \\ \vdots & \vdots & \vdots \\ \alpha_1 & \cdots & \alpha_n \end{pmatrix} \circ \begin{pmatrix} 2k_{1,1}-1 & \cdots & 2k_{1,n}-1 \\ 2k_{2,1}-1 & \cdots & 2k_{2,\mathrm{n}}-1 \\ \vdots & \vdots & \vdots \\ 2k_{m,1}-1 & \cdots & 2k_{m,n}-1 \end{pmatrix}$$

where $k_{i,j}$ represents the j-th bit of k_i. If all constants k are randomly picked, the probability that $\mathbf{A}'$ is singular is at most $\left(\frac{1}{\sqrt{2}} + o(1)\right)^n$ [20]. For $n = 8$, the exact probability is 0.52 [21]. This probability suggests that, with random k_i, the attacker may have to construct $m > n$ observations to ensure that ICA gets n linearly independent channels.

Toy example. In the previous example, assume the attacker can control a constant k that XORed to the intermediate state x. The attacker measures one observation $\mathbf{l}_1$ when $k = 0$. Then, the attacker repeats his measurements with exactly the same plaintext inputs, using $k = 1$. In this case, the measured leakage signal becomes $\mathbf{l}_2 = \mathbf{1} - \mathbf{x}_1 + \mathbf{x}_2$. Considering the whitening stage, we can omit the constant term α_0 and write the model as

$$\mathbf{l} = \begin{pmatrix} \mathbf{l}_1 \\ \mathbf{l}_2 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix}$$

Apparently, $\mathbf{l}$ forms a valid input for ICA ($m = n = 2$).

Considering XOR is a common operation in symmetric cryptography, finding such constant in practice should be easy. Since ICA does not need the actual value of k, in many block ciphers, the round key serves as a good candidate. The benefit of this approach, is the constructed observations are guaranteed to be different enough. Many other proposals we considered do not survive the whitening stage: their observations contain too much correlated information that only a few components remain significant after the whitening stage.

Other solutions. Apart from our XOR-constant method, the rest solutions can be categorized as follow:

– **Utilizing the multiple sample points:** Perhaps the cheapest way to construct multi-channel observations is taking full advantage of the leakage traces. The core idea of this strategy, is to construct multi-channel observations from the multiple sample points on the trace. Leakage traces usually contain many sample points corresponding to the same cryptographic operation. As the side channel leakage is an instantaneous quantity, such samples capture the changes of leakage over time.

  - **Directly using the multiple sample points as independent observations.** A trivial approach would be directly using these sample points as multi-channel observations. In general, samples from the same cryptographic operation are usually strongly correlated and fail the whitening stage. However, our experiments in Section 4 confirm this approach is effective with certain implementation: if the implementation performs DES's permutation P bit-by-bit, the power consumption of each bit is naturally separated in time. In this case, the attacker can directly use the leakage samples of different intermediate state bits as different observations.
  - **Using dimension reduction techniques to create independent observations.** Alternatively, the attacker can also use dimension reduction techniques, like *Principal Component Analysis* (PCA) [22] or *Linear Discriminant Analysis* (LDA) [23], to construct several useful channels. In our experiments, their output components can safely pass the whitening stage. However, they often fail to provide multiple effective observations, whether by constructing observations with large noise (PCA) or providing only one effective observation (LDA).
- **Signal decomposition.** This strategy suggests we can decompose a single channel signal into a set of additional components, and use those components as independent observations. A typical example is Singular Spectrum Analysis, which was introduced to SCA by Del Pozo and Standaert in 2015 [11]. Other notable methods include wavelet transform [24] and Empirical Model Decomposition [25]. Nonetheless, In our experiments, none of these decomposition techniques gives good result.

Finally, considering the specific measurement settings, constructing multi-channel observations might be easier for non-power leakages, such as electromagnetic emanations [4], acoustic emanations [5], etc.

Noise tolerance Like all noisy statistic models, noise significantly affects the performance of ICA. To our knowledge, most noisy ICA studies focus on the cases where the Signal-to-Noise Ratio (SNR) is relatively high. In the SCA context, the side channel measurements often contain much larger noise. For this reason, designing a more robust ICA is essential for ICA's application in SCA. Compared with the standard ICA context, ICA-based SCA does have some unusual *a priori* knowledge: all sources follow the Bernoulli distribution with $p = 0.5$. Taking the *a priori* distribution into consideration, ICA with the Maximum Likelihood Principle becomes a more robust choice. The problem of this approach is most ICA algorithms employ optimization techniques with continuous sources, while the Bernoulli distribution is discrete. Fortunately, Belouchrani and Cardoso had proposed an ICA algorithm specialized for discrete sources in 1994 [14]. Their approach estimates the unknown sources and the mixing matrix as well as the additional noise, then maximizes the likelihood via the Expectation-Maximization (EM) algorithm. With moderate adjustments, their EM-ICA can be a more robust candidate for circuit signal recovery.

3.3 Specialized ICA algorithm

Taking the *a priori* distribution into consideration, we present a specialized I-CA based on Belouchrani and Cardoso's EM-ICA [14]. In order to formally describe our ICA algorithm, we assume the sources $\mathbf{X}$ consist of T intermediate states $\{\mathbf{x}(1), ..., \mathbf{x}(T)\}$. Each bit of X forms an independent source, denoted as $\mathbf{x}_i = \{\mathbf{x}_i(1), ..., \mathbf{x}_i(T)\}$. Similarly, the observation $\mathbf{Y}$ can also be written as $\mathbf{Y} = \{\mathbf{y}_1, ..., \mathbf{y}_m\}^\top$, where $\mathbf{y}_i = \{\mathbf{y}_i(1), ..., \mathbf{y}_i(T)\}$. We further assume the additional noise $\mathbf{N}$ follows the multivariate Gaussian distribution, with mean 0 and covariance $R_\mathbf{N}$. The detailed algorithm is presented in Algorithm 1.

Algorithm 1 Specialized ICA for SCA

```
Step 1: Collect $m$ observations $\mathbf{Y} = \{\mathbf{y}_1, ..., \mathbf{y}_m\}^\top$
Step 2: Get $n$ independent components $(\mathbf{IC}_1, \mathbf{IC}_2, ..., \mathbf{IC}_n)$ from FastICA
Step 3: Find the closest binary signals $\tilde{\mathbf{x}} = \{\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, ..., \tilde{\mathbf{x}}_n\}$ for $\mathbf{IC}$
Step 4: Start EM-ICA with $\tilde{\mathbf{x}}$, estimate the initial parameter $\theta^{(0)}$
        while $\Delta L >$ threshold do
            E-Step: Compute the expectation of the log-likelihood $L\left(\theta^{(j)}\right)$ with
            current $\theta^{(j)}$
            M-Step: Compute the $\theta^{(j+1)}$ that maximize the expected log-
            likelihood function $L$
        end while
Step 5: Find the $\hat{\mathbf{x}}$ that maximize the expected log-likelihood function $L$
return  binary signals $\hat{\mathbf{x}}$
```

Step 1: The first step collects multi-channel observations for ICA. Specifically, the attacker measures the leakage traces $\mathbf{l}$ of the intermediate states $\mathbf{x}$. Suppose each trace contains q sample points, $\mathbf{l}$ can be written as a $T \times q$ matrix. The attacker manually chooses a point of interest, or a component after PCA/LDA. The selected component (or column) forms an observation for ICA. Then, considering the features of the specific implementation, the attacker can construct multi-channel observations, through our XOR-constant method or other methods in Section 3.2.

Step 2: After Step 1, we have set up an ideal context for ICA. In this paper, we choose FastICA as our primary ICA algorithm. FastICA is an efficient and popular algorithm invented by Aapo Hyvärinen [13]. It is worth mentioning that FastICA is not stable: if the noise level is relatively high, we often repeat this process several times, in order to find the best outputs for further analysis.

Step 3: Considering FastICA did not use the *a priori* distribution of $\mathbf{x}$, the resultant independent components $\mathbf{IC}$ are not binary signals. Thus, we need to transform the real-valued signals $\mathbf{IC}$ back to binary signals $\tilde{\mathbf{x}}$. Assuming the sources are balanced (0 and 1 appear with the same probability), the trivial approach is:

$$\tilde{\mathbf{x}}_i(t) = \begin{cases} 0 & \mathbf{IC}_i(t) < median(\mathbf{IC}_i(t)) \\ 1 & \mathbf{IC}_i(t) \geqslant median(\mathbf{IC}_i(t)) \end{cases}$$

Step 4: Given $\tilde{\mathbf{x}}$ is a valid source estimate, the EM algorithm starts with $\tilde{\mathbf{x}}$ and searches for the best fitted sources. Specifically, the EM algorithm first estimates the mixing matrix and the noise distribution as [14]:

$$\begin{aligned} R_{\mathbf{yy}} &= T^{-1} \sum_{t=1}^{T} \mathbf{y}(t)\mathbf{y}(t)^\top \\ R_{\mathbf{y}\tilde{\mathbf{x}}} &= T^{-1} \sum_{t=1}^{T} \mathbf{y}(t)\tilde{\mathbf{x}}(t)^\top \\ R_{\mathbf{x}\tilde{\mathbf{x}}} &= T^{-1} \sum_{t=1}^{T} \tilde{\mathbf{x}}(t)\tilde{\mathbf{x}}(t)^\top \\ \theta^{(0)} &= \left(\hat{A}, \hat{R}_{\mathbf{N}}\right) = \left(R_{\mathbf{y}\tilde{\mathbf{x}}} R_{\tilde{\mathbf{x}}\tilde{\mathbf{x}}}^{-1}, R_{\mathbf{yy}} - R_{\mathbf{y}\tilde{\mathbf{x}}} R_{\tilde{\mathbf{x}}\tilde{\mathbf{x}}}^{-1} R_{\mathbf{y}\tilde{\mathbf{x}}}^\top\right) \end{aligned}$$

where $R_{\mathbf{y}\tilde{\mathbf{x}}}$ is the correlation matrix of $\mathbf{y}$ and $\tilde{\mathbf{x}}$. Given the current estimate of the parameters $\theta^{(j)}$, the E-step computes the conditional distribution of the sources $\tilde{\mathbf{x}}$ through the Bayes theorem

$$p\left(\tilde{\mathbf{x}}|\mathbf{y};\theta\right) = \frac{p\left(\mathbf{y}|\tilde{\mathbf{x}};\theta\right) p\left(\tilde{\mathbf{x}}\right)}{p\left(\mathbf{y};\theta\right)}$$

Assume the intermediate state $x(t)$ is uniformly distributed and independent of other $x(t')$,

$$p\left(\tilde{\mathbf{x}}|\mathbf{y};\theta\right) = \prod_{t=1}^{T} p\left(\tilde{\mathbf{x}}(t)|\mathbf{y}(t);\theta\right) = \prod_{t=1}^{T} \left(\frac{p\left(\mathbf{y}(t)|\tilde{\mathbf{x}}(t);\theta\right)}{\sum\limits_{\tilde{\mathbf{x}}(t)=0}^{2^n-1} p\left(\mathbf{y}(t)|\tilde{\mathbf{x}}(t);\theta\right)} \right)$$

Since the noise follows the multivariate Gaussian distribution, $p\left(\mathbf{y}(t)|\tilde{\mathbf{x}}(t);\theta\right)$ can be estimated as

$$p\left(\mathbf{y}(t)|\tilde{\mathbf{x}}(t);\theta\right) = mvnpdf\left(\mathbf{y}(t) - \hat{\mathbf{A}}\tilde{\mathbf{x}}(t), \mathbf{0}, \hat{R}_{\mathbf{N}}\right)$$

where *mvnpdf* stands for the probability density function of the multivariate Gaussian distribution. Then, the log-likelihood function $L(\theta)$ is defined as

$$L\left(\theta\right) = \log\left(p\left(\mathbf{y};\theta\right)\right) = \sum_{t=1}^{T} \log\left(p\left(\mathbf{y}(t);\theta\right)\right)$$

The M-step, on the other hand, finds the parameter $\theta^{(j+1)}$ that maximizes the expected log-likelihood function $L(\theta)$ in the E-step. Specifically, the new parameter can be estimated as

$$\begin{aligned}
R_{\mathbf{y}\tilde{\mathbf{x}}}^{(j)} &= T^{-1} \sum_{t=1}^{T} \sum_{\tilde{\mathbf{x}}(t)=0}^{2^n-1} \mathbf{y}(t)\tilde{\mathbf{x}}(t)^{\top} p\left(\tilde{\mathbf{x}}(t)|\mathbf{y}(t);\theta^{(j)}\right) \\
R_{\tilde{\mathbf{x}}\tilde{\mathbf{x}}}^{(j)} &= T^{-1} \sum_{t=1}^{T} \sum_{\tilde{\mathbf{x}}(t)=0}^{2^n-1} \tilde{\mathbf{x}}(t)\tilde{\mathbf{x}}(t)^{\top} p\left(\tilde{\mathbf{x}}(t)|\mathbf{y}(t);\theta^{(j)}\right) \\
\mathbf{A}^{(j+1)} &= R_{\mathbf{y}\tilde{\mathbf{x}}}^{(j)} \left(R_{\tilde{\mathbf{x}}\tilde{\mathbf{x}}}^{(j)}\right)^{-1} \\
R_{\mathbf{N}}^{(j+1)} &= R_{\mathbf{y}\mathbf{y}} - \mathbf{A}^{(j+1)} R_{\tilde{\mathbf{x}}\mathbf{y}}^{(j)}
\end{aligned}$$

The attacker runs the E-step and M-step iteratively, until the algorithm converges to a local maximum ($\varDelta L < threshold$).

Step 5: The EM algorithm ends with some parameters $\hat{\theta}$ and the corresponding conditional distribution of the sources $\tilde{\mathbf{x}}$. To extract the most likely sources, we simply apply the Maximum Likelihood Principle

$$\hat{\mathbf{x}} = \arg\max_{\tilde{\mathbf{x}}} \left(p\left(\tilde{\mathbf{x}}|\mathbf{y};\theta\right)\right)$$

Toy example. In our example, the attacker gets two observations $\mathbf{l}_1$ and $\mathbf{l}_2$ from different k. In Step 1, the attacker uses $\mathbf{Y} = \mathbf{l}$ as ICA's input. In the noiseless case, $\hat{\mathbf{x}}$ contains two sources. One of the sources corresponds to $\mathbf{x_1}$ while the other corresponds to $\mathbf{x_2}$: the attacker cannot decide the correspondence. Meanwhile, the resultant signal $\hat{\mathbf{x}}_i$ could be equal to the original source ($\mathbf{x}_i$) or the inverse of the original source ($\mathbf{1}-\mathbf{x}_i$). Thus, the ICA result can be written as $\hat{\mathbf{x}} = B(\mathbf{x}) \oplus c$, where B is a bit permutation and c is a constant.

4 Applications in SCA

As our specialized ICA does not require any information from the plaintexts, it sheds light on several more advanced applications in SCA. In the following, we start our discussion by proposing an ICA-based SCA distinguisher, which gives comparable performance in the traditional first/last round SCA context. Furthermore, our approach also brings several more advanced attacks, including directly attacking the middle rounds and reverse engineering with fewer restrictions. Throughout this section, our experiments use leakages acquired from an unprotected software implementation of DES (Riscure Training Card 6). The power consumptions were measured with a LeCroy WaveRunner 610Zi oscilloscope at a sampling rate of 20 MSa/s. The entire trace set contains 20 000 traces, with 80 000 sample points covering the first 3 rounds. An appealing property of this card, is that it performs DES's permutation P bit-by-bit. As the influence of each signal bit is separated in time, the leakage trace naturally provides multi-channel observations. Most experiments in this section take advantage of this property. However, in Section 4.4, we present ICA analysis against the leakages of the Sboxes' input, where such property does not hold. Although the analysis becomes trickier, we can still perform a successful recovery with our XOR-constant method.

4.1 New SCA distinguisher

Although key recovery is not our primary goal, surprisingly, in our experiments, our specialized ICA can serve as a competitor for common key recovery attacks (DPA [2], CPA [26], LRA [27] etc.). Specifically, let us focus on one of the Sboxes (S_5) in the first round. Figure 3(a) presents the performance of traditional univariate CPA with 10 to 200 traces. Since the leakages from unprotected software implementation can be easily exploited, in Figure 3(a), the correct key (red line) stands out after 60 traces, although the distinguishing margin is quite small [4].

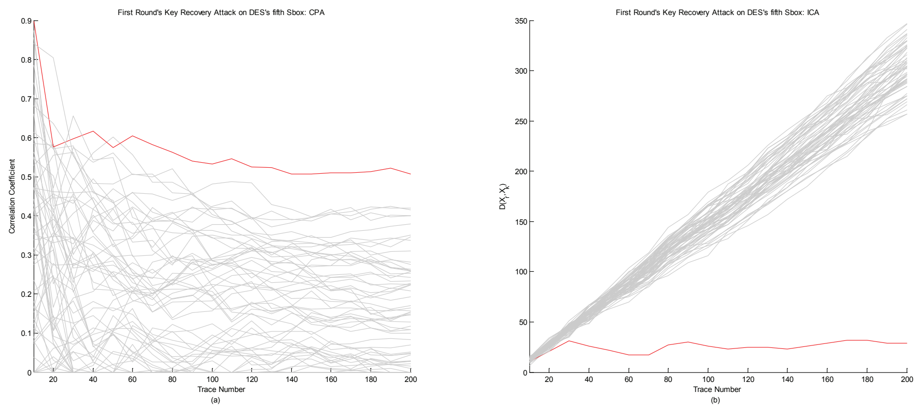

Fig. 3. Attacking S_5 in the first round: CPA v.s. ICA

On the other hand, in this particular implementation, we can also use our specialized ICA with several leakage points on the trace. In our experiments, we picked 5 points of interest ($m = 5$) from the leakages that correspond to S_5's output ($n = 4$). Similar to CPA, the attacker computes the guessed Sbox output sequences $\mathbf{X}_{\hat{k}}$ from the guessed key $\hat{k}$ and the known plaintext sequences $\{P_1, P_2, ..., P_T\}$:

$$\mathbf{X}_{\hat{k}} = \left\{ \mathrm{DES}_{S_5}(\hat{k}, P_1), \mathrm{DES}_{S_5}(\hat{k}, P_2), ..., \mathrm{DES}_{S_5}(\hat{k}, P_T) \right\}$$

[4] The maximal correlation coefficient of incorrect keys stays close to the correct one.

where DES_{S_5} represents the corresponding DES encryption in the first round. Denote the output signal of our ICA-based SCA as $\mathbf{X_r}$. Following our discussion in Section 2.2, we can only recover the correct signal up to its ICA equivalent. Namely, if $\mathbf{X_r}$ is the signal that ICA returned, applying any bit permutation B or flipping any sign also gives a correct ICA result. Thus, we choose the distance between the guessed key signal $\mathbf{X}_{\hat{k}}$ and its closest equivalent of $\mathbf{X_r}$ as our distinguish value. In the following, $||\mathbf{v}||_1$ stands for the L_1 norm (Manhattan norm) of $\mathbf{v}$, whereas $\mathbf{X}_{\hat{k},i}$ stands for the i-th bit of $\mathbf{X}_{\hat{k}}$.

$$D_{ICA}(\hat{k}) = \mathbf{D}\left(\mathbf{X}_{\hat{k}}, \mathbf{X_r}\right) = \min_{B}\left\{\sum_{i=1}^{n} dist\left(\mathbf{X}_{\hat{k},i}, \mathbf{X}_{\mathbf{r},B(i)}\right)\right\}$$
$$dist\left(\mathbf{X}_{\hat{k},i}, \mathbf{X}_{\mathbf{r},B(i)}\right) = \min\left\{\left\|\mathbf{X}_{\hat{k},i} - \mathbf{X}_{\mathbf{r},B(i)}\right\|_1, \left\|\mathbf{X}_{\hat{k},i} - \overline{\mathbf{X}_{\mathbf{r},B(i)}}\right\|_1\right\}$$

Figure 3(b) presents the performance of our ICA-based SCA with 10 to 200 traces. The correct key stands out after only 30 traces, which shows a slight advantage over CPA (60 traces). Besides, our ICA-based SCA provides a larger distinguishing margin: the distance with correct key stays stable after 50 traces, while the distances with incorrect keys increase linearly with the number of traces. In other words, our ICA-based SCA gives more reliable key recovery, especially for smaller trace sets.

4.2 Extending SCA to the Middle Rounds

Considering ICA does not take a guess-and-determine procedure, in theory, our ICA-based SCA can be applied to the leakages of any encryption operation, without the information about the plaintext or ciphertext. This feature becomes crucial when the target implementation uses power countermeasures to protect the first/last few rounds. In traditional SCAs, the attacker has to increase the guessed key space, in order to compute the middle round's intermediate states. This is hardly an issue for our ICA-based SCA: the middle round's intermediate states can be recovered without a pre-determined "signal list" (Figure 1).

Figure 4 demonstrates the results of recovering the 8 Sboxes' outputs ($n = 4$) in the second round. Since the implementation computes the 8 Sboxes sequentially, the attacker can locate each Sbox separately on the trace. Following our discussion above, we can evaluate our recovery with the distance between the I-CA returned signal ($\mathbf{X_r}$) and its closest ICA equivalent of the correct signal ($\mathbf{X_c}$). Since $\mathbf{D}\left(\mathbf{X_r}, \mathbf{X_c}\right)$ increases with the number of traces, we further define the success rate of an ICA recovery as:

$$\mathbf{Succ}(\mathbf{X_r}) = \frac{\mathbf{D}\left(\mathbf{X_r}, \mathbf{X_c}\right)}{nT}$$

From the definition of $\mathbf{D}\left(\mathbf{X_r}, \mathbf{X_c}\right)$, it is not hard to see that the success rate is at least 0.5. If the success rate equals to 0.5, the ICA-returned signal is no better than a random guess. On the other hand, if the success rate is significantly higher

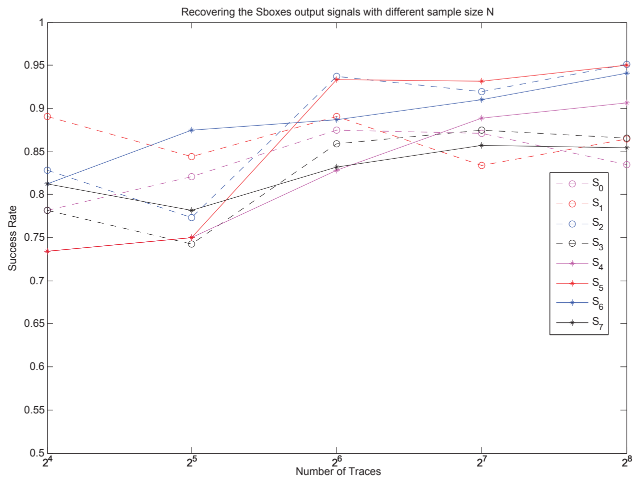

Fig. 4. Recovering the 8 Sboxes' outputs in the second round

than 0.5, ICA does learn some information about the underlying secret signals. The success rates of all 8 attempts in Figure 4 are over 0.8, which suggests our ICA-based attack learns most signal bits correctly. Unlike other SCAs, the success rate of our ICA-based SCA does not increase with larger trace set. This is because we are attacking the second round: in the first round, despite the secret key, we know for sure which leakages come from the same intermediate states. As the leakages with the same intermediate state can be aggregated together, this property helps us reduce the random noises and improve the performance of SCA, especially for larger trace sets. However, in the second round, we knows literally nothing about the intermediate states. We do not know whether two traces have the same intermediate state or not. Thus, each leakage point has to be analyzed independently, where the leakages of $\{x(1), x(2), ..., x(T-1)\}$ have no direct influence on the recovery of $x(T)$. In our experiments, when the trace set is extremely small (≤ 64), adding more traces gives more stable recovery. Further increasing the number of traces has little improvement on the success rate: ICA reveals more intermediate states, with the cost of longer running time. Nonetheless, as our goal here is merely key recovery, 80% accuracy is more than enough for further cryptanalysis. With the Sbox outputs of the second round recovered, the attacker can choose any round-reduced cryptanalysis to further exploit the secret key. Noted this attack is not specific to the second round: it can be applied to any round in the encryption process, as long as the implementation is unprotected.

4.3 Reverse Engineering on Sbox

Despite the applications in key recovery, reverse engineering seems to be a more natural application for our ICA-based SCA. In order to infer the underlying cryptographic operations, SCARE usually requires to recover the secret intermediate states first. Indeed, the experiments in Section 4.1 already confirm our ICA-based SCA can recover some information about the intermediate states. In the following, we further investigate our ICA-based SCA's application in reverse engineering. Specifically, let us first consider how to recover secret Sboxes.

Assume the target chip implements a customized DES with secret Sboxes. In the first round, the attacker can compute the output of the Expansion E. Similar to most SCAREs, the secret key is treated as a part of the secret Sbox ($S'(x) = S(x \oplus k)$), which means the output of E can be regarded as the input of S'. Since the Sboxes are secret, the attacker cannot take a key guess and analyze the leakages with traditional SCAs. Our ICA-based SCA works for this case, as long as the attacker can pick several independent leakage points. Figure 5(a) demonstrates the recovery of all 8 Sboxes' outputs ($n = 4$) in the first round with 5 ($m = 5$) manually picked leakage points. The analysis procedure is exactly the same as Section 4.1, except for the key distinguish step in the end. In all 8 attempts, our approach works very well with only 100 traces, recovering more than 80% of the intermediate states correctly. Since the attacker knows the Sbox inputs, adding more traces gives better recovery. In Figure 5(a), all 8 attempts report over 95% success rate after the number of traces reaches 400. Meanwhile, ICA with manually picked leakages always gives a few faulty recovery, even if the trace set is large (Figure 5(a)).

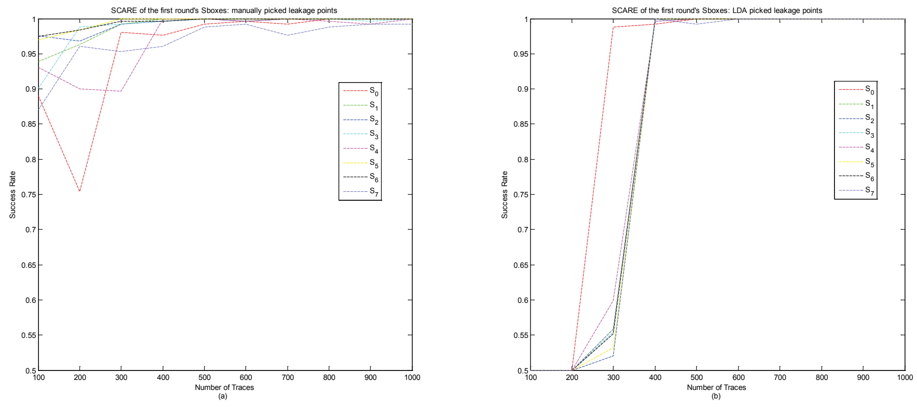

Fig. 5. Reverse engineering the first round's 8 Sboxes with ICA

The major drawback of picking leakages manually is the attacker may not know how to pick valid leakages in a non-profiled setting. There are a few rules of thumb, such as choosing timely disjunct leakage points with a high SNR (through NICV [28] or other point-of-interest choosing techniques). However, there are no solid theoretical explanations for such rules. For SCARE, since we have the Sboxes' input, LDA seems to be a much better choice. In our experiments, as the implementation performs the permutation bit-by-bit, the first 5 components form a valid input for ICA. The benefit of LDA is twofold: on the one hand, the attacker does not have to choose leakage points himself. All he has to do is to estimate an approximate range on the trace corresponding to the target computation. On the other hand, in Figure 5(b), our attack with LDA shows better recovery with larger trace sets. In fact, all 8 attempts achieve 100% accurate recovery with 600 traces. Noted in our traces set, the target operation lasts for 300-400 sample points. It is well known that LDA are not suitable for the cases where the number of traces is smaller than the range of interest [23]. This explains the fact that LDA gives poor results when the trace set is smaller than 400. Since the attacker has both the Sbox input and output now, writing the input and output in sequence gives the equivalent Sbox (up to ICA's ambiguity).

4.4 Reverse Engineering on Feistel Round Function

So far, all our experiments rely on the specific implementation of DES: as the implementation naturally provides multi-channel observations, the attacker can directly build ICA's input from the measured traces. For other ciphers or other implementations, this nice property does not always hold. Following our discussion in Section 3.2, if the implementation provides only one valid leakage function, the attacker has to construct multi-channel observations himself. In the following, our experiment demonstrates how the attacker can build his observations with our XOR-constant method and recover the output of the first round function.

Assume the attacker is reverse engineering a customized version of DES, where both the Sboxes and the linear permutation are altered. For convenience, we assume the attacker already knows the initial permutation IP and the expansion permutation E. Let F_{k1} denote the first round function of DES and (L_0, R_0) denote the initial input state (after IP). The second round's right state can be written as:

$$R_1 = L_0 \oplus F_{k1}(R_0)$$

Thus, the first Sbox's input in the second round can be written as

$$X = E_0(L_0 \oplus F_{k1}(R_0)) = E_0(L_0) \oplus E_0(F_{k1}(R_0))$$

where E_0 stands for taking the 6 least significant bits after the expansion E. Clearly, we can use the corresponding bits in L_0 as our XORed constant and recover the intermediate states $E_0(F_{k1}(R_0))$ $(n = 6)$. Specifically, in our experiments, we choose $E_0(L_0) = \{0x01, 0x02, 0x04, 0x08, 0x10, 0x20\}$ $(m = 6)$. For each value of $E_0(L_0)$, we set the other bits in L_0 to random numbers and let R_0 take 64 fix values $\{R_0(1), ..., R_0(64)\}$ $(T = 64)$. Considering the following recovery, the attacker can choose 64 random values, or set $\{R_0(1), ..., R_0(64)\}$ to anything he likes. Thus, we have 6 groups of 64 traces, where the intermediate state (S_0's input) sequences are the same, except for $E_0(L_0)$. The attacker can then pick one leakage point on the trace, and build 6 64-length leakage observations. As our discussion in Section 3.2, these observations can be regarded as the results of 6 different leakage functions with the same sources $\{E_0 \circ F_{k1}(R_0(1)), ..., E_0 \circ F_{k1}(R_0(64))\}$. Considering the noise, the attacker can also repeat the above measurements several times: all settings stay the same, except for the random bits in L_0. Since the input of S_0 stays the same while the inputs of other Sboxes vary each time, averaging different measurements together can further improve the Signal-to-Noise Ratio. Specifically, in our experiments, we repeat the above measurements 10 times and get 6 trace sets, with 640 traces in each set.

Figure 6(a) presents the recovery with one manually picked leakage point. Our analysis works well with 6 sets, recovering over 90% of the intermediate state bits correctly without any repetition. If the attacker repeats the measurement one more time and increases the trace set size to 128, our ICA-based SCA successfully recovers $\{E_0 \circ F_{k1}(R_0(1)), ..., E_0 \circ F_{k1}(R_0(64))\}$, up to ICA's ambiguity. On the other hand, if the attacker cannot build 6 trace sets, our analysis still works with inadequate observations. With 4 or 5 observations, our ICA-based SCA still learns about 80% of the intermediate state bits correctly.

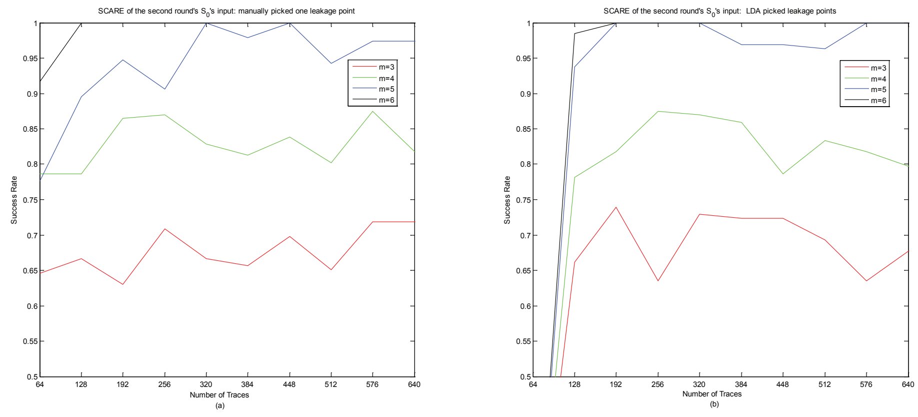

Fig. 6. Reverse engineering the first round's outputs of DES

It is worth mentioning though, finding a valid leakage point for ICA might be difficult. For this reason, we also present our ICA-based SCA with LDA in Figure 6(b). Unlike the previous section, LDA in this section simply takes the first component as our new observation. The benefit of LDA is the attacker does not have to explicitly pick one leakage point: instead, he may pick an approximate range and let LDA automatically find the most useful component. Figure 6(b) shows our recovery with 50 leakage points in LDA. Similar to the previous sections, LDA only gives valid results after the number of traces is much larger the input range ($\gg 50$). Another benefit of LDA shines when there are only 5 observations available: compared with Figure 6(a), the LDA approach gives better recovery. Noted since the attacker does not have the actual S_0's input, here we simply assume $\{E_0 \circ F_{k1}(R_0(1)), ..., E_0 \circ F_{k1}(R_0(64))\}$ are 64 different values and use $\{1, ..., 64\}$ as LDA's labels. As the first round's outputs are already recovered, the attacker can further perform other attacks to recover the inner structure of F_{k1}. It is worth mentioning that this attack is not specific to DES. Indeed, it works for any cipher with Feistel scheme, regardless of the inner structure or the specific implementation.

Remarks Due to the ambiguities of ICA, our recovery cannot learn the actual intermediate states, only its ICA equivalent. This is not an issue in SCARE: as most cryptographic components in SCARE are secret, most SCAREs only learn an equivalent form of the original cipher [9]. The difference between the recovered components and the original are usually cancelled by the following recovery. Take our recovery above for instance, if the original intermediate state is X, our ICA-based SCA returns $\hat{X} = B(X) \oplus c$, where B is a bit permutation and c is a constant. The following Sbox computation can be written as:

$$S(X) = S \circ B^{-1}(B(X) \oplus c \oplus c) = S \circ B^{-1}\left(\hat{X} \oplus c\right) = \; S'(\hat{X})$$

Since the attacker knows nothing about S, $\hat{X}$ and S' might as well be an valid form of the original cipher.

5 Discussions and Perspectives

This section presents a detailed discussion about our ICA-based SCA, demonstrating its competitive advantages against other techniques as well as more potential applications in the future.

5.1 Comparison with Collision Attacks

As a state recovery technique, collision attacks [6,9,8] is the closest competitor for our ICA-based SCA. Indeed, these two attacks share many features in their attack model. Both attacks require an approximate range of interest, in order to build effective templates or ICA inputs. In theory, both attacks apply to any implementation; although in practice, most experimental verifications come from sequential software implementations [9]. Nonetheless, most of the applications in Section 4 cannot use collision attacks. The major difficulty comes from the "online profilling" step, where the attacker has to find another computation of the exact same Sbox for profiling. Furthermore, the attacker must know the S-boxes' input in this profiling set, at least up to its permutation equivalent. This assumption limits the profiling set to the first round, as the Sboxes' inputs of the second round are not accessible. If the implementation protects the first round, collision attacks cannot build effective templates for further analysis. Even if the first round is unprotected, ciphers like DES still give collision attacks a hard time: as the round function uses 8 different Sboxes, collision attacks have to perform template matching with leakages from different rounds. Despite the fact that the leakages from different rounds might be different, the second round's (or other middle rounds') leakage cannot use averaging or LDA, which makes the measurements much noisier. Besides, collision attacks learn only 1 intermediate state in each template matching procedure, whereas our ICA-based SCA recovers T intermediate states with T traces. However, our ICA-based SCA does have one major drawback: it only handles linear leakages. If the leakages contain significant non-linear components, collision attacks may be the only choice, thanks to the "online profilling" step. For clarity, we list the similarities and differences between collision attacks and our ICA-based SCA in Table 2.

Table 2. Similarities and differences: collision attack v.s. ICA-based SCA

	Collision Attack	ICA-based SCA
Target	Intermediate states	Intermediate states
Point of interest	Approximate	Approximate
Implementation	Sequential & Software	Sequential & Software
Profiling	"Online profilling"	None
Attacked round	First/Last	Any
Assumption on leakage	None	Linear

5.2 Potential Applications

In theory, ICA is a quite powerful tool, suitable for various applications in SCA. As the first step in this direction, there are a lot of potential applications that our paper does not cover. In the following discussion, we list a few interesting applications of our ICA-based SCA. Noted these applications may require more advanced ICA techniques than our proposal in Section 3.

- **Separating parallel signals**: In traditional SCA, if the leakage contains two parallel components, we often choose one as our target and leave the other as the *switching noise* [29]. Since parallel signals can be regarded as independent components, in certain circumstances, ICA might be able to separate those signals. In traditional SCAs, such separation can significantly reduce the switching noise and improve the performance of side channel attacks.
- **Noise reduction**: In electroencephalography and image processing, ICA is a popular tool for removing artifacts and noises. In side channel analysis, we can expect the similar applications of ICA. The experiments in Section 4.1 already confirm ICA's potential in this direction. However, using ICA as a preprocessing tool still requires more research effort.

5.3 Future Improvements

In this section, we present some interesting extensions in this direction, which may further expand the applications of our ICA-based SCA.

- **Single channel ICA**: Generally speaking, applying ICA in SCA will be much easier, if ICA works with a single observation. So far, there are several techniques in this direction [30,24,25], although they do not work well with side channel leakages. It is worth mentioning that the EM algorithm still works when $m < n$, although its performance is far from satisfying.
- **Non-linear ICA**: One major limitation of our approach is typical ICA requires the mixing process ($\mathbf{A}$) to be linear. In other words, in SCA, the leakage model is limited to the weighted HW model. Although the weighted HW model suits many software implementations quite well [27], it might be interesting to ask whether ICA can be extended to non-linear leakages. There exist some ICA algorithms dealing with non-linear mixtures. Nonetheless, applying non-linear ICA to SCA still requires more research effort.
- **Parallel hardware implementations**: In Section 4, our experiments mainly focus on unprotected software implementations. As hardware implementations are getting more and more popular, whether our ICA-based SCA works for hardware implementations is an interesting question. Theoretically, the common leakage model in hardware implementations—the Hamming Distance (HD) model—is still a linear model. Suppose the attacker knows the last state in the target register, the HD model alone should not hinder our ICA-based SCA. However, the assumption of knowing the last state is not always reasonable, especially for the SCARE applications. Another issue with hardware implementations is they usually involves parallel operations, which significantly reduces the Signal-to-Noise Ratio. We believe ICA might be effective in some hardware implementations, although there is still much to be done in this direction.

6 Conclusion

Despite their threat to various embedded devices, typical side channel attacks are actually more like a “guess-and-determine” procedure. Instead of directly learning any secrets from the leakage, SCA takes guesses about the secret key and verifies them with the side channel leakages. In this paper, we propose an algorithm that directly learns the secret intermediate states (“signals”) from the observed leakages. Specifically, we show that under certain circumstances, the signal recovery problem can be regarded as a *Blind Source Separation* problem, and solved by the well-studied *Independent Component Analysis*. In order to find valid inputs for ICA, we propose several methods to construct multi-channel observations from the side channel leakages. In addition, to further exploit the specific features of circuit signals, we introduce a customized EM-ICA algorithm. Experiments show our analysis works well in certain ICA models, recovering most of the secret signal correctly with only a few hundred traces. Furthermore, our

ICA-based SCA brings new possibilities to the current non-profiled SCA study, including attacking the middle round's encryption and reverse engineering with fewer restrictions. Considering ICA is a more aggressive tool than most previous SCA techniques, we believe our ICA-based SCA is a promising tool for the future SCA study.

Acknowledgements

We would like to thank Prof. Ming Tang and Dr. Carolyn Whitnall for the inspiring discussions on this topic. We would also like to thank the anonymous reviewers for providing valuable comments. This work is supported by the National Basic Research Program of China (No.2013CB338002) and National Natural Science Foundation of China (No. 61272476, 61672509 and 61232009).

References

1. Kocher, P.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In Koblitz, N., ed.: Advances in Cryptology — CRYPTO 96. Volume 1109 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (1996) 104–113
2. Kocher, P., Jaffe, J., Jun, B.: Differential Power Analysis. In Wiener, M., ed.: Advances in Cryptology — CRYPTO 99. Volume 1666 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (1999) pp. 388–397
3. Quisquater, J.J., Samyde, D.: ElectroMagnetic Analysis (EMA): Measures and Counter-measures for Smart Cards. In Attali, I., Jensen, T., eds.: Smart Card Programming and Security. Volume 2140 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2001) 200–210
4. Agrawal, D., Archambeault, B., Rao, J., Rohatgi, P.: The EM Side-Channel(s). In Kaliski, B., Koç, ç., Paar, C., eds.: Cryptographic Hardware and Embedded Systems — CHES 2002. Volume 2523 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2003) 29–45
5. Genkin, D., Shamir, A., Tromer, E.: RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis. In Garay, J., Gennaro, R., eds.: Advances in Cryptology — CRYPTO 2014. Volume 8616 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2014) 444–461
6. Gérard, B., Standaert, F.X.: Unified and optimized linear collision attacks and their application in a non-profiled setting: extended version. Journal of Cryptographic Engineering **3**(1) (2013) 45–58
7. Clavier, C.: An Improved SCARE Cryptanalysis Against a Secret A3/A8 GSM Algorithm. In McDaniel, P., Gupta, S., eds.: Information Systems Security. Volume 4812 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2007) pp. 143–155
8. Clavier, C., Isorez, Q., Wurcker, A.: Complete SCARE of AES-Like Block Ciphers by Chosen Plaintext Collision Power Analysis. In Paul, G., Vaudenay, S., eds.: Progress in Cryptology — INDOCRYPT 2013. Volume 8250 of Lecture Notes in Computer Science. Springer International Publishing (2013) pp. 116–135

9. Rivain, M., Roche, T.: SCARE of Secret Ciphers with SPN Structures. In Sako, K., Sarkar, P., eds.: Advances in Cryptology — ASIACRYPT 2013. Volume 8269 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2013) pp. 526–544
10. Chari, S., Rao, J., Rohatgi, P.: Template Attacks. In Kaliski, B., Koç, ç., Paar, C., eds.: Cryptographic Hardware and Embedded Systems — CHES 2002. Volume 2523 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2003) 13–28
11. Merino Del Pozo, S., Standaert, F.X.: Blind Source Separation from Single Measurements Using Singular Spectrum Analysis. In Güneysu, T., Handschuh, H., eds.: Cryptographic Hardware and Embedded Systems – CHES 2015. Volume 9293 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2015) 42–59
12. Wiki: Blind signal separation. https://en.wikipedia.org/wiki/Blind_signal_separation
13. Hyvärinen, A., Oja, E.: Independent component analysis: algorithms and applications. Neural Networks **13** (2000) 411 – 430
14. Belouchrani, A., Cardoso, J.F.: Maximum Likelihood Source Separation By the Expectation-Maximization Technique: Deterministic and Stochastic Implementation. In: in Proc. NOLTA. (1995) 49–53
15. Hyvarinen, A.: Fast and robust fixed-point algorithms for independent component analysis. Neural Networks, IEEE Transactions on **10**(3) (May 1999) 626–634
16. Cardoso, J.: High-Order Contrasts for Independent Component Analysis. Neural Computation **11**(1) (Jan 1999) 157–192
17. Bell, A., Sejnowski, T.: An Information-Maximization Approach to Blind Separation and Blind Deconvolution. Neural Computation **7**(6) (Nov 1995) 1129–1159
18. Bach, F.R., Jordan, M.I.: Kernel Independent Component Analysis. J. Mach. Learn. Res. **3** (March 2003) 1–48
19. Tony Bell: An ICA page - papers, code, demos, links. http://cnl.salk.edu/~tony/ica.html
20. Bourgain, J., Vu, V.H., Wood, P.M.: On the singularity probability of discrete random matrices. Journal of Functional Analysis **258**(2) (2010) 559 – 603
21. Eric W. Weisstein: A057982: Number of singular n X n (-1,1)-matrices . http://oeis.org/A057982
22. Archambeau, C., Peeters, E., Standaert, F.X., Quisquater, J.J.: Template Attacks in Principal Subspaces. In Goubin, L., Matsui, M., eds.: Cryptographic Hardware and Embedded Systems — CHES 2006. Volume 4249 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2006) pp. 1–14
23. Standaert, F.X., Archambeau, C.: Using Subspace-Based Template Attacks to Compare and Combine Power and Electromagnetic Information Leakages. In Oswald, E., Rohatgi, P., eds.: Cryptographic Hardware and Embedded Systems - CHES 2008. Volume 5154 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2008) 411–425
24. Lin, J., Zhang, A.: Fault feature separation using wavelet-ICA filter . NDT&E International **38**(6) (2005) 421 – 427
25. Mijović, B., De Vos, M., I., G., Taelman, J., Van Huffel, S.: Source Separation From Single-Channel Recordings by Combining Empirical-Mode Decomposition and Independent Component Analysis. Biomedical Engineering, IEEE Transactions on **57**(9) (Sept 2010) 2188–2196
26. Brier, E., Clavier, C., Olivier, F.: Correlation Power Analysis with a Leakage Model. In Joye, M., Quisquater, J.J., eds.: Cryptographic Hardware and Embedded Systems — CHES 2004. Volume 3156 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2004) pp. 16–29

27. Doget, J., Prouff, E., Rivain, M., Standaert, F.X.: Univariate side channel attacks and leakage modeling. Journal of Cryptographic Engineering **1**(2) (2011) pp. 123–144
28. Bhasin, S., Danger, J.L., Guilley, S., Najm, Z.: NICV: Normalized inter-class variance for detection of side-channel leakage. In: Electromagnetic Compatibility, Tokyo (EMC'14/Tokyo), 2014 International Symposium on. (2014) 310–313
29. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks Revealing the Secrets of Smart Cards. Springer (2007)
30. Davies, M., James, C.: Source separation using single channel {ICA}. Signal Processing **87**(8) (2007) 1819 – 1832 Independent Component Analysis and Blind Source Separation.

A Reversing the second round's input state in AES

Since all our experiments in Section 4 uses DES as our target, we present another example with AES in this section. Assuming our target is a customized version of AES, where both the Sboxes and the diffusion layers are altered. Similar to Section 4.4, here we aim to recover the input state of the second round. In SPN ciphers, since all plaintext bits enter the round function, we cannot use plaintext bits as the XOR-contant. However, as ICA does not need the actual value of the XOR-constant, the secret round key serves as a good candidate. In order to ensure the signals are not affected, the attacker changes the plaintext according to the secret key choice. For instance, if the attacker measures the leakages of plaintext sequence $\{P_1, P_2, ..., P_T\}$ with secret key k_1, in the next measurement session, the attacker uses plaintexts $\{P_1 \oplus \Delta k, P_2 \oplus \Delta k, ..., P_T \oplus \Delta k\}$ where $\Delta k = k_1 \oplus k_2$. Thus, the first round encryption in two measurement sessions is exactly the same, except for the last round key. Although we do not know its value (the key schedule is also secret), this round key serves as the XOR-constant. In our experiment, we have measured 10 trace sets using different master keys ($n = 10$), with 256 different plaintexts in each set ($T = 256$). Each measurement is repeated 10 times, so that one trace set contains exactly 2560 traces. In this section, our experiments use power leakages acquired from an unprotected software implementation of AES (ATMega-163). The power consumptions were measured with a Picoscope 3206D oscilloscope at a sampling rate of 20 MSa/s.

Figure 7(a) demonstrates the recovery with one manually picked leakage point. Our analysis works well with 8 or 10 sets, recovering over 80% of the intermediate state bits correctly without any repetition. Further increasing the trace set size slightly improves the performance. On the other hand, if the attacker can only build 4 or 6 trace sets, our analysis still learns about 75% of the intermediate state bits correctly. Similar to Section 4.4, the LDA approach gives better recovery when trace size is significantly larger than the range of interest (300). Note that LDA performs extremely well with 6 observations: Figure 1 shows that the successful rate increases from 80% to 95%.

It is worth mentioning that in this experiment, our measurement uses an unusual assumption, where the attacker can alter the plaintext according to the

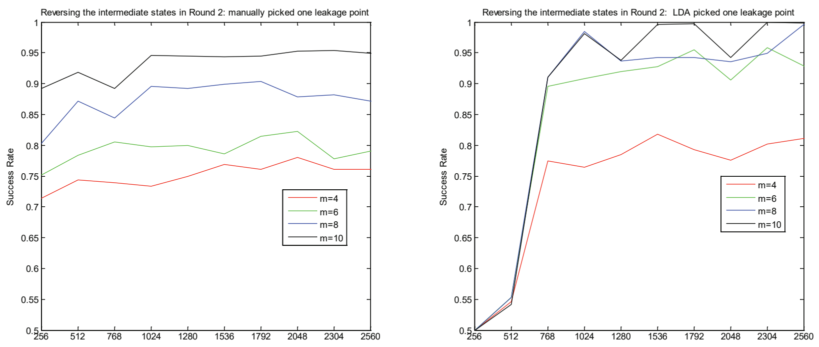

Fig. 7. Reverse engineering the first round's outputs of AES

key difference. In theory, this is quite close to the "related-key assumption" , where the attacker does not know the secret key, but knows the key difference. We are not sure how practical is this assumption in realistic attack. However, this is not an issue for SCARE: since our goal is to determine the secret cipher, it is reasonable to assume the attacker can set the master key. In key-recovery applications like Section 4.2, this assumption may still be problematic. Besides, this attack also relies on the fact that the first round key in AES is the master key. Assuming the key schedule is not given to the attacker, he cannot predict the value (or the difference) of any round key, unless the round key is the master key. Considering designers usually prefer light-weight key schedules, using the master key as the first round key (or whiten key) is not rare in today's block ciphers. Nonetheless, our attack in this section does not work with SPN ciphers with complex key schedules.