Manifold Learning Side-Channel Attacks against Masked Cryptographic Implementations

Jianye Gao[1], Xinyao Li[1], Changhai Ou[1,*], Zhu Wang[2], Fei Yan[1]

[1] School of Cyber Science and Engineering, Wuhan Univeristy, Hubei, China.

[2] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China.

Abstract: Masking, as a common countermeasure, has been widely utilized to protect cryptographic implementations against power side-channel attacks. It significantly enhances the difficulty of attacks, as the sensitive intermediate values are randomly partitioned into multiple parts and executed on different times. The adversary must amalgamate information across diverse time samples before launching an attack, which is generally accomplished by feature extraction (e.g., Points-Of-Interest (POIs) combination and dimensionality reduction). However, traditional POIs combination methods, machine learning and deep learning techniques are often too time consuming, and necessitate a significant amount of computational resources. In this paper, we undertake the first study on manifold learning and their applications against masked cryptographic implementations. The leaked information, which manifests as the manifold of high-dimensional power traces, is mapped into a low-dimensional space and achieves feature extraction through manifold learning techniques like ISOMAP, Locally Linear Embedding (LLE), and Laplacian Eigenmaps (LE). Moreover, to reduce the complexity, we further construct explicit polynomial mappings for manifold learning to facilitate the dimensionality reduction. Compared to the classical machine learning and deep learning techniques, our schemes built from manifold learning techniques are faster, unsupervised, and only require very simple parameter tuning. Their effectiveness has been fully validated by our detailed experiments.

Keywords: manifold learning; side-channel attacks; masking; dimensionality reduction; machine learning

1 Introduction

Unlike traditional attacks that conquer a cryptographic algorithm by finding its weakness in mathematical design, Side-Channel Attacks (SCAs) conquer it by exploiting its vulnerabilities in physical implementation. Cryptographic implementations on devices such as SIM cards[1], laptop and desktop computers [2, 3], and smartphones [4], have been successfully conquered in recent years. SCAs pose a very serious threat to their security and attract wide attentions. Power side-channel attacks, such as Differential Power Analysis (DPA) [5], Correlation Power Analysis (CPA) [6], Template Attacks (TA) [7], and Collision Attacks (CA) [8], extract confidential information through the power consumption of a cryptographic system. These attacks utilize a certain linear relationship between intermediate values and leakages. To protect the cryptosystems against the above power side-channel attacks, countermeasures such as masking techniques [9, 10], are widely employed.

Benefiting from secret sharing, the sensitive intermediate value (e.g., the output of Sbox in AES-128) in masked implementations is randomly divided into multiple shares [9]. Different shares are executed at distinct moments, necessitating that attackers exploit information distributed across different time

[*]Corresponding author.

samples. To launch a successful attack, the attacker should detect the location of each share, and combine the joint leakage of the sensitive variable. An example is, Oswald et al. employed educated guess that performed an exhaustive search over all possible $(d+1)$-tuples of time samples against dth-order masking in a selected window [11]. Another example is, Rioja at al. proposed Estimation of Distribution Algorithms (EDAs) which combined the POIs selection steps with the profiling and key recovery steps [12]. However, these traditional methods are highly time-consuming, especially when against higher-order masking with more shares and longer power traces. This significantly complicates the adversary's ability to extract the secret key from side-channel information. Durvaux et al. devised a novel approach to POI selection based on Projection Pursuits (PP) [13]. Nevertheless, it falls short when the number of shares is greater than 2 as pointed by Cagli et al. in [14]. Obviously, the above mentioned schemes do not significantly reduce the complexity of analysis and evaluations on masked implementations.

To further improve the attack and evaluation ability on masked implementations, rather than directly detecting the location of shares and combining them, Kernel Discriminant Analysis (KDA) [14], on the other hand, projects high-dimensional traces leaked from higher-order masked implementations onto low-dimensional space through a nonlinear kernel function and performs effectively in third- and fourth-order attacks. However, the KDA attack necessitates complex setting of parameters, and large number of power traces. Classical machine learning techniques such as Principal Component Analysis (PCA) [15] and Linear Discriminant Analysis (LDA) [16], are often used in power side-channel attacks and work very well. PCA is a linear transformation technique that projects the samples onto a lower-dimensional space while preserving the maximum amount of variance. LDA aims to find a linear combination of features that maximizes the separation between classes. However, they are only applicable for linear dimensionality reduction, and do not work against masked implementations. Non-linear dimensionality reduction techniques should be taken into consideration in this case.

Deep learning techniques such as Convolutional Neural Network (CNN) and Multi-Layer Perceptron (MLP), have been applied as competitive profiled attacks in [17]. Taking advantage of their possible employment without pre-processing, deep learning techniques can directly process power traces with hundreds or thousands of samples. Their strong feature extraction ability makes them extract the most obvious linear and non-linear leakage [18, 19]. Other recent studies also indicated that deep learning techniques often outperformed statistical-based attack methods [20–22]. Unlike the most recent works considering that the feature selection from power traces is known or pre-selected, the authors in [23] investigated several different feature selection scenarios. However, these deep learning techniques are geared towards hyper-parameter tuning and finding efficient neural architectures, which can be time-consuming and often task-specific. Moreover, the high requirement on the number of samples for training has not been significantly reduced. These drawbacks restrict their applications in real-world scenarios.

To improve the information extraction ability of power side-channel distinguishers against masked implementations, while reducing the requirements on the number of samples and difficulty on training of parameters and models, this paper undertakes the first study on manifold learning power side-channel attacks and their applications against masked implementations. The leaked information, which manifests as the manifold of high-dimensional power traces, is effectively harnessed to achieve superior performance in the attacks. Three classical manifold learning techniques, Laplacian Eigenmaps (LE), Locally Linear Embedding (LLE) and ISOMAP, are employed. To reduce the complexity and facilitate the dimensionality reduction, we further introduce explicit mappings into manifold learning by establishing a polynomial mapping relationship between the high-dimensional power traces and their low-dimensional representations. Manifold learning techniques neither necessitate custom kernel functions like KDA, nor do they require complex parameter settings and a larger number of power traces in training like deep learning techniques. Moreover, these three manifold learning solutions are unsupervised, which renders dimensionality reduction against masked implementations in manifold learning highly straightforward and efficient. Experiments verify the superiority of our schemes.

The rest of this paper is organized as follows. Section 2 gives the notations used in this paper, introduces TA and countermeasures. Non-linear dimensionality reduction techniques such as KDA, and deep learning schemes MLP and CNN, are also introduced here. The theory and general framework of manifold learning methods are presented in Section 3. Three typical techniques LE, LLE and ISOMAP, and their formulation, are also presented in this section. Our explicit mapping of manifold learning is given in Section 4. Experiments are conducted in Section 5 to illustrate their superiority. Finally, conclusion and future work are given in Section 6.

2 Preliminaries

2.1 Notations

Let calligraphic letter $\mathcal{X}$ denote a set, the corresponding uppercase letter X ($\boldsymbol{X}$) denote a random variable (vector) over $\mathcal{X}$, and the corresponding lowercase letter x ($\boldsymbol{x}$) denote the corresponding realization. Let $\boldsymbol{x}[k]$ denote the k-th entry of vector $\boldsymbol{x}$, and $\|\boldsymbol{x}\| = \sqrt{\sum_{k=1}^{n} (\boldsymbol{x}[k])^2}$ represent Euclidean norm for $\boldsymbol{x} \in \mathbb{R}^n$. A bold capital sans-serif letter $\mathsf{X} = \{\mathsf{X}[i,j]\}_{m\times n}$ denotes a real-valued matrix with n rows and m columns, whose element that lies in the cross of i-th row and j-th column is represented by $\mathsf{X}[i,j]$.

2.2 Template Attack

As one of the most mature side-channel attacks, power side-channel analysis can be divided into two categories: non-profiled attacks and profiled attacks. The former, e.g. CPA, directly performs attacks on the power traces sampled from the target device. The latter, e.g. Template Attack (TA), assumes that the adversary has a clone device of the target device, and he can collect enough samples to profile accurate models to enhance his attack on the target device. Therefore, Profiled attacks are usually more powerful than non-profiled attacks in this case.

Let τ_i denote the target plaintext byte in the i-th encryption, κ^* denote the corresponding sub-key, z denote the corresponding intermediate value satisfying $z_i = \text{Sbox}(\tau_i \oplus \kappa)$, and "Sbox" is the look-up table operation in AES-128. TA consists of two separate stages: the profiling stage and the attack stage. In the profiling stage, the adversary obtains a profiling set $\mathcal{D}_{profiling} = \{\boldsymbol{x}_i, \tau_i, \kappa\}(i = 1, \ldots, N_{profiling})$ with a total of $N_{profiling}$ power traces sampled from his clone device. Here the sub-key $\kappa \in \{0, 1, \ldots, 255\}$ can be fixed or random, since the adversary can fully control his clone device and change parameters. He then divides them into several categories according to leakage characteristics, e.g., the Hamming weight of their corresponding intermediate values z_i-s. Then, he estimates the templates including a multivariate Gaussian distribution with a mean vector and a covariance matrix for each category. This is usually accompanied by the selection of POIs, since TA performed on these samples with obvious leakage is more effective. In the attack stage, the adversary gets a set $\mathcal{D}_{attack} = \{\boldsymbol{x}_i, \tau_i\}(i = 1, \ldots, N_{attack})$ with a total of N_{attack} power traces from the target implementation. The secret key in the target implementation is fixed but unknown. For each guessing sub-key κ^*, the adversary matches the attack power traces with templates, and finally tries to recover the key following the Maximum Likelihood strategy.

2.3 Masking

Due to the serious threats on cryptographic systems posed by side-channel attacks, corresponding countermeasures like masking have been attracted wide concerns. Masking [9] is a typical application of secret sharing. For a d-th order masked cryptographic implementation, its intermediate value z (e.g., the output of Sbox in AES-128) can be divided into $d + 1$ shares s_1, s_2, …, s_{d+1} satisfying:

$$z = s_1 \circ s_2 \circ \cdots \circ s_{d+1}. \tag{1}$$

Here $\circ$ represents group operation, such as bitwise XOR for boolean masking. The first d shares s_1, s_2, …, s_d are random and the last share s_{d+1} is computed according to Eq. 1. In the most of software implementations, these shares are manipulated at different times, and leakages occur at different time samples. To exploit the leakage, the attacker has to find the leaky location of these shares, i.e., Points-Of-Interest (POIs). However, the adversary will face with high complexity when seeking the location of these shares, and this complexity increases rapidly as the number of shares increases.

2.4 KDA

To reduce the complexity of power side-channel attacks against masked implementations, a typical strategy is dimensionality reduction, which tries to reduce the number of dimensions while retaining as much information as possible. Let n denote the number of sample points on a power trace $\boldsymbol{x}_i$, i.e., $\boldsymbol{x}_i \in \mathbb{R}^n$. Dimensionality reduction serves as an extractor $\varepsilon : \mathbb{R}^n \rightarrow \mathbb{R}^m$ mapping the raw power trace $\boldsymbol{x}_i$ to its corresponding low-dimensional representation $\boldsymbol{y}_i \in \mathbb{R}^m$, where $m \ll n$. In other words, we can consider the power consumption leaked from a masked cryptographic implementation as high-dimensional non-linear samples, and attempt to find a low-dimensional mapping that exhibits a linear relationship with the masked intermediate value z in a low-dimensional space. In this case, we remove the redundancy in its dimensions and reduce the complexity of attacks.

Cagli et al. did a very beautiful work and introduced the non-linear dimensionality reduction technique named Kernel Discriminant Analysis (KDA) to reduce the dimension of power traces [14]. Given a set of labelled power traces $(\boldsymbol{x}_i^{z_i})_{i=1,...,N}$ (z_i is the corresponding label, e.g., the output of Sbox in AES-128) and the kernel function $\phi(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i \cdot \boldsymbol{x}_j)^d$ ($\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are power traces, "$\cdot$" is dot product and d stands for the order of masking), the procedure of KDA are shown as follows: (1) construct a matrix $\mathbf{S}$ (acting as between-class scatter matrix) as:

$$\mathbf{S} = \sum_{z \in \mathcal{Z}} N_z (\boldsymbol{s}_z - \boldsymbol{s}_h)(\boldsymbol{s}_z - \boldsymbol{s}_h)^\intercal, \tag{2}$$

where N_z denotes the number of traces in the class z, $\boldsymbol{s}_z$ and $\boldsymbol{s}_h$ are two N-size column vectors whose entries are given by:

$$\boldsymbol{s}_z[j] = \frac{1}{N_z} \sum_{i:z_i=z}^{N_z} \phi(\boldsymbol{x}_j^{z_j}, \boldsymbol{x}_i^{z_i}), \tag{3}$$

and

$$\boldsymbol{s}_h[j] = \frac{1}{N} \sum_{i=1}^{N} \phi(\boldsymbol{x}_j^{z_j}, \boldsymbol{x}_i^{z_i}). \tag{4}$$

Here N denotes the total number of traces. (2) Construct a matrix $\mathbf{Q}$ (acting as within-class scatter matrix) as:

$$\mathbf{Q} = \sum_{z \in \mathcal{Z}} \mathbf{K}_z (\mathbf{I}_{N_z} - \mathbf{N}) \mathbf{K}_z^\intercal, \tag{5}$$

where $\mathbf{I}_{N_z}$ is a $N_z \cdot N_z$ identity matrix, $\mathbf{N}$ is a $N_z \cdot N_z$ matrix with all entries equal to $\frac{1}{N_z}$, and $\mathbf{K}_{\boldsymbol{z}}$ is the $N \cdot N_z$ sub-matrix of $\mathbf{K} = (\phi(\boldsymbol{x}_i^{z_i}, \boldsymbol{x}_j^{z_j}))_{i=1,...,N, j=1,...,N}$ storing only columns indexed by the index i such that $z_i = z$. Afterwards, KDA regularizes the matrix $\mathbf{Q}$ as: $\mathbf{Q} = \mathbf{Q} + \mu \mathbf{I}_{N_z}$. (3) Find the non-zero eigenvalues $\lambda_1, ..., \lambda_Q$ and the corresponding eigenvectors $\boldsymbol{v}_1, ..., \boldsymbol{v}_Q$ of $\mathbf{Q}^{-1}\mathbf{S}$. (4) Finally, the projection of a power trace $\boldsymbol{x}$ over the j-th non-linear d-th order discriminant component can be computed as:

$$\varepsilon_j^{\text{KDA}}(x) = \sum_{i=1}^{N} \boldsymbol{v}_j[i] \phi(\boldsymbol{x}_i^{z_i}, \boldsymbol{x}). \tag{6}$$

Side-channel attacks such as multi-variable template attack and CPA, can be performed on the dimension-reduced traces after the above 4 steps.

It is noteworthy that KDA applies the kernel trick to Linear Discriminant Analysis (LDA) [16]. The kernel function $\phi(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x} \cdot \boldsymbol{y})^d$ is associated with a non-linear mapping $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}^f$ where $f > n$.

Thus, KDA is equivalent to mapping the power traces to a higher-dimensional space through Φ and then performing LDA (see [14] for details). Since the mapping Φ is non-linear, KDA is also a non-linear mapping in this case. The projection in the above Step 4 can also happen in profiling stage of a profiled attack (e.g., TA), since nonlinear dimensionality reduction like KDA, is usually time-consuming, the projection of power traces makes the adversary quickly obtain the corresponding low-dimensional samples, and thereby facilitating the model training.

2.5 MLP and CNN

Deep learning is a subset of machine learning that uses neural networks to learn from data. It has gained significant attention in recent years due to its remarkable success in various domains, including computer vision [24], natural language processing [25], speech recognition [26], etc. Deep learning models such as Multi-Layer Perceptron (MLP) [27] and Convolutional Neural Networks (CNNs) [28], have shown promising results in power side-channel attacks.

MLP is a feed-forward neural network architecture that possesses multiple layers of neurons. Each neuron in the MLP receives input from the previous layer, processes it using a nonlinear activation function, and produces an output that is passed to the next layer. In the domain of power side-channel attacks, the MLP model can be leveraged for feature extraction and classification of power traces. Specifically, the neurons in the first layer of the MLP receive the power traces as input, and the neurons in the last layer produce the classification output. Meanwhile, the neurons in the intermediate layers perform nonlinear transformations of the input to learn the underlying patterns presenting in the power traces.

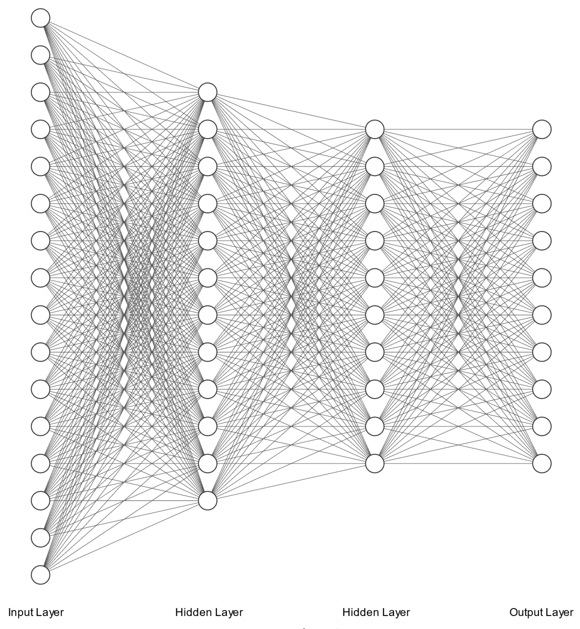


Figure 1: MLP Architecture.

CNNs are a type of neural network that is particularly effective for image analysis and recognition. It consists of multiple layers, including convolutional layers, pooling layers, and dense layers. Like MLP, a CNN can be used to extract features from power traces in power side-channel attacks. The convolutional layers use filters to extract spatial patterns from the power traces, while the pooling layers down-sample the output of the convolutional layers to reduce computational complexity. The dense layer uses the extracted features to classify the power traces and achieve key recovery.

CNN and MLP are particularly effective for classification tasks that involve high-dimensional input data. Both of them have been shown to outperform traditional machine learning techniques, such as Support Vector Machines (SVMs) and Random Forests, in side-channel attacks [29].

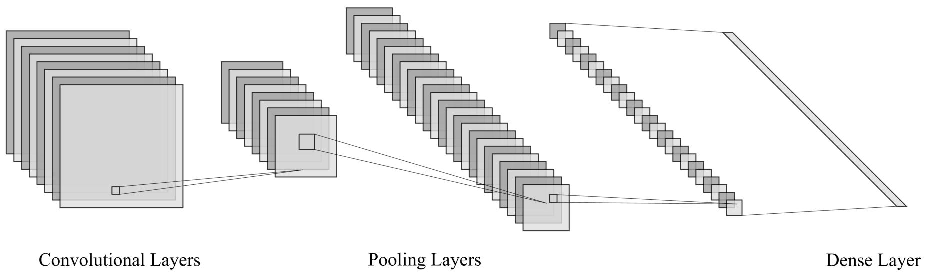


Figure 2: CNN Architecture.

3 Manifold Learning

Manifold learning techniques such as ISOMAP [30], Locally Linear Embedding (LLE) [31] and Laplacian Eigenmaps (LE) [32], show outstanding performance in nonlinear dimensionality reduction and attract wide attention. These techniques are a branch of machine learning, that aim to map high-dimensional samples into a low-dimensional space while preserving the local structure of the original data set. In other words, they try to preserve the relationships between neighboring data points (e.g., power traces) while reducing the dimensionality. This is particularly useful in cases where high-dimensional samples are difficult to visualize or analyze. In this section, we first introduce a general framework for manifold learning based on graph embedding, which helps us better understand manifold learning and derive its explicit mapping. Then, we present three typical manifold learning methods and show how they can be reformulated to fit the general framework mentioned above.

3.1 General Framework for Manifold Learning Methods

Most of manifold learning methods reduce the dimensionality of the sample points from the view of graph embedding. Precisely, they describe the relationship between the sample points through an undirected weighted graph $\boldsymbol{G} = \{\mathcal{V}, \mathbf{W}\}$, where $\mathcal{V}$ refers to the vertex set and $\mathbf{W}$ refers to the similarity matrix. Each element of matrix $\mathbf{W}$ represents the similarity for a pair of vertices. $\mathbf{W}$ can be calculated through various similarity criteria. In fact, the difference between manifold learning methods is their similarity criteria for calculating matrix $\mathbf{W}$. For example, LLE exploits the local neighborhood relationship, while LE utilizes the Gaussian kernel function.

Taking advantage of the fact that the difference between manifold learning methods is their similarity criteria for calculating matrix $\mathbf{W}$, we can give a general framework for them. Here we introduce the degree matrix $\mathbf{D}$ and the Laplacian matrix $\mathbf{L}$ of a graph $\boldsymbol{G}$. $\mathbf{D}$ is a diagonal matrix and its element, which represents the degree of corresponding vertex, is defined as $\mathbf{D}[i,i] = \sum_{i \neq j} \mathbf{W}[i,j]$. Here $\mathbf{L}$ is defined as $\mathbf{L} = \mathbf{D} - \mathbf{W}$. The graph embedding of a graph $\boldsymbol{G}$ can be defined as a technique that aims to find a low-dimensional vector representation, which accurately captures the similarity relationship between the pairs of vertices in $\boldsymbol{G}$ [33]. The objective function that the graph embedding minimizes can be expressed as follows:

$$\min_{\mathbf{Y}} \sum_{i,j=1}^{N} \mathbf{W}[i,j] \left\| \boldsymbol{y}_i - \boldsymbol{y}_j \right\|_2^2 . \tag{7}$$

To simplify the graph embedding problem in power side-channel attacks, a constraint can be added as $\sum_{i=1}^{N} \mathbf{B}[i,i] \cdot \boldsymbol{y}_i \boldsymbol{y}_i^{\intercal} = \mathbf{A}$, where $\mathbf{B}$ is a diagonal constraint matrix and $\mathbf{A}$ is a constant matrix. Typically, $\mathbf{B}$ is set to degree matrix $\mathbf{D}$ and $\mathbf{A}$ is identity matrix $\mathbf{I}_N$. Thus, the above graph embedding problem is converted to the following optimization problem:

$$\begin{aligned} &\min_{\mathbf{Y}} \sum_{i,j=1}^{N} \frac{1}{2}\mathbf{W}[i,j] \cdot \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2 = \min_{\mathbf{Y}} \operatorname{tr}\left(\mathbf{YLY}^\intercal\right), \\ &\text{s.t.} \quad \sum_{i=1}^{N} \mathbf{B}[i,i] \cdot \boldsymbol{y}_i\boldsymbol{y}_i^\intercal = \mathbf{YBY}^\intercal = \mathbf{A}, \end{aligned} \tag{8}$$

where $\mathbf{Y} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_N\}$, and $\boldsymbol{y}_i \in \mathbb{R}^m$ refers to the corresponding low-dimensional representation of input the power trace $\boldsymbol{x}_i$.

By the Rayleigh–Ritz theorem, the above graph embedding optimization problem is converted to solving the generalized eigenvalue problem:

$$\mathbf{LY} = \lambda\mathbf{BY}. \tag{9}$$

Here λ refers to the eigenvalue of the Laplacian matrix $\mathbf{L}$. The low-dimensional embedding coordinates are obtained by the eigenvectors corresponding to the m smallest non-zero eigenvalues.

3.2 Formulation of Manifold Learning Methods

In this subsection, three typical manifold learning methods, LE, LLE and ISOMAP, will be presented using the general framework introduced in Section 3.1 from the point view of graph embedding. As discussed in the previous Section 3.1, the manifold learning methods differ in the similarity criteria. Thus, we concentrate on the calculation of matrices $\mathbf{W}$ and $\mathbf{B}$ when introducing manifold learning methods.

3.2.1 Laplacian Eigenmaps

Laplacian Eigenmaps (LE) [32] is a local feature preserving algorithm, which aims to maintain local properties of power traces. Here we consider each power trace as a node in manifold learning. LE builds a graph from neighborhood information of the power trace set, i.e., nodes that are close in the original space are also close after mapping. If two nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ (i.e., the i-th and the j-th power traces $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$) are adjacent, their low-dimensional representations $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ are also adjacent. It is not difficult to find that the target of LE is the same as the graph embedding when similarity matrix $\mathbf{W}$ is criticized by the distance between nodes (i.e., power traces). Thus, LE naturally follows the way of graph embedding.

The procedures of LE in general framework are as follows:

(1) Neighbors selection. We need to define the adjacency graph $\boldsymbol{G}$ containing all sample points before select neighbors for them. Each power trace is regarded as a sample point in this case as explained before. Generally, we use k-Nearest Neighbors (k-NN) strategy to construct the adjacency graph. Nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are adjacent if node $\boldsymbol{x}_i$ is among the k nearest neighbors of node $\boldsymbol{x}_j$, or node $\boldsymbol{x}_j$ is among the k nearest neighbors of node $\boldsymbol{x}_i$. Here Euclidean norm is used to measure the distance between two nodes.

(2) Computing reformulation matrix. The entry of similarity matrix $\mathbf{W}$ can be directly calculated by Gaussian kernel function as follows:

$$\mathbf{W}[i,j] = \begin{cases} e^{-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{t}}, & \text{if nodes } \boldsymbol{x}_i \text{ and } \boldsymbol{x}_j \text{ are adjacent} \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

where t is a hyper-parameter that measures the distance between different pairs of neighbors. t is mostly set to a positive integer. The attacker can also simply set $\mathbf{W}[i,j]$ to 1 if nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are adjacent and 0 otherwise, of which the case also corresponds to $t = \infty$. LE aims to minimize the same objective function as graph embedding in Eq. (7). Thus, LE naturally follows the graph embedding formulation. As a result, $\mathbf{B} = \mathbf{D}$ and $\mathbf{A} = \mathbf{I}_N$, where $\mathbf{D}[i,i] = \sum_{i \neq j} \mathbf{W}[i,j]$.

(3) Computing the low-dimensional embedding. The low-dimensional representation can be calculated by solving the generalized eigenvalue problem given in Eq. (9).

There are three hyper-parameters k, m and t in LE. k and m also appear in other manifold learning methods like LLE and ISOMAP, and should be reasonably set. We will discuss them in Section 5.3. The complexity of constructing adjacency graph in LE is $\mathcal{O}\left(nN^2\right)$, and the complexity of computing reconstruction weight matrix W is less than $\mathcal{O}(knN)$. Computing low-dimensional embedding takes $\mathcal{O}\left(mN^2\right)$ operations where m stands for the output dimension.

3.2.2 Locally Linear Embedding

Locally Linear Embedding (LLE), like LE, is also a local feature preserving algorithm. LLE considers the structure of data as linearity in local sense. Therefore, a node $\boldsymbol{x}_i$ (i.e., power trace) is approximated by the linear combination $\sum_{j=1}^{k}\mathbf{M}[i,j]\boldsymbol{x}_j$ of its neighbor nodes $\boldsymbol{x}_j$-s. Unlike the similarity matrix $\mathbf{W}$ in general framework and LE, $\mathbf{M}$ here stands for reconstruction weight matrix in LLE. In other words, $\boldsymbol{x}_i \approx \sum_{j=1}^{k}\mathbf{M}[i,j]\boldsymbol{x}_j$ where $\boldsymbol{x}_j$ is among the k nearest neighbors of point $\boldsymbol{x}_i$ and $\mathbf{M}[i,j]$ refers to the reconstruction weight, which represents the contribution of power trace $\boldsymbol{x}_j$ to $\boldsymbol{x}_i$'s reconstruction. In this way, weights reconstruction is constructed between each node and its neighboring nodes. The reconstructed weight vectors keep the local linear structure of high-dimensional power traces. The local linear structure is also maintained in the low-dimensional space in this case, i.e. $\boldsymbol{y}_i \approx \sum_{j=1}^{k}\mathbf{M}[i,j]\boldsymbol{y}_j$.

The procedures of LLE in general framework are as follows:

(1) Neighbors selection. Similar to LE, the first step of LLE is also to find the k nearest neighbors for each node. The procedure of neighbors selection can be aligned with LE.

(2) Computing reconstruction weight matrix. To get the approximate reconstruction of $\mathbf{X}$, we minimize the object function (i.e., the error of reconstruction) as:

$$\epsilon(\mathbf{M}) = \sum_{i=1}^{N}\left\|\boldsymbol{x}_i - \sum_{j=1}^{k}\mathbf{M}[i,j]\boldsymbol{x}_j\right\|^2, \tag{11}$$

where $\mathbf{M}[i,j]$ denotes the reconstruction weight between two nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. Here we define $\mathbf{M}[i,j] \neq 0$ if $\boldsymbol{x}_j$ is adjacent to $\boldsymbol{x}_i$. Otherwise, $\mathbf{M}[i,j] = 0$. Besides, $\sum_{j=1}^{k}\mathbf{M}[i,j] = 1$. In order to minimize the reconstruction error, the reconstruction weight matrix $\mathbf{M} = \{\mathbf{M}[i,j]\}_{N\times N}$ satisfies symmetry.

(3) Computing reformulation matrix. The low-dimensional representation $\mathbf{Y}$ in LLE is obtained by minimizing the loss function (i.e., the objective function):

$$\begin{aligned}\psi(\mathbf{Y}) &= \sum_{i=1}^{N}\left\|\boldsymbol{y}_i - \sum_{j=1}^{k}\mathbf{M}[i,j]\boldsymbol{y}_j\right\|^2 \\ &= \left\|\mathbf{Y}(\mathbf{I}-\mathbf{M}^\mathsf{T})\right\|^2 \\ &= \operatorname{tr}(\mathbf{Y}\mathbf{L}'\mathbf{Y}^\mathsf{T}),\end{aligned} \tag{12}$$

where

$$\mathbf{L}' = (\mathbf{I}-\mathbf{M})^\mathsf{T}(\mathbf{I}-\mathbf{M}). \tag{13}$$

It is clear that the loss function in Eq. (12) is similar to Eq. (8). Besides, since $\sum_i \mathbf{M}[i,j] = 1$ for all possible i, thus we can further obtain:

$$\begin{aligned}
\sum_j \mathbf{L}^{'}[i,j] &= \sum_j [(\mathbf{I}-\mathbf{M})^{\mathsf{T}}(\mathbf{I}-\mathbf{M})][i,j] \\
&= \sum_j \mathbf{I}[i,j] - \mathbf{M}[i,j] - \mathbf{M}[j,i] + (\mathbf{M}^{\mathsf{T}}\mathbf{M})[i,j] \\
&= 1 - \sum_j \mathbf{M}[i,j] - \sum_j \mathbf{M}[j,i] + \sum_j \sum_k \mathbf{M}[k,i] \cdot \mathbf{M}[k,j] \\
&= 1 - 1 - \sum_j \mathbf{M}[j,i] + \sum_k \mathbf{M}[k,i] \\
&= 0.
\end{aligned} \tag{14}$$

Therefore, the matrix $\mathbf{L}^{'}$ can be considered as Laplacian matrix of a graph. If we set the entry of similarity matrix $\mathbf{W}[i,j] = (\mathbf{M} + \mathbf{M}^{\mathsf{T}} - \mathbf{M}^{\mathsf{T}}\mathbf{M})[i,j]$, obtaining low-dimensional embedding in LLE is equivalent to obtaining the graph embedding of graph $\boldsymbol{G}^{'}$ whose Laplacian matrix is $\mathbf{L}^{'}$. Thus, $\mathbf{B} = \mathbf{D} = \mathbf{I}_N$ and $\mathbf{A}$ is set to $\mathbf{I}_N$.

(4) Computing the low-dimensional embedding. The low-dimensional representation in LLE can be calculated by solving the generalized eigenvalue problem given in Eq. (9).

There are two hyper-parameters k and m needed to be set for LLE on the above 4 steps, which will be discussed in Section 5.3. The complexity of computing reconstruction weight matrix is $\mathcal{O}(nNk^3)$. Like LE, computing low-dimensional embedding also takes $\mathcal{O}\left(mN^2\right)$ operations.

3.2.3 ISOMAP

ISOMAP [30] is a representative of isometric mapping methods and maintains the global structure of the original data. Euclidean distance used in MDS [34] cannot exactly represent the distance relationship between power traces (i.e., nodes) on manifold structure. For example, it is clearly not appropriate to directly use the Euclidean distance between two nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ to present their distance on the curved manifold surface. Here geodesic distance can correctly reflect this shortest distance. In this sense, ISOMAP improves MDS by using geodesic distance instead of Euclidean distance.

The procedures of ISOMAP are as follows:

(1) Neighbors selection. Similar to LE, the first step of ISOMAP is also to find the k nearest neighbors for each node. The procedure of neighbors selection is aligned with LE.

(2) Calculate the approximate geodesic distance. Geodesic distance is typically approximated by the shortest path between two nodes. However, the exact value of the geodesic distance is difficult to calculate. Therefore, we initialize the approximate geodesic distance matrix as $\mathbf{D}_G = \{\mathbf{D}_G[i,j]\}_{N\times N}$, where $\mathbf{D}_G[i,j] = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|$ for two adjacent nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ and $\mathbf{D}_G[i,j] = \infty$ otherwise. We then perform Floyd or Dijkstra algorithm on adjacency graph $\boldsymbol{G}$ to find the shortest paths as the approximation of geodesic distance.

(3) Computing reformulation matrix. ISOMAP uses MDS to obtain the low-dimensional embedding that maintains the essential geometry of power traces. It converts the approximate geodesic distance matrix into the corresponding inner product matrix by the following function:

$$\tau(\mathbf{S}) = -\frac{\mathbf{HSH}}{2}. \tag{15}$$

Here centralization matrix $\mathbf{H} = \{\mathbf{H}[i,j]\}_{N\times N}$ and $\mathbf{H}[i,j] = \delta_{ij} - \frac{1}{N}$. Here $\delta_{ij} = 1$ for $i = j$. Otherwise, $\delta_{ij} = 0$. Let $\mathbf{D}_Y$ denote the matrix of Euclidean distance between nodes after dimensionality reduction, i.e., $\mathbf{D}_Y[i,j] = \|\boldsymbol{y}_i - \boldsymbol{y}_j\|$. ISOMAP aims to find an embedding that preserves the distance information between nodes, thus ISOMAP minimizes the following objective function:

$$\psi(\mathbf{Y}) = \|\tau(\mathbf{D}_G) - \tau(\mathbf{D}_Y)\|_{L^2}, \tag{16}$$

where $\|\mathbf{A}\|_{L^2} = \sqrt{\sum_{ij}(\mathbf{A}[i,j])^2}$ and $\mathbf{A}$ is an arbitrary matrix. As proved in [33], $\tau(\mathbf{D}_G)$ can be considered as the Laplacian matrix of a graph. If we set the entry of similarity matrix $\mathbf{W}[i,j] = \tau(\mathbf{D}_G)[i,j]$ for $i \neq j$ and $\mathbf{W}[i,j] = 0$ otherwise, obtaining low-dimensional embedding in ISOMAP is equivalent to obtaining the graph embedding of graph G whose Laplacian matrix is $\tau(\mathbf{D}_G)$. Thus, $\mathbf{B}$ equals to the identity matrix $\mathbf{I}_N$. Unlike LE and LLE, matrix $\mathbf{A}$ is set to a diagonal matrix composed of the eigenvalues of $\tau(\mathbf{D}_G)$.

(4) Computing the low-dimensional embedding. The low-dimensional representation in ISOMAP can be calculated by solving the generalized eigenvalue problem in Eq. (9).

Like LLE, there are two hyper-parameters k and m needed to be set for ISOMAP, we will discuss them in Section 5.3. The complexity is $\mathcal{O}(kN + N\log N)$ when using Dijkstra algorithm to find the shortest path between two power traces. If the shortest paths between any two power traces are calculated, the complexity is $\mathcal{O}(N^2(k + \log N))$, where k is the number of nearest neighbors, and N is the number of power traces we use in an attack. Computing low-dimensional embedding takes $\mathcal{O}\left(mN^2\right)$ operations.

4 Explicit Mapping for Manifold Learning Methods

As discussed in Section 2.3, the selection of POIs is a critical task in higher-order power attacks. Fortunately, there are several ways to bypass such a difficult task. Instead of directly finding accurate POIs, dimensionality reduction techniques can greatly reduce the dimension of power traces and achieve feature extraction. Then, power side-channel attacks can be applied to the dimension-reduced sample points. In this way, dimensionality reduction works as a extractor, which aims to find a way to effectively combine the components of sample points. When this comes to manifold learning, the power side-channel leakage can be simply regarded as the observation of this manifold structure. The intrinsic dimension of the side-channel leakage is always much smaller than the corresponding dimension of power traces. However, a significant drawback if we directly apply the original manifold learning techniques into side-channel attacks is: no explicit mapping between high-dimensional sample points and low-dimensional representations. The complexity of LE, LLE and ISOMAP given in Section 3 is very high in this case, just like KDA. To reduce the complexity, with the help of well trained manifold learning "models", we can try to find the corresponding low-dimensional representation of each newly coming power trace by explicit mapping, and a solution will be given in this section.

4.1 Explicit Mapping in Manifold Learning Methods

A explicit mapping solution for manifold learning methods is to assume that there is a certain mapping relationship between the power traces and their low-dimensional representation. In the training stage, the explicit mapping solution "trains" the mapping relationship. After the assumed mapping relationship is determined, the remaining power traces can be mapped into their low-dimensional representation according to this assumption. This can facilitate both the profiling and attacks. By using the explicit mapping solution, we are able to rapidly gain approximate low-dimensional representation corresponding to the power traces. Linear assumption, for instance, suggests that there is a linear mapping between high- and low-dimensional representations:

$$\boldsymbol{y}_i = \mathbf{U}\boldsymbol{x}_i, \quad \mathbf{U} \in \mathbb{R}^{n \times N}. \tag{17}$$

Here $\mathbf{U}$ is a unique projection matrix corresponding to the linear transformation from power trace $\boldsymbol{x}_i$ to its low-dimensional representation $\boldsymbol{y}_i$. Based on this linear assumption, a lot of methods have been proposed, such as Locality Preserving Projections (LPP) [35], Neighborhood Preserving Embedding (NPE) [36] and Neighborhood Preserving Projections (NPP [37]. These projections can be exploited for unprotected implementations with linear leakage.

To deal with high-dimensional nonlinear leakage for higher-order masked implementations, a nonlinear mapping assumption is essential. Hong et al. [38] assumed that the k-th component $\boldsymbol{y}_i[k]$ of $\boldsymbol{y}_i$ was a polynomial of degree p in $\boldsymbol{x}_i$ in the following manner:

$$\boldsymbol{y}_i[k] = \sum_{q=1}^{p}\sum_{j=1}^{n} \boldsymbol{v}_k[q,j]\left(\boldsymbol{x}_i[j]\right)^q . \tag{18}$$

Here q stands for exponent of the component of $\boldsymbol{x}_i$, $\boldsymbol{v}_k$ is the vector of polynomial coefficients indexed by the q and j, which is defined as:

$$\boldsymbol{v}_k = \begin{pmatrix} \boldsymbol{v}_k[q=p, j=n] \\ \vdots \\ \boldsymbol{v}_k[q=2, j=1] \\ \boldsymbol{v}_k[q=1, j=n] \\ \vdots \\ \boldsymbol{v}_k[q=1, j=1] \end{pmatrix}.$$

By assuming the polynomial mapping relationship, we aim to find a polynomial approximation to the implicit mapping from the high-dimensional power traces to their low-dimensional embedding representation. Compared with the linear assumption previously used, a polynomial mapping is more accurate for nonlinear leakage from masked implementations.

4.2 Polynomial Mapping Framework

By applying the polynomial assumption to the framework of manifold learning, Eq. (8) can be derived to:

$$\begin{aligned} &\min_{\boldsymbol{v}_k} \sum_k \boldsymbol{v}_k^\intercal \mathbf{X}_p \mathbf{L} \mathbf{X}_p \boldsymbol{v}_k, \\ &\text{s.t.} \quad \boldsymbol{v}_j^\intercal \mathbf{X}_p \mathbf{B} \mathbf{X}_p \boldsymbol{v}_k = \delta_{jk} \end{aligned} \tag{19}$$

as described in [38]. Here $\delta_{jk} = 1$ for $j = k$. Otherwise, $\delta_{jk} = 0$. $\mathbf{X}_p = \left[\ \boldsymbol{x}_1^p, \boldsymbol{x}_2^p, \ldots, \boldsymbol{x}_N^p\ \right]$ and $\boldsymbol{x}_i^p$ is defined as:

$$\boldsymbol{x}_i^p = \begin{pmatrix} \overbrace{\boldsymbol{x}_i \odot \boldsymbol{x}_i \odot \cdots \odot \boldsymbol{x}_i}^{p} \\ \vdots \\ \boldsymbol{x}_i \odot \boldsymbol{x}_i \\ \boldsymbol{x}_i \end{pmatrix}, \tag{20}$$

where "$\odot$" stands for Hadamard product. For arbitrary matrices $\mathbf{A} = \{\mathbf{A}[i,j]\}_{m\times n}$ and $\mathbf{B} = \{\mathbf{B}[i,j]\}_{m\times n}$, $(\mathbf{A} \odot \mathbf{B})[i,j] = \mathbf{A}[i,j] \cdot \mathbf{B}[i,j]$. As discussed in the general framework of manifold learning in Section 3.1, the optimal problem in Eq. (19) can be converted to the corresponding generalized eigenvalue problem:

$$\mathbf{X}_p \mathbf{L} \mathbf{X}_p^\intercal \boldsymbol{v}_i = \lambda \mathbf{X}_p \mathbf{B} \mathbf{X}_p^\intercal \boldsymbol{v}_i. \tag{21}$$

Here λ refers to the eigenvalue of the Laplacian matrix $\mathbf{L}$. Once $\boldsymbol{v}_i$-s $(i = 1, \ldots, m)$ are computed, the explicit nonlinear mapping from the high-dimensional power traces to the low-dimensional embedding representation can be given by Eq. (18).

In summary, the explicit polynomial mapping of manifold learning methods consists of two stages: the training stage and mapping stage. In the training stage, we use a set of power traces $\mathbf{X}$ to train the explicit manifold model, i.e., to calculate polynomial coefficients $\boldsymbol{v}_i$-s $(i = 1, 2, \ldots, m)$. We can summarize the procedures of the training stage of explicit polynomial mapping of manifold leaning methods in Alg. 1. Specifically, we first compute the similarity matrix $\mathbf{W}$ according to specific manifold learning method (as discussed in Section 3.2) in Step 1 of Alg. (1). We then compute the corresponding constraint matrix $\mathbf{B}$ in Step 2. We then generate $\mathbf{X}_p$ according to Eq. (20) in Step 3, finally solve the generalized eigenvalue problem in Eq. (21) and get $\boldsymbol{v}_i$ $(i = 1, 2, \ldots, m)$ in Step 4.

Algorithm 1: The Training Stage of Explicit Polynomial Mapping of Manifold Leaning Methods.

```
Input: Power traces $\mathbf{X}$ used for training.
Output: Polynomial coefficients $\boldsymbol{v}_i$-s ($i = 1, \dots, m$).
1 Compute similarity matrix $\mathbf{W}$:
    For LE:
      $$\mathbf{W}[i,j] = \begin{cases} e^{-\frac{\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2}{t}}, & \text{if nodes } \boldsymbol{x}_i \text{ and } \boldsymbol{x}_j \text{ are adjacent} \\ 0, & \text{otherwise.} \end{cases}$$
    For LLE:
      $$\mathbf{W}[i,j] = (\mathbf{M} + \mathbf{M}^\intercal - \mathbf{M}^\intercal \mathbf{M})[i,j].$$
    For ISOMAP:
      $$\mathbf{W}[i,j] = \tau\left(\mathbf{D}_G\right)[i,j].$$
2 Compute the corresponding constraint matrix $\mathbf{B}$:
    For LE:
      $$\mathbf{B} = \mathbf{D}.$$
    For LLE and ISOMAP:
      $$\mathbf{B} = \mathbf{I}_N.$$
3 Generate $\mathbf{X}_p$ according to Eq. (20).
4 Solve the generalized eigenvalue problem in Eq. (21), and get $\boldsymbol{v}_i$-s ($i = 1, 2, \dots, m$).
```

After we obtain the polynomial coefficients $\boldsymbol{v}_i$-s ($i = 1, 2, \dots, m$) in the training stage, the mapping relationship between a power trace and its low-dimensional representation can be determined. For any power trace $\boldsymbol{x}$, its low-dimensional representation $\boldsymbol{y}$ can be directly and rapidly calculated as:

$$\boldsymbol{y} = \begin{pmatrix} \sum_{q=1}^{p} \sum_{j=1}^{n} \boldsymbol{v}_1[q,j]\left(\boldsymbol{x}[j]\right)^q \\ \vdots \\ \sum_{q=1}^{p} \sum_{j=1}^{n} \boldsymbol{v}_m[q,j]\left(\boldsymbol{x}[j]\right)^q \end{pmatrix}. \tag{22}$$

according to Eq. (18). It is noteworthy that, as we have mentioned before, the explicit polynomial mapping can be used for both profiling and attack in a profiled attack. Specifically, the low-dimensional representation of profiling power traces can be used to construct templates, and the low-dimensional representation of the attack traces can be used for key recovery in the attack stage.

4.3 Complexity Analysis

Here we analyze the complexity of the whole procedures of explicit polynomial mapping. In the training stage, the complexity of generating $\mathbf{X}_p$ in Eq. (20) is $\mathcal{O}\left(Nnp\right)$. The complexity of solving the eigenvalue problem in Eq. (21) is $\mathcal{O}\left(mn^2p^2\right)$. Mapping a power trace into the low-dimensional embedding takes $\mathcal{O}\left(mnp\right)$ operations. Comparing the complexity of manifold learning methods with and without explicit polynomial mapping, we can find that the complexity of explicit mapping is close to its original manifold learning method in the training stage. However, once the explicit mapping relationship is established, the complexity of mapping a power trace is much lower than the original manifold learning methods, since m and p are usually much smaller than N. This advantage becomes more significant in profiled attacks especially for masked implementations, since they usually require a large number of power traces for profiling, and the computation of low-dimensional representations of traces through original manifold learning methods is computationally expensive. Besides, the explicit mapping solution is more suitable to cope with the attack traces in the attack stage of a profiled attack. The explicit mapping is able to directly map attack traces into their low-dimensional representations, instead of re-computing the mapping relationship in the original manifold learning methods.

5 Experiments

In this section, we show that how manifold learning techniques work when coping with the first-order masked AES implementation, experimentally test the influence of the hyper-parameters, and finally compare their performance with KDA, MLP and CNN.

5.1 Experimental Setups

Our experiments are performed on one of the most popular open power trace set called ASCAD datasets [29] that aim at providing a bench-marking reference for the SCA community. The ASCAD datasets comprise several versions depending on the underlying implementation and architecture. For convenience, we use the fixed key version, which contains 50, 000 profiling traces and 10, 000 attack traces sampled from an 8-bit AVR microprocessor ATMega328P running a first-order Boolean masked AES-128 implementation. Here the classical table re-computation method is used. The targeted intermediate value is the output of the third masked Sbox during the first round of AES. Each trace consists of 700 sample points and is labeled by the unmasked Sbox output. Some necessary metadata, such as plaintexts, real keys and masks, are also provided.

In our experiments, we test whether our proposed manifold learning schemes can be able to extract features from higher-order non-linear leakage and facilitate the attack in the first order. Specifically, we first train the explicit manifold learning model, i.e., the polynomial mapping between the power traces and their low-dimensional representation. A total number of 12, 800 power traces are used for this training, and each intermediate value includes 50 traces. Then, we can map the remaining profiling power traces in ASCAD dataset into a m-dimensional representation through the explicit mapping, where $m \ll n = 700$. We then profile the templates using these dimension-reduced power traces, and perform a multivariate template attack. Guessing Entropy (GE) [39], which refers to the index of the correct sub-key in the vector that consists of all guessing values ranked by their likelihood scores in descending order, is exploited as a metric in evaluations.

5.2 Template Construction

In the profiling phase, we describe the characteristics of dimension-reduced power traces by multivariate Gaussian distribution, i.e., Gaussian templates. Generally, there are several ways to construct templates. A popular choice is to construct templates according to the intermediate value, i.e., the output of Sbox $z = \mathrm{Sbox}(\tau \oplus \kappa)$. For AES-128, we can construct 256 templates and each template corresponds to a possible intermediate value. In addition, we can also take the power leakage characteristics of target implementation into consideration. The Hamming weight model is one of the most widely used power leakage model in side-channel analysis. If the target implementation leaks the Hamming weight information of data, moving two different intermediate values with the same Hamming weight to a register will lead to similar power consumption. In this case, we can simply construct 9 templates according to the Hamming weight of the intermediate value (i.e., $\mathrm{HW}(z)$).

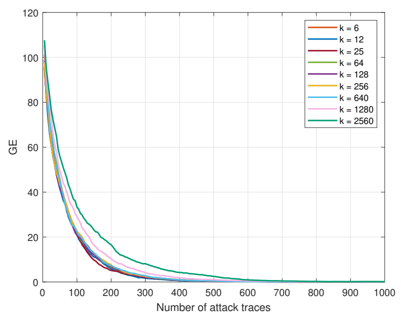
5.3 Hyper-parameters Evaluation

This section concentrates on the evaluation of several critical hyper-parameters for manifold learning techniques LE, LLE and ISOMAP. Besides better understanding their influence on the attack performance, we hope the detailed evaluations could serve as guidelines for potential users to apply the manifold learning techniques. We test the influence of the number of nearest neighbors k, the target dimension m and the weight coefficient t in Gaussian kernel of LE. We set the polynomial degree in Eq. (18) to 2 and the number of traces to construct template to $50,000$. To avoid performance fluctuation, each set of hyper-parameters is repeated for 500 times in the attack phase. The power traces are independently and randomly selected from ASCAD dataset in each iteration.

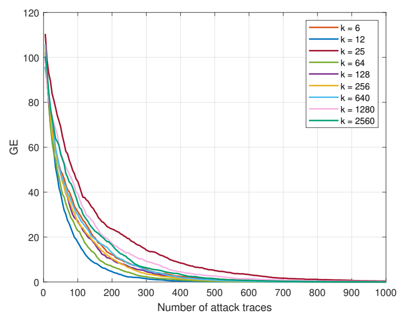
To evaluate the effect of hyper-parameters, we first conduct a preliminary test to select a base set of hyper-parameters. It is worth noting that although all three techniques LE, LLE and ISOMAP have a hyper-parameter in k-NN, i.e., the number of the nearest neighbors k. However, the mathematical principles of them are different. Specifically, ISOMAP exploits the nearest neighbors to find the shortest path between nodes. For LLE, k represents that each node is approximated by its k nearest neighbors. For LE, we maintain the adjacent relationship among each node and its k nearest neighbors. This makes their optimal value of k different. In detail, k is set to 25, 12, and 256 for LE, LLE and ISOMAP respectively, since they work almost the optimal in our experiments. $m = 10$ for all of them and $t = 10^5$ for LE. In each subsection, we fix the other hyper-parameters and vary a hyper-parameter in a reasonable range to investigate its influence. For the sake of convenience, we denote the explicit polynomial mapping of LE, LLE and ISOMAP as ELE, ELLE and EISOMAP in the later sections.

5.3.1 k-Nearest Neighbors

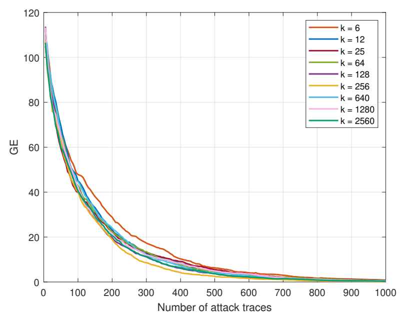
The number of nearest neighbors k is a critical parameter in manifold learning. As discussed in Section 3.2, we select k nearest neighbors for each node (power trace) in the first step of ELE, ELLE and EISOMAP. k should be kept in a reasonable range that not only maintains the connection between neighbors, but also retains the global structure of manifolds.



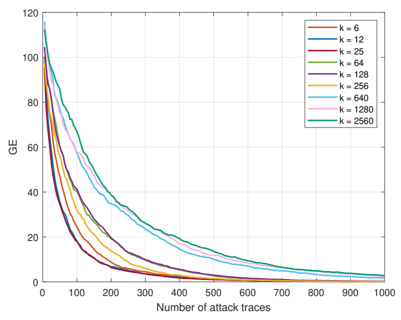
(a) ELE with 9-class TA



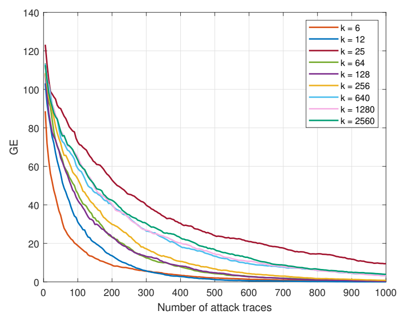
(b) ELLE with 9-class TA



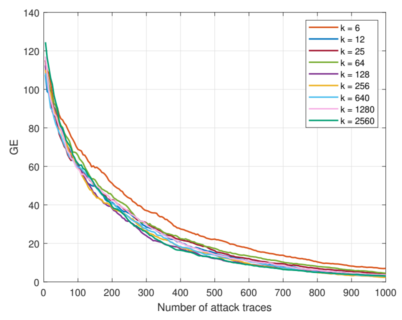
(c) EISOMAP with 9-class TA



(d) ELE with 256-class TA



(e) ELLE with 256-class TA



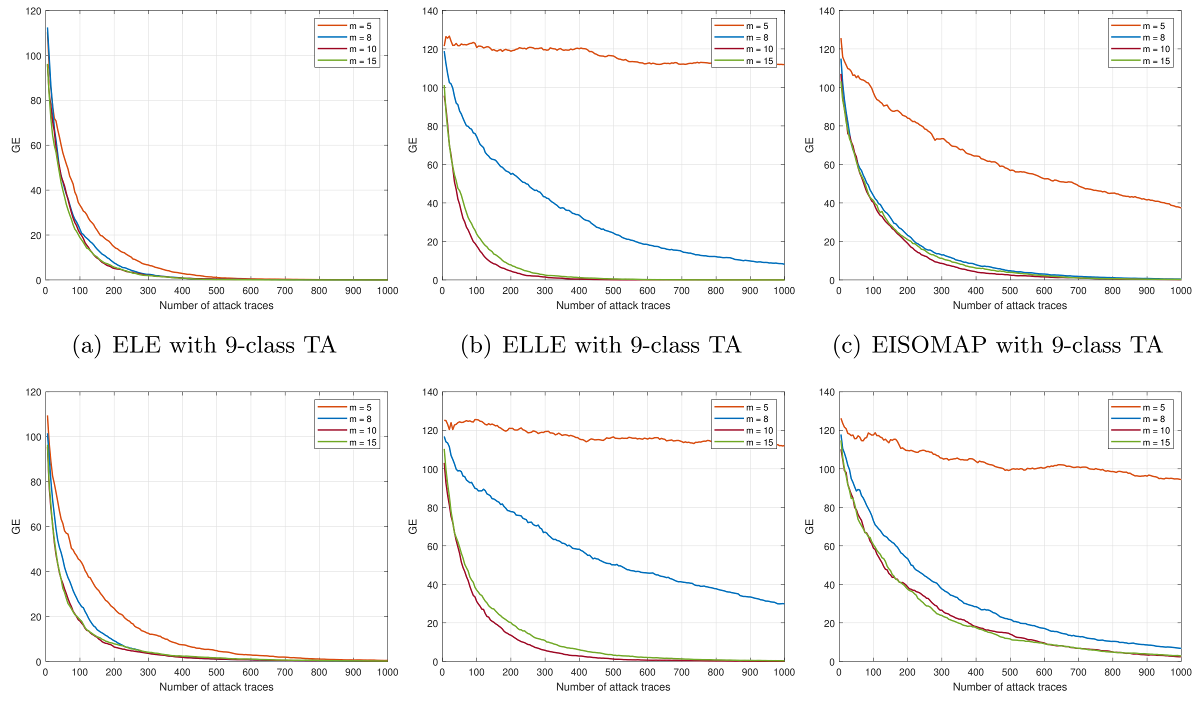
(f) EISOMAP with 256-class TA

Figure 3: The effect of the parameter k in k-NN.

We vary k to investigate its influence on the attack performance, and the corresponding experimental results of three manifold learning techniques ELE, ELLE and EISOMAP are shown in Fig. 3. Obviously, all three manifold learning techniques successfully perform their first-order attacks on the ASCAD dataset. As the number of attack traces increases, the corresponding GE decreases rapidly and finally converges to 0. TA with 9 classes outperforms those with 256 classes when k is optimal. Besides, the effect of k is more prominent when applying 256-class TA. For ELE and EISOMAP, $k = 25$ and 256 are optimal respectively. For ELLE, $k = 12$ (6) is optimal when applying 9-class (256-class) TA.

5.3.2 Target Dimension

The target dimension m determines the number of eigenvectors selected when solving the generalized eigenvalue problem in Eq. (21). We vary m from 5 to 15, and the corresponding guessing entropy of three manifold learning techniques ELE, ELLE and EISOMAP is shown in Fig. 4. Obviously, all these three manifold techniques work very well and successfully perform first-order attacks when m is set to 8 or 15. The increase of m first leads to higher correlation between dimension-reduced traces and Hamming weights of intermediate values. However, the increase of m also means larger cost in the attack phase. Their performance becomes stable when $m > 10$. This indicates that it is difficult to significantly improve their performance by continuing to increase m.



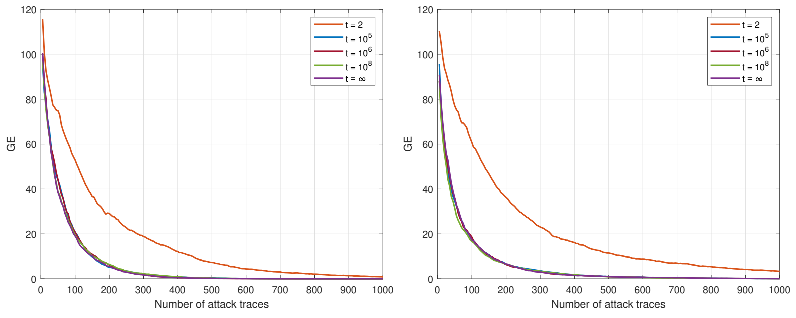
(a) ELE with 9-class TA (b) ELLE with 9-class TA (c) EISOMAP with 9-class TA

(d) ELE with 256-class TA (e) ELLE with 256-class TA (f) EISOMAP with 256-class TA

Figure 4: The effect of target dimension m.

5.3.3 Weight Coefficient

Weight coefficient t in Gaussian kernel is a unique hyper-parameter of ELE, which measures the difference between different pairs of adjacent nodes in the view of distance. The method for setting t is discussed in the second step of LE in Section 3.2.1. We test several values of t in our experiments, including the case of $t = \infty$, where $\mathbf{W}[i,j]$ is simply set to 1 if nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are adjacent and 0 otherwise. The experimental results are shown in Fig. 5. Obviously, t has a limited effect on the attack performance for the most of cases, except for $t = 2$ whose performance is relatively poor. Therefore, the adversary could flexibly set t (e.g., 10^5) to get a remarkable performance, or simply let $t = \infty$ for the sake of reducing complexity.



(a) ELE with 9-class TA (b) ELE with 256-class TA

Figure 5: The effect of weight coefficient t.

5.4 Comparisons

With the help of hyper-parameters evaluation in Section 5.3, here to further illustrate the superiority of our schemes, we compare ELE, ELLE and EISOMAP with KDA and the deep learning techniques MLP and CNN. All the hyper-parameters in manifold learning are set to be nearly optimal as discussed in Section 5.3. Specifically, in the training stage, the number of nearest neighbors k is set to 25, 12, and 256 for LE, LLE and ISOMAP respectively, m is set to 10 for all of them, and t is set to 10^5 for LE. Here 9-class TA is applied. For KDA, we also experimentally test each hyper-parameter and select the one with the best attack performance. For MLP and CNN, the trained models, which are claimed to be best in [29], are used in our experiments. As our previous experiments did, each attack here is repeated for 500 times, and the results are shown in Fig. 6.

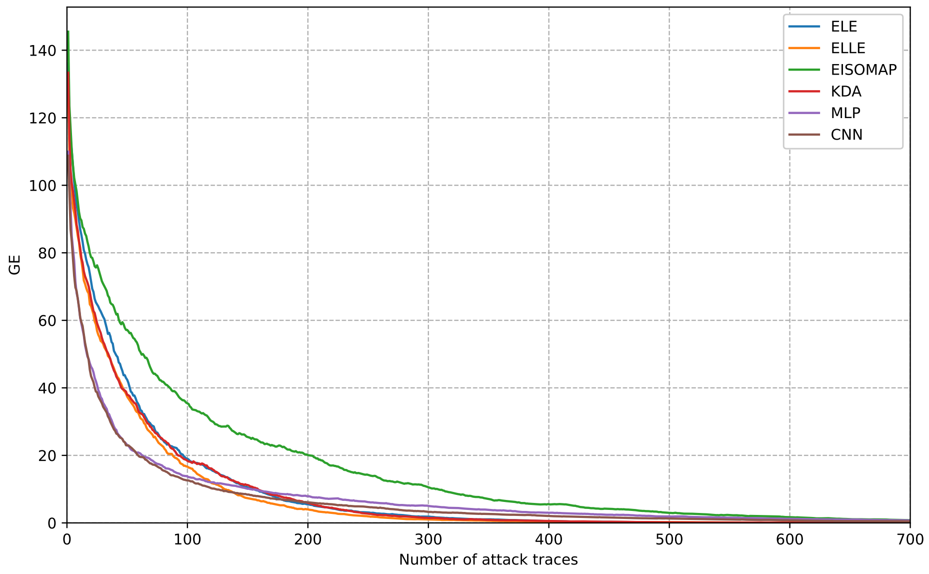


Figure 6: Comparison between manifold learning techniques (ELE, ELLE and EISOMAP), KDA and deep learning techniques (CNN and MLP).

The attack performance of KDA and ELE in Fig. 6 are close, and ELLE has a outstanding performance in the case of the number of attack traces is large than 100. Noting that MLP and CNN work better

Table 1: Guessing entropy in attacks with manifold learning techniques (ELE, ELLE and EISOMAP) and deep learning techniques (MLP and CNN).

Techniques	GE < 10	GE < 5	GE < 1
ELE	155	206	354
ELLE	130	**181**	**301**
EISOMAP	307	422	674
KDA	158	210	343
MLP	152	233	657
CNN	**123**	300	549

than ELE and ELLE when the number of attack traces is approximately less than 150. As the number of attack traces increases, ELE and ELLE catch up from behind and outperform. Compared with MLP and CNN, manifold learning techniques with sufficient attack traces are in the lead. To better represent the efficiency of manifold learning techniques, we list the required minimum number of attack traces to reach certain values of GE in Table 1. Obviously, ELLE requires the smallest number of attack traces to fulfill the last two conditions, fully illustrating its effectiveness in attack.

By varying the parameters in the three manifold learning techniques ELE, ELLE and EISOMAP within a certain range, we find that the attack performance does not change rapidly. However, it is very difficult to find a set of good parameters (e.g., optimizer, learning rate and architecture parameters) to achieve a successful attack for MLP and CNN in experiments. Therefore, manifold learning schemes are more robust when hyper-parameters are poorly set. In other words, hyper-parameters are easier to set than deep learning-based schemes such as MLP and CNN. These conclusions fully illustrate the superiority of our manifold learning techniques.

6 Conclusions and Future Works

Information extraction for masked implementation is a crucial task in power side channel attacks and evaluations. KDA and deep learning techniques such as CNN and MLP, make the dimensionality reduction of high-dimensional nonlinear leakage samples practical. However, KDA is noise-sensitive and low efficient, and deep learning techniques suffer from high computation cost and complex hyper-parameters adjustment. This paper undertook the first study on manifold learning power side-channel attacks and their applications on higher-order masking. Manifold learning techniques such as ISOMAP, LLE and LE, were exploited to reduce the dimension of high-dimensional non-linear leakage on power traces to their intrinsic dimensionality representation. Moreover, we presented the nonlinear explicit mapping of manifold learning to reduce the complexity of mapping. Compared to KDA, manifold learning is much more efficient and unsupervised. Compared to deep learning techniques such as MLP and CNN, manifold learning takes less time, requires fewer parameters to be set and is more robust when the hyper-parameters are poorly set. These advantages make manifold learning meaningful when attacking masked implementations.

This paper provided the first attempt on manifold learning against masked implementations, and well demonstrated its significant performance. This study promotes the understanding and usage of dimensionality reduction in higher-order power side-channel attacks and provides a reference for leakage evaluations. There are still several interesting topics in our future works. For instance, manifold learning techniques against much higher-order masked implementations, and better representation of adjacency relationships. We believe that manifold learning techniques will bring more surprises to the analysis and evaluations on masked cryptographic implementations.

References

[1] Junrong Liu, Yu Yu, François-Xavier Standaert, Zheng Guo, Dawu Gu, Wei Sun, Yijie Ge, and Xinjun Xie. Small Tweaks Do Not Help: Differential Power Analysis of MILENAGE Implementations in 3G/4G USIM Cards. In Computer Security - ESORICS 2015 - 20th European Symposium on Research in Computer Security, Vienna, Austria, September 21-25, 2015, Proceedings, Part I, volume 9326 of Lecture Notes in Computer Science, pages 468–480. Springer, 2015.

[2] Daniel Genkin, Adi Shamir, and Eran Tromer. RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis. In Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I, volume 8616 of Lecture Notes in Computer Science, pages 444–461. Springer, 2014.

[3] Yuval Yarom, Daniel Genkin, and Nadia Heninger. CacheBleed: A Timing Attack on OpenSSL Constant Time RSA. In Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings, volume 9813 of Lecture Notes in Computer Science, pages 346–367. Springer, 2016.

[4] Pierre Belgarric, Pierre-Alain Fouque, Gilles Macario-Rat, and Mehdi Tibouchi. Side-Channel Analysis of Weierstrass and Koblitz Curve ECDSA on Android Smartphones. In Topics in Cryptology - CT-RSA 2016 - The Cryptographers' Track at the RSA Conference 2016, San Francisco, CA, USA, February 29 - March 4, 2016, Proceedings, volume 9610 of Lecture Notes in Computer Science, pages 236–252. Springer, 2016.

[5] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings, volume 1666 of Lecture Notes in Computer Science, pages 388–397. Springer, 1999.

[6] 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, volume 3156 of Lecture Notes in Computer Science, pages 16–29. Springer, 2004.

[7] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template Attacks. In Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers, volume 2523 of Lecture Notes in Computer Science, pages 13–28. Springer, 2002.

[8] Kai Schramm, Thomas J. Wollinger, and Christof Paar. A New Class of Collision Attacks and Its Application to DES. In Fast Software Encryption, 10th International Workshop, FSE 2003, Lund, Sweden, February 24-26, 2003, Revised Papers, volume 2887 of Lecture Notes in Computer Science, pages 206–222. Springer, 2003.

[9] Maxime Nassar, Youssef Souissi, Sylvain Guilley, and Jean-Luc Danger. RSM: A small and fast countermeasure for AES, secure against 1st and 2nd-order zero-offset SCAs. In 2012 Design, Automation & Test in Europe Conference & Exhibition, DATE 2012, Dresden, Germany, March 12-16, 2012, pages 1173–1178. IEEE, 2012.

[10] Jean-Sébastien Coron, Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. Higher-Order Side Channel Security and Mask Refreshing. In Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers, volume 8424 of Lecture Notes in Computer Science, pages 410–424. Springer, 2013.

[11] Elisabeth Oswald, Stefan Mangard, Christoph Herbst, and Stefan Tillich. Practical Second-Order DPA Attacks for Masked Smart Card Implementations of Block Ciphers. In Topics in Cryptology - CT-RSA 2006, The Cryptographers' Track at the RSA Conference 2006, San Jose, CA, USA, February 13-17, 2006, Proceedings, volume 3860 of Lecture Notes in Computer Science, pages 192–207. Springer, 2006.

[12] Unai Rioja, Lejla Batina, Jose Luis Flores, and Igor Armendariz. Auto-tune POIs: Estimation of distribution algorithms for efficient side-channel analysis. Comput. Networks, 198:108405, 2021.

[13] François Durvaux, François-Xavier Standaert, Nicolas Veyrat-Charvillon, Jean-Baptiste Mairy, and Yves Deville. Efficient Selection of Time Samples for Higher-Order DPA with Projection Pursuits. In Constructive Side-Channel Analysis and Secure Design - 6th International Workshop, COSADE 2015, Berlin, Germany, April 13-14, 2015. Revised Selected Papers, volume 9064 of Lecture Notes in Computer Science, pages 34–50. Springer, 2015.

[14] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Kernel Discriminant Analysis for Information Extraction in the Presence of Masking. In Smart Card Research and Advanced Applications - 15th International Conference, CARDIS 2016, Cannes, France, November 7-9, 2016, Revised Selected Papers, volume 10146 of Lecture Notes in Computer Science, pages 1–22. Springer, 2016.

[15] Cédric Archambeau, Eric Peeters, François-Xavier Standaert, and Jean-Jacques Quisquater. Template Attacks in Principal Subspaces. In Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings, volume 4249 of Lecture Notes in Computer Science, pages 1–14. Springer, 2006.

[16] Nicolas Bruneau, Sylvain Guilley, Annelie Heuser, Damien Marion, and Olivier Rioul. Less is More - Dimensionality Reduction from a Theoretical Perspective. In Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings, volume 9293 of Lecture Notes in Computer Science, pages 22–41. Springer, 2015.

[17] Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. Breaking Cryptographic Implementations Using Deep Learning Techniques. In Security, Privacy, and Applied Cryptography Engineering - 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings, volume 10076 of Lecture Notes in Computer Science, pages 3–26. Springer, 2016.

[18] Xiangjun Lu, Chi Zhang, Pei Cao, Dawu Gu, and Haining Lu. Pay Attention to Raw Traces: A Deep Learning Architecture for End-to-End Profiling Attacks. IACR Trans. Cryptogr. Hardw. Embed. Syst., 2021(3):235–274, 2021.

[19] Loïc Masure, Nicolas Belleville, Eleonora Cagli, Marie-Angela Cornelie, Damien Couroussé, Cécile Dumas, and Laurent Maingault. Deep Learning Side-Channel Analysis on Large-Scale Traces - A Case Study on a Polymorphic AES. In Computer Security - ESORICS 2020 - 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14-18, 2020, Proceedings, Part I, volume 12308 of Lecture Notes in Computer Science, pages 440–460. Springer, 2020.

[20] Lichao Wu, Guilherme Perin, and Stjepan Picek. The Best of Two Worlds: Deep Learning-assisted Template Attack. IACR Trans. Cryptogr. Hardw. Embed. Syst., 2022(3):413–437, 2022.

[21] Xiang Li, Ning Yang, Aidong Chen, Weifeng Liu, Xiaoxiao Liu, and Na Huang. Power Analysis Attack Based on Lightweight Convolutional Neural Network. In Frontiers in Cyber Security - 5th

International Conference, FCS 2022, Kumasi, Ghana, December 13-15, 2022, Proceedings, volume 1726 of Communications in Computer and Information Science, pages 105–118. Springer, 2022.

[22] Pei Cao, Chi Zhang, Xiangjun Lu, Dawu Gu, and Sen Xu. Improving Deep Learning Based Second-Order Side-Channel Analysis With Bilinear CNN. IEEE Trans. Inf. Forensics Secur., 17:3863–3876, 2022.

[23] Guilherme Perin, Lichao Wu, and Stjepan Picek. Exploring Feature Selection Scenarios for Deep Learning-based Side-channel Analysis. IACR Trans. Cryptogr. Hardw. Embed. Syst., 2022(4):828–861, 2022.

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States, pages 1106–1114, 2012.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pages 5998–6008, 2017.

[26] Awni Y. Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. Deep Speech: Scaling up end-to-end speech recognition. CoRR, abs/1412.5567, 2014.

[27] Zdenek Martinasek, Petr Dzurenda, and Lukas Malina. Profiling power analysis attack based on MLP in DPA contest V4.2. In 39th International Conference on Telecommunications and Signal Processing, TSP 2016, Vienna, Austria, June 27-29, 2016, pages 223–226. IEEE, 2016.

[28] Amit Garg and Nima Karimian. Leveraging Deep CNN and Transfer Learning for Side-Channel Attack. In 22nd International Symposium on Quality Electronic Design, ISQED 2021, Santa Clara, CA, USA, April 7-9, 2021, pages 91–96. IEEE, 2021.

[29] Emmanuel Prouff, Rémi Strullu, Ryad Benadjila, Eleonora Cagli, and Cécile Dumas. Study of Deep Learning Techniques for Side-Channel Analysis and Introduction to ASCAD Database. IACR Cryptol. ePrint Arch., page 53, 2018.

[30] Tenenbaum Joshua B, Silva Vin de, and Langford John C. A global Geometric Framework for Nonlinear Dimensionality Reduction. Science, 15(5500):2319–2323, 2000.

[31] Sam T. Roweis and Lawrence K. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. Science, 290(5500):2323–2326, 2000.

[32] Mikhail Belkin and Partha Niyogi. Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. Neural Comput., 15(6):1373–1396, 2003.

[33] Shuicheng Yan, Dong Xu, Benyu Zhang, HongJiang Zhang, Qiang Yang, and Stephen Lin. Graph Embedding and Extensions: A General Framework for Dimensionality Reduction. IEEE Trans. Pattern Anal. Mach. Intell., 29(1):40–51, 2007.

[34] Ingwer Borg and Patrick JF Groenen. Modern Multidimensional Scaling: Theory and Applications. Springer Science & Business Media, 2005.

[35] Xiaofei He and Partha Niyogi. Locality Preserving Projections. In Advances in Neural Information Processing Systems 16 [Neural Information Processing Systems, NIPS 2003, December 8-13, 2003, Vancouver and Whistler, British Columbia, Canada], pages 153–160. MIT Press, 2003.

[36] Xiaofei He, Deng Cai, Shuicheng Yan, and HongJiang Zhang. Neighborhood Preserving Embedding. In 10th IEEE International Conference on Computer Vision (ICCV 2005), 17-20 October 2005, Beijing, China, pages 1208–1213. IEEE Computer Society, 2005.

[37] Yanwei Pang, Lei Zhang, Zhengkai Liu, Nenghai Yu, and Houqiang Li. Neighborhood Preserving Projections (NPP): A Novel Linear Dimension Reduction Method. In Advances in Intelligent Computing, International Conference on Intelligent Computing, ICIC 2005, Hefei, China, August 23-26, 2005, Proceedings, Part I, volume 3644 of Lecture Notes in Computer Science, pages 117–125. Springer, 2005.

[38] Hong Qiao, Peng Zhang, Di Wang, and Bo Zhang. An Explicit Nonlinear Mapping for Manifold Learning. IEEE Trans. Cybern., 43(1):51–63, 2013.

[39] 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, volume 5479 of Lecture Notes in Computer Science, pages 443–461. Springer, 2009.