IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2025, No. 1, pp. 450–474. DOI:10.46586/tches.v2025.i1.450-474

Perceived Information Revisited II

Information-Theoretical Analysis of Deep-Learning Based Side-Channel Attacks

Akira Ito[1], Rei Ueno[2] and Naofumi Homma[3]

[1] NTT Social Informatics Laboratories, Nippon Telegraph and Telephone Corporation, 3–9–11 Midori-cho, Musashino-shi, Tokyo, 180-8585, Japan
akria.itoh@ntt.com

[2] Kyoto University, Yoshidahommachi, Sakyo-ku, Kyoto, 606-8501, Japan
ueno.rei.2e@kyoto-u.ac.jp,

[3] Tohoku University, 2–1–1 Katahira, Aoba-ku, Sendai-shi, Miyagi, 980-8577, Japan
naofumi.homma.c8@tohoku.ac.jp

Abstract. Previous studies on deep-learning-based side-channel attacks (DL-SCAs) have shown that traditional performance evaluation metrics commonly used in DL, like accuracy and F1 score, are not effective in evaluating DL-SCA performance. Therefore, some previous studies have proposed new alternative metrics for evaluating the performance of DL-SCAs. Notably, perceived information (PI) and effective perceived information (EPI) are major metrics based on information theory. While it has been experimentally confirmed that these metrics can give the attack success rate (SR) for DL-SCAs, their theoretical validity remains unclear.

In this paper, we propose a new theoretically valid performance evaluation metric called latent perceived information (LPI), which serves as an alternative to the existing metrics. LPI is defined as the mutual information between the output of the feature extractor of a neural network (NN) model and the intermediate value, representing the potential attack performance of the trained model. First, we prove that LPI provides an upper bound on the SR of a DL-SCA by modeling and formulating DL-SCA as a communication channel. Additionally, we clarify the conditions under which PI and EPI theoretically provide an upper bound on the SR from the perspective of LPI. For practical computation of LPI, we present two methods. One utilizes the Kraskov (KSG) estimator, a common mutual information estimator, and the other is based on logistic regression. While the KSG estimator is computationally intensive, it yields accurate LPI values. In contrast, the logistic regression is faster but provides a lower bound for LPI. Through experimental attacks on AES software and hardware implementations with masking countermeasures, we demonstrate that the LPI values estimated by these two methods are significantly similar, indicating the reliability and soundness of our proposed estimation techniques. Furthermore, we show that, by using the logistic regression as a classifier, we can significantly improve the attack performance of the trained model when the difference between the SR upper bound by the LPI and its actual SR is large. This indicates that LPI represents the potential for performance improvement in the trained model. Therefore, our study contributes to optimizing the distinguisher for attack performance using the trained model.

Keywords: Profiled side-channel attacks · Perceived information · Success rate · Deep learning · Information theory

Received: 2024-07-15 Accepted: 2024-09-15 Published: 2024-12-09

1 Introduction

1.1 Background

Deep-learning-based side-channel attack. Deep-learning based side-channel attacks (DL-SCAs) have been an active research topic in the field of cryptographic implementation [MHM14, MPP16, CDP17, PHJ+19, HHGG20, UXT+22, PPM+23, TUX+23, SM23] because of their effectiveness. Profiled DL-SCA can achieve high performance in a key-recovery SCA on AES as well as in side-channel assisted chosen-ciphertext attacks on post-quantum key encapsulation mechanisms, even if the implementation is protected using masking and random delay [ZBHV19, WAGP20, ZBHV21, LZC+21, UXT+22, TUX+23]. An attacker using DL-SCA requires fewer assumptions and less prior knowledge about the target implementation than those using other SCAs because the model can implicitly learn these assumptions and knowledge during the profiling phase if the attacker has a profiling device. For example, conventional SCAs, such as correlation power analysis [BCO04] and template attacks [CRR02], require modifications to the attack algorithm and preprocessing of traces based on countermeasures employed in the target implementation. For example, these attacks require preprocessing to align the traces when targeting implementations with random delay-based countermeasures. Conversely, in DL-SCA, a neural network (NN) effectively counteracts these measures by utilizing a sufficient number of traces during the NN training. A thorough investigation of the theory and practice of DL-SCAs is imperative to comprehend potential threats posed by SCAs to cryptographic implementations.

Performance metrics of DL-SCA. In a DL-SCA, selecting an objective function, referred to as a loss function, is essential because it directly affects the training efficacy of the model (i.e., its attack performance). In multi-class classification problems in machine learning, *negative log-likelihood (NLL)*, often referred to as *categorical cross-entropy* in deep learning contexts, is frequently utilized as a loss function because it is considered as a surrogate loss for enhancing accuracy [MMZ23]. Most studies on DL-SCA have also utilized the NLL; however, it also sometimes contradicts the attack performance of the model, as described in Section 2.3. Furthermore, most traditional performance metrics such as F1 score and accuracy in machine learning are likely to be unsuitable for predicting the attack performance in DL-SCA as reported in [PHJ+19]. Thus, the success rate (SR) has been commonly used for a quantitative evaluation of the performance of DL-SCAs [SMY09].

Mutual information (MI), perceived information (PI), and success rate (SR). Masure et al. presented the pioneering theoretical analysis of model training using NLL in DL-SCA [MDP20]. They proved that the NLL asymptotically converges to a cross-entropy (CE) function as the number of traces utilized approaches infinity. Furthermore, they demonstrated that *perceived information (PI)* can be derived from CE. Given a model, the PI is a lower bound on the mutual information (MI) $I(Z; \boldsymbol{X})$ between a side channel trace $\boldsymbol{X}$ and an intermediate value Z. MI can be estimated using PI by identifying model parameters that minimize CE. Besides, de Chérisey et al. [dCGRP19] presented an inequality that upper-bounds the SR by MI, thereby making MI crucial for evaluating the achievable SR. Further, when the PI equals the MI, the distribution modeled by the NN is equal to the true distribution, thereby yielding an optimal distinguisher for the most potent attack [HRG14]. Note, however, that the computation and precise estimation of MI are usually quite difficult; this is a reason why, in practice, we require an alternative metric that is computable and theoretically valid for evaluating SCA.

Conjecture on PI–SR inequality. In [MDP20], Masure et al. revealed the significance of identifying parameters that minimized NLL, which is approximately equivalent to CE in SCAs. However, the NLL loss function cannot attain its minimum value during model

training. In fact, parameters that minimize the NLL may not exist depending on the model architecture, which makes it crucial to understand the relationship between the SR and non-minimized NLLs from both theoretical and practical perspectives. Given this context, Masure et al. [MDP20] conjectured that an inequality similar to the one proposed by de Chérisey et al. [dCGRP19] may exist between the PI and SR; that is, the PI is an upper bound of the SR for a given model. If this inequality holds, the upper bound of the SR for the model can be evaluated from the PI (or equivalently, NLL), thereby allowing the rapid prediction of model performance during training without resorting to computationally intensive SR estimation through key recovery. Therefore, validating this inequality has significant practical implications.

Effective CE/PI (ECE/EPI) and conjecture on EPI–SR inequality. Grosso and Standaert [GS18] empirically observed that the PI–SR inequality does not hold in the context of a soft analytical side-channel attack (SASCA). Subsequently, Ito et al. [IUH22b] constructed a counterexample that disproved the conjecture by Masure et al. [MDP20]. They demonstrated that the SR remained invariant to variations in the inverse temperature β of the softmax function in the model output, whereas CE/NLL changed. Their findings indicated that an increase in β results in an unbounded increase in the CE without changing the SR; therefore, a high NLL, which is approximately equal to the CE, does not inherently signify an unsuccessful attack. Further, they introduced *effective CE (ECE)* as the CE minimized with respect to β and defined *effective PI (EPI)* using ECE. ECE and EPI were defined to solve the uncertainty of SR. Accordingly, Ito et al. conjectured a similar inequality that upper-bounds SR by EPI and demonstrated its experimental validity. However, this conjecture is yet to be theoretically substantiated. Proving this inequality would theoretically validate the estimation of SR from NLL loss. Further, the theoretical foundations would aid the study on DL-SCA in practical aspects to improve its performance and develop countermeasures. Therefore, clarifying the relationship among PI, EPI, and SR and exploring more appropriate evaluation metrics than these metrics are important in this research domain.

1.2 Our contributions

The contribution of this study is threefold.

Information-theoretical analysis of DL-SCA. Although previous studies examined DL-SCA from an information-theoretic perspective, no communication channel for DL-SCA has been developed. In this study, we establish a new communication channel model for DL-SCA for its detailed analysis. Based on our communication channel, we introduce *latent perceived information (LPI)* as a new metric theoretically relating to SR, which plays an essential role in our analysis. LPI is defined by the MI between the feature extractor output of the model and the intermediate value. We derive an information-theoretical inequality that links the model in DL-SCA to the SR. Namely, we formally prove that the LPI is an upper bound of SR, as stated by the *LPI–SR inequality*.

Revealing conditions when EPI–SR inequality holds. As the LPI–SR inequality is formally proven, the EPI–SR inequality is also valid if the LPI and EPI are equivalent. We identify conditions under which the EPI–SR inequality holds using the relationship between the LPI and EPI. Consequently, we demonstrate that the conjecture of the EPI–SR inequality holds when the distribution modeled by an NN is well-calibrated with the inverse temperature and matches the true distribution of the intermediate value given the feature extractor output.

LPI estimation methods and improvement of DL-SCA performance through LPI. We propose two methods for estimating LPI in practice. One utilizes the Kraskov (KSG) estimator, which estimates mutual information, and the other is based on logistic regression. In general, estimating MI, including LPI, is extremely difficult. However, the LPI is defined using the output of the feature extractor, whose dimension is usually low. This renders the LPI estimation feasible in practice. Through experimental attacks on AES software and hardware implementations with masking countermeasures, we demonstrate that the LPI values estimated using both methods are very close, which indicates that the LPI would be accurately estimated. Furthermore, we also demonstrate that the LPI–SR inequality is useful to gauge the potential for improving the attack performance of NN models. Subsequently, we present a concrete method to improve the model attack performance by using a logistic regression-based classifier and evaluate it experimentally.

1.3 Paper organization

Section 2 introduces the mathematical notations used in this paper and the DL-SCA. Section 3 presents the information-theoretical analyses on DL-SCA. Section 4 presents the relationship among PI, EPI, and LPI and shows the condition when the EPI–SR inequality holds. Section 5 presents LPI estimation methods for DL-SCAs. Section 6 demonstrates the experimental DL-SCAs on masked AES implementations to verify the validity of our theoretical analysis. Finally, Section 7 concludes the paper.

2 Preliminaries

2.1 Notations

A calligraphic letter (e.g., $\mathcal{X}$) represents a set, a lowercase variable (e.g., x) represents an element of the corresponding set (i.e., $x \in \mathcal{X}$), and an uppercase variable (e.g., X) represents a random variable over the corresponding set (i.e., X for $\mathcal{X}$), unless otherwise defined. Let $\Pr(A)$, p, and q represent the probability of an event A, true density or mass function, and probability density or mass function represented by an NN, respectively. For example, the true probability mass function of discrete random variables X and Y is given by $p_{X,Y}(x,y) = \Pr(X = x, Y = y)$. We may omit the subscripted random variables if the random variables of the probability distribution are obvious. For example, we may write $p(x,y)$ for $p_{X,Y}(x,y)$. The conditional probability distribution is denoted by $p_{X|Y}(x \mid y) = p(x \mid y) = p(x,y)/p(y)$ if $p(y) \neq 0$; otherwise, $p_{X|Y}(x \mid y) = 0$. Let $\mathbb{E}$ represent the expectation. For example, $\mathbb{E}_X[f(X)]$ represents the expectation of $f(X)$ in terms of X, where $f : \mathcal{X} \to \mathbb{R}$ represents a (measurable) function. In this paper, we omit brackets and simply write $\mathbb{E}_X f(X)$ if there is no confusion. Let $H(X) = -\mathbb{E}_X \log p_X(X)$ represent an entropy function, where log is the binary logarithm. The MI between X and Y is defined as $I(X;Y) = H(X) - H(X \mid Y) = \mathbb{E}_{X,Y} \log p_{X,Y}(X,Y)/(p_X(X)p_Y(Y))$. A sequence of m random variables/vectors independently sampled from a distribution of X is represented by independent copies as $X^m = (X_1, X_2, \ldots, X_m)$.

2.2 Overview of profiled DL-SCA

This study focuses on SCAs on block ciphers, particularly AES because AES is the most widely used symmetric key cipher and is the target of attacks in much of the DL-SCA studies [IUH22b, WAGP20, BIK+24]. The DL-SCA consists of profiling and attack phases. During the profiling phase, an NN is trained to model the conditional distribution corresponding to the device leakage characteristics. Let $\mathcal{S}_p = \{ (\boldsymbol{X}_i, Z_i) \mid 1 \le i \le m_{\mathsf{pro}} \}$ be a training dataset used in the profiling phase, where $\boldsymbol{X}_i$, Z_i, and $m_{\mathsf{pro}} \in \mathbb{N}$ represent the i-th side-channel trace of the i-th observation, corresponding intermediate value (e.g.,

first-round Sbox output in typical SCAs on software AES implementation), and the number of traces used in the profiling phase, respectively. We assume that $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_{m_{\mathsf{pro}}}$ and $Z_1, Z_2, \ldots, Z_{m_{\mathsf{pro}}}$ are independent and identically distributed (i.i.d.) random variables over $\mathcal{X} = \mathbb{R}^{n_\ell}$ and $\mathcal{T} = \{0,1\}^n$, respectively. Here, $n_\ell \in \mathbb{N}$ is the number of sample points in a trace, and $n \in \mathbb{N}$ is the bit length of the intermediate value (in the case of AES, $n = 8$). Let $\theta \in \mathbb{R}^{n_\theta}$ represent the NN parameters, where n_θ denotes the dimension of the parameters. The profiling phase estimates adequate model parameters $\hat{\theta}$ using the training dataset $\mathcal{S}_p$. Optimal parameters are obtained by solving the minimization problem of the CE loss function, which is defined as

$$\mathrm{CE}(q_\theta) = -\mathbb{E}_{Z,\boldsymbol{X}} \log q_\theta(Z \mid \boldsymbol{X}) = -\int \sum_z p_{Z,\boldsymbol{X}}(z, \boldsymbol{x}) \log q_\theta(z \mid \boldsymbol{x})\, d\boldsymbol{x}, \tag{1}$$

where Z and $\boldsymbol{X}$ are the random variables of label z and trace $\boldsymbol{x}$, respectively, and q_θ represents the conditional probability distribution modeled by the NN with parameters θ.

In Equation (1), $\mathrm{CE}(q_\theta)$ takes the minimum value if and only if $p_{Z|\boldsymbol{X}} = q_\theta$ [Bis06, GBC16]. We can model the true distribution p if we can determine the optimal parameters $\hat{\theta}$ that make $\mathrm{CE}(q_{\hat{\theta}})$ sufficiently small; however, we cannot calculate Equation (1) because it contains the integral and summation of the unknown probability distribution p. Therefore, we approximate $\mathrm{CE}(q_\theta)$ using the training data $\mathcal{S}_p$ via the Monte Carlo method as

$$\mathrm{CE}(q_\theta) \approx L(q_\theta) = -\frac{1}{m_{\mathsf{pro}}} \sum_{i=1}^{m_{\mathsf{pro}}} \log q_\theta(Z_i \mid \boldsymbol{X}_i). \tag{2}$$

The approximation of CE in Equation (2) is called the negative log-likelihood (NLL). According to the law of large numbers, the NLL converges almost surely to $\mathrm{CE}(q_\theta)$ as $m \to \infty$ for fixed q_θ.

In the attack phase, we estimate the secret key k^* of the target device by using the trained model. Let $\mathcal{S}_a = \{\, (\boldsymbol{X}_i, T_i) \mid 1 \le i \le m_{\mathsf{atk}} \,\}$ be the dataset used in the attack phase, where $\boldsymbol{X}_i$, T_i, and $m_{\mathsf{atk}} \in \mathbb{N}$ represent the side-channel trace at the i-th observation, and corresponding plaintext or ciphertext, and the number of attack traces, respectively. During the attack phase, we calculate the NLL for each hypothetical key candidate $k \in \mathcal{K}$ using the intermediate value $Z_i^{(k)}$ calculated from T_i as

$$L^{(k)}(q_{\hat{\theta}}) = -\frac{1}{m_{\mathsf{atk}}} \sum_{i=1}^{m_{\mathsf{atk}}} \log q_{\hat{\theta}}(Z_i^{(k)} \mid \boldsymbol{X}_i).$$

Here, $Z_i^{(k)}$ is given as the output of the selection function (e.g., $Z_i^{(k)} = \mathrm{Sbox}(T_i \oplus k)$). The correct key is estimated as the key candidate with the lowest NLL value, which is equivalent to approximating and comparing the results

$$\mathrm{CE}^{(k)}(q_{\hat{\theta}}) = -\mathbb{E} \log q_{\hat{\theta}}(Z^{(k)} \mid \boldsymbol{X}),$$

for each key candidate k. It has been proven that such a DL-SCA is optimal (i.e., it maximizes the SR) if we have $q_{\hat{\theta}} = p_{Z|\boldsymbol{X}}$ [IUH21, IUH22b].

Hereafter, we simply denote the number of traces by m instead of m_{atk} and m_{pro}.

2.3 SCA evaluation metrics

Success rate (SR) is a common and promising metric for a quantitative evaluation of (DL-)SCAs [SMY09, MDP20, WAGP20], as well as guessing entropy (GE). SR is the probability that the rank of the correct key is one, and GE is the expected rank of the

correct key. The SR and GE with m traces in an SCA are formally defined as

$$\mathrm{SR}_m = \Pr(\mathrm{rank}(k^*, m) = 1) = \Pr(\hat{K}_m = k^* \mid K = k^*),$$
$$\mathrm{GE}_m = \mathbb{E}[\mathrm{rank}(k^*, m)],$$

respectively, where k^* is the correct key, $\mathrm{rank}(k, m)$ is the rank of a key k with m-trace SCA, $\hat{K}_m$ is the estimated correct key value from m attack traces, and K is the random variable of secret key. For a DL-SCA, the correct key rank is formally defined by

$$\mathrm{rank}(k^*, n_a) = 1 + \sum_{\hat{k} \in \mathcal{K}_{\setminus k^*}} \mathbb{1}_{\left\{L^{(k^*)}(q_{\hat{\theta}}) \geq L^{(\hat{k})}(q_{\hat{\theta}})\right\}},$$

where $\mathbb{1}_{\mathcal{A}}$ is an indicator function on a set $\mathcal{A}$ and $\mathcal{K}_{\setminus k^*}$ is a set consisting of all key candidates except for the correct key. Note that SR_m, GE_m, and rank are function of trained model parameters $\hat{\theta}$ in the case of DL-SCA.

SR directly represents the performance of SCAs, while GE is important for estimating the complexity of key enumeration algorithms [PSG16]. Importantly, GE is bounded from above and below by SR [IUH21] (i.e., we have $2 - \mathrm{SR}_m \leq \mathrm{GE}_m \leq |\mathcal{K}|(1 - \mathrm{SR}_m) + \mathrm{SR}_m$, where $\mathcal{K}$ is the set of all key candidates). For example, if $\mathrm{SR}_m = 0.9$, then the upper bound on GE in [IUH21] yields $\mathrm{GE}_m \leq 26.5$. Thus, in this paper, we conduct an information-theoretical analysis of SR, which is crucial for evaluating the key recovery performance of DL-SCA in an information-theoretical manner and is related to the key enumeration complexity through GE.

Contradiction between SCA and DL metrics. As described in Section 2.2, NLL is commonly used as a key metric for evaluating NN models. However, there have been reports of contradictions between SCA metrics (e.g., SR and GE) and NLL [IUH21]. For example, an overfitted NN model can sometimes outperform one with a smaller NLL. Overfitting occurs when an NN is trained for too many epochs, leading to a larger validation loss. In the machine learning community, the model with the smallest validation loss is generally considered the best for generalization. However, as shown in [IUH22b], an overfitted NN can sometimes perform better than a model with a small validation loss. Specifically, the attack performance (e.g., SR) can improve with overfitting, even if this increases validation NLL. In fact, our experiments in Section 6 (Figure 3) also demonstrate that SR remains consistent even when NLL significantly increases (i.e., PI decreases).

2.4 Conventional communication channel model and MI–SR inequality

Figure 1 shows a communication channel model developed by de Chérisey et al. [dCGRP19] that represents an SCA, where K, $T^m = (T_1, T_2, \ldots, T_m)$, $Z^m = (Z_1, Z_2, \ldots, Z_m)$, $\boldsymbol{X}^m = (\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_m)$, and $\hat{K}_m$ represent the secret key byte, tuple of m plaintext/ciphertext bytes, secret intermediate values targeted by SCA, side-channel traces, and secret key byte estimated using m traces, respectively.[1] In the figure, "Adversary" refers to any key-recovery algorithm that operates in polynomial time based on the given inputs. Let $\mathrm{SR}_m = \Pr(K = \hat{K}_m)$ denote the SR of the optimal SCA with m traces. Based on this model, de Chérisey et al. proved the following theorem, which states that the optimal SR is bounded from above by the MI between the side-channel trace and intermediate value.

[1] In the previous section, the correct key was defined as a constant value k^*, whereas here it is introduced as a random variable K. This is necessary to derive an upper bound on SR by information theory. The definition of SR given here is consistent with the definition in the previous section if the secret key is chosen uniformly at random and the attack difficulty is independent of the choice of secret key (i.e., $\forall k, k'; \Pr(\hat{K}_m = k \mid K = k) = \Pr(\hat{K}_m = k' \mid K = k')$). In fact, we have $\Pr(K = \hat{K}_m) = \sum_{k^* \in \mathcal{K}_m} \Pr(\hat{K}_m = K \mid K = k^*) \Pr(K = k^*) = \Pr(\hat{K}_m = k^* \mid K = k^*)$.

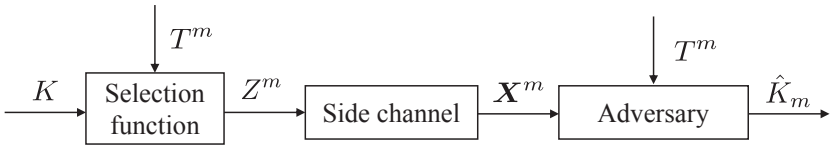


Figure 1: Communication channel of SCA by de Chérisey et al. [dCGRP19].

Theorem 1 (MI–SR inequality [dCGRP19])**.** *In the communication channel shown in Figure 1, the SR of an optimal SCA with m traces, defined as $\mathrm{SR}_m = \Pr(\hat{K}_m = K)$, is bounded from above as*

$$\xi(\mathrm{SR}_m) \leq mI(Z; \boldsymbol{X}), \tag{3}$$

where $\xi : [0,1] \to \mathbb{R}_{\geq 0}$ is defined as

$$\xi(r) = H(K) - H_2(r) - (1-r)\log(2^n - 1). \tag{4}$$

Here, $H_2 : [0,1] \ni r \mapsto -r\log r - (1-r)\log(1-r) \in [0,1]$ is a binary entropy function, where we define $0 \log 0 = 0$.

In [dCGRP19], de Chérisey et al. demonstrated experimentally that this theorem can be used for a precise estimation of the achievable SR for an optimal SCA. However, this inequality is used only to evaluate the optimal SCA, and it cannot be used to evaluate a DL-SCA with the model parameters $\hat{\theta}$. In fact, an adversary in the communication channel is supposed to use an optimal distinguisher. In addition, neither the communication channel nor Equation (3) includes the model parameter $\hat{\theta}$. Thus, it is impossible to discuss the SR of DL-SCA with the model parameters $\hat{\theta}$. This motivated us to develop a new communication channel model for discussing the SR of DL-SCAs.

3 Information-theoretical analyses on DL-SCA

3.1 Overview

We model the DL-SCA as a communication channel. Based on this communication channel, we then derive an inequality in which the SR of the DL-SCA is bounded from above by the MI between the output of the feature extractor of NN and the intermediate value Z. Finally, we discuss the relationship between our inequality and the MI–SR inequality reported by de Chérisey et al. and show that our inequality is more useful for evaluating the performance of DL-SCAs. Our major technical contribution here includes the establishment of the communication channel model of DL-SCA and the discovery of the LPI–SR inequality as a new upper bound of SR rather than the development of new proof techniques.

3.2 Communication channel model of DL-SCA

Figure 2 shows the proposed communication channel model of DL-SCA for a given NN model. This paper assumes that an NN model consists of a *feature extractor* f that extracts information from the given trace and a *classifier* g that performs classification from the information extracted by the feature extractor. Generally, the classifier is the last layer with a softmax activation function, and the feature extractor comprises the layers before the last layer. Our assumption that the activation function of the last layer is the softmax function is not a strong limitation because the softmax function is the most commonly used when performing classification using an NN model [GBC16, Chapter 6.2.2.3]. In fact, the softmax function is also widely used in DL-SCA [WAGP20, ZBD+21]. This study focuses

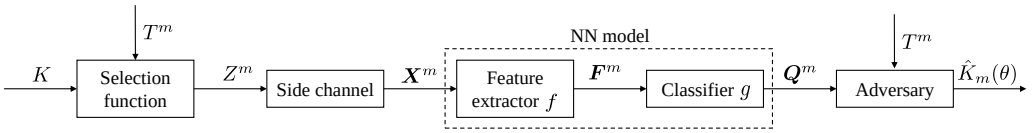


Figure 2: Communication channel of DL-SCA.

on attacks against a block cipher such as AES. We describe the definition of random variables and the meanings of this model.

- $m \in \mathbb{N}$ represents the number of traces used in the attack phase.
- $\theta \in \mathbb{R}^{n_\theta}$ represents the model parameters of the neural network, where n_θ represents the number of dimensions of the parameters.
- K and $\hat{K}_m(\theta)$ represent the correct and estimated partial secret keys, respectively. They belong to the key space $\mathcal{K} = \{0,1\}^n$, where n represents the bit length of the secret key (i.e., $n = 8$ for AES). Here, the estimated key is expressed as $\hat{K}_m(\theta)$ to show that it depends on the NN parameters θ and the number of traces m.
- $T^m = (T_1, T_2, \ldots, T_m)$ represents the tuple of m plaintexts/ciphertexts corresponding to the attack traces, where $T_1, T_2, \ldots, T_m$ denote n-bit plaintext/ciphertexts. In this study, they are assumed to be sampled from the uniform distribution and i.i.d.
- $Z^m = (Z_1, Z_2, \ldots, Z_m)$ represents the tuple of m intermediate values corresponding to the attack traces. Each intermediate value is given by $Z_i = \phi(K, T_i), i \in \{1, 2, \ldots, m\}$ with a selection function $\phi : \mathcal{K} \times \mathcal{T} \to \mathcal{Z}$. For example, we frequently have $\phi(K, T_j) = \mathrm{Sbox}(K \oplus T_j)$ in attacking software AES implementations.
- $\boldsymbol{X}^m = (\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_m)$ represent the tuple of m side-channel traces. Each trace $\boldsymbol{X}_i$ represents a random variable over an n_ℓ-dimensional Euclidean space $\mathbb{R}^{n_\ell}$. These random variables are assumed to be i.i.d.
- $\boldsymbol{F}^m = (\boldsymbol{F}_1, \boldsymbol{F}_2, \ldots, \boldsymbol{F}_m)$ represents a tuple of the feature extractor outputs of the NN for the attack traces. For each i, $\boldsymbol{F}_i$ represents a vector and is given by $\boldsymbol{F}_i = f_\theta(\boldsymbol{X}_i)$, where $f_\theta : \mathcal{X} \to \mathcal{F} = \mathbb{R}^{n_f}$ represents the feature extractor of NN with the model parameters θ, and n_f represents its output dimension.
- $\boldsymbol{Q}^m = (\boldsymbol{Q}_1, \boldsymbol{Q}_2, \ldots, \boldsymbol{Q}_m)$ represents a tuple of the outputs of the NN for the attack traces. For each i, $\boldsymbol{Q}_i$ represents a vector and is given by $\boldsymbol{Q}_i = g_\theta(f_\theta(\boldsymbol{X}_i))$, where $g_\theta : \mathcal{F} \to \mathcal{Q} = \mathbb{R}^{n_q}$ represents the NN classifier with the model parameters θ, and n_q represents the number of output classes. For example, if the NN predicts the probability of intermediate values (i.e., $n_q = 2^n$), the output $\boldsymbol{Q}_i$ represents a 256-dimensional vector consisting of the outputs of the softmax function as $\boldsymbol{Q}_i = (q_\theta(0 \mid \boldsymbol{X}_i), q_\theta(1 \mid \boldsymbol{X}_i), \ldots, q_\theta(2^n - 1 \mid \boldsymbol{X}_i))$ for each i, where $n = 8$ for AES. Further, if the NN predicts the probability of the HW of the intermediate value, it is an $(n+1)$-dimensional vector given by $\boldsymbol{Q}_i = (q_\theta(0 \mid \boldsymbol{X}_i), q_\theta(1 \mid \boldsymbol{X}_i), \ldots, q_\theta(n \mid \boldsymbol{X}_i))$ for each i.

In Figure 2, "Adversary" estimates the secret key by using all outputs $\boldsymbol{Q}^m$ of the NN and the plaintexts/ciphertexts T^m. In [dCGRP19], de Chérisey et al. implied that a Markov chain $(K, T^m) \to (Z^m, T^m) \to (\boldsymbol{X}^m, T^m) \to (\boldsymbol{F}^m, T^m) \to (\boldsymbol{Q}^m, T^m) \to (\hat{K}_m(\theta), T^m)$ holds. This communication channel model does not depend on how the NN is trained or used during an attack. This means that, for example, the model is valid even if any loss

function, such as the CER [ZZN+20] or ranking loss [ZBD+21], is used as the loss function during training.

Based on the communication channel illustrated in Figure 2, we define a new information-theoretical metric named the LPI, which plays an essential role in our analysis and has desirable properties for evaluating DL-SCA[2].

Definition 1 (Latent Perceived Information (LPI))**.** Let θ represent the model parameters of a neural network and $q_\theta(Z \mid \boldsymbol{X})$ represent the conditional probability distribution modeled by the NN. In the communication channel in Figure 2, the LPI of the model is defined as

$$\mathrm{LPI}(q_\theta) \coloneqq I(Z;\boldsymbol{F}) = H(Z) - H(Z \mid \boldsymbol{F}).$$

3.3 Relationship between the SR and model outputs

We prove the following theorem, which states that inequality between $\boldsymbol{F}$ and SR holds on the communication channel in Figure 2, which we call the LPI–SR inequality.

Theorem 2 (LPI–SR inequality)**.** *In the communication channel shown in Figure 2, the SR with m traces using the NN with model parameters θ, defined as $\mathrm{SR}_m(\theta) = \Pr(\hat{K}_m(\theta) = K)$, is bounded from above as*

$$\xi(\mathrm{SR}_m(\theta)) \leq mI(Z;\boldsymbol{F}) = m\mathrm{LPI}(q_\theta), \tag{5}$$

where ξ denotes the function defined in Equation (4).

Proof. The proof is based on Fano's inequality [CT06, Theorem 2.10.1] similarly to the MI–SR inequality [dCGRP19, Lemma 2]. In their proof, they used the data processing inequality [CT06, Theorem 2.8.1] on the Markov chain of $(K, T^m) \to (Z^m, T^m) \to (\boldsymbol{X}^m, T^m)$ represented in Figure 1, given as

$$I(K, T^m; \boldsymbol{X}^m, T^m) \leq I(Z^m, T^m; \boldsymbol{X}^m, T^m).$$

In this proof, we alternatively focus on the Markov chain of $(K, T^m) \to (Z^m, T^m) \to (\boldsymbol{X}^m, T^m) \to (\boldsymbol{F}^m, T^m)$ in Figure 2, implying that the corresponding data processing inequality is

$$I(K, T^m; \boldsymbol{F}^m, T^m) \leq I(Z^m, T^m; \boldsymbol{F}^m, T^m). \tag{6}$$

According to the relationship between MI and entropy, the left-hand side of Equation (6) can be written as

$$\begin{aligned} I(K, T^m; \boldsymbol{F}^m, T^m) &= H(K, T^m) - H(K, T^m \mid \boldsymbol{F}^m, T^m) \\ &= H(K) + mH(T) - H(K \mid \boldsymbol{F}^m, T^m) \\ &= H(K) + mH(T) - H(K \mid \boldsymbol{F}^m, T^m, \hat{K}_m(\theta)) \\ &\geq H(K) + mH(T) - H(K \mid \hat{K}_m(\theta)). \end{aligned}$$

[2] In [PBP21], Perin et al. conducted an information-theoretical analysis of DL-SCA focusing on $I(Z;\hat{Z})$, where $\hat{Z}$ represents the predicted label by the NN. This would be similar to LPI, but their analysis was devoted to discussing the relation between $I(Z;\hat{Z})$ and MIs of hidden layers to evaluate the generalization of the NN for early stopping, rather than establishment of communication channel and evaluation of SR. Thus, their motivation, focus, goal, and results are different from those of this paper. Actually, the Markov chain focused in [PBP21] does not include K, while ours includes it to analyze the SR. One advantage of our proposed method over theirs is its theoretical validity. Their method is based on the information bottleneck principle, which has been observed in NN generalization and empirically confirmed but not proven. In contrast, our proposed LPI provides a proven upper bound on SR, making it more theoretically sound than their method.

The equation $H(K \mid T^m, \boldsymbol{F}^m) = H(K \mid T^m, \boldsymbol{F}^m, \hat{K}_m(\theta))$ holds because $\hat{K}_m(\theta)$ is a deterministic function of T^m and $\boldsymbol{F}^m$. Using Fano's inequality, we obtain

$$H(K \mid \hat{K}_m(\theta)) \leq H_2(\mathrm{SR}_m(\theta)) + (1 - \mathrm{SR}_m(\theta)) \log(2^n - 1).$$

Thus, it holds

$$I(K, T^m; \boldsymbol{F}^m, T^m) \geq H(K) + mH(T) - H_2(\mathrm{SR}_m(\theta)) - (1 - \mathrm{SR}_m(\theta)) \log(2^n - 1). \quad (7)$$

The right-hand side of Equation (6) can be written as

$$\begin{aligned} I(Z^m, T^m; \boldsymbol{F}^m, T^m) &= H(Z^m, T^m) - H(Z^m, T^m \mid \boldsymbol{F}^m, T^m) \\ &= H(T^m) + H(Z^m \mid T^m) - H(Z^m \mid \boldsymbol{F}^m, T^m) \\ &= mH(T) + I(Z^m; \boldsymbol{F}^m \mid T^m). \end{aligned} \quad (8)$$

By combining Equations (6) to (8) with [IUH22a, Lemma 4.2], we conclude that

$$H(K) - H_2(\mathrm{SR}_m(\theta)) - (1 - \mathrm{SR}_m(\theta)) \log(2^n - 1) \leq I(Z^m; \boldsymbol{F}^m \mid T^m) \leq mI(Z; \boldsymbol{F}),$$

as required. □

Theorem 2 states that an inequality similar to the one proved by Cherisey et al. is applicable to DL-SCAs. On the left-hand side of Equation (5), $\xi(\mathrm{SR}_m(\theta))$ represents the entropy required to achieve a given $\mathrm{SR}_m(\theta)$ in the DL-SCA using an NN model parameterized by θ with m traces. For example, if the bit length of the secret key is eight bits, an SR of 100% in the DL-SCA requires eight bits of key entropy. Consequently, we deduce $\xi(\mathrm{SR}_m(\theta)) = \xi(1) = 8$, as anticipated. Meanwhile, $mI(Z; \boldsymbol{F})$ quantifies the information concerning the intermediate value that the NN model can potentially extract from the m traces. Therefore, this inequality indicates that, to achieve a certain SR, information about intermediate values potentially extracted by the NN (i.e., $mI(Z; \boldsymbol{F})$) must surpass the information required to achieve it (i.e., $\xi(\mathrm{SR}_m(\theta))$).

In Equation (5), the MI $I(Z; \boldsymbol{F})$ measures the extent of information regarding the intermediate value that can potentially be retrieved from the feature extractor output. This concept is intrinsically linked to PI and EPI, as described in Section 4.

Remark 1 (Upper bound vs. lower bound). Upper bounds of SR are crucial for leakage assessment and security validation against DL-SCA. Conversely, lower bounds of SR are important from the attacker's perspective and for improving DL-SCA. This paper primarily focuses on upper bounds, as it centers on the information-theoretical analysis of DL-SCA using MI. Since MI represents the maximum amount of information in communication, it is suitable for deriving upper bounds. However, it is less useful for discussing lower bounds, which are better derived from probability-theoretical analysis. For a lower bound of SR in DL-SCA developed for ranking loss, refer to [ZBD+21], and for another based on a probability concentration inequality, see [IUH21].

3.4 Difference from MI–SR inequality

We discuss the differences between the MI–SR inequality and our LPI–SR inequality to demonstrate that Equation (5) is more effective in evaluating DL-SCA performance.

In [dCGRP19], de Chérisey et al. established the inequality $\xi(\mathrm{SR}_m) \leq mI(Z; \boldsymbol{X})$ for the SR of side-channel attacks, not limited to DL-SCA. Here, SR_m represents the probability of successful attacks by any SCA using m traces. The function $\xi : [0, 1] \to \mathbb{R}_{\geq 0}$ converts the SR to entropy. Thus, $\xi(\mathrm{SR}_m)$ on the left-hand side represents the entropy of the key required for a certain SR by *any* SCA. Conversely, $mI(Z; \boldsymbol{X})$ quantifies the amount of information regarding intermediate values retrieved from m side-channel traces,

where any extraction method is acceptable. This implies that to achieve a given SR, the information about the intermediate values extractable from m traces $mI(Z;\boldsymbol{X})$ must exceed the necessary information of the secret key $\xi(\mathrm{SR}_m)$.

Further, this inequality must hold for DL-SCA. Let $\mathrm{SR}_m(\theta)$ denote the SR with model parameter θ, and $\xi(\mathrm{SR}_m(\theta)) \leq mI(Z;\boldsymbol{X})$ must hold. However, this inequality may be too loose to evaluate the attack performance (i.e., SR) of a specific model because the right-hand side does not depend on the model parameters. For example, if θ represents random values in the initial learning phase, the SR is assumed to be $\mathrm{SR}_m(\theta) \approx 2^{-n}$, which is as high as a random guess, regardless of the number of traces m. However, the right-hand side increases monotonically with m when $I(Z;\boldsymbol{X}) > 0$ because the MI $I(Z;\boldsymbol{X})$ is a constant independent of the model parameters. Therefore, the MI–SR inequality suggests that the attack can succeed with high probability (e.g., $\mathrm{SR}_m(\theta) = 1$) given a sufficient number of traces, even for ineffective model parameters. Hence, the MI–SR inequality is unsuitable for evaluating DL-SCA models.

In contrast, $\mathrm{LPI}(q_\theta) = I(Z;\boldsymbol{F})$ on the right-hand side of Equation (5) depends on the model parameters. For example, if the model parameters are unsuitable for the attack, the feature extractor output is likely to be a random number with little relevance to the intermediate values. In such cases, the MI between Z and $\boldsymbol{F}$ is approximately zero. Thus, $\xi(\mathrm{SR}_m(\theta)) \leq mI(Z;\boldsymbol{F}) \approx 0$, followed by $\mathrm{SR}_m(\theta) \approx 2^{-n}$.[3] This suggests that the LPI–SR inequality is more suitable for evaluating DL-SCA than the MI–SR inequality.

In addition, the difficulty in MI estimation may differ significantly between $I(Z;\boldsymbol{X})$ and $\mathrm{LPI}(q_\theta) = I(Z;\boldsymbol{F})$. The feature extractor output $\boldsymbol{F}$ has a lower dimensionality than $\boldsymbol{X}$, and therefore, the LPI estimation is dimensionally reduced to extract information on intermediate values compared with the MI estimation. Therefore, estimating $\mathrm{LPI}(q_\theta)$ is easier than estimating $I(Z;\boldsymbol{X})$. The methodology for estimating LPI is detailed in Section 5.

Remark 2 (Strength of LPI–SR inequality). As mentioned earlier, for a given trained model, the LPI–SR inequality provides a stronger evaluation than the MI–SR inequality. However, this does not mean that the LPI–SR inequality is always tight. While LPI represents the amount of information about the intermediate values extracted by the feature extractor, it does not indicate how effectively the classifier utilizes the feature extractor's output. Therefore, even if the LPI is large, the classifier may still perform poorly. In such cases, the LPI–SR inequality will be loose. In Section 6, we experimentally demonstrate a scenario where the LPI–SR inequality is loose in AES hardware implementations (i.e., there is a discrepancy between the performance expected from the LPI–SR inequality and the actual performance), and we show that this gap can be bridged by using an appropriate classifier.

4 Relationship with other perceived information metrics

4.1 Overview

In this section, we first review the PI and its related information quantity, EPI. Next, we explain the relationship among PI, EPI, and LPI and demonstrate that SR can be estimated when PI and EPI are equal to LPI. Further, we examine when PI and EPI are equal to LPI in terms of the conditional probability distributions.

4.2 Review of PI

Definition of PI. In EUROCRYPT 2011 [RSVC+11], Renauld et al. defined PI as follows:

[3] Note that ξ takes a global minimum value of 0 at 2^{-n} [IUH22a, Lemma 5.1]. This is deduced from the fact that the entropy required for a random guess of an n-bit key is zero.

Definition 2 (Perceived Information (PI))**.** Let θ represent the model parameters of an NN and $q_\theta(Z \mid \boldsymbol{X})$ represent the conditional probability distribution modeled by this model. The PI of the model is defined as

$$\mathrm{PI}(q_\theta) \coloneqq H(Z) + \mathbb{E} \log q_\theta(Z \mid \boldsymbol{X}) = H(Z) - \mathrm{CE}(q_\theta).$$

PI has two important properties.

(i) PI can be approximated by NLL. We obtain $\mathrm{PI}(q_\theta) = H(Z) - \mathrm{CE}(q_\theta)$ using the CE $\mathrm{CE}(q_\theta) = -\mathbb{E} \log q_\theta(Z \mid \boldsymbol{X})$. Further, the CE can be approximated using NLL $L(q_\theta)$ when the number of traces m is sufficiently large, implying that $\mathrm{PI}(q_\theta) \approx H(Z) - L(q_\theta)$. In other words, a decrease in the value of the NLL loss function during training is equivalent to an increase in PI.

(ii) PI is a lower bound of MI. Using the conditional entropy $H(Z \mid \boldsymbol{X}) = -\mathbb{E} \log p(Z \mid \boldsymbol{X})$, the MI between the trace and intermediate value is given by $I(Z; \boldsymbol{X}) = H(Z) - H(Z \mid \boldsymbol{X})$, where $p(Z \mid \boldsymbol{X})$ represents the true conditional distribution of the intermediate value Z given trace $\boldsymbol{X}$. According to the non-negativity of the Kullback–Leibler (KL) divergence $D_{\mathrm{KL}}(p_{Z|\boldsymbol{X}} \parallel q_\theta) = \mathbb{E} \log \frac{p(Z|\boldsymbol{X})}{q_\theta(Z|\boldsymbol{X})} \geq 0$ [CT06, Theorem 2.6.3], we have

$$H(Z \mid \boldsymbol{X}) = -\mathbb{E} \log p(Z \mid \boldsymbol{X}) \leq -\mathbb{E} \log q_\theta(Z \mid \boldsymbol{X}) = \mathrm{CE}(q_\theta).$$

Therefore, MI is bounded from below as $I(Z; \boldsymbol{X}) \geq H(Z) - \mathrm{CE}(q_\theta) = \mathrm{PI}(q_\theta)$.

Conjecture on the PI–SR inequality. These properties suggest an intuitive interpretation: the PI represents the amount of information of the intermediate value that the NN can extract from the trace [RSVC+11]. To explain this, we consider the following properties. A decrease in the value of the NLL loss function increases the amount of information that an NN can extract (i.e., the PI). Then, the PI becomes equal to the MI when the NN can retrieve the information most successfully (i.e., when the PI is at its maximum). Meanwhile, when the model training completely fails (i.e., when NLL loss is large), the PI is approximately equal to or less than zero, and the inequality $I(Z; \boldsymbol{X}) \geq \mathrm{PI}(q_\theta)$ becomes meaningless. This can be considered as the model that does not extract any information about intermediate values from the traces. Based on these observations, we hypothesize that the PI represents the amount of information that the model can extract. Thus, Masure et al. [MDP20] conjectured that $\mathrm{SR}_m(\theta)$ is bound above by $\mathrm{PI}(q_\theta)$ as

$$\xi(\mathrm{SR}_m(\theta)) \leq m\mathrm{PI}(q_\theta),$$

which we call the PI–SR inequality in this study. Yet, this is not proven formally. Indeed, this is not always true, as described in Section 4.3.

4.3 Review of EPI

Transformation by inverse temperature. Ito et al. [IUH22b] showed that the conjecture on the PI–SR inequality is not always true by constructing a counterexample. They used the fact that a transformation of the softmax function using the inverse temperature $\beta > 0$ does not change the SR, whereas the PI varies depending on β. For the conditional probability distribution $q_\theta(Z \mid \boldsymbol{X})$ modeled by the NN, the transformed conditional probability distribution $q_\theta^{(\beta)}(Z \mid \boldsymbol{X})$ with an inverse temperature $\beta > 0$ is given by

$$q_\theta^{(\beta)}(z \mid \boldsymbol{x}) = \frac{(q_\theta(z \mid \boldsymbol{x}))^\beta}{\sum_{z'} (q_\theta(z' \mid \boldsymbol{x}))^\beta}. \tag{9}$$

Ito et al. showed that $\mathrm{CE}(q_\theta^{(\beta)}) \to \infty$ and $\mathrm{PI}(q_\theta^{(\beta)}) \to -\infty$ hold as $\beta \to \infty$ [IUH22b, Prpoposition 3], which indicates that PI can take a negative value and be arbitrarily small by changing β. Hence, we can create a counterexample for the PI–SR inequality conjecture using a sufficiently large β because the SR is invariant to this transformation for $\beta > 0$. Further, this suggests that the idea of PI expressing the amount of information that a model can retrieve is incorrect.

Definition of ECE and EPI. Ito et al. defined ECE and EPI to rectify the problem of the inverse temperature of PI. Their basic idea was calibrating the CE and PI to β to solve their uncertainty for an identical SR.

Definition 3 (Effective CE (ECE) and Effective PI (EPI) [IUH22b])**.** Let $q_\theta^{(\beta)}$ represent the probability distribution transformed with the inverse temperature $\beta \geq 0$ defined in Equation (9). Using the same notation as Definition 2, ECE and EPI of q_θ are defined as

$$\mathrm{ECE}(q_\theta) \coloneqq \min_{\beta \geq 0} \mathrm{CE}(q_\theta^{(\beta)}) = \min_{\beta \geq 0} -\mathbb{E} \log q_\theta^{(\beta)}(Z \mid \boldsymbol{X}),$$
$$\mathrm{EPI}(q_\theta) \coloneqq \max_{\beta \geq 0} \mathrm{PI}(q_\theta^{(\beta)}) = \max_{\beta \geq 0} \left(H(Z) + \mathbb{E} \log q_\theta^{(\beta)}(Z \mid \boldsymbol{X}) \right) = H(Z) - \mathrm{ECE}(q_\theta),$$

respectively.[4]

Conjecture on EPI–SR inequality. Similar to Masure et al., Ito et al. also conjectured that the SR is bounded above by the EPI as

$$\xi(\mathrm{SR}_m(\theta)) \leq m\mathrm{EPI}(q_\theta),$$

which we refer to as EPI–SR inequality in this paper. Ito et al. [IUH22b] experimentally confirmed its validity and precision; that is, EPI enables precise SR evaluation through the inequality reported by de Chérisey et al. However, this is yet to be proven formally.

4.4 Relationship among PI, EPI, LPI, and MI

The LPI–SR inequality was formally proven; therefore, if LPI and EPI/PI are equivalent, the EPI/PI–SR inequality is valid. We discuss the relationship among PI, EPI, LPI, and MI and describe them when the above two conjectures hold.

First, we have the following inequality among the PI, EPI, LPI, and MI. LPI is a lower bound of MI, tighter than PI and EPI.

Theorem 3 (Order of PI, EPI, LPI, and MI)**.**

$$\mathrm{PI}(q_\theta) \leq \mathrm{EPI}(q_\theta) \leq \mathrm{LPI}(q_\theta) \leq I(Z; \boldsymbol{X}). \tag{10}$$

Proof. From the definition, $\mathrm{PI}(q_\theta) \leq \mathrm{EPI}(q_\theta)$ holds. We have $\mathrm{LPI}(q_\theta) = I(Z; \boldsymbol{F}) \leq I(Z; \boldsymbol{X})$ according to the data processing inequality on the Markov chain of $Z^m \to \boldsymbol{X}^m \to \boldsymbol{F}^m$ in Figure 2. We then demonstrate that $\mathrm{EPI}(q_\theta) \leq I(Z; \boldsymbol{F})$ holds. As $\mathrm{EPI}(q_\theta) = H(Z) - \min_{\beta \geq 0} \mathrm{CE}(q_\theta^{(\beta)})$ and $\mathrm{LPI}(q_\theta) = I(Z; \boldsymbol{F}) = H(Z) - H(Z \mid \boldsymbol{F})$ hold, it suffices to show that

$$\min_{\beta \geq 0} \mathrm{CE}(q_\theta^{(\beta)}) \geq H(Z \mid \boldsymbol{F}).$$

Fix any β. Here, it holds that

$$\mathrm{CE}(q_\theta^{(\beta)}) = -\mathbb{E} \log q_\theta^{(\beta)}(Z \mid \boldsymbol{X}) \overset{(a)}{=} -\mathbb{E} \log q_\theta^{(\beta)}(Z \mid \boldsymbol{X}, \boldsymbol{F}) \overset{(b)}{=} -\mathbb{E} \log q_\theta^{(\beta)}(Z \mid \boldsymbol{F}), \tag{11}$$

[4] In the original paper, these are defined using inf and sup; however, in this paper, they are defined using min and max for simplicity, as $\mathrm{CE}(q_\theta^{(\beta)})$ always has a global minimum in terms of $\beta \geq 0$.

because (a) if $\boldsymbol{X}$ is given, conditioning $(Z \mid \boldsymbol{X})$ by $\boldsymbol{F}$ does not change the resulting distribution as $\boldsymbol{F}$ is deterministic function of $\boldsymbol{X}$, and (b) the probability of Z given by the model is determined solely from $\boldsymbol{F}$ without $\boldsymbol{X}$. Let $D_{\mathrm{KL}}(p \parallel q)$ denote the Kullback–Leibler (KL) divergence between p and q. Because KL divergence is always non-negative [CT06, Theorem 2.6.3], we have

$$\begin{aligned} D_{\mathrm{KL}}\left(p(Z \mid \boldsymbol{F}) \,\middle\|\, q_\theta^{(\beta)}(Z \mid \boldsymbol{F})\right) = \mathbb{E}\log\frac{p(Z \mid \boldsymbol{F})}{q_\theta^{(\beta)}(Z \mid \boldsymbol{F})} \geq 0 \\ \Leftrightarrow \quad -\mathbb{E}\log q_\theta^{(\beta)}(Z \mid \boldsymbol{F}) \geq -\mathbb{E}\log p(Z \mid \boldsymbol{F}) = H(Z \mid \boldsymbol{F}). \end{aligned} \tag{12}$$

Combining Equations (11) and (12), we conclude that $\mathrm{CE}(q_\theta^{(\beta)}) \geq H(Z \mid \boldsymbol{F})$ holds for any β. This completes the proof. □

From Theorem 3, we have the following corollary.

Corollary 1.

$$\mathrm{CE}(q_\theta) \geq \mathrm{ECE}(q_\theta) \geq H(Z \mid \boldsymbol{F}) \geq H(Z \mid \boldsymbol{X}).$$

From Theorem 3, the conjectures of the PI–SR and EPI–SR inequalities hold if the equality in Equation (10) holds. We first prove Theorem 4, which states that the condition for the EPI–SR inequality holds.

Theorem 4 (Equality condition between EPI and LPI)**.** $\mathrm{EPI}(q_\theta) = \mathrm{LPI}(q_\theta)$ *holds if and only if*

$$D_{\mathrm{KL}}\left(p(Z \mid \boldsymbol{X}) \,\middle\|\, q_\theta^{(\beta')}(Z \mid \boldsymbol{X})\right) = D_{\mathrm{KL}}\Big(p(Z \mid \boldsymbol{X}) \,\Big\|\, p(Z \mid f_\theta(\boldsymbol{X}))\Big),$$

where D_{KL} *represents the KL divergence and* $\beta' = \arg\min_{\beta \geq 0} \mathrm{CE}(q_\theta^{(\beta)})$*; that is,* $q_\theta^{(\beta')}$ *represents the probability distribution used to calculate* $\mathrm{EPI}(q_\theta)$*.*

Proof. Recall that $\mathrm{LPI}(q_\theta) = H(Z) - H(Z \mid \boldsymbol{F})$ and $\mathrm{EPI}(q_\theta) = H(Z) - \min_{\beta>0} \mathrm{CE}(q_\theta^{(\beta)})$. The condition $\mathrm{LPI}(q_\theta) = \mathrm{EPI}(q_\theta)$ is equivalent to

$$H(Z \mid \boldsymbol{F}) = \min_{\beta \geq 0} \mathrm{CE}(q_\theta^{(\beta)}) = \mathrm{CE}(q_\theta^{(\beta')}). \tag{13}$$

$H(Z \mid \boldsymbol{F}) = -\mathbb{E}\log p(Z \mid \boldsymbol{F}) = -\mathbb{E}\log p(Z \mid f_\theta(\boldsymbol{X}))$ holds because of the definition of conditional entropy. Therefore, Equation (13) is equivalent to

$$\begin{aligned} & \mathbb{E}\log p(Z \mid f_\theta(\boldsymbol{X})) = \mathbb{E}\log q_\theta^{(\beta')}(Z \mid \boldsymbol{X}) \\ \Leftrightarrow \quad & \mathbb{E}\log\frac{p(Z \mid \boldsymbol{X})}{p(Z \mid f_\theta(\boldsymbol{X}))} = \mathbb{E}\log\frac{p(Z \mid \boldsymbol{X})}{q_\theta^{(\beta')}(Z \mid \boldsymbol{X})} \\ \Leftrightarrow \quad & D_{\mathrm{KL}}\Big(p(Z \mid \boldsymbol{X}) \,\Big\|\, p(Z \mid f_\theta(\boldsymbol{X}))\Big) = D_{\mathrm{KL}}\left(p(Z \mid \boldsymbol{X}) \,\middle\|\, q_\theta^{(\beta')}(Z \mid \boldsymbol{X})\right), \end{aligned} \tag{14}$$

as required for the sufficiency proof. Equation (14) can be inversely transformed to Equation (13) as required for the necessity proof. This completes the proof. □

Theorem 4 states that EPI is equal to LPI if the distance between the true distribution $p(Z \mid \boldsymbol{X})$ and the distribution $p(Z \mid f_\theta(\boldsymbol{X}))$ is the same as the distance between the true distribution and the modeled probability $q_\theta^{(\beta')}(Z \mid \boldsymbol{X})$. An important case is that $p(Z \mid f_\theta(\boldsymbol{X})) = q_\theta^{(\beta')}(Z \mid \boldsymbol{X})$ holds. Its intuitive explanation is provided below.

Since f_θ is a feature extractor, $f_\theta(\boldsymbol{X})$ represents the extracted information about the intermediate values. The true distribution $p(Z \mid f_\theta(\boldsymbol{X}))$ makes an optimal prediction about

Algorithm 1 LPI estimation using KSG estimator

Input: Trained model parameters θ, validation (or test) set $\mathcal{S}_a = \{(\boldsymbol{X}_i, Z_i) \mid 1 \leq i \leq m\}$, batch size m', and number of iterations n_b
Output: estimated value $\widehat{\mathrm{LPI}}$

$(\boldsymbol{F}_i)_{i=1}^{m} \leftarrow \texttt{GetFeatureMap}(\theta, (\boldsymbol{X}_i)_{i=1}^{m})$
for $j = 1, 2, \ldots, n_b$ **do**
  $(\boldsymbol{F}_i, Z_i)_{i=1}^{m'} \leftarrow \texttt{Sample}((\boldsymbol{F}_i, Z_i)_{i=1}^{m})$ ▷ Sampling with replacement.
  $\widehat{\mathrm{LPI}}_j \leftarrow \mathrm{KSG}((\boldsymbol{F}_i)_{i=1}^{m'}, (Z_i)_{i=1}^{m'})$
end for
$\widehat{\mathrm{LPI}} \leftarrow \frac{1}{n_b} \sum_{j=1}^{n_b} \widehat{\mathrm{LPI}}_j$

Z based on this intermediate value information. Furthermore, when the feature extractor parameters are fixed, this true distribution $p(Z \mid f_\theta(\boldsymbol{X}))$ minimizes NLL. Therefore, the distribution $p(Z \mid f_\theta(\boldsymbol{X}))$ gives the optimal distinguisher for the attack in terms of both attack performance and NLL. However, since the true distribution $p(Z \mid f_\theta(\boldsymbol{X}))$ is usually unknown, we use the alternative distribution $q_\theta(Z \mid \boldsymbol{X})$, which is obtained from the model classifier g_θ. If this distribution $q_\theta(Z \mid \boldsymbol{X})$ matches $p(Z \mid f_\theta(\boldsymbol{X}))$, which means that the classifier g_θ is the best in terms of SR and NLL, LPI and PI are equivalent. When estimating EPI, we apply a transformation using inverse temperature β', which is invariant with respect to SR, to the classifier g_θ and minimize NLL. Therefore, from our LPI–SR inequality, we can say that the EPI–SR inequality holds if the classifier is optimal in terms of both SR and NLL (i.e., $p(Z \mid f_\theta(\boldsymbol{X})) = q_\theta^{(\beta')}(Z \mid \boldsymbol{X})$), except for the degrees of freedom for the inverse temperature.

Finally, the equality condition between the PI and LPI is that $q_\theta(Z \mid \boldsymbol{X}) = q_\theta^{(\beta')}(Z \mid \boldsymbol{X}) = p(Z \mid \boldsymbol{F})$. This represents a situation where the trained model $q_\theta(Z \mid \boldsymbol{X})$ is equal to the distribution $q_\theta^{(\beta')}(Z \mid \boldsymbol{X})$ obtained by minimizing the CE on the inverse temperature, and it is also equal to the distribution $p(Z \mid \boldsymbol{F})$ simultaneously. In [IUH22b], it was shown experimentally that $q_\theta(Z \mid \boldsymbol{X}) = q_\theta^{(\beta')}(Z \mid \boldsymbol{X})$ rarely holds, which implies that the PI–SR inequality may not be valid. We will also demonstrate this experimentally in Section 6.

5 Estimation of LPI

In this section, we introduce two methods for the estimation of LPI. One method is to directly use the Kraskov (KSG) estimator, which is an asymptotic unbiased estimator of MI. The other method is to estimate a lower bound for LPI by using logistic regression to approximate the best model classifier. The method using the KSG estimator can asymptotically estimate the true LPI but is computationally expensive. Logistic regression, on the other hand, is not guaranteed to estimate the true LPI but has the advantage of being computationally less expensive. We confirm experimentally in Section 6 that the values estimated by both methods are very close.

5.1 Direct estimation

Since LPI is the MI between the output of the feature extractor $\boldsymbol{F}$ and the intermediate value Z, it can be estimated using an MI estimator. To avoid bias in the estimation, the MI must be estimated using a validation or test dataset, which is not used for updating the model parameters.

In this paper, we employ the KSG estimator, a common MI estimator that utilizes the k-nearest neighbor (k-NN) method. Typically, the output of the feature extractor in models used for SCAs is limited to a few dozen dimensions (e.g., the feature extractor of

the model proposed by Wouters et al. [WVdHG+20] has around 10 dimensions), which is relatively small compared to the dimension of side-channel trace. Therefore, even with the KSG estimator based on the k-NN method, the impact of the curse of dimensionality on estimation accuracy is expected to be far smaller than the case of estimating $I(Z;\boldsymbol{X})$.

However, if the MI between the intermediate values and the traces is very small due to masking or other countermeasures, the LPI will also become small, which requires a large number of traces to achieve sufficient estimation accuracy. Thus, we need to reduce the computational complexity of the KSG estimator because its worst-case computational complexity is $O(m^2)$ when using m traces. To address this, we use the bootstrap method to reduce computational complexity. Algorithm 1 shows our algorithm to estimate LPI using the KSG estimator. We first determine an appropriate batch size m' smaller than m and the number of iterations n_b. In line 1, we obtain the feature maps $(\boldsymbol{F}_i)_{i=1}^m$ from the trained model $\boldsymbol{\theta}$ by inputting the traces $(\boldsymbol{X}_i)_{i=1}^m$ into the model. In line 3, we resample m' data from the m traces. This resampling simulates sampling from the empirical distribution generated by the m data points. In line 4, we estimate the mutual information with the KSG estimator using the resampled m' data. The KSG estimator is available in open-source libraries such as NPEET[5], so it can be easily estimated using these. This process is repeated n_b times, and the average of these estimates is taken as the final MI estimate.

Since the KSG estimator is asymptotically unbiased, increasing m' and n_b can minimize the bias in the MI estimate. However, the computational complexity of the estimation grows proportionally with n_b and m'. Therefore, n_b and m' must be chosen based on a trade-off between estimation accuracy and computational complexity.

5.2 Logistic regression based estimation

The second method for estimating MI involves using logistic regression. The basic idea is that, for a given NN model, if we can identify the optimal classifier for the last layer, we can calculate a good lower bound for LPI.

Let $\boldsymbol{W}$ and $\boldsymbol{b}$ be the weight and bias of the last layer, respectively. The classifier (i.e., the last layer) is represented by

$$g_\theta(\boldsymbol{f}) = \mathrm{softmax}(\boldsymbol{W}\boldsymbol{f} + \boldsymbol{b}), \tag{15}$$

where $\boldsymbol{f}$ is the input feature map. The model defined in Equation (15) is known as multi-class logistic regression. When the parameters of the feature extractor are fixed, the optimal parameters of $\boldsymbol{W}$ and $\boldsymbol{b}$ can be obtained by performing logistic regression fitting. When NLL is used as the error function, optimizing the logistic regression parameters becomes a convex optimization problem, implying that its global optimum can be analytically found by, for example, the Newton–Raphson method. Therefore, the NLL (and CE) of the trained model can be reduced by optimizing the last layer as a logistic regression.

The estimation procedure for LPI using logistic regression is shown in Algorithm 2. This algorithm takes the trained parameters θ, the validation (or test) dataset $\mathcal{S}_a$, and the number of cross-validation folds c. In line 1, the outputs of the feature extractor $(\boldsymbol{F}_i)_{i=1}^m$ are obtained from the trained parameters θ and the validation traces $(\boldsymbol{X}_i)_{i=1}^m$. Next, in lines 2–7, we perform c-fold cross-validation of logistic regression using these feature maps $(\boldsymbol{F}_i)_{i=1}^m$ and intermediate values $(Z_i)_{i=1}^m$ to obtain the average NLL value of the logistic regressions across all evaluations. In line 2, we split the dataset $(\boldsymbol{F}_i, Z_i)_{i=1}^m$ into c equal folds $(\mathcal{F}_k)_{k=1}^c$. In lines 3–6, we perform the logistic regression fitting and NLL calculation. In line 4, we fit the logistic regression using the k-th fold (set) to obtain the parameter $\boldsymbol{\omega}$. In line 5, we use the parameter $\boldsymbol{\omega}$ to perform inference on the remaining data set $(\boldsymbol{F}_i, Z_i)_{i=1}^m - \mathcal{F}_k$ that was not used for training[6], and calculate the NLL. This is done for

[5] https://github.com/gregversteeg/NPEET

[6] Here, subtraction represents removing the data contained in $\mathcal{F}_k$ from $(\boldsymbol{F}_i, Z_i)_{i=1}^m$.

Algorithm 2 LPI estimation using logistic regression

Input: Trained model parameters θ, validation (or test) set $\mathcal{S}_a = \{(\boldsymbol{X}_i, Z_i) \mid 1 \le i \le m\}$, number of folds c
Output: estimated value $\widehat{\mathrm{LPI}}$

1: $(\boldsymbol{F}_i)_{i=1}^m \leftarrow \texttt{GetFeatureMap}(\theta, (\boldsymbol{X}_i)_{i=1}^m)$
2: $(\mathcal{F}_k)_{k=1}^c \leftarrow \texttt{Split}((\boldsymbol{F}_i, Z_i)_{i=1}^m)$ ▷ Splitting $\mathcal{S}_a$ into c equal folds.
3: **for** $k = 1, 2, \ldots, c$ **do**
4: $\quad \boldsymbol{\omega} \leftarrow \texttt{FitLogisticRegression}(\mathcal{F}_k)$
5: $\quad \widehat{\mathrm{NLL}}_k \leftarrow \texttt{Inference}(\boldsymbol{\omega}, (\boldsymbol{F}_i, Z_i)_{i=1}^m - \mathcal{F}_k)$
6: **end for**
7: $\widehat{\mathrm{LPI}} \leftarrow \frac{1}{c}\sum_{k=1}^c \widehat{\mathrm{NLL}}_k$

all folds. In line 7, the average NLL is then returned as the estimated LPI. It is known that increasing the number of folds in cross-validation reduces the generalization error. Therefore, by using a sufficiently large c and number of traces m, we can approximate the cross-entropy of logistic regression accurately. The computational complexity of this algorithm is $O(c \cdot m \cdot s)$ when logistic regression is trained using the L-BFGS method, where s represents the maximum number of iterations for the L-BFGS method. For example, in the Python Scikit-learn library [PVG+11], $s = 100$ by default. Since cs is expected to be smaller than the number of traces m, this algorithm estimates the LPI value more quickly than using the KSG estimator proposed in Section 5.1.

The validity of this algorithm can be explained as follows. Let θ be the parameters of the trained NN model, and let θ' be the result of updating the parameters of the last layer by treating it as a logistic regression. For simplicity, we will assume that an infinite amount of training data is available, thus preventing the models from overfitting the training set (note that when estimating LPI using logistic regression during actual evaluation, a validation set is used; overfitting does not need to be considered).

Based on these assumptions, the NLL or CE of θ' must be smaller than that of θ (i.e., $-\mathbb{E}\log q_{\theta'}(Z;\boldsymbol{X}) \le -\mathbb{E}\log q_\theta(Z;\boldsymbol{X}) = \mathrm{CE}(q_\theta)$). Additionally, since the parameter space of logistic regression includes degrees of freedom due to inverse temperature, the CE of θ' must also be smaller than the effective cross-entropy (ECE); that is, $\mathrm{CE}(q_{\theta'}) \le \mathrm{ECE}(q_\theta)$.

Furthermore, $H(Z) - \mathrm{CE}(q_{\theta'})$ is also smaller than LPI, which is followed by $\mathrm{LPI}(q_\theta) \ge H(Z) - \mathrm{CE}(q_{\theta'}) \ge H(Z) - \mathrm{ECE}(q_\theta)$. In other words, $\mathrm{CE}(q_{\theta'})$ provides a more accurate evaluation of LPI than ECE. Specifically, if the true distribution $p(Z \mid \boldsymbol{F})$ is within the hypothesis set of logistic regression, then with an infinite number of traces, the lower bound of the LPI estimated using this method matches the true value of LPI.

Given that logistic regression is a linear model with a relatively small model capacity, assuming that the true distribution is included in the hypothesis set might seem optimistic. However, as shown in Section 6, the lower bound of the LPI estimated by logistic regression is close to the LPI estimated by the KSG estimator in Section 5.1, suggesting that the true distribution $p(Z \mid \boldsymbol{F})$ can be approximately included in the hypothesis set.

6 Experimental validation

6.1 Dataset and experimental setup

This section validates our theoretical analyses through experimental DL-SCAs on masked AES software and hardware implementations. We evaluate the empirical values of PI, EPI, and LPI and examine whether the relationship among them mentioned in Section 4 and each SR inequality hold experimentally. In this experiment, we do not use the validation set because this experiment aims to confirm the validity of our analysis, and we do not

perform the hyperparameter-tuning of the model. We directly use the test (attack) set to calculate PI, EPI, and LPI. If these metrics are to be used in the actual evaluation of the trained model, it is necessary to prepare a separate validation set for evaluation.

Software target. We used a first-order Boolean masked AES software provided implementation by ASCAD [BPS+20]. We implemented it on an Atmel Xmega128D4-AU eight-bit microcontroller and acquired the side-channel traces because we required more traces than the ASCAD dataset for improving the accuracy of estimating PI, LPI, and EPI. We acquired the side-channel traces for two different keys, which correspond to profiling (i.e., NN training) and attack phases. The target microcontroller is mounted on a ChipWhisperer CW308 UFO baseboard. The ChipWhisperer CW1200 capture box generated the base clock, and the clock frequency was set to 50 MHz. The microcontroller was connected to a Keysight DSOX6004A oscilloscope at a sampling rate of 1.33 GSa/s for acquiring side-channel traces. The number of traces acquired was 100,000 each for the training and attack phases. We used a selection function of $Z^{(k)} = \mathrm{Sbox}(T \oplus k)$, as in many previous studies on (DL-)SCA on ASCAD and AES software implementations.

Hardware target. We used a public dataset released by Ito et al.[7], which was also used in the evaluation in [IUH22b]. The AES hardware implementation, presented in [UHA17], employs threshold implementation (TI) as a masking scheme [NRS11, RBN+15] with a byte-serial architecture. The implementation was conducted on a Xilinx Kintex-7 FPGA on a SAKURA-X board, and its power traces were acquired at a sampling rate of 455 MSa/s. The number of traces was 1,500,000 for training and 500,000 for the attacks. Similar to [IUH22a], we used a selection function targeting the register transition between the first and fifth bytes of the inversion output, denoted by

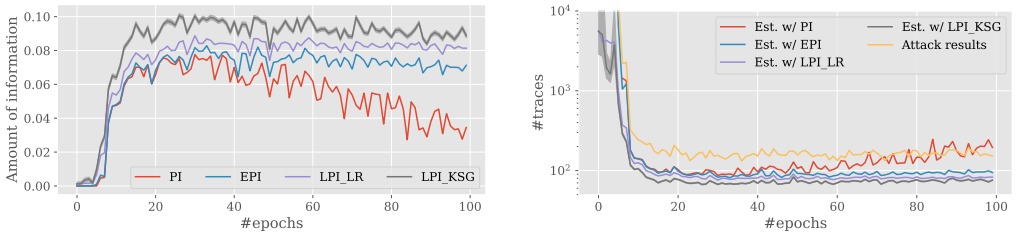
$$Z^{(k[1],k[5])} = \mathrm{Inv}(\Delta_f(T[1]) \oplus \Delta_f(k[1])) \oplus \mathrm{Inv}(\Delta_f(T[5]) \oplus \Delta_f(k[5])),$$

where $T[1]$ and $T[5]$ represent the first and fifth byte of plaintext, respectively; $k[1]$ and $k[5]$ represent the first and fifth byte of the secret key, respectively; Δ_f represents the isomorphic mapping from AES field (i.e., $\mathrm{GF}(2^8)$) to Canright's tower field (i.e., $\mathrm{GF}(((2^2)^2)^2)$) [Can05]; and Inv represents the $\mathrm{GF}(((2^2)^2)^2)$ inversion in Canright's Sbox (See [UHA17] for the details). We must guess two consecutive key bytes to employ an XOR-based selection function. Hence, the partial key length in the attack is $n = 16$ for the AES hardware implementation in our experiment.

Neural networks for $q_{\hat{\theta}}(Z \mid X)$ and its training. For evaluating the software implementation, we employ an NN architecture presented in [WAGP20] for implementing it with a random delay measure of 50 sample points. We set the learning rate, batch size, and maximum number of epochs to 1e−2, 512, and 100, respectively. We normalize the side-channel traces using `feature standardization` presented by Wouters et al. in [WAGP20]. For the hardware implementation, we used another neural network architecture proposed in [IUH22b] and set the learning rate, batch size, and maximum number of epochs to 1e−2, 1000, and 500, respectively. We normalize the side-channel traces using `feature scaling between -1 and 1` proposed in [WAGP20].

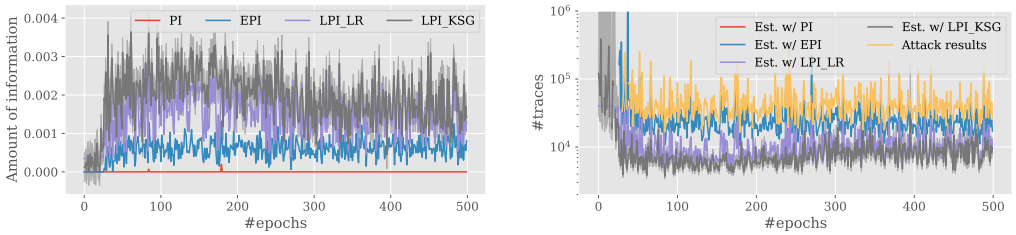
LPI estimation. In the LPI estimation using the KSG estimator described in Section 5.1, it is necessary to determine the number of samplings n_b and batch size m'. In this experiment, for the evaluation of the software implementation, $n_b = 100$ and $m' = 8,192$

[7] The dataset of side-channel traces is publicly available at https://github.com/ECSIS-lab/perceived_information_revisited/tree/main. The AES hardware is publicly available at https://github.com/ECSIS-lab/curse_of_re-encryption/tree/main/Masked_AES_hardware.



(a) PI values. **(b)** Number of traces to achieve an SR of 90%.

Figure 3: DL-SCA on the AES software implementation.



(a) PI values. **(b)** Number of traces to achieve an SR of 90%.

Figure 4: DL-SCA on the AES hardware implementation.

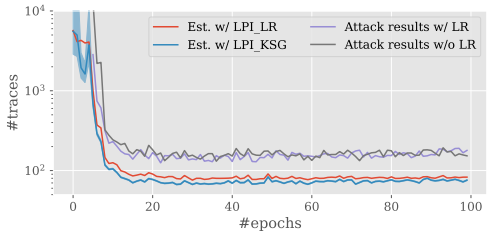
were used. For the hardware implementation evaluation, $n_b = 1,000$ and $m' = 8,192$ were used. All test traces were used to evaluate the KSG estimator. We used the KSG estimator provided by the Python library NPEET.

For the LPI estimation using logistic regression (i.e., Algorithm 2), the test set was used for evaluation, with $c = 5$ for both the software and hardware implementations. The training and inference of logistic regression were performed using the LogisticRegression class from cuML [Tea23], which is a GPU-accelerated machine learning library compatible with Scikit-learn [PVG+11]. We changed the hyperparameter C, which determines the strength of regularization. For C, we tried all of $\{0.01, 0.1, 1\}$ and used the one with the lowest average NLL in cross-validation as the estimated LPI.

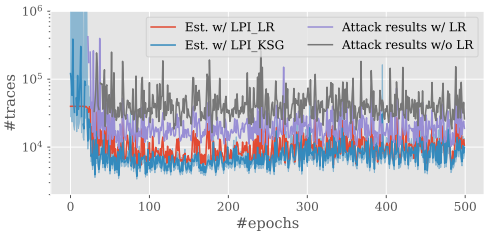
6.2 Evaluation result

Figures 3 and 4 report experimental results on the AES software and hardware implementations, respectively. Here, Figures 3(a) and 4(a) display the estimated PI, EPI, and LPI values during training, while Figures 3(b) and 4(b) display the empirical and estimated numbers of traces to achieve an SR of 90% using the model and inequalities. In Figures 3 and 4, LPI_LR and LPI_KSG mean the LPI values estimated by logistic regression and KSG estimator, respectively. Precisely, Figures 3(a) and 4(a) show the estimated PI, EPI, and LPI values during training, where the horizontal axis represents the number of epochs and the vertical axis represents the amount of information. Figures 3(b) and 4(b) show the empirical and estimated numbers of traces to achieve an SR of 90% by the model at each epoch. The red, blue, purple, and gray curves correspond to the number of traces required to achieve the SR estimated from the PI, EPI, LPI with logistic regression, LPI with KSG estimator, respectively, using the corresponding inequality. The yellow line denotes the empirical number of traces in an actual attack using the model.

We first focus on the results of AES software implementation. From Figure 3(a), we observe that the PI value increases over the first 20 epochs and decreases subsequently.



(a) Attack results on the AES software implementation.

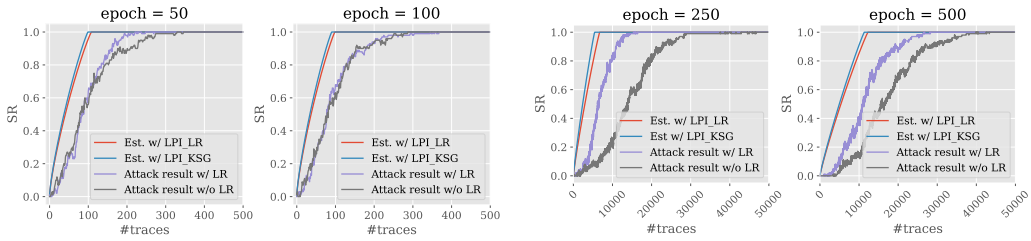


(b) Attack results on the AES hardware implementation.

Figure 5: Number of traces to achieve an SR of 90% by improved attack.

The values of EPI and LPI do not (relatively) decrease after 20 epochs. Therefore, after 20 epochs, the PI indicates a degradation in the attack performance of the model, whereas the EPI and LPI indicate a slight change in attack efficacy. In fact, Figure 3(b) indicates that the number of traces estimated from PI exceeds the number of traces in the actual attack at the last epoch, thereby disproving the conjecture on the PI–SR inequality in the experiment, in addition to the theoretical argument. In contrast, the number of traces estimated from the EPI and LPI are smaller compared to those for the actual attack, and they correlate proportionally. The attack performance can be estimated from the LPI and EPI values through SR inequalities. The results show that PI $<$ EPI $<$ LPI holds, and the difference between LPI and EPI is relatively small. Therefore, the EPI is appropriate for predicting attack performance, and the EPI–SR inequality is valid. In addition, it can be seen that the difference between the LPI values estimated by the KSG estimator and by logistic regression is small. In other words, this suggests that the classifier obtained by logistic regression is close to the optimal one.

Next, we focus on the results of the AES hardware implementation. First, from Figure 4(a), we can see that PI is zero for almost all epochs. If the PI–SR inequality holds, this means that the attack will fail for almost all epochs. However, as shown in Figure 4(b), the attack succeeds with the models trained from around the 30th epoch onwards, indicating that the PI–SR inequality does not hold. While PI is almost zero, EPI and LPI are significantly greater than zero in the epochs where the attack succeeded. In fact, as shown in Figure 4(b), the number of traces estimated from either EPI or LPI provides a lower bound on the number of traces actually required for a successful attack. However, the difference between EPI and LPI is large, and the lower bound estimated from EPI is more accurate. While the LPI–SR inequality has been proven, the EPI–SR inequality is only a conjecture and has not been proven. Furthermore, if LPI and EPI are significantly different, why the EPI–SR inequality holds is still unclear and has not been proven from our analyses. Nevertheless, at least within this experiment, it was found that both the EPI–SR inequality and the LPI–SR inequality hold[8]. Finally, when we look at the differences in the methods used to estimate LPI, we can see that the difference between the estimation results of LPI_LR and LPI_KSG is small, and the difference in the estimation of the number of traces required for an attack is also small. This indicates that the LPI value can be accurately estimated by logistic regression instead of MI estimators.



(a) SRs on the AES software implementation at the 50th and 100th epochs.

(b) SRs on the AES hardware implementation at the 250th and 500th epochs.

Figure 6: SRs at the middle and the end of the model training.

6.3 Improved attack using logistic regression as classifier

Figure 4(b) has shown that the LPI–SR inequality is clearly loose. LPI is the mutual information between the output of the feature extractor $\boldsymbol{F}$ and the intermediate value Z. In other words, the LPI–SR inequality ignores the model's classifier. Therefore, the reason that the LPI–SR inequality is loose in Figure 4(b) may be due to the poor performance of the classifier in the trained model. In other words, in this case, the attack performance is likely to be improved if we can improve the classifier.

Accordingly, we trained a logistic regression model on the training set to improve the performance of the classifier and used it to calculate the number of traces required for key recovery. Specifically, we first input the training data into the pre-trained model to obtain the feature maps. Next, we used these feature maps to perform cross-validation of the logistic regression and selected a regularization parameter C from the set $\{0.01, 0.1, 1\}$. We then used the obtained regularization parameter C to train the logistic regression model on the entire training data. This trained logistic model was then used as the last layer classifier, and we performed attacks on test traces.

Figures 5 and 6 show the experimental attack results. Figures 5(a) and 5(b) present the results of attacks on AES software and hardware implementations for all epochs, respectively. Figures 6(a) and 6(b) show the SRs of attacks on AES software and hardware implementations using the trained models at the middle and the end of the model training as examples (i.e., 50th and 100th epochs for the software implementation, and 250th and 500th epochs for the hardware implementation). For comparison, the results of attacks using the original trained models are also shown in these figures. From these figures, we find that the attack performance on the hardware implementation is significantly improved in most epochs, while the performance improvement seems trivial for the software implementation. This indicates that the looseness of the LPI–SR inequality in the hardware implementation evaluation shown in Figure 4 is due to the poor performance of the model's classifier. In fact, when logistic regression is used as the output layer, the LPI–SR inequality provides a more precise lower bound evaluation. This indicates that the LPI–SR inequality allows us to estimate the room for performance improvement for a given model. Additionally, these figures show the logistic regression performed well without overfitting, even though the training data for NN model training was also used to train the logistic regression model. This is likely because the logistic regression is a linear model with limited capacity, which suppresses overfitting. These results demonstrate that the proposed LPI–SR inequality can accurately estimate the model's performance based on the NN model's feature extractor.

[8] At epoch 270, the EPI–SR inequality does not appear to be satisfied, but in fact, the attack fails at this point and there is no plot in "Attack results". Therefore, the EPI–SR inequality holds at this epoch as well.

7 Concluding remarks

This study conducted an information theoretical analysis of DL-SCA. We developed a communication channel model for DL-SCA and defined a novel metric called LPI. Based on this model, we derived the LPI–SR inequality. Then, we discussed the relationships among LPI, PI, and EPI to validate the conjecture on the EPI–SR inequality through the LPI–SR inequality, and we commented on the similarity between LPI and EPI. For the practical computation of LPI, we proposed two methods to estimate LPI using a KSG estimator and logistic regression. Finally, we conducted experimental DL-SCAs on masked AES software and hardware implementations to validate the correctness of our analyses. Within the scope of our experiments, it was found that EPI provides an accurate lower bound on the number of traces for successful attacks using a given model. However, the EPI–SR inequality lacks proof, and in the current situation, it does not provide a theoretical guarantee that this lower bound is always correct. In contrast, the LPI–SR inequality is formally proven, ensuring a theoretically guaranteed lower bound. Additionally, it can enhance attack performance by improving the classifier. In this sense, the LPI offers a more reliable security evaluation than the EPI.

The experiments in this study were conducted for cases in which the selection function has good properties, such as bijection. Major practical ciphers, including AES, are included in this scope. However, the generality of our analysis for other cases is unclear. In the future, the validity of our analysis for other ciphers should be investigated. Furthermore, clarifications regarding the accuracy of the proposed LPI estimation method and the development of better alternatives should be pursued in the future.

Acknowledgments

We appreciate Olivier Rioul and Julien Béguinot for their inspiring discussion, which enhanced this study. This research was supported by JSPS KAKENHI Grant No. 22H04999/21H04867 and JST CREST Grant No. JPMJCR19K5, Japan.

References

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004*, Lecture Notes in Computer Science, pages 16–29, Berlin, Heidelberg, 2004. Springer.

[BIK+24] Elie Bursztein, Luca Invernizzi, Karel Král, Daniel Moghimi, Jean-Michel Picod, and Marina Zhang. Generalized power attacks against crypto hardware using long-range deep learning. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2024:472–499, 2024.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[BPS+20] Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas. Deep learning for side-channel analysis and introduction to ASCAD database. *Journal of Cryptographic Engineering*, 10(2):163–188, 2020.

[Can05] D. Canright. A very compact S-box for AES. In *International Workshop on Cryptographic Hardware and Embedded Systems*, volume 3659 of *Lecture Notes in Computer Science*, pages 441–455. Springer, 2005.

[CDP17] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Convolutional neural networks with data augmentation against jitter-based countermeasures. In *Cryptographic Hardware and Embedded Systems – CHES 2017*, volume 10529 of *Lecture Notes in Computer Science*, pages 45–68. Springer, 2017.

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In *International Workshop on Cryptographic Hardware and Embedded Systems*, LNCS, pages 13–28, 2002.

[CT06] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, USA, 2006.

[dCGRP19] Eloi de Chérisey, Sylvain Guilly, Olivier Rioul, and Pablo Piantanida. Best information is most successful: Mutual information and success rate in side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(2):49–79, 2019.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[GS18] Vincent Grosso and François-Xavier Standaert. Masking proofs are tight and how to exploit it in security evaluations. In *EUROCRYPT (2)*, pages 385–412. Springer, 2018.

[HHGG20] Benjamin Hettwer, Tobias Horn, Stefan Gehrer, and Tim Güneysu. Encoding power traces as images for efficient side-channel analysis. In *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 46–56, 2020.

[HRG14] Annelie Heuser, Olivier Rioul, and Sylvain Guilley. Good is not good enough - deriving optimal distinguishers from communication theory. In *CHES*, pages 55–74. Springer, 2014.

[IUH21] Akira Ito, Rei Ueno, and Naofumi Homma. Toward optimal deep-learning based side-channel attacks: Probability concentration inequality loss and its usage. Cryptology ePrint Archive, Report 2021/1216, 2021. https://ia.cr/2021/1216.

[IUH22a] Akira Ito, Rei Ueno, and Naofumi Homma. On the success rate of side-channel attacks on masked implementations: Information-theoretical bounds and their practical usage. In *ACM SIGSAC Conference on Computer and Communications Security (CCS 2022)*, pages 1521–1535, 2022.

[IUH22b] Akira Ito, Rei Ueno, and Naofumi Homma. Perceived information revisited: New metrics to evaluate success rate of side-channel attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(4), 2022.

[LZC+21] Xiangjun Lu, Chi Zhang, Pei Cao, Dawn Gu, and Haining Lu. Pay attention to raw traces: A deep learning architecture for end-to-end profiling attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(3):235–274, 2021.

[MDP20] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. A comprehensive study of deep learning for side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(1):348–375, 2020.

[MHM14] Zdenek Martinasek, Jan Hajny, and Lukas Malina. Optimization of power analysis using neural network. In Aurélien Francillon and Pankaj Rohatgi, editors, *Smart Card Research and Advanced Applications*, pages 94–107, Cham, 2014. Springer International Publishing.

[MMZ23] Anqi Mao, Mehryar Mohri, and Yutao Zhong. Cross-entropy loss functions: Theoretical analysis and applications. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

[MPP16] Houssem Maghrebi, Thibault Portigliatti, and E. Prouff. Breaking cryptographic implementations using deep learning techniques. *Security, Privacy, and Applied Cryptography Engineering (SPACE)*, 10076:3–26, 2016.

[NRS11] Svetla Nikova, Vincent Rijmen, and Martin Schläffer. Secure hardware implementation of nonlinear functions in the presence of glitches. *Journal of Cryptology*, 24(2):292–321, 2011.

[PBP21] Guilherme Perin, Ileana Buhan, and Stjepan Picek. Learning when to stop: A mutual information approach to prevent overfitting in profiled side-channel analysis. In *Constructive Side-Channel Analysis and Secure Design*, pages 53–81, 2021.

[PHJ+19] Stjepan Picek, Annelie Heuser, Alan Jovic, Shivam Bhasin, and Francesco Regazzoni. The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, (1):209–237, 2019.

[PPM+23] Stjepan Picek, Guilherme Perin, Luca Mariot, Lichao Wu, and Lejla Batina. SoK: Deep learning-based physical side-channel analysis. *ACM Computing Surveys*, 55(11):1–35, 2023.

[PSG16] Romain Poussier, François-Xavier Standaert, and Vincent Grosso. Simple key enumeration (and rank estimation) using histograms: An integrated approach. In *Cryptographic Hardware and Embedded Systems*, pages 61–81, 2016.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[RBN+15] Oscar Reparaz, Begül Bilgin, Svetla Nikova, Benedikt Gierlichs, and Ingrid Verbauwhede. Consolidating masking schemes. In *Advances in Cryptology—CRYPTO 2015*, pages 764–783, 2015.

[RSVC+11] Mathieu Renauld, François-Xavier Standeart, Nicolas Veyrat-Charvillon, Dina Kamel, and Denis Flandre. A formal study of power variability issues and side-channel attacks for nanoscale devices. In *Advances in Cryptology—Eurocrypt 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 109–128, 2011.

[SM23] Marvin Staib and Amir Moradi. Deep learning side-channel collision attack. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(3):422–444, 2023.

[SMY09] François-Xavier Standeart, Tal G. Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In *Advances in Cryptology—Eurocrypt 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 443–461, 2009.

[Tea23] RAPIDS Development Team. *RAPIDS: Libraries for End to End GPU Data Science*, 2023. https://rapids.ai.

[TUX+23] Yutaro Tanaka, Rei Ueno, Keita Xagawa, AKira Ito, Junko Takahashi, and Naofumi Homma. Multiple-valued plaintext-checking side-channel attacks on post-quantum KEMs. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(3), 2023.

[UHA17] Rei Ueno, Naofumi Homma, and Takafumi Aoki. Toward more efficient DPA-resistant AES hardware architecture based on threshold implementation. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, volume 10348 of *Lecture Notes in Computer Science*, pages 50–64, 2017.

[UXT+22] Rei Ueno, Keita Xagawa, Yutaro Tanaka, Akira Ito, Junko Takahashi, and Naofumi Homma. Curse of re-encryption: A generic power/EM analysis on post-quantum KEMs. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 1:296–322, 2022.

[WAGP20] Lennert Wouters, Victors Arribas, Benedikt Gierlichs, and Bart Praneel. Revisiting a methodology for efficient CNN architectures in profiling attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(3):147–168, 2020.

[WVdHG+20] Lennert Wouters, Jan Van den Herrewegen, Flavio D. Garcia, David Oswald, Benedikt Gierlichs, and Bart Preneel. Dismantling DST80-based immobiliser systems. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(2):99–127, 2020.

[ZBD+21] Gabriel Zaid, Lilian Bossuet, François Dassance, Amaury Habrard, and Alexandre Venelli. Ranking loss: Maximizing the success rate in deep learning side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(1):25–55, 2021.

[ZBHV19] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. Methodology for efficient CNN architectures in profiling attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(1):1–36, 2019.

[ZBHV21] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. Efficiency through diversity in ensemble models applied to side-channel attacks – a case study on public-key algorithms –. *IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES)*, 2021(3):60–96, 2021.

[ZZN+20] Jiajia Zhang, Mengce Zheng, Jiehui Nan, Honggang Hu, and Nenghai Yu. A novel evaluation metric for deep learning-based side channel analysis and its extended application to imbalanced data. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(3):73—-96, 2020.