SCAUL: Power Side-Channel Analysis With Unsupervised Learning

Keyvan Ramezanpour, *Student Member, IEEE*,
Paul Ampadu, *Senior Member, IEEE*, and William Diehl, *Member, IEEE*

Abstract—Existing power analysis techniques rely on strong adversary models with prior knowledge of the leakage or training data. We introduce side-channel analysis with unsupervised learning (SCAUL) that can recover the secret key without requiring prior knowledge or profiling (training). We employ an LSTM auto-encoder to extract features from power traces with high mutual information with the data-dependent samples of the measurements. We demonstrate that by replacing the raw measurements with the auto-encoder features in a classical DPA attack, the efficiency, in terms of required number of measurements for key recovery, improves by 10X. Further, we employ these features to identify a leakage model with sensitivity analysis and multi-layer perceptron (MLP) networks. SCAUL uses the auto-encoder features and the leakage model, obtained in an unsupervised approach, to find the correct key. On a lightweight implementation of AES on Artix-7 FPGA, we show that SCAUL is able to recover the correct key with 3,700 power measurements with random plaintexts, while a DPA attack requires at least 17,400 measurements. Using misaligned traces, with an uncertainty equal to 20 percent of the hardware clock cycle, SCAUL is able to recover the secret key with 12,300 measurements while the DPA attack fails to detect the key.

Index Terms—LSTM auto-encoder, power analysis, sensitivity analysis, side-channel analysis, unsupervised learning

1 Introduction

Side-Channel Analysis (SCA) using power consumption or electromagnetic (EM) emanations from electronic devices is a powerful tool for inferring information about hardware/software characteristics and processed data in a computing platform. Side-channel analysis refers to a technique in which behavior of a computing platform, including power consumption, EM radiation, timing and memory access, are observed to retrieve secret information. An SCA attack that analyzes the power traces or EM signals is usually referred to as power/EM analysis.

Power analysis (PA) has especially been employed to compromise the security of different crypto-systems running on a computing platform. Examples include secret key recovery from elliptic-curve cryptography (ECC) running on iOS and Android devices [1] and McEliece cryptosystem implemented on FPGA [2], attacks on Xilinx bitstream encryption [3], recovering the secret key of postquantum key exchange protocols [4], [5], key recovery of Advanced Encryption Standard (AES) [6], symmetric encryption systems [7], [8] and breaking the security of smart cards [9].

Existing power analysis techniques can be categorized into two groups of model-based and profiling attacks. In model-based attacks, prior knowledge of the leakage model is assumed that defines a relationship between the power consumption of a device and the processed data. In differential power analysis (DPA) [10], the measurements are clustered into two or more classes of similar traces, according to the leakage model. Statistics of the traces, e.g., the mean of power samples in first-order DPA, represent the traces in each cluster. The inter-cluster difference of the statistics is used as a measure to identify the correct data. In correlation power analysis (CPA) [11], the correlation coefficient between the power samples and the leakage model is used as the statistic to identify the correct data.

A commonly used leakage model is Hamming Weight (HW), according to which the power consumption of a logic block is correlated with the HW of the processed data [12]. Hamming Distance (HD) is also a popular model for power consumption corresponding to memory transitions, e.g., registers of microprocessors, in which the power is correlated with the HD between the initial and final values of memory elements [13]. Additionally, particular features of measured power traces might also be correlated with single bits of the data, e.g., the most significant bit (MSB) as used in [14]. Switching glitches in hardware implementation of logic functions and toggling activity of internal nodes of the circuit are also shown to depend on data [6], [15].

Model-based power analysis relies on a significant amount of prior knowledge of the details of the hardware including the hardware architecture, CMOS technology, the specific implementation of cipher operations, power delivery circuitry, and even the layout of interconnects in integrated circuits [16]. Profiling techniques use actual power measurements of a device corresponding to known processed data to develop more accurate leakage models. In [17], a stochastic model is employed in which a polynomial function of data with random coefficients represents the mean of the leakage signal assumed to follow a Gaussian

• *The authors are with the Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA 24061.*
E-mail: {rkeyvan8, ampadu, wdiehl}@vt.edu.

Manuscript received 15 Jan. 2020; revised 23 July 2020; accepted 26 July 2020. Date of publication 30 July 2020; date of current version 8 Oct. 2020. (Corresponding author: Keyvan Ramezanpour.)
Digital Object Identifier no. 10.1109/TC.2020.3013196

distribution. The coefficients are estimated using linear or ridge regression [17], [18].

Machine learning has been employed to develop leakage models in a profiling-based approach. Support vector regression (SVR) and multi-layer perceptron (MLP) neural networks are used in [19] and [20], respectively, to develop a mapping from the bitwise representation of data to the power samples. A partition-based approach is introduced in [21], in which a set of power traces corresponding to known data are clustered using an unsupervised clustering algorithm such as k-means. The clusters constitute the leakage model; the sets of data at each cluster have similar power consumption.

Whether a leakage model is obtained a priori or through profiling, power analysis also requires suitable statistics of the power traces as distinguishers. Classical techniques such as DPA and CPA use predefined statistics such as first or higher order moments and the correlation coefficient of power samples with a leakage model. Mutual information and KolmogorovSmirnov statistics are also used as alternative distinguishers [22], [23]. More recent techniques use profiling to extract suitable statistics. In template attacks [24], multivariate Gaussian distribution is assumed for the power traces. The mean and covariance of the distribution depends on the data and are estimated in a profiling step. More advanced techniques exploit supervised learning algorithms, such as support vector machine (SVM), decision tree (DT) and random forest (RF) [25], and deep learning [14], [26], to extract relevant features of power traces used as distinguishers.

While profiling and supervised learning techniques are the most powerful SCA attacks on cryptographic implementations, their success rate rapidly degrades if the training set, captured during profiling on a reference device, slightly deviates from the measurements on the target device under attack. It is shown in [27] that the accuracy of an MLP neural network in attacking AES running on an ATxmega128D4 microcontroller drops from 88.5 percent to less than 13.7 percent if the MLP is trained with power measurements on one board, and used to attack the same microcontroller, but on a different board. Having access to the identical hardware as the target is a major limitation of profiling-based techniques.

In this work, we introduce an unsupervised learning technique for side-channel analysis, called SCAUL, which does not require any training set for profiling or a prior knowledge of the leakage model. We employ a Long Short-Term Memory (LSTM) auto-encoder to extract features from power traces. An MLP neural network is used to map the power features to the processed data for a key candidate. Using *sensitivity* analysis, a leakage model is estimated for each key candidate. The power features are clustered based on the identified leakage model, and the correct key is chosen as the candidate that exhibits the maximum inter-cluster difference. We demonstrate the success of SCAUL on a lightweight implementation of AES on FPGA, even with non-aligned power traces.

The contributions of this work include: 1) We introduce an LSTM auto-encoder that extracts data-dependent features from measurements in an unsupervised approach. It allows for a horizontal processing of power traces which improves the efficiency of an attack significantly. 2) We develop sensitivity analysis with MLP neural networks to detect a leakage model in an unsupervised approach. 3) We introduce the SCAUL technique for power analysis which recovers the secret key without requiring prior knowledge on a leakage model or profiling, even when data-dependent features are distributed over random time samples.

The rest of the paper is organized as follows. In Section 2, an overview of prior work and different classes of existing power analysis techniques is presented. Section 3 provides the mathematical description of the SCAUL methodology. The realization of SCAUL with neural networks, including the LSTM auto-encoder for feature extraction and sensitivity analysis with MLP networks, is shown in Section 4. A case study of the SCAUL attack on an FPGA implementation of AES is discussed in Section 5. Experimental results are presented in Section 6 and the paper concludes in Section 7.

2 Background

2.1 Attack Model

A general model for power analysis involves a key-dependent cipher operation $F_k() : \mathbb{F}_2^m \rightarrow \mathbb{F}_2^n$, a known input data to the operation as $Z \in \mathbb{F}_2^m$ and an unknown variable at the output of the operation as $X \in \mathbb{F}_2^n$ called the intermediate or sensitive variable. In most block ciphers the operation under attack is the nonlinear S-box function $S()$ while m and n are the number of bits at the input and output of the operation. The input Z and the intermediate variable X are m, n-bit subsets of the input plaintext and the cipher state, respectively. Further, the secret parameter k is an m-bit subset of the entire secret key. Under this model, the S-box operation can be represented as $X = F_k(Z) = S(Z \oplus k)$, in which $\oplus$ is the bitwise XOR operation.

The fundamental property of a cipher operation exploited in power analysis to detect the secret key is *independence* of the output bits of the operation from the input. Formally, using the binary representation of the intermediate variable X as $\bar{\mathbf{x}} = (x_i)_{i=0,1,..,m}$, we have $H(\bar{\mathbf{x}}_r|Z) = H(\bar{\mathbf{x}}_r)$, in which $H()$ is the Shannon Entropy and $\bar{\mathbf{x}}_r$ is any combination of $r \in [1, m]$ bits of $\bar{\mathbf{x}}$. However, $H(\bar{\mathbf{x}}_r|Z, k) = 0$; i.e., having the secret key, the cipher operation is a deterministic relation while without knowledge on k, the operation is a random transformation. In power analysis, the power consumption of the hardware implementation of the cipher operation is given as a vector of N samples denoted by $\mathbf{T} \in \mathbb{R}^N$. It is assumed that $I(\mathbf{T}; \bar{\mathbf{x}}) > 0$, in which $I(a; b)$ is the mutual information between random variables a and b.

Using the above properties, the primary idea of a power analysis technique is as follows. Having a set of input values to the cipher operation and the corresponding power traces during execution of the operation, an attacker calculates the values of the intermediate variable for all possible values of the secret key k. Let $\bar{\mathbf{x}}_{k^*} = F_{k^*}(Z)$ denote the output of the cipher operation with the input Z and a key candidate k^*. If k^* is the correct key, $I(\mathbf{T}; \bar{\mathbf{x}}_{k^*}) > 0$ since $I(\mathbf{T}; \bar{\mathbf{x}}) > 0$, otherwise, $I(\mathbf{T}; \bar{\mathbf{x}}_{k^*}) = 0$. Hence, the mutual information between the power traces and the intermediate variable calculated for a key candidate can be considered as a metric to rank key candidates. The highest rank belongs to the correct key.

To implement a power attack in practice, the mutual information between the intermediate variable and power

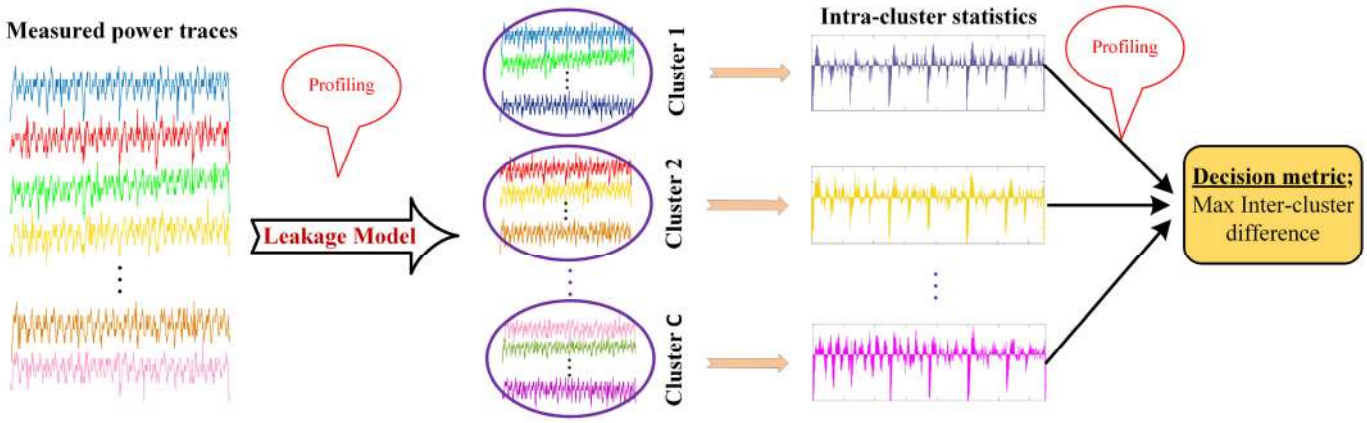

Fig. 1. Conceptual description of model-based power analysis techniques using a leakage model.

traces is usually expressed in the form of a leakage model. Let the function $L() : \mathbb{F}_2^m \rightarrow \mathbb{R}^N$ denote the leakage model which maps the m-bit intermediate variable X to the vector of power trace $\mathbf{T}$ with N samples. A generic nonlinear leakage model can be expressed as the following algebraic normal form [18]

$$\tilde{\mathbf{T}} = L(X) = \alpha_0 + \sum_{U \in \mathbb{F}_2^m \setminus \{0\}} \alpha_U X^U + \epsilon, \tag{1}$$

in which $\tilde{\mathbf{T}}$ is the data-dependent component of the leakage power, ϵ is a noise and $X^U = \prod_{i=1}^{m} x_i^{u_i}$ is a monomial of degree $d = HW(U)$ representing the product of bits of X at the positions where the corresponding bits of U are 1.

2.2 Model-Based Attacks

In classical model-based power analysis, including DPA and CPA, it is assumed that there is at least one sample of the power trace $\mathbf{T}$ which is correlated with the intermediate variable X according to (1), with fixed coefficients α_U for all samples. The power sample which exhibits the highest correlation with the leakage model is chosen as the point of interest (POI) for ranking the key candidates. In profiling-based attacks, different coefficients might be estimated for every sample of the power trace.

The overall procedure of a model-based attack is shown in Fig. 1. A set of S power traces each with N samples, denoted by $\mathbf{T}_j = (t_{j,1}, t_{j,2}, \ldots, t_{j,N}), j = 1, 2, \ldots, S$, with the corresponding inputs Z_j to a cipher operation $F_k()$ is available. For every key candidate k^*, the corresponding intermediate variable X for a power trace j is calculated as $X_{j,k^*} = F_{k^*}(Z_j)$. The power traces are grouped into C clusters, in general, according to the calculated intermediate values and the leakage model. For example, in an HW model, it is assumed that the power traces corresponding to the intermediate variables with the same HW are similar. Hence, the clusters are $\mathbb{H}_c = \{X_{j,k^*} | HW(X_{j,k^*}) = c\}$ with $c \in \{0, 1, 2, \ldots, m\}$ for an m-bit X, and $HW(X) = \sum_{i=0}^{m-1} x_i$ is the Hamming weight of X.

In classical techniques, statistics of the power traces in a cluster are calculated for every sample of the trace. In first order DPA, the mean of samples of power traces, i.e., $\bar{\mathbf{T}}_c = (\bar{t}_{c,1}, \bar{t}_{c,2}, \ldots, \bar{t}_{c,N})$, is used as the cluster statistic, in which $\bar{t}_{c,n} = \mathrm{E}_{X_{j,k^*} \in \mathbb{H}_c}[t_{j,n}], n = 1, 2, \ldots, N$. In a difference of means (DoM) test, the difference between the means of power samples between any two combinations of clusters is used as the statistic to rank the key candidate k^* [28]; the correct key exhibits the maximum difference. Alternatively, in mutual information analysis (MIA), the mutual information between the measurements and the model is used as the decision metric to rank the key candidates [29]. Higher order cluster statistics can also be used to attack low-order masked implementations as in [30] and [31]. The profiling-based leakage model can also be used in classical DPA attacks for the purpose of clustering as exploited in [18], [21].

Rather than clustering, in correlation power analysis, a mathematical model, as in (1), is used to characterize the data-dependent leakage [32], [33]. Thus, CPA can be considered as a generalization of DPA and MIA in which the number of clusters is equal to the space size of the intermediate variable X. The Pearson's correlation coefficient between the measured power and the estimated leakage according to the model is used to rank the key candidates. The coefficients of the leakage model can be assumed a priori, as in an HW or HD model, or obtained through profiling in a stochastic model.

Profiling techniques are also used in identifying proper cluster statistics and decision metrics in model-based attacks. A popular profiling power analysis is template attack (TA) in which the probability density function (pdf) of a cluster is estimated in a *profiling phase*, given a set of power traces corresponding to *known* intermediate variables [34]. These traces should be collected from a reference hardware, with known secret key, identical to the device under attack. The pdf of clusters are called *templates* whose parameters depend on data. During the *attack phase*, a measured power trace, from the device with unknown secret key, is matched with the templates using decision statistics such as maximum likelihood (ML) or Bayesian statistics, i.e., maximum a posteriori (MAP) estimation, to estimate the value of the intermediate variable. Profiling can be used for both leakage modeling and distribution estimation as in [19].

Classical model-based power analysis techniques assume aligned measurements in which the data-dependent features of power traces always appear at the same time sample which is a major limitation for two reasons: 1) timing of the measurements might not be precise relative to the timing of the device under attack; and 2) simple countermeasures such as addition of random clock jitters can result in the attack failure. Deep learning has been employed to

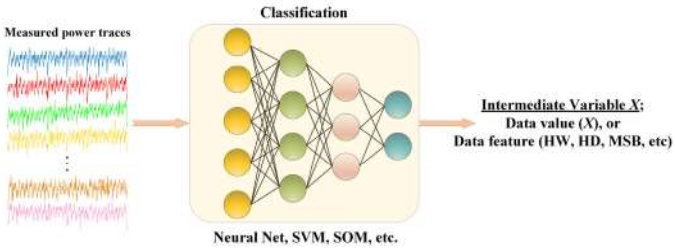

Fig. 2. Conceptual description of supervised power analysis techniques.

address the issue of misaligned power traces. In [14], a convolutional neural network (CNN) is used to classify power traces according to a leakage model, including HW and MSB, for every key candidate. By adding a small perturbation to the weights of a trained neural network for a key candidate, the sensitivity of cluster probabilities to the perturbation is calculated. The correct key exhibits the highest sensitivity.

2.3 Supervised Learning Attacks

Modern profiling-based techniques employ supervised learning to incorporate modeling of the leakage, extracting the proper statistics and decision metrics into a single algorithm, as shown in Fig. 2. In the terminology of supervised learning, the profiling phase of the attack corresponds to *training* and the attack or exploitation phase is equivalent to *inference* or *test*.

In supervised learning attacks, a set of power traces with known intermediate variables constitutes the training set. The label of a power trace, used in training, is the value of the corresponding intermediate variable. Supervised learning attacks have the flexibility to incorporate a leakage model if available. For example, if it is known that the power consumption is correlated with the HW of data, then the training labels would be the HW of the intermediate variable. Classical machine learning techniques use a leakage model for classification [35], [36]. Further, these techniques usually require a dimension reduction algorithm, such as principal component analysis (PCA), to select points of interest of power traces [25].

Power analysis based on deep learning has been shown to be the most powerful profiling attack. The major advantages of deep learning include: 1) dimension reduction algorithms are not necessary to identify POIs; the neural networks can learn the most relevant features and decision metrics for best classification, 2) the data-dependent features can be identified even in misaligned traces, and 3) higher-order statistics required to attack masked implementations can be identified by neural networks. In [26], convolutional neural networks, long short-term memory and a stacked auto-encoder are employed for feature extraction and classification. The results show that deep learning techniques outperform classical machine learning algorithms such as SVM and RF. Multi-layer perceptron neural networks have also been shown to outperform template attacks [37].

The SCA attacks based on supervised learning require a set of labelled training data collected from a reference device with known secret key. The optimal parameters of the neural networks are obtained to minimize the difference between the output of the neural network and the labels of the training set. Hence, the neural networks learn the leakage of the particular device from which the training set is collected. In the following, we describe the SCAUL methodology to learn the leakage of secret data without a training set.

3 SCAUL Methodology

The overall procedure of the proposed side-channel analysis with unsupervised learning (SCAUL) approach is shown in Fig. 3. The steps of SCAUL are explained below.

1) We encode the information content of all samples of power traces into an intermediate neural representation in an unsupervised approach. We employ an auto-encoder, realized with LSTM neural networks, for this purpose.
2) We use a sensitivity analysis for estimating a proper leakage model, using the features of the auto-encoder, for every key candidate. For this purpose, a multi-layer perceptron neural network is trained to estimate the bits of the intermediate variable from the power features for all key candidates.
3) Following the proposed sensitivity analysis, a slight perturbation is added to the weights of the trained MLP neural networks. The variation of data features at the output of the MLP as a result of the perturbation

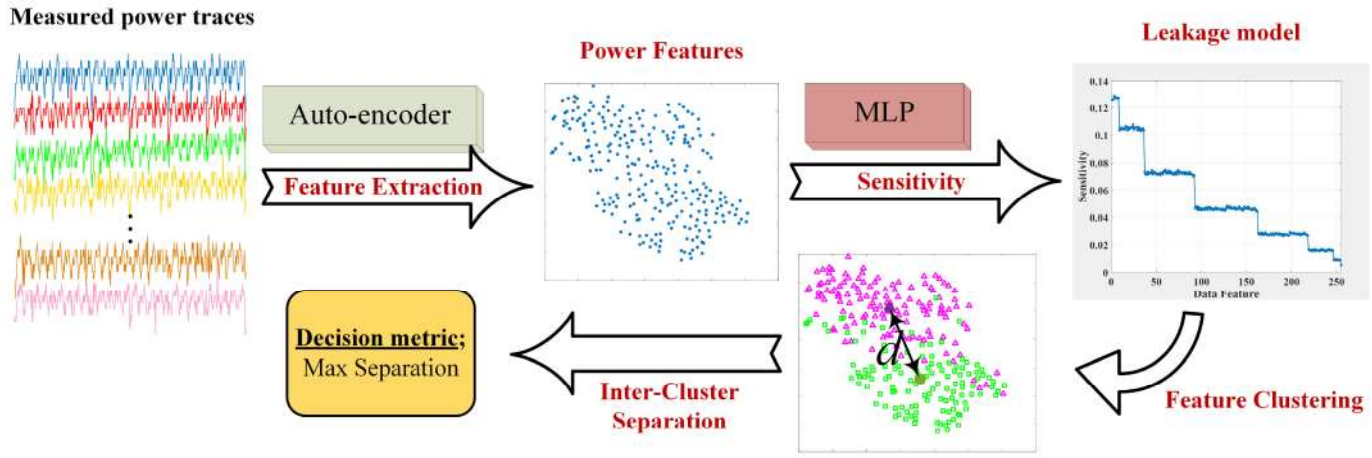

Fig. 3. Conceptual description of the proposed unsupervised learning technique for power analysis (SCAUL).

is measured. Data features with lowest variation constitute the leakage model.

4) The power features, extracted by the auto-encoder, are clustered based on the estimated leakage model for all key candidates. The key candidate that results in the highest inter-cluster difference is chosen as the correct key.

The mathematical background, supporting the claims of SCAUL, and the details on the structure of the neural networks used for realizing the SCAUL technique, are explained in the following sections.

3.1 Unsupervised Feature Extraction

We employ an auto-encoder to extract features from raw power measurements that include the information content of all power samples. Auto-encoders are the most prominent concepts of unsupervised learning in identifying data structure. Depending on the architecture of the encoder/decoder and the optimization goal, auto-encoders can be considered as generic nonlinear denoising filters and learning algorithms for estimating data distribution, identifying local manifold structure of data, and dimensionality reduction.

The basic concept of auto-encoders is simple. The input to the auto-encoder is $\hat{\mathbf{T}}$ which is a corrupted, or noisy, version of the original data $\mathbf{T}$. The encoder is a function $e() : \mathbb{R}^N \rightarrow \mathbb{R}^D$ that maps the input data into an internal representation space and the decoder $d() : \mathbb{R}^D \rightarrow \mathbb{R}^N$ maps the representation back into the input space to reconstruct the original data. The encoder and decoder functions are obtained by minimizing a loss function $\mathcal{L}(\mathbf{T}, \hat{\mathbf{T}})$ between the original and reconstructed data. In a denoising auto-encoder (DAE) [38], the optimization goal is to minimize the mean of the loss function, i.e., $\mathrm{E}[\mathcal{L}(\mathbf{T}, \hat{\mathbf{T}})]$. The most popular loss functions are the squared error and cross-entropy.

Auto-encoders with a proper optimization goal can also be considered as manifold learning algorithms. A contractive auto-encoder (CAE) uses a regularization mechanism in the optimization problem to restrict the space of encoder/decoder functions. The optimization goal of CAE is

$$e, d = \arg\min_{e,d} \mathrm{E}\left[\mathcal{L}(\mathbf{T}, \hat{\mathbf{T}}) + \lambda \left\| \frac{\partial e(\mathbf{T})}{\partial \mathbf{T}} \right\|_F^2\right], \tag{2}$$

in which $\|.\|_F^2$ is the Frobenius norm and λ is a design parameter. The first term in the above loss function is the reconstruction error and the second term is the regularization with λ providing a trade-off between them. If data is concentrated on a low-dimensional manifold, the CAE will learn a stochastic mapping from the input to the manifold. It is shown in [38] that DAE can also learn the data manifold when the dimension of the internal representation is constrained.

An alternative perspective to auto-encoders is an algorithm that extracts features with maximum mutual information with the data. Consider the encoder and decoder functions $e()$ and $d()$ with parameters W_e and W_d, respectively. The goal is to find features $\mathbf{f}$ that have maximum mutual information with the original data $\mathbf{T}$. The mutual information is

$$I(\mathbf{T}; \mathbf{f}) = H(\mathbf{T}) - H(\mathbf{T}|\mathbf{f}). \tag{3}$$

Assume the measurement $\hat{\mathbf{T}} = \mathbf{T} + \mathbf{N}$ is a corrupted version of the original data $\mathbf{T}$ with additive Gaussian noise $\mathbf{N} \sim \mathcal{N}(0, \Sigma)$, in which Σ is the covariance of the noise. We can demonstrate that the auto-encoder with the following objective function extracts features with maximum mutual information with the original data.

$$\hat{W}_e, \hat{W}_d = \arg\min_{W_e, W_d} \left\{ \mathrm{E}_{(\mathbf{T}, \hat{\mathbf{T}})} \left[(\hat{\mathbf{T}} - \mathbf{T})^T \Sigma^{-1} (\hat{\mathbf{T}} - \mathbf{T}) \right] + H(\mathbf{T}) \right\}. \tag{4}$$

The proof is provided in Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TC.2020.3013196. Assuming the auto-encoder has converged to the optimal solution, the first term in (4) is simply the mean-square error (MSE) between the input and output of the auto-encoder while the second term is the regularization on the entropy of the output.

By comparing the optimization goal of (4) with the generic objective function in (2), we note that optimizing the max information auto-encoder is a regularized maximum likelihood problem. The loss function, for a Gaussian random process, is simply the MSE between the measured and reconstructed data and the regularization is minimizing the entropy of the reconstruction. Minimizing the MSE for a Gaussian process is equivalent to maximum likelihood objective.

In this paper, we assume a multivariate Gaussian distribution for the measurements, and we employ MSE as the loss function of the auto-encoder. Instead of the entropy, we use the constraint on the dimensionality of the internal representation $\mathbf{f}$ as the regularization. Since the distribution of the output, i.e., $\hat{p}(\mathbf{T}; W_e, W_d)$, is constrained by the structure of the auto-encoder, lower dimensionality of the representation implies lower entropy of the output. Hence, under a Gaussian assumption for noise, the MSE auto-encoder can be considered as a sub-optimal solution for the max information auto-encoder.

According to the above discussion, the auto-encoder attenuates measurement noises which have small mutual information with the data-dependent features. The noisy components of measurements, if not attenuated, might add constructively, for an incorrect key candidate, which results in a larger inter-cluster difference than data-dependent features. Further, the auto-encoder encodes all data-related information of power traces into a low-dimensional internal representation. This is especially important with misaligned measurements where information about data is distributed over different samples.

3.2 Leakage Modeling With Sensitivity Analysis

Given the extracted features from power measurements, we identify which data features are encoded into the auto-encoder features using sensitivity analysis. We recall the leakage model of (1) in which the individual terms constitute the *data features*. We postulate that power traces with similar features are related to the intermediate values with similar data features.

For an m-bit intermediate variable X, the number of monomials X^U in (1) is equal to $2^m - 1$ with $U \in \mathbb{F}_2^m \backslash \{0\}$. Let $M_U() : \mathbb{R}^D \rightarrow \mathbb{U}_d$ be an unbiased estimator of a monomial X^U from power features, in which $\mathbb{U}_d$ is the field of

values for a monomial of degree $d = HW(U)$. The uncertainty in the estimate of a data feature X^U can be measured by the information content of power traces about the data.

A large amount of mutual information between the observation $\mathbf{f}$ and a parameter $\theta = X^U$ means that the conditional entropy $H(\mathbf{f}|\theta)$ is small, which implies that the conditional log-likelihood function $\log p(\mathbf{f}|\theta)$ is concentrated. The first derivative of a concentrated log-likelihood function, called the *score*, has large variations. The variance of the score is defined as Fisher information, i.e.,

$$\mathcal{I}(\theta) = \mathrm{E}_{\mathbf{f}}\left[\left(\frac{\partial}{\partial\theta}\log p(\mathbf{f}|\theta)\right)^2\right]. \tag{5}$$

The inverse of Fisher information sets a lower bound, called the *Cramer-Rao* bound, on the mean-square error of any unbiased estimate of the parameter θ. Hence, the uncertainty in the estimate of a data feature X^U is inversely proportional to the information content of measurements about data. If the power consumption is not correlated with a particular data feature X^{U^*}, i.e., small mutual information, the estimate of X^{U^*} has a large uncertainty. This is the basis of *sensitivity analysis* to identify those data features which are highly correlated with the power traces.

Assume we have the optimal estimator function $\hat{\theta} = M_U(\mathbf{f})$ of the monomial X^U. By adding a small perturbation to the optimal estimator, we obtain the perturbed estimate $\tilde{\theta}$. Similar to the CRB, we can demonstrate that the MSE between the optimal and the perturbed estimates is bounded by the inverse of the Fisher information. Hence, we get

$$\mathrm{E}_{\mathbf{f}}\left[(\tilde{\theta} - \hat{\theta})^2\right] \geq \mathcal{I}^{-1}(\theta). \tag{6}$$

The proof is shown in Appendix B, available in the online supplemental material. The left side of the inequality represents the sensitivity of an unbiased estimate of the parameter to small perturbations.

The above analysis shows that if the observations $\mathbf{f}$ have a large information content about a parameter θ, the sensitivity of an estimate $\hat{\theta} = M_U(\mathbf{f})$ to slight perturbations in the estimator function is small. This is consistent with the analysis of shrinkage amount of coefficients in a stochastic leakage model as discussed in [18].

3.3 Inter-Cluster Difference

The auto-encoder features include information about all processes running in parallel with the cipher operation under attack. As an example, in an FPGA implementation, the logic circuits corresponding to the state machine of the algorithm, embedded processors and clock network, all contribute to the power consumption of the FPGA chip. In supervised learning, the labelled training sets help neural networks identify the component of power consumption directly related to the operation under attack. Since the auto-encoder in SCAUL learns the information content of measurements in an unsupervised approach, the extracted features include the components of measurements relating to various processes.

To filter out the components of auto-encoder features carrying information about the secret data, we estimate a leakage model, for a key candidate. According to the attack model in Section 2.1, only the leakage model of the correct key has a nonzero mutual information with the measurements. The inter-cluster difference is a common metric in SCA to indirectly evaluate the mutual information between measurements and the leakage of secret data.

Assume the auto-encoder extracts features $\{\mathbf{f}_j\}_{j=1}^S$ from a number of S power traces. For a key candidate k, the values $\{X_j\}_{j=1}^S$ of the intermediate variable corresponding to the S power traces are calculated, and a leakage model $L^{(k)}()$ is estimated using the sensitivity analysis (as shown in the following section). Assume the mean of the estimated leakage is denoted by $\mu_L^{(k)} = \mathrm{E}_j[L^{(k)}(X_j)]$. We divide the power features into two clusters $C_0^{(k)} = \{\mathbf{f}_j | L^{(k)}(X_j) < \mu_L^{(k)}\}$ and $C_1^{(k)} = \{\mathbf{f}_j | L^{(k)}(X_j) > \mu_L^{(k)}\}$. The mean of features in the clusters is $\bar{\mathbf{f}}_c^{(k)} = \mathrm{E}_{\mathbf{f}_j \in C_c^{(k)}}[\mathbf{f}_j], c = 0, 1$. For a D-dimensional auto-encoder feature space, the mean features $\bar{\mathbf{f}}_c^{(k)} = \{\bar{f}_{c,i}^{(k)}\}_{i=0}^{D-1}, c = 0, 1$ are also D-dimensional. The correct key is the one with largest inter-cluster difference, i.e.,

$$\hat{k} = \arg\max_k \left(\max_i |\bar{f}_{0,i}^{(k)} - \bar{f}_{1,i}^{(k)}|\right). \tag{7}$$

According to the above discussion, SCAUL is a methodology to evaluate the mutual information between power measurements and the intermediate variable. In contrast, supervised learning SCA employs neural networks as *function approximators* that map the measurements to the space of the intermediate variable. As explained in the following section, SCAUL employs neural networks to extract information content of measurements (LSTM auto-encoder), estimate the leakage of the intermediate variable (sensitivity analysis on MLP networks) and find the part of the information that relates to the intermediate variable (inter-cluster difference).

4 Realizing SCAUL With Neural Networks

4.1 LSTM Auto-Encoder

Recurrent neural networks (RNN) are popular for learning temporal models of time series in applications such as natural language processing (NLP) [39], speech recognition and acoustic modeling [40]. A Long short-term memory neural network is a special type of RNN that can learn both local (short-term) and long-term temporal dependence of a signal. Convolutional neural networks (CNNs) have also been employed for processing time series. While CNNs are powerful in learning position invariance features, LSTM networks are stronger in learning temporal models [41].

The basic cell of an LSTM neural network is shown in Fig. 4. It consists of internal states $\mathbf{c}$ and $\mathbf{h}$, the latter of which is the output of the cell at every time instance. The input to the cell is processed by a fully-connected (FC) network with *tanh* activation. The internal state at each time instant is a weighted sum of the previous state and the processed input. The weighting process, called *gate*, controls the memory of the cell. The *forget gate* determines how much information of previous states is preserved at the current time instant and the *input gate* controls the amount of input activation to be added to the state. The output of the cell at time t, i.e., $\mathbf{h}_t$, is calculated by applying *tanh* activation to

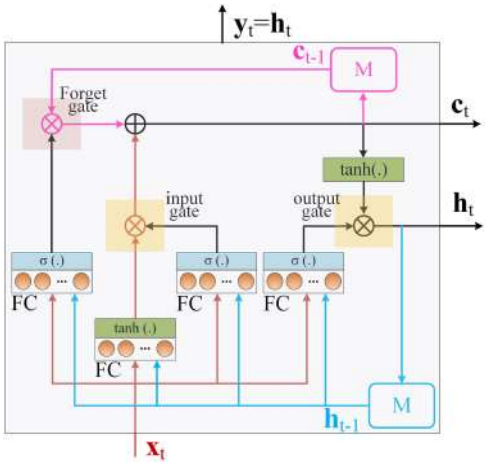

Fig. 4. The basic cell in long short-term memory (LSTM) neural network.

the internal state $\mathbf{c}_t$ followed by the *output gate*. This gate controls how much of the internal state activation flows to the output. The weights used in the above three gates are determined by three FC units with sigmoid activation. The input to these FC units is the current input to the cell, i.e., $\mathbf{x}_t$, and the previous output, i.e., $\mathbf{h}_{t-1}$.

The proposed LSTM auto-encoder for feature extraction is shown in Fig. 5. The encoder and decoder are LSTM networks with two layers shown in a *time-unrolled* representation; each of the encoder and decoder consists of a stack of two LSTM cells, as in Fig. 4, however, the processing steps of the cell are unrolled through time with the corresponding input to the cell shown explicitly at every time instant.

The inputs to the encoder are the samples of the power traces provided by a sliding window of length w and a stride of s; i.e., at every time instant, the input is a vector of w consecutive samples of the trace and the window is offset by s samples relative to the previous time instant. This is similar to a convolutional layer in CNNs. The first input to the decoder is zero while the following inputs are the individual samples of the power trace in the reverse order. The outputs of the decoder are the samples of the filtered power traces in the reverse order.

The loss function for optimizing the parameters of the auto-encoder is the MSE between the input of the encoder and output of the decoder. A constraint on the internal state of the encoder/decoder cells is considered as a regularization. After training the auto-encoder with all available power traces, the internal state $\mathbf{c}$ of the top-layer encoder cell is chosen as the features of power traces. We point out that the $\mathbf{c}$ state contains most of the information about the data. According to Fig. 4, the $\mathbf{h}$ state is derived from the $\mathbf{c}$ state. Further, the $\mathbf{c}$ state of the top layer also contains information processed by the preceding layers.

4.2 Leakage Modeling With MLP

After extracting data-dependent features from power traces with the auto-encoder, we estimate the intermediate variable from the features with a multi-layer perceptron neural network. We train an MLP network for every key candidate in which the input is the power features, and the output is the bits of the intermediate variable calculated for the key candidate.

The architecture of the MLP used in this work is shown in Fig. 6. It consists of three hidden layers with ReLU activation. The output layer is a set of m neurons, corresponding to m bits of the intermediate variable, with sigmoid activation. The number of neurons at each layer, used in our experiments for power analysis of AES, is shown in the figure. The size of power features, that is equal to the size of the $\mathbf{c}$ state of the LSTM auto-encoder, is 100, and the intermediate variable is one byte (8 bits) of the AES state at the output of an S-box.

To facilitate training of the MLP, the input power features are normalized over all measurements. Let $\mathbf{c}_j^{(2)}, j = 1, 2, \ldots, S$ denote the power features, extracted at the top-layer encoder cell of the auto-encoder in Fig. 5, corresponding to S power measurements. The input to the MLP is then

$$\tilde{\mathbf{f}}_j = \frac{\mathbf{c}_j^{(2)} - \min_j(\mathbf{c}_j^{(2)})}{\max_j(\mathbf{c}_j^{(2)}) - \min_j(\mathbf{c}_j^{(2)})}, \tag{8}$$

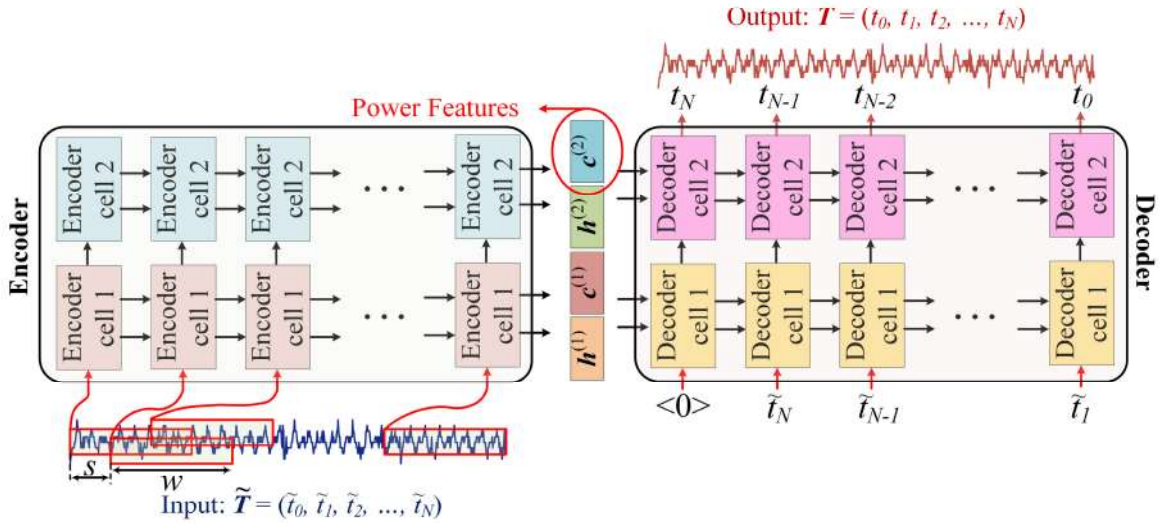

Fig. 5. Proposed LSTM auto-encoder for extracting features of power traces, with sliding window processing of input power traces and c state of top encoder cell selected as power feature.

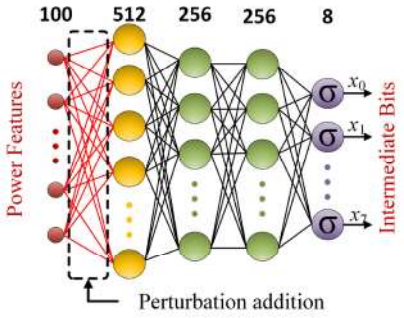

Fig. 6. Multi-layer perceptron (MLP) neural network for estimating bits of intermediate variable from power features and sensitivity analysis.

in which the normalization is carried out element-wise. Hence, the inputs to the MLP are within [0,1]. The outputs are the bits of the intermediate variable corresponding to the jth measurement denoted by $\mathbf{x}_j = (x_{j,0}, x_{j,1}, \ldots, x_{j,m-1})$ with values in $\{0,1\}$.

After training the MLP with the normalized power features in (8) and the corresponding intermediate variable $\mathbf{x}_j$, a small perturbation is added to the weights of the MLP. The perturbation is added at the first layer as shown in Fig. 6. Let $\mathbf{W}_{0,1}$ denote the weights of the trained MLP from input to the first hidden layer. The perturbed network has the weights $\tilde{\mathbf{W}}_{0,1} = \mathbf{W}_{0,1} + \boldsymbol{\delta}$ in which $\boldsymbol{\delta}$ is a small constant. The estimated intermediate variable at the output of the perturbed network is $\tilde{\mathbf{x}}_j = (\tilde{x}_{j,0}, \tilde{x}_{j,1}, \ldots, \tilde{x}_{j,m-1})$. We calculate the variation of the perturbation in a monomial $\tilde{X}_j^U = \prod_{i=0}^{m-1} \tilde{x}_{j,i} \cdot u_i$ for $U \in \mathbb{F}_2^m \backslash \{0\}$ as

$$\Delta_U = \mathrm{E}_j\left[\left|\tilde{X}_j^U - X_j^U\right|\right], \tag{9}$$

in which the expectation is over all measurements.

According to the analysis of Section 3.2, the lower variation of (9) implies that the data feature has a more significant contribution to the power consumption. Hence, in the leakage model of (1), we set

$$\hat{\alpha}_U = 1 - \frac{\Delta_U}{\max_U \Delta_U}. \tag{10}$$

We cluster the coefficients based on the variations of (9), and select the coefficients in the cluster with smallest variations as the leakage model. This is similar to the constraint on the degree of the model in the ridge regression technique of [18].

The power features are divided into two clusters based on the leakage model, obtained in an unsupervised approach. The correct key is chosen as the key candidate with the highest inter-cluster difference, as explained in Section 3.3. Inter-cluster difference is an indirect measure for the information content of power features about the secret data.

5 Case Study on AES

Advanced encryption standard is a worldwide standard for secret-key cryptography. Several block ciphers have also adopted structures similar to AES. In this section, we demonstrate the SCAUL attack on an FPGA implementation of AES. The principles of the attack are the same for any cipher with a key-dependent operation in which the input or output is known, and its power consumption is dependent on the processed data.

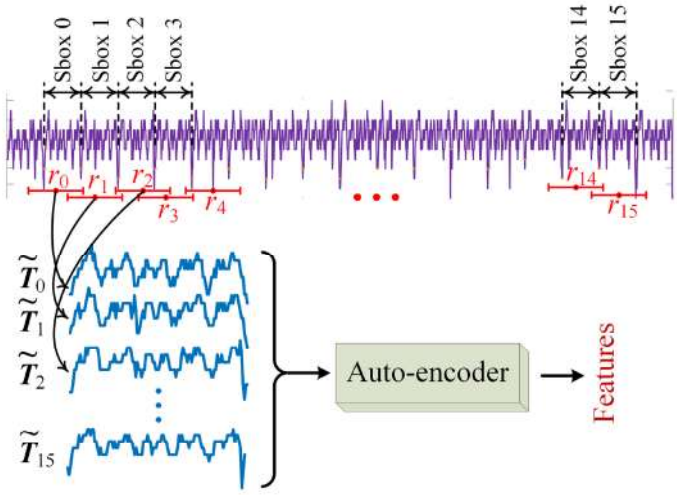

Fig. 7. Horizontal processing of power measurements during round 1 of AES by the auto-encoder.

The secret state of AES consists of 128 bits arranged in 4×4 bytes. The cipher operations are carried out in 10 rounds with a composition of four transformations: *AddRoundKey*, *SubBytes*, *ShiftRows* and *MixColumns*. At the beginning, the plaintext is loaded into the state and the secret key is added. The next operation is S-box, or *SubBytes*, which is a nonlinear transformation operating on individual bytes of the state. Next, *ShiftRows* and *MixColumns* operations follow to complete one round.

The target of a typical power attack on AES is the S-box operation in round 1. By denoting ith byte of the plaintext and the secret key as P_i and K_i, respectively, the key-dependent cipher operation under attack is $X_i = S(P_i \oplus K_i)$ in which $S()$ is the S-box function. The intermediate variable is X_i which is unknown but correlated with the power consumption during the operation of $S()$. Using the power measurements, the intermediate variable can be estimated. Hence, the input to the S-box, i.e., $P_i \oplus K_i$, can be calculated using the inverse S-box operation. Given the plaintext byte P_i, the corresponding byte of the secret key, i.e., K_i, will be recovered.

In our experiments, we use a lightweight implementation of AES (236 slices) on Artix-7 FPGA. The S-box function is implemented with a look-up table (LUT). At every clock cycle, the S-box is applied on one byte of the state. Hence, the power trace of round 1 corresponds to 16 S-box operations, as shown in Fig. 7. The power traces corresponding to S-box operations are selected using measurement windows of length l at positions $r_i, i = 0, 1, \ldots, 15$. The length of the windows are chosen based on the uncertainty in the timing of the measurements. If there is an uncertainty of Δl between the measurements and the clock signal of the hardware, the length l must be at least $l_c + \Delta l$, in which l_c is the length of a clock cycle, so that the power traces include all power samples of the corresponding S-box operation.

Processing of power traces as in Fig. 7 is similar to the *horizontal* attacks of [2], [4] in which similar patterns of power consumption through time, corresponding to the same key subset, are analyzed to recover the key. However,

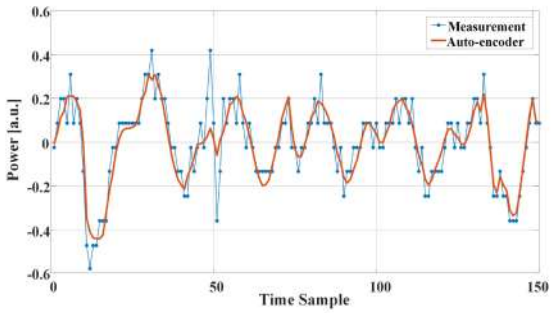

Fig. 8. Power trace measured during operation of a S-box and filtered by LSTM auto-encoder.

in the SCAUL attack on AES, the power traces correspond to different key subsets. The main mechanism enabling such horizontal processing of power traces in AES is the feature extraction via auto-encoders. The auto-encoder can identify data-dependent features irrespective of the value of the intermediate variable. Using all power traces through time improves the accuracy of feature detection, hence, significantly reduces the required number of power measurements to recover the key, as shown in the next section.

The filtering effect of the LSTM auto-encoder on power measurements is shown in Fig. 8. It is observed that while the underlying patterns of the power consumption are preserved at the output of the auto-encoder, strong noisy samples are filtered. The auto-encoder learns the patterns that repeat in most traces and filters out instantaneous variations that have low mutual information with the measurements. The extracted features from the power traces with an LSTM auto-encoder with 100 neurons in its FC components are also shown Fig. 9.

Since the LSTM auto-encoder has 100 neurons in the FC components, the extracted features also have a dimension of 100. The features are shown in the 2-dimensional plot of Fig. 9 using t-SNE algorithm [42]. Each point in the plot represents the mean of all features corresponding to the same intermediate variable. The non-uniform distance between the points reflects data-dependency; the intermediate values with similar power features result in *similar* power consumption. However, this similarity is not necessarily on individual samples of power traces. Instead, the data-dependent features of the traces, which might happen at different time samples, are similar.

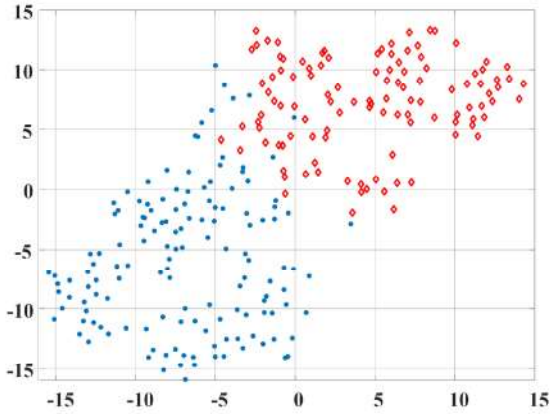

Fig. 9. Extracted features from power consumption of S-box operations in AES plotted in 2 dimensions using t-SNE algorithm.

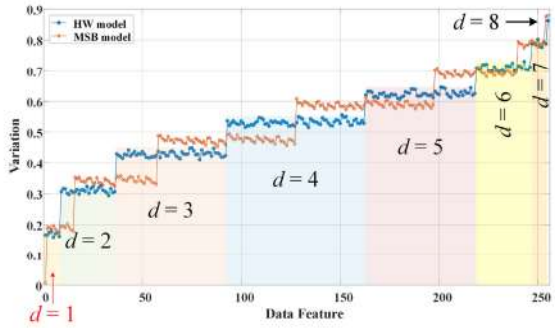

Fig. 10. Variation of data features calculated with sensitivity analysis on MLP neural network.

The power features extracted by the auto-encoder are mapped to the bits of intermediate variable using the MLP network of Fig. 6. According to the sensitivity analysis of Section 3.2, the coefficients of the leakage model are estimated. The variation of the monomials in (1) as a result of small perturbation on the weights of the trained MLP for the correct key candidate, calculated according to (9), is shown in Fig. 10. The curve labeled with "HW model" is obtained for actual power measurements on FPGA. It is observed that the variation of monomials corresponding to degree $d = 1$ are similar and lower than higher degree terms. This implies a HW leakage model; individual bits have the largest contribution to the power consumption with the same weights. This model can be verified using a DPA attack with an HW model, as shown in the next section.

To further verify the capability of the sensitivity analysis with MLP, we conduct a hypothetical experiment as follows. We group the power features into two clusters as shown in Fig. 9. Then, we assign the values of the intermediate variable with the most significant bit of 1 to one cluster and the values with MSB of 0 to the other cluster. We train an MLP with the power features and this hypothetical intermediate variable. The variation of the data features as a result of perturbation on the MLP weights is also shown in Fig. 10 labeled with "MSB model". It is observed that the variation corresponding to MSB (the first data feature) has the lowest variation. It implies that the power consumption is correlated with the MSB of the intermediate variable.

6 Experimental Results

We demonstrate the SCAUL attack on an FPGA implementation of AES using the Flexible Open-source workBench fOr Side-channel analysis (FOBOS) [43]. The FOBOS instance uses a NewAE CW305 Artix-7 FPGA target for the AES implementation, and Digilent Nexys 7 as the control board for synchronization with a host PC and target FPGA. We measure the power consumption of the target FPGA during encryption of multiple random plaintexts with 125 samples per clock cycle.

In all our experiments, we use the LSTM auto-encoder of Fig. 5 with 100 neurons in the FC components of both encoder and decoder cells. The sliding window at the input of the encoder has a length of 10 samples and a stride of 2. The LSTM auto-encoder and MLP neural network of Fig. 6, for sensitivity analysis, are implemented in Tensorflow. The

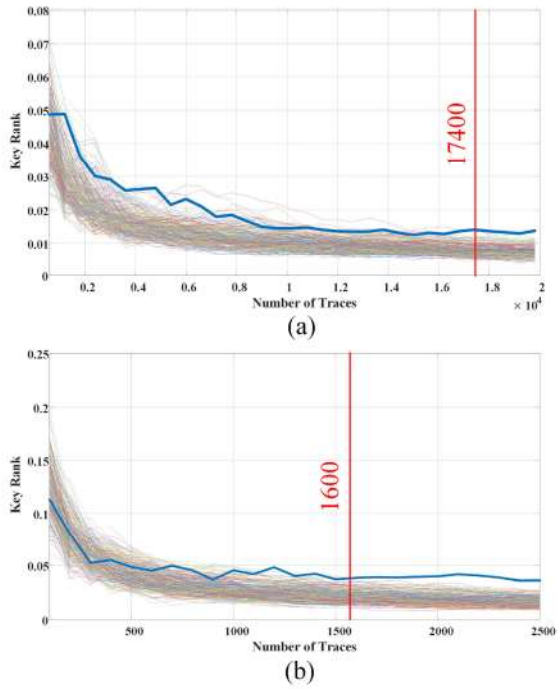

Fig. 11. Rank of key candidates versus number of traces in a DPA attack with HW leakage model; (a) maximum difference of power samples, (b) maximum difference of power features.

Adaptive moment estimation (Adam) algorithm is used for training all neural networks.

On a PC with Intel Core-i7 CPU, 16 GB RAM, and Nvidia GeForce GTX 1080 GPU, the LSTM auto-encoder takes around 20 minutes to train with more than 20K traces. Training of the MLP network with sensitivity analysis requires around 25 seconds for a key candidate and around 1.8 hours for all 256 possible values of the key candidate. The auto-encoder and the extracted leakage model are reused for all bytes of the entire secret key. Including clustering, the overall time for a SCAUL attack would take around 2.5 hours for recovering the entire 128-bit key of AES.

6.1 Power Analysis With Leakage Model

In the first experiment, we conduct a model-based power attack using the HW leakage model as a basis for comparing the performance of the proposed unsupervised learning approach in recovering the leakage model and the correct key. We employ a DPA attack using both the individual power samples, as in classical techniques, and the power features extracted by the LSTM auto-encoder. The latter reveals the capability of the auto-encoder in extracting data-dependent features.

The results of a classical DPA attack with the HW model are shown in Fig. 11a. For every key candidate the intermediate variable X is calculated based on which the power traces are grouped into two clusters C_0 and C_1; power traces corresponding to the intermediate variable X with $HW(X) < 4$ are in cluster C_0 and those with $HW(X) > 4$ belong to C_1. The mean trace of each cluster is obtained by averaging all traces in the cluster. The absolute values of the difference between samples of the mean traces in two clusters are calculated. The rank of a key candidate is the maximum absolute difference.

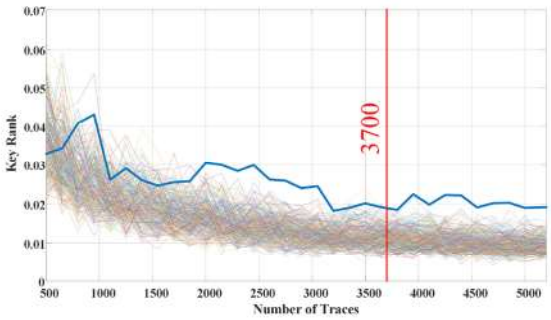

Fig. 12. Rank of key candidates versus number of traces in a SCAUL attack with a leakage model obtained with sensitivity analysis.

It is observed in Fig. 11a that with at least 17,400 power traces, corresponding to the power measurement during the encryption of 17,400 random plaintexts, the rank of the correct key is always larger than all incorrect key candidates in a DPA attack. Considering the fact that the attack is successful with the HW model also justifies the results of Fig. 10 in which the sensitivity analysis with the MLP suggests an HW leakage model.

To verify the effect of the auto-encoder, we repeat a DPA attack similar to the above attack by replacing the raw power traces with the power features of (8) extracted by the auto-encoder. The rank of key candidates versus the number of power measurements (encryptions) is shown in Fig. 11b. It is observed that using the power features, only 1,600 measurements are sufficient to identify the correct key, i.e., an improvement of more than $10\times$ in attack efficiency compared to classical DPA.

The significant effect of the auto-encoder in improving the performance of the DPA attack implies that the auto-encoder extracts the most relevant features of the power traces that depend on the processed data. The noisy samples of the power measurements add constructively in a DPA attack for some key candidates which results in large inter-cluster difference and hinders detection of the correct key. However, the auto-encoder has the ability to identify data-dependent features even in the presence of noise.

6.2 SCAUL Attack

The data-dependent features of power measurements extracted by the auto-encoder can also be used to identify the leakage model efficiently. We repeat a similar DPA attack with results shown in Fig. 11b but this time we employ the leakage model identified by the sensitivity analysis of Section 3.2 instead of the HW model. The rank of key candidates versus the number of power measurements is shown in Fig. 12. We notice that the correct key takes the highest rank if at least 3,700 measurements (encryptions) are available. This is a degradation of around $2.3\times$ in efficiency compared to a model-based attack.

We point out that the power features used in the SCAUL attack with the results in Fig. 12 are the same features as in Fig. 11b. The larger amount of measurements required in SCAUL compared to a model-based attack is the cost of detecting a proper leakage model. In other words, if a prior information is available, it can be used to achieve higher efficiency with the auto-encoder features. Otherwise, more measurements are required to retrieve the information.

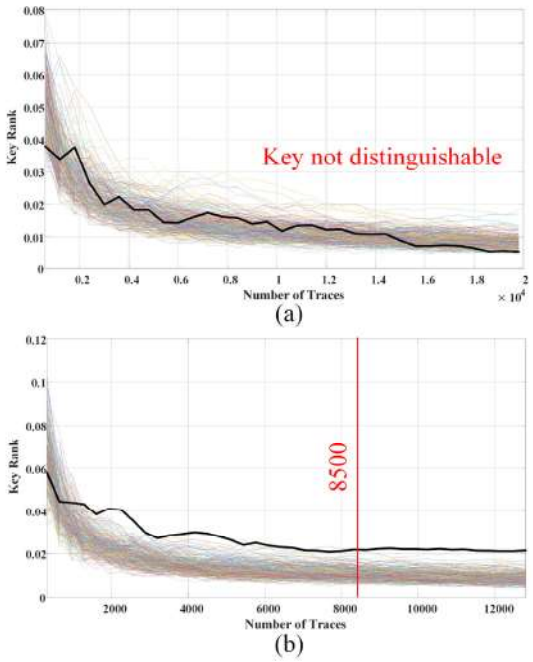

Fig. 13. Rank of key candidates versus number of traces in a DPA attack with HW leakage model and misaligned traces; (a) maximum difference of power samples, (b) maximum difference of power features.

6.3 Misaligned Traces

In addition to unsupervised leakage detection and significant efficiency improvement, another major advantage of SCAUL over classical power analysis techniques is the ability to recover the secret key even with non-synchronous measurements.

In this experiment, we let the measurement windows for extracting power traces of individual S-box operations, as shown in Fig. 7, take on a random shift. Specifically, we locate the windows on positions $r_i + s_i, i = 0, 1, \ldots, 15$ in which s_i's are uniformly distributed random variables in $[-12.5, 12.5]$ and r_i's are the accurate positions of S-box operations. It simulates a scenario in which the timing of the measurements is imprecise with an uncertainty equal to 20 percent of the hardware clock cycle. We set the length of the windows to 150 samples to cover the entire duration of a S-box operation amid the misalignment.

The result of a classical DPA attack with HW model and misaligned traces is shown in Fig. 13a. As expected the correct key is not distinguishable with 20K measurements since classical techniques assume data-dependent features appear at the same time sample. However, by using the auto-encoder power features, the correct key is recovered with only 8,500 measurements as shown in Fig. 13b. This experiment demonstrates that the data-dependent features are encoded into the internal representation of the auto-encoder even if they are spread over different time samples.

The result of a SCAUL attack with misaligned measurements and sensitivity analysis for leakage model detection is shown in Fig. 14. It is observed that even without using prior knowledge of a leakage model and with misaligned traces, SCAUL is able to recover the correct key with 12,300 measurements. Hence, the extracted features of the auto-encoder contain all information about data-dependent samples of the power traces sufficient for leakage detection.

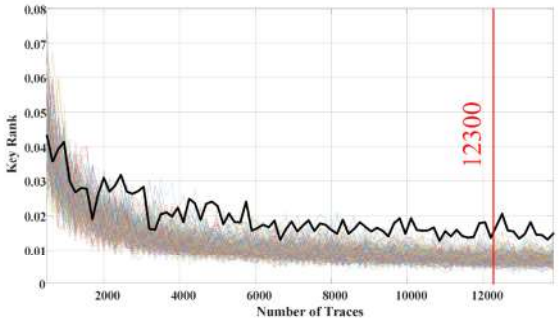

Fig. 14. Rank of key candidates versus number of traces in a SCAUL attack with a leakage model obtained with sensitivity analysis and misaligned power traces.

7 Conclusion

We introduced an unsupervised learning approach for side-channel analysis, called SCAUL, capable of extracting information about data processed on hardware without requiring prior knowledge on the leakage model or training data. At the heart of SCAUL, there is an auto-encoder that encodes the data-dependent samples of the power measurements into a neural representation with the highest mutual information with the secret data, and which is used for identifying a proper leakage model using sensitivity analysis. On a light-weight implementation of AES on Artix-7 FPGA, we demonstrated that an LSTM auto-encoder can improve the efficiency of a classical model-based DPA attack by $10\times$. We also showed that SCAUL is able to identify a proper leakage model from the auto-encoder features and recover the correct key with less than 3,700 measurements, compared to 17,400 traces required in a DPA attack. With imprecise measurements in which the timing uncertainty is around 20 percent of the hardware clock cycles, SCAUL can still recover the secret key with 12,300 measurements while classical DPA fails to detect the key with more than 20K traces.

Acknowledgments

This work was supported in part by NSF Award no. 1931639.

References

[1] D. Genkin, L. Pachmanov, I. Pipman, E. Tromer, and Y. Yarom, "ECDSA key extraction from mobile devices via nonintrusive physical side channels," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 1626–1638.

[2] C. Chen, T. Eisenbarth, I. von Maurich, and R. Steinwandt, "Horizontal and vertical side channel analysis of a McEliece cryptosystem," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 6, pp. 1093–1105, Jun. 2016.

[3] A. Moradi and T. Schneider, "Improved side-channel analysis attacks on xilinx bitstream encryption of 5, 6, and 7 series," in *Proc. Int. Workshop Constructive Side-Channel Anal. Secure Des.*, 2016, pp. 71–87.

[4] A. Aysu, Y. Tobah, M. Tiwari, A. Gerstlauer, and M. Orshansky, "Horizontal side-channel vulnerabilities of post-quantum key exchange protocols," in *Proc. IEEE Int. Symp. Hardware Oriented Security Trust*, 2018, pp. 81–88.

[5] M. J. Kannwischer, A. Genêt, D. Butin, J. Krämer, and J. Buchmann, "Differential power analysis of XMSS and SPHINCS," in *Proc. Int. Workshop Constructive Side-Channel Anal. Secure Des.*, 2018, pp. 168–188.

[6] S. Mangard, N. Pramstaller, and E. Oswald, "Successfully attacking masked AES hardware implementations," in *Proc. Int. Workshop Cryptographic Hardware Embedded Syst.*, 2005, pp. 157–171.

[7] A. Heuser, S. Picek, S. Guilley, and N. Mentens, "Side-channel analysis of lightweight ciphers: Does lightweight equal easy?" in *Proc. Int. Workshop Radio Frequency Identification: Secur. Privacy Issues*, 2016, pp. 91–104.
[8] T. Moos, A. Moradi, and B. Richter, "Static power side-channel analysis–An investigation of measurement factors," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 2, pp. 376–389, Feb. 2020.
[9] H. J. Mahanta, A. K. Azad, and A. K. Khan, "Power analysis attack: A vulnerability to smart card security," in *Proc. Int. Conf. Signal Process. Commun. Eng. Syst.*, 2015, pp. 506–510.
[10] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proc. Annu. Int. Cryptol. Conf.*, 1999, pp. 388–397.
[11] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Proc. Int. Workshop Cryptogr. Hardware Embedded Syst.*, 2004, pp. 16–29.
[12] M. Alioto, L. Giancane, G. Scotti, and A. Trifiletti, "Leakage power analysis attacks: A novel class of attacks to nanometer cryptographic circuits," *IEEE Trans. Circuits Syst. I: Reg. Papers*, vol. 57, no. 2, pp. 355–367, Feb. 2010.
[13] J.-S. Coron, C. Giraud, E. Prouff, S. Renner, M. Rivain, and P. K. Vadnala, "Conversion of security proofs from one leakage model to another: A new issue," in *Proc. Int. Workshop Constructive Side-Channel Anal. Secure Des.*, 2012, pp. 69–81.
[14] B. Timon, "Non-profiled deep learning-based side-channel attacks with sensitivity analysis," *IACR Trans. Cryptogr. Hardware Embedded Syst.*, vol. 2019, pp. 107–131, 2019.
[15] R. Sadhukhan, P. Mathew, D. B. Roy, and D. Mukhopadhyay, "Count your toggles: A new leakage model for pre-silicon power analysis of crypto designs," *J. Electron. Testing*, vol. 35, pp. 605–619, 2019.
[16] D. Das, M. Nath, B. Chatterjee, S. Ghosh, and S. Sen, "STELLAR: A generic EM side-channel attack protection through ground-up root-cause analysis," in *Proc. IEEE Int. Symp. Hardware Oriented Secur. Trust*, 2019, pp. 11–20.
[17] W. Schindler, K. Lemke, and C. Paar, "A stochastic model for differential side channel cryptanalysis," in *Proc. Int. Workshop Cryptographic Hardware Embedded Syst.*, 2005, pp. 30–46.
[18] W. Wang, Y. Yu, F.-X. Standaert, J. Liu, Z. Guo, and D. Gu, "Ridge-based DPA: Improvement of differential power analysis for nanoscale chips," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 5, pp. 1301–1316, May 2018.
[19] D. Jap, M. Stöttinger, and S. Bhasin, "Support vector regression: Exploiting machine learning techniques for leakage modeling," in *Proc. 4th Workshop Hardware Architectural Support Secur. Privacy*, 2015, Art. no. 2.
[20] S. Yang, Y. Zhou, J. Liu, and D. Chen, "Back propagation neural network based leakage characterization for practical security analysis of cryptographic implementations," in *Proc. Int. Conf. Inf. Secur. Cryptol.*, 2011, pp. 169–185.
[21] C. Whitnall and E. Oswald, "Robust profiling for DPA-style attacks," in *Proc. Int. Workshop Cryptographic Hardware Embedded Syst.*, 2015, pp. 3–21.
[22] B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel, "Mutual information analysis," in *Proc. Int. Workshop Cryptographic Hardware Embedded Syst.*, 2008, pp. 426–442.
[23] C. Whitnall, E. Oswald, and L. Mather, "An exploration of the kolmogorov-smirnov test as a competitor to mutual information analysis," in *Proc. Int. Conf. Smart Card Res. Adv. Appl.*, 2011, pp. 234–251.
[24] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *Proc. Int. Workshop Cryptogr. Hardware Embedded Syst.*, 2002, pp. 13–28.
[25] L. Lerman, G. Bontempi, and O. Markowitch, "Side channel attack: An approach based on machine learning," *Center Adv. Secur. Res. Darmstadt*, pp. 29–41, 2011.
[26] H. Maghrebi, T. Portigliatti, and E. Prouff, "Breaking cryptographic implementations using deep learning techniques," in *Proc. Int. Conf. Secur. Privacy Appl. Cryptogr. Eng.*, 2016, pp. 3–26.
[27] H. Wang, M. Brisfors, S. Forsmark, and E. Dubrova, "How diversity affects deep-learning side-channel attacks," in *Proc. IEEE Nordic Circuits Syst. Conf., NORCHIP Int. Symp. Syst.-on-Chip*, 2019, pp. 1–7.
[28] F.-X. Standaert, B. Gierlichs, and I. Verbauwhede, "Partition vs. comparison side-channel distinguishers: An empirical evaluation of statistical tests for univariate side-channel attacks against two unprotected CMOS devices," in *Proc. Int. Conf. Inf. Secur. Cryptol.*, 2008, pp. 253–267.
[29] L. Batina, B. Gierlichs, E. Prouff, M. Rivain, F.-X. Standaert, and N. Veyrat-Charvillon ,"Mutual information analysis: A comprehensive study," *J. Cryptol.*, vol. 24, no. 2, pp. 269–291, 2011.
[30] E. Prouff, M. Rivain, and R. Bevan, "Statistical analysis of second order differential power analysis," *IEEE Trans. Comput.*, vol. 58, no. 6, pp. 799–811, Jun. 2009.
[31] T. Schneider and A. Moradi, "Leakage assessment methodology," in *Proc. Int. Workshop Cryptographic Hardware Embedded Syst.*, 2015, pp. 495–513.
[32] A. Biryukov, D. Dinu, and J. Großschädl, "Correlation power analysis of lightweight block ciphers: From theory to practice," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.*, 2016, pp. 537–557.
[33] M. Repka and M. Varchola, "Correlation power analysis using measured and simulated power traces based on hamming distance power model-attacking 16-bit integer multiplier in FPGA," *Int. J. Comput. Netw. Inf. Secur.*, vol. 7, no. 6, 2015, Art. no. 10.
[34] O. Choudary and M. G. Kuhn, "Efficient template attacks," in *Proc. Int. Conf. Smart Card Res. Adv. Appl.*, 2013, pp. 253–270.
[35] S. Picek, A. Heuser, A. Jovic, and A. Legay, "Climbing down the hierarchy: Hierarchical classification for machine learning side-channel attacks," in *Proc. Int. Conf. Cryptol. Africa*, 2017, pp. 61–78.
[36] S. Picek *et al.*, "Side-channel analysis and machine learning: A practical perspective," in *Proc. Int. Joint Conf. Neural Netw.*, 2017, pp. 4095–4102.
[37] Z. Martinasek, L. Malina, and K. Trasy, "Profiling power analysis attack based on multi-layer perceptron network," in *Computational Problems in Science and Engineering*. Berlin, Germany: Springer, 2015, pp. 317–339.
[38] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 1096–1103.
[39] W. Yin, K. Kann, M. Yu, and H. Schütze, "Comparative study of CNN and RNN for natural language processing," 2017, *arXiv: 1702.01923*.
[40] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Proc. 15th Annu. Conf. Int. Speech Commun. Assoc.*, 2014, pp. 338–342.
[41] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2015, pp. 4580–4584.
[42] L. V. D. Maaten and G. Hinton,"Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. Nov, pp. 2579–2605, 2008.
[43] CERG, "Flexible open-source workbench for side-channel analysis (FOBOS)," Oct. 2016. [Online]. Available: https://cryptography.gmu.edu/fobos/

Keyvan Ramezanpour (Student Member, IEEE) is working toward the PhD degree in electrical engineering with the Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, Virginia. His areas of research include secure implementation of cryptographic algorithms on field programmable gate arrays (FPGA), Systems-on-Chip (SoC) and application-specific integrated circuits (ASIC), and unsupervised deep learning techniques for security assessment of hardware implementations. His research interests also include adaptive and reconfigurable circuits for addressing security vulnerabilities of cryptographic implementations to fault attacks and designing circuit-level primitives for true random number generators (TRNG), and physically unclonable functions (PUF).

Paul Ampadu (Senior Member, IEEE) received the PhD degree from Cornell University, Ithaca, New York, in 2004. He is a professor of ECE and co-director of the Advanced Research Institute at Virginia Tech, Blacksburg, Virginia. His research interests include secure and resilient NoC, SoC, and IoT. He is the author of three books and over a hundred research articles. He served for two terms on the IEEE CAS Board of Governors, and he is currently the PES representative on the IEEE IoT Council and IEEE-USA Committee on Transportation and Aerospace Policy. He is the co editor-in-chief of the *IEEE IoT Magazine*.

William Diehl (Member, IEEE) received the BA degree from Duke University, Durham, North Carolina, the MS degree from the Naval Postgraduate School, Monterey, California, and the PhD degree from George Mason University, Fairfax, Virginia. He is a research assistant professor with the Bradley Department of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University (Virginia Tech), Blacksburg, Virginia, and director of the Signatures Analysis Laboratory (SAL) at Virginia Tech, Blacksburg, Virginia. His area of research is secure and efficient implementations of cryptographic algorithms in hardware and software, including field programmable gate arrays (FPGA), Systems-on-Chip (SoC), microcontrollers, and soft core microprocessors.