# CS293 Report
# Side Channel Attack with Machine Learning

**Hongyuan You, Furkan Kocayusufoglu**
Department of Computer Science
{hyou,furkan}@cs.ucsb.edu

## Abstract

Side channel attacks (SCAs) have gained a large amount of attention in cryptography world in the past decades. An adversary may effectively reveal secret keys and sensitive data using leaking signals which are emitted from physical cryptographic devices, including power consumption, electromagnetic radiation and response times. For profiled side-channel attacks, template attack (TA) and machine learning based attack are two popular categories of approaches. While the most commonly used template attacks focus more on high-dimensional statistical modelling under Gaussian noise assumptions, machine learning encompasses tools to solve problems in more flexible scenarios and provides more efficient apparatus for side-channel attacks. The underlying intuition is that, recovering secret keys from side signals in cryptography is similar to learning target function from sampled data in machine learning. In this survey, we will go through applicable machine learning methods which has been found to perform efficiently in side channel attacks.

Side channel attacks are well known nowadays and has traditionally been exploited with simple power analysis (SPA) and differential power analysis (DPA) [1]. While SPA and DPA have difficulties in varying keys and overlapping complicated signals, template attack (TA) and machine learning based attack (MLA) are potential alternatives to extract secret keys under more flexible conditions, and further have shown their success in attacking smart-card running DES [2] or (masked) AES [3, 4]. In this review, we will first describe how template attack works in a general SCA setting and a specific version of TA under particular multivariate Gaussian assumptions. Next we introduce machine learning techniques used in MLA, such as dimension reduction for leakage traces and binary classification for secret key bits. Though template attack can be thought of a machine learning approach as well, here we refer to template attack and other machine learning based attack separately in a sense that TA is parametric and MLA is nonparametric.

## 1 Template Attack

Template attack has been introduced by Chari [5] and improved to more efficient implementations. Intuitively, the larger amount of information we acquire from cryptographic devices, the more precise is the model of power consumption that we can estimate. Thus the fundamental assumption of template attack is that positions of leakage trace vectors closely correlate with the plaintext and key values that generated them. Notice that it is difficult to obtain enough training profiling traces from a targeted device, a clone device becomes an alternative information source, which should be very similar to the targeted device and kept under control of the adversary for the convenience of data acquisition. The first stage of TA will be performed on the clone device, and the learned model will be used for the second stage of TA on the targeted device.

**Problem Setting**: Consider a real device that runs an encryption or decryption algorithm on a known plaintext $p \in T$ with a $D$-bit binary secret key $k \in K = \{0,1\}^D$ which is stored on the device and generates a targeted intermediate signal $\boldsymbol{y}(p,k)$. The mapping $\boldsymbol{y}$ maps the plaintext (or the ciphertext) and the private key to a value which forms the deterministic or ground-truth part of a leakage signal, for instance, power consumption signal, or AES S-box outputs during the first round:

$$\boldsymbol{y}(p,k) = \mathit{Sbox}[p \oplus k] \tag{1}$$

where $\mathit{Sbox}[\cdot]$ is a non-linear substitution operation. The measured leakage signal $\boldsymbol{x}$ is generated by a device-specific unknown function $\varphi$ and an independent additive noise $\boldsymbol{r}$ as follows:

$$\boldsymbol{x}(p,k) = \varphi(\boldsymbol{y}(p,k)) + \boldsymbol{r} \tag{2}$$

For this particular pair of plaintext $p$ and key $k$, an adversary will track down the leakage signal from the clone device through $m$ measurements (as simulated in Fig. 1), which in result is denoted as a high-dimensional leakage trace vector $\boldsymbol{x}(p,k) \in \mathbb{R}^m$ in the Euclidean space. For each key, the adversary will perform such trace recording process for $n$ times and form a set of training profiling traces $\mathcal{X}_a(p,k) = \{\boldsymbol{x}_i(p,k) \in \mathbb{R}^m : i = 1, \cdots, n\}$ in order to estimate a leakage model as $\mathcal{P}[\boldsymbol{x}(p,k)|\theta(p,k)]$, where $\theta(p,k)$ completely specifies the probability distribution $\mathcal{P}$ and is named as a "template".

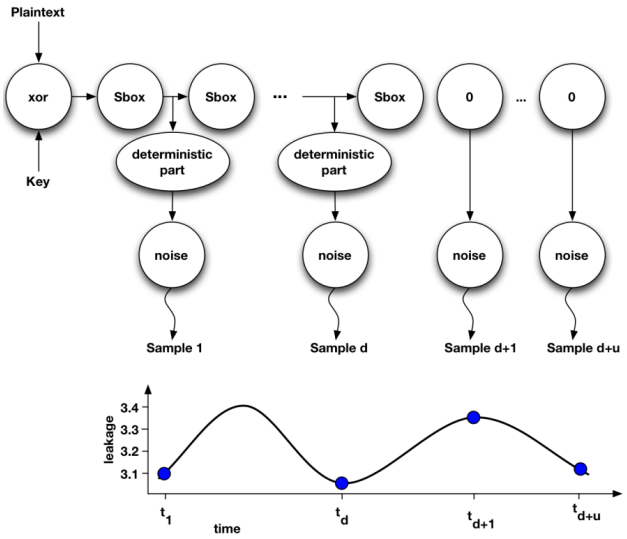


Figure 1: The diagram of a possible leakage simulator [6] used in template attack. Each time, sample $\boldsymbol{x}_i(t) \in \mathbb{R}$ (which is an entry of leakage trace vector $\boldsymbol{x}_i \in \mathbb{R}^m$) is defined as the sum of a deterministic part representing the intermediate value during encryption and a random Gaussian noise. The deterministic part of leakage signal corresponds to the output of S-box, iterated for each time sample. The adversary can record leakage traces with a particular sampling frequency.

**General TA Approach**: The template attack consists of two stages: the *profiling phase* (which is not presented in DPA) and the *attacking phase*, which are known as training and testing in statistical learning paradigm. In the profiling phase, the model parameters will be estimated from acquired traces $\mathcal{X}_a(p,k)$ through maximum likelihood approach. For a given pair of $(p,k)$, we have

$$\hat{\theta}(p,k) = \arg\max_{\theta} \Pi_{\boldsymbol{x} \in \mathcal{X}_a(p,k)} \mathcal{P}[\boldsymbol{x}(p,k)|\theta(p,k)] \tag{3}$$

Since the adversary obtains more than one pair of plaintext and secret key, a set of template $\{\hat{\theta}(p,k) : p \in \mathrm{T}, k \in \mathrm{K}\}$ will be built in the same manner. Next, in the attacking phase, the adversary will monitor both the input plaintext $p^*$ to the targeted device and the leakage signal $\boldsymbol{x}^*$. As the plaintext is known in advance, the adversary will gain more information about the secret key $k^*$ by estimating

the posterior probability $\mathcal{P}[\hat{\theta}(p^*, k)|\boldsymbol{x}^*]$ under the Bayes' rule:

$$\hat{k} = \arg\max_{k \in K} \mathcal{P}[\hat{\theta}(p^*, k)|\boldsymbol{x}^*] \tag{4}$$

$$= \arg\max_{k \in K} \frac{\mathcal{P}[\boldsymbol{x}^*|\hat{\theta}(p^*, k)]\mathcal{P}[\hat{\theta}(p^*, k)]}{\mathcal{P}(\boldsymbol{x}^*)} \tag{5}$$

$$= \arg\max_{k \in K} \mathcal{P}[\boldsymbol{x}^*|\hat{\theta}(p^*, k)]\mathcal{P}[\hat{\theta}(p^*, k)] \tag{6}$$

where the *apriori* probability $\mathcal{P}[\hat{\theta}(p^*, k)]$ should be estimated or assigned by the adversary and is set as a uniform distribution if there is no prior knowledge on it. The optimal $\hat{k}$ is picked so that it maximizes the probability that template $\theta(p^*, \hat{k})$ occurred.

**Multivariate Gaussian Case**: The most commonly used distribution used in template attack is multivariate normal distribution by which the time sample are assumed to be independent. In detail, if we assume $\mathcal{P}$ follows multivariate normal distribution $\mathcal{N}(\boldsymbol{x}|k; \boldsymbol{\mu}_{p,k}, \boldsymbol{\Sigma}_{p,k})$, then the likelihood of a trace $\boldsymbol{x}$ originating from $(p, k)$ can be written as:

$$\mathcal{P}[\boldsymbol{x}(p, k)|\theta(p, k)] = ((2\pi)^m |\boldsymbol{\Sigma}_{p,k}|)^{-1/2} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_{p,k})^T \boldsymbol{\Sigma}_{p,k}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_{p,k})\right) \tag{7}$$

where $\boldsymbol{\mu}_{p,k}$ and $\boldsymbol{\Sigma}_{p,k}$ are respectively the underlying mean vector and covariance matrix of $m$-variate traces associated with $(p, k)$. Thus in the profiling phase, one can compute the maximum likelihood estimation of $\boldsymbol{\mu}_{p,k}$ and $\boldsymbol{\Sigma}_{p,k}$ by:

$$\hat{\boldsymbol{\mu}}_{p,k} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}_i(p, k) \tag{8}$$

$$\hat{\boldsymbol{\Sigma}}_{p,k} = \frac{1}{n-1} \sum_{i=1}^{n} (\boldsymbol{x}_i(p, k) - \hat{\boldsymbol{\mu}}_{p,k})^T (\boldsymbol{x}_i(p, k) - \hat{\boldsymbol{\mu}}_{p,k}) \tag{9}$$

Once the profiling phase is done, the attack phase allows the adversary to monitor an input plaintext $p^*$ and to classify an observed trace $\boldsymbol{x}^*$ which is currently unclear on its correspond secret key $k^*$. If we assume $\mathcal{P}[\hat{\theta}(p^*, k)]$ is uniform, then the attacking step is reduced to choose $\hat{k}$ with maximum likelihood $\mathcal{N}(\boldsymbol{x}^*|\hat{k}; \boldsymbol{\mu}_{p^*,\hat{k}}, \boldsymbol{\Sigma}_{p^*,\hat{k}})$.

**Discussions:** A choice of the proper underlying distribution for leakage signals has also been investigated in other works. The original TA paper suggests multivariate Gaussian distribution and is thus parameterized by its mean and covariance. Meanwhile, the adversary still needs to validate the multivariate normal hypothesis through some statistical tests in the literature, for instance, the kurtosis test [7] (based on the kurtosis estimation) or the Mardia's test [8] (based on skewness and kurtosis measures). However, the normality assumption is too restrictive in most cases and may not be truly necessary for side channel attacks. In the next section we will show how other machine learning approaches tackle this problem without specific distribution assumptions.

Recent works have proposed improved implementations to overcome its statistical difficulties and high computational complexity. One improving direction would be to calculate the logarithm of the multivariate normal distribution, so that the smallest absolute value of log-likelihood indicates the correct secret key [9]. The other improved point is, instead of using a separate covariance matrix per secret key, one can use a pooled covariance matrix for all possible secret keys [10], by which $\boldsymbol{\Sigma}(p) = \boldsymbol{\Sigma}(p, k_1) = \boldsymbol{\Sigma}(p, k_2) = \cdots = \boldsymbol{\Sigma}(p, k_{|K|})$ and it is possible to pool the covariance estimates into a pooled covariance matrix as well:

$$\hat{\boldsymbol{\Sigma}}_{p,k} = \frac{1}{|K|(n-1)} \sum_{k \in K} \sum_{i=1}^{n} (\boldsymbol{x}_i(p, k) - \hat{\boldsymbol{\mu}}_{p,k})^T (\boldsymbol{x}_i(p, k) - \hat{\boldsymbol{\mu}}_{p,k}). \tag{10}$$

The pooled covariance primarily captures noise in leakage signal which shows no correlation with secret keys but is correlated across traces. Eventhough these implementations work well in practice for short traces, TA presents shortcomings in configurations characterized by long traces since parametric Gaussian is ill-posed in very high dimensions. The common solution would be to first apply dimensionality reduction techniques and then analyze in resulting embedded space.

## 2 Machine Learning Based Attacks

When the Gaussian leakage assumption is relaxed, there is no way to compute templates for the leakage. A recent line of works [11, 12, 6, 4] has investigated this distribution-agnostic setting based on machine learning techniques that exploit discriminating criteria to build classifications directly from the raw or dimension reduced dataset.

**Dimension Reduction:** To overcome the curse of high dimensionality in the analysis of leakage traces, dimension reduction is a necessary preprocessing step that speeds up the learning process, reduces the storage space and improves the quality of models. One of the most popular approaches, principle component analysis (PCA) [13] has been used in SCA by Archambeau [14]. PCA first projects leakage trace vectors into a new set of uncorrelated directions, named principle components, and then selects several most variant components which by assumption contains the largest amount of information. Other alternative dimension reduction algorithms can be considered as well, such as minimum redundancy maximum relevance (mRMR) filter [15] and self organizing map (SOM) [16]. All the followed up classification approaches can work on reduced dimensions calculated in this step.

**Multi-Binary-Classification Approach:** Unlike in template attack where we estimate models and search for optimal key $k$ with maximum likelihood criterion, we consider to directly train and run a classifier that accurately predicts the secret key given the input plaintext and leakage signal. Formally, assume that the plaintext $p$ is fixed and uncorrelated with leakage signal $\boldsymbol{x}$, we have $\{(\boldsymbol{x}_i, k_i) : \boldsymbol{x}_i \in \mathcal{X}, k_i \in \{0,1\}^D\}$ as our training set, where $\boldsymbol{x}_i$ and $k_i$ are respectively the input (leakage trace) and output (secret key). Instead of tackling with a single-compound-label problem whose output space is $\{0,1\}^D$ and suffers generalization difficulties [17], we transform it into a multi-binary-classification (MBC) problem where each classifier reads leakage traces and predicts a certain binary bit in the secret key. Now the training set for predicting the $j$-th bit of key now likes like $\mathcal{D}_{(j)} = \{(\boldsymbol{x}_i, k_i'^{(j)}) : \boldsymbol{x}_i \in \mathcal{X}, k_i'^{(j)} \in \{-1, 1\}\}$. Here we use $k'$ to better fit conventional notations in binary classification literature. Fig. 2 outlines the pipeline for the MBC approach.

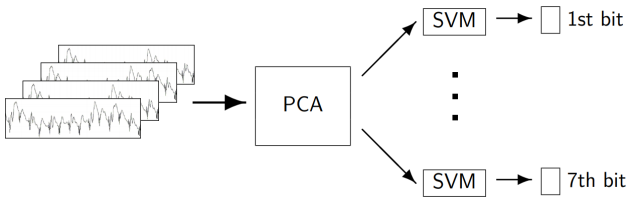


Figure 2: The pipeline for machine learning based side channel attacks [18]. A dimension reduction step will be first applied on the leakage trace vectors. After that, multiple binary classifiers are trained to predict a single bit of the targeted secret key.

**SVM, Random Forests and Deep Neural Networks:** There are a variety of techniques to build a classifier for the binary problem above, such as the standard SVM [19], the least squares SVM (LS-SVM) [20] and the random forest [21]. The classifier in both the standard SVM and LS-SVM takes the form as $k_i'^{(j)} = \text{sign}[w^T\phi(\boldsymbol{x}_i) + b]$ where $\phi(\boldsymbol{x}) : \mathbb{R}^m \longrightarrow \mathbb{R}^{m_\phi}$ is the kernel function that (non-linearly) maps $\boldsymbol{x}_i$ to some implicit higher dimensional spaces. The standard SVM (Eq.11) aims to calculate a separating hyperplane so that distances between closest data points in two classes (support vectors) and the hyperplane get maximized. The LS-SVM (Eq.12) is a variation of SVM, which introduces a least squares loss function and working with equalities instead of solving quadratic programs with inequalities in SVM. Fig. 3 illustrates the decision boundary learned by the LS-SVM after the PCA is applied. Refer [22] for a detailed relationship between SVM and LS-SVM, and refer [2, 3, 23] for the application of SVM and LS-SVM in side channel attacks.

$$\text{SVM:} \quad \min_{w,b,c} \quad \frac{1}{2}w^T w + \frac{\gamma}{2}\sum_{i=1}^{n} c_i \qquad (11)$$
$$s.t. \quad k_i'^{(j)}[w^T\phi(\boldsymbol{x}_i) + b] \geq 1 - c_i, \quad i = 1, \cdots, n$$

$$\text{LS-SVM:} \quad \min_{w,b,c} \quad \frac{1}{2}w^T w + \frac{\gamma}{2}\sum_{i=1}^{n} c_i^2 \qquad (12)$$
$$s.t. \quad k_i'^{(j)}[w^T\phi(\boldsymbol{x}_i) + b] = 1 - c_i, \quad i = 1, \cdots, n$$

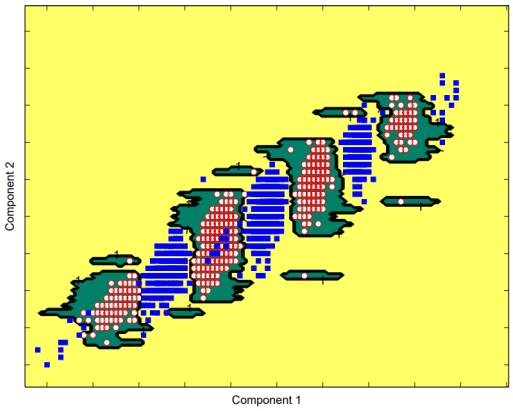


Figure 3: The decision boundary of LS-SVM with an RBF kernel in the embedded space generated by PCA [3]. The horizontal and vertical axes are two principle components from training trace vectors. The red and blue points represent two classes which have respectively 0 and 1 in the $j$-th bit of secrect key.

Another popular classification technique, random forest (RF), is composed of many binary decision trees and the final prediction is computed through a majority vote among the classification results from all trees. In order to reduce the high variance in large decision trees, each decision tree in the random forest is trained on a different subset of training dataset. Fig. 4 simply shows how a single decision tree classifies secret key bits using the leakage signals. Refer to [6, 4] for the successful application in the SCA context.

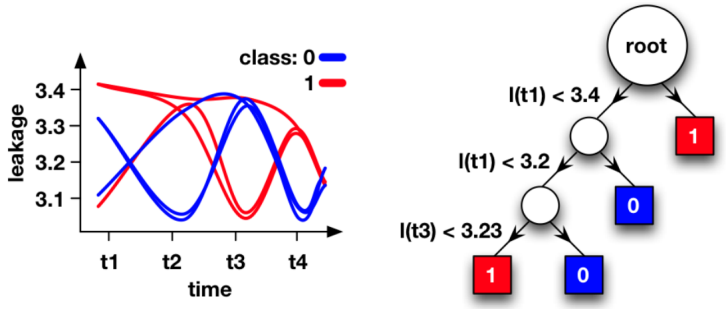


Figure 4: A binary decision tree in a random forest that classifies the secret key bit given leakage signals [6]. The tree will forward the input to one of the possible branch starting from current node until a leaf is reached. A majority voting is performed among decision trees to give a final prediction.

The last class of classifiers that we will mention in this review is deep neural networks [24], which learn more abstract representations from the raw data with stacking many non-linear layers. Under SCA context, deep neural networks differ from usual classifiers in the profiling phase: there is no need to perform either preprocessing with dimension reduction techniques or bit-wise binary classification, since these deep models allow better feature extraction [25]. However, it has been shown that not all architectures of deep neural networks work equally well in SCA tasks [24].

**Power Trace Alignment:** Misalignment is another realistic concern in side channel attack through leakage trace signals, which may result in incorrect leakage models and useless attacks. The reasons

behind the trace misalignment can be the synchronization issues between the clone device and the targeted device, or the clock variability and instability, or some intentional countermeasures such as delays and modulations. One efficient technique for trace alignment is dynamic time warping (DTW) that is essentially a dynamic programming algorithm. As illustrated in Fig 5, DTW minimizes the distances between trace times $t_i$ and $t_j$ of $\boldsymbol{x}$ and $\boldsymbol{y}$ respectively, in a backtracking manner:

$$\mathrm{DTW}_{t_i,t_j} = (\boldsymbol{x}_{t_i} - \boldsymbol{y}_{t_j})^2 + \min\left(\mathrm{DTW}_{t_i,t_j-1}, \mathrm{DTW}_{t_i-1,t_j}, \mathrm{DTW}_{t_i-1,t_j-1}\right) \tag{13}$$

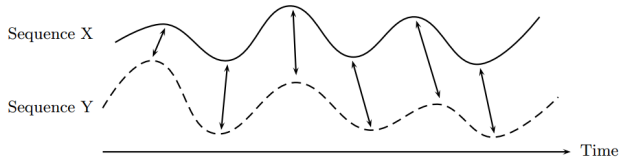


Figure 5: Time alignment of two leakage signals [26]. Dashed arrows indicate aligned points.

## 3 Conclusions

To conclude, both the template attack (TA) and the machine learning based attack (MLA) retrieve meaningful information from vast amounts of leakage signals. These approaches share the same two-stage paradigm: first estimate a leakage model in the profiling phase and then extract the secret key in the attacking phase. While TA relies more on the probability distribution assumptions that need parameters to figure out the underlying templates, MLA is amenable to (nonparametric) supervised learning framework that directly predicts binary bits with dimension reduced traces. Further research works in this area may focus on how to develop attacking approaches against countermeasures and protections, or how to adapt TA and MLA to more complicated scenarios.

## References


[1] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Annual International Cryptology Conference*, pp. 388–397, Springer, 1999.

[2] L. Lerman, G. Bontempi, and O. Markowitch, "Side channel attack: an approach based on machine learning," *Center for Advanced Security Research Darmstadt*, pp. 29–41, 2011.

[3] G. Hospodar, B. Gierlichs, E. De Mulder, I. Verbauwhede, and J. Vandewalle, "Machine learning in side-channel analysis: a first study," *Journal of Cryptographic Engineering*, vol. 1, no. 4, p. 293, 2011.

[4] L. Lerman, G. Bontempi, and O. Markowitch, "A machine learning approach against a masked aes," *Journal of Cryptographic Engineering*, vol. 5, no. 2, pp. 123–139, 2015.

[5] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 13–28, Springer, 2002.

[6] L. Lerman, R. Poussier, G. Bontempi, O. Markowitch, and F.-X. Standaert, "Template attacks vs. machine learning revisited (and the curse of dimensionality in side-channel analysis)," in *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pp. 20–33, Springer, 2015.

[7] K. Nordhausen, H. Oja, D. E. Tyler, *et al.*, "Tools for exploring multivariate data: The package ics," *Journal of Statistical Software*, vol. 28, no. 6, pp. 1–31, 2008.

[8] K. V. Mardia, "Measures of multivariate skewness and kurtosis with applications," *Biometrika*, vol. 57, no. 3, pp. 519–530, 1970.

[9] S. Mangard, E. Oswald, and T. Popp, *Power analysis attacks: Revealing the secrets of smart cards*, vol. 31. Springer Science & Business Media, 2008.

[10] O. Choudary and M. G. Kuhn, "Efficient template attacks," in *International Conference on Smart Card Research and Advanced Applications*, pp. 253–270, Springer, 2013.

[11] T. Bartkewitz and K. Lemke-Rust, "Efficient template attacks based on probabilistic multi-class support vector machines," in *International Conference on Smart Card Research and Advanced Applications*, pp. 263–276, Springer, 2012.

[12] J. Heyszl, A. Ibing, S. Mangard, F. De Santis, and G. Sigl, "Clustering algorithms for non-profiled single-execution attacs on exponentiations," in *International Conference on Smart Card Research and Advanced Applications*, pp. 79–93, Springer, 2013.

[13] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[14] C. Archambeau, E. Peeters, F.-X. Standaert, and J.-J. Quisquater, "Template attacks in principal subspaces," in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 1–14, Springer, 2006.

[15] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[16] T. Kohonen, "The self-organizing map," *Neurocomputing*, vol. 21, no. 1-3, pp. 1–6, 1998.

[17] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *International Journal of Data Warehousing and Mining (IJDWM)*, vol. 3, no. 3, pp. 1–13, 2007.

[18] G. Wolf, "Machine learning approach to side-channel attacks," in *Information Security - Theory vs. Reality*, 2013.

[19] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[20] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural processing letters*, vol. 9, no. 3, pp. 293–300, 1999.

[21] A. Liaw, M. Wiener, *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[22] J. Ye and T. Xiong, "Svm versus least squares svm," in *Artificial Intelligence and Statistics*, pp. 644–651, 2007.

[23] H. He, J. Jaffe, and L. Zou, "Side channel cryptanalysis using machine learning," *CS229 Project*, 2012.

[24] Y. Bengio *et al.*, "Learning deep architectures for ai," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[25] H. Maghrebi, T. Portigliatti, and E. Prouff, "Breaking cryptographic implementations using deep learning techniques," in *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pp. 3–26, Springer, 2016.

[26] M. Müller, "Dynamic time warping," *Information retrieval for music and motion*, pp. 69–84, 2007.