future internet

MDPI

Article

Non-Profiled Unsupervised Horizontal Iterative Attack against Hardware Elliptic Curve Scalar Multiplication Using Machine Learning

Marcin Aftowicz [1], Ievgen Kabin [1], Zoya Dyka [1,2] and Peter Langendörfer [1,2,*]

[1] Leibniz-Institut für Innovative Mikroelektronik—IHP, 15236 Frankfurt (Oder), Germany; aftowicz@ihp-microelectronics.com (M.A.); kabin@ihp-microelectronics.com (I.K.); dyka@ihp-microelectronics.com (Z.D.)
[2] Wireless Systems, Brandenburgische Technische Universität Cottbus-Senftenberg, 03046 Cottbus, Germany
* Correspondence: peter.langendoerfer@b-tu.de

Abstract: While IoT technology makes industries, cities, and homes smarter, it also opens the door to security risks. With the right equipment and physical access to the devices, the attacker can leverage side-channel information, like timing, power consumption, or electromagnetic emanation, to compromise cryptographic operations and extract the secret key. This work presents a side channel analysis of a cryptographic hardware accelerator for the Elliptic Curve Scalar Multiplication operation, implemented in a Field-Programmable Gate Array and as an Application-Specific Integrated Circuit. The presented framework consists of initial key extraction using a state-of-the-art statistical horizontal attack and is followed by regularized Artificial Neural Networks, which take, as input, the partially incorrect key guesses from the horizontal attack and correct them iteratively. The initial correctness of the horizontal attack, measured as the fraction of correctly extracted bits of the secret key, was improved from 75% to 98% by applying the iterative learning.

Keywords: side channel analysis; machine learning; horizontal attack; non-profiled attack; FPGA; ASIC

Citation: Aftowicz, M.; Kabin, I.; Dyka, Z.; Langendörfer, P. Non-Profiled Unsupervised Horizontal Iterative Attack against Hardware Elliptic Curve Scalar Multiplication Using Machine Learning. *Future Internet* **2024**, *16*, 45. https://doi.org/10.3390/fi16020045

Academic Editor: Gianluigi Ferrari

Received: 1 December 2023
Revised: 11 January 2024
Accepted: 17 January 2024
Published: 29 January 2024

1. Introduction

According to IoT Analytics [1], the number of connected IoT devices by 2027 will amount to around 29 billion, and 80% of the market is shaped by three main technologies: WiFi, Bluetooth, and Cellular technology. The fast, wireless communication gives a strong foundation to the development of "smart" industry, cities, homes, health monitoring, autonomous vehicles, or Wireless Sensor Networks (WSN). WSNs are widely used to monitor physical conditions, such as temperature, humidity, or pollution levels, in cities and agriculture, or for predictive maintenance in industry, by measuring vibrations or noise during machine operation. Some of their applications expose the communicating devices to unauthorized physical access. In conjunction with the constraint of very limited resources, the rapid growth of the market, and the ever-growing pressure on low prices and high performance, this exposure can lead to dangerous security negligence. In the past few years, there have been notable disclosures of microarchitectural vulnerabilities even in high-end processors—Spectre [2] and Meltdown [3]—which should only sensitize us towards the subject of hardware security. While those vulnerabilities affected mostly processors used in consumer PCs, the most recently published attack—BUSted [4]—concerned the biggest manufacturer of embedded processors, which constitute the majority of "things" in the IoT world.

Security is an important aspect and concern in the World Wide Web, and the same applies to the Internet of Things. Information security and data integrity can be achieved through the use of cryptography. There are two main approaches in cryptography: symmetric and asymmetric; the last one is also known as public-key cryptography. In symmetric

cryptography, the same key is used to encrypt and decrypt information; therefore, the security depends on keeping the common key secret, as well as on its length. The standardized symmetric approach—the Advanced Encryption Standard (AES) based on the Rijndael block cipher [5]—was published in 2001 [6] by the National Institute of Standards and Technology (NIST) in the USA. In public-key cryptography, the encryption and decryption happen using a pair of keys. The public key can be openly distributed and used to encrypt information, which can only be decrypted by a corresponding private key. Keeping the private key secret is the priority. The first standardized public-key cryptosystem was the Rivest–Shamir–Adleman (RSA) cryptosystem, patented by the cryptographers Ronald Rivest, Adi Shamir, and Leonard Adleman in 1977 [7] and published in 1978 [8]. The next standardized public-key cryptosystem—the Elliptic Curve Cryptosystem (ECC)—was proposed independently by two cryptographers: by Neal Koblitz, in 1985, as a journal paper that was published in 1987 [9], and by Viktor Miller, presented in 1985 at a conference and published in 1986 in the conference proceedings [10].

Modern cryptosystems usually use a hybrid approach and leverage public-key cryptography to establish a secure communication channel and agree on a common symmetric key, i.e., the Elliptic Curve Diffie–Hellman key agreement protocol ECDH (recommended in standard ISO-11770-3) in combination with the AES symmetric key cipher. For example, the Elliptic Curve Integrated Encryption Scheme (ECIES) was proposed in 2001 by Victor Shoup [11] with the goal of accelerating the encryption and decryption significantly. But, public-key cryptography is also commonly used to guarantee information integrity through the usage of Digital Signature Algorithms (DSAs), wherein the (encrypted) information digest is signed by a sender using their private key and the authenticity of the sender and integrity of the information can be tested by anyone using the public key of the sender. One such algorithm is the Elliptic Curve Digital Signature Algorithm (ECDSA) [12]. It was suggested in a response to NIST's (National Institute of Standards and Technology) request for public comments on their proposal for Digital Signature Standards. Later, it was accepted as a standard by the International Standards Organization (ISO-14888-3), American National Standards Institute (ANSI X9.62), and Institute of Electrical and Electronics Engineers (IEEE 1363), and as the U.S. Government Federal Information Processing Standard (FIPS 186) [13]. The main advantage of using Elliptic Curve Cryptography over RSA is its short key size. The same level of security can be achieved using a 256-bit key for ECDSA as can be achieved using a 3072-bit-long key for RSA [14].

The modern cryptosystems are extremely hard to break. It would take more years than the number for which Earth has existed to solve the complex mathematic problems that are the basis of modern cryptography (factorisation of big numbers, discrete logarithm problem (DLP), or elliptic curve discrete logarithm problem (ECDLP)) or to brute-force modern ciphers using the largest super computers available to men. However, there are ways to compromise their security in merely hours. The cryptographic operations can be implemented either as software libraries or as auxiliary hardware components. They require additional processing time, more energy consumption, or larger chip area, which is especially problematic in resource-constrained IoT or WSN devices. The processing of cryptographic primitives is inevitably correlated to the transistor-switching activity of the underlying hardware, whether it comprises a dedicated circuitry or a part of a general-purpose processing pipeline. Dynamic power consumption, the electromagnetic field emanation, heat dissipation, and, above all, the timing information are just some of many side channels, which can be leveraged to uncover the secret. The first one to present how this could be accomplished was Paul Kocher, in 1996 [15], who showed how to break the RSA and Data Encryption Standard (DES) algorithms using only timing and power consumption information. This was considered the beginning of Side Channel Analysis (SCA), a branch of cryptoanalysis focusing on finding new ways to leverage side channels to uncover secrets in cryptosystems, but also on finding new methods to mitigate potential attacks and implement countermeasures.

Since 2012, when the famous AlexNet [16] Convolutional Neural Network (CNN) won the ImageNet competition, machine learning has gained popularity also among the SCA community. The supervised learning paradigm of feeding the neural networks with large amounts of data and corresponding labels fits perfectly into the profiling scenarios of SCA, where electromagnetic traces (EMTs) or power traces (PTs) of multiple cryptographic operations are recorded and the corresponding secret values are provided as labels. While the profiling attacks make sense in SCA attacks against AES, in public-key cryptography, the secret key is ephemeral and only a single recording of a cryptographic operation with given inputs is possible. Especially dangerous are attacks against signature generation if the secret—not the private key of the sender but a randomly generated big number—can be revealed, analyzing only a single execution trace of the cryptographic operation. Knowledge of the secret random number applied for the signature generation allows one to calculate, easily, the private key of the sender. Our work is an extension of a conference paper [17], published in the 26th Euromicro Conference Series on Digital System Design (DSD), proposing a semi-supervised non-profiled side channel analysis attack against an Elliptic Curve Scalar Multiplication (ECSM) hardware implementation. The analysis combines a horizontal attack to derive initial key candidates and a highly regularized Support Vector Machine (SVM) classification step to iteratively correct those candidates. We have also discovered another work [18] discussing a similar framework, which we describe in the related work section, outlining the differences with our approach. Our contributions in this work are as follows:

1. We present a summary of existing horizontal non-profiled solutions against ECSM implementations.
2. We show that even simple Artificial Neural Networks can be used to increase the correctness of horizontal attacks, and we analyze the influence of the regularization of those networks on the attack correctness.
3. We sensitize our readers to the importance of precisely citing the software versions used for neural network training and the hyperparameters used. Using the example of our own analysis, we show that the regularization is implemented differently in two popular Python packages: scikit-learn and tensorflow.

Finally, we show how machine-learning-aided Side Channel Analysis is sensitive to hyperparameters and that finding the optimal set of parameters is a matter of trial and error.

The rest of this paper is structured in the following way. Section 2 describes the context of our work and the brief history and development of the field of SCA attacks, with a focus on horizontal single-trace non-profiled attacks against ECSM implementations. In Section 3, we introduce our hardware accelerator and the two versions of its implementation: an FPGA implementation utilizing the sequencing of addresses on the bus and an ASIC implementation compiled with the compile_ultra option during logical synthesis. Section 4 explains how we have acquired the power and electromagnetic traces for our design. Later, we describe two machine learning classification methods: the Support Vector Machine in Section 5 and the Artificial Neural Network in Section 6. Next, in Section 7, we describe our iterative framework, which consists of trace preprocessing, a statistical horizontal attack, and machine learning classification using labels obtained during the horizontal attack. In Section 8, we show the results of the attack using the power and electromagnetic traces of both design implementations, evaluate the different regularization parameters, and compare the artificial neural networks' implementations using the scikit-learn framework and the tensorflow, providing details about the hyperparameters used. We conclude our work with Section 9, where we summarize our results and present possible improvements and additional analysis steps that could increase the correctness of the attack even further.

2. Related Work

There has been a great interest in using machine learning for side channel analysis in the last decade. The growing number of conference papers and journal articles have

required systemization, which has been reflected by multiple reviews and state-of-the-art articles published throughout the years.

The first to comprehensively study applying ML techniques in the context of SCA were Hospodar, Gierlichs, and colleagues [19]. In their article, they compared the accuracy of the LS-SVM algorithm with a Template Attack (TA) using the power trace of an AES software implementation. They examined different parameter sets, numbers of traces used, and preprocessing methods and concluded that properly tuned ML algorithms outperformed the TA. In 2014, Jap and Breier [20] prepared an overview of ML-based SCA attacks on various cryptosystems including one ECC [21] and one RSA [22]. Later, in 2016, in a very extensive review of secure architectures with an emphasis on physical attacks by Hodgers, Ragazzoni, and colleagues [23], the authors stated that the usage of ML for SCA is still in the early stages and comparisons of such methods are not purposeful due to differences in the ML algorithms used and target cryptosystem implementations. Nevertheless, the authors cited a handful of papers categorizing them based on the ML algorithm used. One year later, the book "Hardware Security and Trust" was published with a chapter from Papachristodoulou, Batina, and Mentens [24] summarizing the SCA attacks against ECC, including ML methods. In 2018 Benadjila, Prouff, and colleagues [25] presented the first comprehensive study of deep learning (DL) methods used in SCA and introduced an open-source dataset consisting of electromagnetic emanation (EM) traces of a software, masked AES implementation, as a benchmark for future researchers interested in DL. In recent years, the usage of ML for SCA against public-key cryptography has only been mentioned in [26–28]. The only extensive summary can be found in the state-of-the-art chapter of Hettwer's doctoral thesis [29], where he was able to list 12 publications concerning the usage of ML in SCA against ECC and RSA.

As can be observed, the side channel analysis of public-key cryptography implementations has always been less represented in the literature. One of the reasons is the small number of open-source datasets. According to [30], there are 11 commonly used open-source datasets for AES and only four for ECC [18,31,32]. Despite the lack of open-source datasets, there are many proprietary datasets used, wherein the authors only describe the used implementations but do not publish the traces. This prevents comprehensive comparisons but still contributes to the progression of the field. The high accessibility of datasets with AES implementation has always spurred interest among researchers. The following AES-dedicated summaries were published in 2019 [33], 2020 [34], and 2021 [30,35]. The growing number of publications regarding AES implementations has led to situations when researchers even forget to mention what the target cryptosystem is, assuming, by default, some AES implementation. Moreover, conducting SCA attacks against AES allows leveraging multiple power traces associated with the same key while EC-based cryptosystems give much less information to the attacker. The key is ephemeral; it is used only once due to either scalar randomization or nonce randomization between consecutive Elliptic Curve Scalar Multiplications (ECSMs). It is harder to create a profile of a device or use methods that require a larger corpus of data, like DL.

Nevertheless, there are still some works dedicated to SCA attacks against ECC. It is not our purpose to list them all in this article, but we will focus on a particular subgroup of attacks. First and foremost, we are interested in horizontal attacks. They give the attackers only one trace of ECSM to distinguish the key bits (it consists of between 200 and 300 iterations, based on the length of the scalar or the blinding/splitting method used). Furthermore, the attacker should not have access to other traces with known scalar, captured on the same system; hence, single-trace, non-profiled attacks. We were only interested in the usage of ML methods in such a limited scenario. Since the non-profiled SCA attacks provide no key for training, the natural choice comprises the unsupervised learning methods.

In 2010, Batina et al. [36] used Principal Component Analysis (PCA) on the power trace of an FPGA implementation of the Edwards curve with a random-order-execution countermeasure in place. The visualization of the absolute average value of principal

components allowed the authors to distinguish ECSM iterations involving the processing of key bit "zero" from key bit "one". Although it was later shown that this countermeasure can be broken with a simple SPA attack and the difference to the mean attack [37], this was the first use of PCA in the context of non-profiled SCA.

Another example of unsupervised learning was the usage of clustering algorithm k-means by Heyszl et al. [21]. The authors took measurements of current flow and electromagnetic emanation and used them both to divide ECSM iterations into two clusters—"zeroes" and "ones"—in a fully unsupervised manner. The EM traces were measured on different places of the chip. Despite the fact that the implementation was protected using the randomized projective coordinates, the leakage was based on conditional register access, depending on the key bit processed. Later, Specht et al. [38] improved the idea and used PCA as preprocessing step for clustering analysis and reduced the algorithmic complexity of the attack, especially when using multiple parallel side channels, although they experimentally showed that including more side channels did not improve the non-profiled attack. The chosen clustering algorithm was expectation maximization instead of the k-means algorithm. An interesting remark regarding the PCA was that the exploitable leakage was not present in the first few components, but was particularly present in the fourth component. The same component was chosen for visualization by Batina et al. in their already-mentioned work [36]. In [39], Kabin et al. used k-means to cluster a hardware implementation of ECSM and PCA, but not as a preprocessing step; rather, they used it as a distinguisher method. They used PCA on an entire ECSM, treating clock cycles representing a single ECSM iteration as one feature vector. Later, they classified points (iterations) based only on the first principal component. All points with a first principal component smaller than zero were assumed to process different scalar bits than those with a first component higher than zero. They also observed that the leakage was not present in the first principal component, and, therefore, k-means clustering outperformed PCA.

Another advantage of k-means was presented in our own work [40], where we showed that in the case of an unbalanced scalar (where the number of ones is different from the number of zeros), the k-means method provides robust results that are invariant to the imbalance. The comparison was made with a state-of-the-art statistical method, the difference to the mean, which performs well only in a balanced scenario.

All of the described attacks aimed at FPGA implementations of cryptographic operations; however, in 2017, Järvinen et al. [41] attacked a software implementation of an NIST P-256 curve with the ω-NAF method with precomputations on point P. The target platform was an 8-bit microprocessor. This time, the k-means clustering was used not to distinguish ECSM iterations but rather to cluster the power traces of partial multiplications. The assumption behind the attack was that multiplications belonging to the same cluster were performed using the same precomputed values, and hence, the attacker needed to brute-force all combinations of precomputed values or use correlation analysis to limit the search space. With the increasing bit width of the processor and window size of the ω-NAF method, the precomputed values were repeated less often, making the brute-force part of the attack unfeasible, although the authors showed in a simulation that a 16-bit implementation might still be vulnerable.

PCA and the k-means algorithm are common in the world of SCA; they are well understood and provide interpretable results. But, there is a newer group of methods in the SCA toolbox that deliver promising results, albeit with a cost (lack of explainability), namely, the Autoencoders (AENs). AENs are a type of Feed-Forward Artificial Neural Network (ANN), trained in a supervised fashion but with a simple trick. While common ANNs are built from stacked layers of neurons, where, with each consecutive layer, the number of neurons decreases (Figure 1a), in AENs, the layers get smaller until they reach some minimal size, then increase back to the original size of the input layer (Figure 1b). Since the input layer and output layer are of the same size, the network can be trained to simply recreate the input. By doing so, it learns a compressed representation of the dataset in the latent space since all input vectors need to be mapped into the latent space by the

encoder part of the network and recreated back to the original space by the decoder part. After the training has been completed and the condition of satisfying recreation loss has been achieved, the decoder part can be discarded and the encoder can be used to compress input vectors into their latent representation, a non-linear mapping with fewer dimensions. In his Master's thesis, Aljuffri used multiple methods based on AENs to break software implementations of ECSM [42]. All methods follow a similar pattern of deep embedded clustering (DEC), where the AEN, or some variant of it, is trained on ECSM iterations and the latent space representation is used for subsequent clustering (a similar approach was presented later in the Master's thesis of Xu [43]). The author also describes an improved version of the attack wherein the clustering results are incorporated into the loss function used to train the encoder. None of the presented single-trace non-profiled attacks achieved remarkable correctness, but they did show improvement in cases of misaligned traces over a naïve k-means clustering (proposed by Heyszl in [21]).

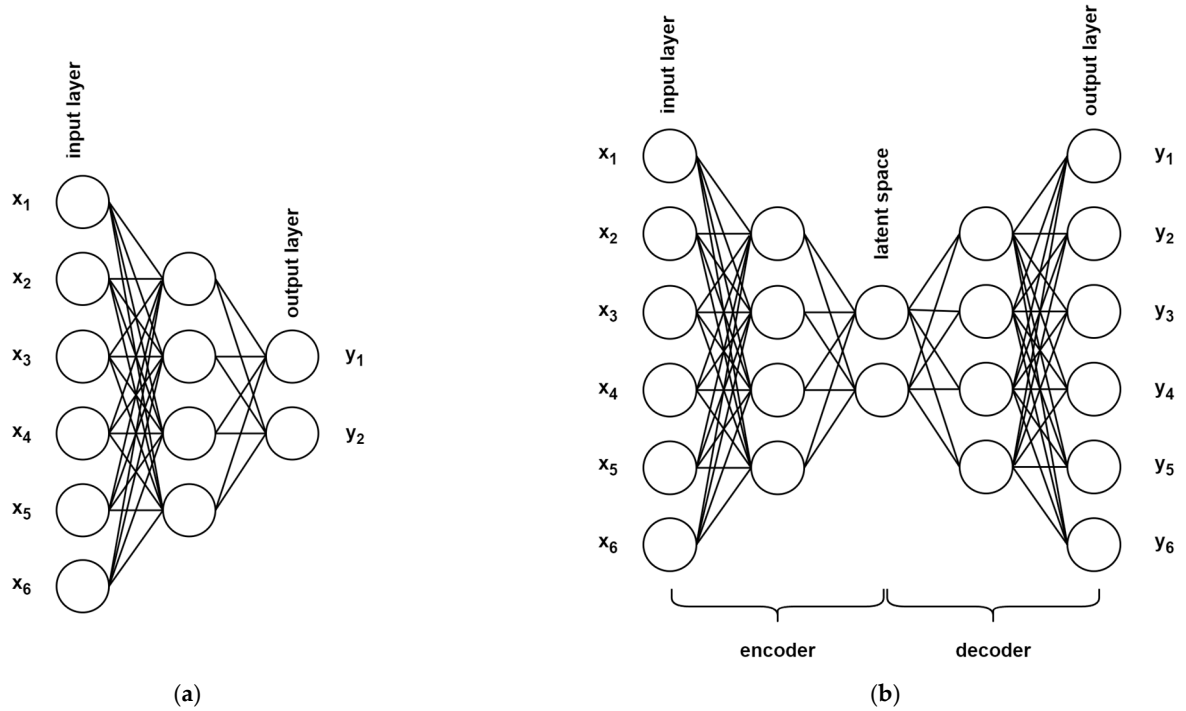


Figure 1. Depiction of (**a**) Feed-Forward Neural Network architecture and (**b**) autoencoder architecture.

The usage of unsupervised learning and, especially, clustering analysis was again leveraged by Nascimento et al. [44] as the first step of processing. They used three methods—the k-means, fuzzy k-means, and expectation maximization algorithms—to analyze a software implementation of ECSM. The leakage assessment framework used in their work was first proposed by Perin et al. in [45], wherein each clock cycle of the ECSM iteration was attacked separately and then combined using majority voting or log-likelihood. In their work, Nascimento et al. used multidimensional (multivariate) clustering, considering all clock cycles at once, to capture higher-order leakage, but noted that this approach is more sensitive to noise. The choice of correct clock cycles is crucial, but since the non-profiling framework does not give the attacker any knowledge about the key, the so-called Points of Interest (POIs) need to be derived in a different way. After the initial clustering they measured intrinsic cluster quality to assess which clusters were dense and far away from each other and which samples did not fit into the discovered patterns. A dimensionality reduction step consists of outlier detection and elimination. The result of clustering analysis is used to recover the first set of key candidates for all analyzed traces. The goal of the next step is to refine the list of POIs to reduce them even further and increase the likelihood of choosing POIs associated with the leakage. The authors ran a *t*-test with two groups of

traces, those where the corresponding bit was assumed to be "one" and those where it was assumed to be "zero". Samples with the highest t-score were chosen as the next round's POIs and the key was recovered based on the refined list. The subject of eliminating POIs not containing leakage was used in context of ECC, in [18], by Perin et al. The authors proposed an unsupervised, non-profiled iterative SCA framework using Cluster Leakage Assessment for initial key candidates' assumption and a Convolutional Neural Network with dropout as a final distinguisher and for the elimination of POIs.

The choice of POIs was also addressed by Ravi et al. in [46]. Some points detected as POIs are not key- but data-dependent and simply add noise to the analysis. The authors assumed that there are usually more than one key-dependent POIs in a power trace and that those points should correlate with each other, while multiple data-dependent points will not. They ran a pairwise covariance computation between all pairs of POIs and ended up with a covariance matrix. It was known that samples close to each other would be correlated; therefore, pairs of samples further away from the covariance diagonal, which had a high magnitude of covariance, were considered as key-dependent; hence, these were chosen as POIs for further analysis. The covariance calculation increased the correctness of multivariate k-means clustering, leading to a more successful attack against an FPGA implementation of ECSM. It did, however, fail in case of an attack against a software implementation, probably due to misalignment.

The choice of POIs can also be driven by the knowledge of the underlying algorithm, and so, in [47], Sim et al. focused on the key identification phase of each ECSM iteration. The identification phase was defined as the time when the key bit was stored in a temporal variable/register. The authors attacked a hardware and software implementation of ECSM, always considering only the beginning of each iteration using the k-means, fuzzy k-means, and expectation maximization algorithms, and recovered the scalar successfully. The ECSM iterations were aligned as part of the preprocessing.

In Table 1, we summarize all related publications that we could have found and which used machine learning in the context of the Side Channel Analysis of Elliptic Curve Scalar Multiplication (or when the authors stated that the analysis could be used for ECSM). All authors assumed no knowledge about the processed key and all of the listed works used machine learning in either the preprocessing step or as a final classifier for ECSM iterations, i.e., to tell which iterations processed a "one" and which processed a "zero". For software (SW) implementations, we also report the software library used. Moreover, we provide details about the platforms running the implementation and the type of captured trace (PT—Power Trace, EMT—Electromagnetic Trace). The last entry is dedicated to our conference publication [17] and this extended work. The abbreviations are listed below Table 1 for readability.

Our work, similar to the non-profiled framework idea presented in [18], consists of applying a horizontal attack and subsequently training a neural network with noisy labels obtained from the attack. The idea can be extended to leverage any unsupervised attack presented in Table 1 to subsequently train a supervised classifier with high regularization to improve the success rate. The main difference between our work and [18] is that we attacked hardware implementations, not software implementations. Moreover, the authors in [18] used their knowledge about the implementation and isolated only subparts of the ECSM iterations with known leakage, while we used the whole trace without discarding any values. In [18], the authors used a Deep Convolutional Neural Network with dropout layers and augmented their dataset to improve the generalization of their network and to prevent overfitting. We have used a simple ANN (without convolutional layers, since we do not need to deal with trace desynchronization) with only one hidden layer of 100 neurons. As a consequence, our network is much simpler; therefore, we decided to use the L1 and L2 regularization in place of dropout. Our work may serve as a validation that the framework proposed in [18] can be successfully applied against hardware implementations of ECSM and that it can be even simplified in such cases. We have also conducted an additional analysis of the influence of the regularization parameter on the success rate of the attack.

Table 1. Summary of related work.

Ref.	Implementation	Platform	Trace Type	Preprocessing	Classifier
[36]	HW: FPGA	Virtex-II Pro	PT [1]	mean of consecutive samples, PCA [3]	visual inspection
[21]	HW: FPGA	Spartan-3	PT, EMT [2]	sum of squared values in each clock cycle	k-means
[38]	HW: FPGA	Spartan 3A	EMT	PCA	EM [4] clustering
[41]	SW: simulation	8-bit AVR microcontroller	PT	averaging over multiple traces (same point means same precomputed values)	k-means
[42]	SW	-	PT	SdAEN [5]	k-means; IDEC [6]
[44]	SW: µNaCl	ARM Cortex-M4	EMT	low-pass filter, outliers replaced with median of non-outliers	k-Means, fuzzy k-means, EM clustering, SOSD [7], SOST [8], MIA [9]
[46]	HW: FPGA SW: µNacl	Virtex-5 ATmega2560	EMT	k-means, DoM [10], covariance, variance, range as POI selection trace alignment (for SW)	thresholding
[47]	HW: FPGA SW: mbedTLS, OpenSSL	SASEBO-GII FPGA AVR XMEGA 128D4	PT, EMT	low-pass filter, POI [11] selection with SOST, trace alignment	k-means, fuzzy k-means, EM, SPA [12]
[39]	HW: FPGA	Spartan 6	PT	sum of squared values in each clock cycle	difference to the mean and k-means
[18]	SW: µNaCl	ARM Cortex-M4	EMT	low-pass filter, Gradient Visualization for POI selection	CLA [13] (k-means, fuzzy k-means, EM clustering) followed by CNN [14]
[17] This	HW: FPGA, ASIC	Spartan 6	PT, EMT	sum of squared values in each clock cycle	difference to the mean followed by SVM [15] followed by ANN [16]

[1] Power Trace, [2] Electromagnetic Trace, [3] Principal Component Analysis, [4] Expectation Maximization, [5] Stacked Denoising Autoencoder, [6] Improved Deep Embedded Clustering, [7] Sum of Squared Differences, [8] Sum of Squared *t*-values, [9] Mutual Information Analysis, [10] Difference of Means, [11] Points of Interest, [12] Simple Power Analysis, [13] Clustering Leakage Assessment, [14] Convolutional Neural Network, [15] Support Vector Machine, [16] Artificial Neural Network.

3. Target Platforms

Our implemented design serves as a hardware accelerator for Elliptic Curve Scalar Multiplication (ECSM), specifically for the NIST Elliptic Curve B-233. The operation it performs is also denoted as kP, where the scalar k is an up-to-233-bit binary number and $P = (x, y)$ is a point on the EC B-233. The coordinates x and y are elements of the extended binary Galois field $GF\ (2^{233})$ with the irreducible polynomial $f(t) = t^{233} + t^{74} + 1$. They are also represented as up-to-233-bit-long binary numbers. The accelerator is based on the Montgomery kP algorithm using Lopez–Dahab projective coordinates [48]. The algorithm processes each bit of the scalar k iteratively from left to right (from its Most Significant Bit (MSB) to the Least Significant Bit (LSB)) using field operations. One ECSM iteration consists of six field multiplications, five field squarings, three field additions, and write-to-register operations and takes only 54 clock cycles. The processing sequence of the calculations, i.e., comprising additions, multiplications, and squaring operations of finite field elements, is independent of the bit value of the processed scalar k, which was the reason in the past to denote Montgomery ladders as sources of resistance against timing attacks and simple power analysis attacks. Please note that the resistance of the Montgomery ladder against simple SCA attacks is based on the important assumption that the addressing of different registers comprises indistinguishable operations regarding their energy consumption [49]. Nowadays, equipment allows one to measure even long power and electromagnetic traces with a high time resolution applying a high sampling rate. This allows one to observe the differences in the measured power shapes caused by the key-dependent addressing of the registers. Dependent on the target platform, both the options used for design synthesis as well as the frequency of the shapes corresponding to the processing of the key bit value '1' can be distinguishable from the shapes corresponding to the processing of the key bit value '0' [50]. The general structure of the design is depicted in Figure 2.

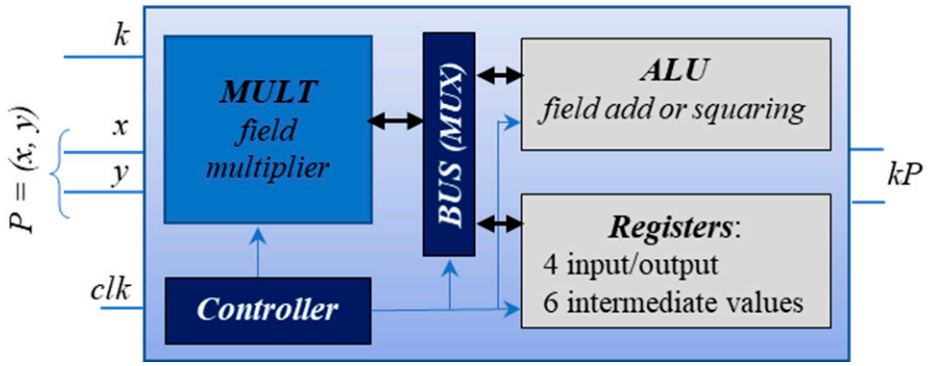


Figure 2. General structure of our hardware accelerator.

The **Controller** block governs the sequence of field operations within the design and the data flow among different blocks and chooses the specific operation to be executed in the ongoing clock cycle. Whenever the **ALU** block is active, it is responsible for carrying out either addition (utilizing bitwise XOR) or squaring operations. Meanwhile, the always-active **MULT** block, functioning as the field multiplier, computes a field product, employing the four-segment Karatsuba multiplication method iteratively. Throughout the years, we have developed different versions of the ECSM accelerator. In this work, we use the following variants of the accelerator:

- Design_ultra is a specific instance of our ECSM design that was compiled with the Synopsys Design Compiler using the compile_ultra feature. This high-effort compilation process involves logic-level and gate-level synthesis and optimization, aiming to achieve an optimum quality of results. The compile_ultra feature leverages all DC Ultra capabilities, balancing timing and area limitations to deliver a compact circuit that meets stringent timing criteria. The exhaustive compilation enhances synthesis quality, resulting in improved performance, reduced design area, and lower power consumption. Design_ultra was manufactured into an ASIC. in the IHP 250 nm technology [51]. The compile options increased the resistance of the design against SCA attacks. The authors applied a non-profiled horizontal statistical attack—the comparison to the mean, which is a univariate attack on each clock cycle of the ECSM iteration, capable of leveraging only the first-order leakage—and achieved 76% correctness for the power trace (PT) and 77% for electromagnetic trace (EMT) for one of the candidates. We use their attack as the first step in our analysis.
- Design_seq is another instance of our ECSM design, and it introduces a countermeasure to the bus-addressing sequence. In clock cycles where no block is specifically addressed by the **Controller** block, Design_seq implements regular addressing instead of some default constant address, ensuring that each of the 11 blocks, including registers, is called in sequence and its contents are written to the bus. This sequential addressing, from 1 to 11 (or 1 to B in hexadecimal), adds a layer of security against attacks like microprobing, especially for the register containing the scalar k, which is never written to the bus. This countermeasure aims to reduce the effectiveness of horizontal SCA attacks on the design by introducing artificial noise in the form of regular addressing when the bus is not in use.

  More details about the ECSM designs can be found in [50].

4. Power and Electromagnetic Trace Acquisition

Design_ultra was produced in the IHP 250 nm technology for a maximum working frequency of 20 MHz (50 ns minimal clock cycle period) and bonded to a printed circuit board (PCB). In our experiments, we used 4 MHz as the working clock frequency. The die area was 2.7 mm^2, with dimensions of 2.49 mm $\times$ 1.09 mm. Design_seq was implemented in VHDL and ported to a Spartan-6 FPGA. The ECSM design is also running at 4 MHz.

During a kP execution, we measured the current through the FPGA and ASIC using a current probe from the security test equipment from Riscure [52] and the electromagnetic emanation using an MFA-R-75 EM probe from Langer [53]. Samples captured using the oscilloscope were saved to a file. We refer to power measurement as power trace, denoted as PT, and electromagnetic measurements as electromagnetic trace, denoted as EMT. The traces were captured using a LeCroy Waverunner 610Zi oscilloscope with a 2.5 GS/s sampling rate, i.e., with 625 measurement points per clock cycle. We used a 232-bit-long random number as the scalar k and the base point G of EC B-233 [13] as point P.

5. SVM Classification

The Support Vector Machines, introduced in 1992 by Bernhard Boser, Isabelle Guyon, and Vladimir Vapnik [54] in the form we know them today, are robust classification models that can leverage different kernels in order to construct non-linear decision boundaries without the need of projecting the training set into a non-linear space. These models try to fit a decision rule to the training set, aiming to maximize the distance between data points and the decision boundary (thick line in Figure 3). The distance is referred to as a margin and data points laying on the margin (dotted line) are called support vectors, hence the name.

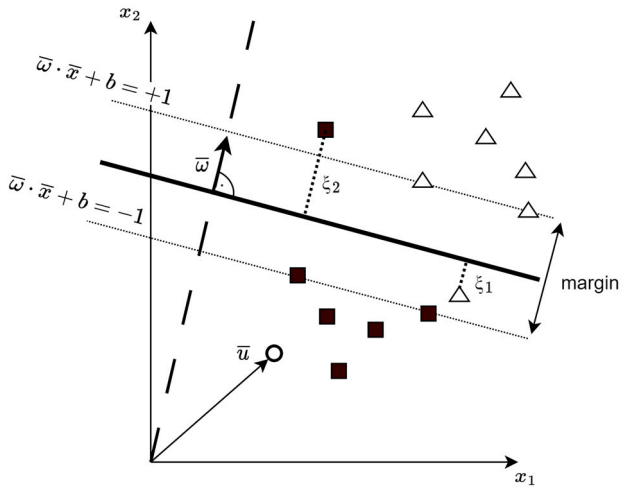


Figure 3. Geometrical representation of optimal hyperplane separating negative examples (black squares) from positive examples (white triangles).

The hyperplane, also called the decision boundary, can be characterized using a normal vector (perpendicular) to that hyperplane, marked as $\overline{\omega}$ in Figure 3. In order to see on which side of the decision boundary an unknown vector $\overline{u}$ resides, a decision function is utilized:

$$\overline{\omega} \cdot \overline{u} + b \geq 0. \tag{1}$$

In hard margin classification, the training consists of maximizing the margin such that the decision rule (1) for positive data points (input vectors) yields values equal to or greater than 1 (they lay outside of the margin and not merely on the right side of the decision boundary) and smaller than or equal to -1 for negative input vectors:

$$y_j(\overline{\omega} \cdot \overline{u} + b) \geq 1, \text{where} y_j = \begin{cases} +1 \; for \; positive \; samples \\ -1 \; otherwise \end{cases}. \tag{2}$$

The margin maximization is equivalent to the minimization of

$$\min_{\omega, b} \frac{1}{2} \|\omega\|^2 \tag{3}$$

with respect to the constraint given in (2).

Hard margin classification is convenient when the data points are easily separable and one wants to avoid misclassifications at all costs. In case of our attack, the initial labels derived from *the comparison to the mean* method are known to be imprecise, and the mislabeled data points would distort the decision boundary and the model might learn wrong classifications. The idea is to introduce a regularization parameter to allow for some data points to be left misclassified, keep the decision boundary smooth, and, therefore, retain a large margin. An SVM model with regularization is called a soft margin classifier and was introduced by Corinna Cortes and Vladimir Vapnik in 1995 [55]. The margin maximization from (3) is modified as follows:

$$\min_{\omega,b,\xi} \frac{1}{2}\|\omega\|^2 + C\sum_{j=1}^{n-2} \xi_j. \tag{4}$$

We denote each data point in the training set with the subscript j since we classify each slot to belong to either the positive or negative class. The new slack variable ξ_j determines the distance from the misclassified sample (slot) to the decision boundary (marked in Figure 3). It is a penalty term for every misclassification and is derived during the training processes. The regularization parameter C takes values between 0 and 1 and needs to be set before the training procedure as a hyperparameter. A value of 1 means no regularization, and values close to 0 mean high regularization and hence a smaller influence of misclassified data points on the decision boundary. The constraint from (2) takes the following form for the soft margin classifier:

$$y_j(\overline{\omega} \cdot \overline{u} + b) \geq 1 - \xi_j \forall j = 1, \ldots, n-2 \; and \; \xi_j > 0 \tag{5}$$

The training algorithm optimizes (4) given the constraint from (5), using every sample from the training set x_j as the unknown input vector $\overline{u}$; in other words, it tries out different hyperplanes separating the training set such that it maximizes the margin but satisfies the constraint that all samples should lay outside of the margin with some tolerance of samples laying on the wrong side of the decision boundary. The SVM training does not explicitly result in $\overline{\omega}$. The decision boundary (the optimal hyperplane) is given as a linear combination of support vectors, i.e., the vectors from the training set. Every ECSM iteration processes either the key bit "zero" or "one"; therefore, all samples correspond to a single ECSM iteration for one input vector during SVM training and classification. The samples are compressed according to (8), which we describe in Section 7—the iterative framework.

6. Artificial Neural Networks

ANNs are built from layers of neurons, like the Neural Network presented in Figure 1. Every single neuron in the network takes a weighted sum of all its inputs and returns, as output, the result of a non-linear activation function. The naming convention comes from biology, where the neuron is activated whenever the incoming signals from its synapsis excite it above a certain threshold. The output of one neuron is connected to all neurons in the next layer, where, again, a linear combination of all inputs is passed through a non-linear activation function.

The process of ANN learning starts with random weight initialization for all neurons in all layers. The input vector is processed by the network and some output value is calculated. The output of the network will usually differ from the expected result (target), and so, the error of the network is calculated using a loss (error) function. As the next step, the error for each neuron is calculated and used by the optimization algorithm to change the weights of the model. This process is repeated for each sample (input vector) in the dataset. When the network has processed all vectors from the dataset, the process is repeated until the error satisfies some condition or a predefined number of repetitions is achieved. Every such iteration (processing of the entire dataset) is called an epoch in neural network training. The training can be therefore seen as the optimization process to find the

minimum of the cost function, which is parametrized by the weights of the model. The weights, on the other hand, influence the decision function realized by the network itself. They change the way the network makes its predictions. If the error in the training set is very small, the network has learned how to exactly map the input vector to the output target value but may fail to generalize well to other inputs not seen during the training. Such a case is called overfitting, and we say that the model has high variance. The other extreme is when the model does not perform well on the training set, due to either not enough complexity or too few epochs in order to minimize the training loss (error). Such a model has high bias and the training results in underfitting. The correct training approach is to find a trade-off between variance and bias, choosing the model complexity that is enough to approximate the target function but not too complex to avoid overfitting, like in Figure 4. We need to include minimizing model complexity into the optimization target of the training function, as well as not create overly complex networks for trivial problems.

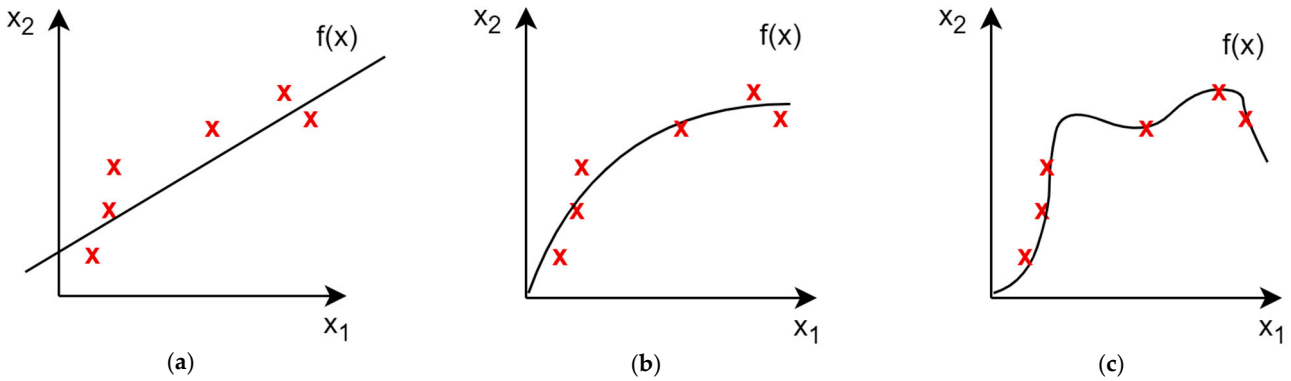


Figure 4. Three cases of model performance: (**a**) underfitting, (**b**) a good bias–variance trade-off, and (**c**) overfitting, given training set samples (red X).

The complexity of the model grows with the number of its trainable parameters (weights). Every non-zero weight increases complexity; therefore, keeping the absolute value of weights small, or even dropping some connections, contributes to fighting overfitting, hence producing simpler decision functions. This can be achieved by using a regularization term during training or adding special dropout layers to the network.

If $\mathcal{L}$ is a loss function, parametrized by the input vector X, output value y, and vector of all weights θ, then the regularized loss function, $\hat{\mathcal{L}}$, will be increased by the Lp norm of θ. The regularization parameter λ specifies how much the norm of the weight vector influences the loss function and needs to be set as a hyperparameter before training:

$$\hat{\mathcal{L}}(\theta, X, y) = \mathcal{L}(\theta, X, y) + \lambda \|\theta\|_p. \tag{6}$$

It is visible that keeping the norm small minimizes the regularized loss function.

7. The Iterative Framework

A power trace (PT) or electromagnetic trace (EMT) acquired during an ECSM operation can be regarded as a sequence of its individual samples. The algorithm's iterative nature, i.e., the processing of the key bit by bit, and its consistent time execution provide the opportunity for the attacker to partition the samples into smaller segments, representing the individual iterations of the ECSM operation, i.e., the handling of a single key bit. In the hardware implementation, the ECSM iterations are equally long (54 clock cycles) and processed one after another without any breaks; therefore, there is no need for a trace alignment step, like in software implementations. We refer to segmented portions, the single ECSM iterations, as "slots". Within each cycle, there are S samples, determined by the oscilloscope's sampling rate *Sampling_Rate* and the clock signal frequency F: $S = \frac{Sampling\_Rate}{F}$. In our experiments, S = 625 samples, applying *Sampling_Rate* = 2.5 Gsample/sec and F = 4 MHz.

Our framework consists of the following:

- A preprocessing step, wherein we compress all values in each clock cycle, i.e., we represent S samples in each clock cycle using only a single value, which is calculated corresponding to the selected compression method; for example, it can be the arithmetic mean of all S samples or the maximum value.
- Then, a horizontal attack is used to derive initial key candidates, which are expected to be somewhat incorrect. We have applied the *comparison to the mean* method used in [50] to have a baseline for our attack, but it could be some other statistical or machine-learning-based attack, which produces initial key candidates.
- At the end, an iterative classification algorithm is used to improve the labeling and hopefully extract the secret scalar. In [17], we used Support Vector Machines; here, we introduce Artificial Neural Networks to the framework. The classifiers are trained based on the initial key candidates derived from the horizontal attack, i.e., one classifier per key candidate, and new candidates are computed. The new candidates are used for subsequent training until the candidates do not change anymore between iterations. We initialize new ANNs after every iteration of the framework although, according to our experiments, there is no difference in the result of the framework whether we reuse already-trained ANNs from the previous iteration or initialize new ones each time. Both the SVM and ANN methods produce single classification for each slot. We used compressed values as inputs, i.e., there are only 54 features, one for each clock cycle of the ECSM iteration. The general idea is shown in Figure 5.

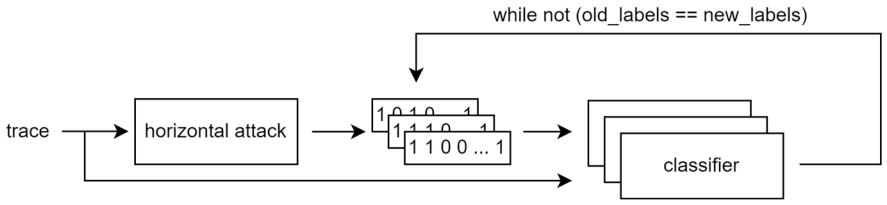


Figure 5. The general structure of the unsupervised framework.

7.1. Preprocessing

All samples belonging to a single clock cycle are compressed using a sum of squared values, according to the following equation:

$$x_i = \sum_{s=1}^{S} {v_{i,s}}^2, \tag{7}$$

Here, v is a sample captured by an oscilloscope, i denotes the clock cycle index in the slot ($1 \leq i \leq 54$), and s is a sample number in the ith clock cycle ($1 \leq s \leq S$). There are $n-2$ slots, where n is the bit length of the key and can maximally amount to 233 in the attacked design.

7.2. Horizontal Attack

We have applied the *comparison to the mean* method used in [50] to have a baseline for our attack. Moreover, we have used the derived key candidates to initialize the semi-supervised learning. The method works as follows.

Firstly, a mean slot is constructed, i.e., an arithmetic mean of the ith clock cycle in all slots is computed, according to

$$\overline{x}_i = \frac{\sum_{j=1}^{n-2} x_{j,i}}{n-2}. \tag{8}$$

Here, j denotes the slot number ($1 \leq j \leq n-2$). For each of the j slots, the value that represents its ith clock cycle, i.e., the value $x_{j,i}$, is compared against the corresponding average value $\overline{x}_i$. If $x_{j,i}$ is higher than or equal to the average, we assumed that this slot

corresponds to the processing of the key bit value 'one'; otherwise, it corresponds to the processing of the key bit value 'zero'. The classification process can be described as below:

$$\hat{k}_{i,j} = 1 \; if \; x_{j,i} \geq \bar{x}_i, \; otherwise \; \hat{k}_{i,j} = 0 \tag{9}$$

This method results in 54 key candidates $\hat{k}_i$, i.e., one per clock cycle, but the attacker also needs to try out the inverse of the key. The disadvantage of this method is that it can only leverage information from one clock cycle and does not pick up the higher-order information present in multiple clock cycles at once.

7.3. Neural Network Iterative Classification

The performed horizontal attack results in 54 key candidates, which we know can contain only first-order leakage, since each of them is calculated based on only one clock cycle. In [17], we used SVM to classify the traces; in this work, we have used Artificial Neural Networks (ANNs) instead. The input to the classification algorithms contains all clock cycles of a single ECSM iteration. We have used the compressed values (8) as the input vector (the uncompressed scenario was computationally much more intensive). Before being fed to the ANN or SVM, the input vector is standardized to zero mean and unit standard deviation.

Since there are 54 key candidates, we have trained 54 completely separate ANNs with the same hyperparameters. The hypothesis is that if the leakage is present in any of the clock cycles, the associated ANNs should find the solution based on the partially correct key candidate from that clock cycle obtained with the horizontal attack. It was shown in [18] and in [17] that regularization helps in combating noisy labels and that this approach is capable of improving the results of the attack.

In our case, the input vector of the ANN, just like for the SVM case, represents one ECSM iteration and consists of compressed values of all clock cycles of that ECSM iteration $x_j = (x_1, x_2, \ldots, x_{54})$, so the input layer has a size of 54. The regularized network predicts one output value for each ECSM iteration. After all input vectors are fed to the classifier, we obtain the new key candidate, which is used to train another ANN with the same hyperparameters. Our training set encompasses the entire trace; we do not hold any testing set on the side. We derive the new key candidate by classifying all ECSM iterations and use all ECSM iterations for subsequent training. The process continues either a fixed number of times or until the key candidates do not change anymore between the rounds.

7.4. Attack Evaluation

As mentioned before, in a non-profiled scenario, we cannot tell whether the resulting key bits have been correctly extracted or not until we compare them with the real key for evaluation. The horizontal attack determines only which scalar bits belong together and not which ones are "zeroes" and which are "ones". Therefore, we always compare the extracted key value and its inverse with the real key to determine the correctness of the attack (similarly, the adversary needs to try out both solutions). If the keys' correctness is 0%, it means that all bits were guessed wrongly and the inverse of the key has 100% correctness. We simplify the evaluation and express the correctness as a relative correctness between 50% and 100%. Our attack is based on a univariate horizontal attack leveraging the first-order leakage present in only a single clock cycle. From the designers' point of view, it is interesting to know which clock cycles contain (significant) SCA leakage; hence, we always report the number of key candidates that extracted the key with a relative correctness greater than 95%. It is also true for key candidates that were subsequently corrected by the ANN since the initial key candidate was based on information from only one clock cycle.

8. Results

We firstly implemented our ANN using the scikit-learn's MLPClassifier from version 1.3.2 [56] with the default parameters: one hidden layer with 100 neurons, the ReLu activation function, and 200 training epochs, with a constant learning rate of 0.001 and an Adam optimizer of cross-entropy loss. The Python version used was 3.9.18. Since we wanted to investigate the influence of regularization on the success rate of the attack, the only parameter we changed was the λ parameter for L2 regularization. Then, we wanted to port this network into tensorflow v2.15.0 [57] to try out L1 regularization as well (it is not available in sci-kit learn), but there seemed to be differences in the implementation of regularization since the network with the exact same hyper parameters yielded different results for the same values of the regularization parameter. The problem seems to be known, but we have not found any answer for its cause. We were forced to repeat the analysis for tensorflow, trying out the same λ parameters for L1 and for L2 regularization, keeping other hyperparameters intact.

8.1. Search for Optimal L1 and L2 Parameters

We tried different regularization parameters and recorded the relative correctness of our attacks. The λ parameter was changed by one order of magnitude at a time. The default setting for scikit-learn is $\lambda = 0.0001$, and for tensorflow, $\lambda = 0.01$. Our intuition, based on [17,18], was that the regularization should be higher than any default value, but because of the discrepancy between the Python packages, we started our investigation for $\lambda = 0.001$ and finished with $\lambda = 10$. Every attack resulted in 54 key candidates, and we reported only the best-performing key candidates' correctness. We evaluated the power trace (PT) and electromagnetic trace (EMT) of Design_seq and Design_ultra. We used scikit-learn and tensorflow implementations while, in scikit-learn, only L2 regularization is available. We also reported the maximal correctness for the horizontal attack using the difference to the mean as "initial" and SVM results from [17] in Table 2. The SVM was not used in this framework but is given for reference. We show the results after only one iteration of ANN training.

The results from Table 2 show the relative correctness described previously at the end of Section 7 (the iterative framework). The correctness tells us how many bits of the secret key were correctly discovered. The higher the correctness is, the more successful the attack was. The correctness is reported for the best-performing key candidate among 54.

We observed that the L2 regularization, which worked for the scikit-learn framework, does not work the same way for tensorflow. The optimal values for every framework and regularization type were different. The scikit-learn-based neural network performs best with $\lambda = 10$, tensorflow L2 regularization with $\lambda = 1$, and L1 regularization with $\lambda = 0.01$. The SVM outperformed all other methods for the first iteration. Similar to what was found in [17], attacks against *Design_ultra* were less successful, showing how synthesis options used during hardware compilation play significant roles in resistance against SCA attacks. The subject was explained in [50].

Table 2. Correctness of the best extracted key candidate after one iteration of the framework. The SVM classification is just given for comparison.

		λ				
Analyzed Trace	**Method**	**0.001**	**0.01**	**0.1**	**1**	**10**
Design_seq (FPGA) EMT	initial			96%		
	SVM C = 0.3			100%		
	scikit-learn L2		96%			**100%**
	tensorflow L2	96%		98%	**100%**	98%
	tensorflow L1	96%	**99%**	52%	57%	52%
Design_Seq (FPGA) PT	initial			88%		
	SVM C = 0.3			99%		
	scikit-learn L2		88%			**94%**
	tensorflow L2	88%		91%	**97%**	69%
	tensorflow L1	**88%**		53%	52%	
Design_ultra (ASIC) EMT	initial			75%		
	SVM C = 0.3			90%		
	scikit-learn L2		75%			**82%**
	tensorflow L2	75%		77%	84%	**86%**
	tensorflow L1	**75%**		55%	56%	58%
Design_ultra (ASIC) PT	initial			74%		
	SVM C = 0.3			89%		
	scikit-learn L2		74%			**81%**
	tensorflow L2	74%		76%	80%	**84%**
	tensorflow L1	74%	**75%**	63%	52%	62%

8.2. Stopping Condition

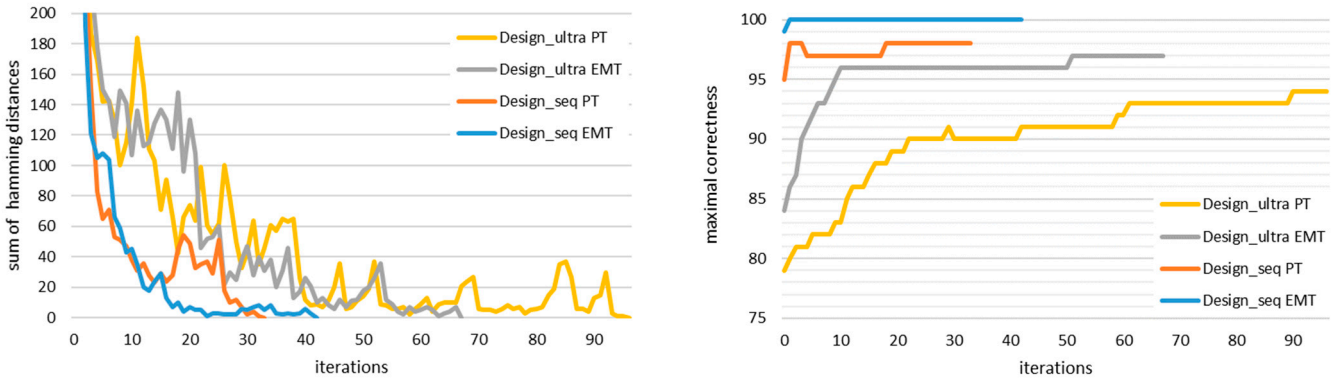
In our second experiment, we decided to observe how the correctness changed throughout the iterations. The predicted key candidates were used in the next iteration as training labels for newly initialized neural networks. Just like in [17], we observed that the longer the framework runs, the fewer bits change in all key candidates between iterations. Therefore, we monitored the hamming distance (HD) between consecutive key candidates to see whether the framework converges to any particular solution. Since there are 54 candidates, each 230 bits long, the maximal cumulative hamming distance is 12,420. If all key candidates do not change between the iterations, the cumulative hamming distance is 0 and the framework stops. We have reported the maximal correctness of the last iteration (with a cumulative hamming distance of 0) for both designs in Table 3. We have also included the SVM results from [17] although the results between the first and last rounds of SVM did not differ. In the first row, we show the initial correctness of the horizontal attack, just like in Table 2, and the correctness in the last iteration. We have also reported how many iterations were needed to achieve no change between consecutive key candidates for all ANNs. Both tables are connected in the following way: we have taken all λ parameters and reported the results after the first iteration in Table 2. We continued the training until the cumulative hamming distance was equal to zero and reported the result of the last iteration in Table 3. Table 2 includes all λ parameters, and we have marked the best parameters with a bold font. Only those best parameters are later shown in Table 3. Since the attack was fully unsupervised, we tried out all λ parameters reported previously in Table 2, but only summarized the best results in Table 3.

Table 3. Attack results after letting the framework run until the cumulative hamming distance amounted to 0.

Analyzed Trace	Method	Max Score	Scores > 95%	Iterations
Design_seq (FPGA) EMT	initial	96%	1	-
	SVM C = 0.3	100%	4	1
	scikit-learn L2 λ = 10	100%	5	43
	tensorflow L2 λ = 1	100%	4	29
	tensorflow L1 λ = 0.01	100%	4	41
Design_Seq (FPGA) PT	initial	88%	-	-
	SVM C = 0.3	99%	1	1
	scikit-learn L2 λ = 10	98%	1	34
	tensorflow L2 λ = 1	98%	1	48
	tensorflow L1 λ = 0.01	90%	-	90
Design_ultra (ASIC) EMT	initial	75%	-	-
	SVM C = 0.3	90%	-	1
	scikit-learn L2 λ = 10	97%	6	68
	tensorflow L2 λ = 1	98%	4	64
	tensorflow L1 λ = 0.01	79%	-	52
Design_ultra (ASIC) PT	initial	74%	-	-
	SVM C = 0.3	89%	-	1
	scikit-learn L2 λ = 10	94%	-	97
	tensorflow L2 λ = 1	92%	-	39
	tensorflow L1 λ = 0.01	74%	-	56

Similar to the results presented in Table 2, the percentages of correctly guessed bits in the best-performing key candidate have been reported. The higher a percentage is, the lower the remaining brute-force complexity needed to uncover the real secret scalar is. Moreover, we have reported the number of key candidates exceeding the 95% relative correctness. Those candidates are connected with clock cycles where leakage is present, i.e., during those clock cycles, some key-dependent operation is executed in hardware. Although the scikit-learn and tensorflow implementations of regularization are different, with the right choice of λ, both frameworks achieved similar correctness, with scikit-learn's small advantage of discovering more key candidates with correctness higher than 95%. It took different numbers of iterations to achieve the stopping condition; therefore, we took a look at how many iterations were actually needed to achieve the same correctness as in the final iteration. We observed how the cumulative hamming distance changes between iterations for each implementation, monitoring the maximal correctness of each iteration at the same time.

In Figure 6, we show the results of the analysis for the scikit-learn implementation reported in Table 3. We measured the cumulative hamming distance between two iterations in the following way: for each of the 54 key candidates, we checked the hamming distance between the initial key candidate, used for training in the nth iteration, and the resulting key candidate, resulting from the classification of the entire power or electromagnetic trace in the nth iteration of the framework. Since we were interested in finding the best stopping condition, we wanted to focus on the last iterations, not the first; therefore, we have limited the vertical axis when showing the cumulative hamming distance.



(**a**) (**b**)

Figure 6. Trend of (**a**) the cumulative hamming distance of all key candidates and (**b**) the correctness of the best key candidate throughout iterations of the framework trained using the ANN implemented in scikit-learn with L2 regularization and $\lambda = 10$.

In general, we can observe that the cumulative hamming distance decreases throughout iterations. This is not true for every pair of consecutive iterations, but a general trend is visible. Moreover, we observed that low regularization parameters (0.0001, 0.001, 0.01) resulted in just a repetition of the key candidates obtained during the horizontal attack. The ANNs were simply trained to repeat the candidates, trusting the labelling completely. If the labels do not change in the results of ANN training, it might be a good indicator that the regularization parameter is too low. Unfortunately, we do not have any similar recommendation for the upper bound of λ. We have observed that independent from the correctness of the results, λ values higher than 0.01 give a stable downward trend of the cumulative hamming distances and that converging to the cumulative hamming distance of 0 is no indicator of attacks' successfulness, i.e., although the key candidates do not change anymore, they are not necessarily correct. The correctness of attacks did gradually improve throughout iterations, but only in the case of the optimal regularization parameters. In any other case, even high initial correctness ended up in random key guesses.

We also tried stopping the training of individual ANNs when only their respective key candidates did not change between iterations, but this led to worse results; therefore, even the global condition of achieving the cumulative hamming distance of zero might not guarantee the highest possible correctness. It is possible that training beyond this level, with appropriate regularization, will give even better results.

8.3. Number and Position of Best Key Candidates

Although ANNs leverage multivariate leakage information from all clock cycles, every key candidate resulting from the horizontal attack is based on a single clock cycle only. Those clock cycles can be considered leaking information and are the first to be investigated by the cryptographic implementation designer. Moreover, we investigated the key candidates based on which our framework provided the best correctness, to see whether the good correctness of the final result correlated with the good correctness of the initial horizontal attack. We have shown, based on the scikit-learn implementation, the initial and last round's correctness for every clock cycle of a single ECSM iteration in Figure 7.

We can observe that the initial high correctness of the key candidate correlates with the high correctness of the final result and, hence, the ANN increases the correctness via subsequent training iterations. Please note that the performed attack results into a slightly higher correctness of the best key candidates in comparison to the analysis of the same, but uncompressed traces of Design_ultra, applying Fourier transformation [58], i.e., the performed attack currently has the highest success rate against the optimized design.

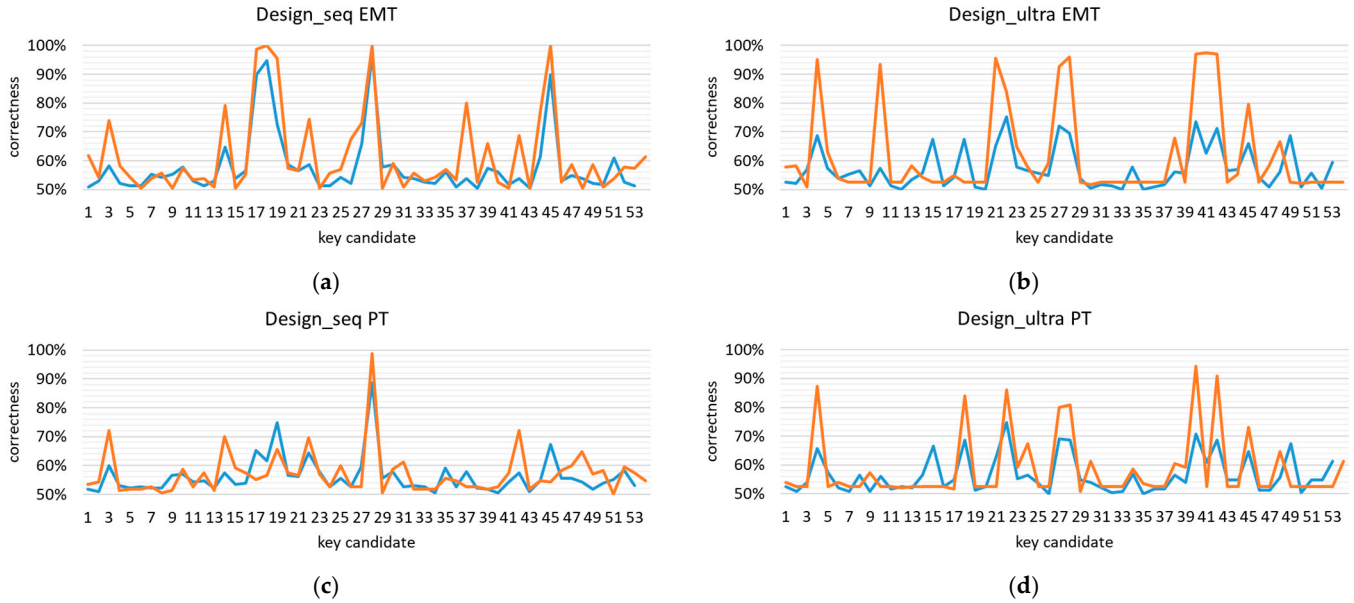


Figure 7. The correctness of each key candidate as an indication of leakage in a given clock cycle, for initial horizontal attack (blue) and last iteration of the scikit-learn implementation (orange) for (**a**) EMT of Design_seq, (**b**) EMT of Design_ultra, (**c**) PT of Design_seq, and (**d**) PT of Design_ultra.

9. Conclusions

Similar to the framework used by Perin et al. in [18], we have shown that it is possible to correct the initial key candidates resulting from a horizontal attack by training a highly regularized machine learning classifier. We have proven that the right choice of the regularization parameter can lead to breaking both analyzed Elliptic Curve Scalar Multiplication hardware accelerators with a high correctness. In our experiments, we increased the results of the initial horizontal attack from 75% up to 98% using an ANN and achieved better results than in our previous implementation using the SVM [17]. In this work, we have also reached higher correctness than the authors of [58], attacking the same traces of the design compiled with the compile_ultra setting. We have also shown that even simple neural networks can be utilized without the need of leveraging deep learning architectures although it must be noted that our implementations are very regular and no trace synchronization is necessary. The proposed framework is fully unsupervised, in a way that it does not require any knowledge about the cryptographic key, but it is very sensitive to the choice of the regularization parameters, and possibly to other hyperparameters of the neural network as well. Even more confusing is the fact that both open-source implementations of neural networks, scikit-learn and tensorflow, implement regularization in different ways and even using the same hyperparameters renders differences in the attack correctness and general attack performance. They also propose different values for default parameters (scikit-learn sets $\lambda = 0.0001$, and tensorflow, $\lambda = 0.01$).

We have proposed a stopping condition for the iterative framework to be the cumulative hamming distance between two consecutive key candidates, equal to zero (a sum of the hamming distances of all candidates between two iterations). It is worth noting that in our experiments, stopping any of the key candidates before the entire framework reached the stopping condition, decreased the results of the attack. Therefore, waiting until all candidates reach this point seems reasonable.

Our attack required trying out different magnitudes of the L2 regularization parameters, and if applied for different designs, it would probably require trying out different hyperparameters. We provided an intuition on determining the lower band for the regularization parameter. The regularization is not strong enough if the stopping condition is met already after the first iteration, i.e., all key candidates simply do not change.

Our framework could benefit from an analysis of which extracted key candidates have the highest probability of being correct to reduce the remaining brute-force complexity and leverage information from multiple side-channels during a single attack like in [38,59].

Author Contributions: Conceptualization, framework implementation, model training, optimal parameter search, writing—original draft preparation, M.A.; cryptographic design implementation, traces acquisition, statistical analysis, I.K.; writing—review and editing, Z.D., I.K. and P.L.; supervision and project administration Z.D. and P.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to concerns associated with the publication of sensitive information to comply with German export laws governing cryptographic hardware accelerators. The authors are highly motivated to publish the acquired power and electromagnetic traces, knowing their importance for the Side Channel Analysis community; yet, this precautionary measure is essential to prevent any potential risks or vulnerabilities that could arise from unauthorized access to the disclosed information, ensuring compliance with legal and security protocols.

Conflicts of Interest: Marcin Aftowicz, Ievgen Kabin and Zoya Dyka are employee of IHP—Leibniz-Institut für innovative Mikroelektronik. The authors declare no conflicts of interest.

References

1. Sinha, S. State of IoT 2023: Number of Connected IoT Devices Growing 16% to 16.7 Billion Globally. IoT Analytics GmbH, 24 May 2023. Available online: https://iot-analytics.com/number-connected-iot-devices/ (accessed on 3 November 2023).
2. Kocher, P.; Horn, J.; Fogh, A.; Genkin, D.; Gruss, D.; Haas, W.; Hamburg, M.; Lipp, M.; Mangard, S.; Prescher, T.; et al. Spectre Attacks: Exploiting Speculative Execution. In Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 19–23 May 2019; pp. 1–19.
3. Lipp, M.; Schwaz, M.; Gruss, D.; Prescher, T.; Haas, W.; Mangard, S.; Kocher, P.; Genkin, D.; Yarom, Y.; Hamburg, M.; et al. Meltdown. Available online: https://arxiv.org/pdf/1801.01207.pdf (accessed on 30 November 2023).
4. Pinto, S.; Rodrigues, C. Hand Me Your SECRET, MCU!: Microarchitectural Timing Attacks on Microcontrollers Are Practical. Presented at the Black Hat Asia, Singapore, 9–12 May 2023; Available online: https://www.youtube.com/watch?v=xso4e4BdzFo (accessed on 30 November 2023).
5. Daemen, J.; Rijmen, V. *The Design of Rijndael: AES—The Advanced Encryption Standard*; Springer: Berlin, Germany; London, UK, 2011.
6. Advanced Encryption Standard (AES), FIPS 197. Available online: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197-upd1.pdf (accessed on 30 November 2023).
7. Rivest, R.L.; Shamir, A.; Adleman, L.M. Cryptographic Communications System and Method. U.S. Patent 4,405,829, 20 September 1983.
8. Rivest, R.L.; Shamir, A.; Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* **1978**, *21*, 120–126. [CrossRef]
9. Koblitz, N. Elliptic curve cryptosystems. *Math. Comput.* **1987**, *48*, 203–209. [CrossRef]
10. Miller, V.S. Use of Elliptic Curves in Cryptography. In Proceedings of the Conference on the Theory and Application of Cryptographic Techniques, Santa Barbara, CA, USA, 18–22 August 1985; Springer: Berlin/Heidelberg, Germany, 2000; Volume 218, pp. 417–426. [CrossRef]
11. Shoup, V. A Proposal for an ISO Standard for Public Key Encryption. Available online: https://eprint.iacr.org/2001/112 (accessed on 30 November 2023).
12. Johnson, D.; Menezes, A.; Vanstone, S. The Elliptic Curve Digital Signature Algorithm (ECDSA). *Int. J. Inf. Secur.* **2001**, *1*, 36–63. [CrossRef]
13. Digital Signature Standard (DSS), NIST FIPS 186-4. Available online: https://csrc.nist.gov/pubs/fips/186-4/final (accessed on 30 November 2023).
14. Barker, E. *NIST Special Publication 800-57 Part 1 Revision 5: Recommendation for Key Management*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2020. [CrossRef]
15. Kocher, P.C. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Advances in Cryptology—CRYPTO'96: 16th Annual International Cryptology Conference, Santa Barbara, CA, USA, 18–22 August 1996*; Koblitz, N., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1996; pp. 104–113. [CrossRef]
16. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. In Proceedings of the 26th Annual Conference on Advances in Neural Information Processing Systems NeurIPS'12, 2012; Available online: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf (accessed on 30 November 2023).

17. Aftowicz, M.; Kabin, I.; Dyka, Z.; Langendoerfer, P. Non-profiled semi-supervised horizontal attack against Elliptic Curve Scalar Multiplication using Support Vector Machines. In Proceedings of the 26th Euromicro Conference Series on Digital System Design (DSD), Durres, Albania, 6–8 September 2023.
18. Perin, G.; Chmielewski, L.; Batina, L.; Picek, S. Keep it Unsupervised: Horizontal Attacks Meet Deep Learning. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2020**, *2021*, 343–372. [CrossRef]
19. Hospodar, G.; Gierlichs, B.; De Mulder, E.; Verbauwhede, I.; Vandewalle, J. Machine learning in side-channel analysis: A first study. *J. Cryptogr. Eng.* **2011**, *1*, 293–302. [CrossRef]
20. Jap, D.; Breier, J. Overview of machine learning based side-channel analysis methods. In Proceedings of the 2014 International Symposium on Integrated Circuits, Singapore, 10–12 December 2014; pp. 38–41.
21. Heyszl, J.; Ibing, A.; Mangard, S.; de Santis, F.; Sigl, G. Clustering Algorithms for Non-profiled Single-Execution Attacks on Exponentiations. In *Sublibrary: SL 4, Security and Cryptology, Proceedings of the Smart Card Research and Advanced Applications: 12th International Conference, CARDIS 2013, Berlin, Germany, 27–29 November 2013*; Revised Selected Papers; Francillon, A., Rohatgi, P., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2014; Volume 8419, pp. 79–93. [CrossRef]
22. Perin, G.; Imbert, L.; Torres, L.; Maurine, P. Attacking Randomized Exponentiations Using Unsupervised Learning. In Proceedings of the Constructive Side-Channel Analysis and Secure Design: 5th International Workshop, COSADE 2014, Paris, France, 13–15 April 2014; Revised Selected Papers. Prouff, E., Ed.; Springer: Cham, Switzerland, 2014; Volume 8622, pp. 144–160. [CrossRef]
23. Hodgers, P.; Regazzoni, F.; Gilmore, R.; Moore, C.; Oder, T. State-of-the-Art in Physical Side-Channel Attacks and Resistant Technologies. Secure Architectures of Future Emerging Cryptography (SAFEcrypto) D7.1. February 2016. Available online: https://ec.europa.eu/research/participants/documents/downloadPublic?documentIds=080166e5a63fd691&appId=PPGMS (accessed on 30 November 2023).
24. Papachristodoulou, L.; Batina, L.; Mentens, N. Recent Developments in Side-Channel Analysis on Elliptic Curve Cryptography Implementations. In *Hardware Security and Trust: Design and Deployment of Integrated Circuits in a Threatened Environment*; Sklavos, N., Chaves, R., Di Natale, G., Regazzoni, F., Eds.; Springer International Publishing: Cham, Swtizerland, 2017; pp. 49–76. [CrossRef]
25. Benadjila, R.; Prouff, E.; Strullu, R.; Cagli, E.; Canovas, C. Study of Deep Learning Techniques for Side-Channel Analysis and Introduction to ASCAD Database. *J. Cryptogr. Eng.* **2018**, *2018*, 53.
26. Taouil, M.; Aljuffri, A.; Hamdioui, S. Power Side Channel Attacks: Where Are We Standing? In Proceedings of the 2021 16th International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS), Montpellier, France, 28–30 June 2021; pp. 1–6.
27. Jovic, A.; Jap, D.; Papachristodoulou, L.; Heuser, A. Traditional Machine Learning Methods for Side-Channel Analysis. In *Security and Artificial Intelligence: A Crossdisciplinary Approach*; Batina, L., Bäck, T., Buhan, I., Picek, S., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Swtizerland, 2022; Volume 13049, pp. 25–47. [CrossRef]
28. Krček, M.; Li, H.; Paguada, S.; Rioja, U.; Wu, L.; Perin, G.; Chmielewski, Ł. Deep Learning on Side-Channel Analysis. In *Security and Artificial Intelligence: A Crossdisciplinary Approach*; Batina, L., Bäck, T., Buhan, I., Picek, S., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2022; Volume 13049, pp. 48–71. [CrossRef]
29. Hettwer, B. Deep Learning-Enhanced Side-Channel Analysis of Cryptographic Implementations. Ph.D. Thesis, Ruhr-Universität Bochum, Bochum, Germany, 2021. [CrossRef]
30. Picek, S.; Perin, G.; Mariot, L.; Wu, L.; Batina, L. SoK: Deep Learning-based Physical Side-channel Analysis. *ACM Comput. Surv.* **2023**, *55*, 1–35. [CrossRef]
31. Weissbart, L. 25519 WolfSSL. Available online: https://github.com/leoweissbart/MachineLearningBasedSideChannelAttackonEdDSA (accessed on 30 November 2023).
32. Chmielewski, Ł. REASSURE (H2020 731591) ECC Dataset. Zenodo. 16 January 2020. Available online: https://zenodo.org/records/3609789 (accessed on 30 November 2023).
33. Masure, L.; Dumas, C.; Prouff, E. A Comprehensive Study of Deep Learning for Side-Channel Analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2019**, *2020*, 348–375. [CrossRef]
34. Jin, S.; Kim, S.; Kim, H.; Hong, S. Recent advances in deep learning-based side-channel analysis. *ETRI J.* **2020**, *42*, 292–304. [CrossRef]
35. Kaur, J.; Lamba, S.; Saini, P. Advanced Encryption Standard: Attacks and Current Research Trends. In Proceedings of the 2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), Greater Noida, India, 4–5 March 2021; pp. 112–116.
36. Batina, L.; Hogenboom, J.; Mentens, N.; Moelans, J.; Vliegen, J. Side-channel evaluation of FPGA implementations of binary Edwards curves. In Proceedings of the 2010 17th IEEE International Conference on Electronics, Circuits and Systems—(ICECS 2010), Athens, Greece, 12–15 December 2010; pp. 1248–1251.
37. Kabin, I.; Dyka, Z.; Klann, D.; Mentens, N.; Batina, L.; Langendoerfer, P. Breaking a fully Balanced ASIC Coprocessor Implementing Complete Addition Formulas on Weierstrass Elliptic Curves. In Proceedings of the 2020 23rd Euromicro Conference on Digital System Design (DSD), Kranj, Slovenia, 26–28 August 2020; pp. 270–276. [CrossRef]
38. Specht, R.; Heyszl, J.; Kleinsteuber, M.; Sigl, G. Improving Non-profiled Attacks on Exponentiations Based on Clustering and Extracting Leakage from Multi-channel High-Resolution EM Measurements. In *Sublibrary: SL 4, Security and Cryptology,*

Proceedings of the 6th International Workshop on Constructive Side-Channel Analysis and Secure Design, Berlin, Germany, 13–14 April 2015; Revised selected papers; Mangard, S., Poschmann, A.Y., Eds.; Lecture Notes in Computer Science; Springer: Cham, Swtizerland, 2015; Volume 9064, pp. 3–19. [CrossRef]
39. Kabin, I.; Aftowicz, M.; Varabei, Y.; Klann, D.; Dyka, Z.; Langendoerfer, P. Horizontal Attacks using K-Means: Comparison with Traditional Analysis Methods. In Proceedings of the 2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS), Canary Islands, Spain, 24–26 June 2019; pp. 1–7.
40. Aftowicz, M.; Kabin, I.; Dyka, Z.; Langendoerfer, P. Clustering versus Statistical Analysis for SCA: When Machine Learning is Better. In Proceedings of the 2021 10th Mediterranean Conference on Embedded Computing (MECO), Budva, Montenegro, 7–10 June 2021; pp. 1–5.
41. Järvinen, K.; Balasch, J. Single-Trace Side-Channel Attacks on Scalar Multiplications with Precomputations. In Proceedings of the Smart Card Research and Advanced Applications: 15th International Conference, CARDIS 2016, Cannes, France, 7–9 November 2016; Revised Selected Papers. Lemke-Rust, K., Tunstall, M., Eds.; Springer: Cham, Switzerland, 2017; Volume 10146, pp. 137–155. [CrossRef]
42. Aljuffri, A. Exploring Deep Learning for Hardware Attacks. Master's Thesis, Delft University of Technology, Delft, The Netherlands, 2018. Available online: http://resolver.tudelft.nl/uuid:c0dddd21-bdc1-4641-bd5d-4abdbd7fe35f (accessed on 30 November 2023).
43. Xu, T. A Novel Simple Power Analysis (SPA) Attack against Elliptic Curve Cryptography (ECC). Ph.D. Thesis, Northeastern University, Boston, MA, USA, 2021. [CrossRef]
44. Nascimento, E.; Chmielewski, Ł. Applying Horizontal Clustering Side-Channel Attacks on Embedded ECC Implementations. In *Sublibrary: SL 4, Security and Cryptology, Proceedings of the International Conference on Smart Card Research and Advanced Applications, Lugano, Switzerland, 13–15 November*; Revised selected papers; Eisenbarth, T., Teglia, Y., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2018; Volume 10728, pp. 213–231. [CrossRef]
45. Perin, G.; Chmielewski, Ł. A Semi-Parametric Approach for Side-Channel Attacks on Protected RSA Implementations. In *Sublibrary: SL 4, Security and Cryptology, Proceedings of the International Conference on Smart Card Research and Advanced Applications, Bochum, Germany, 4–6 November 2015*; Revised Selected Papers; Homma, N., Medwed, M., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2016; Volume 9514, pp. 34–53. [CrossRef]
46. Ravi, P.; Jungk, B.; Jap, D.; Najm, Z.; Bhasin, S. Feature Selection Methods for Non-Profiled Side-Channel Attacks on ECC. In Proceedings of the 2018 IEEE 23rd International Conference on Digital Signal Processing (DSP), Shanghai, China, 19–21 November 2018; pp. 1–5.
47. Sim, B.-Y.; Kang, J.; Han, D.-G. Key Bit-Dependent Side-Channel Attacks on Protected Binary Scalar Multiplication. *Appl. Sci.* **2018**, *8*, 2168. [CrossRef]
48. López, J.; Dahab, R. Fast Multiplication on Elliptic Curves Over GF(2m) without precomputation. In Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems, Worcester, MA, USA, 12–13 August 1999; Koç, Ç.K., Paar, C., Eds.; Springer: Berlin/Heidelberg, Germany, 1999; Volume 1717, pp. 316–327. [CrossRef]
49. Joye, M.; Yen, S.-M. The Montgomery Powering Ladder. In Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems, Redwood Shores, CA, USA, 13–15 August 2002; Kaliski, B.S., Koç, C.K., Paar, C., Eds.; Springer: Berlin/Heidelberg, Germany, 2003; pp. 291–302. [CrossRef]
50. Kabin, I. Horizontal Address-Bit SCA Attacks against ECC and Appropriate Countermeasures. Ph.D. Thesis, BTU Cottbus—Senftenberg, Cottbus, Germany, 2023. [CrossRef]
51. IHP-Solutions: Foundry Service, SiGe BiCMOS Technology. Available online: https://www.ihp-solutions.com/services (accessed on 8 January 2024).
52. Riscure, Driving Your Security forward—Riscure. Available online: https://www.riscure.com/ (accessed on 8 January 2024).
53. Langer EMV-Technik GmbH. Available online: https://www.langer-emv.de/de/index (accessed on 8 January 2024).
54. Boser, B.E.; Guyon, I.M.; Vapnik, V.N. A training algorithm for optimal margin classifiers. In Proceedings of the Fifth Annual Workshop on Computational Learning Theory, Pittsburgh, PA, USA, 27–29 July 1992; pp. 144–152. [CrossRef]
55. Cortes, C.; Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
56. Pedregosa, F.; Varoquaus, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
57. Google. TensorFlow. Available online: https://www.tensorflow.org (accessed on 22 February 2023).

58. Kabin, I.; Dyka, Z.; Klann, D.; Aftowicz, M.; Langendoerfer, P. FFT based Horizontal SCA Attack against ECC. In Proceedings of the 2021 11th IFIP International Conference on New Technologies, Mobility and Security (NTMS), Paris, France, 19–21 April 2021; pp. 1–5.
59. Genevey-Metat, C.; Gérard, B.; Heuser, A. Combining sources of side-channel information. In Proceedings of the Cybersecurity Conferences Series C&ESAR'19, Rennes, France, 2019; Available online: https://hal.science/hal-02456646v1/document (accessed on 30 November 2023).