Journal of
Sensor and Actuator Networks

MDPI

Article

Machine Learning Attacks and Countermeasures on Hardware Binary Edwards Curve Scalar Multipliers

Charis Dimopoulos [1,2,*], Apostolos P. Fournaris [2] and Odysseas Koufopavlou [1]

[1] Electrical and Computer Engineering Department, Rion Campus, University of Patras, 26504 Rion-Patras, Greece; odysseas@ece.upatras.gr
[2] Industrial Systems Institute, Research Center ATHENA, Patras Science Park, 26504 Platani-Patras, Greece; fournaris@isi.gr
* Correspondence: c.dimopoulos@upnet.gr

Abstract: Machine Learning techniques have proven effective in Side Channel Analysis (SCA), enabling multiple improvements over the already-established profiling process of Template Attacks. Focusing on the need to mitigate their impact on embedded devices, a design model and strategy is proposed that can effectively be used as a backbone for introducing SCA countermeasures on Elliptic Curve Cryptography (ECC) scalar multipliers. The proposed design strategy is based on the decomposition of the round calculations of the Montgomery Power Ladder (MPL) algorithm and the Scalar Multiplication (SM) algorithm into the underlined finite field operations, and their restructuring into parallel-processed operation sets. Having as a basis the proposed design strategy, we showcase how advanced SCA countermeasures can be easily introduced, focusing on randomizing the projective coordinates of the MPL round's ECC point results. To evaluate the design approach and its SCA countermeasures, several simple ML-based SCAs are performed, and an attack roadmap is provided. The proposed roadmap assumes attackers that do not have access to a huge number of leakage traces, and that have limited resources with which to mount Deep Learning attacks. The trained models' performance reveals a high level of resistance against ML-based SCAs when including SCA countermeasures in the proposed design strategy.

Keywords: hardware security; machine learning; side channel attacks; embedded devices; Elliptic Curve Cryptography

check for updates

Citation: Dimopoulos, C.; Fournaris, A.P.; Koufopavlou, O. Machine Learning Attacks and Countermeasures on Hardware Binary Edwards Curve Scalar Multipliers. *J. Sens. Actuator Netw.* **2021**, *10*, 56. https://doi.org/10.3390/jsan10030056

Academic Editor: Davide Patti

Received: 31 May 2021
Accepted: 3 August 2021
Published: 16 August 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The impact Elliptic Curve Cryptography (ECC) has made as an approach to Public-Key Cryptography is profound, mainly by offering various digital signature, key agreement and certificate schemes. These solutions have been deployed in a wide variety of applications, which range from high-performance computing systems to low-end IoT and embedded devices.

A goal when developing an implementation of this kind should be a flexible and adaptable solution. In practice, this means that alongside high computational performance, efficient resource management is essential in cases where the execution environments might be resource constrained. By exploiting this approach, the suitable hardware-accelerated operations should be selected with the premise that they can provide the best trade-off between performance improvements and flexibility. In the majority of ECC-based implementations [1–3], the main computational barrier is related to the scalar multiplication operation, mainly realized through a series of point addition and point doubling operations. This fact makes it an ideal point of interest for optimization improvements in most ECC-based implementations, especially those targeted towards embedded systems. Literature confirms this fact, with extensive studies focusing on the optimization of the EC scalar multiplication and its arithmetic operations on Finite Fields in terms of speed, resources and power consumption improvements [4,5].

Despite the obvious aspects of flexibility and performance enhancement that need to be addressed during the design phase, another crucial consideration should be made in regard to the multiplier's Side Channel attack (SCA) resistance. As a side channel attack, we define the attempt of an adversary to extract sensitive information during an algorithmic execution. This information can be accessed through the implementation's side channels (power consumption, electromagnetic radiation, timing characteristics, etc.) and through extensive statistical analysis it can potentially reveal the hidden secret. Thus, the need for resistance against these attacks has been addressed in literature [6,7], denoting the wide range of side channel attacks that can be executed on these multiplier implementations. Both the aspects of performance enhancements and side channel attack resistance are in direct influence of the adopted scalar multiplication algorithm and the overall multiplier architecture.

The most powerful form of side channel attacks have traditionally been Template Attacks, proposed in [8]. They belong in the family of profiling side channel attacks, in which a profile for a given device/cryptographic primitive is generated, mapping the behavior during the execution of the primitive. This process, however, requires the use of a system identical with the device under attack, in order to capture the target's execution behavior (EM or Power Analysis) and generate the profile. In recent research efforts, there has been a shift towards performing machine learning and neural-network based analysis, since the essence of this type of attack is the profiling of the device under attack [9,10]. This attempt has been mainly focused on implementations of symmetric cryptography (DES, AES) and RSA when referring to public-key cryptography. Regarding Elliptic Curve Cryptography (ECC), there are works evaluating popular machine learning classification algorithms [11], with the trend being the further use of Deep Learning techniques.

Being motivated by the above, this paper initially describes the SCA landscape for Elliptic Curve cryptography, and the potential of profiling attacks such as TA and ML-based SCAs is highlighted. Using the formulation of the above description, the paper offers the following contributions on the topic:

- A scalar multiplier design strategy is proposed; the fine-graining of decomposition operations in the SM computations enables the introduction of SCA countermeasures that are hard to bypass for an attacker. Given that the MPL Scalar multiplication algorithm's EC point operations (point addition and doubling) in each round can be parallelized, we decompose them into their underlying finite field operations and propose their merging in a unified SM round computation flow with parallel stages. In each of those stages, SCA countermeasures can be introduced.
- An example use case of a scalar multiplier for a Binary Extension field is provided, so as to showcase the practicality of the design approach. In addition, in line with the proposed design strategy, an advanced side channel attack resistance enhancement roadmap is provided. This enhancement relies on the re-randomization of the point operation projective coordinates results in each MPL round. This is achievable by performing a finite field multiplication of each round's generated coordinates with a unique per-round random number.
- We describe a simple ML SCA roadmap showing how to attack the proposed use-case implementation (that follows the paper's design strategy and SCA countermeasure proposal) using simple ML algorithms that require small/medium sized number of leakage traces. The mounted attacks employ three ML models (Random Forest, SVM and MLP algorithms). We demonstrate actual simple ML-based SCAs mounted on two use case SM implementations, an unprotected one (GF operation MPL parallelism exploitation with no projective coordinate randomization) and a protected one (GF operation MPL parallelism exploitation with projective coordinate randomization), and we validate the resistance of the proposed design approach (with SCA countermeasures) as this is applied in the use case BEC scalar multiplier. The results show that the protected implementation is highly resistant against simple ML-based SCAs using any of the three modeling approaches.

The rest of the paper is organized as follows. Section 2 briefly describes the underlying mathematical foundation of ECC; Section 3 presents an overview of SCA on Elliptic Curve Cryptography and possible countermeasures; Section 4 describes the proposed fine-grained GF unification model and design strategy, as well as its expansion to support SCA countermeasures on the MPL architecture; Section 5 describes the machine learning algorithms applied in this work; Section 6 analyzes the experimental process that was followed and Section 7 discusses the outcome of the analysis executed on the impact of countermeasures. Section 8 concludes the work.

2. Elliptic Curve and Binary Edwards Curve Background

Elliptic Curve Cryptography (ECC) is now widely accepted and adopted as a solution for security applications based on asymmetric key cryptography. The mathematical complexity mainly lies on the point operation of the scalar multiplication process ($e \cdot P$), in which the scalar e is an element of a Finite Field and P denotes a point on the elliptic curve defined by the specific Field. The National Institute of Standardization Technology (NIST) has standardized the default EC equation form—the Weierstrass ECs. Their main drawback, however, is that not all points can be operated upon uniformly and symmetrically. The existence of those exception points makes this family of curves susceptible to side channel attacks, further increasing the required overhead needed for side channel resistance examination.

Binary Edwards Curves

The need to have an intrinsic mechanism for realizing a complete addition formula for every Elliptic Curve point led to the introduction of the Edwards Elliptic curve [12]. In [13], this family of curves, referred to as Binary Edwards Curves (BEC) was fully defined over binary extension fields $GF(2^k)$ in addition to the original definition over $GF(p)$.

$$d_1(x+y) + d_2(x^2+y^2) = xy + xy(x+y) + x^2y^2 \quad (1)$$

The generic form of the curve can be seen in Equation (1), where the two constant values that define the shape of the BEC (d_1 and d_2) must both be elements of $GF(2^k)$ meeting the requirements $d_1 \neq 0$ and $d_2 \neq d_1^2 + d_2^2$. An important advantage of BECs is their symmetry, the property declaring that for a given BEC point $P(x_1, y_1)$ there exists a $P(y_1, x_1)$ on the same curve. Paired with the fact that point addition and doubling operations follow the same mathematical formulation, it leads to an innate resistance against some simple side channel attacks (SSCA) such as Simple Power Attacks (SPA) [14]. The trade-off for this resistance is the increased number of $GF(2^k)$ operations compared to Weierstrass curves. Despite this, BECs present a regularity and completeness, since they have no BEC exception points in the addition formulas. This is valid as long as the trace $Tr(d_2) = 1$ or, equivalently, there is no element $t \in GF(2^k)$ that can satisfy the equation $t^2 + t + d_2 = 0$.

For BECs, explicit addition and doubling formulas exist in both the affine and projective coordinate planes [13]. However, by restricting d_1 and d_2 through an equation between them, the BEC Equation (1) can be degraded to a simplified form that is easily handled in cryptography. In order to avoid the significant cost of the inversion operation, a shift from affine to projective coordinates is realized in practice. Despite the ability of point doubling in projective coordinates to be performed using point addition formulas, additional simplifications can be applied to them for doubling, thereby further reducing the necessary $GF(2^k)$ operations, as well as the computation delay for this operation. The drawback of this simplification is that the protection against SSCAs described above is contradicted, thus raising the need for additional countermeasures. For a conversion to be performed from a BEC $\acute{P} : (X, Y, Z)$ point in projective coordinates to the corresponding $P : (x, y)$ affine point, the formula $x = X/Z$ and $y = Y/Z$ is used [15].

Despite the existence of various algorithms performing scalar multiplication, not many can be considered to provide resistance against information leakage. One scalar multiplication algorithm that has been proved to be resistant against some comparative and sophisticated Simple Power Attacks [6,7] is the popular Montgomery Power Ladder (MPL) algorithm. As can be seen in Algorithm 1 where MPL is presented, there is considerable regularity in each round of operations (step two of Algorithm 1), while dummy operations are absent during MPL execution. The MPL regularity can be justified by the constant number of identical point operations that are performed in each scalar round related to the corresponding key bit [16], which prohibits an attacker to use key-related irregularities (different number of type of operations for each bit of the secret e) for SSCAs/SPAs.

Algorithm 1: SPA Resistant MPL Algorithm

Input: P : BEC base point $\in EC(GF(2^k))$,
$e = (e_{t-1}, e_{t-2}, ...e_0) \in GF(2^k)$
Output: $e \cdot P$
1. $R_0 = \mathcal{O}$, $R_1 = P$
2. **For** $i = t - 1$ **to** 0
   If $(e_i = 0)$ then
   (a) $R_1 = R_0 + R_1$, $R_0 = 2 \cdot R_0$
   else
   (b) $R_0 = R_0 + R_1$, $R_1 = 2 \cdot R_1$
   end if
3. **Return** R_0

In Figure 1, the control flow graph for the MPL Algorithm is shown to better visualize the process followed during its execution.

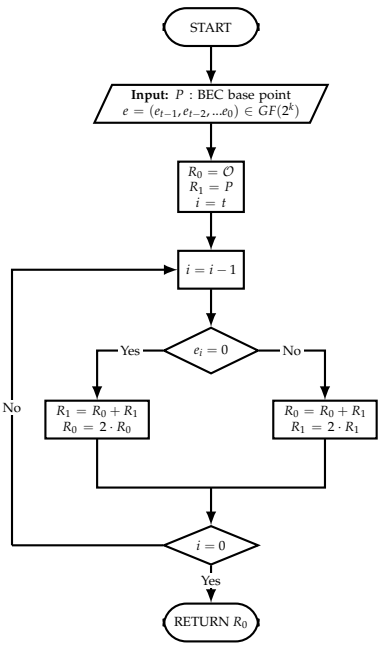

Figure 1. Control flow graph for MPL Algorithm.

While BECs synergize well—from a security standpoint—with the MPL algorithm due to their completeness and uniformity, this synergy is diminished by the BEC doubling optimization [12,13]. This fact, alongside MPL's vulnerability against Advanced SCAs (ASCAs), including Correlation and Differential SCAs [1,7], highlights the need to include additional countermeasures into the scalar multiplier implementation for broader SCA resistance.

3. ECC Side Channel Attacks and Countermeasures

Scalar Multiplication (SM) constitutes the main attack point of SCAs in Elliptic Curve Cryptography algorithms since the involved operations have a strong correlation with the secret scalar. The types of SCA attacks on SM that exist today can vary between simple or advanced on one hand and vertical or horizontal on the other, as seen in [17–19]. For the collection of the leakage traces/observations $L_i(t)$ to be realized in the vertical method, sets of same or different inputs are utilized at each time t ($t \in \{1, 2 \cdots p\}$) in the SM implementation within the total p times. Each observation is associated with the t_{th} execution of the implementation for an operation i. For the horizontal procedure, just a single execution is sufficient to capture the leakage traces of the unit under attack. In this case, a distinct time period is represented by each collected trace during the time frame of the execution [17].

If the double-and-add algorithmic option is followed in SM, the probability of simple SCAs being deployed in the implementation is increased substantially. These attacks are usually horizontal in nature, able to be mounted by only a distinct leakage trace. A straightforward counter against these type of SCAs is the utilization of SM algorithms that are highly regular, i.e., the scalar bit that is being processed in each round does not appear to have a link to the round's operations (e.g., MPL or Double and Always Add) [16]. Nevertheless, there exist some SCAs that have the ability to thwart this regularity, mainly by altering the EC base point used as an input of the SM [20]. For this objective, Comparative SCAs were introduced (initially based on Power Attacks but adapted to Electromagnetic Emission attacks as well). In such attacks the adversary is able to control the SM input, provide a few inputs (usually two) that are arithmetically related (e.g., Point P and $2 \cdot P$), collect the SM leakage traces and compare them using some statistical tool. Most widely known such attacks that manage to defeat many regular/balanced SM algorithms are the doubling attack (collision based attack) [21] (DA) and its variant, the relative doubling attack (RDA) [22], or the chosen plaintext attack in [23] (also known as the 2-Torsion Attack (2-TorA) for ECC). The above comparative attacks can be further refined by using special input points as is the case in Zero PA (ZPA) or Refined PA (RPA). In the former, an exceptional EC point P_0 (which can induce a zero P coordinate at round i) is loaded to the implementation, thus allowing scalar bit recovery by exploiting the vulnerability at the ith round.

Besides the attacks already mentioned, there exist other advanced SCAs that can be executed with either one or many collected leakage traces on EC SM. One of the most dominant types in this category are the Differential Side Channel Attacks (DSCAs), initially introduced by Kocher in [24], that depend on power consumption characteristics. DSCAs follow the hypothesis test principle [18,25], where a series of hypotheses $\acute{e}$ on the secret e (or part of the secret, for example some bit of the scalar e) are made for many p inputs X_i, where $i \in \{1, 2, \cdots p\}$ are used in some leaky internal operation of the algorithm along with some other known values (derived from each X_i, e.g., the plaintext or the ciphertext). The hypothetical result leakage is then estimated using some prediction model. The predicted leakages (for all predictions and all inputs X_i) are compared against all real leakage traces by using a suitable distinguisher Dis for all inputs X_i, in order to choose which hypotheses are correct. There are various distinguishers that have been used in literature, leading to sophisticated DSCAs. Most notably, the Correlation SCA demands less traces in order to recover a secret value when compared to the DSCA [26], and the Collision Correlation Attack [27–29] has the ability to be executed even if an adversary is unable to freely influence the inputs that the SM implementation demands [17]. Apart from vertical attacks, there is a broad range of horizontal advanced SCAs exploiting the fact that each O_i operation, when performed in off-the-shelf hardware, is partitioned into a succession of simpler operations (e.g., word-based field multiplications) that are all related to the scalar bit. In this case, a process similar to a vertical differential SCA is mounted using leakage partitions from a single trace. Attacks such as these are the Horizontal

Correlation Analysis attack (HCA) [30], the Big Mac Attack [31], or the Horizontal Collision Correlation Attack (HCCA) [18,19].

Advanced SCAs can follow various leakage models and may take many forms [32], but in general they fit the above description that assumes a leakage $L_i(t)$ from an intermediate computation $O_i(t)$ at round i of the SM algorithm can be modeled by taking into account the computational and storage leakage $\delta(O_i(t))$ under the presence of noise, as shown in the following equation:

$$L_i(t) = \delta(O_i(t)) + N_i(t) \quad (2)$$

where $N_i(t)$ is noise intrinsically associated with the device being tested and is independent of O_i. In the above equation, t corresponds to the time variable within the timeslot that O_i is active.

Traditionally, countermeasures can fit into two different classes, leakage masking and leakage hiding [1,14]. When applying the hiding method, suitable logic is included in the computation flow of the SM in order to make the leakage of the device independent from both the operations executed and their intermediate values. This can be achieved either by implementing cryptographic algorithms in such a way that the power consumption is random, or by implementing cryptographic operations that consume an equal amount of power for all operations and for all data values [14]. In practice, information hiding is realized by the implementation of suitable circuits that can add noise to the monitored side channel. These mainly include components that offer power scrambling redundancy or power consumption balancing (dual rail). A similar outcome can be achieved in the algorithmic level of the EC SM by introducing dummy calculations, or by modifying the algorithm so that $\delta(O_i(t))$ appears similar to $\delta(O_j(t))$ (where O_j is a different SM intermediate operation) for all t within a given timeslot that each one of the operations are active. The MPL algorithm, in conjunction with the nature of BECs, offers great conditions for developing such countermeasures. On one hand, a designer is able to parallelize multiple operations in the time domain with the help of the MPL; on the other, BECs further contribute to the aforementioned hiding procedure with their completeness and uniformity. Many realizations of the above SCA protection approach cannot effectively thwart some Advanced SCAs or, in addition, some Comparative SCAs, including the Doubling Attack (DA) group, i.e., normal and Relative (RDA) [21,22].

The goal of masking is the decoupling of sensitive data from their corresponding leakage trace. Crucial for this method is the existence of some randomization technique (multiplicative or additive) upon the secret information that has a sensitivity to leakage $O_i(t)$ [20]. Three well-established EC SM masking techniques have been used in practice (appearing in several variations), originally proposed by Coron in [33]. The values that are the targets of randomization by those countermeasures are the EC input point P (i.e., base point blinding), the projective coordinates in the (X, Y, Z) domain of the same point and the EC scalar vector (Algorithm 1 e) of the SM. These mentioned randomization methods have been extended in ECC with several forms after their introduction in literature. In terms of applicability, though, the two latter ones (projective coordinate and scalar blinding) are easier to implement. This is due to the fact that the first described method, point blinding technique, demands the inclusion and storage overhead of an additional random point that is generated in every multiplication round, which introduces additional complexity [34].

Profiling Attacks

The above existing SCA research (especially univariate Differential SCAs) creates momentum for the description of a generic DSCA mechanism that will be able to utilize a generic-compatible distinguisher that uses a generic leakage model on a generic device for which no information is provided. However, such a generic case is not realistic, as analyzed in [35], and it is concluded that existing DSCAs (with a specific distinguisher, model and insight on the device under test and the implemented SM algorithm) may be less effective than anticipated. Whitnall et al. [35] indicate that another type of SCA can be considerably more potent than DSCAs, i.e., profiling attacks. The most widely used profiling attack is

the template attack (TA); it requires two identical devices, which we refer to as the profiling and the attacked device. In the profiling device, a profiling phase occurs, where the attacker identifies the operation that produces information leakage ($\delta(O_i(t))$) and provides as input to the device all possible values (ideally) of a secret value portion (e.g., one byte or one bit) that influences O_i. This will produce all possible different states of this operation for which the attacker collects the leakage $L_i(t)$. During the profiling phase of a TA on SM, the scalar bit can act as the TA input, and the point addition or point doubling operation values associated with the scalar bit value can act as $O_i(t)$. The collected traces, along with the associated input values are used, in order to estimate a leakage model (by calculating the parameters of a leakage Probability Density Function (PDF)) which is considered the TA's template (or profile in general). In the second TA phase (attack phase), an attacker uses an attacked device with an unknown secret (i.e., secret scalar) and collects leakage traces of $O_i(t)$ for various inputs t using the same trace-collection mechanism and parameters as in the previous case. Using some discriminator, the attacker tries to identify from the template an appropriate leakage trace that has the highest probability of matching the unknown secret leakages, and then retrieves the secret in the template associated with the selected leakage trace. The attack exists in several variations [36,37], with various PDF estimators and even in online form (Online TA) where the profile is created during the collection of the traces on the device under attack [38].

The concept of creating a leakage model based on labeled leakage traces has been further researched in academia by adopting Machine Learning (ML) techniques in order to create an ML-produced leakage model. With ML, an adversary does not need to create a perfect leakage model (using all possible inputs of a secret value block) but instead allows an ML model to be built with a non-exhaustive series of leakage traces (that can only be corresponded to specific, instead of all, secret block values). As the leakage becomes more noisy (either due to the hiding or the masking countermeasures implemented), more accurate results can be extracted by applying an approach based on ML profiling [9,10,20].

The analysis carried out above leads us to the conclusion that an SM implementation cannot be considered secure if it does not incorporate methods to protect against a varied range of SCAs. As several researchers point out [36,38,39], Profiling SCAs can overcome several of the existing countermeasures. Thus, the necessity of a more advanced randomization implementation is highlighted for the entirety of the computation flow during an SM.

4. Proposed SCA Countermeatures on MPL

The analysis in the previous section indicates that the MPL SM algorithm is not sufficient to thwart many of the existing Advanced SCAs, including the formidable profiling attacks. Thus, SM MPL should be infused with advanced countermeasures based on masking and/or hiding (using randomization). However, MPL has some interesting features that can be exploited in order to make ASCAs difficult. These features stem from the regularity of each SM round. Each MPL round always performs point addition and doubling. Moreover, the order of execution of these operations within an SM round is not important so the two operations can be performed in parallel. Given the fact that most of the SM SCAs focus on the leakage of a single point operation (usually point doubling), merging point doubling with point addition by performing them in parallel can potentially make an attacker's work harder. In fact, this parallelism approach forces the attacker to change the leakage observation operation from point addition or doubling to one full MPL round. Given that the leakage of each point operation has an associated intrinsic noise level, the leakage of two combined operations will potentially increase this integrated noise. The identification of each one of point operations is, thus, very hard to pinpoint, making attacks harder to mount. So, assuming that O^A is the point addition operation and O^D is the doubling operation in a single MPL round, the MPL round leakage for a certain round i would be $L_i^R(t) = \delta(O_i^A(t)) + \delta(O_i^D(t)) + N_i^A(t) + N_i^D(t)$. For the sake of simplicity, we can exclude, from this point onwards, the parameter of time in the above equation, assuming a constant time slot T for each MPL round and considering the leakage at an

MPL round instead of a single time instance within the round. In addition, note that $N_i^A(t)$ and $N_i^D(t)$ correspond to the noise associated with point addition and point doubling, respectively. Unfortunately, $\delta(O_i^A) + \delta(O_i^D)$ could potentially have non-trivial information leakage of the SM secret value (i.e., the scalar bit on a given SM MPL round) on a single O_i^A or O_i^D. The issue can be mitigated by a design approach that more deeply fuses O_i^A and O_i^D, i.e., on the EC underlying Finite Field operation level.In textbook point addition and doubling design and implementation, EC points are represented in the projective space (EC point projective coordinate), and O_i^A or O_i^D can be partitioned into many finite field multiplications and additions/subtractions that need to be executed in discrete time instances within the MPL round execution time slot T. The O_i^A and O_i^D operations that are performed in parallel within the MPL round could be considered a unified round operation, with inputs and outputs that can be point addition and point doubling coordinate results. This unified round operation can then be fine-grained into a complex sequence of finite field operations that will collectively contribute to the point addition and point doubling result of an MPL round. These GF operations can be synchronized with the time units of the MPL round's time slot T, based on their data dependence and the capabilities of the underlying computing device. The latter constraint concerns the number of GF operations that can be performed in parallel on a given hardware architecture which has a finite number of operating units (finite field multipliers and adders/subtractors) within the SM architectural design (the typical case in hardware SM implementation and some software ones).To identify which GF operations can be performed in parallel in a given time slot of T, a Data Flow Graph analysis of the GF multiplication and addition sequence is realized. This analysis is performed for a single unified MPL round computation while also assuming a specific finite number of parallel processing units (for finite field multiplication and addition) exist. Such analysis can reveal that the computation can be partitioned into z parallel stages (corresponding to a single time slot of T) where each stage performs p concurrent GF operations while processing for both O_i^A and O_i^D. Assuming that the v_{th} GF operation within the j_{th} parallel-processing stage is denoted by $O_i^{P(v,j)}$, leakage $L_i^R(t_j)$ (the leakage within round i for the time t_j of the j_{th} parallel stage) becomes:

$$L_i^R(t_j) = \sum_{v=1}^{p} \delta(O_i^{P(v,j)}(t_j)) + N_i^R(t_j) \tag{3}$$

where $N_i^R(t_j)$ corresponds to the noise that exists during the computation of leakage round i for the time t_j of the j_{th} parallel stage. This noise is related to both point addition and doubling computations. Thus, the overall leakage of a single i_{th} MPL round, where O_i^A and O_i^D are merged into unified computations and decomposed into GF operations in each time point t_j for all j values inside T, will be the following:

$$L_i^R(T) = \{L_i^R(t_1), L_i^R(t_2), L_i^R(t_3), ..., L_i^R(t_z)\} \tag{4}$$

From the above equation and relevant analysis, it can be remarked that the MPL SM round computation has now been shifted from a processing model closely related to the scalar bit operation (related to O_i^D or O_i^A) to a processing model of regularly executed autonomous, unified stages of parallel operations that are associated with both O_i^D or O_i^A, and loosely relate the the scalar bit at round i. Using the proposed fine-grained GF unification stages approach, the O_i^D and O_i^A are processed in a unified way.

Without any additional countermeasure, in general, the BEC MPL implementations suffer from MPL's lack of resistance against Advanced SCAs (ASCAs), including Differential SCAs, Correlation SCAs, etc. [1], but–most importantly–against Profiling Attacks [7,39]. The inclusion of the fine-grained GF unification stages approach could potentially reduce the MPL leakage. However, t-test leakage assessment results performed in [34], where a similar approach is used, reveal that there is still enough leakage associated with the base point and the secret scalar to reveal the secret. This is an expected result, since $\delta(O_i^{P(v,j)}(t_j))$

does not model only the computation leakage during a single parallel stage, but also the intermediate values' storage leakage at this stage. Specifically, for power consumption side channels where dynamic power consumption is leaking information, the register storage operations form a distinct leakage pattern for scalar bit $s_i = 1$ or $s_i = 0$ when all $O_i^{P(v,j)}(t_j)$ are included. However, no actual attack is performed in [34] to show if this leakage can be exploited, especially for potent Profiling SCAs. The aforementioned fine-grained GF unification stages approach, as modeled in Equations (3) and (4), relies on the intrinsic scrabbling of the implementation parallelism to hide the leaked information. To enhance this information hiding with masking, an additional source of randomness can be introduced in the computation process of each MPL round i. This can be achieved by including some random computation in some of the $O_i^{P(v,j)}(t_j)$ operations included in a parallel stage.

For the sake of efficiency, it suffices to include random operations within only a few parallel stages of each MPL round and to let the randomness propagate to the remaining stages. The choice one needs to make in order to identify the best-possible stage to add the random operation is dependent on three parameters. The first parameter is the available parallel computation units that the implementation can support (e.g., the number of parallel finite field multipliers and adders). Secondly, the number of units that are in an idle stage (i.e., awaiting correct inputs from another computation) and lastly, the data dependency that exists between the different parallel stages in the executed MPL round. Furthermore, the inclusion of random operations in some parallel stages should not lead to a false computation result of the MPL round. In some cases, the same outcome can be achieved by including additional parallel stages, so that the randomly injected information per round is handled correctly.

The above masking mechanism can be realized by appropriately adapting existing randomization techniques in the research literature. Coron's countermeasures [33] constitute the basis of many algorithmic SCA-resistance solutions. Among them, projective coordinates randomization may be the most easy to deploy (and simultaneously one of the most efficient to deploy). In this approach, the input point $(X : Y : Z)$ is randomized by multiplying its coordinates with a randomly generated number h, thus, forming $(h \cdot X : h \cdot Y : h \cdot Z)$. At first glance, this point seems to be a different one than the original, when viewed in projective coordinates; however, in reality it is the same point in affine coordinates. Coordinate randomization is applied once in the MPL. Before the first round of the algorithm, however, this approach provides no protection against RPAs. Adopting and adapting the coordinate randomization countermeasure to the proposed fine-grained GF unification stages mechanism and enhancing its potency, we propose the inclusion, in some parallel stages of each MPL round's i, a different multiplicative randomization of all the round's projective coordinate point outcomes (both point addition and doubling results), using a round's random number h_i.

Instantiating the Complete SCA Resistance Mechanism Based on the Proposed Approach

The above proposed SCA resistance approach can be instantiated both in hardware and software implementations that support parallelism in their computation flow. It is expected that the application of the proposed approach in software implementations that have high level of device noise ($N_i^R(t)$) due to parallel processing of unrelated processes (e.g., Operating System functions), will considerably increase the noise level. Such noise, in combination with the proposed added random operations per parallel computation stage, will make the Signal-to-Noise (SNR) ratio significant enough to make SCAs difficult to mount [39]. In hardware implementations though, where the noise is only related to the ASIC or FPGA chip's hardware resources, the impact of noise is small, and SCA resistance relies mostly on the potency of the proposed solution. For the above reason, in the remainder of the paper, the analysis is focused on hardware implementations of the proposed SCA resistance mechanism in BEC ECCs.The explicit formulas for point addition and point doubling that are defined in [13] can be decomposed into the single $GF(2^k)$ operations presented in Table 1. Based on those operations, a possible realization of the

proposed fine-grained GF unification stages technique can be presented in Table 2 [34]. In this table, each row corresponds to a parallelism stage; thus, one MPL round is concluded in 13 stages ($z = 13$). In the last two stages of the MPL round i, random operations have been included (marked in the table with a different cell color). In these operations, the projective coordinates of all output BEC points are multiplicatively randomized using a unique random value h_i in each round (each coordinate is multiplied with h_i). The hardware design adopts the hardware architecture described in [34]. Each parallelism stage in the hardware architecture includes three finite field multipliers (M1 to M3) and three adders (Ad1 to Ad3) that can operate in parallel.

Table 1. Point Operations Partial Results [34].

Point Addition $(X_3 : Y_3 : Z_3) = (X_1 : Y_1 : Z_1) + (X_2 : Y_2 : Z_2)$		**Point Doubling** $(X_{3D} : Y_{3D} : Z_{3D}) = 2(X_1 : Y_1 : Z_1)$
$A = X_1 \cdot X_2$	$U_3 = U_1 + U_2$	$DA = X_1 \cdot X_1$
$B = Y_1 \cdot Y_2$	$U_4 = L \cdot U_3$	$DC = Y_1 \cdot Y_1$
$C = Z_1 \cdot Z_2$	$U_5 = F + U_4$	$DE = Z_1 \cdot Z_1$
$D = d_1 \cdot C$	$U = C \cdot U_5$	$DB = DA \cdot DA$
$E = C \cdot C$	$V_1 = A \cdot B$	$DD = DC \cdot DC$
$F = d_1 d_1 \cdot E$	$V_2 = G \cdot H$	$DH = DA \cdot DE$
$G_1 = X_1 + Z_1$	$V_3 = d_1 \cdot E$	$DI = DC \cdot DE$
$G_2 = X_2 + Z_2$	$V_4 = V_1 + V_2$	$DL = DE \cdot DE$
$G = G_1 \cdot G_2$	$V_5 = V_3 + V_4$	$DF = d_1 \cdot DL$
$H_1 = Y_1 + Z_1$	$V_6 = L \cdot V_5$	$DJ = DH + DI$
$H_2 = Y_2 + Z_2$	$V_7 = D \cdot F$	$DO = d_2 \cdot DJ$
$H = H_1 \cdot H_2$	$V_8 = V_7 + V_6$	$DM = DB + DD$
$I = A + G$	$V = Z_3 + V_8$	$DG = d_1 d_2 \cdot DM$
$J = B + H$	$M_1 = A + D$	$DK = DG + DO$
$K_1 = X_1 + Y_1$	$N_1 = G + D$	$DL_1 = DF + DJ$
$K_2 = X_2 + Y_2$	$O_1 = M_1 \cdot N_1$	$DL_2 = DH + DD$
$K = K_1 \cdot K_2$	$M_2 = B + D$	$DL_3 = DI + DB$
$L = d_1 \cdot K$	$N_2 = H + D$	$DX_3 = DL_2 + DK$
$U_1 = K + I$	$O_2 = M_2 \cdot N_2$	$DY_3 = DL_3 + DK$
$U_2 = J + C$	$P_1 = D \cdot O_1$	$DZ_3 = DL_1 + DG$
$P_2 = D \cdot O_2$		
$X_3 = V + P_1$		
$Y_3 = V + P_2$		
$Z_3 = U$		

In Table 2, it is revealed that the parallel operations of each stage that appears in Table 1 for an MPL round, are associated with both point operation (addition or doubling) in this round. Moreover, two extra stages have been introduced in the table in line with the leakage model of Equation (3) (no random operations in parallel stages). Those stages (marked in blue in the table) add SCA resistance following the hiding resistance approach. This hiding countermeasure introduces a timing and computing overhead of approximately 7.69%, which can be considered a satisfactory trade-off. Despite the fact that for all z stages the processing that occurs in each round has high regularity in regards to the scalar bit value, the storage leakage footprint might be different depending on the scalar bit. This can be observed from Algorithm 1 steps 2a or 2b, where the point addition and point doubling results are stored in R1 and R0, respectively, for $e_i = 0$ and in R0 and R1, respectively, for $e_i = 1$. This storage in different registers, which is related to e_i, is not masked/hidden when the proposed fine-grained GF unification stages approach is adopted without the random operations in each parallel stage, which hide the values stored in the implementations' registers. Despite the fact that this information might not be exploitable in traditional SCAs/ASCAs, it may provide significant benefit in profiling attacks where the attacker has the ability to create a concrete profile of the device. In the following sections, we perform analytic practical evaluation of the proposed SCA countermeasure as is implemented in a BEC scalar multiplier, following the computation flow of Table 2. The analysis is focused on simple ML-based SCAs that do not require a huge number of leakage traces for training and validation.

Table 2. Paralleling BEC point addition and doubling $GF(2^k)$ operations on the i_{th} MPL Round ($v = 6$ and $z = 11$ or $z = 13$).

Inputs		$(X_1 : Y_1 : Z_1)$		$(X_2 : Y_2 : Z_2)$			**Leakage**
Stage	**M1**	**M2**	**M3**	**Ad1**	**Ad2**	**Ad3**	$L_i^R(T)$
1	A	B	DA	G1	G2	K1	$L_i^R(t_1)$
2	G	DC	DB	H1	H2	K2	$L_i^R(t_2)$
3	H	DD	DE	I	-	-	$L_i^R(t_3)$
4	C	DH	DI	J	-	DM	$L_i^R(t_4)$
5	V2	V1	K	DJ	DL2	DL3	$L_i^R(t_5)$
6	DO	E	DG	U2	V4	U1	$L_i^R(t_6)$
7	D	V3	L	DK	-	U3	$L_i^R(t_7)$
8	DL	F	U4	M1	V5	N1	$L_i^R(t_8)$
9	V7	V6	O1	M2	N2	U5	$L_i^R(t_9)$
10	DF	O2	U	V8	DX3	DY3	$L_i^R(t_{10})$
11	-	P2	P1	DL1	-	V	$L_i^R(t_{11})$
12	rDY3	rZ3	rDX3	DZ3	Y3	X3	$L_i^R(t_{12})$
13	rY3	rX3	rDZ3	-	-	-	$L_i^R(t_{13})$
Outputs		$(X_3 : Y_3 : Z_3)$		$(X_{3D} : Y_{3D} : Z_{3D})$			
Rand Outputs		$(rX_3 : rY_3 : rZ_3)$		$(rX_{3D} : rY_{3D} : rZ_{3D})$			

-: idle r: random number; M1: $O_i^{P(1,j)}$, M2: $O_i^{P(2,j)}$, M3: $O_i^{P(3,j)}$; Ad1: $O_i^{P(4,j)}$, Ad2: $O_i^{P(5,j)}$, Ad3: $O_i^{P(6,j)}$.

5. Machine Learning Attacks

Machine Learning has been—for several years—an active field of research, with innumerable obscure practical applications such as stealing an ML classifier through Deep Learning [40], and even in the cyber-security domain has been used as a defensive tool for malware detection [41]. For the assessment of the SCA resistance of our ECC Multiplier, the following ML methodology was applied.

In [20], a time-efficient Convolutional Neural Network (CNN)-based approach is realized, focusing on dimensionality reduction in order to handle the high computational complexity of CNN attacks. In the proposed CNN model, the optimal number of convolutional blocks is used to build powerful feature extractors within the cost limit. The model consists of three combinational layers, each one containing a pair of convolutional layers and one pooling layer. The overall architecture is complemented by a dimensionality-reduction module and a class imbalance module in order to optimize the collected dataset prior to the training process. However, ref. [20] does not assess the ECC SM implementations against small datasets and simple Machine Learning algorithms. Thus, in order to form a complete performance analysis for small and medium sized datasets, this paper uses two supervised Machine Learning algorithms and one simple artificial neural network, complementing and extending the findings in [20]. For the work in [20] and in this paper, the same unprotected and protected implementations of the ECC SM accelerator have been utilized to create the trained models.

5.1. Random Forest (RF)

Random Forest belongs to the class of supervised machine learning algorithms, operating on the basis of bagging many random decision trees to give a more accurate classification. Random Forests [42] can be defined as a group of tree predictors that are combined in a such way that each tree is dependent on a random vector which is sampled independently and with the same distribution for all the trees in the forest. Structurally, they resemble diagrams made of nodes and edges, where these nodes can be of the types *root* (top of the tree), *internal* and *leaf* (end nodes). These leaf nodes have as values the target label that we want each sample trace to hopefully converge correctly to (in this side-channel analysis context, it is the binary classification of bit '0' or bit '1'). To enhance the accuracy, the final decision is made through a majority vote among a set of trees, each with randomized feature sample value.

5.2. Support Vector Machine (SVM)

Support Vector Machine [43] also belongs to the supervised category of Machine Learning algorithms and can be utilized for both classification and regression analysis. The basis of its functionality is the construction of a hyperplane, which separates the training data points of any class. A larger margin between the nearest training data points means a better separation and a lower generalization error of the classifier. In cases where a linear hyperplane between the data points is inadequate to provide a clear separation, a non-linear kernel function can be used, essentially mapping the data points into high-dimensional feature spaces.

5.3. Multilayer Perceptron (MLP)

Part of the class of feed-forward Artificial Neural Networks (ANN), the Multilayer Perceptron typically refers to a network of interconnected perceptrons (introduced by [44]), often in a multitude of layers. The three main layer categories involve an *input* layer, one or more *hidden* layers and an *output* layer. The connections between the nodes (neurons) for all but the input layer are characterized by their weight w value. This value is essential for the correct application of the activation function in mapping the weighted input to an output of the node. In essence, an MLP model can be considered properly trained when each of those weights for all layers has converged to the correct value. The supervised learning technique used in this case is called 'backpropagation', in which the weights are adjusted based on the output errors relative to the desired value.

6. Experimental Process

The evaluation process requires the collection of a series of traces. The collection of the required traces was achieved in a controlled environment as previously described in [34]. The SAKURA-X board [45], featuring a Kintex-7 cryptographic FPGA chip, is regarded as a widely used tool for its low-noise trace collection characteristics. The trace collection mechanism described and implemented in [46] has been utilized for leakage trace sampling, irregardless of the underlying executed implementation (AES, ECC, etc.). Thus, in our process it was used as the control loop of the conducted leakage assessment mechanism, enabling the collection of waveforms for the two versions (protected and unprotected). In order to capture the generated traces, a PicoScope 5000D Series Oscilloscope with a sampling rate of 1 GS/s was connected through a pre-amplifier to a resistor onboard the SAKURA-X, in order to measure the power consumption.

6.1. Sample Trace Formation

In contrast to the approach used in [34], each collected trace, which corresponds to a full scalar multiplier, is split into trace blocks, each of them representing a Scalar Multiplication round for a known random scalar bit. Given the clear visual pattern of each round (as it can be seen in [34]) in the collected traces, we were able to accurately identify the timeslot T corresponding to a single SM round. Given that each round in the MPL algorithm takes constant time, each collected trace was divided into n timeslots T, where n is the number of MPL rounds or, in other words, the number of scalar bits. Moreover, given that each round performs the necessary processing for a single scalar bit, and also given that in the trace-collection process known scalars were used, each trace block (corresponding to a single round for a known scalar bit) was labeled as a processing of '0' or '1', respectively. Using this splitting strategy for the traces, with each block representing a single bit, we were able to quickly and efficiently collect a vast amount of samples without wasting a lot of time. The above process constitutes the profiling phase, where the attacker has full access on the device and is operating using a known secret scalar e. Figures 2 and 4 show the waveforms captured during the execution of one round in the unprotected SM implementation, labeled as processing of scalar bit '0' or '1', respectively. Likewise, Figures 3 and 5 show the waveforms for a single round in the protected SM version of the accelerator, with the randomization countermeasure rounds shown in detail. Note that

there are 11 clearly visible patterns in Figures 2 and 4 corresponding to the 11 stages of Table 2's unprotected approach. Similarly, in Figures 3 and 5, there are 13 patterns that indicate the stages of the protected implementation of Table 2.

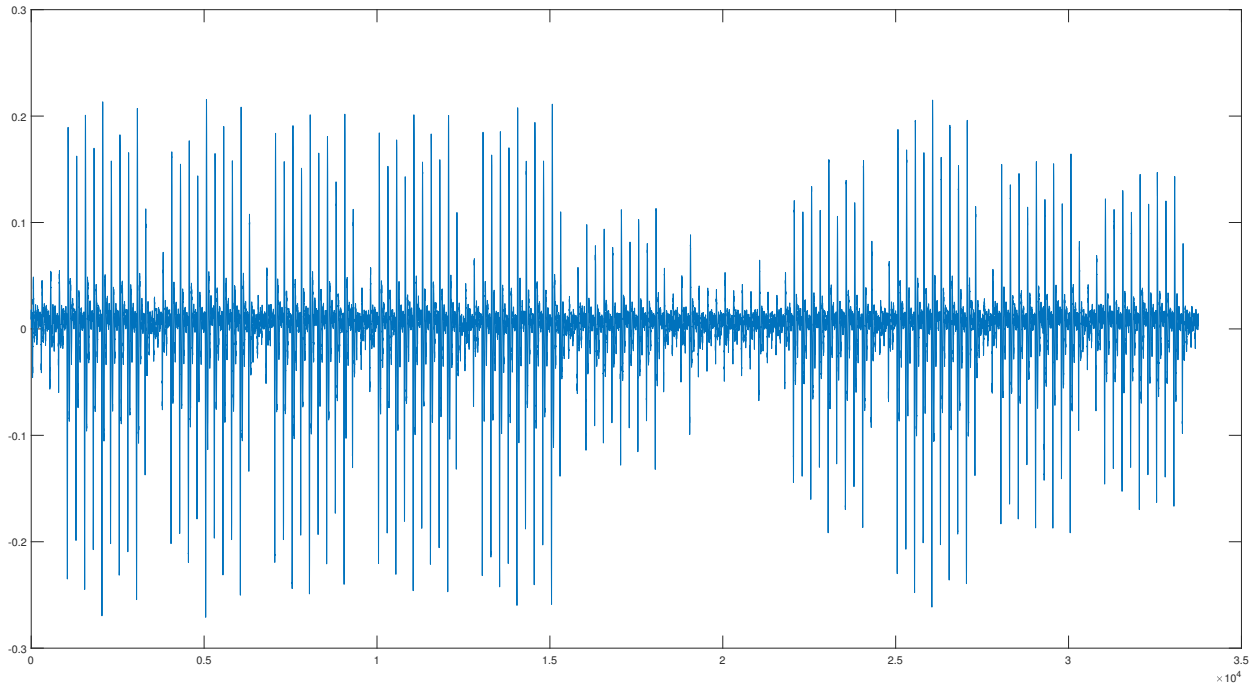

Figure 2. Sample of Unprotected implementation representing bit 0.

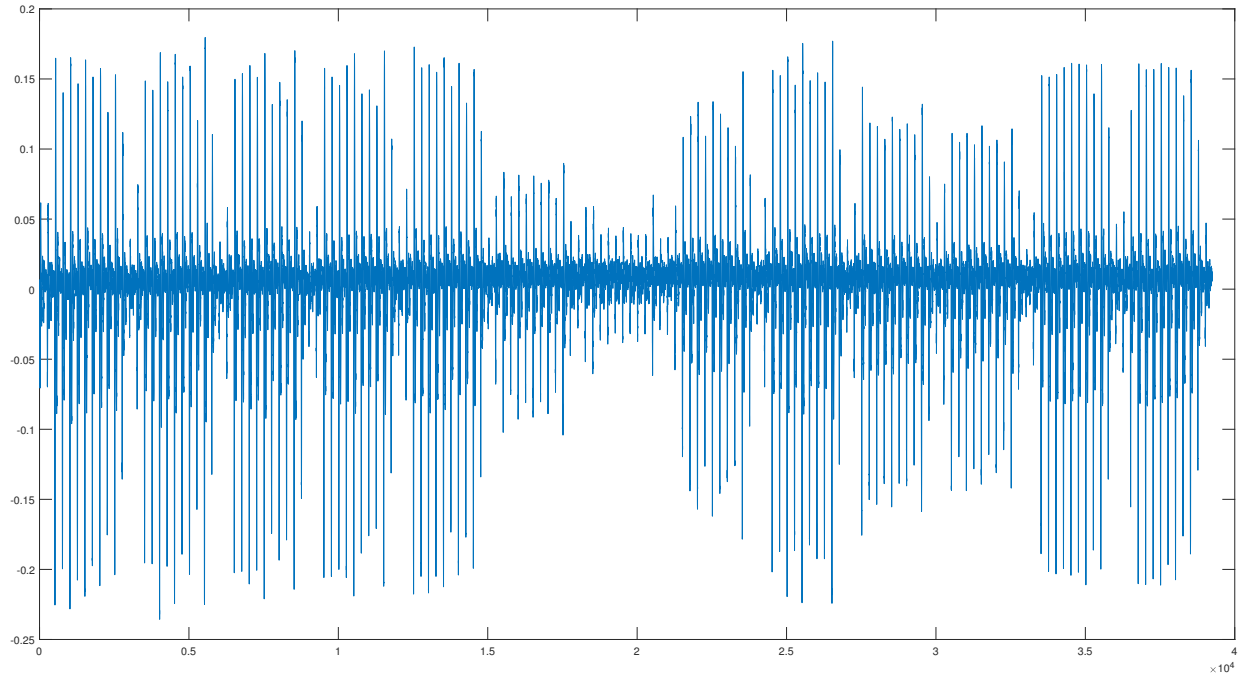

Figure 3. Sample of Protected implementation representing bit 0.

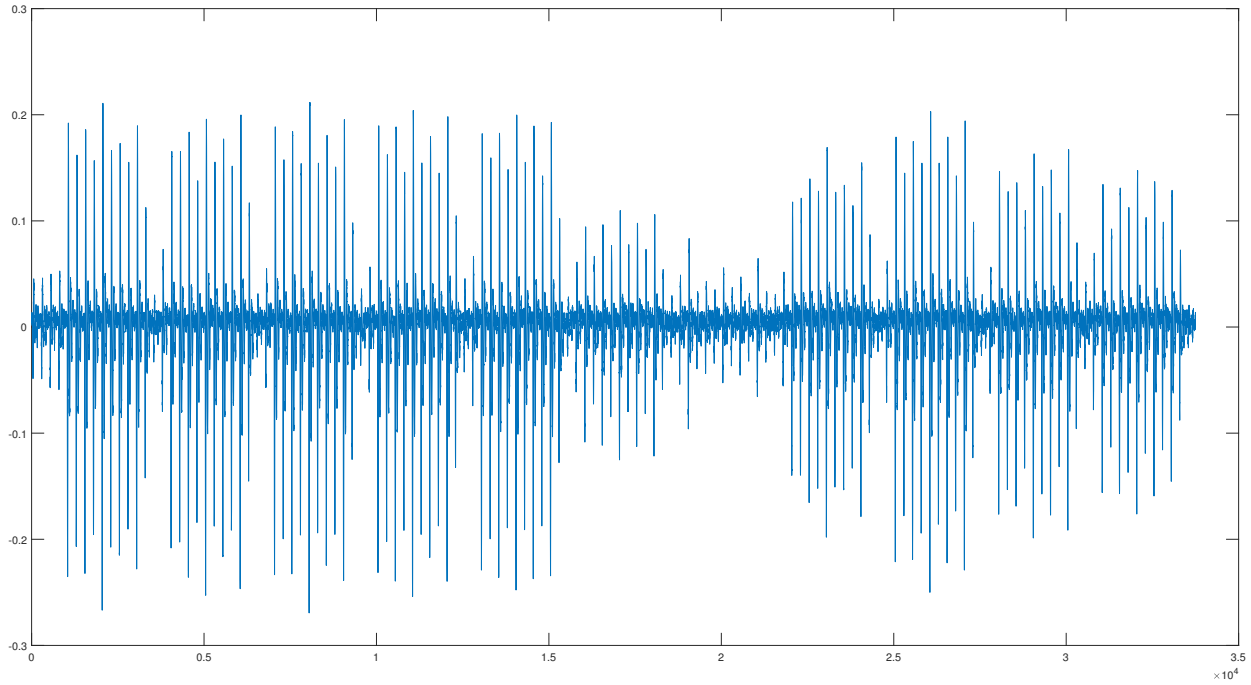

Figure 4. Sample of Unprotected implementation representing bit 1.

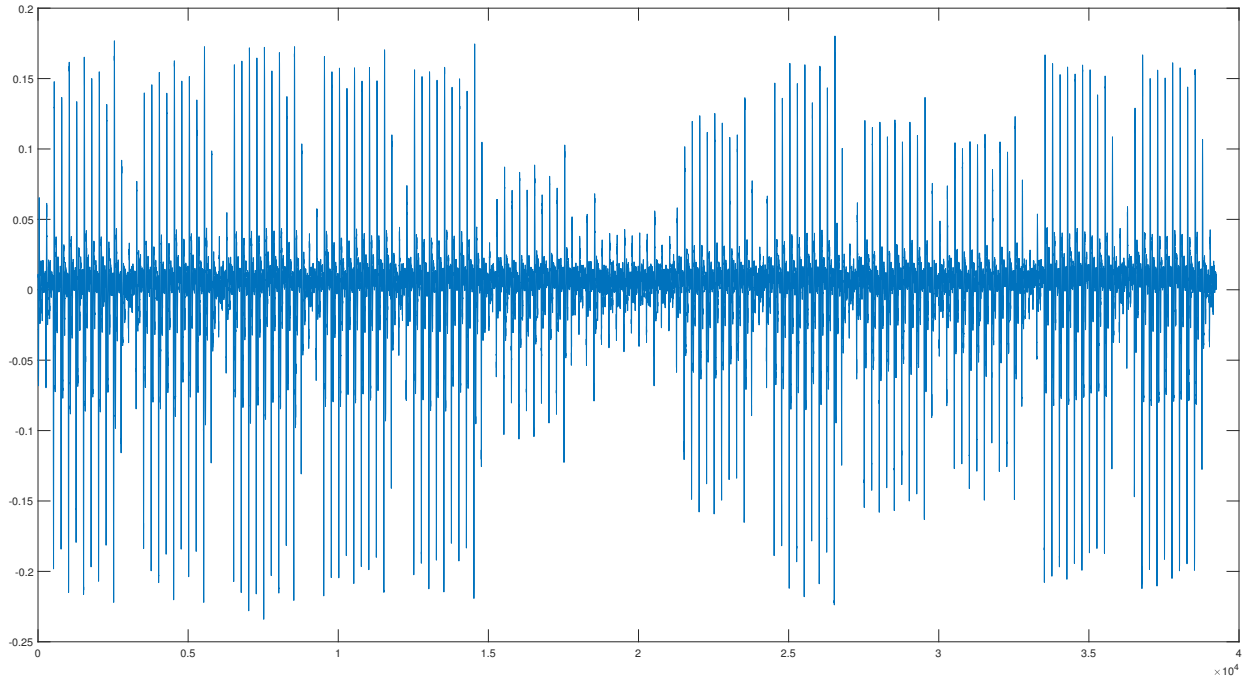

Figure 5. Sample of Protected implementation representing bit 1.

The low-noise captured traces can be considered as very well aligned, without the need for averaging or post-acquisition alignment, due to the combination of the SAKURA-X, the collection mechanism and the PicoScope 5000D. In total, 1000 full Scalar Multiplications were captured, which upon further splitting amounted to a maximum of N = 232,233 different sample block traces labeled '0' and '1'. These raw sample traces consist of 33,750 sampling points each, ready to be analyzed for feature generation. In order to train the different models, the Scikit-Learn library [47] was utilized, offering a fast and simple development environment with good functionality. Characterizing a dataset as

small or medium sized for ML-based side channel analysis is considerably dependent on the targeted cryptography algorithm (and relevant implementation) on which the attack is intended. Typically, simple Machine Learning algorithms use small- to medium-size datasets while Deep Learning algorithms (e.g., Neural Networks) use larger datasets for training. In the research literature, the few public SCA raw trace datasets that exist can provide an indication of the dataset size characterization. In the ASCAD database (one of the well-used public databases of DL SCA models for symmetric key algorithms) [48] the authors collect around 100,000 traces to build the DL models. Similarly, in the study of [49], the authors study DL models for symmetric key using 500,000 traces. When constructing models for public key cryptography algorithms, the needed number of traces to train a model can increase even further. Thus, the number of traces needed to train SCA models of size more than 100,000 can be considered a large dataset. In simple ML SCAs for ECC, such as the work of [11] which is focused on an unprotected ECC FPGA implementation, a reasonably sized dataset of 14,000 traces has been used. While in our paper we have collected a large number of traces (1000 scalar multiplications have been captured with 233 bits of the scalar processed in each one of them, thus resulting in 232,233 traces), we only needed to use 5000 traces for our models, which is considerably less than the dataset size used in other works (e.g., 14,000 traces in [11]). For this reason, our traces dataset can be considered of small or medium size. It should be noted that increasing the dataset size in the ML training did not improve the accuracy of the models.

6.2. Feature Generation

An important and crucial step to take into account when training any machine learning or neural network model is choosing the features of the dataset most capable of delivering as high prediction accuracy as possible. In this effort, targeting efficient model generation and training time, we adopted the strategy depicted in [11,50], where signal properties of the collected traces are used as features. Based on the recommendation in these papers, the following signal properties for every trace were measured to be used as features:

- Mean of Absolute Value
- Kurtosis
- Median Frequency
- Mean Frequency
- Slope Sign Change

The computation of these signal properties was achieved through the usage of already-available toolsets implemented in a MATLAB environment, as well as in-house built Python scripts, verifying with two different implementations, the correctness of our feature generation process. A preliminary observation on the values of the features, indicates that Kurtosis and Slope Sign Change might have the highest contribution regarding the final classification decision each model makes. After generating the appropriate values, the formation of the feature array is completed, appropriately assigning the correct labels to each feature value of the collected traces. This means that for each captured trace, such as those shown in Figures 2–5, the raw 33,750 signal data points are now reduced to a five-value array containing the computed features.

It is a prevalent conclusion that this feature generation process, in which signal properties are used as features rather than raw signal data, allows for a significant decrease in input data amount necessary to train the different ML models. The training time is, thus, greatly shortened for the majority of the models, allowing for quick evaluation and assessment of the implemented countermeasures, as well as of the impact each tuned parameter has on the accuracy of the trained model.

6.3. Feature Extraction

Noise is considered one of the biggest obstacles when attempting a successful side channel attack. In order to overcome this barrier, it has been shown [51] that using statistical methods such as Principal Component Analysis (PCA) as a pre-processing method has

the potential to lead to a reduction in noise in the leaked side channel data. This fact elevates techniques such as PCA into potentially valuable feature extraction mechanisms. During this process, the data that are evaluated to be redundant and not able to provide any contribution to the critical leakage are then removed from the finalized feature set. In our case and in similarity with the above feature generation procedure, PCA was applied on the raw signal traces containing all the collected information. This leads to a significant reduction in needed data to just two computed Principal Components. These values are then labeled correctly before being fed as input data into our models. As mentioned above, this additional way of reducing the training portion of the data can lead to shortened training time and better classification accuracy. Out of the three machine learning algorithms described in Section 5, Support Vector Machines seem to provide better performance, compared to the other algorithms, when using PCA pre-processing, due to their high-dimensionality characteristic.

6.4. Validation

Caution during the training of any Machine Learning model should be exercised in order to avoid the effects of underfitting and overfitting. Underfitting happens in cases where the dataset is excessively large or there are errors in the input data. The model then cannot correctly train on the available dataset, producing wrong classification results. On the other side of the spectrum, when the trained model can predict the training data with incredibly high accuracy, this is a sign of overfitting, as the model cannot generalize its predictions to new input data. To overcome these effects, the *K-Fold* Cross Validation technique has been adopted. More specifically, the input dataset that has been generated with the signal properties as features is divided into training and testing data portions. For our model, the training portion of the data is 20% of the total samples collected. As is apparent, the training of the model is completed using the training portion of the data, while the testing portion is only used to assess the classification accuracy without any impact during the model training phase (holdout method). The whole dataset is then subjected to the same procedure *K-1* times, where in each iteration a different percentage of datapoints reflects the training set and the testing set, respectively. The model is, thus, trained *K* times using a different pair of subsets, after which the final classification accuracy is considered the mean value of each iteration. The cross validation process ensures that the training of the model does not lean heavily into one-dimensional learning of the dataset, which causes inaccuracy in new testing subsets.

7. Results

By configuring and experimenting with different parameter settings, the performance of each trained model is assessed in terms of learning time and classification efficiency. The accuracy is calculated for the aforementioned algorithms—Random Forest, Support Vector Machine and Multilayer Perceptron—and is defined as the percentage of traces each algorithm can correctly identify as bit '0' or bit '1'. Additionally, a 5-fold validation method was adopted for our experiments.

7.1. Random Forest Analysis

For the Random Forest (RF) Classifier, the two important parameters that can be tuned to review the accuracy of the model is the tree depth and the total number of trees in the forest. As can be shown by Figure 6, with Tree Depth value less than 10 there is a slight drop in accuracy in retrieving the key bit in the unprotected implementation, and a maximum accuracy of 94.1%. Regarding the protected implementation, a steady behavior can be observed by tuning tree depth, with a maximum accuracy of 61.5%.

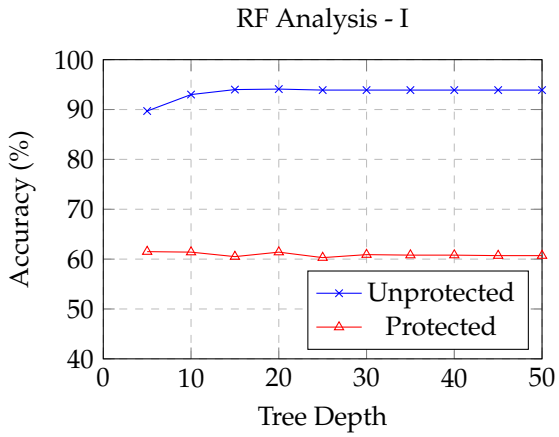

Figure 6. Analysis of Tree Depth values on accuracy for RF model.

The impact the number of trees has on the accuracy of the Random Forest model can be seen in Figure 7. There appears to be a consistent performance of the model regardless of the number of trees in the forest, again with a minor dip when the number of trees is 10 in the unprotected kernel of our implementation. The maximum accuracy recovered was 94.1%. Similar observations can be made for the protected implementation of our multiplier, maintaining a steady accuracy in regards to number of trees, with a maximum value of 62%.

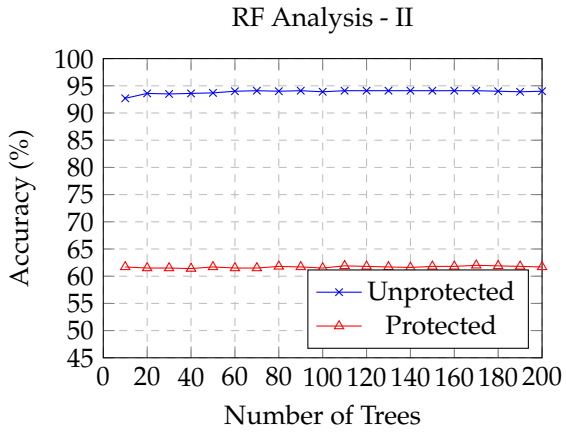

Figure 7. Analysis of number of Trees value on accuracy for RF model.

7.2. Support Vector Machine Analysis

In assessing the Support Vector Machine algorithm, there are several hyper-parameters that need to be fine-tuned beforehand. Using at its core several nonlinear kernel functions, we tested the impact the gamma value has on the Support Vector Machine model. Typically, a low gamma value indicates low bias and high variance. Figure 8 presents the accuracy for each of the chosen kernel functions: Radial Basis Function (RBF), Sigmoid and Polynomial on the unprotected version of our multiplier. The Polynomial kernel seems to be unaffected by the various gamma values, with a steady performance and a maximum accuracy of 89.1% for gamma = 1. For the RBF kernel, low gamma values greatly contribute to a higher accuracy, reaching 97.2% for gamma = 1 and dropping under 70% for gamma values greater than 40. The sigmoid kernel has a consistently poor performance across the range of values, and is unable to recover the key bits efficiently even for the unprotected version.

Measuring the performance of the SVM model after the countermeasures proposed in Section 4 results in accuracies for the different kernel functions as presented in Figure 9. The best performer, though by a small margin and only on low gamma values, is the Polynomial kernel, with a maximum accuracy of 64.6% for gamma = 1. For higher gamma

values, every kernel has a consistent accuracy around 60%, with the Sigmoid kernel again being unable to achieve a good accuracy score.

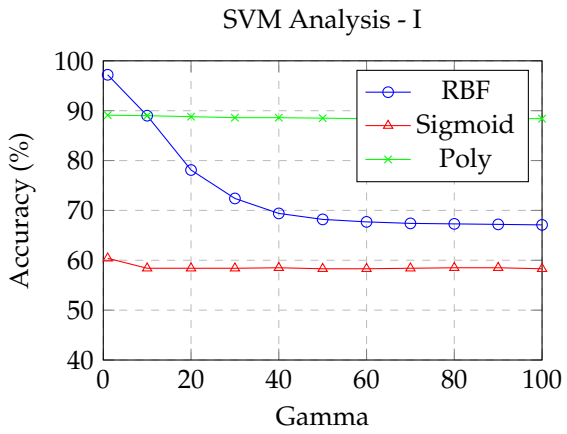

Figure 8. Analysis of SVM kernel accuracy for Unprotected implementation.

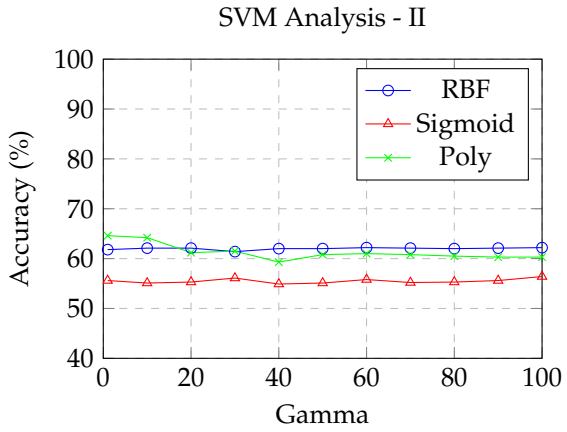

Figure 9. Analysis of SVM kernel accuracy for Protected implementation.

Typically, in an attempt to lower the classification error, the C parameter can be tuned, effectively controlling the rate by which the SVM optimization can avoid misclassifying each training example. This is achieved through the modification of the margin hyperplane, where for large values of C the optimizer chooses a smaller hyperplane, thus increasing the odds of correct classification. Conversely, choosing a small C value will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points. Figure 10 shows the retrieved accuracy for different values of C using the RBF kernel.

By trying to achieve greater accuracy results only for the protected version of the SM implementation and as described already in Section 6.3, the PCA feature-extraction technique was utilized. The number of features selected was $n_{features} = 2$, in order to achieve the most efficient trade-off between training time and accuracy. Presented in Figure 11 are the percentage gains in bit-classification accuracy that the PCA approach provides, as compared with the other feature sets on the SVM algorithm, since it has been proven to be the greatest beneficiary of the PCA technique.

The main outcome is that the PCA technique offers minimal gains to the classification accuracy of the SVM algorithm. It can be noted that the Sigmoid kernel is the one with the greatest increase in percentage accuracy for the protected implementation, with a maximum gain of 8.6%. These gains in no case can be considered high enough to lead to an eventual successful bit recovery attempt in a consistent manner. This small improvement in accuracy with the PCA technique can be explained by the fact that the original captured traces innately contain a low amount of noise, due to the low-noise capabilities of the

SAKURA-X platform. The effect of PCA as a noise-reduction method is thus significantly reduced, which leads to a performance similar to the non-PCA approach.

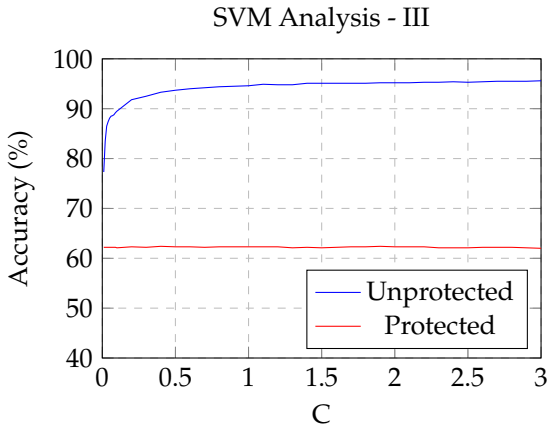

Figure 10. Impact on accuracy of C value for SVM model.

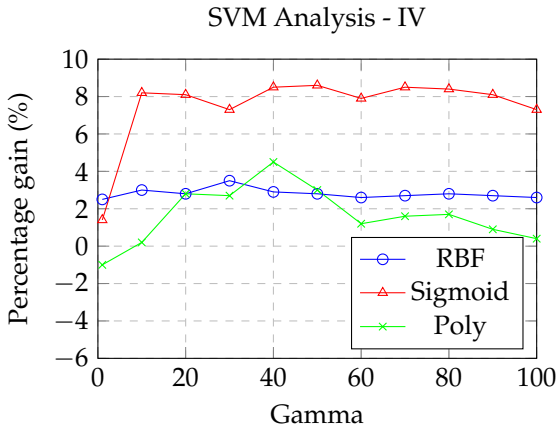

Figure 11. PCA comparison on SVM with $n\_features = 2$.

7.3. Multilayer Perceptron Analysis

During the training of the Multilayer Perceptron model, the two parameters that were analyzed for their impact in classification accuracy were the learning rate and the batch size. The learning rate hyperparameter controls the amount by which the weight values should be altered during the weight-update process (backpropagation), after the optimization algorithm estimates the error gradient for the current state of the model. Small values of this parameter lead to higher training times, whereas high values might lead to an unstable training process. The batch size parameter defines the amount of samples the model should work through before updating its internal parameters. For our MLP model, one hidden layer was used with a size of 100, and it utilized the 'adam' solver for weight optimization.

Similarly with the previous machine learning models, during the evaluation of the MLP algorithm, we observe an above-average classification accuracy for the unprotected implementation, reaching levels of 97.7%, a percentage capable of inducing a successful key bit recovery. This is true for both parameter-tuning metrics, as for both the learning rate (Figure 12) and the batch size (Figure 13) this percentage is consistently at a high level. In contrast, the accuracy level achieved when evaluating the protected implementation could not exceed 63.7%, a percentage deemed incapable of leading to a key bit recovery.

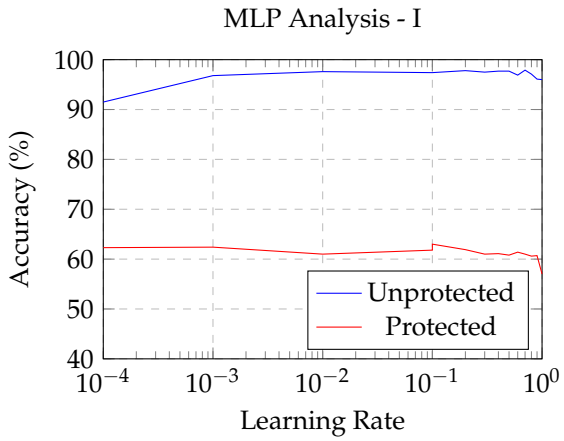

Figure 12. Impact on accuracy of Learning Rate on MLP model.

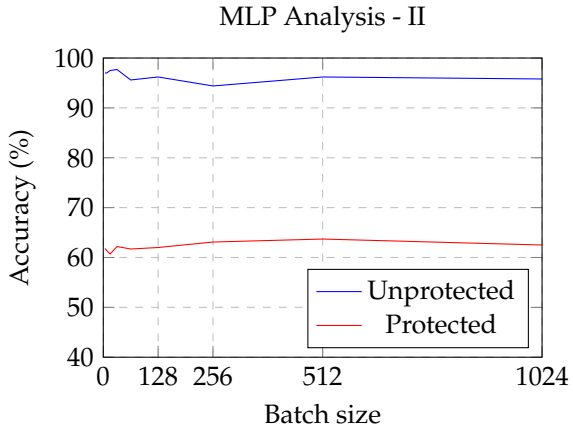

Figure 13. Impact on accuracy of Batch size on MLP model.

7.4. Comparison with Other Works

As far as the authors are aware, there are no relevant ECC-based ML attacks to make a solid comparison with in regards to the protection techniques used in our implementation.

For the unprotected implementation, the most relevant work is [11,52], where the authors have applied an SVM model on a 4-bit implementation of ECC leakage traces. Thus, in Table 3, a comparison of the maximum classification accuracy achieved by each unprotected implementation is presented. All implementations achieve high accuracy for the available algorithms. For reference, the maximum accuracy of the proposed protected implementation is also shown, denoting the clear improvements in bit-classification-attempt resistance.

Table 3. Maximum Accuracy comparison between implementations.

	SVM	**RF**	**MLP**
Mukhtar et al. [11]	87%	89.4%	82.6%
Saeedi et al. [52]	96%	-	-
Proposed Unprotected	97.2%	94.1%	97.7%
Proposed Protected	64.6%	62%	63.7%

8. Conclusions

In this paper, the current landscape on simple and advanced SCAs for EC scalar multipliers was described and the potential of profiling attacks, such as TA and ML-based SCAs, was highlighted. Moreover, a fine-grained GF unification stages model and overall design approach was proposed, and relevant leakage models were presented. The proposal

relies on the MPL algorithm's ability to perform all point operations in parallel within a computation round. This led to a decomposition of the point operations in their underlying finite field operations and the redistribution of those operations in parallel stages that concurrently implement finite field operations for both point operations on an MPL round. The approach is further enhanced by the injection, in some parallel stage, of finite field operations that perform multiplicative randomization. This process extends the projective coordinate randomization countermeasure found in relevant literature, showcasing the transparent migration of randomization inside the parallel stages of the proposed fine-grained GF unification stages approach. This randomization is performed for each MPL round with a different random value per round. Furthermore, in the paper, the proposed fine-grained GF unification stages approach was practically evaluated against ML-based SCAs that utilize three ML models (Random Forest, SVM and MLP). A detailed roadmap on how to mount such attacks on MPL-based SM was made using an existing implementation that follows the proposed fine-grained GF unification stages approach. Actual ML-based SCAs were mounted onto two SM implementations, an unprotected one and a protected one (that latter using our proposed approach with projective coordinate re-randomization).

Analyzing the data resulting from the implemented attacks on the fine-grained GF unification stages based SM implementations, using different and popular SCA applications with simple Machine Learning algorithms (that do not require a significant amount of leakage traces, as traditionally needed in deep learning approaches), we can safely conclude the effectiveness of the applied countermeasures. For all the tested models, the unprotected version of the scalar multiplier did not manage to conceal important information from the attempts to recover the key bits, reaching up to 95% accuracy in some cases. This means that the provided leakage model reveals a significant amount of secret scalar information to an attacker, regardless of the GF unification in each round, and is not appropriate for ML-based SCA resistance. On the contrary, after applying the re-randomization countermeasure, the ML models were unable to achieve an accuracy that can be considered satisfactory in order to reveal the secret key bits. Thus, the masking mechanism of random operation within the MPL round's parallel stages, which can be easily included in the proposed design approach, seems to be potent enough to thwart several different ML-based SCAs, such as the ones described in this paper.

Author Contributions: The authors of the paper have contributed to the work as follows: conceptualization, C.D. and A.P.F.; methodology, C.D. and A.P.F.; software, C.D.; validation, C.D. and A.P.F.; formal analysis, C.D. and A.P.F.; investigation, C.D. and A.P.F.; resources, A.P.F. and O.K.; data curation, C.D. and A.P.F.; writing—original draft preparation, C.D. and A.P.F.; writing—review and editing, O.K.; visualization, O.K.; supervision, A.P.F. and O.K.; project administration, O.K.; funding acquisition, A.P.F. and O.K. All authors have read and agreed to the published version of the manuscript.

Funding: This paper's work has received funding from the European Union's Horizon 2020 research and innovation program CONCORDIA under grant agreement No. 830927. Additionally, Apostolos Fournaris' work has also received funding from the European Union's Horizon 2020 research and innovation programme CPSoSaware under grant agreement No. 871738.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data underlying this article will be shared on reasonable request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Fan, J.; Verbauwhede, I. An Updated Survey on Secure ECC Implementations: Attacks, Countermeasures and Cost. In *Cryptography and Security: From Theory to Applications*; Naccache, D., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2012; Volume 6805, pp. 265–282.
2. Houssain, H.; Badra, M.; France, C.; Al-somani, T.F.; Member, S. Comparative Study of Elliptic Curve Cryptography Hardware Implementations in Wireless Sensor Networks. *Int. J. RFID Secur. Cryptogr.* **2012**, *1*, 67–73. [CrossRef]

3. Verri Lucca, A.; Mariano Sborz, G.A.; Leithardt, V.R.Q.; Beko, M.; Albenes Zeferino, C.; Parreira, W.D. A Review of Techniques for Implementing Elliptic Curve Point Multiplication on Hardware. *J. Sens. Actuator Netw.* **2021**, *10*, 3. [CrossRef]
4. Hankerson, D.; Menezes, A.J.; Vanstone, S. *Guide to Elliptic Curve Cryptography*; Springer: New York, NY, USA, 2003.
5. Marzouqi, H.; Al-Qutayri, M.; Salah, K. Review of Elliptic Curve Cryptography processor designs. *Microprocess. Microsyst.* **2015**, *39*, 97–112. [CrossRef]
6. Fan, J.; Guo, X.; De Mulder, E.; Schaumont, P.; Preneel, B.; Verbauwhede, I. State-of-the-art of secure ECC implementations: A survey on known side-channel attacks and countermeasures. In Proceedings of the 2010 IEEE International Symposium on Hardware-Oriented Security and Trust, Anaheim, CA, USA, 13–14 June 2010; pp. 76–87. [CrossRef]
7. 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: Cham, Switzerland, 2017; pp. 49–76.
8. Chari, S.; Rao, J.R.; Rohatgi, P. Template Attacks. *Lect. Notes Comput. Sci.* **2003**, *2523*, 13–28. [CrossRef]
9. Lerman, L.; Poussier, R.; Bontempi, G.; Markowitch, O.; Standaert, F.X. Template attacks vs. Machine learning revisited (and the curse of dimensionality in side-channel analysis). *Lect. Notes Comput. Sci.* **2015**, *9064*, 20–33. [CrossRef]
10. Gilmore, R.; Hanley, N.; O'Neill, M. Neural network based attack on a masked implementation of AES. In Proceedings of the 2015 IEEE International Symposium on Hardware-Oriented Security and Trust, HOST 2015, Washington, DC, USA, 5–7 May 2015; pp. 106–111. [CrossRef]
11. Mukhtar, N.; Mehrabi, M.; Kong, Y.; Anjum, A. Machine-Learning-Based Side-Channel Evaluation of Elliptic-Curve Cryptographic FPGA Processor. *Appl. Sci.* **2018**, *9*, 64. [CrossRef]
12. Bernstein, D.; Lange, T. Faster addition and doubling on elliptic curves. In *International Conference on the Theory and Application of Cryptology and Information Security*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 1–22.
13. Bernstein, D.; Lange, T.; Farashahi, R. Binary edwards curves. In *International Workshop on Cryptographic Hardware and Embedded Systems*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 244–265.
14. Mangard, S.; Oswald, E.; Popp, T. *Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security)*; Springer: New York, NY, USA, 2007.
15. Fournaris, A.P.; Koufopavlou, O. Affine Coordinate Binary Edwards Curve Scalar Multiplier with Side Channel Attack Resistance. In Proceedings of the 2015 Euromicro Conference on Digital System Design, Madeira, Portugal, 26–28 August 2015; pp. 431–437. [CrossRef]
16. Joye, M.; Yen, S.M. The Montgomery Powering Ladder. In *CHES '02: Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems*; Springer: London, UK, 2003; pp. 291–302.
17. Fournaris, A.P. Fault and Power Analysis Attack Protection Techniques for Standardized Public Key Cryptosystems. 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: Cham, Switzerland, 2017; pp. 93–105. [CrossRef]
18. Bauer, A.; Jaulmes, E.; Prouff, E.; Wild, J. Horizontal and Vertical Side-Channel Attacks against Secure RSA Implementations. In *Topics in Cryptology, CT-RSA 2013*; Dawson, E., Ed.; Springer: Berlin/Heidelberg, Germany, 2013; Volume 7779, pp. 1–17.
19. Bauer, A.; Jaulmes, E.; Prouff, E.; Wild, J. Horizontal Collision Correlation Attack on Elliptic Curves. In *Selected Areas in Cryptography—SAC 2013*; Lange, T., Lauter, K., Lisonk, P., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2014; Volume 8282, pp. 553–570.
20. Mukhtar, N.; Fournaris, A.P.; Khan, T.M.; Dimopoulos, C.; Kong, Y. Improved Hybrid Approach for Side-Channel Analysis using Efficient Convolutional Neural Network and Dimensionality Reduction. *IEEE Access* **2020**, *8*, 184298–184311. [CrossRef]
21. Fouque, P.A.; Valette, F. The Doubling Attack Why Upwards Is Better than Downwards. In *International Workshop on Cryptographic Hardware and Embedded Systems*; Walter, C., Koc, C., Paar, C., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2003; Volume 2779, pp. 269–280.
22. Yen, S.; Ko, L.; Moon, S.; Ha, J. Relative doubling attack against Montgomery Ladder. In *International Workshop on Cryptographic Hardware and Embedded Systems*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 117–128.
23. Yen, S.M.; Lien, W.C.; Moon, S.J.; Ha, J. Power Analysis by Exploiting Chosen Message and Internal Collisions—Vulnerability of Checking Mechanism for RSA-Decryption. In *International Workshop on Cryptographic Hardware and Embedded Systems*; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3715, pp. 183–195.
24. Kocher, P.; Jaffe, J.; Jun, B. Differential Power Analysis. In *Annual International Cryptology Conference*; Springer: Berlin/Heidelberg, Germany, 1999; pp. 388–397.
25. Koc, C.K. *Cryptographic Engineering*, 1st ed.; Springer: New York, NY, USA, 2008.
26. Amiel, F.; Feix, B.; Villegas, K. Power Analysis for Secret Recovering and Reverse Engineering of Public Key Algorithms. In *Selected Areas in Cryptography*; Adams, C., Miri, A., Wiener, M., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2007; Volume 4876, pp. 110–125.
27. Bogdanov, A.; Kizhvatov, I.; Pyshkin, A. Algebraic Methods in Side-Channel Collision Attacks and Practical Collision Detection. In *Progress in Cryptology—INDOCRYPT 2008*; Chowdhury, D., Rijmen, V., Das, A., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5365, pp. 251–265.
28. Moradi, A. Statistical Tools Flavor Side-Channel Collision Attacks. In *Advances in Cryptology—EUROCRYPT 2012*; Pointcheval, D., Johansson, T., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7237, pp. 428–445.

29. Feix, B.; Roussellet, M.; Venelli, A. Side-Channel Analysis on Blinded Regular Scalar Multiplications. In *Progress in Cryptology—INDOCRYPT 2014*; Meier, W., Mukhopadhyay, D., Eds.; Lecture Notes in Computer Science; Springer: New York, NY, USA, 2014; Volume 8885, pp. 3–20.
30. Clavier, C.; Feix, B.; Gagnerot, G.; Roussellet, M.; Verneuil, V. Horizontal Correlation Analysis on Exponentiation. In *Information and Communications Security*; Soriano, M., Qing, S., López, J., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6476, pp. 46–61.
31. Walter, C. Sliding Windows Succumbs to Big Mac Attack. In *Cryptographic Hardware and Embedded Systems—CHES 2001*; Ko, E., Naccache, D., Paar, C., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2001; Volume 2162, pp. 286–299.
32. Doget, J.; Prouff, E.; Rivain, M.; Standaert, F.X. Univariate side channel attacks and leakage modeling. *J. Cryptogr. Eng.* **2011**, *1*, 123–144. [CrossRef]
33. Coron, J.S. Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems. In *International Workshop on Cryptographic Hardware and Embedded Systems*; Springer: Berlin/Heidelberg, Germany, 1999; pp. 292–302.
34. Fournaris, A.P.; Dimopoulos, C.; Moschos, A.; Koufopavlou, O. Design and leakage assessment of side channel attack resistant binary edwards Elliptic Curve digital signature algorithm architectures. *Microprocess. Microsyst.* **2019**, *64*, 73–87. [CrossRef]
35. Whitnall, C.; Oswald, E.; Standaert, F.X. The myth of generic DPA and the magic of learning. In *Cryptographers' Track at the RSA Conference*; Springer: Berlin/Heidelberg, Germany, 2014; Volume 8366, pp. 183–205. [CrossRef]
36. Medwed, M.; Oswald, E. Template attacks on ECDSA. In *International Workshop on Information Security Applications*; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5379, pp. 14–27.
37. Archambeau, C.; Peeters, E.; Standaert, F.X.; Quisquater, J.J. Template attacks in principal subspaces. In *International Workshop on Cryptographic Hardware and Embedded Systems*; Springer: Berlin/Heidelberg, Germany, 2006; Volume 4249, pp. 1–14. [CrossRef]
38. Batina, L.; Chmielewski, Ł.; Papachristodoulou, L.; Schwabe, P.; Tunstall, M. Online template attacks. In *International Conference on Cryptology in India*; Springer: Berlin/Heidelberg, Germany, 2014; Volume 8885, pp. 21–36.
39. Papachristodoulou, L.; Fournaris, A.P.; Papagiannopoulos, K.; Batina, L. Practical Evaluation of Protected Residue Number System Scalar Multiplication. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2018**, *2019*, 259–282. [CrossRef]
40. Shi, Y.; Sagduyu, Y.; Grushin, A. How to steal a machine learning classifier with deep learning. In Proceedings of the 2017 IEEE International Symposium on Technologies for Homeland Security (HST), Waltham, MA, USA, 25–26 April 2017; pp. 1–5. [CrossRef]
41. Vishwakarma, R.; Jain, A.K. A Honeypot with Machine Learning based Detection Framework for defending IoT based Botnet DDoS Attacks. In Proceedings of the 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 23–25 April 2019; pp. 1019–1024. [CrossRef]
42. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
43. Cortes, C.; Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
44. Rosenblatt, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychol. Rev.* **1958**, *65*, 386–408. [CrossRef] [PubMed]
45. SAKURA-X. Available online: http://satoh.cs.uec.ac.jp/SAKURA/ (accessed on 2 June 2021).
46. Moschos, A.; Fournaris, A.; Koufopavlou, O. A Flexible Leakage Trace Collection Setup for Arbitrary Cryptographic IP Cores. In Proceedings of the IEEE International Symposium on Hardware Oriented Security and Trust (HOST), Washington, DC, USA, 30 April–4 May 2018.
47. Scikit-Learn Library. Available online: https://scikit-learn.org/stable/ (accessed on 2 June 2021).
48. Prouff, E.; Strullu, R.; Benadjila, R.; Cagli, E.; Canovas, C. Study of Deep Learning Techniques for Side-Channel Analysis and Introduction to ASCAD Database. *IACR Cryptol. ePrint Arch.* **2018**, *2018*, 53.
49. 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]
50. Mukhtar, N.; Kong, Y. Hyper-parameter optimization for machine-learning based electromagnetic side-channel analysis. In Proceedings of the 26th International Conference on Systems Engineering, ICSEng 2018, Sydney, Australia, 18–20 December 2018; pp. 1–7. [CrossRef]
51. Batina, L.; Hogenboom, J.; Van Woudenberg, J.G. Getting more from PCA: First results of using principal component analysis for extensive power analysis. In *Cryptographers' Track at the RSA Conference*; Springer: Berlin/Heidelberg, Germany, 2012. [CrossRef]
52. Saeedi, E.; Hossain, M.S.; Kong, Y. Side channel analysis of an elliptic curve crypto-system based on multi-class classification. In Proceedings of the 2015 6th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Dallas, TX, USA, 13–15 July 2015; pp. 1–7. [CrossRef]