Template Attacks against ECC : practical implementation against Curve25519

Antoine Loiseau, Maxime Lecomte, Jacques J A Fournier

Template Attacks against ECC : practical implementation against Curve25519

Antoine Loiseau
Univ. Grenoble Alpes, CEA, Leti, MINATEC Campus, F-38054 Grenoble
Mines Saint Etienne, Gardanne France
Grenoble, France
antoine.loiseau@cea.fr

Maxime Lecomte
Univ. Grenoble Alpes, CEA, Leti, MINATEC Campus, F-38054 Grenoble
Grenoble, France
maxime.lecomte@cea.fr

Jacques J. A. Fournier
Univ. Grenoble Alpes, CEA, Leti, MINATEC Campus, F-38054 Grenoble
Grenoble, France
jacques.fournier@cea.fr

***Abstract*—This paper introduces a new profiling attack that targets elliptic curves-based cryptographic implementations. This attack exploits leakages from the conditional swap operation used in implementations using the Montgomery Ladder as a scalar multiplication method for calculating kP in constant time. In addition, our attack requires only one attack trace. This paper shows how the attack is performed on the mbedTLS Curve25519 function and why conventional coordinates randomization countermeasures do not prevent this type of attack. Then, a new countermeasure that is efficient against the presented attack will be proposed and tested. This work was carried out on the implementation of mbedTLS from Curve25519.**

***Index Terms*—Template Attacks, PCA, LDA, Asymmetric Cryptography, Curve25519, ECC, Constant Time, mbedTLS**

I. Introduction

The increasing number of deployed IoT devices brings new challenges for elliptic curves cryptography, mainly in terms of security of their implementation. Among physical attacks, we can discern side-channel attacks (SCA). The purpose of side-channel attacks is to extract information from physical measurements; Computation time [1], power consumption [2], electro-magnetic emanation (EM) [3]. Initially introduced by Coron [4], several works have developed side channel attacks against elliptic curve cryptography (ECC). From these attacks, several countermeasures have been developed to protect ECC implementations.

However, the constant improvements of techniques used for side channel attacks require ECC implementations to be improved and secured. This can be done by the development of new countermeasures.

Templates Attacks (TA) introduced by Chari *et al.* [5] allow targeting several cryptographic algorithms and their implementations. Whatever the algorithm targeted, the method remains the same: a first *profiling phase* allows building leakage templates and a second *attack phase* uses them to infer a secret key.

Medwed and Oswald [6] introduced Templates Attack for elliptic curves cryptography and the ECDSA signature protocol. In the first phase, the authors performed a DPA on the scalar multiplication computation for the selection of points of interest during the *profiling phase*. This limitation may lead to the attack being impossible in a real implementation of an ECDSA signature due to the ephemeral scalar used during the scalar multiplication step.

Batina *et al.* [7] proposed an Online Template Attack and led to a very strong attack model which brings the attack close to real cases. However, the attacker must be able to submit specific base points to the target device for the *profiling phase*. This constraint cannot be achieved in some cryptographic protocols such as signature protocols that use a fixed base point.

Our attack target the conditional swap operation of a Montgomery Ladder implementation. Nascimento *et al.* [8] performed a similar attack on the same operation. Nascimento and Chmielewski [9] improved the attack via an horizontal approach. In this paper, we use dimension reduction such as Principal Components Analysis (PCA) and Linear Discriminant Analysis (LDA) to improve the succes rate of our attack. In addition, we propose a completely online attack like the one proposed by Batina *et al.* [7] but the attack model is improved for a realistic application. Moreover, we propose a new tested countermeasure to protect implementations against this attack.

In this paper, we propose to study the implementation of the Curve25519 elliptic curve. This curve is used for key exchange protocol or signature protocol like EdDSA. We focus on the implementation provided by mbedTLS [10] which implements the TLS standard [11]. This library is open source and can be easily used on any general purpose microcontroller. However, the weakness used by our attack is not inherent to mbedTLS implementation. This attack can be used against almost any Curve25519 implementation and more precisely any constant time implementation of the Montgomery Ladder. Moreover, it is well-known that a randomization countermeasure [4] can protect an ECC implementation against template attacks. However, our results demonstrate that this countermeasure is useless against this kind of template attack. The TA of Nascimento *et al.* [8] bypasses the coordinates randomization countermeasure because this attack is based on address register. In our case, we perform TA on the entire conditional swap operation and the exploited leakage is located in the conditional bit. Moreover, we run our attack on a 32 bits ARM-based core (Cortex M4 at 168 MHz) where Nascimento *et al.* [8] uses a 8 bits AVR-based core (ATmega328P at 7 MHz).

978-1-7281-7405-1/20/$31.00

Paper Organization

The paper is organized as follows. First of all, Section II recalls some properties of ECC and their implementations. Section III exposes the main strategy used by our template attack and our attack against Curve25519 implementation of mbedTLS. Section IV presents the results of the attack. Section V analyses the efficiency of classical countermeasures against our attack. Finally, a new countermeasure is proposed to secure ECC implementation against the attack in Section VI.

II. Elliptic Curves implementations

Miller [12] and Koblitz [13] introduced elliptic curves based cryptography. Both authors use the Weierstrass model defined on a finite field of prime characteristic p or an extension of the field $\mathbb{F}_2$. This model is based on two short equations of Weierstrass's general model:

$$y^2 = x^3 + ax + b \tag{1}$$

$$y^2 + xy = x^3 + ax^2 + b \tag{2}$$

With equation 1 when the curve is defined on $\mathbb{F}_p$ and equation 2 in the case of extensions of $\mathbb{F}_2$. The set of points of the elliptic curve form an abelian group. A point to infinity is added to the set of affine points of the curve in order to form a group for the Weierstrass model. This point is the neutral element of the group. The fundamental operation that can be performed on the curve is the point addition: $P+Q$, where P and Q are points curve. Then, the operation kP is built which consists in the addition of the point P, k times. This operation is at the heart of many cryptographic protocols.

In cryptographic applications, Miller and Koblitz use the Elliptic Curve Discrete Logarithm Problem (ECDLP) to build cryptographic primitives. In this way, the main operation of ECC based cryptographic protocols is the scalar multiplication kP, where the scalar k and the point kP define respectively the private and public key for ECC. Standards define the way to use ECC based cryptography: FIPS 186 [14] for signature protocol or NIST SP 800-56r2 [15] for key exchange. Both standards are largely deployed in the industry.

The main step within these protocols is the kP operation also called scalar multiplication. In order to perform the latter, specific algorithms are necessary. The simplest algorithm is the *Double & Add* method which is close to the structure of the exponentiation algorithm *Square & Multiply*. However, computation time-efficient algorithms should be used for cryptographic applications. In addition to that, *Double & Add* algorithm is weak against SCA and must be modified to add countermeasures. The first modified scalar multiplication algorithm is the *Double & Add Always* which is sensitive to side-channel attacks (eg. register address based) and fault attacks.

The Montgomery Ladder [16] algorithm is one of many solutions for SCA resistant implementations. At the beginning, the purpose of this scalar multiplication method was to speed up of the Pollard method and ECM for factorization. Joye and Yen [17] give a generic form of this method called Montgomery Powering Ladder. In the following, we refer to the Montgomery Powering Ladder as the Montgomery Ladder. The Algorithm 1 gives details on this scalar multiplication method.

Algorithm 1 Montgomery Ladder

Require: $P, k = (k_{t-1}, ..., k_0)_2$
1: $R_0 \leftarrow \mathcal{O}$
2: $R_1 \leftarrow P$
3: **for** $j = t-1$ **to** 0 **do**
4: **if** $k_j = 0$ **then**
5: $R_1 \leftarrow R_0 + R_1$
6: $R_0 \leftarrow 2R_0$
7: **else**
8: $R_0 \leftarrow R_0 + R_1$
9: $R_1 \leftarrow 2R_1$
10: **end if**
11: **end for**
12: **return** $R_0 = kP, R_1 = kP + P$

All scalar multiplication algorithms split the kP computation into a series of point addition and doubling. Both operations depend on the chosen point representation. The reader may refer to [18] for details on point representation. The choice of the point representation does not change our attack because addition and doubling operations are not targeted.

A. Curve25519

The Weierstrass model is mainly used in the NIST standard FIPS 186 [14] which defines a set of elliptic curves over $\mathbb{F}_p$ and $\mathbb{F}_{2^n}$. However, this standard is old and several improvements have been made on the efficiency of elliptic curves arithmetic and the security against physical attacks. Edwards' work [19] introduces a new model of elliptic curve. The democratization of the Edwards model is mainly due to the work of Bernstein *et al.* [20]–[23]. This model allows an efficient arithmetic and high intrinsic security against physical attacks thanks to the complete group law and the ability to perform a Montgomery Ladder efficiently. This elliptic curve form is defined over large prime fields.

$$x^2 + y^2 = 1 + dxy \tag{3}$$

The different implementations which use the prime field model define these curves in different ways. Usually, the Montgomery form [16] of equation 3 is used in key exchange algorithms. Additionally, the Twisted Edwards Form [24] is used for signature protocols. The mbedTLS library [10] has been chosen for the latter.

The RFC 7748 [25] specifies the key exchange protocol over Curve25519. This standrard uses Bernstein' work [20] which defines a new Montgomery Curve [16]. The Curve25519 is defined over the finite field of large characteristic $p = 2^{255-19}$. Moreover, equation 4 defines the group of points of Curve25519.

$$y = x^3 + 48662x^2 + x \tag{4}$$

The advantage given by this curve is the fast operations on x-coordinate point due to the order of the curve which can be divided by 4. As a matter of fact, we can construct a very fast and secure scalar multiplication with the Montgomery Ladder [17].

In our contribution, we target the mbedTLS implementation of the RFC 7748 [25]. However, our attack is not specific to only this implementation. The attack is independent to the specific case of Curve25519. We chose this implementation and curve with the goal to cover commercial implementations of the Montgomery Ladder.

B. Montgomery Ladder implementations

The Montgomery Ladder algorithm [17] is efficient for computing a scalar multiplication kP. The algorithm 1 presents the details of the Montgomery Ladder. This algorithm provides security for scalar multiplication against SPA thanks to the homogeneous computation whatever the key bit is. It is different from the *Double & Add* algorithm where the operations performed at each step depend on the key bit; in the case where the current key bit is 1, an Add operation is added.

Nevertheless, the algorithm 1 still suffers from other vulnerabilities. The conditional branch `if` can lead to a difference on the time computation of the kP operation. This difference depends on the key bit. Timing attacks can be carried out. In order to protect the Montgomery Ladder against this kind of attack, the `if` branch is removed and a conditional swap operation is added such as `cswap`, which swaps R_0 and R_1 for a given conditional bit k_j. This operation must be performed in constant time, *i.e.* `cswap` performs the same operations for the key bit 0 or 1 and any inputs R_0 and R_1. Finally, the algorithm 2 gives the constant time Montgomery Ladder.

Algorithm 2 Constant Time Montgomery Ladder

Require: $P, k = (k_{t-1}, ..., k_0)_2$
1: $R_0 \leftarrow \mathcal{O}$
2: $R_1 \leftarrow P$
3: **for** $j = t - 1$ **to** 0 **do**
4: `cswap`(R_1, R_0, k_j)
5: $R_1 \leftarrow R_0 + R_1$
6: $R_0 \leftarrow 2R_0$
7: `cswap`(R_1, R_0, k_j)
8: **end for**
9: **return** $R_0 = kP, R_1 = kP + P$

The `cswap` shall perform a swap between R_0 and R_1 if the conditional bit k_j is 1, otherwise `cswap` operation performs the same instructions on R_0 and R_1 without changing them. A point on an elliptic curve is a set of coordinates which are vectors of n registers or words. The size of the words depends on the size of the data path of the processor core that executes the algorithm. In order to swap two coordinates of a point with a conditional bit b, one of the following equations can be applied n times on each words W_1 and W_0 composing a coordinate:

$$W_1 = W_1 + b(W_0 - W_1), \quad W_0 = W_0 - b(W_0 - W_1) \tag{5}$$

$$W_1 = W_1 \oplus b \wedge (W_0 \oplus W_1), \quad W_0 = W_0 \oplus b \wedge (W_0 \oplus W_1) \tag{6}$$

In order to use the equation 6, the input conditional bit b shall be set to 0 if null or $0xF...F$ if set. This transformation can be performed with $b \leftarrow ((b-1) >> (w-1)) - 1$, where w is the data width of the considering architecture. The Nascimento *et al.* [8] TA targets the equation 6 to perform a `cswap`.

The mbedTLS library [10] uses a different equation equivalent to equation 5. The idea is to perform a multiplication by 1 and 0 independently of the input bit b.

$$W_{1,i} = (b-1)W_{1,i} + bW_{2,i}, \quad W_{2,i} = (b-1)W_{2,i} + bW_{1,i} \tag{7}$$

The point representation of R_0 and R_1 is the projective x-coordinate where coordinates X and Z represent a point. Each coordinate is a series of n registers. In our case, we use a general purpose off-the-self device which embeds a 32 bit Cortex M4 architecture, thus the width of registers is 32 bits. In addition to that, we target the Curve25519 implementation of mbedTLS [10], in this way $n = 8$. The `mbedtls_mpi` structure gives the integer representation of mbedTLS. In this structure, we have the sign of the integer (type `int`), the number of words also called limbs (type `size_t`) and a pointer the integer data (type `mbedtls_mpi_uint` equivalent to `uint32_t` in our case). Each coordinate of a point is a mbedTLS integer. the conditional swap shall applied to each coordinate of R_0 and R_1. Two `mbedtls_mpi` structures have to be exchanged in order to perform this operation in constant time. The figure 1 shows the details of the conditional swap in mbedTLS.

The figure 1 uses equations 7 to perform the conditional swap. In the case of kP computation over the Curve25519, the `for` loop is performed 8 times to swap data between X and Y. Lines 17 and 18 swap the sign of X and Y and lines 23, 24 swap words of X and Y. Each multiplication by `swap` (the conditional bit) performs a multiplication by 1 and by 0. In the case that `swap` is null, the Hamming Weight of `Y->p[i] * swap` is 0; if `swap` is set, the Hamming Weight of the result is different from 0. Therefore, this difference of Hamming Weight in a Template Attack (TA) can be exploited.

III. Template Attacks

Chari *et al.* introduced Template Attacks (TA) in [5]. The attack seeks to characterize the leakage of a circuit to create a template which is applied to another circuit to infer information about the secret key manipulated in the second cicuit. The attack can be split into two steps:

1) The first step is to build the template. For each possible key we construct a separate template. The leakage is

```
1  int  mbedtls_mpi_safe_cond_swap( mbedtls_mpi *X,
       mbedtls_mpi *Y, mbedtls_mpi_uint swap )
2  {
3      int ret, s;
4      size_t i;
5      mbedtls_mpi_uint tmp, r[3];
6
7      if( X == Y )
8          return( 0 );
9
10     /* make sure swap is 0 or 1 in a time-
           constant manner */
11     swap = (swap | (unsigned char)-swap) >> 7;
12
13     MBEDTLS_MPI_CHK( mbedtls_mpi_grow( X, Y->n )
           );
14     MBEDTLS_MPI_CHK( mbedtls_mpi_grow( Y, X->n )
           );
15
16     s = X->s;
17     X->s = X->s * ( 1 - swap ) + Y->s * swap;
18     Y->s = Y->s * ( 1 - swap ) +    s * swap;
19
20     for( i = 0; i < X->n; i++ )
21     {
22         tmp = X->p[i];
23         X->p[i] = X->p[i] * ( 1 - swap ) + Y->p[
               i] * swap;
24         Y->p[i] = Y->p[i] * ( 1 - swap ) +
               tmp * swap;
25     }
26
27 cleanup:
28     return( ret );
29 }
```

Fig. 1. mbedTLS function to perform a conditional swap between X and Y.

modelled by a multivariate normal distribution. For a set of measurment traces μ_i and for the key k, construct the mean trace $\bar{\mu}_k$ and its covariance S_k are constructed. The template for the key k is composed of $\bar{\mu}_k$ and S_k.

2) The second step consists in applying the template to a set of attack traces by computing the probability density function (pdf). The pdf for $p_{k,x}$ a vector x is given by:

$$p_{k,x} = \frac{1}{\sqrt{(2\pi)^m |S_k|}} \exp\left(-\frac{1}{2}(x-\bar{\mu}_k)' S_k^{-1}(x-\bar{\mu}_k)\right)$$

In order to combine results for different attack traces, the following formula is applied onto the pdf: $\log P_k = \sum_x \log p_{k,x}$. The highest $logP_k$ should correspond to the correct key.

A. Our template attack

In our case, we target the `cswap` operation during the computation of the Montgomery Ladder (algorithm 2). We aim to recover the secret key bits used during the `cswap` operation.

Different scenarios can be considered for different protocols:

- **Scenario 1 (S1):** The attacker wants to hack an ECDSA [14] signature generation. In this case, the attacker has only one Montgomery Ladder computation trace to apply the template. At each signature generation, the device picks a random k to compute kG. The secret key used to compute the signature can be found from the knowledge of k. From this observation, the attacker has only two traces for `cswap` computations for each key bit k_j which correspond to the `cswap` computations of each step of the algorithm 2. Therefore, the number of attack traces available is 2.
- **Scenario 2 (S2):** The attacker wants to hack an ECDH [15], [26] session with ephemeral keys. In this case, two computations of algorithm 2 is performed with the same private key k but with different base points. We have 4 `cswap` computations for each bit k_j of the private key. So, the number of attack traces available is 4.

However, the real number of `cswap` operations at each step of algorithm 1 depends on the chosen point representation. Algorithm 2 uses two points R_0 and R_1 and if projective coordinates are chosen, then three coordinates $(X : Y : Z)$ represent each point. 6 `cswap` computations are thus necessary for each bit k_j.

The following discussion is focused on the mbedTLS implementation which uses projective coordinates without y coordinate. In this case, there are 4 `cswap` operations that uses equation 7 for each secret bit k_j. In the scenario S1 using the mbedTLS implementation we can reach 4 attack traces; 8 attack traces in the scenario S2. In the following, the worst case scenario will be studied: S1. With more attack traces, the scenario S2 shall have better success rate of secret bit recovery than S1.

The attacker model follows the assumptions :

1) The attacker has a similar device to the targeted device where he can perform any number of kG computations for a chosen k.
2) The attacker cannot chose the base point. To attack ECDSA, this assumption is important.
3) The attacker can measure the appropriate number of attacks traces as required by the scenarios S1 or S2.

The mbedTLS implementation of Curve25519 could use the point randomization as countermeasure against SCA. We assume that the countermeasure is enabled on both devices (template building device and attacked device). As explained in [6], [7], [27], this countermeasure should protect ECC implementation against TA which target data manipulations.

B. Dimensional Reduction

To perform an efficient TA, the dimensions of the problem have to be reduced. The reduction allows reducing the complexity of the template computation and avoiding the so called curse of dimentionnality. The first idea consists in selecting some points of interest (PoIs) with the maximal leakage information. A school book method for this is to use the T-test to find some PoIs. We have only two classes of hypothesis key: $k_j = 0$ and $k_j = 1$. With a T-test and a set of labeling EM emanation traces of a `cswap` operation with random inputs, we can extract some PoIs from local peaks of the T-test curve.

This approach is naive and some advanced techniques of dimension reduction can be used. Some techniques of components analysis like Principal Component Analysis (PCA) or Linear Discriminant Analysis (LDA) can be used to extract information from a set of labelling consumption curves. This idea has been used in many papers such as [28]–[30] and for non-profiled attacks against ECC in [31].

C. Principal Components Analysis

The Principal Component Analysis (PCA) [32] reduces the space of traces to a subspace, where the first components of this subspace maximize the variance between classes. In order to perform this reduction, we have to consider the following empirical covariance matrix:

$$\bar{S} = \sum_{k=1}^{|\mathcal{K}|} N_t(\bar{\mu}_k - \hat{\mu})(\bar{\mu}_k - \hat{\mu})^T \tag{8}$$

Where $\mathcal{K}$ is the set of possible keys, in our case $\mathcal{K} = \{0, 1\}$ and N_t is the number of traces in each class. Moreover, $\hat{\mu}$ is the average of mean traces $(\bar{\mu}_k)_{k=1,...,N}$. Let r be the rank of $\bar{S}$ and $\lambda_1, ..., \lambda_r$ the eigenvalues. We have the corresponding eigenvectors $\alpha_1, ..., \alpha_r$. If the α_i vectors are listed in the decreasing order of the values λ_i, it can be shown that each eigenvalue λ_i equals the variance of the data projected onto principal component α_i. Using that, the data set can be projected on the subspace composed of the n first α_i which leads to reducing the dimension and maximizing the variance of the data set. Let us consider the projective matrix W composed the first n values of α_i. In this case, the template can be built as following:

$$\bar{x}_k = W^T \bar{\mu}_k, \qquad S_k = W^T S_k W$$

Where S_k is the covariance matrix of traces with the label key k.

D. Linear Discriminant Analysis

Fischer's Linear Discriminant Analysis (LDA) [33] tries to maximize the inter classes distance which is related to maximize the Rayleigh quotient:

$$\alpha_1 = \text{argmax}_\alpha \frac{\alpha^T \bar{S} \alpha}{\alpha^T \hat{S} \alpha} \tag{9}$$

Where the matrix $\bar{S}$ is the covariance matrix previously defined by the equation 8 and $\hat{S}$ is the intra-class scatter matrix defined as:

$$\hat{S} = \sum_{k \in \mathcal{K}} \sum_{i=1}^{N_t} (\mu_{k,i} - \bar{\mu}_z)(\mu_{k,i} - \bar{\mu}_z)^T \tag{10}$$

The LDA then corresponds to an eigenvalue decomposition of the matrix $\hat{S}^{-1}\bar{S}$. In fact, for any eigenvector α_i and associated eigenvalue λ_i of $\hat{S}^{-1}\bar{S}$, we have:

$$\frac{\alpha_i^T \bar{S} \alpha_i}{\alpha_i^T \hat{S} \alpha_i} = \lambda_i \tag{11}$$

Then, the leading λ_i maximizes the equation 9. Finally, we can build our template in a way similar to the PCA template.

IV. Results

A. Experimental setup

Our experiments target an implementation of mbedTLS embedded on a general purpose off-the-self device running at its max frequency of 168 MHz. We measured the near field electromagnetic emanations during the computation of kG using a digital oscilloscope (DSO) from Rohde & Schwarz (RTO 2024). It has a bandwidth of 2 GHz and the sample rate is set 1 GS/s. The DSO is connected to a Langer probe, an ICR HH 100 27 with a bandwidth of 6 GHz. The probe is placed over the IC package at less than 1mm from the surface. The scale of the EM axis of figures is qualitative and corresponds to the raw output of the oscilloscope ADC.

B. Synchronisation

The computation of kP operation on the Curve25519 takes 160 ms and is composed of 255 steps, one by key bit k_i. Unfortunately we cannot easily locate the different steps on the trace. The arithmetic of the targeted implementation is not constant time and depends on the computation's inputs. Consequently, we cannot know the timing of the different step. As we aim to locate the swap operation, we try to directly find their leakage patterns.

Algorithm 2 shows that each step contains 2 `cswap` operation. Actually, each `cswap` is composed of two swaps, one for the X coordinate and one for the Z coordinate. Hence, we have 255×4 swaps during a computation. Figure 2 shows the positions of the swaps with respect to the key bits computed. In the notation swap x.y, x corresponds to the manipulated key bit and y corresponds to the index of the swap. This figure shows that it is expected to find 254 patterns of 4 consecutive swaps. Then, the first step consists in identifying a group of 4 parts corresponding to the 4 consecutive swaps. We achieved this manually by visually inspecting the EM trace. The goal is to find a pattern constituted of 4 identical parts. The Figure 3 shows this pattern.

Once the pattern found, we compute the cross-correlation between the full trace and the selected pattern. For each profiling trace we locate 254 patterns using cross-correlation between each trace and the pattern extracted from the first trace. We did not extract the first four swaps corresponding to the first bit as they have a significant different leakage from the others. Indeed, R_0 is always initialized to $\mathcal{O}$ which causes a different leakage. We extract the last two swaps using cross-correlation with the second half of the pattern. Finally, we concatenated the swaps corresponding to the same key bit. We obtain 254 sub traces for each full trace. We used this method to build the profiling set and the attack set.

We can observe 254 main peaks corresponding to the 254 groups of 4 swaps.

C. T-test

In order to confirm that the mbedTLS swap implementation has 2 distinct leakages when processing a 0 our a 1, we compute a T-test to verify that the bit value has a significant impact on the trace's mean. Figure 4 shows the result of

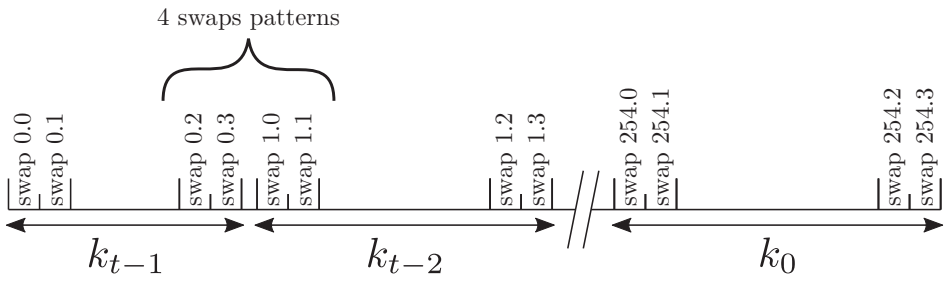

Fig. 2. Location of the swap operation during the computation

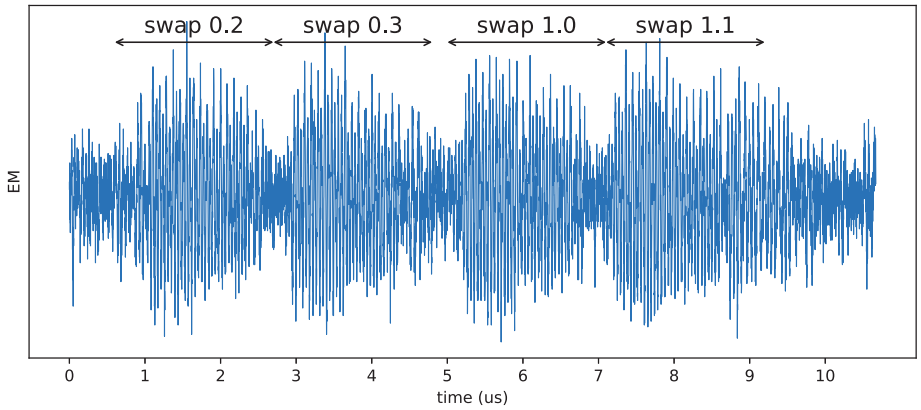

Fig. 3. EM traces of 4 consecutive swaps

the test between the subtraces corresponding to 0 and the subtraces corresponding to 1. We observe 4 groups of leakages corresponding to the 4 swaps. The first peak corresponds to the sign swap (line 17-18 of Figure 1). The 8 following peaks correspond to the 8 32-bits conditional swaps (line 23-24 the Figure 1). The peaks exceed, by far, the usual T-test threshold. This confirms that the swap operation is leaking with respect to the computed bit value.

We split the set of 100 traces between profiling traces and attack traces. We choose 90 traces for the profiling and 10 to evaluate the attack. We recall that the goal is to retrieve the key using only one attack trace.

D. Results of the Template Attack

We apply a template attack using the three dimension reduction method presented in Section III. For the PCA and LDA reduction, we used the scikit-learn library [34].

The results are presented in Figure 5. The abscissa axis corresponds to the number of traces used during the profiling phase. The ordinate axis corresponds to the percentage of correctly guessed key bits on the attack set. The results show that the LDA reduction allows reaching a 95% correct bit guess. That means that this attack can recover a large part of the key and that the swap operation have to be protected against template analysis.

Moreover, this implementation uses the coordinates randomization countermeasure which should protect the implementation against this kind of attack, [6], [7], [27]. Even in the case of the school book selection of PoIs (T-test), we have a success rate of approximately 65% which means some secret key bits are leaking from the protected scalar multiplication with coordinate randomization.

To recover the last unknown bits, the method presented by Lange *et al.* in [35] can be used.

E. Unsupervised attack

We have used so far the same device to acquire profiling traces and attack traces, which gives a high success rate to our attack. In a real life environment, an attacker would apply the built templates to another device. Additionally, in the case where the attacker has no access to a device at the profiling phase, or in the case that the attacker cannot submit his own key for labeling traces, our proposed attack is impossible to implement.

However, we can use the specificity of the Curve25519 implementation to improve the feasibility of the attack in a real case. In fact, the implementation standard [25] gives some requirements on the scalar k to compute kP on Curve25519. The 254th bit shall be 1 and the three last bits shall be 0. An attacker can acquire computation traces for several kP on the targeted device. The method to find the `cswap` operation in the section IV-B can be used to identify the `cswap` for the first bit 1 and these last three bits 0. Finally, the attacker can use the knowledge of this bits to label the `cswap` traces and

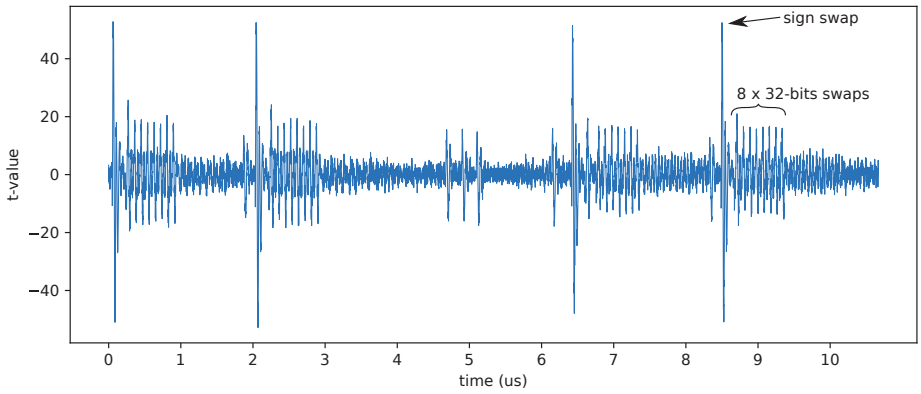

Fig. 4. Ttest between swaps corresponding to 0 and 1.

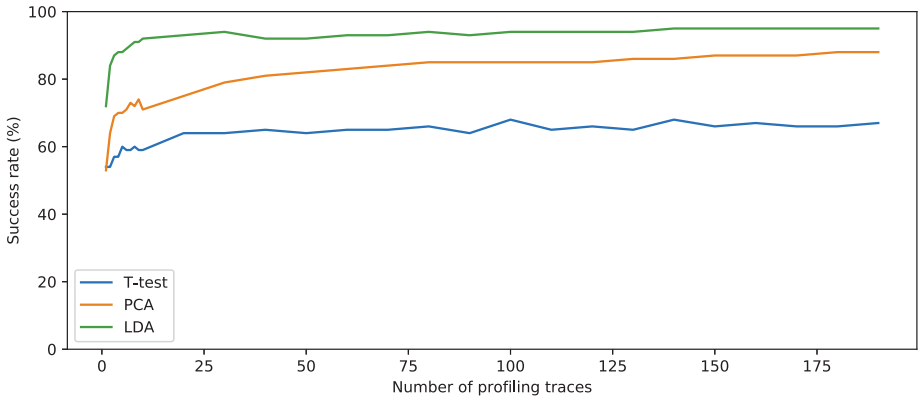

Fig. 5. Success rate w.r.t the dimension reduction method.

apply our attack. Note that the attacker should use only the `cswap` at the end of the Montgomery Ladder step for the 2 `cswap` corresponding to the first bit. Indeed, the very first `cswap` uses the neutral point $\mathcal{O}$ as input which can alter the template building due to its specific leakage.

Finally, our attack can be done on any ECC implementation which use `cswap` operation. The attacker model is very close to a real case :

- The attacker cannot submit a specific base point during the profiling and attack phase.
- The attacker cannot submit a specific scalar k during the profiling phase.
- The attacker does not know the output point during the profiling and attack phase.
- The attacker has a similar device to build template.
- The attacker can acquire EM traces from the targeted device.
- The attacker has only one EM trace of the targeted kP to recover k (scenario S1).

However, in this model, the profiling phase can be very long. In fact, the attacker should acquire thousands of EM traces of any kP to build an efficient template. If the targeted device computes only a few kP operations during a day, an attacker would wait several days to build efficient templates. Once an attacker has built an efficient template, he can recover all future k on the targeted device.

V. Inefficiency of classical countermeasures against our attack

First of all, our attack bypasses the coordinate randomization as explained in section IV. However, several countermeasures exist in the literature to protect the scalar multiplication in ECC against side channel attacks.

- **Coordinates randomization [4]** : the coordinate randomization changes the projective representation of a point $(X : Z)$ by $(\lambda X : \lambda Z)$, where λ is random. As we have seen in section IV-D, this countermeasure does not bring any security against our TA.
- **Point Blinding [4]** : the point blinding hides the base point with a random point R. This countermeasure has the same effect as coordinates randomization, the coordinates of the base point is random. Like, coordinates randomization, this countermeasure does not add any protection against our TA.
- **Random split of the secret key** : we can perform a random split in different ways :

- Additive [36] : we change kP by $(k-r)P + rP$, where r is random.
- Multiplicative [37] : we compute kP with $Q = k'S$, $S = rP$ and $k' = kr^{-1}[|E|]$, where r is random.
- Euclidian [38] : we compute kP with $Q = k_1P + k_2S$, $S = rP$, $k_1 = k[r]$ and $k_2 = \left\lfloor \frac{k}{r} \right\rfloor$, where r is random.

These methods do not provide any protection against our attack. In fact, our TA can find all intermediate keys in one trace. In this case, S2 is similar to S1.

- **Scalar blinding [4]** : the scalar blinding change the secret key k by $k+r|E|$, where r is random and $|E|$ the cardinal of the curve. This countermeasure provides no protection against our attack in the scenario S1. However, for the scenario S2, we cannot use several kP computation. In this case, S2 is similar to S1. However, during the profiling phase this countermeasure prevents the traces from being labelled. It implies that an attacker would have access to a copy of targeted device to perform profiling phase of our attack. In addition, the unsupervised attack of section IV-E is impossible.

In summary, well-known countermeasures to protect ECC do not add any protection against our TA. We propose a new countermeasure to protect ECC implementation against this attack. Moreover, the proposed countermeasure can be used for any elliptic curve models or any cryptographic implementation which requires a `cswap` operation.

VI. New countermeasure

The idea behind our countermeasure is to hide the multiplication by the bit value shown in equation 7. We add two random values r_0, r_1 and modify the formula in the following way:

$$W_{1,i} = (b + r_0)W_{2,i} + (1 - (b - r_1))W_{1,i} - r_0W_{2,i} - r_1W_{1,i} \quad (12)$$

$$W_{2,i} = (b + r_0)W_{1,i} + (1 - (b - r_1))W_{2,i} - r_0W_{1,i} - r_1W_{2,i} \quad (13)$$

With the original equation, the result of the operation that manipulates the bit was 0 or a random 32-bit value. With the new implementation, in both cases the result is a random value on 32-bits. The overhead due to the countermeasure is negligible. In the case of the mbedTLS implementation, we measured an additional time computation of 1% of the unprotected kP. This countermeasure was added to the mbedTLS `cswap` implementation. The Figure 6 shows some details about the implementation of the countermeasure.

```
1  int  mbedtls_mpi_safe_cond_swap( mbedtls_mpi *X,
       mbedtls_mpi *Y, mbedtls_mpi_uint swap )
2  {
3      int ret, s;
4      size_t i;
5      mbedtls_mpi_uint tmp, r[2];
6
7      if( X == Y )
8          return( 0 );
9
10     HAL_RNG_GenerateRandomNumber(&hrng, &r[0]);
11     HAL_RNG_GenerateRandomNumber(&hrng, &r[1]);
12
13     /* make sure swap is 0 or 1 in a time-
           constant manner */
14     swap = (swap | (unsigned char)-swap) >> 7;
15
16     MBEDTLS_MPI_CHK( mbedtls_mpi_grow( X, Y->n )
           );
17     MBEDTLS_MPI_CHK( mbedtls_mpi_grow( Y, X->n )
           );
18
19     s = X->s;
20     X->s = X->s * ( 1 - (swap - r[0]) ) + Y->s *
            (swap + r[1]) - r[0] * X->s - r[1] * Y
           ->s;
21     Y->s = Y->s * ( 1 - (swap - r[0]) ) +    s *
            (swap + r[1]) - r[0] * Y->s - r[1] * s;
22
23     for( i = 0; i < X->n; i++ )
24     {
25         tmp = X->p[i];
26         X->p[i] = X->p[i] * ( 1 - (swap - r[0])
               ) + Y->p[i] * (swap + r[1]) - r[0] *
                X->p[i] - r[1] * Y->p[i];
27         Y->p[i] = Y->p[i] * ( 1 - (swap - r[0])
               ) +    tmp * (swap + r[1]) - r[0] *
               Y->p[i] - r[1] * tmp;
28     }
29
30 cleanup:
31     return( ret );
32 }
```

Fig. 6. mbedTLS function to perform a conditional swap between X and Y with our countermeasure of section VI.

A. Efficiency of our countermeasure

Let us challenge our countermeasure against our TA for the 2 scenarios : S1 and S2. First of all, Figure 7 illustrates the 4 swaps patterns observed with the countermeasure. We observe that it is harder to distinguish the four swaps which makes the swaps identification task more difficult. Moreover, Figure 8 shows the result of the T-test computed in the same way that the previous one. It shows that the leakages are strongly reduced as the maximum value is under 10, whereas it was over 50 for the measurements without countermeasures.

The success rate obtained with the countermeasure are presented Figure 9 for the scenario S1. As we can see the results are not better than a random draw. Similar results have been obtained for the scenario S2. This shows that the blinding countermeasure protect the ECC implementation against our TA.

VII. Conclusion

In this paper, we presented a new template attack which targets ECC implementations. This attack uses a leakage on the conditional swap operation used for constant time Montgomery Ladder implementations. We recovered 95% of the key bits with only one measured EM attack trace. This specificity allows us to attack protocols like ECDH, ECDSA or EdDSA. Moreover, even if we use coordinates randomization to protect the scalar multiplication against side channel attack,

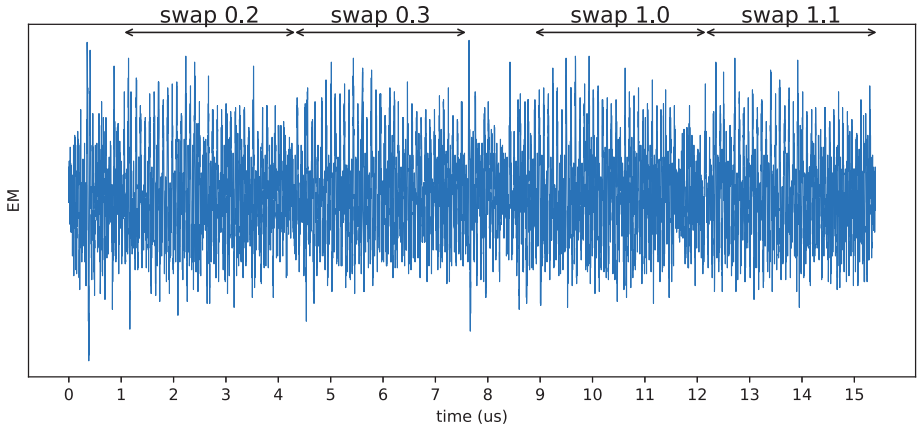

Fig. 7. EM traces of 4 consecutive swaps.

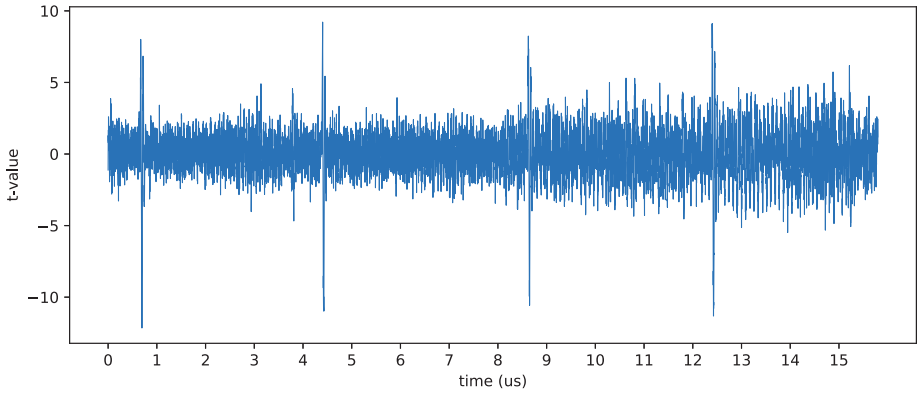

Fig. 8. Ttest between swaps corresponding to 0 and 1.

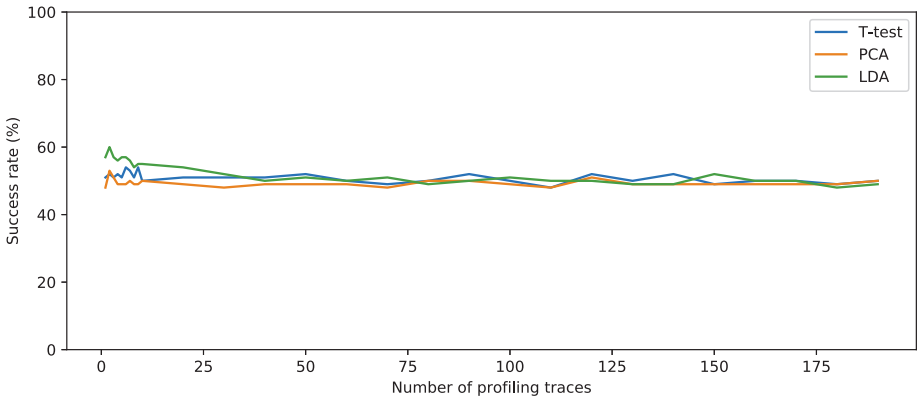

Fig. 9. Success rate w.r.t the dimension reduction method.

our attack can infer key bits. The simplicity and efficiency of our attack challenges the implementation choices made by cryptographic libraries and the countermeasures necessary to secure them. This observation led us to propose a new countermeasure that is efficient against our attack. The protection provided does not significantly downgrade the performance of the implementation. As a future work, it would be interesting to characterize the efficiency of our countermeasure against clustering methods based on deep learning. The efficiency of these methods [39] is a new challenge for ECC side channel security.

REFERENCES

[1] P. C. Kocher, "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, California, USA, August 18-22, 1996, Proceedings*, pp. 104–113, 1996.

[2] P. C. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," in *Proceedings of the 19th International Advances in Cryptology Conference (CRYPTO'99)*, no. 1666 in LNCS, pp. 388–397, Springer-Verlag, 1999.

[3] J.-J. Quisquater and D. Samyde, "Electromagnetic analysis (ema): Measures and counter-measures for smart cards," in *Smart Card Programming and Security* (I. Attali and T. Jensen, eds.), (Berlin, Heidelberg), pp. 200–210, Springer Berlin Heidelberg, 2001.

[4] J. Coron, "Resistance against differential power analysis for elliptic curve cryptosystems," in *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, Proceedings*, pp. 292–302, 1999.

[5] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, pp. 13–28, 2002.

[6] M. Medwed and E. Oswald, "Template attacks on ECDSA," in *Information Security Applications, 9th International Workshop, WISA 2008, Jeju Island, Korea, September 23-25, 2008, Revised Selected Papers*, pp. 14–27, 2008.

[7] L. Batina, L. Chmielewski, L. Papachristodoulou, P. Schwabe, and M. Tunstall, "Online template attacks," *J. Cryptographic Engineering*, vol. 9, no. 1, pp. 21–36, 2019.

[8] E. Nascimento, L. Chmielewski, D. Oswald, and P. Schwabe, "Attacking embedded ECC implementations through cmov side channels," in *Selected Areas in Cryptography - SAC 2016 - 23rd International Conference, St. John's, NL, Canada, August 10-12, 2016, Revised Selected Papers*, pp. 99–119, 2016.

[9] E. Nascimento and L. Chmielewski, "Applying horizontal clustering side-channel attacks on embedded ECC implementations," in *Smart Card Research and Advanced Applications - 16th International Conference, CARDIS 2017, Lugano, Switzerland, November 13-15, 2017, Revised Selected Papers*, pp. 213–231, 2017.

[10] "mbedTLS." https://tls.mbed.org/.

[11] T. Dierks and E. Rescorla, "The transport layer security (TLS) protocol version 1.2," *RFC*, vol. 5246, pp. 1–104, 2008.

[12] V. S. Miller, "Use of elliptic curves in cryptography," in *Advances in Cryptology - CRYPTO '85, Santa Barbara, California, USA, August 18-22, 1985, Proceedings*, pp. 417–426, 1985.

[13] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 243–264, 1987.

[14] P. Gallagher, "Digital signature standard (dss)."

[15] E. Barker, L. Chen, A. Roginsky, A. Vassilev, R. Davis, and S. Simon, "Recommendation for pair-wise key establishment using integer factorization cryptography."

[16] P. L. Montgomery, "Speeding the pollard and elliptic curve methods of factorization," *Mathematics of Computation*, vol. 48, pp. 243–264, 1987.

[17] M. Joye and S. Yen, "The montgomery powering ladder," in *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, pp. 291–302, 2002.

[18] H. Cohen, G. Frey, R. Avanzi, C. Doche, T. Lange, K. Nguyen, and F. Vercauteren, eds., *Handbook of Elliptic and Hyperelliptic Curve Cryptography*. Chapman and Hall/CRC, 2005.

[19] H. Edwards, "A normal form for elliptic curves," vol. 44, no. 3, pp. 393–422.

[20] D. J. Bernstein, "Curve25519: New diffie-hellman speed records," in *Public Key Cryptography - PKC 2006, 9th International Conference on Theory and Practice of Public-Key Cryptography, New York, NY, USA, April 24-26, 2006, Proceedings*, pp. 207–228, 2006.

[21] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B. Yang, "High-speed high-security signatures," in *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, pp. 124–142, 2011.

[22] D. J. Bernstein, C. Chuengsatiansup, and T. Lange, "Curve41417: Karatsuba revisited," *IACR Cryptology ePrint Archive*, vol. 2014, p. 526, 2014.

[23] D. J. Bernstein, S. Josefsson, T. Lange, P. Schwabe, and B. Yang, "Eddsa for more curves," *IACR Cryptology ePrint Archive*, vol. 2015, p. 677, 2015.

[24] D. J. Bernstein, P. Birkner, M. Joye, T. Lange, and C. Peters, "Twisted edwards curves," in *Progress in Cryptology - AFRICACRYPT 2008, First International Conference on Cryptology in Africa, Casablanca, Morocco, June 11-14, 2008. Proceedings*, pp. 389–405, 2008.

[25] A. Langley, M. Hamburg, and S. Turner, "Elliptic curves for security," *RFC*, vol. 7748, pp. 1–22, 2016.

[26] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3." RFC 8446, Aug. 2018.

[27] M. Dugardin, L. Papachristodoulou, Z. Najm, L. Batina, J. Danger, and S. Guilley, "Dismantling real-world ECC with horizontal and vertical template attacks," in *Constructive Side-Channel Analysis and Secure Design - 7th International Workshop, COSADE 2016, Graz, Austria, April 14-15, 2016, Revised Selected Papers*, pp. 88–108, 2016.

[28] C. Archambeau, E. Peeters, F. Standaert, and J. Quisquater, "Template attacks in principal subspaces," in *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, pp. 1–14, 2006.

[29] F. Standaert and C. Archambeau, "Using subspace-based template attacks to compare and combine power and electromagnetic information leakages," in *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, pp. 411–425, 2008.

[30] E. Cagli, C. Dumas, and E. Prouff, "Enhancing dimensionality reduction methods for side-channel attacks," in *Smart Card Research and Advanced Applications - 14th International Conference, CARDIS 2015, Bochum, Germany, November 4-6, 2015. Revised Selected Papers*, pp. 15–33, 2015.

[31] R. Specht, J. Heyszl, M. Kleinsteuber, and G. Sigl, "Improving non-profiled attacks on exponentiations based on clustering and extracting leakage from multi-channel high-resolution em measurements," in *Constructive Side-Channel Analysis and Secure Design* (S. Mangard and A. Y. Poschmann, eds.), (Cham), pp. 3–19, Springer International Publishing, 2015.

[32] I. T. Jolliffe, "Principal component analysis," in *International Encyclopedia of Statistical Science*, pp. 1094–1096, 2011.

[33] R. A. Fisher, "THE STATISTICAL UTILIZATION OF MULTIPLE MEASUREMENTS," *Annals of Eugenics*, vol. 8, pp. 376–386, Aug. 1938.

[34] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pp. 108–122, 2013.

[35] T. Lange, C. van Vredendaal, and M. Wakker, "Kangaroos in side-channel attacks," in *Smart Card Research and Advanced Applications - 13th International Conference, CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers*, pp. 104–121, 2014.

[36] C. Clavier and M. Joye, "Universal exponentiation algorithm," in *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, no. Generators, pp. 300–308, 2001.

[37] E. Trichina and A. Bellezza, "Implementation of elliptic curve cryptography with built-in counter measures against side channel attacks," in *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, pp. 98–113, 2002.

[38] M. Ciet and M. Joye, "(virtually) free randomization techniques for elliptic curve cryptography," in *Information and Communications Security, 5th International Conference, ICICS 2003, Huhehaote, China, October 10-13, 2003, Proceedings*, pp. 348–359, 2003.

[39] M. Carbone, V. Conin, M. Cornelie, F. Dassance, G. Dufresne, C. Dumas, E. Prouff, and A. Venelli, "Deep learning to evaluate secure RSA implementations," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2019, no. 2, pp. 132–161, 2019.