J Cryptogr Eng (2019) 9:21–36
https://doi.org/10.1007/s13389-017-0171-8

REGULAR PAPER

Online template attacks

Lejla Batina[1] · **Łukasz Chmielewski**[2] · **Louiza Papachristodoulou**[1] · **Peter Schwabe**[1] · **Michael Tunstall**[3]

Received: 4 September 2016 / Accepted: 29 July 2017 / Published online: 12 August 2017

Abstract Template attacks are a special kind of side-channel attacks that work in two stages. In a first stage, the attacker builds up a database of template traces collected from a device which is identical to the attacked device, but under the attacker's control. In the second stage, traces from the target device are compared to the template traces to recover the secret key. In the context of attacking elliptic curve scalar multiplication with template attacks, one can interleave template generation and template matching and reduce the amount of template traces. This paper enhances the power of this technique by defining and applying the concept of *online template attacks*, a general attack technique with minimal assumptions for an attacker, who has very very limited control over the template device. We show that online template attacks need only one power consumption trace of a scalar multiplication on the target device; they are thus suitable not only against ECDSA and static elliptic curve Diffie–Hellman (ECDH), but also against elliptic curve scalar multiplication in ephemeral ECDH. In addition, online template attacks need only one template trace per scalar bit and they can be applied to a broad variety of scalar multiplication algorithms. To demonstrate the power of online template attacks, we recover scalar bits of a scalar multiplication using the double-and-add-always algorithm on a twisted Edwards curve running on a smartcard with an ATmega163 CPU.

Keywords Side-channel analysis · Template attacks · Scalar multiplication · Elliptic curves

This work was supported in part by the Technology Foundation (STW) through Project 12624 SIDES, by the Netherlands Organization for Scientific Research NWO through Veni 2013 Project 13114 and Project ProFIL 628.001.007, and the ICT COST action IC1204 TRUDEVICE. Permanent ID of this document: 14c4b76aa264503f89f93abc9baf72c3.

✉ Łukasz Chmielewski
lukchmiel@gmail.com

Lejla Batina
lejla@cs.ru.nl

Louiza Papachristodoulou
louiza@cryptologio.org

Peter Schwabe
peter@cryptojedi.org

Michael Tunstall
michael.tunstall@cryptography.com

[1] Radboud University Nijmegen, PO Box 9010, 6500 GL Nijmegen, The Netherlands

[2] Riscure, Delftechpark 49, 2628 XJ Delft, The Netherlands

[3] Rambus Cryptography Research Division, 425 Market Street 11th Floor, San Francisco, CA 94105, USA

1 Introduction

Side-channel attacks exploit various physical leakages of secret information or instructions from cryptographic devices, and they constitute a constant threat for cryptographic implementations. We focus on power analysis attacks that exploit the power consumption leakage from a device running some cryptographic algorithm. Attacking elliptic curve cryptosystems (ECC) with natural protection against side-channel attacks, e.g., implementations using Edwards curves, is quite challenging. This form of elliptic curves, proposed by Edwards in 2007 [17] and promoted for cryptographic applications by Bernstein and Lange [5], has several advantages compared to elliptic curves in Weierstrass form. For instance, the fast and complete formulas for addition and doubling make these types of curves more appealing for memory-constrained devices and at the same time resistant to classical simple power analysis (SPA) techniques.

Although considered a very serious threat against ECC implementations, differential power analysis (DPA), as proposed in [14,31], cannot be applied directly to ECDSA or ephemeral elliptic curve Diffie–Hellman (ECDH) because the secret scalar is used only once. This is incompatible with the requirement of DPA to see large number of power traces of computations on the same secret data. In order to attack various asymmetric cryptosystems, new techniques that reside between SPA and DPA were developed, most notably collision [3,18,23,40,41,44] and template attacks [15,35,39]. The efficiency of most of those collision-based attacks is shown only on simulated traces; no practical experiments on real ECC implementations have verified these results. To the best of our knowledge, only two practical collision-based attacks on exponentiation algorithms are published, each of which relies on very specific assumptions and deals with very special cases. Hanley et al. exploit collisions between input and output operations of the same trace [19]. Wenger et al. in [42] performed a hardware-specific attack on consecutive rounds of a Montgomery ladder implementation. However, both attacks are very restrictive in terms of applicability to various ECC implementations as they imply some special implementation options, such as the use of López-Dahab coordinates, where field multiplications use the same key-dependent coordinate as input to two consecutive rounds. In contrast, our attack is much more generic as it applies to arbitrary choices of curves and coordinates, and many scalar multiplication algorithms.

Related work Collision attacks exploit leakages by comparing two portions of the same or different traces to discover when values are reused. The Big Mac attack [41] is the first theoretical attack on public key cryptosystems, in which only a single trace is required to observe key dependencies and collisions during an RSA exponentiation. Witteman et al. in [43] performed a similar attack on the RSA modular exponentiation in the presence of blinded messages. Clavier et al. introduced in [13] horizontal correlation analysis, as a type of attack where a single power trace is enough to recover the private key. They also extended the Big Mac attack by using different distinguishers. Horizontal correlation analysis was performed on RSA using the Pearson correlation coefficient in [13] and triangular trace analysis of the exponent in [12].

The first horizontal technique relevant to ECC is the doubling attack, presented by Fouque and Valette in [18]. Homma et al. in [23] proposed a generalization of this attack to binary right-to-left, m-ary and sliding-window methods. An attack proposed by Bauer et al. in [3] is a type of horizontal collision correlation attack on ECC, which combines atomicity and randomization techniques. A recent attack on ECC is horizontal cross-correlation [20]; the approach is similar to [43] but uses only a single trace. The most recent horizontal attacks on ECC are single-trace attacks on a software implementation of scalar multiplication with precomputed points (for example, an m-ary implementation) [25]. The presented attacks target a single trace and employ clustering and correlation to recover scalar bits.

Template attacks are a combination of statistical modeling and power analysis attacks consisting of two phases, as follows. The first phase is the *profiling* or *template-building* phase, where the attacker builds templates to characterize the device by executing a sequence of instructions on fixed data. The second phase is the matching phase, in which the attacker matches the templates to actual traces of the device. The attacker is assumed to possess a device which behaves the same as the target device, in order to build template traces while running the same implementation as the target. Medwed and Oswald demonstrated in [35] a practical template attack on ECDSA. However, their attack required an offline DPA attack on the EC scalar multiplication operation during the template-building phase, in order to select the points of interest. They also need 33 template traces per key bit. Furthermore, attacks against ECDSA and other elliptic curve signature algorithms only need to recover a few bits of the ephemeral scalar for multiple scalar multiplications with different ephemeral scalars and can then employ lattice techniques to recover the long-term secret key [4,15,39]. This is not possible in the context of ephemeral ECDH: An attacker gets only a single trace and needs to recover sufficiently many bits of this ephemeral scalar from side-channel information to be able to compute the remaining bits through, for example, Kangaroo techniques.

Another template attack on ECC is presented in [21]. This attack exploits register location-based leakage using a high-resolution inductive EM probe; therefore, the attack is considerably expensive to execute. A template attack on a wNAF ECC algorithm is presented in [45]. However, this attack is applied to an implementation that is not protected with either scalar randomization or base-point randomization. Furthermore, contrary to our approach, all of the above attacks require multiple traces to construct a template.

This paper is an extended version of the original paper on online template attacks [2]. In the meantime, from the original paper on OTA until this extended version is written, two related works are published that verify the applicability of OTA on different curves. The first work from Dugardin et.al [16] performed OTA on Weierstrass (Brainpool and NIST) curves using EM emanations. A follow-up work from Özgen et. al [38] verified that OTA can successfully give the correct prediction on the scalar bit, by using distinguishers from machine learning (classification methods).

Our contribution In this paper, we introduce an adaptive template attack technique, which we call *online template attacks* (OTA). This technique is able to recover a complete scalar from only one power trace of a scalar multiplication using

this scalar. The attack is characterized as *online*, because we create the templates *after* the acquisition of the target trace. While we use the same terminology, our attack is not a typical template attack; i.e., no preprocessing template-building phase is necessary. Our attack functions by acquiring one target trace from the device under attack and comparing patterns of certain operations from this trace with templates obtained from the attacker's device that runs the same implementation. Pattern matching is performed at suitable points in the algorithm, where key bit-related assignments take place by using an automated module based on the Pearson correlation coefficient.

The attacker needs only very limited control over the device used to generate the online template traces. The main assumption is that the attacker can choose the input point to a scalar multiplication, an assumption that trivially holds even without any modification to the template device in the context of ephemeral ECDH. It also holds in the context of ECDSA, if the attacker can modify the implementation on the template device or can modify internal values of the computation. This is no different than for previous template attacks against ECDSA.

Our methodology offers a generic attack framework, which is applicable to various forms of curves (Weierstrass, Edwards and Montgomery curves) and implementations. As a proof of concept, we attack the doubling operation in the double-and-add-always algorithm. Contrary to the doubling attack [18], our attack can be launched against right-to-left algorithms and Montgomery ladder. We further note that Medwed and Oswald perform a very special template attack based on a set of assumptions: DPA performed in advance to find intermediate points for templates, implementation with Hamming weight leakage and applicability only to ECDSA. Online template attacks do not have these restrictions; they need only a single target trace and only a single template trace per key bit. The advantages of our attack over previously proposed attacks are as follows:

- It does not require any cumbersome preprocessing template-building phase, but a rather simple post-processing phase.
- It does not assume any previous knowledge of the leakage model.
- It does not require full control of the device under attack.
- It works against SPA-protected and to some extent DPA-protected implementations with unified formulas for addition and doubling.
- Countermeasures such as scalar randomization and changing point representation from affine to (deterministic) projective representation inside the implementation do not prevent our attack.
- It is applicable to the Montgomery ladder and to constant-time (left-to-right and right-to-left) exponentiation algorithms.
- It is experimentally confirmed on an implementation of double-and-add-always scalar multiplication on the twisted Edwards curve used in the Ed25519 signature scheme.
- Our attack is a chosen input attack—it means that the adversary needs to control the input of a scalar multiplication (but not the scalar). Most ECC implementations use inputs in affine (or compressed affine) coordinates and internally convert to projective representation. In this paper, we show how to apply the attack if an attacker controls either the projective coordinates input or affine input (even if it is compressed).
- Our attack works when the target trace and the template traces are acquired from the same device and also when the target trace is acquired from a different device than the template traces, as shown in our experimental results. Note that using the same device for both templates and target traces, would actually make the attack easier, because there would be no vertical or horizontal misalignment in the traces.

Online template attacks require only one *target trace* and one *online template trace* per key bit. We therefore claim that our technique is the most efficient practical side-channel attack applicable to ephemeral scalar ECC. When applied to ECDSA, the proposed attack can be used in combination with lattice techniques similar to [4,39], in order to derive the whole private key from a few bits of multiple ephemeral keys.

As mentioned in the previous subsection, this paper is an extended version of the original paper on online template attacks [2]. We present the theoretic primitives of the attack and verify our theory with new experiments with different types of input, namely with 256-bit projective input, with the reduced 255-bit projective coordinates and finally with affine coordinates. These experiments verify the applicability of OTA and provide a complete practical setting for this type of side-channel attack.

Organization of the paper This paper is organized as follows. We introduce and explain OTA in Sect. 2. Section 3 gives specific examples of how the attack applies to different scalar multiplication algorithms. Section 4 presents our practical OTA on double-and-add-always scalar multiplication. A discussion of how the proposed attack can be applied to implementations that include countermeasures that randomize the algorithm or operands is given in Sect. 5. Finally, Sect. 6 summarizes our contribution and concludes the paper.

2 Online template attacks

We define an online template attack as a side-channel attack with the following conditions:

1. The attacker obtains only one power trace of the cryptographic algorithm involving the targeted secret data. This trace is called the *target trace*. We call the device from which the target trace is obtained the *target device*. This property makes it possible to attack scalar multiplication algorithms with ephemeral scalar and with randomized scalar.
2. The attacker is generating template traces *after* having obtained the target trace. These traces are called *(online) template traces*.
3. The attacker obtains the template traces on the target device or a similar device[1] *with very limited control over it*, i.e., access to the device to run several executions with chosen public inputs. The attacker does not rely on the assumption that the secret data are the same for all template traces.
4. At least one assignment in the exponentiation algorithm is made depending on the value of particular scalar bit(s), but there are no branches with key-dependent computations. Since we are attacking the doubling operation, this key-dependent assignment should be during doubling. As a counterexample, we note that the binary right-to-left add-always algorithm for Lucas recurrences [29] is resistant to the proposed attack, because the result of the doubling is stored in a non-key-dependent variable.

In the following, we show that online template attacks are feasible and can be applied against implementations of various scalar multiplication algorithms. In fact, we show that we need only a single template trace per scalar bit. Transfer of the approach to the corresponding exponentiation algorithms (for example in RSA or DSA) is straightforward. Transfer to other cryptographic algorithms is clearly not trivial; we consider online template attacks as a specialized means to attack scalar multiplication and exponentiation algorithms.

2.1 Attack description

Template attacks consist of two phases, *template building* for characterizing the device and *template matching*, where the characterization of the device together with a power trace from the device under attack is used to determine the secret [34]. Therefore, the first condition of our proposed attack is typically fulfilled by all attacks of this kind.

[1] By similar device, we mean the same type of microcontroller running the same algorithm. Observe that the target device may be the same as the target one.

It is well known that template attacks against scalar multiplication can generate templates "on-the-fly," i.e., interleaving the template-building and matching phases. See, for example, [35, Sec. 5.3]. We take this idea further by building templates after the target trace has been obtained (condition 2). The attacker, being able to do things in this order, needs only limited control over the target device. Moreover, the attacker is not affected by randomization of the secret data during different executions of the algorithm, since he always has to compare his template traces with the same target trace.

The basic idea consists of comparing the target trace and an online template trace while executing scalar multiplication and then finding similar patterns between them, based on hypothesis on a bit for a given operation. The target trace is obtained only once with input $\boldsymbol{P}$. For every bit of the scalar, we need to obtain an online template trace with input $k\boldsymbol{P}$, $k \in \mathbb{Z}$, where k is chosen as a function of our hypothesis on this bit. The attack requires to send a different point to the device, thus generating a template trace for each of these points. Each template trace should be then compared with the part of the target trace that corresponds to the manipulated bit. However, due to jitter, for example, it may be not easy to determine that part. Therefore, we compare the template trace with the target trace at each sample offset.

We performed pattern matching for our traces using an automated module based on the Pearson correlation coefficient, $\rho(X, Y)$, which measures the linear relationship between two variables X and Y. For power traces, the correlation coefficient shows the relationship between two points of the trace, which indicates the leakage of key-dependent assignments during the execution of a cryptographic algorithm. The leakage can be due to differences in Hamming weight or Hamming distance of the variables, but the exact leakage model does not affect online template attacks in any way. Extensions to other leakage models and distinguishers are straightforward. Our pattern matching corresponds to a list of the correlation coefficients that show the relationship between all samples from the template trace to the same consecutive amount of samples in the target trace. If our hypothesis on the given key bit is correct, then the pattern match between our traces at the targeted operation will be high (in our experiments it reached 99%).

In this way, we can recover the first i bits of the key. Knowledge of the first i bits provides us with complete knowledge of the internal state of the algorithm just before the $(i+1)$th bit is processed. Since at least one operation in the loop depends on this bit, we can make a hypothesis about the $(i+1)$th bit, compute an online template trace based on this hypothesis and correlate this trace with the target trace at the relevant predetermined point of the algorithm.

A separate question is how many templates need to be created per attacked bit. In this paper, we show that only a single template trace per key bit is sufficient if a correct template can

be safely recognized from any incorrect template, see Sect. 4 for examples. Essentially, for each experiment we establish a correlation threshold to recognize correct templates from incorrect ones; then, we can create one template (for bit 0, for example), and depending on the template correlation and the threshold, we can recover the scalar bit. If the difference between the correlation for correct and incorrect templates is sufficiently large, then the threshold can be learned either through profiling or by computing two templates (for 0 and 1) in the first few iterations.

Furthermore, note that if a bit is incorrectly identified then all subsequent templates would not match with the target trace. In case this happens, then it is possible to backtrack to the last successful matching and restart the attack.

3 Applying the attack to scalar multiplication algorithms

3.1 Attacking the left-to-right double-and-add-always algorithm

The core idea and feasibility of the attack is demonstrated by considering the example of the double-and-add-always algorithm described in Algorithm 1. We note that the first execution of the loop always starts by doubling the input point $\boldsymbol{P}$, for all values of k. We assume that $k_{x-1} = 1$. Depending on the second most significant key bit k_{x-2}, the output of the first iteration of the algorithm will be either $2\boldsymbol{P}$ or $3\boldsymbol{P}$. For any point $\boldsymbol{P}$, we can, therefore, get a power trace for the operation $2\boldsymbol{P}$, i.e., we let the algorithm execute the first two double-and-add iterations. In our setup, we can zoom into the level of one doubling, which will be our template trace. Then we perform the same procedure with $2\boldsymbol{P}$ as the input point to obtain the online template trace that we want to compare with the target trace. If we assume that the second most significant bit of k is 0, then we compare the $2\boldsymbol{P}$ template with the output of the doubling at first iteration. Otherwise, we compare it with the online template trace for $3\boldsymbol{P}$.

Algorithm 1: The left-to-right double-and-add-always algorithm

Input: $\boldsymbol{P}, k = (k_{x-1}, k_{x-2}, \ldots, k_0)_2$
Output: $\boldsymbol{Q} = k \cdot \boldsymbol{P}$

$\boldsymbol{R}_0 \leftarrow \boldsymbol{P}$;
for $i \leftarrow x - 2$ **down to** *0* **do**
  $\boldsymbol{R}_0 \leftarrow 2\boldsymbol{R}_0$;
  $\boldsymbol{R}_1 \leftarrow \boldsymbol{R}_0 + \boldsymbol{P}$;
  $\boldsymbol{R}_0 \leftarrow \boldsymbol{R}_{k_i}$;
end
return $\boldsymbol{R}_0$

Assuming that the first $(i-1)$ bits of k are known, we can derive the ith bit by computing the two possible states of $\boldsymbol{R}_0$ after this bit has been treated and recover the key iteratively. Note that only the assignment in the ith iteration depends on the key bit k_i, but none of the computations do, so we need to compare the trace of the doubling operation in the $(i+1)$th iteration with our original target trace. To decide whether the ith bit of k is zero or one, we compare the trace that the doubling operation in the $(i+1)$th iteration would give for $k_{i+1} = 0$ with the target trace. For completeness, we can compare the target trace with a trace obtained for $k_{i+1} = 1$ and verify that it has lower pattern match percentage; in this case, the performed attack needs two online template traces per key bit. However, if during the acquisition phase the noise level is low and the signal is of good quality, we can perform an efficient attack with only our target trace and a single trace for the hypothetical value of $\boldsymbol{R}_{k_{i+1}}$.

Note that the method above assumes that one template trace is acquired to recover a single bit. It is possible to acquire multiple template traces to recover multiple bits at the same time, for example, three template traces can be produced to recover 2 bits at once. However, attacking single bits is more efficient in terms of storage of template traces and offline precomputation than attacking a group of bits. This is an advantage of building online templates compared to the usual template attacks. In particular, sequentially attacking 2 bits requires 2 template traces; if a template is similar to the target trace, then the bit is guessed correctly; otherwise, the bit is incorrect. Attacking both bits at once requires 3 template traces (the fourth choice can be implied if none of the 3 templates matches the attacked trace). In general, attacking n bits simultaneously requires an offline computation of $(2^n - 1)$ template traces.

3.2 Attacking the right-to-left double-and-add-always algorithm

In this section, we examine the binary right-to-left add-always algorithm of [27], as Algorithm 2. Contrary to Algorithm 1, the computations in the main loop of the above algorithm clearly depend on the key.

Algorithm 2: Binary right-to-left double-and-add-always algorithm

Input: $\boldsymbol{P}, k = (k_{x-1}, k_{x-2}, \ldots, k_0)_2$
Output: $\boldsymbol{Q} = k \cdot \boldsymbol{P}$

$\boldsymbol{R}_0 \leftarrow \boldsymbol{O}$;
$\boldsymbol{R}_1 \leftarrow \boldsymbol{P}$;
for $i \leftarrow 0$ **up to** *x-1* **do**
  $b \leftarrow 1 - k_i$;
  $\boldsymbol{R}_b \leftarrow 2\boldsymbol{R}_b$;
  $\boldsymbol{R}_b \leftarrow \boldsymbol{R}_b + \boldsymbol{R}_{k_i}$;
end
return $\boldsymbol{R}_0$

Attacking the right-to-left double-and-add-always algorithm of [27] is a type of key-dependent assignment OTA. We target the doubling operation and note that the input point will be doubled either in the first (if $k_0 = 0$) or in the second iteration of the loop (if $k_0 = 1$). If k is fixed, we can easily decide between the two by inputting different points, since if $k_0 = 1$ we will see the common operation $2\,\boldsymbol{O}$. If the k is not fixed, we simply measure the first two iterations and again use the operation $2\,\boldsymbol{O}$ if the template generator should use the first or second iteration. Once we are able to obtain clear traces, the attack itself follows the general description of Sect. 2. If we assume that the first i bits of k are known and we wish to derive the $(i+1)$th bit, this means that we know the values of $\boldsymbol{R}_0$ and $\boldsymbol{R}_1$ at the start of the $(i+1)$th iteration. By making a hypothesis on the value of the $(i+1)$th key bit, we can decide according to the matching percentage if $\boldsymbol{R}_0$ or $\boldsymbol{R}_1$ was used.

3.3 Attacking the Montgomery ladder

The Montgomery ladder, initially presented by Montgomery in [36] as a way to speed up scalar multiplication on elliptic curves, and later used as the primary secure and efficient choice for resource-constrained devices, is one of the most challenging algorithms for simple side-channel analysis due to its natural regularity of operations. A comprehensive security analysis of the Montgomery ladder given by Joye and Yen in [26] showed that the regularity of the algorithm makes it intrinsically protected against a large variety of implementation attacks (SPA, some fault attacks, etc.). For a specific choice of projective coordinates for the Montgomery ladder, as described in Algorithm 3, one can do computations with only X and Z coordinates, which makes this option more memory efficient than other algorithms.

Algorithm 3: The Montgomery Ladder

Input: $\boldsymbol{P}, k = (k_{x-1}, k_{x-2}, \ldots, k_0)_2$
Output: $\boldsymbol{Q} = k \cdot \boldsymbol{P}$

$\boldsymbol{R}_0 \leftarrow \boldsymbol{P}$;
$\boldsymbol{R}_1 \leftarrow 2\boldsymbol{P}$;
for $i \leftarrow x-2$ **down to** 0 **do**
  $b \leftarrow 1 - k_i$;
  $\boldsymbol{R}_b \leftarrow \boldsymbol{R}_0 + \boldsymbol{R}_1$;
  $\boldsymbol{R}_{k_i} \leftarrow 2 \cdot \boldsymbol{R}_{k_i}$;
end
return $\boldsymbol{R}_0$

The main observation that makes our attack applicable to the Montgomery ladder is that at least one of the computations, namely the doubling in the main loop, directly depends on the key bit k_i. For example, if we assume that the first three bits of the key are 100, then the output of the first iteration will be $R_0 = 2\boldsymbol{P}$. If we assume that the first bits are 110, then the output of the first iteration will be $R_0 = 3\boldsymbol{P}$. Therefore, if we compare the pattern of the output of the first iteration of Algorithm 3 with scalar $k = 100$, we will observe higher correlation with the pattern of $R_0 = 2\boldsymbol{P}$ than with the pattern of $R_0 = 3\boldsymbol{P}$. This is demonstrated in the following working example.

$k = 100$ $\boldsymbol{R_0} = \boldsymbol{P}, \boldsymbol{R_1} = 2\boldsymbol{P}$	$k = 110$ $\boldsymbol{R_0} = \boldsymbol{P}, \boldsymbol{R_1} = 2\boldsymbol{P}$
$b = 1 : \boldsymbol{R_1} = 3\boldsymbol{P}, \boldsymbol{R_0} = 2\boldsymbol{P}$ $b = 1 : \boldsymbol{R_1} = 5\boldsymbol{P}, \boldsymbol{R_0} = 4\boldsymbol{P}$	$b = 0 : \boldsymbol{R_0} = 3\boldsymbol{P}, \boldsymbol{R_1} = 4\boldsymbol{P}$ $b = 1 : \boldsymbol{R_1} = 7\boldsymbol{P}, \boldsymbol{R_0} = 6\boldsymbol{P}$

3.4 Attacking Side-Channel Atomicity

Side-channel atomicity is a countermeasure proposed by Chevallier-Mames et al. [11], in which individual operations are implemented in such a way that they have an identical side-channel profile (e.g., for any branch and any key bit-related subroutine). In short, it is suggested in [11] that the point doubling and addition operations are implemented such that the same code is executed for both operations. This renders the operations indistinguishable by simply inspecting a suitable side channel. One could, therefore, implement an exponentiation as described in Algorithm 4.

Algorithm 4: Side-Channel Atomic double-and-add algorithm

Input: $\boldsymbol{P}, k = (k_{x-1}, k_{x-2}, \ldots, k_0)_2$
Output: $\boldsymbol{Q} = k \cdot \boldsymbol{P}$

$R_0 \leftarrow \boldsymbol{O}$; $R_1 \leftarrow \boldsymbol{P}$; $i \leftarrow x-1$;
$n \leftarrow 0$;
while $i \geq 0$ **do**
  $\boldsymbol{R}_0 \leftarrow \boldsymbol{R}_0 + \boldsymbol{R}_n$;
  $n \leftarrow n \oplus k_i$;
  $i \leftarrow i - \neg n$;
end
return $\boldsymbol{R}_0$

There are certain choices of coordinates and curves where this approach can be deployed by using unified or complete addition formulas for the group operations. For example, the Jacobi form [33] and Hessian [30] curves come with a unified group law and Edwards curves [8,9] even have a complete group law. For Weierstrass curves, Brier and Joye suggest an approach for unified addition in [10].

Simple atomic algorithms do not offer any protection against online template attacks, because the regularity of point operations does not prevent mounting this sort of attack. The point $2\boldsymbol{P}$, as output of the third iteration of Algorithm 4,

will produce a power trace with very similar pattern to the trace that would have the point $2\boldsymbol{P}$ as input. Therefore, the attack will be the similar as the one described in Sect. 3.1; the only difference is that instead of the output of the second iteration of the algorithm, we have to focus on the pattern of the third iteration. In general, when an attacker forms a hypothesis about a certain number of bits of k, the hypothesis will include the point in time where $\boldsymbol{R}_0$ will contain the predicted value. This will mean that an attacker would have to acquire a larger target trace to allow all hypotheses to be tested.

4 Experimental results

This section presents our experimental results. Firstly, in Sect. 4.1 we describe the attacked implementation and the measurement setup that we use to perform attacks. Then, we present experimental results of an OTA with extended projective coordinates of 256-bit in Sect. 4.2; this is the usual input value for our smart card. In Sect. 4.3, we present OTA on extended projective coordinates with reduced 255-bit input. Finally, Sect. 4.4 presents an OTA applied to input points with affine compressed coordinates. All the attacks target are performed iteratively bit by bit, and they five most significant bits of the scalar.

4.1 Target implementation and experimental setup

To validate feasibility and efficiency of our proposed method, we attack an elliptic curve scalar multiplication implementation running on an "ATmega card," i.e., an ATmega163 microcontroller [1] in a smart card. To illustrate that our attack also works if the template device is not the same as the target device, we used two different smart cards: one to obtain the target trace and one to obtain the online template traces.

Our measurement setup uses a Picoscope 5203[2] with sampling rate of 125M samples per second for both target trace and online template traces. This oscilloscope has limited acquisition memory buffer to 32M samples. Since 5 iterations of the scalar multiplication algorithm take around 235 ms, it means that with sampling rate of 125M samples per second we can record a trace of approximately 29.4M samples.

The scalar multiplication algorithm is based on the curve arithmetic of the Ed25519 implementation presented in [24], which is available online at http://cryptojedi.org/crypto/#avrnacl. The elliptic curve used in Ed25519 is the twisted Edwards curve $E : -x^2 + y^2 = 1 + dx^2y^2$ with $d = -(121{,}665/121{,}666)$ and base point

$$\boldsymbol{P} = (15112221349535400772501151409588531511454012693041857206046113283949847762202,\ 46316835694926478169428394003475163141307993866256225615783033603165251855960).$$

For more details on Ed25519 and this specific curve, see [6,7].

We modified the software to perform a double-and-add-always scalar multiplication (see Algorithm 1). The whole underlying field and curve arithmetic is the same as in [24]. This means in particular that points are internally represented in extended coordinates as proposed in [22]. In this coordinate system, a point $\boldsymbol{P} = (x, y)$ is represented as $(X{:}Y{:}Z{:}T)$ with $x = X/Z$, $y = Y/Z$ and $x \cdot y = T/Z$.

4.2 Online template attack with 256-bit projective input

In this subsection, we describe how to apply an OTA if the input supplied to the scalar multiplication is in extended projective coordinates, i.e, if the attacker has full control over all coordinates of the starting point. This is a realistic assumption if a protocol avoids inversions entirely and protects against leakage of projective coordinates by randomization as proposed in [37, Sec. 6]. Recall that for extended coordinates, T is fully determined by X, Y and Z; they are an extension of standard projective coordinates.

The attack targets the output of the doubling operation. We performed pattern matching for our traces as described in Sect. 2.1. In this way, we could determine the leakage of key-dependent assignments during the execution of the algorithm.

We first demonstrate how to attack a single bit, and then, we present our results from recovering the five most significant unknown bits of the scalar (recall that the highest bit is always set to one; see Algorithm 1). The remaining bits can be attacked iteratively in the same way as described in Sect. 2.1; as stated above, we were not able to do so due to technical limitations of our measurement setup.

The first observation from our experiments is that when we execute the same algorithm with the same input point on two different cards, there is a constant vertical misalignment between the two obtained traces, but the patterns look almost identical. This fact validates our choice of the correlation coefficient as our pattern matching metric, since this metric does not depend on the difference in absolute values, and therefore, the constant misalignment does not affect the results. Figure 1 shows this vertical misalignment between the brown trace obtained from the target and the blue trace obtained from the template device for the same instance of the algorithm.

For our target trace, we compute a multiple of a point $\boldsymbol{P}$ with coordinates

[2] http://www.picotech.com/discontinued/PicoScope5203.html.

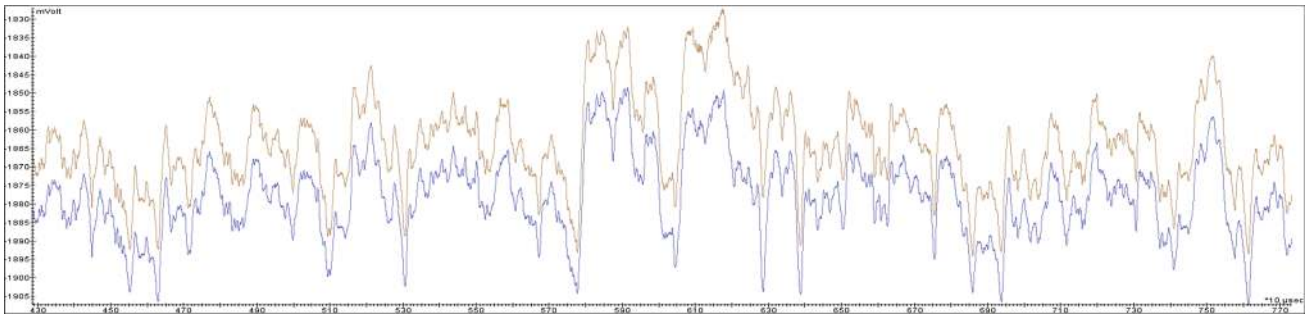

Fig. 1 Similarity between $2\boldsymbol{P}$ on the target card with $2\boldsymbol{P}$ on the templates' card

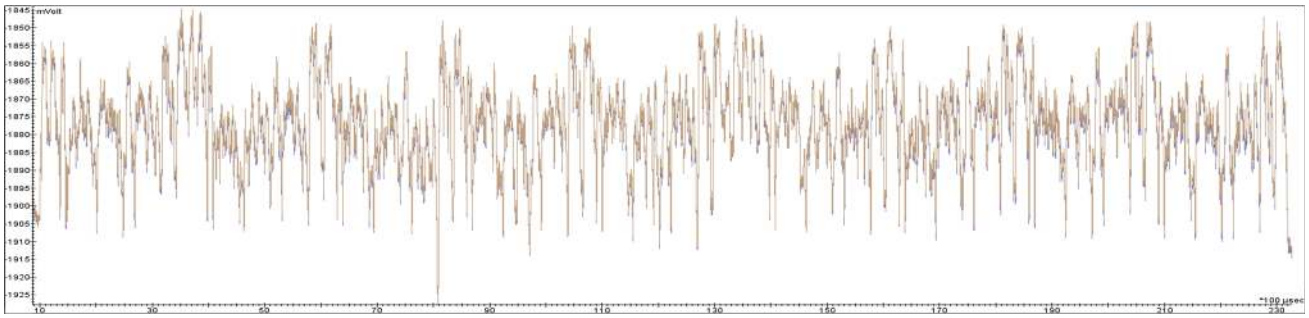

Fig. 2 Difference between $\boldsymbol{P}$ (target trace, *brown*) at second iteration and $2\boldsymbol{P}$ (matching template trace, *blue*) at first iteration. For illustration, both traces are obtained from the same card to avoid vertical misalignment (color figure online)

$$\begin{aligned}
\boldsymbol{P}_x &= \texttt{0x6218E309D40065FCC338B3127F468371}\\
&\quad\ \texttt{82324BD01CE6F3CF81AB44E62959C82A},\\
\boldsymbol{P}_y &= \texttt{0x5501492265E073D874D9E5B81E7F8784}\\
&\quad\ \texttt{8A826E80CCE2869072AC60C3004356E5},\\
\boldsymbol{P}_z &= \texttt{0x0000000000000000000000000000000000}\\
&\quad\ \texttt{0000000000000000000000000000000001},\\
\boldsymbol{P}_t &= \texttt{0x3FC17C25A0F70F2B3113A05A48E6CD8B}\\
&\quad\ \texttt{CD341E229CB10E4833B819EA5D3A8762}.
\end{aligned}$$

We know that the most significant bit of the scalar is 1, so after the first iteration of the double-and-add-always loop the value of $\boldsymbol{R}_0$ is either $2\boldsymbol{P}$ (if the second bit of k is zero) or $3\boldsymbol{P}$ (if the second bit of k is one). We furthermore know from the addition formulas used by the implementation that $\boldsymbol{R}_0 = 2\boldsymbol{P}$ or $\boldsymbol{R}_0 = 3\boldsymbol{P}$ have the following specific representation in extended coordinates:tt

$$\begin{aligned}
2\boldsymbol{P}_x &= \texttt{0xB83008EEB749E519BA5C05E63EDAABA1}\\
&\quad\ \texttt{E2BA0C92037A02796B1D92A636A49746},\\
2\boldsymbol{P}_y &= \texttt{0x910B931F833256DB68C1D2597194A774}\\
&\quad\ \texttt{97C4A9FAD63D042535C511840C51A692},\\
2\boldsymbol{P}_z &= \texttt{0xD098E5677B2A9CCA678238279BFC55B6}\\
&\quad\ \texttt{0A4B5F377438DF015EC2BFCC83B2B922},\\
2\boldsymbol{P}_t &= \texttt{0x180C2E1536BACE17B096A0EE222B0299}\\
&\quad\ \texttt{AAF2CBEE868CEB1D2D74800E735F48D4};
\end{aligned}$$

$$\begin{aligned}
3\boldsymbol{P}_x &= \texttt{0xEAB3BE0B61DEEB0B915B228B3E00376A}\\
&\quad\ \texttt{CB7C487114BCB34CD90A1275BA586422},\\
3\boldsymbol{P}_y &= \texttt{0x2342D54933AFB7E1CA079AE79EC1B9DF}\\
&\quad\ \texttt{DD45D0CB96DE25DF0C4C474C524B6EEC},\\
3\boldsymbol{P}_z &= \texttt{0xA9C7590B5B803C2EAB6BADE97EA9C331}\\
&\quad\ \texttt{1AE83BC98D659AE13A9D4D0AD6F93D2A},\\
3\boldsymbol{P}_t &= \texttt{0x4A9D4C7F687A53B21CFD06DB1400B1EA}\\
&\quad\ \texttt{7AA4434DE904EF2624D001F49B491434}.
\end{aligned}$$

To determine the second bit of the secret scalar k, we generate template traces by inputting exactly those representations of $2\boldsymbol{P}$ and $3\boldsymbol{P}$ and computing the correlation of the first iteration of the template trace with the second iteration of the target trace. At this point, we use that inputs are given in projective representation.

In fact, we will see that the correlation between the correct template trace and the target trace is so much higher than between the wrong template trace and the target trace that just one of the two template traces is sufficient to determine the second bit of k. This is depicted in Figs. 2 and 3; all figures are taken with the six most significant bits of k set to 100110. Figure 2 shows power traces of the second iteration of the target trace (brown) and the first iteration of the $2\boldsymbol{P}$ template trace, i.e., the matching template trace. Figure 3 shows power traces of the second iteration of the target trace

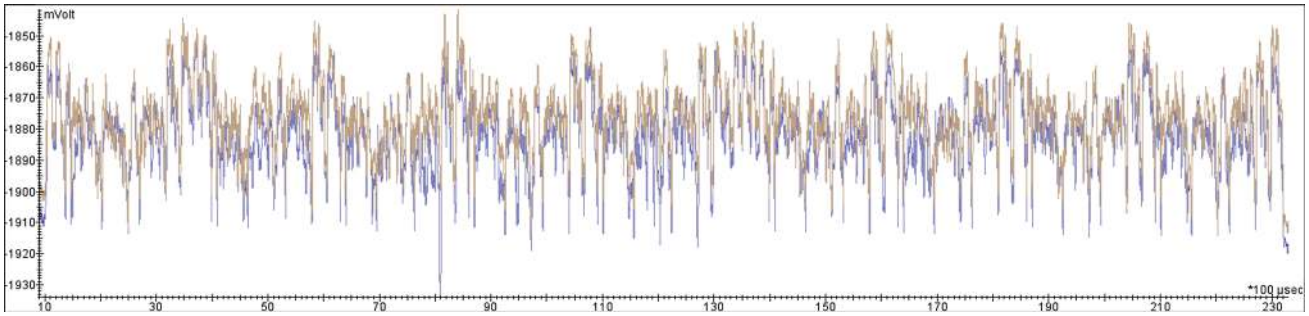

Fig. 3 Difference between $\boldsymbol{P}$ (target trace, *brown*) at second iteration and $3\boldsymbol{P}$ (non-matching template trace, *blue*) at first iteration. For illustration, both traces are obtained from the same card to avoid vertical misalignment (color figure online)

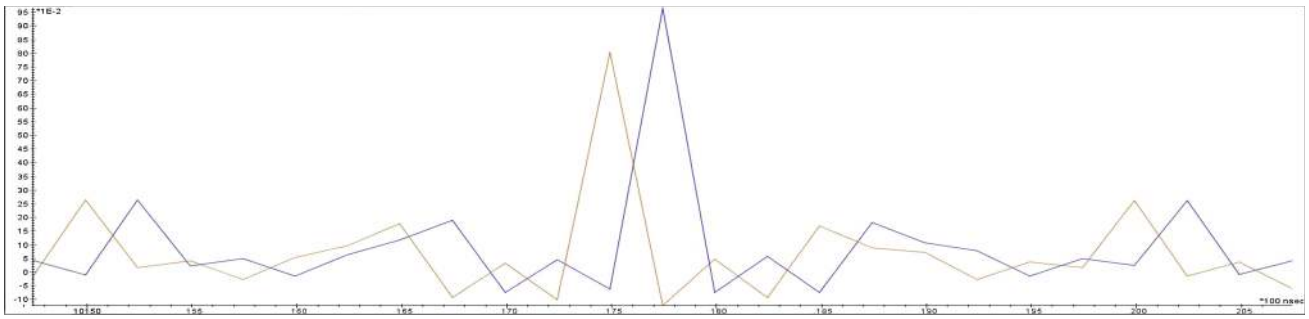

Fig. 4 Pattern matching $2\boldsymbol{P}$ to $\boldsymbol{P}$ (*blue*) and $3\boldsymbol{P}$ to $\boldsymbol{P}$ (*brown*); the target and template traces are obtained from different cards (color figure online)

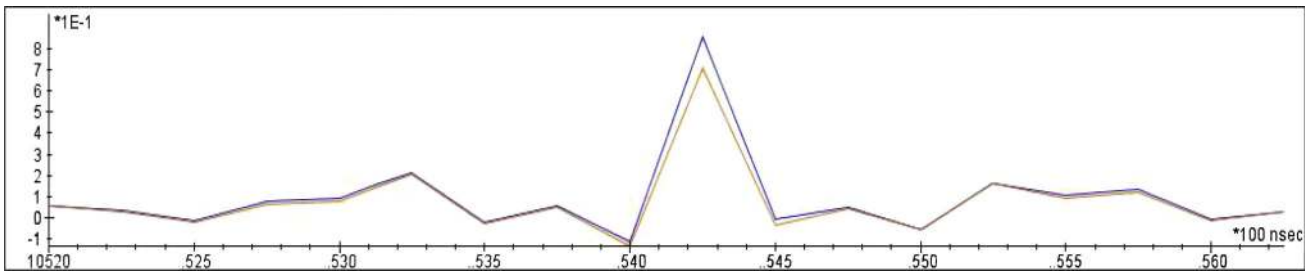

Fig. 5 Pattern matching 2P to P (*blue*) and 3P to P with P (*brown*) obtained for the second most significant scalar bit (color figure online)

(brown) and the first iteration of the $3\boldsymbol{P}$ template trace, i.e., the non-matching template trace.

For validation of our results, we conducted several experiments with different input points from the target card and the template card, and computed the correlation in the obtained power traces. Figure 4 shows the correlation of the template trace (iteration 1) for $2\boldsymbol{P}$ to the target trace (iteration 2) in blue and the correlation of the template trace (iteration 1) for $3\boldsymbol{P}$ to the same target trace (iteration 2) in brown. We notice that the trace obtained from the point $2\boldsymbol{P}$ is almost identical to the pattern obtained from the target trace; as expected, the correlation is at least 97% for all our experiments. On the other hand, the correlation of the target trace with the template trace for $3\boldsymbol{P}$ is at most 83%. To determine the value of one bit, we can thus simply compute only one template trace and decide the value of the targeted bit depending on whether the correlation is above or below a certain threshold set somewhere between 83 and 97%.

The results presented so far are obtained while attacking one single bit of the exponent. When we attack five bits with one acquisition, we observe lower numbers for pattern matching for both the correct and the wrong scalar guess. The correlation results for pattern matching are not so high, mainly due to the noise that is occurring in our setup during longer acquisitions. This follows from the fact that our power supply is not perfectly stable during acquisitions that are longer than 200 ms. However, the difference between correct and wrong assumptions is still remarkable as depicted in Figs. 5, 6, 7, 8 and 9, showing the OTA on five scalar bits

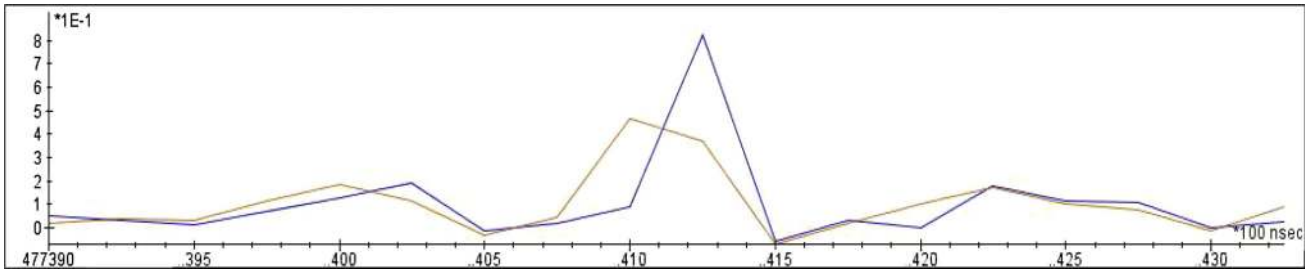

Fig. 6 Pattern matching 4P to P (*blue*) and 5P to P with P (*brown*) obtained for the third most significant scalar bit (color figure online)

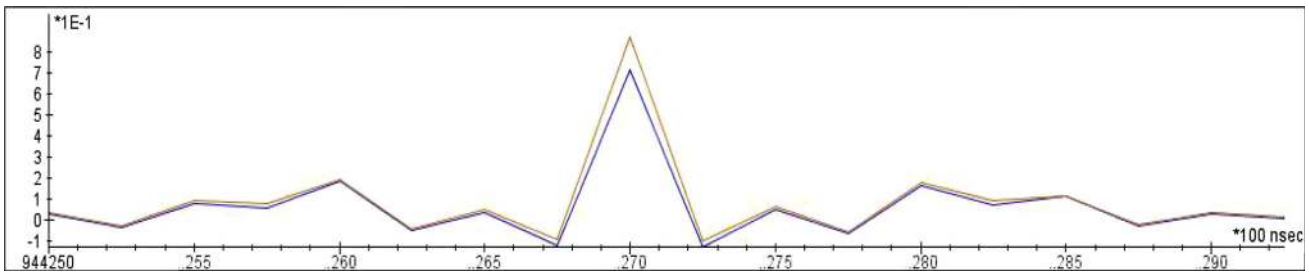

Fig. 7 Pattern matching 8P to P (*blue*) and 9P to P with P (*brown*) obtained for the fourth most significant scalar bit (color figure online)

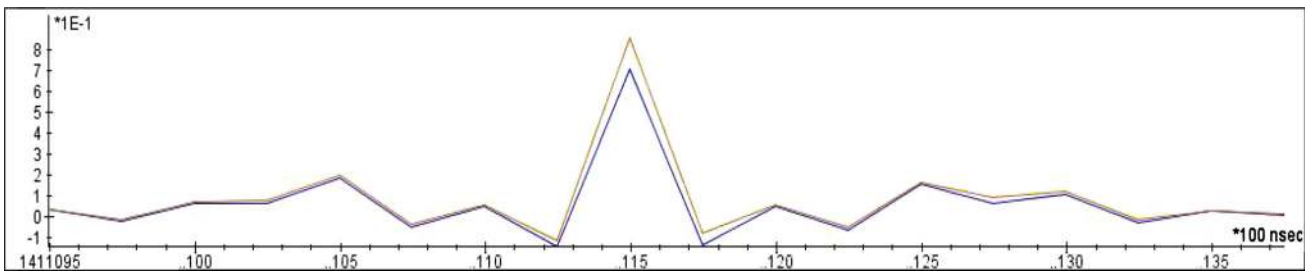

Fig. 8 Pattern matching 18P to P (*blue*) and 19P to P with P (*brown*) obtained for the fifth most significant scalar bit (color figure online)

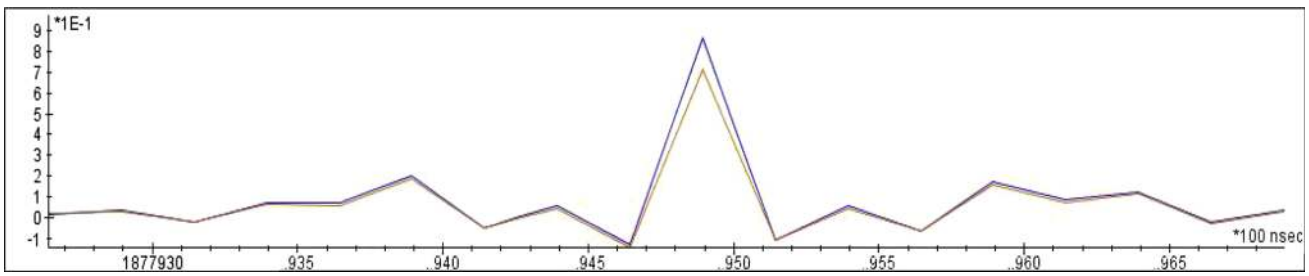

Fig. 9 Pattern matching 38P to P (*blue*) and 39P to P with P (*brown*) obtained for the sixth most significant scalar bit (color figure online)

$k = 100110$ at once.[3] The templates are always acquired during the first iteration and the target trace contains all five iterations; note that the input points for the templates depend on the already recovered exponent bits.

Correct bit assumptions have 84–88% matching patterns, while the correlation for the wrong assumptions drops to 50–72%. Therefore, we can set a threshold for recognizing a bit to be at 80%.

Note that the attack with projective inputs does not make any assumptions on formulas used for elliptic curve addition and doubling. In fact, we carried out the attack for specialized doubling and for doubling that use the same unified addition formulas as addition. The results were similar, and all traces shown above are from the experiments that used unified addition formulas for both addition and doubling.

[3] Observe that the correlation for the incorrect template in Fig. 6 is slightly lower than in the other figures. This can be explained not only by noise, but also by different degrees of similarity between the incorrect inputs for the template traces and the intermediate points for the target trace. Nonetheless, the important fact is that all correlations for the incorrect templates are much lower than the correlations for the correct templates.

4.3 Online template attack with 255-bit projective input

In the previous section, for simplicity, we deliberately ignored the case of coordinates reduction in the field, in order to make the concept of the attack clear. The implementation that we attack, for the sake of efficiency, operates on 256-bit coordinates and not 255-bit coordinates from the field $\mathbb{F}_{2^{255}-19}$. The 256-bit coordinates correspond the coordinates from $\mathbb{F}_{2^{255}-19}$ by applying the modulo $2^{255} - 19$ operation; by using 256 bits, the implementation can save time by not performing some modulo operations.

So far, we assumed that we can send to the card the optimized 256-bit coordinates. It is interesting to examine a more complex attack scenario in which we can only input the 255-bit coordinates. In this section, we show that OTA is successful in this scenario too, a fact that makes OTA a powerful attack technique independent of the prime p of the field used. Fast modular reduction is implemented in [24] by using simple shifts and additions, which are relatively cheap on AVRs.

Our idea is not to attack a whole doubling operation but just a single squaring. More precisely, we show how to perform OTA on the squaring operation of the Z coordinate. First, let us consider the old 256-bit templates. There are two distinct attack cases, namely:

- MSB $= 0$ for the Z projective coordinate (the remaining coordinates can have the most signification bit equal to 1); therefore, the Z coordinate after reduction remains the same.
  In this case, OTA can be applied in a similar way as in Sect. 4.2. We take the old template coordinates and perform a reduction in all the coordinates modulo our prime number $2^{255} - 19$; then, we send those coordinates as input to the card to obtain the new templates.
- The Z coordinate has MSB $= 1$, and therefore, there is a 9 bits difference from the corresponding 256-bit coordinate.
  In this case, the reduced point differs from its 256-bit equivalent in the MSB and in the least significant byte due to the pseudomersenne prime that we use (i.e., $2^{255} - 19$). This case is the most interesting and we will analyze in the remaining part of this section.

We focus on Step D: the computation of Z^2 (see Fig. 11 for the details about the doubling formula that we use); we choose a new Z', such that $\text{MSB}_{Z'} = 0$ and the rest of the bits are the same as the Z coordinate of the old template.

So our new point has only 1 bit difference with the original target trace. For this Z', we recalculate X', Y' and T' using the following equations:

$$X' = X \cdot Z',$$
$$Y' = Y \cdot Z',$$
$$T' = T/Z',$$

where X, Y, Z and T denote the coordinates of the old template.

Figure 10 presents the pattern match between a template trace during computation of $D \leftarrow Z^2$ with template with 1 bit difference, 9 bit difference or wrong template (iteration 1) to the target trace (iteration 2). As expected, the highest peak corresponds to the template with only 1 bit difference, the slightly smaller peak correspond to the 9 bits difference and the lowest peak corresponds to a wrong template. The results obtained from this attack are similar to the previous section, and therefore, we do not present the correlation figures for all 5 bits.

The results above suggest that the templates with 9 bits difference are sufficient for a successful attack although the correlation values are slightly affected. However, for a more noisy setup, 9 incorrect bits may lower the correlation too much. Therefore, we concentrate on an attack that allows only a single bit to be incorrect.

Successful key guesses (for the templates that have at most 1 bit difference) give correlation values between 81 and 86%, while unsuccessful ones are below 76%. The success and unsuccessful rates are different than in Sect. 4.2 because now we concentrate on a single squaring and not the whole doubling. Furthermore, we used different cards for this attack, because one of the cards used for experiments reported in Sect. 4.2 got broken.

4.4 Online template attack with affine input

The attack as explained in the previous sections makes the assumption that the attacker has a full control over the input in projective coordinates. Most implementations of ECC use inputs in affine (or compressed affine) coordinates and internally convert to projective representation. The input is now given as (x, y) and at the beginning of the computation converted to $(x : y : 1 : xy)$. We observe that the points $\boldsymbol{P}$, $2\boldsymbol{P}$ and $3\boldsymbol{P}$ do not have any coordinates in common with the projective representations used internally. Already after the first iteration of the double-and-add-always loop, $Z = 1$ does not hold anymore. Those attacks are more elaborate, since the internal point representation changes at every step of the algorithm.

First we consider an attack on the second most significant bit (which is again set to zero) and the input point $\boldsymbol{P}$ of the target trace with coordinates

$$\boldsymbol{P}_x = \texttt{0x6218E309D40065FCC338B3127F468371}$$
$$\texttt{82324BD01CE6F3CF81AB44E62959C82A},$$
$$\boldsymbol{P}_y = \texttt{0x5501492265E073D874D9E5B81E7F8784}$$
$$\texttt{8A826E80CCE2869072AC60C3004356E5}.$$

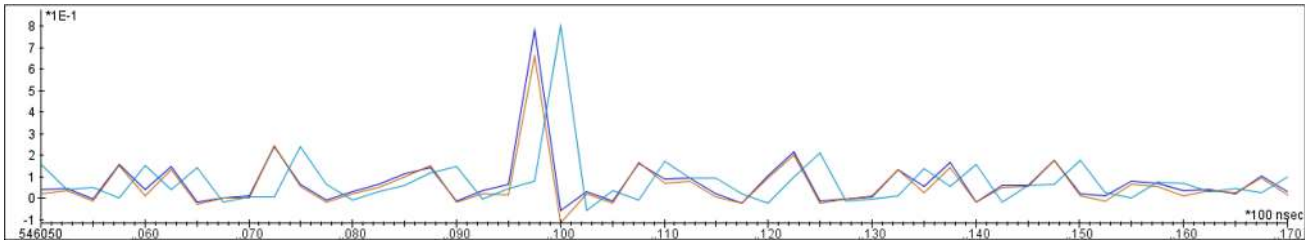

Fig. 10 Pattern matching during computation of D of a template with 1 bit difference (*cyan*), a template with 9 bit difference (*blue*) and a wrong template to the target trace for area of computing D (*brown*) (color figure online)

$$
\begin{aligned}
1: &\quad A \leftarrow (Y_1 - X_1)(Y_2 - X_2)\\
2: &\quad B \leftarrow (Y_1 + X_1)(Y_2 + X_2)\\
3: &\quad C \leftarrow kT_1T_2\\
4: &\quad D \leftarrow 2Z_1Z_2\\
5: &\quad E \leftarrow B - A\\
6: &\quad F \leftarrow D - C\\
7: &\quad G \leftarrow D + C\\
8: &\quad H \leftarrow B + A\\
9: &\quad X3 \leftarrow EF\\
10: &\quad Y3 \leftarrow GH\\
11: &\quad T3 \leftarrow EH\\
12: &\quad Z3 \leftarrow FG
\end{aligned}
$$

Fig. 11 Unified addition/doubling formula from [22]

Choosing the affine versions of $2\boldsymbol{P}$ and $3\boldsymbol{P}$ to generate template traces does not help us now because they do not have any coordinates in common with the projective representations used internally.[4] To successfully perform the attack, we need to modify our approach and take a closer look at the formulas used for point doubling. We illustrate the approach with the unified doubling formula from [22][5] These formulas contain the operations listed in Fig. 11.

The main idea of the attack is to focus on the first multiplication $(Y_1 - X_1)(Y_2 - X_2)$, where in case of a doubling would be $(Y - X)^2$. We give now a detailed description on how to generate the necessary templates for $(Y - X)^2$.

Let us assume that the target trace with the point $\boldsymbol{P}$ is already acquired and that we attack bit b_i, where $0 \leq i < x$. Firstly, depending on already recovered bits of scalar $b_x, \ldots b_{i+1}$ (at the beginning we only know that the most significant bit b_x is 1), the coordinates of P and the bit guess $b_i \in \{0, 1\}$, we can compute the intermediate value $\lambda = Y - X$ that is squared in Step 1 (Fig. 11) during acquisition of the target trace. Secondly, we search for a new point $\boldsymbol{P^i} = (x', y')$ such that $Y' - X' = \lambda$.[6] Such a point $\boldsymbol{P^i}$ cannot always be found on the curve, but we can flip the least significant bit of λ and check whether this point belongs to the curve. If this fails, we flip the bit back and then flip the second least significant bit of λ; we continue this way with subsequent least significant bits until we find a point on the curve. From our experiments, we succeed in finding a point on the curve in a maximum of five trials. Such a point will differ from λ on at most 1 bit (in least significant byte of the coordinate).

Using the method described above, we compute two points $\boldsymbol{P}[\boldsymbol{k_{x-2}} = \boldsymbol{0}]$ and $\boldsymbol{P}[\boldsymbol{k_{x-2}} = \boldsymbol{1}]$, where k_{x-2} indicates the second most significant bit.

$$
\begin{aligned}
\boldsymbol{P}[\boldsymbol{k_{x-2}}=\boldsymbol{0}]_x &= \texttt{0x2B1FDBA73C0BB44A21D59EE599B66E5B}\\
&\quad \texttt{470EA5ADB62777A55254E646F0ADE032},\\
\boldsymbol{P}[\boldsymbol{k_{x-2}}=\boldsymbol{0}]_y &= \texttt{0x03FB65D807F4260BD03B6B58CC706A2D}\\
&\quad \texttt{FC19431688EA79511CFC6524C65AEF7C},\\
\boldsymbol{P}[\boldsymbol{k_{x-2}}=\boldsymbol{1}]_x &= \texttt{0x340C1C83C144EA9C5B707C5081FD770C}\\
&\quad \texttt{F6C2C95FC044604B45ABAEF3F4F3EDAE},\\
\boldsymbol{P}[\boldsymbol{k_{x-2}}=\boldsymbol{1}]_y &= \texttt{0x6D9B33C19315B772941CF4ACE2BEF982}\\
&\quad \texttt{088C51BA4265D2DD78EDE3CA8CE6F852}.
\end{aligned}
$$

Let us assume that, the same as in Sect. 4.2, the six most significant bits of for the scalar k are set to 100110 (recall that the most significant bit is always set to 1). When we compare the trace for $\boldsymbol{P}$ as input at the second iteration to the trace for $\boldsymbol{P}[\boldsymbol{k_x} - \boldsymbol{2} = \boldsymbol{0}]$ at the first iteration during the second squaring operation (computing `A`), we can observe that the two traces are almost identical; see Figure 12 for details.

Figure 13 shows the pattern match between a template trace during computation of $A \leftarrow (Y - X)^2$ with input point $\boldsymbol{P}[\boldsymbol{k_x} - \boldsymbol{1} = \boldsymbol{0}]$ (iteration 1) to the target trace for $\boldsymbol{P}$ (iteration 2) and the pattern match between the template trace (iteration 1) for $\boldsymbol{P}[\boldsymbol{k_x} - \boldsymbol{1} = \boldsymbol{1}]$ to the target trace (iteration 2).

[4] This property follows from the fact that the Z coordinate of $2\boldsymbol{P}$ during the conversion to extended coordinates is always set to `0x01` while the Z coordinate of the point P after being squared in the first iteration of the exponentiation loop does not equal `0x01` with overwhelming probability.

[5] For details, see http://www.hyperelliptic.org/EFD/g1p/auto-twisted-extended-1.html#addition-madd-2008-hwcd-3.

[6] Note that we cannot use $\boldsymbol{P}(X, Y, Z)$ "freely," because now $Z \neq 1$.

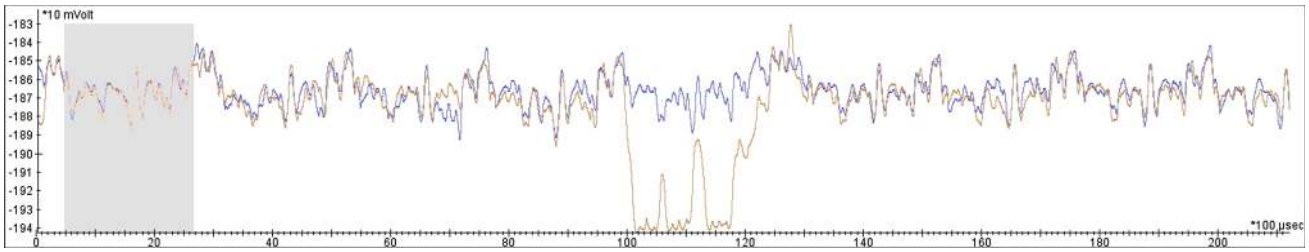

Fig. 12 Comparison between $\boldsymbol{P}$ at the second iteration (*blue*) to $\boldsymbol{P}[\boldsymbol{k_x} - \mathbf{2} = \mathbf{0}]$ at first iteration (*brown*); the area of computing A is marked (color figure online)

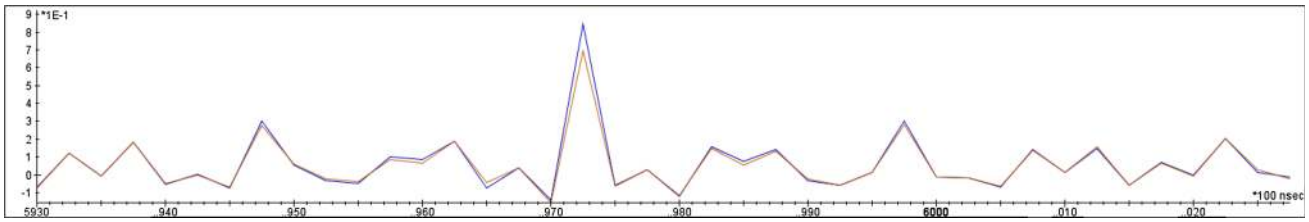

Fig. 13 Pattern matching during the A computation of $\boldsymbol{P}[\boldsymbol{k_x} - \mathbf{2} = \mathbf{0}]$ to $\boldsymbol{P}$ (*blue*) and $\boldsymbol{P}[\boldsymbol{k_x} - \mathbf{2} = \mathbf{1}]$ to $\boldsymbol{P}$ (*brown*) with $\boldsymbol{P}$ obtained for the 2nd most significant scalar bit (color figure online)

We notice that the trace obtained from the point $\boldsymbol{P}[\boldsymbol{k_x} - \mathbf{2} = \mathbf{0}]$ is almost identical to the pattern obtained from the target trace; as expected, the correlation is 86% for the correct key guess and under 73% for the incorrect one.

Since we know the two most significant we can continue the attack for next bits. We repeat the attack for the 5 most significant bit (in total we will know 6 most significant bits since the most significant is always 1).

Using the same method as for computing the points $\boldsymbol{P}[\boldsymbol{k_x} - \mathbf{2} = \mathbf{0}]$ and $\boldsymbol{P}[\boldsymbol{k_x} - \mathbf{2} = \mathbf{1}]$, we compute the point for the 8 subsequent template points. We list them as follows.

$$\boldsymbol{P}[\boldsymbol{k_{x-3}} = \mathbf{0}]_x = \texttt{0xDDBDB790FD617CFBEA18EEFAC3D7E9B5E8C7E0AAD45BA08B63B0B8E99E005747},$$
$$\boldsymbol{P}[\boldsymbol{k_{x-3}} = \mathbf{0}]_y = \texttt{0x0EE63580391125F3438EBB6E384AEE098B8F8333A3A800CF1D76D35513FF595E},$$
$$\boldsymbol{P}[\boldsymbol{k_{x-3}} = \mathbf{1}]_x = \texttt{0xF31755BF0F9F1328EB2E54E207DD4E36907B902AD596D95AA2E5D3138D9E0488},$$
$$\boldsymbol{P}[\boldsymbol{k_{x-3}} = \mathbf{1}]_y = \texttt{0x2B4A5C2C075C5230AD0AD93B793FF8FE8F365F0C3C50A2F7C20F9C0BEDD17B28},$$
$$\boldsymbol{P}[\boldsymbol{k_{x-4}} = \mathbf{0}]_x = \texttt{0x23547DC40EF3F2D370930784C3ACB402F3BEF1CB01A4DD6C5E44AD5CE017BF14},$$
$$\boldsymbol{P}[\boldsymbol{k_{x-4}} = \mathbf{0}]_y = \texttt{0x22D06B73034E069C20F285AE1E67BEF4ACB69B6DCEE39D6A33DE3222C7BDBB84},$$
$$\boldsymbol{P}[\boldsymbol{k_{x-4}} = \mathbf{1}]_x = \texttt{0x507A558593D15CDD3134476723F4E85CFBE11738016C03080284C708E270FC90},$$
$$\boldsymbol{P}[\boldsymbol{k_{x-4}} = \mathbf{1}]_y = \texttt{0x667B8A91051E6551E076114DA2E868C8E15A256663F9D063D4AC42F92DA8A7D6},$$
$$\boldsymbol{P}[\boldsymbol{k_{x-5}} = \mathbf{0}]_x = \texttt{0xF666568414BA6E418A05082EB5901CFAA361C0103DA239A92F1299AA246BAFE6},$$
$$\boldsymbol{P}[\boldsymbol{k_{x-5}} = \mathbf{0}]_y = \texttt{0x1D9931D9308201FB42F54CBC96AC59A777DC17A3A80418E991C8C44CD9169C6C},$$
$$\boldsymbol{P}[\boldsymbol{k_{x-5}} = \mathbf{1}]_x = \texttt{0xF816DE1164F0617FA7A98B03F0804972B66398938A6EDF92DF5A2366B4C3ECF0},$$
$$\boldsymbol{P}[\boldsymbol{k_{x-5}} = \mathbf{1}]_y = \texttt{0x513BCB7EB92B614727BCDCE104A2B5A43F0CC9572999B5035F34743D5FC7BE63},$$
$$\boldsymbol{P}[\boldsymbol{k_{x-6}} = \mathbf{0}]_x = \texttt{0xD5FB7999C3E03EC0CFA16E6E4A2EADE0E8E9E4DDD5439BF7080E60C07EE8045E},$$
$$\boldsymbol{P}[\boldsymbol{k_{x-6}} = \mathbf{0}]_y = \texttt{0x7476658D00EB04256A8265757A6F90D238E5904B9BE42B8DDFB941D9C4050109},$$
$$\boldsymbol{P}[\boldsymbol{k_{x-6}} = \mathbf{1}]_x = \texttt{0x64F1BB3CF1E751164D909F628A58373E647749E70A254D5B1282C0434B3EA80A},$$
$$\boldsymbol{P}[\boldsymbol{k_{x-6}} = \mathbf{1}]_y = \texttt{0x307C54240DC1A35827B597FFED49EE919A9707108779E744C600AF11A1344864}.$$

The pattern match results for the templates are presented in Figs. 14, 15, 16 and 17.

The correlation is 84–87% for the correct key guesses. For the non-matching template point, the correlation value of the matching patterns is at most 73%.

5 Countermeasures and future work

Coron's first and second DPA countermeasures result in scalar or point being blinded to counteract the statistical anal-

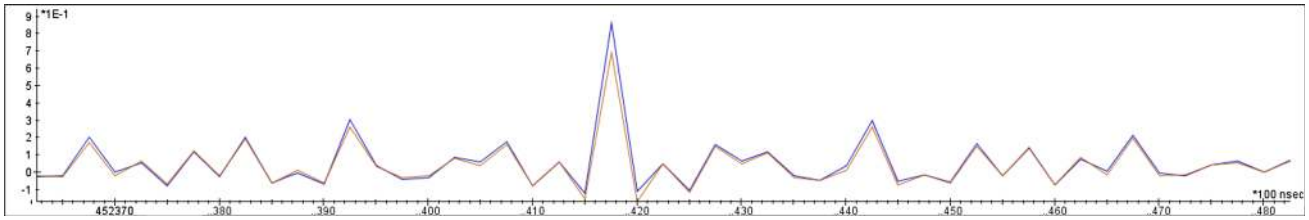

Fig. 14 Pattern matching during computation of A of $\boldsymbol{P}[\boldsymbol{k_{x-3}} = \mathbf{0}]$ to $\boldsymbol{P}$ (*blue*) and $\boldsymbol{P}[\boldsymbol{k_{x-3}} = \mathbf{1}]$ to $\boldsymbol{P}$ (*brown*) with $\boldsymbol{P}$ obtained for the third most significant scalar bit (color figure online)

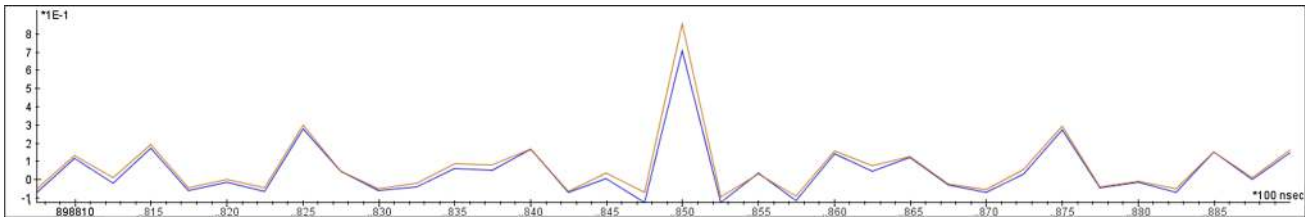

Fig. 15 Pattern matching during computation of A of $\boldsymbol{P}[\boldsymbol{k_{x-4}} = \mathbf{0}]$ to $\boldsymbol{P}$ (*blue*) and $\boldsymbol{P}[\boldsymbol{k_{x-4}} = \mathbf{1}]$ to $\boldsymbol{P}$ (*brown*) with $\boldsymbol{P}$ obtained for the fourth most significant scalar bit (color figure online)

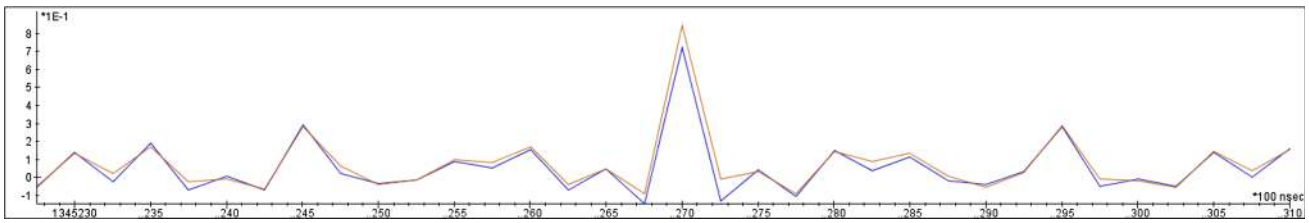

Fig. 16 Pattern matching during computation of A of $\boldsymbol{P}[\boldsymbol{k_{x-5}} = \mathbf{0}]$ to $\boldsymbol{P}$ (*blue*) and $\boldsymbol{P}[\boldsymbol{k_{x-5}} = \mathbf{1}]$ to $\boldsymbol{P}$ (*brown*) with $\boldsymbol{P}$ obtained for the fifth most significant scalar bit (color figure online)

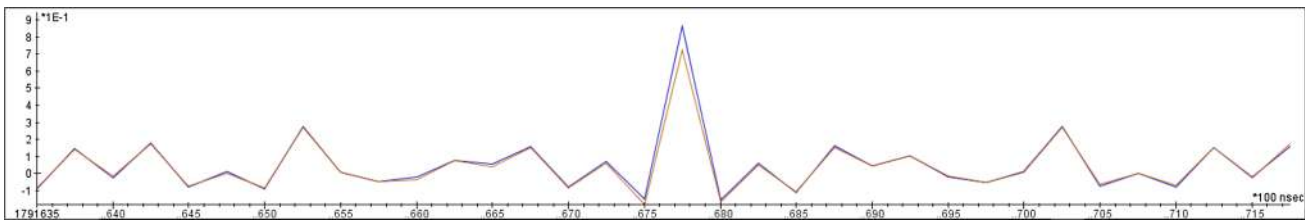

Fig. 17 Pattern Matching during computation of A of $\boldsymbol{P}[\boldsymbol{k_{x-6}} = \mathbf{0}]$ to $\boldsymbol{P}$ (*blue*) and $\boldsymbol{P}[\boldsymbol{k_{x-6}} = \mathbf{1}]$ to $\boldsymbol{P}$ (*brown*) with $\boldsymbol{P}$ obtained for the sixth most significant scalar bit (color figure online)

ysis of DPA attacks [14]. Given that an attacker needs to predict the intermediate state of an algorithm at a given point in time, we can assume that the countermeasures that are used to prevent DPA will also have an effect on the OTA. All proposed countermeasures rely on some kind of randomization, which can be of either a scalar, a point or the algorithm itself. However, if we assume that the attacker has no technical limitations, i.e an oscilloscope with enough memory to acquire the power consumption during an entire scalar multiplication, it would be possible to derive the entire scalar being used from just one acquisition. Therefore, if one depends on scalar blinding [14,32], this method provides no protection against our attack, as the attacker could derive a value equivalent to the exponent.

There are methods for changing the representation of a point, which can prevent OTA and make the result unpredictable to the attacker. Most notably those countermeasures are randomizing the projective coordinates, as proposed in [37, Sec. 6] and randomizing the coordinates through a random field isomorphism as described in [28]. However, inserting a point in affine coordinates and changing to (deterministic) projective coordinates during the execution of the scalar multiplication (compressing and decompressing of a point) do not affect our attack.

We aim exclusively at the doubling operation in the execution of each algorithm. Since most of the blinding techniques are based on the cyclic property of the elliptic curve groups, attacking the addition operation in practice would be an interesting future research topic.

6 Conclusions

In this paper, we presented a new side-channel attack technique, which can be used to recover the private key during a scalar multiplication on ECC with only one target trace and one online template trace per bit. Our attack succeeds against a protected target implementation with unified formulas for doubling and adding and against implementations where the point is given in affine coordinates and changes to projective coordinates representation. By performing our attack on two physically different devices, we showed that key-dependent assignments leak, even when there are no branches in the cryptographic algorithm. This fact enhances the feasibility of OTA and validates our initial claim that one target trace is enough to recover the secret scalar.

References

1. Batina, L., Chmielewski, L., Papachristodoulou, L., Schwabe, P., Tunstall, M.: Online template attacks. In: Progress in Cryptology—INDOCRYPT 2014—15th International Conference on Cryptology in India, New Delhi, India, December 14–17, 2014, Proceedings, pp. 21–36 (2014)
2. Bauer, A., Jaulmes, E., Prouff, E., Wild, J.: Horizontal collision correlation attack on elliptic curves. In: Lange, T., Lauter, K., Lisonek, P. (eds.) Selected Areas in Cryptography—SAC 2013, volume 8282 of LNCS, pp. 553–570. Springer, Berlin (2014)
3. Bauer, A., Jaulmes, E., Prouff, E., Wild, J.: Horizontal collision correlation attack on elliptic curves. In: Lange, T., Lauter, K., Lisonek, P. (eds.) Selected Areas in Cryptography–SAC 2013, volume 8282 of LNCS, pp. 553–570. Springer, Berlin (2014)
4. Bernstein, D.J., Birkner, P., Joye, M., Lange, T., Peters, C.: Twisted Edwards curves. In: Vaudenay, S. (ed.) Progress in Cryptology—AFRICACRYPT 2008, volume 5023 of LNCS, pp. 389–405. Springer, Berlin (2008). http://cr.yp.to/papers.html#twisted
5. Bernstein, D.J., Duif, N., Lange, T., Schwabe, P., Yang, B.-Y.: High-speed high-security signatures. In: Preneel, B., Takagi, T. (eds.) Cryptographic Hardware and Embedded Systems—CHES 2011, volume 6917 of LNCS, pp. 124–142. Springer, Berlin (2011). See also full version [7]
6. Bernstein, D.J., Duif, N., Lange, T., Schwabe, P., Yang, B.-Y.: High-speed high-security signatures. J. Cryptogr. Eng. **2**(2):77–89 (2012). http://cryptojedi.org/papers/#ed25519, see also short version [6]
7. Bernstein, D.J., Lange, T.: Faster addition and doubling on elliptic curves. In: Kurosawa, K. (ed.) Advances in Cryptology—ASIACRYPT 2007, volume 4833 of LNCS, pp. 29–50. Springer, Berlin (2007). http://cr.yp.to/papers.html#newelliptic
8. Bernstein, D.J., Lange, T., Farashahi, R.R.: Binary Edwards curves. In: Oswald, E., Rohatgi, P. (eds.) Cryptographic Hardware and Embedded Systems—CHES 2008, volume 5154 of LNCS, pp. 244–265. Springer, Berlin (2008). http://cr.yp.to/papers.html#edwards2
9. Brier, É., Joye, M.: Weierstraß elliptic curves and side-channel attacks. In: Naccache, D., Paillier, P. (eds.) Public Key Cryptography, volume 2274 of LNCS, pp. 335–345. Springer, Berlin (2002). http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.8.5691&rep=rep1&type=pdf
10. Chevallier-Mames, B., Ciet, M., Joye, M.: Low-cost solutions for preventing simple side-channel analysis: side-channel atomicity. IEEE Trans. Comput. **53**(6), 760–768 (2004)
11. Chevallier-Mames, B., Ciet, M., Joye, M.: Low-cost solutions for preventing simple side-channel analysis: side-channel atomicity. IEEE Trans. Comput. **53**(6), 760–768 (2004)
12. Clavier, C., Feix, B., Gagnerot, G., Roussellet, M., Verneuil, V.: Horizontal correlation analysis on exponentiation. In: Soriano, M., Qing, S., López, J. (eds.) Information and Communications Security, volume 6476 of LNCS, pp. 46–61. Springer, Berlin (2010). http://eprint.iacr.org/2003/237
13. Coron, J.-S.: Resistance against differential power analysis for elliptic curve cryptosystems. In: Koç, Ç.K., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems—CHES'99, volume 1717 of LNCS, pp. 292–302. Springer, Berlin (1999). http://saluc.engr.uconn.edu/refs/sidechannel/coron99resistance.pdf
14. Atmel Corporation. ATMEL AVR32UC Technical Reference Manual. ARM Doc Rev.32002F (2010). http://www.atmel.com/images/doc32002.pdf
15. Dugardin, M., Papachristodoulou, L., Najm, Z., Batina, L., Danger, J.-L., Guilley, S.: 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 (2016)
16. Edwards, H.M.: A normal form for elliptic curves. In: Koç, Ç.K., Paar, C. (eds.) Bulletin of the American Mathematical Society, vol. 44, pp. 393–422 (2007). http://www.ams.org/journals/bull/2007-44-03/S0273-0979-07-01153-6/home.html
17. Fouque, P.-A., Valette, F.: The doubling attack—why upwards is better than downwards. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems—CHES 2003, volume 2779 of LNCS, pp. 269–280. Springer, Berlin (2003). www.ssi.gouv.fr/archive/fr/sciences/fichiers/lcr/fova03.pdf
18. Hanley, N., Kim, H., Tunstall, M.: Exploiting collisions in addition chain-based exponentiation algorithms using a single trace. Cryptology ePrint Archive, Report 2012/485 (2012). http://eprint.iacr.org/2012/485/
19. Hanley, N., Kim, H., Tunstall, M.: Exploiting collisions in addition chain-based exponentiation algorithms using a single trace. In: Nyberg, K. (ed.) Topics in Cryptology—CT-RSA 2015, volume 9048 of LNCS, pp. 431–448. Springer, Berlin (2015)
20. Hanley, N., Kim, H., Tunstall, M.: Exploiting collisions in addition chain-based exponentiation algorithms using a single trace. In: Nyberg, K. (ed.) Topics in Cryptology–CT-RSA 2015, volume 9048 of LNCS, pp. 431–448. Springer, Berlin (2015)
21. Heyszl, J., Mangard, S., Heinz, B., Stumpf, F., Sigl, G.: Localized electromagnetic analysis of cryptographic implementations. In: Dunkelman, O. (ed.) Topics in Cryptology–CT-RSA 2012, volume 7178 of LNCS, pp. 231–244. Springer, Berlin (2012)

22. Homma, N., Miyamoto, A., Aoki, T., Satoh, A., Shamir, A.: Collision-based power analysis of modular exponentiation using chosen-message pairs. In: Oswald, E., Rohatgi, P. (eds.) Cryptographic Hardware and Embedded Systems—CHES 2008, volume 5154 of LNCS, pp. 15–29. Springer, Berlin (2008). http://www.aoki.ecei.tohoku.ac.jp/crypto/pdf/CHES2008_homma.pdf
23. Hutter, M., Schwabe, P.: NaCl on 8-bit AVR microcontrollers. In: Youssef, A., Nitaj, A. (eds.) Progress in Cryptology—AFRICACRYPT 2013, volume 7918 of LNCS, pp. 156–172. Springer, Berlin (2013). http://cryptojedi.org/papers/#avrnacl
24. Järvinen, K., Balasch, J.: Single-trace side-channel attacks on scalar multiplications with precomputations. In: Smart Card Research and Advanced Applications—CARDIS 2016 (2016)
25. Joye, M.: Smart-card implementation of elliptic curve cryptography and DPA-type attacks. In: Quisquater, J.-J., Paradinas, P., Deswarte, Y., El Kalam, A.A. (eds.) Smart Card Research and Advanced Applications VI, volume 135 of IFIP International Federation for Information Processing, pp. 115–125. Springer, Berlin (2004)
26. Joye, M.: Highly regular right-to-left algorithms for scalar multiplication. In: Paillier, P., Verbauwhede, I. (eds.) Cryptographic Hardware and Embedded Systems—CHES 2007, volume 4727 of LNCS, pp. 135–147. Springer, Berlin (2007)
27. Joye, M.: Highly regular right-to-left algorithms for scalar multiplication. In: Paillier, P., Verbauwhede, I. (eds.) Cryptographic Hardware and Embedded Systems—CHES 2007, volume 4727 of LNCS, pp. 135–147. Springer, Berlin (2007). https://www.iacr.org/archive/ches2007/47270135/47270135.pdf
28. Joye, M.: Smart-card implementation of elliptic curve cryptography and DPA-type attacks. In: Quisquater, J.-J., Paradinas, P., Deswarte, Y., El Kalam, A.A. (eds.) Smart Card Research and Advanced Applications VI, volume 135 of IFIP International Federation for Information Processing, pp. 115–125. Springer, Berlin (2004)
29. Joye, M.: Highly regular right-to-left algorithms for scalar multiplication. In: Paillier, P., Verbauwhede, I. (eds.) Cryptographic Hardware and Embedded Systems–CHES 2007, volume 4727 of LNCS, pp. 135–147. Springer, Berlin (2007)
30. Joye, M., Quisquater, J.-J.: Hessian elliptic curves and side-channel attacks. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems–CHES 2001, volume 2162 of LNCS, pp. 402–410. Springer, Berlin (2001)
31. Kocher, P.C.: Timing attacks on implementations of Diffie–Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) Advances in Cryptology—CRYPTO '96, volume 1109 of LNCS, pp. 104–113. Springer, Berlin (1996). http://www.cryptography.com/public/pdf/TimingAttacks.pdf
32. Liardet, P.-Y., Smart, N.P.: Preventing SPA/DPA in ECC systems using the Jacobi form. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) Cryptogaphic Hardware and Embedded Systems—CHES 2001, volume 2162 of LNCS, pp. 391–401. Springer, Berlin (2001)
33. Liardet, P.-Y., Smart, N.P.: Preventing SPA/DPA in ECC systems using the Jacobi form. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) Cryptogaphic Hardware and Embedded Systems–CHES 2001, volume 2162 of LNCS, pp. 391–401. Springer, Berlin (2001)
34. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security). Springer, New York (2007)
35. Montgomery, P.L.: Speeding the Pollard and elliptic curve methods of factorization. Math. Comput. 48(177):243–264 (1987). http://www.ams.org/journals/mcom/1987-48-177/S0025-5718-1987-0866113-7/S0025-5718-1987-0866113-7.pdf
36. De Mulder, E., Hutter, M., Marson, M.E., Pearson, P.: Using Bleichenbacher"s solution to the hidden number problem to attack nonce leaks in 384-bit ECDSA. In: Bertoni, G., Coron, J.S. (eds.) Cryptographic Hardware and Embedded Systems—CHES 2013, volume 8086 of LNCS, pp. 435–452. Springer, Berlin (2013). https://online.tugraz.at/tug_online/voe_main2.getvolltext?pCurrPk=71281
37. Naccache, D., Smart, N.P., Stern, J.: Projective coordinates leak. In: Cachin, C., Camenisch, J. (eds.) Advances in Cryptology—EUROCRYPT 2004, volume 3027 of LNCS, pp. 257–267. Springer, Berlin (2004) https://www.iacr.org/archive/eurocrypt2004/30270258/projective.pdf
38. Özgen, E., Papachristodoulou, L., Batina, L.: Classification algorithms for template matching. In: IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2016, McLean, VA, USA (2016, to appear)
39. Römer, T., Seifert, J.-P.: Information leakage attacks against smart card implementations of the elliptic curve digital signature algorithm. In: Attali, I., Jensen, T. (eds.) Smart Card Programming and Security, volume 2140 of LNCS, pp. 211–219. Springer, Berlin (2001)
40. Schramm, K., Wollinger, T., Paar, C.: A new class of collision attacks and its application to DES. In: Johansson, T. (ed.) Fast Software Encryption, volume 2887 of LNCS, pp. 206–222. Springer, Berlin (2003). https://www.emsec.rub.de/research/publications/new-class-collision-attacks-and-its-application-de/
41. Walter, C.D.: Sliding windows succumbs to Big Mac attack. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems—CHES 2001, volume 2162 of LNCS, pp. 286–299. Springer, Berlin (2001). https://www.comodo.com/resources/research/cryptography/CDW_Ches2001.ps
42. Wenger, E., Korak, T., Kirschbaum, M.: Analyzing side-channel leakage of RFID-suitable lightweight ECC hardware. In: Hutter, M., Schmidt, J.-M. (eds.) Radio Frequency Identification, volume 8262 of LNCS, pp. 128–144. Springer, Berlin (2013). https://online.tugraz.at/tug_online/voe_main2.getvolltext?pCurrPk=71289
43. Witteman, M.F., van Woudenberg, J.G.J., Menarini, F.: Defeating RSA multiply-always and message blinding countermeasures. In Kiayias, A. (ed.) Topics in Cryptology—CT-RSA 2011, volume 6558 of LNCS, pp. 77–88. Springer, Berlin (2011). https://www.riscure.com/benzine/documents/rsacc_ctrsa_final.pdf
44. Yen, S.-M., Ko, L.-C., Moon, S., Ha, J.: Relative doubling attack against montgomery ladder. In: Won, D.H., Kim, S. (eds.) Information Security and Cryptology—ICISC 2005, volume 3935 of LNCS, pp. 117–128. Springer, Berlin (2005). http://islab.hoseo.ac.kr/jcha/paper/ICISC2005.pdf
45. Zhang, Z., Wu, L., Mu, Z., Zhang, X.: A novel template attack on wnaf algorithm of ECC. In: 2014 Tenth International Conference on Computational Intelligence and Security (CIS), pp. 671–675. IEEE (2014)