Online Template Attacks

Lejla Batina[1], Łukasz Chmielewski[2], Louiza Papachristodoulou[1](✉), Peter Schwabe[1], and Michael Tunstall[3]

[1] Digital Security Group, Radboud University Nijmegen, P.O. Box 9010, 6500 GL Nijmegen, The Netherlands
lejla@cs.ru.nl, louiza@cryptologio.org, peter@cryptojedi.org
[2] Riscure BV, Delft, The Netherlands
Chmielewski@riscure.com
[3] Cryptography Research Inc., 425 Market Street, San Francisco, CA 94105, USA
michael.tunstall@cryptography.com

Abstract. In the context of attacking elliptic-curve scalar multiplication with template attacks, one can interleave template generation and template matching to reduce the amount of template traces. This paper enhances the power of this technique by defining and applying the concept of *online template attacks* (OTA); a general attack technique with minimal assumptions for an attacker, who has very limited control over the target device. We show that OTA need only one power consumption trace of a scalar multiplication on the target device; they are thus suitable not only against ECDSA and static Diffie-Hellman, but also against elliptic-curve scalar multiplication in ephemeral Diffie-Hellman. In addition, OTA need only one template trace per scalar bit and they can be applied to almost all scalar-multiplication algorithms. To demonstrate the power of OTA we recover scalar bits of a scalar multiplication using the double-and-add-always algorithm on a twisted Edwards curve running on a smart card with an ATmega163 CPU.

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

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

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. Date: 2014-07-16

W. Meier and D. Mukhopadhyay (Eds.): INDOCRYPT 2014, LNCS 8885, pp. 21–36, 2014.
DOI: 10.1007/978-3-319-13039-2_2

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 [14] and promoted for cryptographic applications by Bernstein and Lange [3], 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 [12,24], cannot be applied directly to ECDSA or ephemeral Diffie-Hellman 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 [1,15,18,33,34,37] and template attacks [28,30,32]. 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 [16]. Wenger et al. in [35] 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.

Previous 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 [34] 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 [36] performed a similar attack on the RSA modular exponentiation in the presence of blinded messages. Clavier et al. introduced in [11] 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 [11] and triangular trace analysis of the exponent in [10]. The first horizontal technique relevant to ECC is the doubling attack, presented by Fouque and Valette in [15]. Homma et al. in [18] proposed a generalization of this attack to binary right-to-left, m-ary, and sliding-window methods. The most recent attack, proposed by Bauer et al. in [1], is a type of horizontal collision correlation attack on ECC, which combines atomicity and randomization techniques.

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 [28] 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 [2,30,32]. This is not possible in the context of ephemeral Diffie-Hellman: 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.

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 Diffie-Hellman. 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 [15], 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 the following:

- 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 some 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.

Online template attacks require only one *target trace* and one *online template trace* per key-bit. We can, therefore, claim that our technique demonstrates 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 [2,32], in order to derive the whole private key from a few bits of multiple ephemeral keys.

Organization of the Paper. This paper is organized as follows. We introduce and explain OTA in Section 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 Section 5. Finally, Section 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 is 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 [21] 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 straight-forward. 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 are used to determine the secret [27]. Therefore, the first condition of our proposed attack is typically fulfilled by all attacks of this kind.

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, [28, 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 traces for inputs $\boldsymbol{P}$ (target trace) and $2\boldsymbol{P}$ (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. 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. We hereby note that the

[1] By similar device we mean the same type of microcontroller running the same algorithm.

template trace is part of the target trace (for instance it corresponds to the first doubling) and it is compared bit-by-bit with the target trace. Therefore, alignment of traces is not necessary.

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 Hamming-weight leakage of key-dependent assignments during the execution of a cryptographic algorithm. 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.

3 Applying the Attack to Scalar-Multiplication Algorithms

3.1 Attacking the Double-and-Add-Always Algorithm

The core idea and feasibility of the attack is demonstrated through an example to 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 target 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 $3\boldsymbol{P}$.

Assuming that the first $(i-1)$ bits of k are known, we can derive the i-th bit by computing the two possible states of $\boldsymbol{R}_0$ after this bit has been treated and in this way recover the key iteratively. Note that only the assignment in the i^{th} 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

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 $\boldsymbol{R}_0$

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 $b \leftarrow 1 - k_i$;
$\boldsymbol{R}_b \leftarrow 2\boldsymbol{R}_b$;
$\boldsymbol{R}_b \leftarrow \boldsymbol{R}_b + \boldsymbol{R}_{k_i}$;
return $\boldsymbol{R}_0$

our original target trace. To decide whether the i^{th} 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 $\boldsymbol{R}_{k_{i+1}}$.

Attacking the right-to-left double-and-add-always algorithm of [21] 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.

3.2 Attacking the Montgomery Ladder

The Montgomery Ladder, initially presented by Montgomery in [29] 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 [23] 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: $\boldsymbol{Q} = k \cdot \boldsymbol{P}$

$\boldsymbol{R}_0 \leftarrow \boldsymbol{P}$;
$\boldsymbol{R}_1 \leftarrow \boldsymbol{2P}$;
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 $R_0 = 3\boldsymbol{P}$.

3.3 Attacking Side-Channel Atomicity

Side-channel atomicity is a countermeasure proposed by Chevallier-Mames et al. [9], 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 [9] 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 [26] and Hessian [22] curves come with a unified group law. Edwards curves [6,7] even have a complete group law. For Weierstrass curves, Brier and Joye suggest an approach for unified addition in [8].

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 experimental setup that we use to perform attacks. Secondly, we present experimental results of an OTA with projective input in Section 4.2; in particular, we present the results when we perform the attack bit-by-bit iteratively or in group of five bits. Finally, Sect. 4.3 presents an OTA with affine input.

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 [13] 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 [19], 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 $d = -(121665/121666)$ and base point

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

For more details on Ed25519 and this specific curve, see [4,5].

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 [19]. This means in particular that points are internally represented in extended coordinates as proposed in [17]. In this coordinate system a point $\boldsymbol{P} = (x, y)$ is represented as $(X : Y : Z : T)$ with $x = X/Z, y = Y/Z, x \cdot y = T/Z$.

4.2 Online Template Attack with Projective Input

In this subsection we describe how to apply an OTA if the input supplied to the scalar multiplication is in projective (or extended) 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 [31, Sec. 6].

The attack targets the output of the doubling operation. We performed pattern matching for our traces as described in Section 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 Section 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

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

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.

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

To determine the second bit of the secret scalar k, we generate template traces by inputting exactly the projective 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.

In fact, from our experiments we observe 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. [3]

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. 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 percentage 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. Correct bit assumptions have $84 - 88\%$ matching patterns, while the percentage for 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 doublings and for doublings that use the same unified addition formulas as addition and obtained similar results.

[3] Figures from experiments and measurements for different points and cards can be found in the full version of the paper in the IACR ePrint archive.

4.3 Online Template Attack with Affine Input

The attack as explained in the previous section makes the assumption that the attacker has 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. We now explain how to adapt the attack to also handle those cases.

The input is now given as (x, y) and at the beginning of the computation converted to $(x : y : 1 : xy)$. However, already after the first iteration of the double-and-add-always loop, $Z = 1$ does not hold anymore. In the following we consider an attack on the second-most significant bit (which is again set to zero) and input point $\boldsymbol{P}$ of the target trace. After one iteration of the double-and-add-always loop, the value of $\boldsymbol{R}_0$ is determined by the value of the second-most significant scalar bit. 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. 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 specialized doubling formulas from [17]. For details, see http://www.hyperelliptic.org/EFD/g1p/auto-twisted-extended-1.html#doubling-dbl-2008-hwcd. These doubling formulas begin with the following operations:

```
A = X^2, B = Y^2, C = 2*Z^2 ...
```

Our idea is not to attack a whole doubling operation but just a single squaring; in the following example we attack the squaring $B = Y^2$. The idea is to use input points $\boldsymbol{Q}$ and $\boldsymbol{R}$ for the template traces, such that the y-coordinate of $\boldsymbol{Q}$ is the same as the Y-coordinate in the internal projective representation of $2\boldsymbol{P}$ and the y-coordinate of $\boldsymbol{R}$ is the same as Y-coordinate of the internal projective representation of $3\boldsymbol{P}$. Unfortunately, such points do not always exist, but our experiments showed that it is sufficient to select points $\boldsymbol{Q}$ and $\boldsymbol{R}$ such that their y-coordinate is *almost the same* as the Y-coordinate of the respective internal projective representation. By almost the same we mean that the y-coordinate is allowed to differ in one bit. This flexibility in choosing the template input allows us to find suitable points with overwhelmingly large probability. When we compare the traces for $\boldsymbol{P}$ as input at the second iteration to the trace for $\boldsymbol{Q}$ at the first iteration during the second squaring operation (computing `B`) then we can observe that the two traces are almost identical; see Figure 1 for details. This figure is taken from an experiment where we have an exact match of the y-coordinate, i.e., we did not have to flip one bit in the expected internal value to find a suitable affine template point.

For validation of our result, we conducted several experiments with different input points using one card (for the sake of simplicity), and found the correlation in the obtained power traces. Let us assume that the scalar is $k = 10$ (let us recall the the most significant bit is always set to 1). Figure 2 shows the pattern match between a template trace during computation of `B` of input point $\boldsymbol{Q}$ (iteration 1) to the target trace for $\boldsymbol{P'}$ (iteration 2) and the pattern match

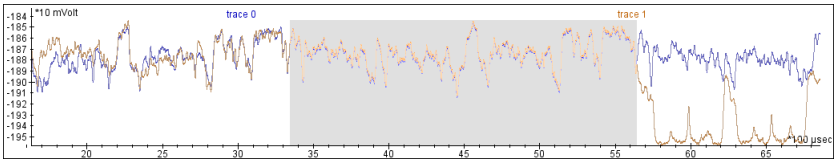

Fig. 1. Comparison between $\boldsymbol{P'}$ at the second iteration to $\boldsymbol{Q}$ at first iteration; the area of computing B is highlighted

between the template trace (iteration 1) for $\boldsymbol{R}$ to the target trace (iteration 2). We notice that the trace obtained from the point $\boldsymbol{Q}$ is almost identical to the pattern obtained from the target trace; as expected, the correlation is at least 96% for exactly matching y-coordinate of the template point and >91% for almost matching y-coordinate. For the non-matching template point the pattern match is at most 84%.

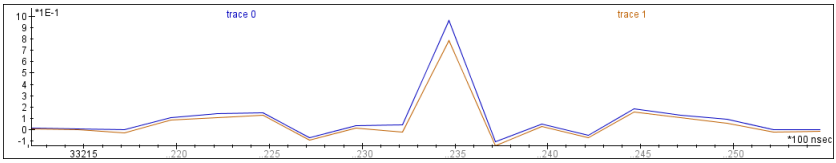

Fig. 2. Pattern Matching $\boldsymbol{Q}$ to $\boldsymbol{P'}$ and $\boldsymbol{R}$ to $\boldsymbol{P'}$ coming from the same card

5 Countermeasures and Future Work

Coron's first and second DPA countermeasures result in scalar or point being blinded to counteract the statistical analysis of DPA attacks [12]. 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 [12,25], this method provides no protection against our attack, as the attacker could derive a value equivalent to the exponent used.

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 randomization and randomizing the coordinates through a random field isomorphism as described in [20]. 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), does 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 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. Bauer, A., Jaulmes, E., Prouff, E., Wild, J.: Horizontal collision correlation attack on elliptic curves. In: Lange, T., Lauter, K., Lisoněk, P. (eds.) SAC 2013. LNCS, vol. 8282, pp. 553–570. Springer, Heidelberg (2014)
2. Benger, N., van de Pol, J., Smart, N.P., Yarom, Y.: "Ooh aah... just a little bit": A small amount of side channel can go a long way. Cryptology ePrint Archive, Report 2014/161 (2014)
3. Bernstein, D.J., Birkner, P., Joye, M., Lange, T., Peters, C.: Twisted Edwards curves. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 389–405. Springer, Heidelberg (2008)
4. Bernstein, D.J., Duif, N., Lange, T., Schwabe, P., Yang, B.-Y.: High-speed high-security signatures. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 124–142. Springer, Heidelberg (2011)
5. Bernstein, D.J., Duif, N., Lange, T., Schwabe, P., Yang, B.Y.: High-speed high-security signatures. Journal of Cryptographic Engineering **2**(2), 77–89 (2012)
6. Bernstein, D.J., Lange, T.: Faster addition and doubling on elliptic curves. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 29–50. Springer, Heidelberg (2007)
7. Bernstein, D.J., Lange, T., Rezaeian Farashahi, R.: Binary Edwards curves. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 244–265. Springer, Heidelberg (2008)
8. Brier, E., Joye, M.: Weierstraß elliptic curves and side-channel attacks. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 335–345. Springer, Heidelberg (2002)
9. Chevallier-Mames, B., Ciet, M., Joye, M.: Low-cost solutions for preventing simple side-channel analysis: Side-channel atomicity. IEEE Transactions on Computers **53**(6), 760–768 (2004)
10. Clavier, C., Feix, B., Gagnerot, G., Giraud, C., Roussellet, M., Verneuil, V.: ROSETTA for single trace analysis. In: Galbraith, S., Nandi, M. (eds.) INDOCRYPT 2012. LNCS, vol. 7668, pp. 140–155. Springer, Heidelberg (2012)
11. Clavier, C., Feix, B., Gagnerot, G., Roussellet, M., Verneuil, V.: Horizontal correlation analysis on exponentiation. In: Soriano, M., Qing, S., López, J. (eds.) ICICS 2010. LNCS, vol. 6476, pp. 46–61. Springer, Heidelberg (2010)

12. Coron, J.-S.: Resistance against differential power analysis for elliptic curve cryptosystems. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 292–302. Springer, Heidelberg (1999)
13. Atmel Corporation. ATMEL AVR32UC technical reference manual. ARM Doc Rev. 32002F (2010)
14. 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)
15. Fouque, P.-A., Valette, F.: The doubling attack – *Why upwards is better than downwards*. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 269–280. Springer, Heidelberg (2003)
16. 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)
17. Hisil, H., Wong, K.K.-H., Carter, G., Dawson, E.: Twisted Edwards curves revisited. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 326–343. Springer, Heidelberg (2008)
18. 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.) CHES 2008. LNCS, vol. 5154, pp. 15–29. Springer, Heidelberg (2008)
19. Hutter, M., Schwabe, P.: NaCl on 8-Bit AVR microcontrollers. In: Youssef, A., Nitaj, A., Hassanien, A.E. (eds.) AFRICACRYPT 2013. LNCS, vol. 7918, pp. 156–172. Springer, Heidelberg (2013)
20. 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. IFIP, vol. 135, pp. 115–125. Springer, Heidelberg (2004)
21. Joye, M.: Highly regular right-to-left algorithms for scalar multiplication. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 135–147. Springer, Heidelberg (2007)
22. Joye, M., Quisquater, J.-J.: Hessian elliptic curves and side-channel attacks. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 402–410. Springer, Heidelberg (2001)
23. Joye, M., Yen, S.M.: The Montgomery powering ladder. In: Kaliski, B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 291–302. Springer, Heidelberg (2002)
24. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
25. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
26. Liardet, P., Smart, N.P.: Preventing SPA/DPA in ECC systems using the jacobi form. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 391–401. Springer, Heidelberg (2001)
27. Mangard, S., Oswald, E., Popp, T.:Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security). Springer New York Inc. (2007)
28. Medwed, M., Oswald, E.: Template attacks on ECDSA. In: Chung, K.-I., Sohn, K., Yung, M. (eds.) WISA 2008. LNCS, vol. 5379, pp. 14–27. Springer, Heidelberg (2009)
29. Montgomery, P.L.: Speeding the Pollard and elliptic curve methods of factorization. Mathematics of Computation **48**(177), 243–264 (1987)

30. 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.) CHES 2013. LNCS, vol. 8086, pp. 435–452. Springer, Heidelberg (2013)
31. Naccache, D., Smart, N.P., Stern, J.: Projective coordinates leak. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 257–267. Springer, Heidelberg (2004)
32. Römer, T., Seifert, J.-P.: Information leakage attacks against smart card implementations of the elliptic curve digital signature algorithm. In: Attali, S., Jensen, T. (eds.) E-smart 2001. LNCS, vol. 2140, pp. 211–219. Springer, Heidelberg (2001)
33. Schramm, K., Wollinger, T., Paar, C.: A new class of collision attacks and Its application to DES. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 206–222. Springer, Heidelberg (2003)
34. Walter, C.D.: Sliding windows succumbs to Big Mac attack. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 286–299. Springer, Heidelberg (2001)
35. Wenger, E., Korak, T., Kirschbaum, M.: Analyzing side-channel leakage of RFID-suitable lightweight ECC hardware. In: Hutter, M., Schmidt, J.-M. (eds.) RFIDsec 2013. LNCS, vol. 8262, pp. 128–144. Springer, Heidelberg (2013)
36. Witteman, M.F., van Woudenberg, J.G.J., Menarini, F.: Defeating RSA multiply-always and message blinding countermeasures. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 77–88. Springer, Heidelberg (2011)
37. Yen, S.-M., Ko, L.-C., Moon, S.-J., Ha, J.C.: Relative doubling attack against montgomery ladder. In: Won, D.H., Kim, S. (eds.) ICISC 2005. LNCS, vol. 3935, pp. 117–128. Springer, Heidelberg (2006)

http://www.springer.com/978-3-319-13038-5

Progress in Cryptology -- INDOCRYPT 2014
15th International Conference on Cryptology in India, New Delhi, India, December 14-17, 2014, Proceedings
Meier, W.; Mukhopadhyay, D. (Eds.)
2014, XXVI, 444 p. 80 illus., Softcover
ISBN: 978-3-319-13038-5