Practical Results of ECC Side Channel Countermeasures on an ARM Cortex M3 Processor

Jacek Samotyja
Bonn-Rhein-Sieg University of Applied Sciences
Grantham-Allee 20
53757 Sankt Augustin, Germany
jacek.samotyja@h-brs.de

Kerstin Lemke-Rust
Bonn-Rhein-Sieg University of Applied Sciences
Grantham-Allee 20
53757 Sankt Augustin, Germany
kerstin.lemke-rust@h-brs.de

ABSTRACT

This paper presents implementation results of several side channel countermeasures for protecting the scalar multiplication of ECC (Elliptic Curve Cryptography) implemented on an ARM Cortex M3 processor that is used in security sensitive wireless sensor nodes. Our implementation was done for the ECC curves P-256, brainpool256r1, and Ed25519. Investigated countermeasures include Double-And-Add Always, Montgomery Ladder, Scalar Randomization, Randomized Scalar Splitting, Coordinate Randomization, and Randomized Sliding Window. Practical side channel tests for SEMA (Simple Electromagnetic Analysis) and MESD (Multiple Exponent, Single Data) are included. Though more advanced side channel attacks are not evaluated, yet, our results show that an appropriate level of resistance against the most relevant attacks can be reached.

CCS Concepts

•**Security and privacy** → **Public key (asymmetric) techniques; Side-channel analysis and countermeasures;** *Embedded systems security;* •**Computer systems organization** → **Embedded software;** *Sensor networks;*

Keywords

Elliptic Curve Cryptography, Scalar Multiplication, Side Channel Countermeasures, ARM Cortex M3 Processor.

1. INTRODUCTION

Implementations of cryptographic protocols are known to be vulnerable to side channel analysis, e.g. [24, 25, 27]. Measuring the timing, the power consumption or the EM emanation of a cryptographic operation may expose secret or private key data. Besides side channel analysis, fault attacks are another threat to cryptographic implementations and many attacks are now known to various crypto systems, e.g., [20].

Tis'16, October 24 2016, Vienna, Austria
© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ISBN 978-1-4503-4575-0/16/10...$15.00
DOI: http://dx.doi.org/10.1145/2996366.2996371

Elliptic Curve Cryptography (ECC) is becoming the most relevant public-key cryptosystem and seems to replace RSA in the long-run because of its advantage regarding computation speed and key length. It finds extensive deployment in the development of new smart security systems such as electronic passports and vehicular Ad-Hoc networks. Private keys are used for the digital signature of messages (ECDSA), the key establishment (ECDH), and decryption (ECIES) [17].

Some applications in the Internet of Things (IoT) such as vehicular Ad-Hoc networks have very demanding timing bounds for the completion of a cryptographic operation that can only be fulfilled by parallelized hardware implementations. In other IoT applications such as Ambient Assistant Living timing constraints for wireless sensor nodes are much more relaxed, and waiting one second for a rarely executed public key crypto operation is acceptable.

Regarding side channel attacks, the most critical ECC operation is the scalar multiplication (*aka* point multiplication) that computes $d\mathbf{P}$ where d is an integer and $\mathbf{P}$ is a point on an elliptic curve E defined over a field $\mathbb{F}_p$. In either case, an evaluation of the level of resistance to side channel and fault attacks is necessary as simple power analysis attacks such that visually inspecting one power trace of a scalar multiplication with a private-key can compromise the entire private parameter d, cf.[17].

The minimum requirement needs to be that the implementation shall prevent a simple read-out of private keys using one measured trace. This may be sufficient for ECDSA wherin the integer d used for the scalar multiplication is a random number that changes in each run, but not for ECDH and ECIES decryption where the attacker can choose the base point and analyze multiple executions with the same integer d. Typically, an appropriate security level requires that countermeasures are integrated in the cryptographic implementation. A valuable state-of-the-art overview on known attacks and countermeasures is given in [13].

In this paper we show implementation results of the scalar multiplication using three curves over prime fieds: the NIST P-256 curve [31], the Brainpool brainpoolP256r1 curve [1], and Bernstein's domain parameter from the implementation Ed25519 [6]. We denote this domain parameter as Ed25519. The implementation was done on an ARM Cortex M3 Processor that is also used in higher quality sensor nodes for security applications. We implemented several side channel countermeasures that are practically tested for its resistance on the most relevant attacks. We provide performance results and side channel susceptibilities for our implementation. We note that the most critical attacks are counteracted. This is an improvement over lightweight ECC libararies such as TinyECC which were not designed for side channel resistance and turned out to be seriously exploitable when tested [32].

We note that an efficient combination of multiple countermeasures that resists all known passive and active attacks is still an open research question. We hope that our work can serve as a step forward towards building an ECC library for lightweight applications with configurable security parameters for side channel countermeasures.

2. ELLIPTIC CURVE CRYPTOGRAPHY

An elliptic curve E is a non-singular curve defined by the following equation over an algebraic field $\mathbb{K}$:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \qquad (1)$$

with the coefficients $a_1, a_2, a_3, a_4, a_6 \in \mathbb{K}$. This equation is known as *Weierstrass equation*.

In this work we deal with elliptic curves defined over a finite field $\mathbb{F}_p$ where $p > 3$ is prime and the discriminant $\delta := -16(4a^3 + 27b^2)$ is not null. The elliptic curve is given by the *short Weierstrass equation*:

$$E : y^2 = x^3 + ax + b\,. \qquad (2)$$

This elliptic curve consists a set of discrete points $\mathbf{P} : (x, y)$. The set of points is expressed by:

$$E(\mathbb{F}_p) = (x,\ y) \in \mathbb{F}_p \times \mathbb{F}_p : y^2 = x^3 + ax + b \cup \infty\,. \qquad (3)$$

∞ denotes the point in infinity [29]. Hasse's theorem provide an estimate of the number of points on E over $\mathbb{F}_p$. The estimated points is given by [29]:

$$p + 1 - 2\sqrt{p} \leq \#E(\mathbb{F}_p) \leq p + 1 + 2\sqrt{p}\,. \qquad (4)$$

A high amount of points increased the level of protection. In affine coordinates, an addition of two points $\mathbf{P}$ and $\mathbf{Q}$ is given by the following equation [29]:

$$\begin{array}{lll} i) & -\mathbf{P} = (x_1,\ -y_1), & \\ ii) & \mathbf{P} + \mathbf{Q} = \infty, & \text{if } \mathbf{P} = -\mathbf{Q}, \\ iii) & \mathbf{P} + \mathbf{Q} = (x_3,\ y_3), & \text{else } \mathbf{P} \neq \mathbf{Q}, \\ & x_3 = \lambda^2 - x_1 - x_2, & \\ & y_3 = \lambda(x_1 - x_3) - y_1, & \text{with} \\ & \lambda = \begin{cases} \frac{y_1 - y_2}{x_1 - x_2} & if\ \mathbf{P} \neq \mathbf{Q} \\ \frac{3x_1^2 + a}{2y_1} & if\ \mathbf{P} = \mathbf{Q} \end{cases} & \end{array} \qquad (5)$$

The elliptic curve can also be represented in projective coordinates (X, Y, Z). The projective equation of the elliptic curve is

$$Y^2Z = X^3 + aXZ^2 + bZ^3\,. \qquad (6)$$

The elliptic curve in *Weierstrass form* is just one representative of elliptic curves. There are many other curves such as Jacobian-, Hessian-, Montgomery-, and Edwards- [9]. The Twisted Edwards (7) curve is defined by the equation

$$E :\ ax^2 + y^2 = 1 + dx^2y^2 \qquad (7)$$

wherin a and d are elements of the underlying field. The parameter $a = 1$ yields an Edwards curve. One advantage of using Edwards curve is their improved intrinsic resistance to simple side channel attacks because of a strongly unified formula for both point addition and point doubling [7]:

$$(x_1, y_1)(x_2, y_2) \mapsto \left(\frac{x_1y_2 + y_1x_2}{1 + dx_1x_2y_1y_2}, \frac{y_1y_2 - ax_1x_2}{1 - dx_1x_2y_1y_2} \right)\,. \qquad (8)$$

Transformation between an elliptic cuve in *short Weierstrass form* and *Twisted Edwards form* exists. These rules are well described in [9, 30, 7].

3. SIDE CHANNEL ANALYSIS ON ECC

Private ECC keys are used for the digital signature of messages (ECDSA), key establishment (ECDH), and decryption (ECIES) [17]. All these schemes compute the scalar multiplication with a known base point and a secret scalar. Adversary capabilities, however, differ:

- For ECDSA, the scalar is a secret random number that is used only once and the base point is the base point of the elliptic curve.
- ECDH can be used with static or one-time key pairs, accordingly, the scalar multiplication with the same scalar is computed multiple times or only once and the base point can be chosen by the attacker.
- ECIES decryption can typically be invoked multiple times with the same scalar and chosen base points.

As result, for ECDSA, the adversary has to compromise the scalar with one single measurement, while many measurements of a secret scalar can be used in ECDH and ECIES decryption.

Computation of the scalar multiplication is efficiently done using the Double-And-Add algorithm shown in Algorithm 1 that is a left-to-right binary multiplication algorithm [4, 12].

Algorithm 1 Double-And-Add algorithm

```
Require: elliptic curve point $\mathbf{P}$ and a positive integer $d = (1, d_{t-1}, ..., d_1, d_0)_2$
Ensure: $\mathbf{Q} = d\mathbf{P}$
1: $\mathbf{Q} \leftarrow \mathbf{P}$;
2: for $i$ from $t-1$ to 0 do
3:     $\mathbf{Q} \leftarrow 2\,\mathbf{Q}$;
4:     If $d_i = 1$ then $\mathbf{Q} \leftarrow \mathbf{Q} + \mathbf{P}$;
5: end for
6: Return $\mathbf{Q}$;
```

Coron [12] has pointed out that a naive implementation of Algorithm 1 is vulnerable if the point doubling (aka *double* operation) and point addition (aka *add* operation) can be visually distinguished in the power measurements of a scalar multiplication. Experimental results for distinguishing point doubling and point addition are well known [17] and constitute the most serious side channel threat. These are simple side-channel attacks, either known as Simple Power Analysis (SPA) or Simple ElectroMagnetic Analysis (SEMA) depending on the measurement process. Besides this, various other attacks are published in the previous years. An overview on known side-channel attacks on the scalar multiplication and countermeasures can be found in [13, 22]. Of recent importance are horizontal attacks that compare whether the same operand is repeatedly used in the same measurement trace of a scalar multiplication, e.g., the doubling attack [14]. Refined Power Analysis (RPA) [15] and Zero-Value Point Attack (ZPA) [3] exploit the leakage of points with a coordinate of zero.

In [28] a powerful technique called "Multiple-Exponent, Single-Data" (MESD) was introduced for an exponentiation algorithm using the 'square-multiply' algorithm and it is directly applicable to the scalar multiplication, see also [14]. Practical MESD results for elliptic curves are shown in [16, 32]. This attack can be seen as a simple form of a template attack [11] that requires a test device with an identical implementation that additionally allows to set the scalar. MESD does a visual inspection of the difference trace of the measurement curve with the secret scalar and the measurement curve with the hypothesized scalar, both using the same base point. MESD checks the timing length of the template matching. In more detail, to attack the i-th bit of the scalar the adversary asks the test device to compute the scalar multiplication with the i-th bit set to one. If the data dependencies vanish in the difference trace for some subsequent double and add operations the guess is correct. Otherwise, if significant differences show up again after one subsequent double operation this indicates divergent processed data and the guess is wrong, i.e., the next secret scalar bit is zero. One measurement trace of the secret scalar can be sufficient for analyzing the difference signal if the data dependencies are sufficiently strong. Online Template Attacks [5] follow a very similar approach.

For side channel countermeasures [13] lists the countermeasures Indistinguishable Point Addition Formulae, Double-And-Add Always, Montgomery Ladder, Randomized Scalar Splitting, Scalar Randomization, Base Point Blinding, Randomized Projective Coordinates, Randomized EC/Field Isomorphisms. According to [13] none of them is effective against all known attacks. Indistinguishable Point Addition Formulae, Double-And-Add Always and Montgomery Ladder are used to prevent SPA/SEMA and timing attacks. The remaining countermeasures are designed to prevent template attacks, DPA/DEMA, Comparative SCA, RPA/ZPA but for each of them at least one attack is known.

4. OUR ECC LIBRARY SECLAB_ECC

We implemented the NIST P-256 curve [31], the Brainpool brainpoolP256r1 curve [1], and Ed25519 [6]. The first two elliptic curves are in the *short Weierstrass form* and differ in the reduction function. The curve P-256 uses the fast reduction and the brainpoolP256r1 the Barrett reduction. The Curve25519 is a *Twisted Edwards curve.* All three curves are implemented in affine and projective coordinates.

For the underlying finite field arithmetic we implemented a standard scalar multiplication with the binary method, cf. Algorithm 1. The finite field operation addition (algo. 2.5), subtraction (algo. 2.6), multiplication (algo. 2.9), reduction (algo. 2.14) and fast reduction (algo. 2.19) are taken from [17]. The division algorithm was taken from [33]. For P-256 and brainpoolP256r1 the algorithms for the point addition and doubling in affine coordinates are implemented according to (5). In projective coordinates we use the algorithm "add-2007-bl" for the addition and the algorithm "dbl-2007-bl" for point doubling, both are from [8]. The Twisted Edwards curve is implemented using the algorithm "add-2008-bbjlp" from [7].

The complete source code was written in programming language C for an ARM Cortex M3 processor and is available at [2].

4.1 Implemented Countermeasures

Our choice of countermeasures is based on previous state-of-the-art surveys [13, 22] and practical considerations for sensor nodes. We aim to study lightweight countermeasures that prevent the most relevant simple attacks. We evaluate each countermeasure on its own. We acknowledge that all of them have known weaknesses that need to be fixed if the weakness can be exploited in a given ECC protocol, usually by combinations of countermeasures. We do not claim that any single countermeasure on its own is sufficient to the battery of all known passive and active attacks. Moreover, the applicability of each attack needs to be analyzed in the given ECC protocol. An efficient combination of multiple countermeasures that resists all known passive and active attacks is still an open research question.

4.1.1 *Double-And-Add Always*

This countermeasure carries out the add operation in each iteration of the Double-And-Add Algorithm in Algorithm 1 in order to make SPA/SEMA attacks more difficult [22, 27, 13, 10, 12]. Its drawback is the presence of a pseudo operation which can be identified, e.g., using fault attacks and the absence of data randomisation.

Algorithm 2 Double-And-Add Always

```
Require: A point $\mathbf{P}$ and a positive integer $d = (1, d_{t-1}, d_{t-2}, ..., d_1, d_0)_2$
Ensure: $\mathbf{Q} = d\mathbf{P}$
1: $\mathbf{Q} \leftarrow \mathbf{P}$;
2: for $i$ from $t-1$ to 0 do
3:     $it_0 \leftarrow 0$;
4:     $it_1 \leftarrow 1 - d_i$;
5:     $\mathbf{Q}[it_0] \leftarrow 2\,\mathbf{Q}[it_0]$;
6:     $\mathbf{Q}[it_1] \leftarrow \mathbf{Q}[it_1] + \mathbf{P}$;
7: end for
8: Return $\mathbf{Q}[0]$;
```

4.1.2 *Montgomery Ladder*

The Montgomery Ladder [22, 27, 13] aims to protect from SPA/SEMA attacks. In contrast to the Double-And-Add Always countermeasure, there is no dummy operation.

Algorithm 3 Montgomery Ladder

```
Require: A positive integer $d = (1, d_{t-1}, d_{t-2}, ..., d_1, d_0)_2$ and a point $\mathbf{P}$
Ensure: $\mathbf{Q}_0 = d\mathbf{P}$
1: $\mathbf{Q}_0 \leftarrow \mathbf{P}$;
2: $\mathbf{Q}_1 \leftarrow 2\mathbf{P}$;
3: for $i = n - 2$ down to 0 do
4:     $\mathbf{Q}_{1-d_i} \leftarrow \mathbf{Q}_0 + \mathbf{Q}_1$;
5:     $\mathbf{Q}_{d_0} \leftarrow 2\mathbf{Q}_{d_1}$;
6: end for
7: Return $\mathbf{Q}_0$;
```

4.1.3 Scalar Randomization

Scalar Randomization was first proposed by [12]. This countermeasure randomly selects a scalar $d' = d + k \cdot \epsilon$, wherin k is the random value and ϵ is the number of points on the elliptic curve. This is possible because the multiplication with a multiple of the group order ϵ results in the neutral element. Therefore it holds $d\mathbf{P} = d'\mathbf{P}$.

Algorithm 4 Scalar Randomization

```
Require: elliptic curve point $\mathbf{P}$, a positive integer $d = (1, d_{t-1}, ..., d_1, d_0)_2$, and the number of points on the elliptic curve $\epsilon$
Ensure: $\mathbf{Q} = d\mathbf{P}$
1: $k \in_R \{0,1\}^n$;
2: $d' = d + k \cdot \epsilon$;
3: $t' = \text{bitlen}(d')$;
4: $\mathbf{Q} \leftarrow \mathbf{P}$;
5: for $i$ from $t' - 1$ to 0 do
6:     $\mathbf{Q} \leftarrow 2\,\mathbf{Q}$;
7:     If $d'_i = 1$ then $\mathbf{Q} \leftarrow \mathbf{Q} + \mathbf{P}$;
8: end for
9: Return $\mathbf{Q}$;
```

In Algorithm 4 we combine this countermeasure with the Double-And-Add algorithm of Algorithm 1. Exposure of the randomized scalar d' can be seen equivalent to the exposure of d.

In practice, we used a 32-bit random number k that results in a 288-bit randomized scalar for 256-bit ECC curves. Clearly, the choice of the bitlength of k is a practical tradeoff between security and performance. Naturally, 32-bit size is a favorable size for a 32-bit microcontroller. From the security point of view given that the adversary has only one side channel measurement with a secret scalar a repetitive recording of 2^{32} side channel measurements of a scalar multiplication with all candidates for k on a test device for template matching is practically out of reach. However, other attacks exist on this countermeasures that make use of the birthday paradox, e.g., [14]. The birthday paradox also applies if the adversary is able to collect a high number of measurements with a secret and a hypothesized scalar. Considering the birthday paradox a 32-bit random number k can be too small.

4.1.4 Randomized Scalar Splitting

Another approach for randomization is Randomized Scalar Splitting [22]. The secret scalar d is randomly split in two values d_1 and d_2 for each scalar multiplication. It holds that $d\mathbf{P} = d_1\mathbf{P} + d_2\mathbf{P}$.

In Algorithm 5 we combine this countermeasure with the Double-And-Add algorithm of Algorithm 1. Exposure of the scalars d_1 and d_2 can be seen equivalent to the exposure of d. Because of two scalar multiplications the runtime is almost doubled.

4.1.5 Randomized Projective Coordinates

Randomized projective coordinates are another countermeasure proposed in [12]. Theoretically, this countermeasure is considered to be highly efficient against a broad range of passive and active attacks except for RPA/ZPA [13, 18]. A point on the elliptic curve with projective coordinates

Algorithm 5 Randomized Scalar Splitting

```
Require: elliptic curve point $\mathbf{P}$, a positive integer $d = (1, d_{t-1}, ..., d_1, d_0)_2$
Ensure: $\mathbf{Q} = d\mathbf{P}$
 1: $d_1 \in_R \{0,1\}^t$;
 2: $d_2 = d - d_1$;
 3: $t_1 = \text{bitlen}(d_1)$;
 4: $t_2 = \text{bitlen}(d_2)$;
 5: $\mathbf{Q} \leftarrow \mathbf{P}$;
 6: for $i$ from $t_1 - 1$ to 0 do
 7:     $\mathbf{Q} \leftarrow 2\,\mathbf{Q}$;
 8:     If $d_{1i} = 1$ then $\mathbf{Q} \leftarrow \mathbf{Q} + \mathbf{P}$;
 9: end for
10: $\mathbf{R} \leftarrow \mathbf{P}$;
11: for $i$ from $t_2 - 1$ to 0 do
12:     $\mathbf{R} \leftarrow 2\,\mathbf{R}$;
13:     If $d_{2i} = 1$ then $\mathbf{R} \leftarrow \mathbf{R} + \mathbf{P}$;
14: end for
15: Return $\mathbf{Q} + \mathbf{R}$;
```

(X, Y, Z) is randomly mapped to another point with projective coordinates $(\lambda X, \lambda Y, \lambda Z)$ with a random scalar $\lambda \neq 0$.

Algorithm 6 Randomized Projective Coordinates

```
Require: elliptic curve point $\mathbf{P}$ in projective coordinates $(X, Y, Z)$ and a positive integer $d = (1, d_{t-1}, ..., d_1, d_0)_2$
Ensure: $\mathbf{Q} = d\mathbf{P}$
1: $\mathbf{Q} \leftarrow \mathbf{P}$;
2: for $i$ from $t - 1$ to 0 do
3:     $\lambda \in_R \{0,1\}^n \setminus \{0\}$;
4:     $(X, Y, Z) \leftarrow (\lambda X, \lambda Y, \lambda Z)$;
5:     $\mathbf{Q} \leftarrow 2\,\mathbf{Q}$;
6:     If $d_i = 1$ then $\mathbf{Q} \leftarrow \mathbf{Q} + \mathbf{P}$;
7: end for
8: Return $\mathbf{Q}$;
```

Algorithm 6 is combined with the Double-And-Add algorithm of Algorithm 1. Note that we applied the randomization in each iteration of the Double-And-Add algorithm. Other choices exist. The randomization can be done once before each scalar multiplication and, alternatively, the randomization can also occur after each point addition and doubling [12]. Applying this randomization inside the Double-And-Add algorithm provides an additional resistance against horizontal attacks. λ was a 32-bit random number. Again, the choice of the bitlength of λ is a practical tradeoff between security and performance, cf. SubSect. 4.1.3.

4.1.6 Randomized Sliding Window

Sliding window methods process a small fix amount of w bits of the secret scalar per iteration. They reduce the number of additions and require the pre-computation of a few points depending on the window size w. From SPA resistance point of view, an adversary additionally has to identify the used precomputated addend. A drawback of this method is that the number of sliding window algorithms for practical use of constrained platforms is limited.

A method offering enhanced resistance is a randomised sliding window method [26].

In Algorithm 7 we adapted the sliding-window exponen-

Algorithm 7 Randomized Sliding Window

```
Require: elliptic curve point $\mathbf{P}$, precomputed Points $\mathbf{P_0}...\mathbf{P_{2^w-1}}$ a positive integer $d = (1, d_{t-1}, ..., d_1, d_0)_2$.
Ensure: $\mathbf{Q} = d\mathbf{P}$
 1: $\mathbf{Q} \leftarrow \mathbf{P}$;
 2: $i \leftarrow t$;
 3: while $i \geq 0$ do
 4:     $w \in_R \{2, 4\}$;
 5:     $l \leftarrow \sum_{j=0}^{w-1} 2^j * d_{i-w+j+1}$;
 6:     for $j$ from 1 to $w$ do
 7:         $\mathbf{Q} \leftarrow \mathbf{Q} + \mathbf{Q}$;
 8:     end for
 9:     $\mathbf{Q} \leftarrow 2\mathbf{P_l}$;
10:     $i \leftarrow i - w$;
11: end while
12: Return $\mathbf{Q}$;
```

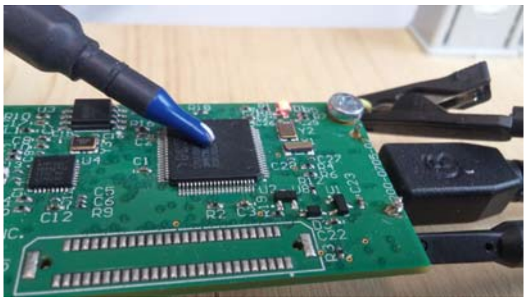


Figure 1: Measurement setup.

tiation algorithm 14.85 of [4]. Our implementation includes a randomized choice of the window size of either $w = 2$ or $w = 4$ for each iteration of the scalar scan and a randomized permutation of the precomputated points in each scalar multiplication. An attack on this countermeasure is described by Walter [34].

5. EXPERIMENTAL RESULTS

5.1 Measurement Set-Up

Our measurement setup consists of a target device and a digital oscilloscope. As the target device we used a high-performance sensor node called LOTUS from MEMSIC Inc. This node provides an ARM Cortex$^{®}$ M3 32-bit processor, a USB and a 802.15.4 Radio interfaces. The microcontroller is clocked with 16 MHz [19]. The USB interface is used for data transfer and power supply. We measured the electromagnetic emanation with the near-field probe LF-U-2,6 from Langer EMV on the microcontroller surface. We scan the surface manually to find a good position. After we have found a good position as shown in Figure 1 we used this for all our experiments. In a good position the processor instructions are clearly visible. As a digital oscilloscope we used the LeCroy WaveRunner 640Zi. The sampling rate is 100 mega samples per second and the signal was amplified by the preamplifier PA 303 from Langer EMV.

5.2 Performance

The execution time of the scalar multiplication depends on the processed data. Using one randomly chosen fixed scalar and Algorithm 1 the resulting execution times are shown in Table 1. The microprocessor was clocked with 16 MHz and the execution time was measured with the oscilloscope.

The execution time in projective coordinates is significantly smaller than in affine coordinates, because the algorithm in affine coordinates performs in each point addition and doubling operation a modular division. The modular division has the largest runtime. The curve P-256 was implemented by the fast reduction and has the smallest measured runtime with 0.51 s, the runtime in affine coordinates is larger by a factor of about 4. In projective coordinates, Ed25519 yielded a runtime of 1.10 s and brainpool256r1 turned out to be the slowest with 1.33 s. In contrast, in affine coordinates, the implementation of brainpoolP256r1 took 2.22 s while Ed25519 had the slowest runtime with 4.51 s. The reason for this is that the modular division was calculated twice in each point operation.

Table 2 also includes the percentage increase in execution time for the implemented countermeasures using projective coordinates. Again, these data are based on the same randomly chosen fixed scalar as in Table 1 and show a randomly chosen special case. The percentage increase of the three curves is roughly identical. The comparison measurement was done with Algorithm 1, the countermeasure Randomized Sliding Window results in an even faster execution time. The mean increase of Randomized Scalar (12.3 %) is slightly smaller than using Coordinate Randomization (15.3 %). In both cases the randomization was done with a 32-bit random number. A randomized scalar is approximately 288 bit long and causes 32 additional iterations in Algorithm 1 which corresponds to a percental increase of approx. 11% which is in good correspondence with the experimental results. The countermeasure Randomized Scalar Splitting uses a 256-bit random number and as result the execution time is approximately doubled. Regarding Randomized Projective Coordinates the percental increase is presumably mainly due to the fact that the randomization is done in each iteration of the double-and-add algorithm. The performance should be significantly better if the the randomization is done only once before the scalar multiplication. From the performance point of view the use of Randomized Sliding Window is of special interest. It can further be observed that the mean performance of Double-And-Add Always (37.7 %) turned out to be better than for the Montgomery Ladder (44.7 %) using our library SECLAB_ECC.

5.3 Side Channel Analysis: SEMA

Our primary objective is that simple side channel analysis is not feasible using one single measurement. For this, we tested the implementations on SEMA and MESD.

In the first step, we tested the unprotected implementations in affine and projective coordinates. This is useful to get first results on the susceptibility to distinguish the double and add operations. Our results are shown in Table 3.

Both the implementations of P-256 and brainpoolP256r1 failed the SEMA test using affine coordinates as differences at the transitions between operations are directly visible. In consequence, the entire sequence of doubling and addition can be read-out.

In Figure 2a we illustrate the differences for brainpoolP256r1: The blocks numbered 2,4 and 6 show modular divisions which take approximately 90% of the runtime that are used in both double and add operations. The structures num-

Table 1: Execution time using a fixed random scalar in affine and projective coordintes.

Implementation	**P-256**	**brainpoolP256r1**	**Ed25519**
Affine Coordinates	2.09 s	2.22 s	4.51 s
Projective Coordinates	0.51 s	1.33 s	1.10 s

Table 2: Runtime using a fixed random scalar in projective coordinates and the percentage increase in execution time for the listed countermeasures.

Implementation	**P-256**	**brainpoolP256r1**	**Ed25519**
Projective Coordinates	0.51 s	1.33 s	1.10 s
Double-And-Add Always	139 %	143 %	131 %
Montgomery Ladder	145 %	154 %	135 %
Scalar Randomization	110 %	117 %	110 %
Randomized Scalar Splitting	186 %	207 %	196 %
Randomized Projective Coordinate	112 %	118 %	116 %
Randomized Sliding Window	86 %	88 %	88 %

Table 3: Side channel results of the scalar multiplication without countermeasures.

	Result			
Implementation	**P-256**	**brainpoolP256r1**	**Ed25519**	**Method**
Affine coordinates	-	-	+	SEMA
Projective coordinates	+	+	+	SEMA
Projektive coordinates	-	-	-	MESD

$-$: vulnerability proven
$+$: vulnerability not proven

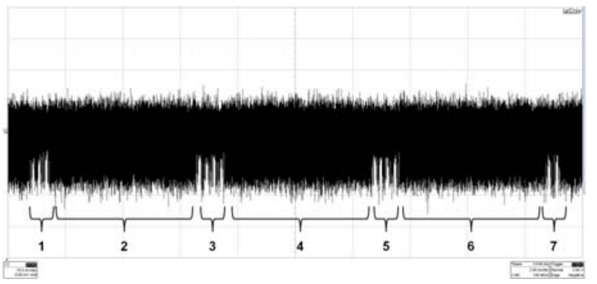
Table 4: Side channel results of the scalar multiplication of the countermeasures.

	Result			
Implementation	**P-256**	**brainpoolP256r1**	**Ed25519**	**Method**
Double-And-Add Always	-	-	-	MESD
Montgomery Ladder	-	-	-	MESD
Scalar Scalar Randomization	+	+	+	MESD
Randomized Scalar Splitting	+	+	+	MESD
Randomized Projective Coordinate	+	+	+	MESD
Randomized Sliding Window	+	+	+	MESD

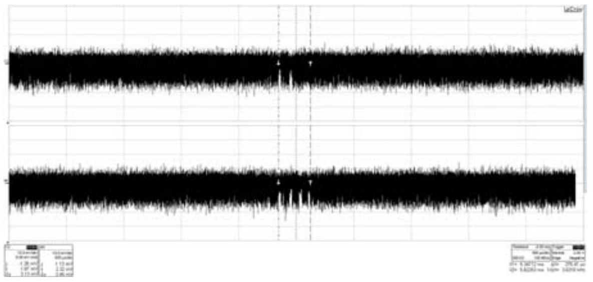
$-$: vulnerability proven
$+$: vulnerability not proven

bered 3 and 5 mark an add operation, the structures marked 1 and 7 mark a double operation. Similarly, differences in the transitions between operations can be seen for the curve P-256. In Figure 2b the upper curve shows the operations double followed by add and the lower curve shows two double operations. Such obvious differences were not detected for Ed25519 which can be explained with the use of an unified addition formula in contrast to P-256 and brainpoolP256r1 which used the formulas of (5). However, data dependent runtime of arithmetic operations can also be seen for Ed25519 in Figure 2c when comparing the pattern of the curves at the vertical dashed lines.

Using projective coordinates, transitions between operations are not directly visible which holds for all three curves. We cannot entirely exclude that a more demanding analysis might reveal other markers which can be used for distinguishing the operations, e.g. by timing analysis of operations. However, in summary, SEMA is much more difficult in projective coordinates.



(a) Elliptic curve *brainpoolP256r1*.


(b) Elliptic curve *P-256*.

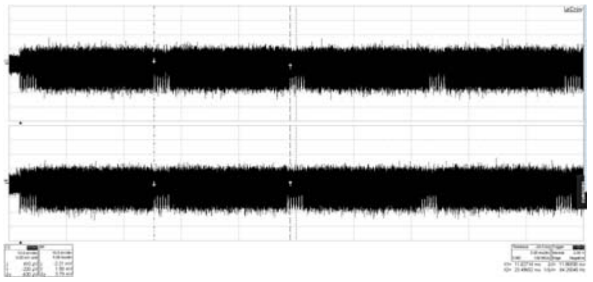

(c) Elliptic curve *Ed25519*.

Figure 2: Detailed view to transitions between arithmetic operations.

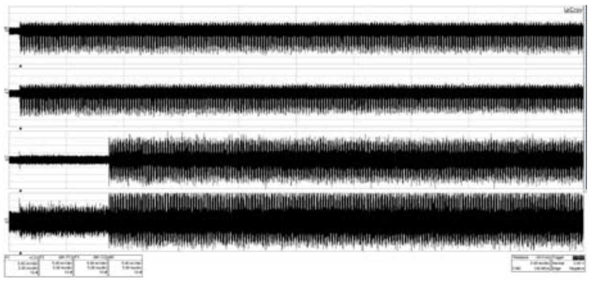
5.4 Side Channel Analysis: MESD

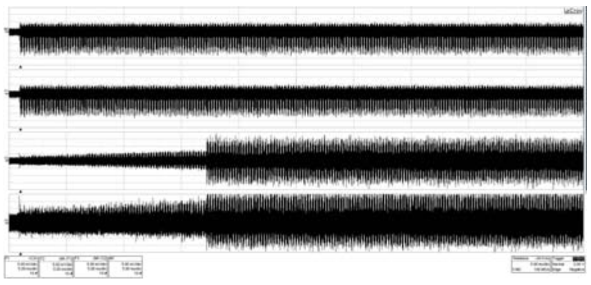
The analysis of MESD was applied to the three curves in projective coordinates. The implementations of the three curves clearly failed the MESD test. This clear observation can be explained with the data dependent runtimes of arithmetic operations.

In each figure, the top measurement shows the mean value of 10 single measurements with the secret scalar, the second measurement shows the mean of 10 single measurements with the hypothesized scalar, the third measurement shows the difference between first and second measurement, and the fourth measurement shows the difference between the first measurement and a single measurement with the hypothesized scalar. It can be seen that the hypothesized scalar can be successively corrected to correspond with the secret scalar.

Similar results for MESD were revealed for the countermeasures Double-And-Add Always and Montgomery Ladder for all three curves.

Countermeasures using data randomization withstand MESD as expected because the precondition of processing identical data in two implementations does not hold anymore with high probability for a 32-bit random number. Exemplarly, we show the application of MESD to the countermeasure Randomized Coordinates in Figure 4.


(a) First hypothetical key.


(b) Second hypothetical key.

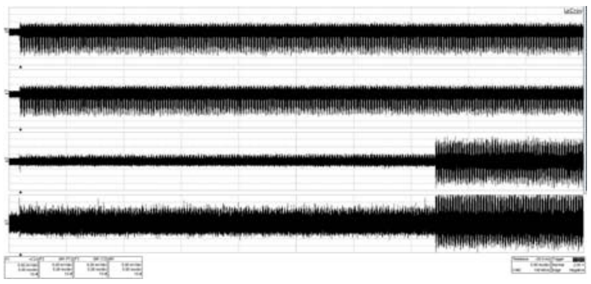

(c) Third hypothetical key.

Figure 3: Viability of MESD.

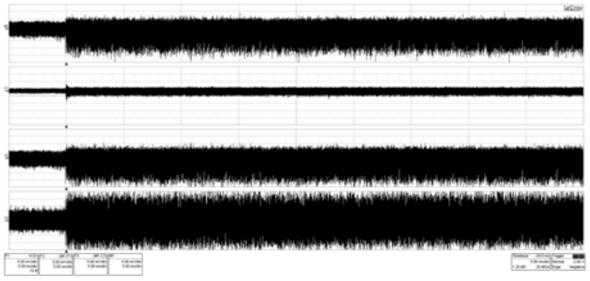


Figure 4: MESD applied to P-256 with Coordinate Randomisation.

6. CONCLUSION

As we did not conduct practical tests for all relevant side channel attacks, e.g. we did not cover horizontal attacks (RPA, ZPA), we cannot claim resistance to all known attacks. However, we are confident that that the most critical attacks can be counteracted. In our implementation, the add and double operations were found to be not distinguishable in projective coordinates. This allows to even skip protections such as Double-And-Add Always or Montgomery Ladder in these circumstances. Using randomization countermeasures we show that also more advanced attacks such as MESD can be prevented.

We note that an efficient combination of multiple countermeasures that resists all known passive and active attacks is still an open research question. We hope that our work can serve as a step forward towards ECC for lightweight applications with configurable security parameters for side channel countermeasures.

2. REFERENCES

[1] RFC 5639: Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation. https://tools.ietf.org/html/rfc5639.

[2] SECLAB_ECC. https://github.com/Thileen/SECLAB_ECC. revision:6d1cdaae8c80d65383eb4267c2603916bfdaf09e.

[3] T. Akishita and T. Takagi. Zero-value point attacks on elliptic curve cryptosystem. In C. Boyd and W. Mao, editors, *Information Security, ISC 2003*, volume 2851 of *LNCS*, pages 218–233. Springer, 2003.

[4] P. C. v. O. Alfred J. Menezes and S. A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 2001.

[5] L. Batina, L. Chmielewski, L. Papachristodoulou, P. Schwabe, and M. Tunstall. Online template attacks. In W. Meier and D. Mukhopadhyay, editors, *Progress in Cryptology - INDOCRYPT 2014*, volume 8885 of *LNCS*, pages 21–36. Springer, 2014.

[6] D. J. Bernstein. Curve25519: New Diffie-Hellman Speed Records. In M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, editors, *PKC 2006*, volume 3958 of *LNCS*, pages 207–228. Springer, 2006.

[7] D. J. Bernstein, P. Birkner, M. Joye, T. Lange, and C. Peters. Twisted Edward Curves. In *Cryptology ePrint Archive*, 2008.

[8] D. J. Bernstein and T. Lange. Explicit-formulas database. http://hyperelliptic.org/EFD.

[9] D. J. Bernstein and T. Lange. Performance evaluation of a new coordinate system for elliptic curves. http://cr.yp.to/newelliptic/newelliptic-20070522.pdf, 2007.

[10] E. Brier and M. Joye. Weierstraß Elliptic Curves and Side-Channel Attacks. In D. Naccache and P. Paillier, editors, *Public Key Cryptography, PKC 2002*, volume 2274 of *LNCS*, pages 335–345. Springer, 2002.

[11] S. Chari, J. R. Rao, and P. Rohatgi. Template attacks. In Jr. et al. [21], pages 13–28.

[12] J. Coron. Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems. In Koç and Paar [23], pages 292–302.

[13] J. Fan, X. Guo, E. D. Mulder, P. Schaumont, B. Preneel, and I. Verbauwhede. State-of-the-art of Secure ECC Implementations: A Survey on Known Side-channel Attacks and Countermeasures. In *HOST 2010, Proceedings of the 2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 76–87, 2010.

[14] P. Fouque and F. Valette. The doubling attack - *Why Upwards Is Better than Downwards.* In C. D. Walter, Ç. K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003*, volume 2779 of *LNCS*, pages 269–280. Springer, 2003.

[15] L. Goubin. A Refined Power-Analysis Attack on Elliptic Curve Cryptosystems. In Y. Desmedt, editor, *Public Key Cryptography - PKC 2003*, volume 2567 of *LNCS*, pages 199–210. Springer, 2003.

[16] J. Ha and S. Moon. Randomized signed-scalar multiplication of ECC to resist power attacks. In Jr. et al. [21], pages 551–563.

[17] D. Hankerson, S. Vanstone, and A. J. Menezes. *Guide to Elliptic Curve Cryptography.* Springer, 2004.

[18] H. Houssain, M. Badra, and T. F. Al-Somani. Power Analysis Attacks on ECC: A Major Security Threat. In *International Journal of Advanced Computer Science and Applications, Volume 3, Issue 6*, 2012.

[19] M. Inc. LOTUS, High-Performance Wireless Sensor Network Platform, Datasheet. http://www.memsic.com/wireless-sensor-networks/.

[20] M. Joye and M. Tunstall. *Fault Analysis in Cryptography.* Springer, 2012.

[21] B. S. K. Jr., Ç. K. Koç, and C. Paar, editors. *Cryptographic Hardware and Embedded Systems - CHES 2002*, volume 2523 of *LNCS*. Springer, 2003.

[22] W. Killmann, T. Lange, M. Lochter, W. Thumser, and G. Wicke. Minimum Requirements for Evaluating Side-Channel Attack Resistance of Elliptic Curve Implementations, Version 1.0.4. *Bundesamt für Sicherheit in der Informationstechnik, Bonn, Germany*, 2011.

[23] Ç. K. Koç and C. Paar, editors. *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99*, volume 1717 of *LNCS*. Springer, 1999.

[24] P. C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In N. Koblitz, editor, *Advances in Cryptology - CRYPTO '96*, volume 1109 of *LNCS*, pages 104–113. Springer, 1996.

[25] P. C. Kocher, J. Jaffe, and B. Jun. Differential Power

Analysis. In M. J. Wiener, editor, *Advances in Cryptology - CRYPTO '99*, volume 1666 of *LNCS*, pages 388–397. Springer, 1999.

[26] P.-Y. Liardet and N. P. Smart. Preventing SPA/DPA in ECC Systems Using the Jacobi Form. In Ç. K. Koç, D. Naccache, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2001*, volume 2162 of *LNCS*, pages 391–401. Springer, 2001.

[27] S. Mangard, E. Oswald, and T. Popp. *Power Analysis Attacks, Revealing the Secrets of Smart Cards.* Vieweg Verlag, 2007.

[28] T. S. Messerges, E. A. Dabbish, and R. H. Sloan. Power Analysis Attacks of Modular Exponentiation in Smartcards. In Koç and Paar [23], pages 144–157.

[29] A. Mirbach. *Elliptische Kurven, Die Bestimmung ihrer Punktezahl und Anwendungen in der Krypotographie.* Verlagshaus Monsenstein und Vannerdat, 2003.

[30] R. Moloney, G. McGuire, and M. Markowitz. Elliptic Curves in Montgomery Form with B=1 and Their Low Order Torsion. In *Cryptology ePrint Archive*, 2009.

[31] NIST. Recommended Elliptic Curves for Federal Government Use. http://csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.pdf, July 1999.

[32] J. Samotyja, K. Lemke-Rust, and M. Ullmann. SEMA and MESD Leakage of TinyECC 2.0 on a LOTUS Sensor Node. In *Cryptology ePrint Archive*, 2015.

[33] S. C. Shantz. From Euclid's GCD to Montgomery Multiplication to the Great Divide. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.140.7944\&rep=rep1\&type=pdf, June 2001.

[34] C. D. Walter. Breaking the Liardet-Smart Randomized Exponentiation Algorithm. In P. Honeyman, editor, *Proceedings of the Fifth Smart Card Research and Advanced Application Conference, CARDIS '02*, pages 59–68. USENIX, 2002.