Horizontal Address-Bit DPA against Montgomery *kP* Implementation

Ievgen Kabin, Zoya Dyka, Dan Kreiser and Peter Langendoerfer
IHP
Im Technologiepark 25
Frankfurt (Oder), Germany

***Abstract*—With the advent of the Internet of Things security features become more and more important. Especially data integrity and authentication of its origin are of utmost importance. Digital signatures are a well-known means to provide these features. In this paper we detail our horizontal Address-Bit DPA attack against a Montgomery *kP* implementation and applied the attack against the ECDSA -algorithm. Please note that this type of attack is especially harmful as it requires the measurement of only one trace to preparing the attack and only one trace for the attack itself. Even more important well-known countermeasures such as the regularity and atomicity principle successful applied against simple SCA attacks and randomization strategies successfully applied against vertical DPA attacks do not prevent our attack from being successful. The attack allowed us to reveal the private key used to generate a digital signature with a correctness of 100%. Our detailed analysis of the attacked *kP* design showed that the addressing of registers is dependent on the currently processed key bit value. The average difference of compressed values in the clock cycles with addressing is 25.9mV^2, whereas the difference of the data storing to the registers is as low as 7.1mV^2. This difference was detected using the difference of means test analysing a single *kP* execution. Thus, the original (vertical) Address-Bit DPA can be significantly simplified. Especially hardware implementations of the *kP* algorithm need a new countermeasure due to the fact that the traditional randomization countermeasures, even applied for the randomizing intermediate data, don't prevent the horizontal Address-Bit DPA.**

Keywords— ECC; Montgomery kP algorithm; FPGA implementation; side channel analysis (SCA) attacks; power traces; horizontal DPA attacks; Address-Bit DPA; ECDSA.

I. INTRODUCTION

Confidentiality of communication, authentication of devices or even persons, non-repudiation and other privacy and security features can be implemented using cryptographic approaches, i.e. using algorithms for the data encryption and digital signatures. The cryptographic keys – nowadays long binary numbers – have to be kept secret. In case of symmetric cryptographic approaches[1] at least two persons apply the same cryptographic key for the encryption as well for the decryption. This key is called *secret* key. In asymmetric cryptographic approaches[2] each person (or device) has pair of keys: the *private* key that has to be kept secret and the *public* key that can be known by everybody. The goal of an attacker is to reveal the *secret* key or the *private* key. If the *secret* key of a communication session is revealed the confidentiality of the communication is lost. If the *private* key of a person is revealed, the identity of the person can be stolen.

Side Channel Analysis (SCA) attacks exploit the fact that physical effects such as time, power consumption and electromagnetic radiation of cryptographic devices can be measured while performing cryptographic operations. The shape of measured power traces (PT) or electromagnetic traces (EMT) depends not only on the implemented circuit (technology, area of the design, etc.) but also on the processed input data and the applied cryptographic key. Thus, the measured traces can be analysed with the goal to reveal the *private* key. The power analysis and electromagnetic analysis attacks are the most researched ones.

The classification of power analysis attacks into simple power analysis (SPA) and differential power analysis (DPA) was introduced in 1999 by P. Kocher [1]. Corresponding to the [1] SPA and DPA were for many years defined as follows:

- simple (power) analysis attacks: an attacker tries to reveal the key via simple optical inspection of one (measured) trace of a cryptographic operation;
- differential (power) analysis attacks: an attacker measures a set of traces of the same cryptographic operation executed with different inputs and applies a statistical analysis for revealing the key.

Generally, the PA attacks should be not successful if the measurements cannot be performed, i.e. if a cryptographic device is tamper resistant, or if the measured power traces are always identical or fully random, independently of the processed input data and applied cryptographic keys. This hold true also if in a power trace the power profile of the processing of the key bit value '1' is indistinguishable from the one of the processing of the key bit value '0'.

[1] "single-key cryptographic algorithm" [4]

[2] "A two-key, public-key cryptosystem, such as an RSA or elliptic curve cryptosystem…" [4]

978-1-5386-3797-5/17/$31.00 ©2017 European Union

A. Background and related work

In the following paragraphs we shortly explain approaches to prevent side channel attacks investigated in the past, namely balancing and randomization strategies.

Balancing strategies. Different countermeasures such as gates whose switching energy doesn't depend on their inputs, for example energy-balanced gates [2], dual-rail logic [3], etc. try to implement a kind of constant energy consumption "on the gate level. On the algorithm level there are the "regularity" and "atomicity" principle. "Regularity" means: the same operation sequence has to be performed independently of the processed key bit (or byte). An example for RSA is the "*square&multiply-always*" approach [4] that is an SPA resistant variant of "*square&multiply*" algorithm. Corresponding to the "*square&multiply*", only one operation – a squaring – has to be executed if the processed key bit value is '0' and two operations – a squaring and a multiplication have to be executed – if the key bit value is '1'. In the "*square&multiply-always*" algorithm both operations have to be executed for the processing of each key bit, independently of its value. Each multiplication, executed for the processing of key bit '0' is a dummy (or fake) operation and is performed only for to make the processing of the key bits '0' and '1' indistinguishable. The Montgomery EC point multiplication algorithm [5] or the "*double&add-always*" algorithm [6] are analogue examples for ECC. Already [4] described the disadvantage of the using long and complicated dummy operations: "This solution is very costly for performance and still may be vulnerable if the act of saving the result can be observed in the power signal." Thus, the next idea was to use the dummy operations more effectively and tiny. For ECC an universal calculation formula was proposed for initially different operation sequences, for EC point doubling and EC point addition [7]. The next level of the "regularity" strategy for EC over *GF(p)* is the "SCA atomicity" principle that was introduced in [8]. This aims at performing all operations, including write-to-register operations, regular to make the power profile of the EC point doubling indistinguishable from the EC point addition. This is especially important for *GF(p)* implementations, because the dummy EC point addition takes a lot of time and energy. Applying dummy field operations is a more efficient strategy. Using this strategy and the atomicity principle authors of [9] obtained a universal operation sequence for the calculation of point doublings as well point as additions using 8 field multiplications and 10 field additions/subtractions. For other kind of ECs – ECs over $GF(2^n)$ – this aspect is less relevant if the Montgomery *kP* algorithm is implemented using Lopez-Dahab projective coordinates [10] for the EC point representation. Corresponding to the Montgomery *kP* algorithm a point addition and a point doubling have to be performed always for processing of each key bit. It requires in total only 6 field multiplications, 5 field squarings and 3 field additions. Power profiles of different field operations are really well distinguishable from each other but their sequence is always the same independent of the key bit value. The computational burden of squarings and additions in $GF(2^n)$ are negligible in comparison to a field multiplication. Thus, the computational burden for processing of each key bit is only 6 field multiplications for performing a point doubling and a point addition if EC over $GF(2^n)$ is used. For EC over *GF(p)* a sequence of 8 field multiplications have to be executed for the processing of a key bit value '0' and two indistinguishable sequences of 8 field multiplications have to be executed for the processing of key bit value '1'. Thus, the processing of a key using EC over *GF(p)* takes more time and energy in comparison to EC over $GF(2^n)$ even if the "*double&add*" and not the "*double&add-always*", algorithm is implemented.

Both balancing principles – the regularity and the atomicity – are well-known and sufficient countermeasures against SPA attacks.

Randomization strategies. Randomization strategies are known as increasing noise [1], [11], different masking technics [12], [13], blinding of inputs [6], randomization of the key [6], randomization of the steps of cryptographic algorithms [14], [15], [16], spatial and temporal jitter strategies [17], run-time clockwise randomization [18].

Most of the listed countermeasures are effective only if more than a single trace are attacked, i.e. against classical DPA [1], CPA [19] and collision attacks [4], [20]. In all these cases the attacker uses his knowledge about inputs to be processed in the algorithm. If instead of the known data random inputs are processed in each new execution, the attacker doesn't have any reliable knowledge about the intermediate data and cannot proof his assumptions using the measured traces. The well-known countermeasures [6] were proposed explicit to avoid the DPA using many traces. The most of randomization techniques cannot prevent "single-trace" analysis, i.e. the SPA attacks corresponding to the classification given above.

Not all single-trace attacks are "simple" analyses. The first single-trace attack using a statistical analysis was introduced in 1999 in [4]. There a cross correlation analysis of an RSA modular exponentiation trace was processed with the goal to distinguish a multiplication from a squaring. The attack was not successful. The next horizontal attack was the Big Mac attack [21]. In 2010 authors of [22] published a Horizontal Correlation Power Analysis attack against the RSA exponentiation. A similar attack is described in [23]. These attacks were successful. In these attacks correlation analysis was applied and they exploited the fact that the multiplication of long integers was implemented using the classical multiplication method (MM). The more partial products are calculated using the classical MM the higher the success chances of the attack. This is due to the fact that by segmenting multiplicands in n segments n^2 partial products have to be calculated, whereby n partial multiplications with each segment as a multiplicand have to be performed. In [24] the possibility to distinguish between two multiplications with completely different operands and two multiplications with at least one common operand was published. Also here it was assumed that the classical MM was implemented.

In [22] not only a Horizontal Correlation Power Analysis attack was introduced, but a new classification of attacks – into vertical and horizontal attacks – was proposed. Attacks using more than one trace for revealing the key were called vertical. Attacks using only one trace for revealing the key are called

horizontal. An SPA is for example a kind of horizontal attacks. The attacks published in [4], [21], [22], [23] and [24] can be called horizontal DPA attacks.

As mentioned above the randomization of projective coordinates of EC point, or point blinding, or key randomization as proposed in [6] are not effective against horizontal attacks. To prevent horizontal DPAs a randomization of intermediate values was proposed in [22]. Examples of such randomizations are randomization of projective coordinates of EC points after each point operation in the *kP* algorithm, or the randomization of the sequence of the EC point operations in the *kP* algorithm [14], or the randomization of the calculation steps of the field multiplication [15], [16], or the clockwise runtime circuit permutation of the field multiplier [18]. Thus, in comparison to vertical DPA attacks, the horizontal DPA attacks are relative simple but nevertheless they require by far more complex kinds of randomization so that the randomization is an effective countermeasure. To the best of our knowledge, the influence of these countermeasures was not investigated using a *kP* design up to now.

An interesting kind of attacks is introduced in [25] - in principle this attack is an SEMA attack but the author measured 8000 traces to prepare this attack so that it needs to be considered as a kind of DEMA.

Another very interesting kind of attack is the Address-Bit DPA that was introduced in 2002 in [26]. It is a vertical attack against a hardware implementation of a Montgomery *kP* algorithm for EC point multiplication. In [26] 500 traces for a *kP* execution with an given scalar k=1111… were measured. Additionally, 500 traces of a *kP* execution with an unknown scalar, i.e. with the *private* key, were measured. The EC points for all these measurements were randomly chosen. Only the scalars were constant. We described this vertical attack here more detailed due to the following facts:

- a phenomenon similar to the one described in [26] causes a high success rate of our horizontal attack, i.e. we performed a *horizontal Address-Bit DPA*;
- in [26] it was shown that key randomization can prevent their vertical Address-Bit DPA, but point blinding and randomization of projective coordinates are not effective;
- the key randomization proposed in [26] as a countermeasure is no longer effective against the *Address-Bit DPA* if the attack can be performed *horizontally*.

B. Contribution and structure of this paper

The contributions of this paper are:

- performing of a low-cost, low-complexity and highly-efficient horizontal DPA attack against our implementation of the Montgomery *kP* algorithm exploiting the same phenomenon as in the vertical Address-Bit DPA [26];
- applying of the horizontal Address-Bit DPA attack against an ECDSA algorithm implementation.

Performing our attack we revealed the *private* key used for a signature generation with a correctness of 100% using only two measurements – the trace of an ECDSA verification was measured to provide the essential information for assessing the correctness of the revealed scalars used in the point multiplications. For revealing the private key 100% successfully only one trace of the execution of the ECDSA signature was measured and analysed.

The rest of this paper is structured as follows. In section II we explain the implementation details of our ECC design. In section III we present in detail how we performed our horizontal DPA attack using the *difference of means* test. In section IV we discuss our horizontal DPA attack against the ECDSA algorithm that revealed the private key used for the digital signature with a correctness of 100%. The paper finishes with short conclusions.

II. Structure of Implemented Montgomery *kP* Designs

In this section we explain the implementation details of the designs investigated here. We implemented the Montgomery *kP* algorithm using Lopez-Dahab projective coordinates [27] based on Algorithm 2 in [28]. Our design is a *kP* accelerator for the NIST EC *B-233* [29]. The structure of our *kP* designs is shown in Fig. 1

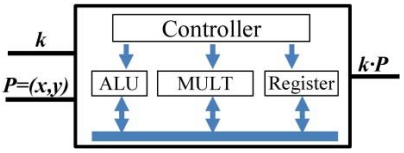

Fig. 1. Structure of our kP designs

The block Controller manages the sequence of the field operations. It controls the data flow between the other blocks and defines which operation has to be performed in the current clock cycle. Depending on the signals of the Controller the block ALU performs addition or squaring of its operands.

The implemented algorithm is regular, the processing of each key bit is a sequence of 6 field multiplications, 5 field squarings, 3 field additions (bitwise XOR) and write to register operations. Furthermore, we applied the *side-channel atomicity* principle: we ensured that the processing of each key bit value is implemented using the same operation sequence, including also the write to register operations in internal blocks of our design. In addition to both SPA countermeasures approaches – the *regularity* and the *atomicity* principle – we increased the inherent resistance of our design by performing additions, squarings and write to register operations always in parallel to the field multiplication. The multiplier takes only 9 clock cycles in our implementation to calculate a field product of 233 bit long operands, i.e. the multiplier is a source of strong noise if the activity of other blocks has to be analysed. This increases the resistance of our design against SCA attacks. The multiplier is implemented using the 4-segment Karatsuba multiplication method (MM), in which all 9 partial multiplications are calculated with different operands. Fig. 2 shows the processing sequence in the main loop of our implementation.

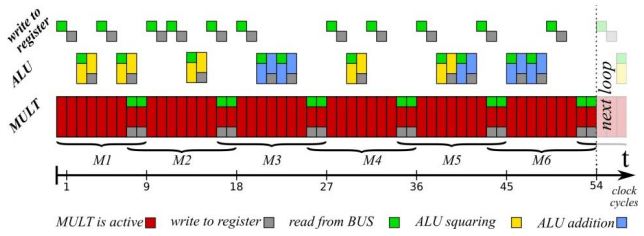

Fig. 2. Main loop implementation in our *kP* designs

The rectangles in Fig. 2 represent the activities of our ECC design. Rectangles that are vertically aligned are processed in parallel i.e. in the same clock cycle. Red rectangles show the activity of the field multiplier in each clock cycle the field multiplier calculates a partial product of 59 bit long partial operands, reduces it and accumulates it into internal output register. To calculate a new product, the field multiplier obtains two 233 bit long operands from the bus (see green squares) and stores the operands (see grey squares). Receiving new operands takes 1 clock cycle each and is performed in parallel to the main activity of the field multiplier. We symbolize this two clock cycles taking processes by a red rectangle that contains a small green and a small grey square.

The activity of the block ALU is shown in Fig. 2 using yellow rectangles for the squaring operation (including the field reduction) and blue rectangles for the field addition that is a bitwise XOR operation. Also here reading operands from the bus is symbolized by small green squares. The storing data into ALU's internal register is symbolized by small grey square.

All write to register operations are depicted in by small green squares for the addressing of the register in one clock cycle and a small grey square for storing operation in the next clock cycle.

In the following section we describe a horizontal DPA attack against our ECC design.

III. Attack Details: Horizontal Address-Bit DPA

A. Measurement setup

We ported our design implemented in VHDL to a Spartan-6 FPGA [30]. Our ECC design is running at 4 MHz using a 232 bit long random number as the scalar k and the base point G [29] of EC *B-233* as point P. We measured the current through the FPGA using the Riscure current probe [31]. The trace was captured using a LeCroy Waverunner 610Zi oscilloscope with a 2.5 GS/s sampling rate, i.e. with 625 measurement points per clock cycle. Fig. 3 shows our measurement setup and a part of a measured power trace.

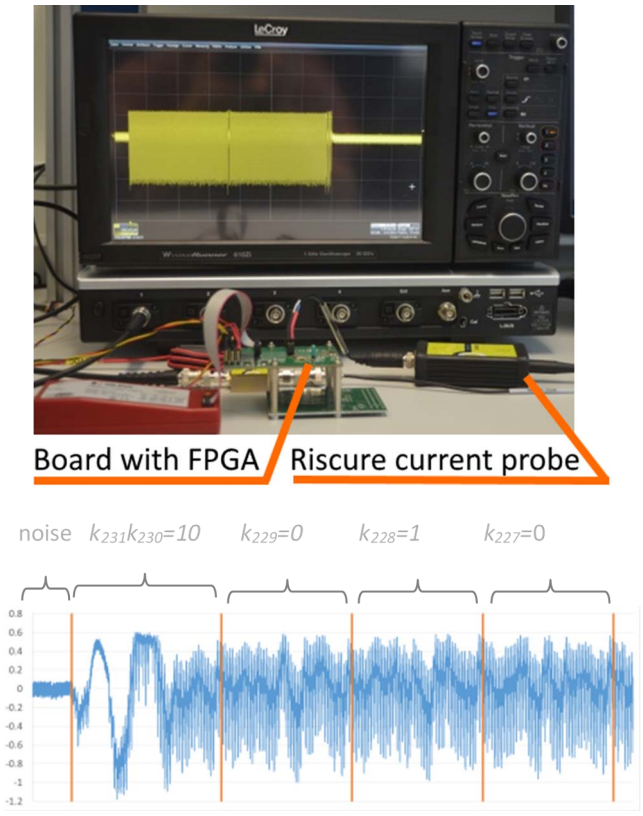

Fig. 3. Measurement setup and a part of the power trace measured for a *kP* execution

B. Compression of the trace

We represented each clock cycle using only one power value – i.e. the average power value of the clock cycle. This increases the success of our attack because it reduces the noise. In order to reduce the noise we used the following formula for the compression:

$$v^{compressed} = \frac{1}{625} \cdot \sum_{i=1}^{625} \left(v_l^{measured} \right)^2 \quad (1)$$

Here $v^{compressed}$ is the average value of a clock cycle; 625 is the amount of measured values $v_l^{measured}$ within the clock cycle. The reasons are:

$$v_l^{measured} = Signal_l \pm Noise_l \quad (2)$$

$$\left(v_l^{measured} \right)^2 = (Signal_l \pm Noise_l)^2 = \\ = Signal_l^2 \pm 2 \cdot Signal_l \cdot Noise_l + Noise_l^2 \quad (3)$$

The squaring leads to the fact that the impact of the noise is significantly reduced. This is due to the fact that the noise is relatively small compared to the signal. This means that the squaring increases the ratio of the signal more than the one of the noise. Furthermore, by averaging (1) the sum of terms $\pm 2 \cdot Signal_i \cdot Noise_i$ from (3) will be near to zero.

In addition the compression helps to reduce the complexity of the statistical analysis. After the compression we excluded

the processing of the two most significant key bits k_{231} and k_{230} of the 232 bit long scalar k from the set of time slots that we used for our attack because they are processed not in the main loop. The part of the measured trace that we prepared for analysis consists of 230 time slots. Each time slot corresponds to the processing of a key bit k_j in the main loop of the Montgomery kP algorithm with $0 \le j \le 229$. Each of these time slots consists of 54 power values (one power value per clock cycle). Thus, each power value of the analysed part of the compressed power trace can be represented as v^i_j, where i is the number of the clock cycle ($1 \le i \le 54$) within the time slot and j is the number of the time slot and the index of the processed key bit ($0 \le j \le 229$).

Fig. 4 shows the part of the compressed PT that corresponds to the processing of the key bits $k_{229}=0$, $k_{228}=1$ and $k_{227}=0$. The power profile of these slots (before compressing) is shown in Fig. 3.

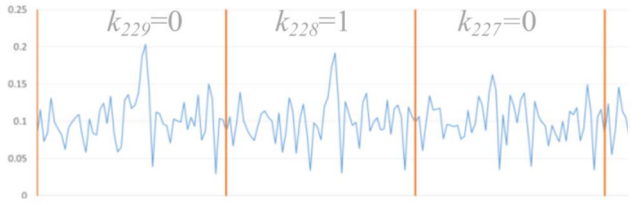

Fig. 4. Part of the compressed PT that corresponds to the processing of the key bits $k_{229}=0$, $k_{228}=1$ and $k_{227}=0$.

C. Statistical analysis using difference of means test

We decided to apply the *difference of means* test for the statistical analysis. Using the 230 time slots we calculated the arithmetical mean of all power values with the same number i and different number j:

$$\overline{v^i} = \frac{1}{230}\sum_{j=0}^{229} v^i_j \qquad (4)$$

Thus, the 54 calculated values $\overline{v^i}$ define the *mean* power profile of the slot.

For each i we obtained one key candidate $k^{candidate\_i}$ using the following assumption: the j^{th} bit of the key candidate is 1 if in the slot with number j the power value with number i – i.e. the power value v^i_j – is smaller than or equal to the average value $\overline{v^i}$. Else the j^{th} bit of the i^{th} key candidate is 0:

$$k^{candidate\_i}{}_j = \begin{cases} 1, & if \quad v^i_j \le \overline{v^i} \\ 0, & if \quad v^i_j > \overline{v^i} \end{cases} \qquad (5)$$

D. Evaluation of the success of the attack

To evaluate the success of the attack we calculated for each key candidate its relative correctness as follows:

$$\delta = \frac{\#RevealedBits(k^{candidate\_i})}{230} \cdot 100\% \qquad (6)$$

Here $\#RevealedBits(k^{candidate\_i})$ is the number of bits in $k^{candidate\_i}$ that have the same value as the processed key. The range of the correctness δ is between 0 and 100%.

To determine $\#RevealedBits(k^{candidate\_i})$ we compared each of the key candidates $k^{candidate\_i}$, for $3 \le i \le 52$, with the processed scalar k directly, i.e. we compared the bits of key candidates $k'_{229}k'_{228}\cdots k'_0$ with the bits of the processed scalar $k_{229}k_{228}\cdots k_0$.

In our implementation the processing of the key bits overlaps (see). This has some impact on the first two and the last two clock cycles per time slot, i.e. on the correctness of revealing 1st, 2nd, 53rd and 54th key candidates. Taking this fact into account, we compared these key candidates not only directly with the processed scalar. Additionally we compared the bits $k'_{229}k'_{228}\cdots k'_1$ of the 1st and 2nd key candidate with the bits $k_{230}k_{228}\cdots k_1$ of the processed scalar; and the bits $k'_{229}k'_{228}\cdots k'_1$ of the 53rd and 54th key candidate with the bits $k_{228}k_{228}\cdots k_0$ of the processed scalar. After comparison we selected "the highest" $\#RevealedBits(k^{candidate\_i})$ for calculating correctness δ.

If the correctness δ of key candidate is close to 0 percent, our assumption in equation (5) needs to be wrong so the opposite assumption will be correct. Thus, the relative correctness $\delta = 0\%$ of a key candidate obtained using assumption (5) will correspond to correctness $\delta = 100\%$ if the opposite assumption is used.

For the attacker the worst-case of the attack result is a correctness of 50 percent which means the *difference of means* test cannot even provide a slight hint whether the key bit processed is more likely a '1' or a '0'. The worst-case from the attacker's point of view is the ideal case from the designer's point of view. We denoted it as the "ideal case" in the rest of the paper.

Fig. 5 shows the correctness δ of the key candidates graphically as bars. The green line refers to the "ideal case". The blue bars show the result of the performed attack using the power trace for the kP execution using the scalar k and point G. The red bars correspond to an additional kP execution with a blinded point and a randomized key: $k' = k + \varepsilon$ and blinded EC point $P' = G + R$, where ε is the order of the base point G and R is a random point on EC *B-233*.

Fig. 5. Correctness of the revealed key candidates: blue bars represent execution of kP, red bars represent execution with a blinded point and randomized key.

The results shown as blue bars are similar to results shown as red bars, i.e. the results of the attack against a *kP* execution with a blinded EC point and a randomized key are very similar to the results of an attack against an unprotected *kP* execution. This demonstrates the well-known fact that the randomization countermeasure proposed in [6] are not effective against horizontal attacks. 17 of the obtained (or inverted) key candidates were extracted with a correctness higher than 80%. In case an attacker knows 80% of bits of a key candidate, he needs to execute only $2^{(1-0.8)*233} = 2^{47}$ *kP* operations to reveal the key using brute-force. In our attack one of key candidates – the 28th key candidate – was revealed with a correctness of 100%.

E. Address-Bit DPA phenomenon

Using the obtained correctness of the key candidates we analysed processes in the main loop of our implementation clock-wise. In most cases the processes that cause a high success rate of the attack are write-to or read-from register operations, whereas only the addresses of the registers depend on the processed key bit value. This phenomenon is similar to the one described in [26].

Our attack: Horizontal Address-Bit DPA. In contrast to [26], where 1000 traces were analysed, we observed the same effect using only a single trace, i.e. we performed a horizontal Address-Bit DPA against the Montgomery ladder. Similar to [26] the data dependent switching of gates was averaged and address dependent switching was accumulated. The power consumption of the two registers *X1* and *X2* observed in clock cycles 28 and 29 of our implementation confirm this fact. In clock cycle 28 of each slot register *X1* is addressed if the processed key bit is '0'. If the processed key bit is '1' register *X2* will be addressed. Addressing register *X1* needs always a little bit more energy than addressing register *X2*. The averaged power consumption for register *X1* is 2.383 mW and for register *X2* it is 2.251 mW. The data is stored in the next clock cycle which needs 1.957 and 1.999 mW in average for register *X1* and *X2* respectively. These results we obtained analysing Power Traces simulated for our design that we synthesized for a 250 nm gate library technology. Experiments using the compressed values of measured traces resulted in similar results: the average difference of values in the clock cycles with addressing is 25.9 mV^2, whereas the difference of the data storing to the registers is as low as 7.1 mV^2. The relative difference of the data storing operation is significantly less than the one of addressing the registers. Thus, this address dependent switching is a marker of the processed key bit value.

In [26] it was stated that point blinding and randomization of projective coordinate didn't prevent the vertical Address-Bit DPA. In [26] the key randomization presented in [6] was applied: even though it was an effective countermeasure against the vertical Address-Bit DPA, it could not prevent our horizontal attack (see Fig. 5).

IV. ATTACK AGAINST ECDSA

In this section we describe how the horizontal Address-Bit DPA can be applied against the ECDSA algorithm [32] with the goal to reveal the private key used for digital signature generation. Fig. 6 shows the attack scenario schematically.

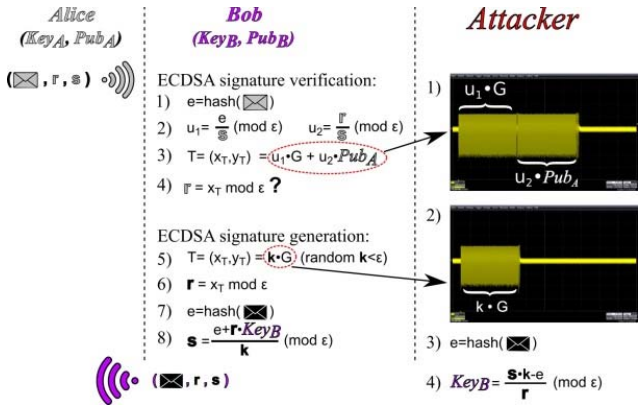

Fig. 6. Horizontal DPA attack against ECDSA.

In our attack the attacker can listen to the communication between Alice and Bob. Bob has a key pair: the public key Pub_B and the private key Key_B. Alice's key pair is Pub_A and Key_A. Corresponding to the ECDSA algorithm, public keys are points of an EC (in our case the EC *B-233*) and private keys are big binary numbers, smaller than the order ε of the base point *G*. Alice sends a message to Bob with a signature (*r, s*), with $r,s< \varepsilon$. Bob and the attacker receive this message and numbers *r, s*. Thus, both can calculate the number *e* (it is the hash of the received message) the scalar u_1 and the scalar u_2 (see steps 1 and 2 of the ECDSA signature verification in Fig. 6). In our experiments scalars u_1 and u_2 are 232 bit long.

The next step of the signature verification is the calculation of the EC point T (see step 3 in Fig. 6). In our scenario the attacker can measure the power trace of this calculation. The power trace consists of two EC point multiplications: first the *kP* operation with the scalar u_1 and the base point *G*, and second the *kP* operation with the scalar u_2 and the EC point Pub_A. Thus, all operands for the calculation of EC point T are known to the attacker too.

After verification Bob sends his answer with a digital signature to Alice (or a message with his signature to another person). To generate a signature Bob performs steps 5)-8) shown in Fig. 6. First, Bob performs the *kP* operation with a random number *k* and the base point *G*. The power trace of this point multiplication is also measured by the attacker. Bob takes the *x*-coordinate of the calculated point *kG* as the number *r* of the signature, calculates hash of the message *e*, and – finally – calculates the number s of the signature using numbers *e*, *r*, *k* and his private key Key_B. Bob sends his message with the signature (i.e. with numbers r and s that were calculated by Bob) to Alice. The Attacker can listen to the communication. Thus, he knows all these numbers.

Then the attacker performs the horizontal Address-Bit DPA (see section III) using the measured power trace with known scalars u_1 and u_2. After the evaluation of the attack results as described in section III-D the Attacker obtains the correctness for revealing of scalar u_1 and u_2 that is shown in Fig. 7.

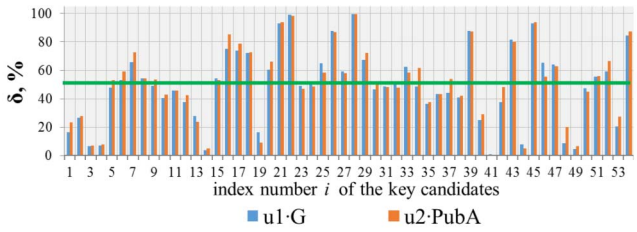

Fig. 7. Correctness of the revealed scalar candidates: blue bars represent execution of $u_1 \cdot G$, red bars represent execution of $u_2 \cdot Pub_A$.

The correctness δ of the revealed scalar candidates is shown graphically as bars in Fig. 7. The green line refers to the "worst case" from the point of view of the attacker. The blue bars show the result of the performed attack using the power trace for the $u_1 \cdot G$ execution. The red bars correspond to $u_2 \cdot Pub_A$ execution. Although scalars u_1 and u_2 are not equal to the scalars k and $(k+\varepsilon)$ in Section III-D, the results of the performed attack are very similar to the results shown in Fig. 5. The 28th key candidate is the best. Its correctness is 99.6%. The next best one is candidate 22 with a correctness of 98.3%.

The next step of the attacker is to perform the horizontal DPA using the power trace of the $k \cdot G$ execution (see step 2 of the Attacker in Fig. 6). Now the scalar k is unknown to the Attacker. Important is that the attacker can obtain all 54 scalar candidates and he can use the evaluation result of the attacks using the known scalars u_1 and u_2. Thus, the attacker knows that the candidates revealed with the highest correctness are the candidates with numbers 28 and 22. The Attacker can compare these two candidates to define the differences. The bit positions with the same values are taken as revealed by the attacker. In our experiments 227 of 230 bits of both candidates are identical. Thus, we performed only $2^{230-227}=2^3=8$ kP operations to reveal the scalar k with a correctness of 100% using brute-force: we know the point P (it is the base point G) and we know the x-coordinate of the $k \cdot G$ execution (it is the number r that Bob sent to Alice).

Using the revealed scalar the attacker can calculate the private key of Bob as follows:

$$Key_B = \frac{s \cdot k - e}{r} \bmod \varepsilon \qquad (6)$$

Finally, the attacker can examine the correctness of the revealed private key of Bob using the EC point multiplication performed for the generation of the Bob's key pair (Key_B, Pub_B): $Key_B \cdot G = Pub_B$.

V. Conclusions

In this paper we presented a horizontal Address-Bit DPA attack. The threatening facts are that the issue that made the attack work are write-to or read-from register operations, but the addressing of the registers, not storing/reading the data itself. The energy for the data storing operations is averaged, if many storing operations are observed and analysed statistically, within many kP execution as described in [26] or within a single kP execution as in our attack. The difference in energy consumption for addressing of two registers is small but not averaged. Thus, it is the marker of the key bit value currently processed. Using a sum of squared measured values as compressing of traces makes this difference more observable: the average difference of compressed values in the clock cycles with addressing is 25.9 mV^2, whereas the difference of the data storing to the registers is as low as 7.1 mV^2. The addressing of registers is not explicitly programmed by the designer so that it goes undetected and cannot be prevented by means such as the atomicity principle or even via randomization of intermediate data. Especially for hardware implementations of ECC operations it is a significant problem due to the fact that the design contains only few registers.

As the horizontal Address-Bit DPA attack needs only one trace for preparation and only one trace for the attack itself, it needs to be considered as a kind of low cost attack. We presented attack results against a kP design that show that the key can be revealed with a correctness of 100% even in case randomization techniques are applied.

Please note that we successfully attacked the execution of an ECDSA and were able to reveal the private key with a correctness of 100 percent. As ECDSA is a means to guarantee data integrity and authentication in the Internet of Things. This means an attacker can cause real harm after getting the private key of an IoT device.

In our next research steps we will investigate means to prevent the attack described here. A potential means are neatly designed dummy gates but this needs further research and experiments.

Acknowledgment

This research has been funded by the Federal Ministry of Education and Research of Germany under grant number 03ZZ052717.

References

[1] Kocher, P., Jaffe, J., Jun, B.: Differential Power Analysis. In: Advances in Cryptology — CRYPTO' 99. pp. 388–397. Springer Berlin Heidelberg, Berlin, Heidelberg (1999).

[2] Tiri, K., Akmal, M., Verbauwhede, I.: A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards. In: Proceedings of the 28th European Solid-State Circuits Conference. pp. 403–406 (2002).

[3] Moore, S., Anderson, R., Cunningham, P., Mullins, R., Taylor, G.: Improving smart card security using self-timed circuits. In: Proceedings Eighth International Symposium on Asynchronous Circuits and Systems. pp. 211–218 (2002).

[4] Messerges, T.S., Dabbish, E.A., Sloan, R.H.: Power Analysis Attacks of Modular Exponentiation in Smartcards. In: Cryptographic Hardware and Embedded Systems. pp. 144–157. Springer, Berlin, Heidelberg (1999).

[5] Montgomery, P.L.: Speeding the Pollard and elliptic curve methods of factorization. Math. Comp. 48, 243–264 (1987).

[6] Coron, J.-S.: Resistance Against Differential Power Analysis For Elliptic Curve Cryptosystems. In: Koç, Ç.K. and Paar, C. (eds.) Cryptographic Hardware and Embedded Systems. pp. 292–302. Springer Berlin Heidelberg, Berlin, Heidelberg (1999).

[7] Brier, É., Joye, M.: Weierstraß Elliptic Curves and Side-Channel Attacks. In: Public Key Cryptography. pp. 335–345. Springer, Berlin, Heidelberg (2002).

[8] Chevallier-Mames, B., Ciet, M., Joye, M.: Low-cost solutions for preventing simple side-channel analysis: side-channel atomicity. IEEE Transactions on Computers. 53, 760–768 (2004).

[9] Giraud, C., Verneuil, V.: Atomicity Improvement for Elliptic Curve Scalar Multiplication. In: Proceedings of the 9th IFIP WG 8.8/11.2 International Conference on Smart Card Research and Advanced Application. pp. 80–101. Springer-Verlag, Berlin, Heidelberg (2010).

[10] López, J., Dahab, R.: Fast Multiplication on Elliptic Curves Over GF(2m) without precomputation. In: Koç, Ç.K. and Paar, C. (eds.) Cryptographic Hardware and Embedded Systems. pp. 316–327. Springer Berlin Heidelberg, Berlin, Heidelberg (1999).

[11] Coron, J.-S., Kizhvatov, I.: An Efficient Method for Random Delay Generation in Embedded Software. In: Cryptographic Hardware and Embedded Systems - CHES 2009. pp. 156–170. Springer, Berlin, Heidelberg (2009).

[12] Goubin, L., Patarin, J.: DES and Differential Power Analysis The "Duplication" Method. In: Cryptographic Hardware and Embedded Systems. pp. 158–172. Springer, Berlin, Heidelberg (1999).

[13] Messerges, T.S.: Securing the AES Finalists Against Power Analysis Attacks. In: Fast Software Encryption. pp. 150–164. Springer, Berlin, Heidelberg (2000).

[14] Oswald, E., Aigner, M.: Randomized Addition-Subtraction Chains as a Countermeasure against Power Attacks. In: Cryptographic Hardware and Embedded Systems — CHES 2001. pp. 39–50. Springer, Berlin, Heidelberg (2001).

[15] Madlener, F., Sötttinger, M., Huss, S.A.: Novel hardening techniques against differential power analysis for multiplication in GF(2n). In: 2009 International Conference on Field-Programmable Technology. pp. 328–334. IEEE (2009).

[16] Stöttinger, M., Madlener, F., Huss, S.A.: Procedures for Securing ECC Implementations Against Differential Power Analysis Using Reconfigurable Architectures. In: Platzner, M., Teich, J., and Wehn, N. (eds.) Dynamically Reconfigurable Systems. pp. 395–415. Springer Netherlands (2010).

[17] Mentens, N., Gierlichs, B., Verbauwhede, I.: Power and Fault Analysis Resistance in Hardware through Dynamic Reconfiguration. In: Cryptographic Hardware and Embedded Systems – CHES 2008. pp. 346–362. Springer, Berlin, Heidelberg (2008).

[18] Dyka, Z., Wittke C., Langendoerfer P.: Clockwise Randomization of the Observable Behaviour of Crypto ASICs to Counter Side Channel Attacks. In Euromicro Conference on Digital System Design – DSD 2015. pp. 551–554.

[19] Brier, E., Clavier, C., Olivier, F.: Correlation Power Analysis with a Leakage Model. In: Cryptographic Hardware and Embedded Systems - CHES 2004. pp. 16–29. Springer, Berlin, Heidelberg (2004).

[20] Fouque, P.-A., Valette, F.: The Doubling Attack – Why Upwards Is Better than Downwards. In: Cryptographic Hardware and Embedded Systems - CHES 2003. pp. 269–280. Springer, Berlin, Heidelberg (2003).

[21] Walter, C.D.: Sliding Windows Succumbs to Big Mac Attack. In: Cryptographic Hardware and Embedded Systems — CHES 2001. pp. 286–299. Springer, Berlin, Heidelberg (2001).

[22] Clavier, C., Feix, B., Gagnerot, G., Roussellet, M., Verneuil, V.: Horizontal correlation analysis on exponentiation. In: In ICICS 2010, volume 6476 of LNCS. pp. 46–61. Springer-Verlag (2010).

[23] Bauer, A., Jaulmes, E., Prouff, E., Wild, J.: Horizontal and Vertical Side-Channel Attacks against Secure RSA Implementations. In: Topics in Cryptology – CT-RSA 2013. pp. 1–17. Springer, Berlin, Heidelberg (2013).

[24] Bauer, A., Jaulmes, E., Prouff, E., Wild, J.: Horizontal Collision Correlation Attack on Elliptic Curves. In: Selected Areas in Cryptography -- SAC 2013. pp. 553–570. Springer, Berlin, Heidelberg (2013).

[25] 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. pp. 231–244. Springer Berlin Heidelberg, Berlin, Heidelberg (2012).

[26] Itoh, K., Izu, T., Takenaka, M.: Address-Bit Differential Power Analysis of Cryptographic Schemes OK-ECDH and OK-ECDSA. In: Cryptographic Hardware and Embedded Systems - CHES 2002. pp. 129–143. Springer, Berlin, Heidelberg (2002).

[27] Hankerson, D., Hernandez, J.L., Menezes, A.: Software Implementation of Elliptic Curve Cryptography over Binary Fields. In: Cryptographic Hardware and Embedded Systems — CHES 2000. pp. 1–24. Springer, Berlin, Heidelberg (2000).

[28] Bock, E.A., Dyka, Z., Langendoerfer, P.: Increasing the Robustness of the Montgomery kP-Algorithm Against SCA by Modifying Its Initialization. In: Innovative Security Solutions for Information Technology and Communications. pp. 167–178. Springer, Cham (2016).

[29] Federal Information Processing Standard (FIPS) 186-4, Digital Signature Standard; Request for Comments on the NIST-Recommended Elliptic Curves: 2015. DOI = http://dx.doi.org/10.6028/NIST.FIPS.186-4

[30] Xilinx Inc.. Spartan-6 Family Overview, Product Specification. DS160 (v2.0) October 25, 2011, http://www.xilinx.com/support/documentation/data_sheets/ds160.pdf

[31] Riscure: Inspector data sheet. Current Probe. https://www.riscure.com/benzine/documents/CurrentProbe.pdf

[32] The Elliptic Curve Digital Signature Algorithm (ECDSA), http://cs.ucsb.edu/~koc/ccs130h/notes/ecdsa-cert.pdf