Journal of Electronic Testing (2021) 37:289–303
https://doi.org/10.1007/s10836-021-05951-3

Resistance of the Montgomery Ladder Against Simple SCA: Theory and Practice

Ievgen Kabin[1] · Zoya Dyka[1] · Dan Klann[1] · Marcin Aftowicz[1] · Peter Langendoerfer[1,2]

Received: 30 September 2020 / Accepted: 27 May 2021 / Published online: 30 June 2021

Abstract
The Montgomery *kP* algorithm i.e. the Montgomery ladder is reported in literature as resistant against simple SCA due to the fact that the processing of each key bit value of the scalar *k* is done using the same sequence of operations. We implemented the Montgomery *kP* algorithm using Lopez-Dahab projective coordinates for the NIST elliptic curve *B-233*. We instantiated the same VHDL code for a wide range of clock frequencies for the same target FPGA and using the same compiler options. We measured electromagnetic traces of the *kP* executions using the same input data, i.e. scalar *k* and elliptic curve point *P*, and measurement setup. Additionally, we synthesized the same VHDL code for two IHP CMOS technologies, for a broad spectrum of frequencies. We simulated the power consumption of each synthesized design during an execution of the *kP* operation, always using the same scalar *k* and elliptic curve point *P* as inputs. Our experiments clearly show that the success of simple electromagnetic analysis attacks against FPGA implementations as well as the one of simple power analysis attacks against synthesized ASIC designs depends on the target frequency for which the design was implemented and at which it is executed significantly. In our experiments the scalar *k* was successfully revealed via simple visual inspection of the electromagnetic traces of the FPGA for frequencies from 40 to 100 MHz when standard compile options were used as well as from 50 MHz up to 240 MHz when performance optimizing compile options were used. We obtained similar results attacking the power traces simulated for the ASIC. Despite the significant differences of the here investigated technologies the designs' resistance against the attacks performed is similar: only a few points in the traces represent strong leakage sources allowing to reveal the key at very low and very high frequencies. For the "middle" frequencies the number of points which allow to successfully reveal the key increases when increasing the frequency.

Keywords Side channel analysis (SCA) attacks · Simple SCA attacks · Electromagnetic analysis (EMA) · Power analysis (PA) · Elliptic curve cryptography (ECC) · Montgomery kP · Montgomery ladder · Lopez-Dahab projective coordinates · FPGA · ASIC

Responsible Editor: L. M. Bolzani Pöhls

✉ Ievgen Kabin
kabin@ihp-microelectronics.com

Zoya Dyka
dyka@ihp-microelectronics.com

Dan Klann
klann@ihp-microelectronics.com

Marcin Aftowicz
aftowicz@ihp-microelectronics.com

Peter Langendoerfer
langendoerfer@ihp-microelectronics.com

[1] IHP – Leibniz-Institut für innovative Mikroelektronik, Frankfurt (Oder), Germany

[2] BTU Cottbus-Senftenberg, Cottbus, Germany

1 Introduction

Nowadays elliptic curve cryptography (ECC) is applied for the exchange of shared secret keys, as well as for mutual authentication of communication partners and for signing or verifying of messages. The elliptic curve point multiplication denoted as *kP* operation is the main and the most time consuming operation for ECC. The scalar *k* is a long binary number and $P=(x, y)$ is a point on the selected elliptic curve (EC). Corresponding to the Elliptic Curve Digital Signature Algorithm (ECDSA) for the signature generation protocol the elliptic curve point multiplication *kG* has to be performed. The EC point *G* is the base point for the selected EC. Its coordinates are public parameters given in [1]. The scalar *k* is a random number. This random number has to be kept secret [2], since otherwise the private key of the user applied for the signature generation can be easily calculated [3]. In ECDH-based protocols for one-side authentication, as for example in [4], the scalar *k* is the private key in the *kP* operation performed and has to be kept secret. Due to these facts we denote the scalar *k* further also as the *key*. The goal of attackers is to reveal the key i.e. the scalar *k*. The algorithm for the *kP* calculation has to be fast and resistant against different attacks, including side channel analysis (SCA) attacks. SCA attacks assume that an attacker has physical access to the device running the cipher algorithm and can measure physical parameters influenced/affected by the working chip. These parameters can for example be the execution time of the analysed cryptographic operation, the energy consumption and its distribution during the execution of the operation clock-by-clock, temperature, electromagnetic emission etc. The physically measureable parameters are a kind of "side effects". Because all these parameters depend on the given input and the processed key, these "side effects" can be analysed with the goal to reveal the key *k*. The Montgomery *kP* algorithm using Lopez-Dahab projective coordinates [5] corresponding to [6] is a bitwise processing of the scalar *k,* see Algorithm 1.

Algorithm 1: Montgomery *kP* using projective Lopez-Dahab coordinates

```
Input: $k = (k_{l-1} \dots k_1 k_0)_2$ with $k_{l-1} = 1$, $P=(x,y)$ is a point of EC over $GF(2^l)$
Output: $kP = (x_1, y_1)$
 1: $X_1 \leftarrow x,\ Z_1 \leftarrow 1,\ X_2 \leftarrow x^4+b,\ Z_2 \leftarrow x^2$        } initialization phase
 2: for $i=l-2$ downto $0$ do                                                         ⎫
 3:     if $k_i=1$                                                                     ⎪
 4:         $T \leftarrow Z_1,\ Z_1 \leftarrow (X_1Z_2+X_2T)^2,\ X_1 \leftarrow xZ_1+X_1X_2TZ_2$  ⎪
 5:         $T \leftarrow X_2,\ X_2 \leftarrow T^4+bZ_2^4,\ Z_2 \leftarrow T^2Z_2^2$          ⎪
 6:     else                                                                           ⎬ main loop
 7:         $T \leftarrow Z_2,\ Z_2 \leftarrow (X_2Z_1+X_1T)^2,\ X_2 \leftarrow xZ_2+X_1X_2TZ_1$  ⎪
 8:         $T \leftarrow X_1,\ X_1 \leftarrow T^4+bZ_1^4,\ Z_1 \leftarrow T^2Z_1^2$          ⎪
 9:     end if                                                                         ⎪
10: end for                                                                            ⎭
11: $x_1 \leftarrow X_1/Z_1$                                                           } conversion of the result back to
12: $y_1 \leftarrow y + (x+x_1)\cdot((X_1+xZ_1)(X_2+xZ_2)+(x^2+y)(Z_1Z_2)) / (xZ_1Z_2)$   } affine coordinates;
13: return $(x_1, y_1)$                                                                 recovering of the y-coordinate
```

It is the most often implemented algorithm for the *kP* operation for ECs over extended binary Galois fields $GF(2^l)$. This algorithm is fast due to the fact that only 6 field multiplications are performed for the processing of a key bit in each iteration of the main loop. Corresponding to this algorithm each key bit is processed with the same type, amount and sequence of operations, independently of the key bit's value i.e. this algorithm is *regular*. This is the reason why the Montgomery *kP* algorithm is referred as resistant against simple SCA attacks in the literature, see for example [7, 8]. Well-known and obvious is the fact that the Montgomery *kP* algorithm using Lopez-Dahab projective coordinates contains many key dependent write-to-register operations. The assertion that the Montgomery *kP* algorithm is resistant against simple side channel analysis attacks is based on the assumption that an attacker cannot distinguish which of the registers is used by a visual inspection of the measured power or electromagnetic trace. The key dependent use of registers in the algorithm is known as its address-bit vulnerability. The first successful vertical attack (i.e. using many recorded traces) exploiting this vulnerability was published by Itoh et al. in 2002 [9] and is known as Address Bit Differential Power Analysis (DPA) attack. A horizontal (i.e. single-trace attack) Address Bit DPA was published in [10].

This paper is based on an earlier version published at [11]. In [11] the following points were addressed:

- porting our implementation of the Montgomery *kP* algorithm – always the same code – to an FPGA for a wide range of different target frequencies using two different compiling options;
- implementing an automated simple SCA attack and applying it against electromagnetic traces measured at our 12 FPGA implementations of the Montgomery *kP* algorithm;
- demonstrating that the resistance of the designs against simple SCA depends significantly on the target frequency for which the design was synthesized and at which it was run i.e. the scalar *k* was successfully revealed for designs synthesized for frequencies of 50 MHz or higher.

  In this papers we investigate additionally if the resistance of our design depends on the target frequency for an ASIC implementation i.e. if similar effects as those observed for FPGAs can as well be observed for ASIC implementations. So we:
- synthesized our *kP* design –for 16 target frequencies for the IHP 250 nm gate library and for 20 target frequencies for the IHP 130 nm gate library;
- applied our automated simple SCA analysis attack, originally published in [11] against simulated power traces of *kP* executions for the above mentioned 36 synthesized designs;
- show that the resistance of the synthesized ASICs against simple SCA depends significantly on the target frequency for which the designs were synthesized, i.e. we observed similar effects as for the FPGA implementations.

Please note that different technologies such as FPGAs and ASICs may have different characteristics when it comes to leakage, i.e. analyzing the behavior of ASICs is essentially needed to get a better understanding whether or not the leakage detected in the FPGA implementation can be found in ASIC implementations as well. So, we show that the leakage is not related to the application of an FPGA but that it is observable in ASICs as well.

The rest of this paper is structured as follows. In section II we describe our implementation of the Montgomery *kP* algorithm and its vulnerability to horizontal differential SCA attacks as well as its regularity that is a basis for resistance against simple SCA attacks. In section III we describe our setup for measuring electromagnetic traces on the attacked FPGA and give examples of the measured traces. In section

IV we explain how we automated the simple side channel analysis attack and evaluated the results of the attacks performed against the FPGA implementations for different target frequencies. In section V we describe the parameters of the designs synthesized for both IHP technologies, give details about the simulation of power traces and present the results of the performed attacks. Conclusions are given in section VI.

2 Our Implementation of the Montgomery *kP*

Our design is a hardware accelerator for the elliptic curve point multiplication for the NIST Elliptic Curve *B-233* [1] i.e. it performs only a *kP* operation. The scalar *k* is an up to 233 bit long binary number and $P=(x, y)$ is a point on EC *B-233*. The coordinates *x* and *y* are elements of the extended binary Galois field $GF(2^{233})$ with the irreducible polynomial $f(t)=t^{233}+t^{74}+1$ and can be represented as 233 bit long binary numbers. Implementing Algorithm 1 in hardware allows to reveal the second most significant bit k_{l-2} always via simple SCA. This is due to the fact that the initialization phase in Algorithm 1 contains initializing register Z_1 with the integer value 1: $Z_1 \leftarrow 1$ (see line 1 in Algorithm 1). Thus, the number of field multiplications with the multiplicand's value 1 depends on the key bit value processed in the main loop of the algorithm:

- If k_{l-2} is'1' two multiplications with the multiplicand $T=Z_1=1$ has to be performed (see line 4 in Algorithm 1).
- If k_{l-2} is'0' four such special multiplications are performed with the multiplicand $Z_1=1$, or $Z_1^2=1$, or $Z_1^4=1$ (see lines 7 and 8 in Algorithm 1).

A field multiplication with such a special operand consumes significantly less energy than a multiplication with usual operands and can easily be seen in the power traces if performed, i.e. the processing of the $k_{l-2}=1$ is distinguishable from the processing of the $k_{l-2}=0$.

To avoid the successful revealing of k_{l-2} we implemented Algorithm 2 based on publications [12, 13] i.e. we implemented a modified version of Algorithm 1. The modification done in Algorithm 2 in comparison to the Algorithm 1 refers to the initialization phase and the processing of k_{l-2}. The operation flow for processing k_{l-2} (see lines 2–8 in Algorithm 2) differs from the operation flow in the main loop (see lines 9–17). The processing of key bit k_{l-2} consists of 5 multiplications, 5 squarings, 3 additions and 8 write to register operations, independent of the value of k_{l-2}. Two dummy multiplications and two dummy squarings are performed for $k_{l-2}=0$ (see line 7, operations $U \leftarrow bX_2^4$ and $U \leftarrow TX_2$). In case $k_{l-2}=1$, one dummy write to register operation is necessary (see line 3 the operation: $T \leftarrow Z_2$). No operations are performed with an operand with the integer value 1.

Algorithm 2: Montgomery *kP* using projective Lopez-Dahab coordinates with processing of k_{l-2} before the main loop

```
Input: $k = (k_{l-1} \dots k_1 k_0)_2$ with $k_{l-1} = 1$, $P=(x,y)$ is a point of EC over $GF(2^l)$
Output: $kP = (x_1, y_1)$
 1: $X_1 \leftarrow x,\ X_2 \leftarrow x^4+b,\ Z_2 \leftarrow x^2$        } shortened initialization phase:
                                                                  the initialization of register $Z_1 \leftarrow 1$
                                                                  is not performed

 2:   if $k_{l-2}=1$
 3:     $T \leftarrow Z_2,\ Z_1 \leftarrow (X_1Z_2+X_2)^2,\ X_1 \leftarrow xZ_1+X_1X_2Z_2$
 4:     $T \leftarrow X_2,\ U \leftarrow bZ_2^4,\ X_2 \leftarrow X_2^4+U,\ U \leftarrow TZ_2,\ Z_2 \leftarrow U^2$
 5:   else                                                        } processing of $k_{l-2}$:
 6:     $T \leftarrow Z_2,\ Z_2 \leftarrow (X_1Z_2+X_2)^2,\ X_2 \leftarrow xZ_2+X_1X_2T$        the sequence of operations
 7:     $T \leftarrow X_1,\ U \leftarrow bX_2^4,\ X_1 \leftarrow X_1^4+b,\ U \leftarrow TX_2,\ Z_1 \leftarrow T^2$   differs from that
 8:   end if                                                        in the main loop

 9:   for $i=l-3$ downto 0 do
10:     if $k_i=1$
11:       $T \leftarrow Z_1,\ Z_1 \leftarrow (X_1Z_2+X_2T)^2,\ X_1 \leftarrow xZ_1+X_1X_2TZ_2$
12:       $T \leftarrow X_2,\ X_2 \leftarrow T^4+bZ_2^4,\ Z_2 \leftarrow T^2Z_2^2$
13:     else                                                      } main loop
14        $T \leftarrow Z_2,\ Z_2 \leftarrow (X_2Z_1+X_1T)^2,\ X_2 \leftarrow xZ_2+X_1X_2TZ_1$
15:       $T \leftarrow X_1,\ X_1 \leftarrow T^4+bZ_1^4,\ Z_1 \leftarrow T^2Z_1^2$
16:     end if
17:   end for

18: $x_1 \leftarrow X_1/Z_1$                                      } conversion of the result back to
19: $y_1 \leftarrow y+(x+x_1)[(X_1+xZ_1)(X_2+xZ_2)+(x^2+y)(Z_1Z_2)] / (xZ_1Z_2)$   affine coordinates;
20: return $(x_1, y_1)$                                           recovering of the y-coordinate
```

The structure of our design is shown in Fig. 1. Our design consists of the following blocks:

- ***ALU*** executes addition and squaring in $GF(2^{233})$;
- ***MULT*** performs the field multiplications;
- ***CONTROLLER*** manages the operation flow including storing the data into registers as well as reading it from the registers;
- ***BUS*** implements the data exchange between the blocks corresponding to the control signals from the ***CONTROLLER***;
- **9 REGISTERS** for the storing of input, output and intermediate data;
- ***INTERFACE*** for exchanging input/output data; the input data are the value of the scalar *k* and the affine coordinates of the EC point ***P*** to be processed and the output data are the affine coordinates of the ***kP*** result.

Figure 2 illustrates the processing sequence in the main loop of Algorithm 2.

In Algorithm 2 each bit of the scalar *k* is processed using 6 field multiplications (denoted as M1, M2, M3, M4, M5 and M6 in Fig. 2), 5 field squarings, 3 field additions and write to register operations. The block ***MULT*** takes two clock cycles for obtaining both multiplicands and 9 clock cycles to calculate a field product corresponding to the 4-segment Karatsuba multiplication method [14]. The intermediate values and the end results are stored into registers. The blocks ***ALU*** and ***MULT*** have internal registers. ***ALU*** has 1 register, ***MULT*** has 3 internal registers for storing of input, output and intermediate values. Each of these two blocks has its own $GF(2^{233})$ field reduction unit.

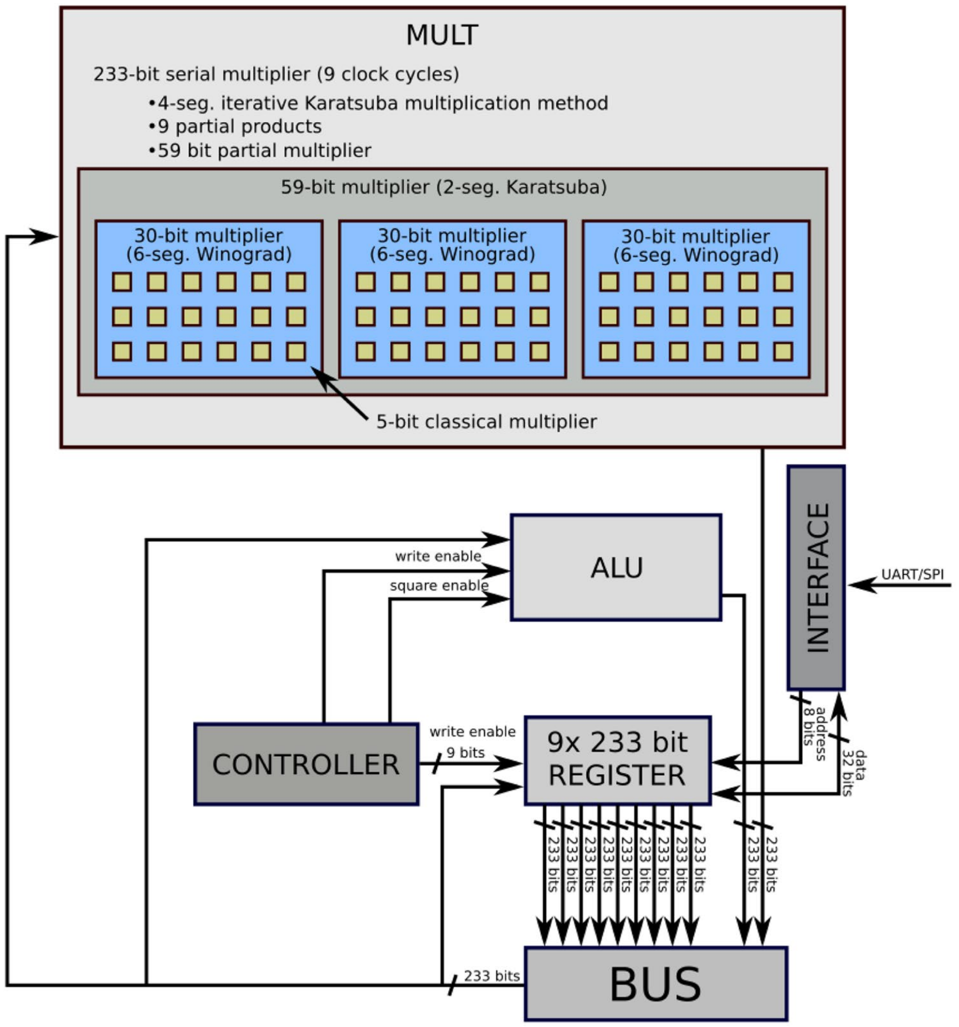

MULT
233-bit serial multiplier (9 clock cycles)
•4-seg. iterative Karatsuba multiplication method
•9 partial products
•59 bit partial multiplier
59-bit multiplier (2-seg. Karatsuba)
30-bit multiplier (6-seg. Winograd)
30-bit multiplier (6-seg. Winograd)
30-bit multiplier (6-seg. Winograd)
5-bit classical multiplier
ALU
write enable
square enable
INTERFACE
UART/SPI
CONTROLLER
write enable
9 bits
9x 233 bit REGISTER
address 8 bits
data 32 bits
233 bits
233 bits
BUS

Fig. 1 Structure of our *kP* design displaying all components

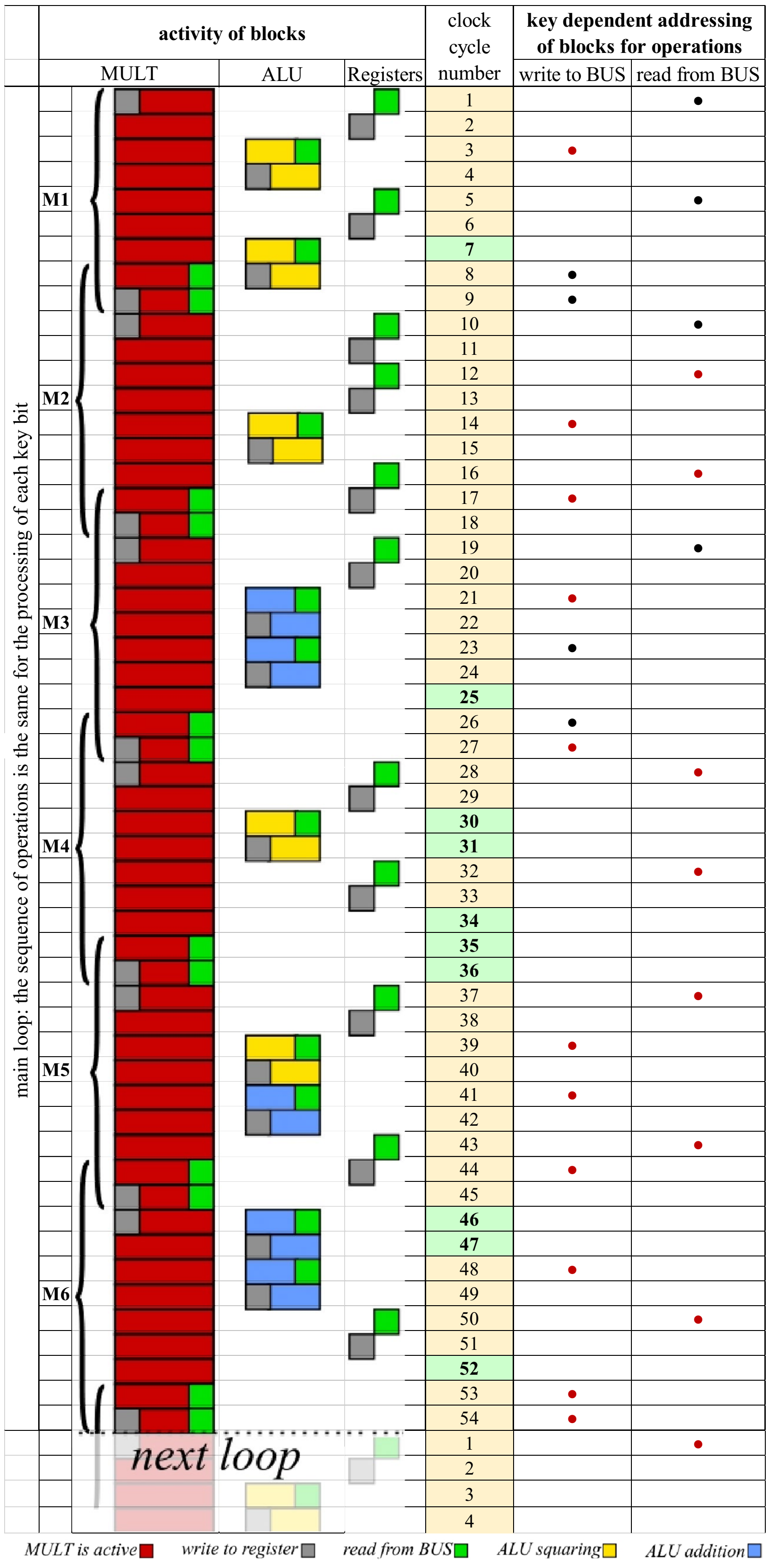
All blocks including the registers can write their outputs to the ***BUS*** that is realized as a multiplexer. It consists of many logic gates that react on the address given by the ***Controller***. The block ***Controller*** controls the data flow between the other blocks and defines which operation has to be performed in the current clock cycle. The *write to BUS* operation connects the output of the addressed block to the inputs of all other blocks. By the *read from BUS* operation only the addressed block accepts the values on its input as data for processing. Figure 2 shows the processing sequence in the main loop of our implementation providing details about activities of each block.

The rectangles in Fig. 2 represent different activities in our ECC design:

- red rectangles show the activity of the field multiplier;
- yellow rectangles show the squaring operation in the ***ALU***;
- blue rectangles show the field addition in the ***ALU***;
- small green squares correspond to the addressing of blocks for the read from BUS operation;
- grey squares represent the storing operation in registers.

Rectangles that are horizontally aligned are processed in parallel i.e. in the same clock cycle. The column denoted “clock cycle” shows the number of the clock cycles. In our implementation the main loop requires 54 clock cycles only. Our implementation is described in more detail in [15] and [16]. In this work we concentrate on exploiting its vulnerabilities by horizontal SCA attacks. Since the use of registers in the Montgomery *kP* algorithm depends on the value of the processed bit k_i of the scalar k, horizontal differential SCA attacks can be successful. Due to the nature of this SCA leakage – the addressing of the registers/blocks – this kind of attacks was denoted as horizontal bus and address bit DPA in [10]. Please note that in Fig. 2 the key dependent addressing of the blocks for the write to ***BUS*** and for the read from ***BUS*** operations are shown for each clock cycle in our implementation using solid dots in red and black respectively.

The red dots in the “write to BUS” and “read from BUS” columns in Fig. 2 mark strong SCA leakage sources. For example in clock cycle 3 register Z_2 has to write its content



clock cycle number	key dependent addressing of blocks for operations: write to BUS	key dependent addressing of blocks for operations: read from BUS
1		●
2		
3	🔴	
4		
5		●
6		
7		
8	●	
9	●	
10		●
11		
12		🔴
13		
14	🔴	
15		
16		🔴
17	🔴	
18		
19		●
20		
21	🔴	
22		
23	●	
24		
25		
26	●	
27	🔴	
28		🔴
29		
30		
31		
32		🔴
33		
34		
35		
36		
37		🔴
38		
39	🔴	
40		
41	🔴	
42		
43		🔴
44	🔴	
45		
46		
47		
48	🔴	
49		
50		🔴
51		
52		
53	🔴	
54	🔴	
1		🔴
2		
3		
4		

Fig. 2 Processing sequence in the main loop of our implementation providing details about activities of each block and SCA-critical addressing of the blocks. Red rectangles show the activity of the field multiplier, yellow rectangles show the squaring operation in the ALU, blue rectangles show the field addition in the ALU, small green squares correspond to the addressing of blocks for the read from BUS operation and grey squares represent the storing the data into registers

to the ***BUS*** if $k_i=1$. If $k_i=0$ register Z_1 is selected to write its content to the ***BUS***. This fact is denoted using the red point in the "write to BUS" column in clock cycle number 3. In the same clock cycle the block ***ALU*** reads the value from the ***BUS***. This operation is performed independent of the key bit value which is the reason why the cell in the column "read from BUS" in clock cycle 3 is empty. The black dots represent a more complex dependence of the block addressing. It takes not only the currently processed key bit value into account but also its previous value i.e. the black dots denote less strong SCA leakage sources than the red dots do. The ***BUS*** reacts on the key dependent address of blocks selected for the write to ***BUS*** operation: the ***BUS*** consumes a key bit value dependent energy in the clock cycles marked with the red or black dots as well as in the immediately following clock cycles. Similar processes occur for the read from ***BUS*** operations. The clock cycles with key dependent addressing are denoted by a light yellow background in Fig. 2. A statistical analysis of the power consumption or electromagnetic emanation of the design in these clock cycles can be exploited for successfully revealing the key. But such attacks are classified as differential SCA, not as a simple SCA attacks. It is expected that simple analysis of traces i.e. attacks using simple visual inspection of traces will not provide reasonably good key extraction results.

3 Measurement Details

The design described in the previous section was synthesized for the Arty Z7-20 board with a Zynq SoC (xc7z-020clg400-1) using Vivado 2018.3 for 7 different clock frequencies: 10 MHz, 50 MHz, 100 MHz, 160 MHz, 200 MHz, 240 MHz and 250 MHz. The default synthesis/implementation strategies allow to get the design with a maximum operating frequency of 200 MHz [17].

We applied the "Flow_PerfOptimized_high" synthesis strategy and "Performance_ExplorePostRoutePhysOpt" implementation strategy in order to make it working faster. The main parameters of the synthesized designs are shown in the Table 1.

We performed a functionality test and measurements of the electromagnetic emanation of the *kP* executions using always the same input data i.e. with the same scalar *k* and the same EC point *P*. The success of simple SCA attacks[1] will be especially high if the key consists of the same number of '0' and '1' bits. We proved this fact on different keys experimentally. In this work, we demonstrated our investigations on the example of a single key.

In all the cases, except for the highest clock frequency of 250 MHz, the result of the *kP* operation was calculated correctly.

[1] Please recall that simple SCA attacks always use only a single trace.

Table 1 Main parameters of designs synthesized for optimized performance

Design frequency, MHz	LUTs as logic out of 53,200 (utilization in %)	Slice Registers out of 106,400 (utilization in %)
10	5831 (10.96%)	3706 (3.48%)
50	5829 (10.96%)	3706 (3.48%)
100	5833 (10.96%)	3706 (3.48%)
160	5963 (11.21%)	3756 (3.53%)
200	6009 (11.30%)	3750 (3.52%)
240	5989 (11.26%)	3735 (3.51%)
250	6037 (11.35%)	3776 (3.55%)

We performed simple EMA attacks for the properly functioning designs only, i.e. we excluded the design synthesized for 250 MHz from our experiments. For each of the 6 designs we captured an electromagnetic trace during one *kP* execution using the near-field probe MFA-R 0.2–75 from Langer and a LeCroy HDO9404-MS oscilloscope with a maximum sampling rate of 40 GS/s. The measurement setup is shown in Fig. 3.

Approximate times for the *kP* execution and the parameters of the measurement setup are given in Table 2.

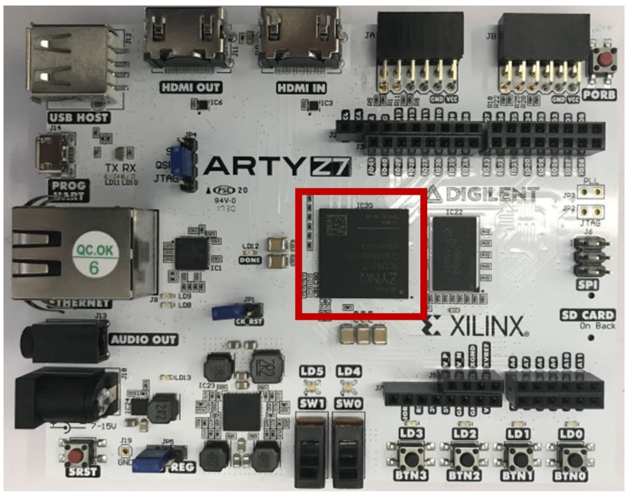
The design operating at a frequency of 240 MHz allows us to perform about 18,500 scalar multiplications per second for the elliptic curve B-233. The screenshots of the captured traces for the six investigated designs are shown in Fig. 4.

The zoomed-in part on each oscillogram represents a fragment of a trace in which about 9 key bits are processed in the main loop. When capturing the traces we used different sampling rates for different designs in order to assure a fair assessment of the attack results. This led to a similar amount of samples captured per clock cycle for all frequencies.[2] As it can be seen in Fig. 4, the shape of the measured trace as well as its amplitude depend significantly on the running frequency.

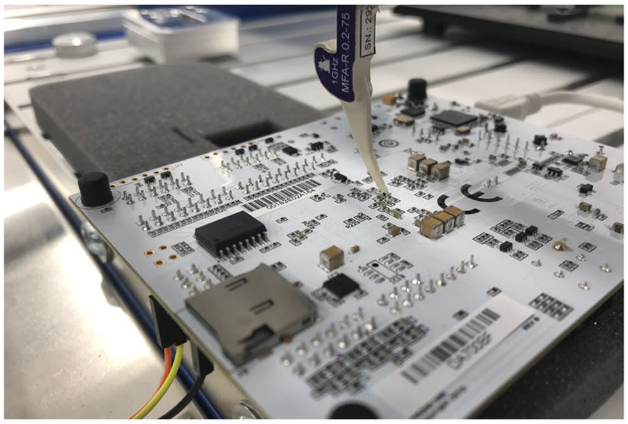
4 Automated Simple EMA Attack

We concentrate in this paper on horizontal analysis attacks that are also called single-trace attacks. Here we performed an automated simple analysis attack. In [10] horizontal differential analysis attacks are described. The difference to [10] is that in this paper we do not calculate any statistical parameters such a mean value, variation, etc. The difference to usual simple analysis attacks is that we automated our attack with the goal to make the attack effective and fast.

[2] We attempted to perform a fair assessment for the simulated as well as for the measured data. So we tried to achieve the same number of samples per processed key for all the simulated cases. In addition we wanted to get as close as possible to the sampling rate applied during our measurements, also to ensure fair comparison. Even though oversampling may not bring much benefit it does not cause harm. Undersampling however may cause information to be missed.



a)



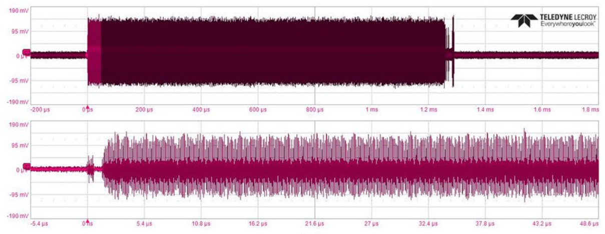
b)

Fig. 3 Device under attack – (**a**) Arty Z7 board from Digilent with a Zynq SoC (highlighted by the red rectangle) and (**b**) the measurement place (decoupling capacitor for the FPGA core logic) on the back side of the board

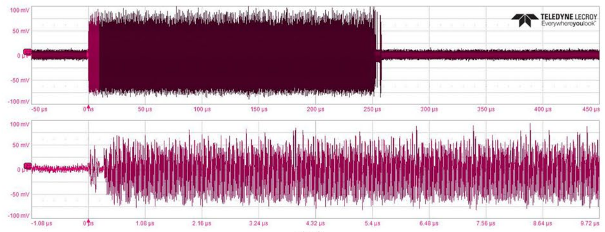
Usually, when running simple power analysis (SPA) or simple electromagnetic analysis (SEMA) the attacker assumes that the attacked design is a bitwise processing of the secret binary number i.e. the *key*, whereby the processing of a key bit value '0' differs from the processing of a key bit value '1'.

Table 2 Measurement parameters and *kP* execution time

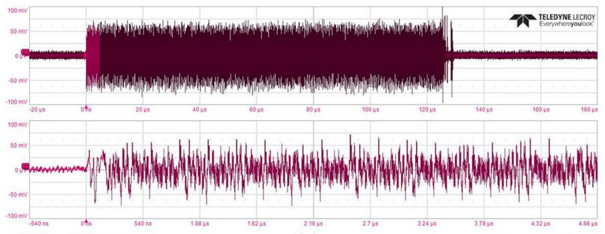
Design frequency, MHz	Sampling rate, GS/s	Samples/ clock cycle	Approximate *kP* execution time, µs	Number of *kP* operations per second
10	2.5	250	1300	769
50	10	200	260	3846
100	20	200	130	7692
160	40	250	81	12,345
200	40	200	65	15,384
240	40	~166.7	54	18,518



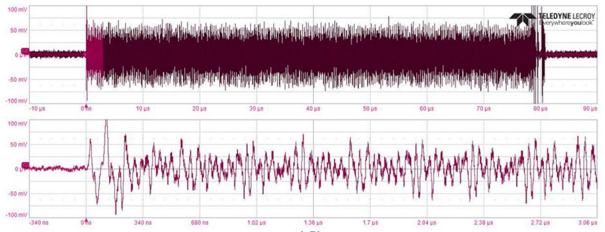
(a)



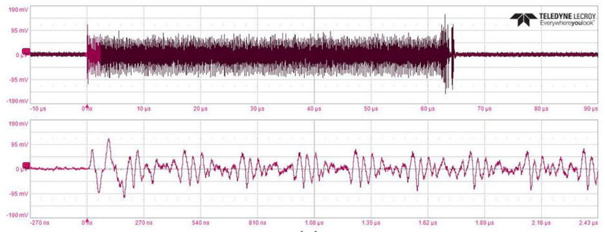
(b)



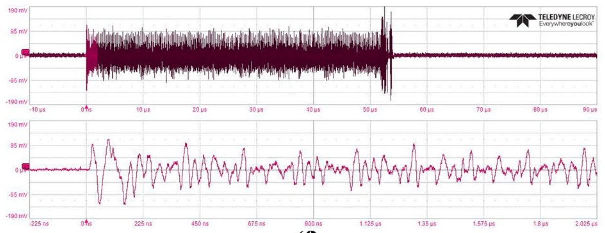
(c)



(d)



(e)



(f)

Fig. 4 Screenshots of the captured traces of the *kP* execution of the designs running at 10 MHz (**a**), 50 MHz (**b**), 100 MHz (**c**), 160 MHz (**d**), 200 MHz (**e**), 240 MHz (**f**) clock frequencies

Due to these assumptions the attacker can apply the following principles in simple analysis attacks:

- The shapes of the processing of key bit values ‘0’ look similar to each other;
- The shapes of the processing of key bit values ‘1’ look similar to each other;
- The shapes of the processing of key bit values ‘0’ are distinguishable from the shapes of the processing of key bit values ‘1’ using simple visual inspection;
- The analysed trace is a set of two different kinds of shapes i.e. a sequence of ‘0’-shapes and ‘1’-shapes;
- The correctly recognized/extracted/revealed sequence of ‘0’- and ‘1’-shapes corresponds to the used secret i.e. to the scalar *k*.

Thus, when executing a simple analysis attack the attacker looks at the measured trace and tries to apply the above listed principles i.e. the instruments used for the analysis are the eyes of the attacker and his natural intelligence. Due to this fact, the success of simple analysis attacks depends directly on the distinguishability of the ‘0’- and ‘1’-shapes. If the difference is significant and can be easily seen the sequence of the shapes in the trace is clear. The key candidate extracted using this clear sequence matches with the real processed key 100% i.e. all bits of the key are revealed correctly. The number of correctly revealed key bits can be used as a criterion of the success of the attack if the attacker knows the actual processed key $k=k_{l-1}k_{l-2}\ldots k_1k_0$. Here l is the length of the key k. The attacker knows the processed scalar only if he analyses the trace measured during signature verification corresponding to ECDSA. This is feasible as the scalars processed when verifying a signature are not secret, i.e. they can be derived from a part of the signature transmitted as a plain text.

Designers (and we too) can use their knowledge about the scalar processed for testing the resistance of their designs i.e. we compare each extracted key candidate $k_{candidate}$ with the processed key k. The correctness δ_1 can be expressed for each extracted key candidate as the relative number of the bits correctly revealed:

$$\delta_1 = \frac{\#correct\_extracted\_bits\left(k_{candidate}\right)}{l} \cdot 100\% \quad (1)$$

Please note that the case $\delta_1 = 0\%$ means that the evaluated key candidate is completely wrong and that the opposite (i.e. the inverted) key candidate will be 100% correct.[3] Taking this fact into account we can calculate the correctness for each key candidate as a value between 50 and 100% as follow:

$$\delta = 50\% + |50\% - \delta_1| \quad (2)$$

The best key candidate is then the one with the maximal correctness. We apply this maximal correctness for evaluating the success of the performed attack. For example if we obtain a large set of key candidates and only one of them has a correctness of 100% it means the attack was 100% successful because the processed key was completely revealed.

It is obvious that the success of the simple analysis attack depends extremely on the length of the analysed trace, distinguishability of ‘0’- and ‘1’-shapes in the trace as well as on the experience of the attacker, whereby the more regular the implemented algorithm is the less distinguishable are ‘0’- and ‘1’-shapes. Even in case of white box cryptography i.e. if a designer with the full knowledge of the implementation details and the processed key tries to perform a simple analysis attack, the simple analysis attack can be a complex task. This can be illustrated using parts of the traces shown in Fig. 4. In the first step the attacker tries to separate the trace into slots i.e. into parts that correspond to the processing of a single key bit. The separation of the trace into slots can be a complex and non-trivial task, especially for non-experts, see for example the zoomed-in traces shown in Fig. 4a, b. This task is more easy for the zoomed-in traces shown in Fig. 4d, f. If slots are successfully separated the attacker tries to classify them into “similar” and “different” slots i.e. the attacker compares the slots with each other and then extracts the sequence of ‘0’- and ‘1’-shapes.

Figure 5 shows the zoomed-in traces shown in Fig. 4d, f separated into slots as well as the processed key bit values.

After splitting the traces into slots the ‘0’- and ‘1’-shapes in the lower trace in Fig. 5 can be easily distinguished from each other. The distinguishability “symptoms” are in the middle of the slots (see black circles in Fig. 5) and at the end of the slots (see green rectangles in Fig. 5).

Please note that our design – i.e. the VHDL code implementing the algorithm – is highly regular, see details in section II. The successful simple analysis attack was not expected and was highly surprising. Working on a fast FPGA implementation of the Montgomery *kP* algorithm we synthesized and analysed the design for the highest frequency 240 MHz first. After a successful SEMA attack we decided to evaluate the resistance of the same VHDL implementation synthesized for other frequencies. Thus, we synthesized the design for 5 different frequencies. To be sure that the SEMA is successful and to accelerate the attack we automated the analysis. We separated the analysed trace into slots and overlapped all shapes in MatLab, whereby we marked all ‘0’-shapes blue and all ‘1’-shapes orange, see Fig. 6. Each slot contains 9000 samples for the trace of the 240 MHz design.

[3] Designers perform attacks with the goal to determine leakage sources. A key candidate extracted with a correctness of 100% as well as a key candidates extracted with a correctness of 0% are clear markers for a strong side channel leakage. Thus, we “inverted” the key candidates with a correctness < 50% and used only the “shorted” range [50%, 100%] for the evaluation of the attack success.

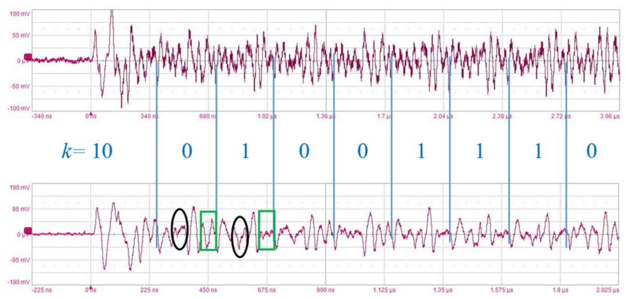


Fig. 5 Zoomed-in traces shown in Fig. 4d, f separated into slots and the processed key bit values

Figure 6 illustrates how different ‘0’- and ‘1’-shapes are. The distinguishability “symptoms” can also be easily seen in the middle and at the end of the slots. Moreover, there are parts in the slots in which the set of all ‘0’-shapes is completely separated from the set of ‘1’-shapes. These parts are marked with black rectangles in Fig. 6 and are shown zoomed-in in the upper part of Fig. 6. We used the fact that the set of all ‘0’- shapes is completely separated from the set of ‘1’-shapes for automating the analysis i.e. we programmed the recognition of such “gaps” between the sets of blue and orange lines. Each sample with such a “gap” allows to clearly distinguish all key bits ‘0’ from all key bits ‘1’ which causes the 100% success of the attack i.e. the number of samples with “gaps” corresponds to the number of key candidates with a correctness $\delta = 100\%$. Due to this fact we define the following additional evaluation criteria for a successful attack:

- number of samples with “gaps”
- maximal “gap” distance

Table 3 shows the results of our automated simple attack performed against each design synthesized applying performance optimizing compile options.

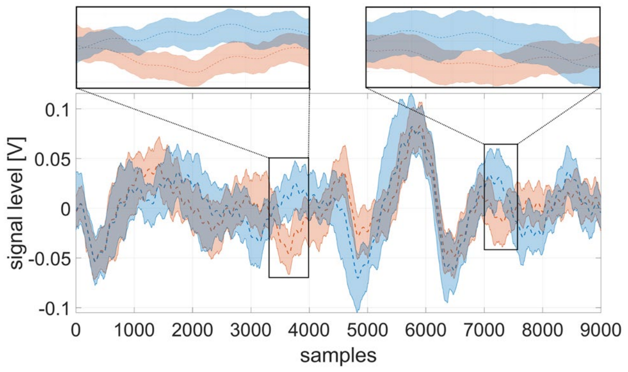


Fig. 6 Trace of the design synthesized for and running at a 240 MHz clock frequency: the trace was partitioned into slots; all shapes were overlapped; all ‘0’-shapes are marked blue; all ‘1’-shapes are marked orange. The dark blue line represents the mean line for all blue lines; the red line line represents the mean line for all orange lines

Table 3 Results of automated SEMA attack against performance optimized designs

Frequency, MHz	samples per slot: N	number of samples with „gaps“, i.e. number of key candidates with correctness $\delta = 100\%$		max „gap“ distance, V
		as a number n	as a relative number n/N, %	
10	13,500	0	0%	0
50	10,800	40	0.37%	0.00616
100	10,800	126	1.17%	0.00983
160	13,500	296	2.19%	0.00451
200	10,800	192	1.78%	0.00274
240	9000	304	3.38%	0.01649

We repeated the attack against traces obtained for designs with default synthesis/implementation strategies for clock frequencies up to 100 MHz to confirm that the attack success does not significantly depend on the applied strategies. Table 4 shows the results of our automated simple attack performed against each design synthesized with the default options.

The parameters of the designs synthesized for the default strategies are shown in the Table 5.

As it can be seen, the utilization of FPGA resources when applying default strategies is almost the same as for performance optimized strategies.

Figure 7 represents graphically the data given in Tables 3 and 4, i.e. it shows the number of key candidates with correctness $\delta = 100\%$ for all designs attacked.

Results shown in Tables 3 and 4; Fig. 7 should alarm all designers: the Montgomery *kP* algorithm that is due to its regularity in the literature declared as resistant against simple SCA is not resistant against simple automated analysis for almost all investigated frequencies. This fact has to be taken into account i.e. designers have to evaluate the resistance of their implementations for each target frequency in order to know for sure how vulnerable their implementations are.

Table 4 Attack results for designs with default compile options

Frequency, MHz	samples per slot: N	number of samples with „gaps“, i.e. number of key candidates with correctness $\delta = 100\%$		max „gap“ distance, V
		as a number n	as a relative number n/N	
10	13,500	0	0%	0
20	13,500	0	0%	0
40	13,500	7	0.05%	0.00428
50	10,800	18	0.17%	0.00368
80	13,500	78	0.58%	0.01053
100	10,800	107	0.99%	0.00554

Table 5 Main parameters of the designs synthesized for default strategies

Design frequency, MHz	LUTs as logic out of 53,200 (utilization in %)	Slice Registers out of 106,400 (utilization in %)
10	5833 (10.96%)	3706 (3.48%)
20	5834 (10.97%)	
40	5833 (10.96%)	
50	5831 (10.96%)	
80	5834 (10.97%)	
100	5833 (10.96%)	

5 Attacking Asic Designs

We decided to perform additional investigations for our design when synthesized as an ASIC with the goal to investigate if the dependence of the design resistance from the target frequency is observable on platforms other than FPGAs. We synthesized the design described in section II for different frequencies using the gate library for the IHP 130 nm and the IHP 250 nm CMOS technologies [18]. For the synthesis we used Synopsys (version K-2015.06-SP2) with the simple *compile* option. Therefore, the logic-level and gate-level synthesis of the design as well as its optimization were performed with the goal to achieve the smallest possible chip area for the given timing requirements. For the simulation we used NC Sim from Cadence (version 12.10-s003) and all the power traces were simulated with a PrimeTime suite from Synopsys (version Q-2019.12-SP1).

In contrast to the measured traces, where only the current value is recorded, power traces simulated for a selected time step contain the mean value of the power consumed in the time step. In other words, the energy of gate switching is"accumulated" over each time interval equal to the simulation step and represented as the mean power for such a time step. Thus, the simulated traces have no information losses.

Table 6 Parameters of the synthesized designs for the 250 nm technology

Clock cycle period, ns	Derived simulation step, ps	Design frequency, MHz	Design area, mm^2	Total Power, mW
250	1000	4	1.384175	4.92
100	400	10	1.384267	12.3
50	200	20	1.384246	24.6
25	100	40	1.384422	49.0
20	80	50	1.384408	61.4
12.5	50	80	1.384507	98.6
10	40	100	1.384344	123
8	32	125	1.385255	156
7	28	~143	1.385452	175
6.25	25	160	1.387625	194
6	24	~167	1.388599	203
5.5	22	~182	1.394378	224
5	20	200	1.399261	256
4.5	18	~222	1.413831	294
4	16	250	1.435373	339
3.5	14	~286	1.559121	414

In our simulations we decided to apply a different time step depending on the target design frequency with the goal to have the same number of simulated points per clock cycle for each frequency. The derived simulation time step and corresponding values for the consumed power were obtained from the applied fine-grained simulation step by compressing/integrating of simulated values. We selected 250 samples per clock cycle as a representative number that we can derive for each of investigated traces. By using 250 derived simulated values for a single clock cycle we obtained 13,500 simulated values per slot for each of the investigated frequencies. This allows us to perform a fair comparison of the attack results obtained by using the traces measured on FPGA and simulated for an ASIC.

Parameters of the designs synthesized for the IHP 250 nm and 130 nm technologies are given in Tables 6

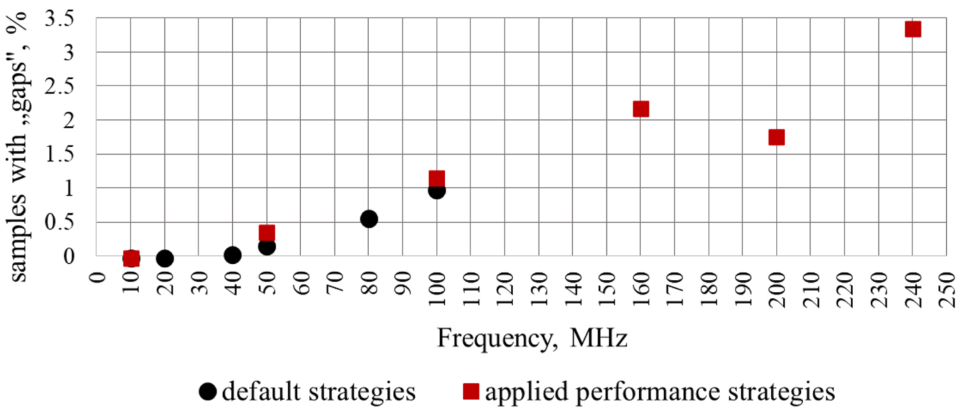


Fig. 7 Graphical representation of the attack results given in Tables 3 and 4

Table 7 Parameters of the synthesized designs for the 130 nm technology

Clock cycle period, ns	Derived simulation step, ps	Design frequency, MHz	Design area, mm^2	Total Power, mW
250	1000	4	0. 275,368	0.649
100	400	10	0. 275,361	1.62
50	200	20	0. 275,368	3.24
25	100	40	0. 275,368	6.47
20	80	50	0. 275,368	8.09
12.5	50	80	0. 275,377	12.9
10	40	100	0. 275,375	16.2
8	32	125	0. 275,375	20.1
7	28	~143	0. 275,375	23.0
6.25	25	160	0. 275,375	25.8
6	24	~167	0. 275,375	26.8
5.5	22	~182	0. 275,380	29.3
5	20	200	0. 275,380	32.4
4.5	18	~222	0. 275,456	35.9
4	16	250	0. 275,480	40.8
3.5	14	~286	0. 276,043	46.6
3.25	13	~308	0. 277,974	49.9
3	12	~333	0. 280,316	54.5
2.75	11	~364	0. 284,981	59.9
2.5	10	400	0. 295,053	66.8

and 7, respectively. The maximum achieved operating frequency for the design synthesized for the 250 nm technology is close to 286 MHz (3.5 ns clock cycle period), meanwhile it is 400 MHz for the 130 nm technology.

Figure 8 shows several clock cycles of the power traces synthesized for the designs with a 4 MHz operating frequency for the 130 nm technology (green line) and the 250 nm technology (blue line), as well as a zoomed-in part of a single clock cycle for each of the technologies. As the 130 nm technology is faster than the 250 nm one, the switching of gates for the 130 nm stops earlier compared to the 250 nm technology, which can be seen in the zoomed-in part.

We performed automated simple power analysis attacks against all the traces simulated for both IHP technologies, exactly in the same way as described in Section IV. Attack results are given in Table 8.

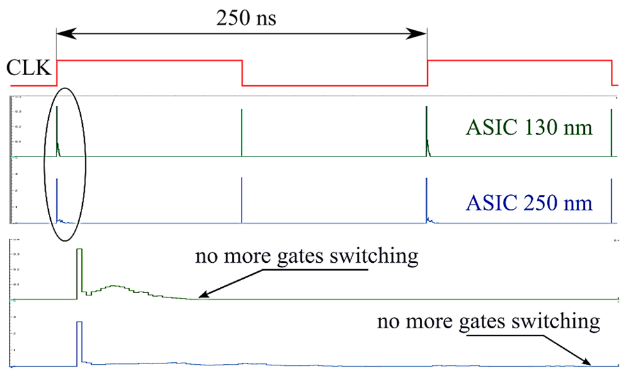


Fig. 8 A part of the power traces synthesized for the IHP 130 nm (green line) and the IHP 250 nm (blue line) technologies and 4 MHz operating frequency

Figure 9 represents graphically the data given in Tables 8 and 3, i.e. it shows the relative number of key candidates with correctness δ = 100% for all attacked ASIC designs and FPGA designs with applied performance strategies. As the maximum relative number of samples with "gaps" for 130 nm technology is less than 1%, Fig. 10 represents graphically the data from Table 8 using different scales of the axis for the 250 nm and the 130 nm technology, respectively.

The interesting fact is that the correctness of keys revealed shows the same trend for the FPGA and ASIC designs in the 250 nm technology in the frequency range of up to 200 MHz. The number of points in an analysed trace, which are strong leakage sources allowing to reveal the key, is increasing for the frequencies up to 160 MHz and starts to decrease in the frequencies interval from 160 to 200 MHz. For the 130 nm technology the maximum number of such points is achieved at 366 MHz frequency and is decreasing significantly at 400 MHz.

The reduced number of points representing strong leakage for the 130 nm technology compared to the 250 nm technology may be explained not only by the use of different technologies but also by the fact, that 130 nm technology is significantly faster, i.e. the time in each clock cycle that corresponds to the active gate switching is by far shorter for the 130 nm technology than the one for the 250 nm. Therefore, the effective number of simulated values where the gates are switching is less for the 130 nm than for the 250 nm technology (see zoomed-in part of the Fig. 8).

Table 8 Attack results for the ASIC designs

Frequency, MHz	samples per slot: N	number of samples with „gaps“, i.e. number of key candidates with correctness $\delta = 100\%$			
		for the 250 nm technology		for the 130 nm technology	
		as a number n	as a relative number n/N, %	as a number n	as a relative number n/N, %
4	13,500	14	0.103704	0	0
10		71	0.525926	0	0
20		143	1.059259	2	0.014815
40		221	1.637037	4	0.02963
50		299	2.214815	4	0.02963
80		464	3.437037	7	0.051852
100		625	4.62963	11	0.081481
125		771	5.711111	14	0.103704
~143		853	6.318519	18	0.133333
160		998	7.392593	16	0.118519
~166		960	7.111111	17	0.125926
~182		893	6.614815	20	0.148148
200		526	3.896296	31	0.22963
~222		514	3.807407	28	0.207407
250		95	0.703704	26	0.192593
~286		45	0.333333	39	0.288889
~308		n/a	n/a	40	0.296296
~333		n/a	n/a	45	0.333333
~364		n/a	n/a	79	0.585185
400		n/a	n/a	17	0.125926

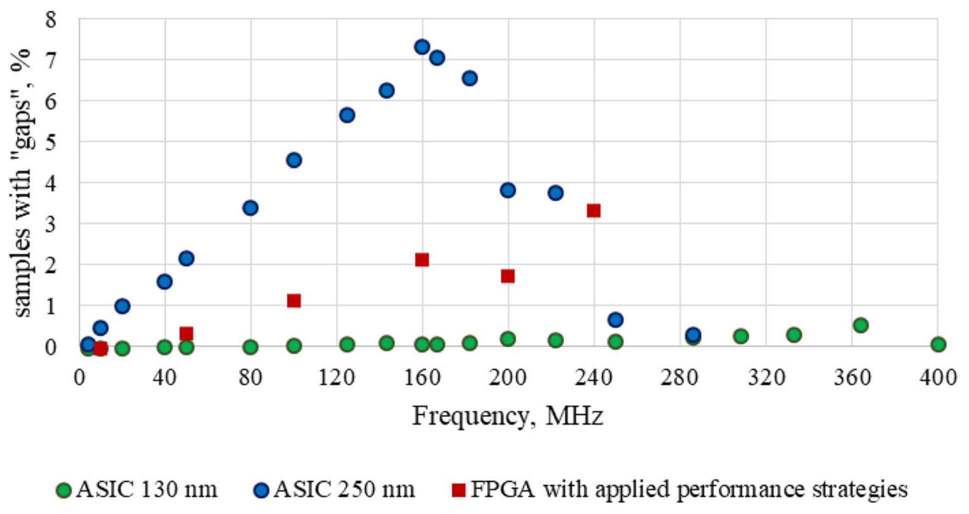


Fig. 9 Graphical representation of the attack results given in Tables 3 and 8

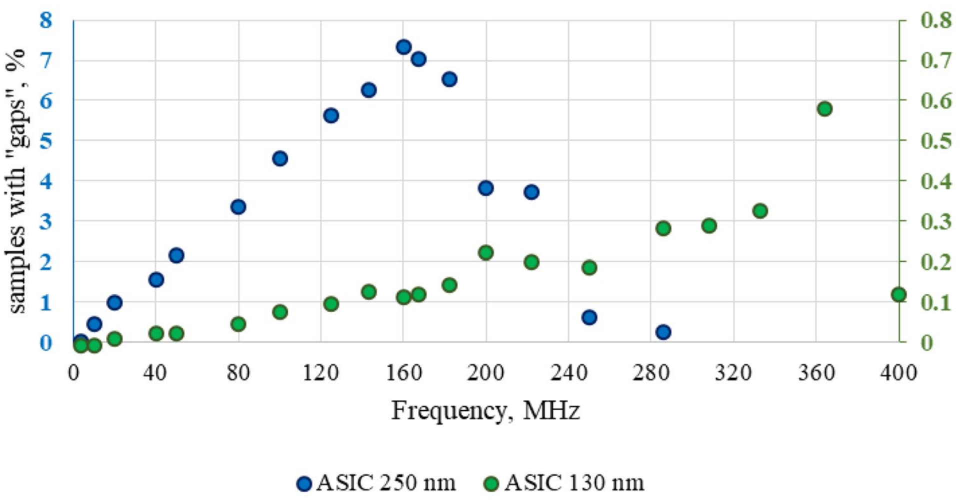


Fig. 10 Graphical representation of the attack results given in Table 8

6 Conclusions

In this paper we reported on a serious issue concerning the SCA resistance of the Montgomery algorithm, i.e. its vulnerability at a certain interval of frequencies that we call "middle" frequencies. This is especially important because the literature reports that this algorithm is resistant against simple side channel analysis attacks. We recorded electromagnetic traces of our own Montgomery implementation that for sure adheres to the regularity principle. We even could extract a key correctly by optical inspection for designs running on an FPGA at higher frequencies. In order to speed up the analysis process we automatized the simple side channel analysis. In addition we synthesized our design for frequencies from 10 to 240 MHz using performance optimization compile options and for 10 to 100 MHz with standard compile options. In both sets of traces, we were able to reveal the key completely, i.e. to extract key candidates with a correctness of 100% for designs running above 50 MHz and above 40 MHz, respectively. The number of key candidates with a correctness of 100% rises with the frequency at which the design is running from 7 at 40 MHz to 107 at 100 MHz for standard compile options. The performance optimized versions show the same trend with 40 correctly extracted key candidates at 50 MHz and up to 304 at 240 MHz. Please note that the vulnerabilities were detected at the frequencies for which the designs for FPGA were synthesized and are not due to overclocking during the execution of the *kP* operation.

In order to investigate if the vulnerability of the design depending on the target frequency is observable also on other platforms than FPGAs we synthesized our *kP* design for different frequencies using gate libraries for two different IHP CMOS technologies using the simple *compile* option. Applying this option the area of the designs was optimized. We synthesized our design for 16 different frequencies for the IHP 250 nm technology and for 20 frequencies for the IHP 130 nm technology and analysed the 16 + 20 = 36 simulated power traces. We decided to generate and analyse so many different versions to ensure a fair and detailed assessment of the frequency dependent vulnerability. We observed a similar trend as for the FPGA implementation: for very low and very high frequencies only few points in analysed traces represent strong leakage sources allowing to reveal the key successfully. For the "middle" frequencies the number of points of the traces that allow to successfully reveal the key increases with increasing frequency.

We are aware of some issues of the Montgomery ladder. It is vulnerable to the horizontal Address Bit attack which was presented in [3, 16]. The main leakage source is the key dependent addressing of the registers in the algorithm. But even though we have a pretty good understanding of this algorithm and the issues when implementing it, we cannot yet explain the behaviour discussed in this paper. But the fact that the vulnerability of the Montgomery ladder increases with the execution frequency is a severe vulnerability. In our future work we will research the reasons behind this behaviour.

Please note that our analysis of measured and simulated traces clearly indicates that when it comes to SCA no assumption about behaviour of algorithms may be taken for granted and as a consequence of this designers need to verify SCA resistance of their implementations thoroughly for each target frequency. This is also our reason to report the vulnerability here without being capable to fully explain it.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Federal Information Processing Standard (FIPS) (2013) Digital Signature Standard; Request for Comments on the NIST-Recommended Elliptic Curves: 186–184. https://doi.org/10.6028/NIST.FIPS.186-4
2. Hankerson D, Menezes A, Vanstone S (2004) Guide to Elliptic Curve Cryptography. Springer-Verlag New York, Inc., ISBN 0-387-95273-X
3. Kabin I, Dyka Z, Kreiser D, Langendoerfer P (2018) Horizontal Address-Bit DEMA against ECDSA. In: 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS), p 1–7
4. NXP Application note AN11875: A1006 Host Reference Implementation for LPC1115. Rev. 0.2 — 10 August 2016
5. López J, Dahab R (1999) Fast Multiplication on Elliptic Curves Over *GF(2m)* without precomputation. In:. Koç ÇK, Paar C (eds) Cryptographic Hardware and Embedded Systems, vol. 1717. Berlin, Heidelberg: Springer Berlin Heidelberg, p 316–327
6. Hankerson D, López J, Menezes A (2000) Software Implementation of Elliptic Curve Cryptography over Binary Fields. In Cryptographic Hardware and Embedded Systems — CHES 2000, p 1–24. https://doi.org/10.1007/3-540-44499-8_1
7. Joye M, Yen S-M (2002) The Montgomery Powering Ladder. Cryptographic Hardware and Embedded Systems - CHES 2002:291–302
8. Fan J, Guo X, De Mulder E, Schaumont P, Preneel B, Verbauwhede I (2010) State-of-the-art of secure ECC implementations: a survey on known side-channel attacks and countermeasures. In: 2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), p 76–87
9. Itoh K, Izu T, Takenaka M (2002) Address-Bit Differential Power Analysis of Cryptographic Schemes OK-ECDH and OK-ECDSA. In: Cryptographic Hardware and Embedded Systems - CHES 2002, p 129–143
10. Kabin I, Dyka Z, Kreiser D, Langendoerfer P (2017) Horizontal address-bit DPA against montgomery kP implementation. In: 2017 International Conference on ReConFigurable Computing and FPGAs (ReConFig), p 1–8
11. Kabin I, Dyka Z, Aftowicz M, Klann D, Langendoerfer P (2020) Resistance of the Montgomery kP Algorithm against Simple SCA: Theory and Practice. In 2020 IEEE Latin-American Test Symposium (LATS), p 1–6. https://doi.org/10.1109/LATS49555.2020.9093678
12. Bock EA, Dyka Z, Langendoerfer P (2016) Increasing the Robustness of the Montgomery kP-Algorithm Against SCA by Modifying Its Initialization. In: Innovative Security Solutions for Information Technology and Communications. Springer, Cham, p 167–178
13. Dyka Z, Bock EA, Kabin I, Langendoerfer P (2016) Inherent Resistance of Efficient ECC Designs against SCA Attacks," in 2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS), p 1–5
14. Dyka Z, Langendoerfer P (2005) Area efficient hardware implementation of elliptic curve cryptography by iteratively applying Karatsuba's method. In: Design, Automation and Test in Europe, Vol.3. p 70–75
15. Alpirez Bock E (2015) SCA resistent implementation of the Montgomery kP-algorithm. Master thesis. https://opus4.kobv.de/opus4-btu/files/3628/Estuardo_AlpirezBock.pdf
16. Kabin I, Kreiser D, Dyka Z, Langendoerfer P (2018) FPGA Implementation of ECC: Low-Cost Countermeasure against Horizontal Bus and Address-Bit SCA. In: 2018 International Conference on ReConFigurable Computing and FPGAs (ReConFig), p 1–7
17. Kabin I, Sosa A, Dyka Z, Klann D, Langendoerfer P (2019) On the Influence of the FPGA Compiler Optimization Options on the Success of the Horizontal Attack. In: 2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig), p 1–5. https://doi.org/10.1109/ReConFig48160.2019.8994807
18. IHP GmbH (2020) Low-Volume & Multi-Project Service. [Online]. Available: https://www.ihp-microelectronics.com/de/services/mpw-prototyping/sigec-bicmos-technologies.html. Accessed 17 Sep 2020

Ievgen Kabin received the Diploma degree in Electronic Systems from the NationalTechnical University of Ukraine ''Kyiv Polytechnic Institute'', Kiev, Ukraine, in 2009. From 2009 to 2010, he was a Lead Engineer with the state-owned enterprise, Scientific Production Center of Energy-efficient Designs and Technologies, ("Tehnoluch"). From 2010 to 2015, he was a Junior Researcher with the E.O. Paton Electric Welding Institute, National Academy of Sciences of Ukraine. Since 2015, he is with IHP-Leibniz Institut für innovative Mikroelektronik, Frankfurt (Oder), Germany. There he is a member of the research group Sensor Networks and Middleware Platforms and has published more than 15 articles. His research interests include side-channel analysis attacks and investigations in the field of Elliptic Curve Cryptography.

Zoya Dyka received her Diploma degree in Radiophysics and Electronics from Taras Shevchenko University Kiev, Ukraine in 1996 and the Ph.D. degree from TechnicalUniversity of Cottbus-Senftenberg, Germany in 2012. Since 2000 she is with the IHP in Frankfurt (Oder). Starting 2013, she was leading a young researcher group in the field of tamper resistant crypto ICs. Since 2018, she is leading a research group in the field of resilient CPSoS. She has authored more than 30 peer reviewed technical articles and filed five patents in the security area of which four are granted. Her research interests include design of efficient hardware accelerators for cryptographic operations, SCA countermeasures, anti-tampering means and resilience.

Dan Klann received his Diploma degree in computer science from the Humboldt University at Berlin in 2008 and his Dr.-Ing. degree in computer science from the Brandenburg University of Technology Cottbus–Senftenberg in 2016. Since 2008, he is with the Sensor Networks and Mobile Middleware Group, IHP, Frankfurt (Oder). His research interests include the development of hardware implementations for UWB-systems and for Elliptic Curve Cryptography.

Marcin Aftowicz got his Bachelor's degree in Automation Technology at the Poznan University of Technology and obtained his Master's degree in Technical Informatics at the Brandenburg University of Technology Cottbus-Senftenberg. He is currently employed as a scientist at IHP in Frankfurt Oder. At this stage his research interests are focused on Machine Learning, Cryptography and Hardware Design in which he pursues a PhD degree.

Peter Langendörfer holds a diploma and a doctorate degree in computer science. Since 2000 he is with the IHP in Frankfurt (Oder). There, he is leading the wireless systems department. From 2012 till 2020 he held the chair for security in pervasive systems at the Technical University of Cottbus-Senftenberg. Since 2020 he has held the wireless systems chair at the Technical University of Cottbus-Senftenberg. He has published more than 145 refereed technical articles, filed 17 patents of which 10 have been granted already. He worked as guest editor for many renowned journals, e.g., Wireless Communications and Mobile Computing (Wiley) and ACM Transactions on Internet Technology. Peter is highly interested in security for resource constraint devices, low power protocols and resilience.