Integration, the VLSI Journal 73 (2020) 50–67

Contents lists available at ScienceDirect

Integration, the VLSI Journal

journal homepage: www.elsevier.com/locate/vlsi

Methods increasing inherent resistance of ECC designs against horizontal attacks

Ievgen Kabin [a,*], Zoya Dyka [a], Dan Klann [a], Peter Langendoerfer [a,b]

[a] IHP – Leibniz-Institut für Innovative Mikroelektronik, Frankfurt (Oder), Germany
[b] BTU Cottbus-Senftenberg, Cottbus, Germany

ARTICLE INFO

Keywords:
Field multiplication
Multiplication methods
Regular schedule
Power traces
Address bit differential power analysis (DPA) attacks
Countermeasure against side channel analysis (SCA) attacks

ABSTRACT

Due to the nature of applications such as critical infrastructure and the Internet of Things etc. side channel analysis attacks are becoming a serious threat. Side channel analysis attacks take advantage from the fact that the behaviour of crypto implementations can be observed and provides hints that simplify revealing keys. A new type of SCA is the so called horizontal differential SCA. In this paper we investigate two different approaches to increase the inherent resistance of our hardware accelerator for the *kP* operation. The first approach aims at reducing the impact of the addressing in our design by realizing a regular schedule of the addressing. In the second approach, we investigated how the formula used to implement the multiplication of $GF(2^n)$-elements influences the results of horizontal DPA attacks against a Montgomery *kP*-implementation. We implemented 5 designs with different partial multipliers, i.e. based on different multiplication formulae. We used two different technologies, i.e. a 130 and a 250 nm technology, to simulate power traces for our analysis. We show that the implemented multiplication formula influences the success of horizontal attacks significantly. The combination of these two approaches leads to the most resistant design. For the 250 nm technology only 2 key candidates could be revealed with a correctness of about 70% which is a huge improvement given the fact that for the original design 7 key candidates achieved a correctness of more than 90%. For our 130 nm technology no key candidate was revealed with a correctness of more than 60%.

1. Introduction

1.1. Motivation

The number human beings whose lives are depending on the dependability of embedded devices is currently growing significantly. In the past dependability of embedded systems was merely an issue in industrial control systems but with the advent of ever-increasing complex telemedicine systems and autonomous driving larger parts of our societies –not say we all – are exposed to threats appearing from malfunctioning systems. On the one hand faults due to ageing and stress of devices may cause dangerous situations on the other hand malicious manipulation may have the same effect. Successful attacks against systems in the field of telemedicine i.e. pacemaker [1,2] and against automotive systems i.e. cars [3] have already been reported.

In order to prevent successful attacks from the very beginning, security issues need to be considered at design time. To illustrate this we focus on automotive use cases. In large cities i.e. those with high number of lanes and high density of cars the number of messages a single car is receiving may go up to 4000 messages per second [4]. In case of large platoons travelling along the highway, a similar number of messages per second may be received by all the cars. As the next actions of the receiving cars e.g. bracing, accelerating or similar depend on those messages it is of utmost importance that the content of those messages was not altered during transport and that these messages origin from authenticated devices e.g. other cars or road side units. This can be ensured by digitally signing the messages before sending and verifying the digital signatures after receiving the messages. But verifying digital signatures is a computationally expensive task that is beyond the processing power of embedded devices at least when it comes to 4000 signatures to be verified per second. So appropriate hardware accelerators for this task are needed. Roadside Units (RSUs) are increasing the complexity of the design task. RSU are deployed along the roads which means that at least some are easily accessible and unprotected so that a

* Corresponding author.
E-mail addresses: kabin@ihp-microelectronics.com (I. Kabin), dyka@ihp-microelectronics.com (Z. Dyka), klann@ihp-microelectronics.com (D. Klann), langendoerfer@ihp-microelectronics.com (P. Langendoerfer).

https://doi.org/10.1016/j.vlsi.2020.03.001
Received 29 November 2019; Received in revised form 19 February 2020; Accepted 2 March 2020
Available online 20 March 2020

potential attacker can use side channel analysis (SCA) attacks to extract the keys and afterwards impersonate an RSU. Due to the related safety issues, this needs to be prevented by design, i.e. the hardware accelerators need not only to be fast but in addition need to be SCA resistant which needs to be taken into account when designing the hardware accelerators.

1.2. Background: SCA attacks

Cryptographic approaches can guarantee confidentiality, data integrity and authentication of communication partners. For scenarios such as car-to-X communication the latter two are key and asymmetric cryptographic approaches e.g. RSA or Elliptic Curve Cryptography (ECC) are normally used. ECC provides the same level of security as RSA based approaches but with significantly shorter cryptographic keys which is one of the reasons why P-256 was standardized for vehicular communication. The main operation for ECC is the Elliptic Curve point multiplication with a scalar, denoted usually as *kP* operation. *P* is a point of the selected EC and has two affine coordinates: $P = (x, y)$. The scalar *k* is a big binary number; it is a private key if the decryption corresponding to the ElGamal approach is performed. For signature generation the scalar *k* is a random number. For signature generation corresponding to Elliptic Curve Digital Signature Standard (ECDSA) [5] each RSU has to use its ECC private key. If an attacker can reveal the random scalar *k* used for a signature generation, he can calculate the private key of the signer. This is the ultimate goal of an attacker. Side channel analysis (SCA) attacks are a suitable means to achieve this goal and can be applied to any implementation soft- or hardware of a cryptographic algorithm.

Due to the fact that the processed scalar *k* either is the private key or the knowledge of the scalar *k* allows to calculate the private key we further denote the scalar *k* also as *key*. SCA attacks exploit the fact that physical parameters such as time, power consumption and electromagnetic radiation of cryptographic devices can be measured directly or indirectly while performing cryptographic operations. The shape of the obtained power traces (PT) or electromagnetic traces (EMT) depends not only on the implemented circuit (technology gates library, 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.

If a single measured trace is not sufficient to extract the key successfully by visual inspection, an attacker can collect many traces. For example, if an elliptic curve cryptography (ECC) design is attacked, the attacker can select specific input data and/or run the device with a selected "key" i.e. a scalar that is processed with the input data in the same manner as the secret key. The collected traces can be analysed using statistical methods. In order to protect cryptographic implementations against SCA attacks, designers try to withhold information from the attacker by blinding the input data, randomization of the key [6] or by randomizing the algorithm steps [7]. Thus, the attacker will no longer reach his/her goal by just altering the input data, since not what the attacker selected is processed but data altered by the implementation. So, the attacker can no longer analyse the influence of the data it provided on the shape of the power traces.

It is important to note that well-known countermeasures such as EC point blinding, randomization of the scalar or EC point coordinates, are only effective against attacks that need more-than-one trace for the analysis. In Ref. [8] such attacks are classified as vertical attacks. Examples of such attacks are:

- collision-based attacks [9–11],
- classical differential power analysis [12],
- correlation power analysis attacks [13].

The above-mentioned countermeasures are not effective against horizontal attacks, i.e. against attacks analysing only one measured trace using statistical methods, for example using the comparison to the mean as described in Ref. [14].[1]

Blinding of the elliptic curve (EC) point *P* or randomization of the projective coordinates of point *P* do not provide protection against horizontal DPA attacks [15]. The randomization of the private key *k* does not hinder horizontal DPA attacks because the revealed randomized key can successfully be used instead of the real private key. This means that implementations protected against vertical DPA by such randomization are not implicitly protected against horizontal DPA.

The *kP* operation implemented in hardware is a bitwise processing of the scalar *k*. Even if each bit of the key is processed using exactly the same sequence of operations, the key-dependent processing of data or addressing of blocks can result in a successfully revealed key if horizontal DPA attacks are applied [16]. Other examples of horizontal attacks are:

- simple power analysis attacks e.g. Ref. [17],
- simple electromagnetic analysis attacks e.g. Ref. [18],
- the Big Mac attack [19],
- the localized electromagnetic analysis attack [20],
- horizontal collision correlation analysis attacks [8,21,22],
- horizontal differential electromagnetic analysis (DEMA) attacks [23, 24].

In this paper we focus on the protection of our ECC hardware accelerator against horizontal attacks as vertical DPA attacks need more measurements than horizontal attacks to be successful, i.e. they are more complex and time-consuming, especially if randomization countermeasures [6] are implemented.[2] In comparison to vertical DPA attacks, horizontal DPA attacks are relatively simple but nevertheless they require by far more complex kinds of countermeasure to ensure that the randomization or regularity are effective.

1.3. Contributions

In this paper, we concentrate on investigations, of how the inherent resistance of hardware accelerators for the *kP* operation can be increased applying simple design methods using our own design as an example. This research is an extension of our previous work partially described in Ref. [28,29]. The contributions reported here are:

- determining significant leakage sources i.e. identification of the addressing of different blocks as the main leakage source
- introduction and verification of a regular schedule of the addressing of the blocks as a suitable remedy against horizontal address bit attacks
- Investigation of the influence of the implemented multiplication method on the success of the performed horizontal DPA attack
- Investigation of the impact of the combination of the means mentioned above on the resistance of the *kP* design i.e. we implemented the *kP* design with the "most resistant" multiplication method and with a regular schedule plan for the block addressing and attacked its power trace.

All attacks discussed in this paper were performed using simulated power traces of the different designs.

The rest of this paper is structured as follows. In section II we explain the implementation details of our basic design including a thorough analysis of its vulnerabilities and our horizontal attack. In the following section we discuss how the vulnerability related to the addressing in our design can be reduced by ensuring a regular addressing. Section IV discusses the impact of different multiplication formula on the success rate of attacks against our design. In section V we discuss the impact of the

[1] We denoted this method in Ref. [14] as "the difference of the means".

[2] Meanwhile more sophisticated attacks e.g. those described in Ref. [25–27] allow to extract secret keys even in case such countermeasures are applied.

combination of the regular addressing and the most resistant partial multiplier. The paper finishes with short conclusions.

2. Our basic *kP* accelerator and its known weaknesses

In this section we lay the basis for the understanding of the investigations discussed in the later sections of this paper. We start by introducing the implementation details of our basic *kP* design. Next, we explain how we perform the horizontal DPA attack, which is followed by discussing the SCA leakage sources in our implementation.

2.1. Structure of our basic kP design

Our basic *kP* design is a hardware accelerator for operations using elliptic curve B-233. B-233 is an EC over binary extended Galois Field $GF(2^n)$ standardized by NIST [5]. The *kP* accelerator obtains a scalar k and two affine coordinates x and y of a point P of the EC B-233 as inputs to process. The numbers x, y and k are up to 233 bit long binary numbers that represent elements of $GF(2^{233})$, with the irreducible polynomial $f(t) = t^{233} + t^{74} + 1$. The hardware accelerator processes the scalar k bitwise according to the Montgomery *kP* algorithm using Lopez-Dahab projective coordinates [30]. Algorithm 1 shows the well-known and mostly referenced Montgomery *kP* algorithm [30].

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

Input: $\mathbf{k} = (k_{l-1} \ldots k_1\, k_0)_2$ with $k_{l-1} = 1$, $\mathbf{P} = (x,y)$ is a point of EC over GF(2^l)
Output: $\mathbf{kP} = (x_1, y_1)$
1: $X_1 \leftarrow x$, $Z_1 \leftarrow 1$, $X_2 \leftarrow x^4 + b$, $Z_2 \leftarrow x^2$
2: **for** i = l-2 **down to** 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**
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: $x_1 \leftarrow X_1/Z_1$
12: $y_1 \leftarrow y + (x + x_1)[(X_1 + xZ_1)(X_2 + xZ_2) + (x^2 + y)(Z_1Z_2)]/(xZ_1Z_2)$
13: **return** (x_1, y_1)

Algorithm 2 shows the modified Montgomery *kP* algorithm published in Ref. [14] that we implemented and investigated here. This allows to perform all operations in parallel to the field multiplications, we will later discuss the resulting benefits.

Algorithm 2
Modified Montgomery algorithm for the *kP* operation

Input:
$\mathbf{k} = (k_{l-1} \ldots k_1\, k_0)_2$ with $k_{l-1} = 1$, $P=(x,y)$ *is a point of EC over* $GF(2^l)$
Output: $\mathbf{kP} = (x_1, y_1)$
//initialization
1: $X_1 \leftarrow x$, $X_2 \leftarrow x^4 + b$, $Z_2 \leftarrow x^2$.
//processing second most significant bit
2: **if** $k_{l-2} = 1$ **then**
3: $T \leftarrow Z_2$, $Z_1 \leftarrow (X_1Z_2 + X_2)^2$, $X_1 \leftarrow X_1Z_2X_2 + xZ_1$
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**
6: $T \leftarrow Z_2$, $Z_2 \leftarrow (X_1Z_2 + X_2)^2$, $X_2 \leftarrow X_1X_2T + xZ_2$
7: $T \leftarrow X_1$, $U \leftarrow bX_2^4$, $X_1 \leftarrow X_1^4 + b$, $U \leftarrow TX_2$, $Z_1 \leftarrow T^2$.
8: **end if**
//main loop
9: **for** i **from** $l - 3$ **downto** 0 **do**
10: **if** $k_i = 1$ **then**
11: $T \leftarrow X_1$, $Z_1 \leftarrow (X_1Z_2 + X_2Z_1)^2$, $X_1 \leftarrow xZ_1 + X_1X_2TZ_2$
12: $T \leftarrow X_2$, $X_2 \leftarrow X_2^4 + bZ_2^4$, $Z_2 \leftarrow T^2Z_2^2$.
13: **else**

(*continued on next column*)

Algorithm 2 (*continued*)

14: $T \leftarrow Z_2$, $Z_2 \leftarrow (X_2Z_1 + X_1Z_2)^2$, $X_2 \leftarrow xZ_2 + X_1X_2TZ_1$
15: $T \leftarrow X_1$, $X_1 \leftarrow X_1^4 + b\,Z_1^4$, $Z_1 \leftarrow T^2\,Z_1^2$.
16: **end if**
17: **end for**
//end of the main loop
18: $x_1 \leftarrow X_1/Z_1$
19: $y_1 \leftarrow y + (x + x_1)[(X_1 + xZ_1)(X_2 + xZ_2) + (x^2 + y)(Z_1Z_2)]/(xZ_1Z_2)$
20: **return** (x_1, y_1)

In our implementation the main loop of Algorithm 2 (see lines 9–17) consists of multiplications, squarings and additions of $GF(2^{233})$-elements. The division of $GF(2^{233})$-elements is performed only once at the end of the algorithm (see line 19 in Algorithm 2). It is calculated as a sequence of field squarings and field multiplications using the little theorem of Fermat: $x(t)(t)^{-1} \bmod f(t) = x(t)^{2^n-2} \bmod f(t)$.

Fig. 1 shows the structure of our basic *kP*-accelerator.

The block ***MULT*** calculates the field product. The block ***ALU*** performs addition (bitwise XOR) or squaring of its operands depending on the signals of the block ***Controller***. Additionally, the design contains ten 233 bit long registers for saving input and output as well as intermediate values:

- the registers x and y contain the affine coordinates of the input point P as well as the result of the *kP* operation;
- register k contains the input scalar k;
- register b contains the parameter b of EC *B-233;*
- registers X_1, X_2, Z_1, Z_2, X_3 and X_4 are necessary for the implementation of the main loop of the algorithm.

2.2. BUS and Controller: implementation details

Our basic design contains 12 components: 10 registers, ***ALU*** and ***MULT***. These components have different addresses. Table 1 contains the address of each component of our *kP* design, as hexadecimal numbers.

The ***Controller*** is a state machine that contains also a 32-bit register. Bits 27–24 of this register are reserved for the addressing of the 12 components for the *write-to-bus* operation, see addresses in Table 1. 13 bits of the register are reserved for the addressing of the components for the *read-from-bus* operation. Other bits of the ***Controller's*** register are flags that determine the kind of the currently performed operation and manage the execution of the operation clock-by-clock. Thus, using the addresses of the 12 components the block ***Controller*** manages the data flow between the components. Additionally, the block ***Controller*** manages the sequence of field operations. Depending on the signals of the ***Controller*** (i.e. depending on the value of certain bits of the ***Controller***'s register) the block ***ALU*** performs addition or squaring. The ***ALU*** takes two clock cycles for a squaring (one clock cycle for obtaining of the operand and second one for the calculation in fact) and 4 clock cycles for an addition of two elements: the 1st operand will be read from the ***BUS*** and written to the internal ***ALU*** register then the second operand will be read from the ***BUS*** and added to the internal register. The multiplier takes two clock cycles for obtaining both multiplicands from the ***BUS*** and

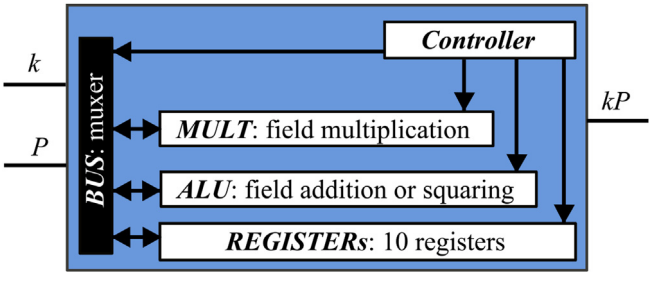


Fig. 1. Structure of our basic *kP*-accelerator.

Table 1
Addresses of components of our design.

block	*MULT*	*ALU*	*REGISTERs*									
			x	y	k	b	X_1	X_2	X_3	X_4	Z_1	Z_2
address	6	7	8	9	0	1	4	5	A	B	2	3

only 9 clock cycles to calculate a field product corresponding to the 4-segment Karatsuba MM. The multiplier has two internal registers for its inputs and one register for its output.

The ***BUS*** is realized as a muxer. It consists of many logic gates that react on the address given by the ***Controller***. 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.

Fig. 2 shows the processing sequence in the main loop of our implementation providing details about the activities of each block.

The rectangles in Fig. 2 represent different activities in our ECC design. Rectangles that are vertically aligned are processed in parallel i.e. in the same clock cycle. All write to register operations are depicted by small green squares for the addressing of the register in one clock cycle and a small grey square for the storing operation in the next clock cycle. Red rectangles show the activity of the field multiplier. In our implementation the multiplier is always active, i.e. obtaining both multiplicands is performed in parallel to the calculation of the last two of the 9 partial products. The activity of the block ***ALU*** is shown in Fig. 2 using yellow rectangles for the squaring operation and blue rectangles for the field addition.

The numbers in the rectangles in Fig. 2 represent the addresses of the components of our design for the write-to-bus operation i.e. these numbers show which component writes its output value to the ***BUS*** in the corresponding clock cycle of the main loop beginning from clock cycle 1 up to clock cycle 54. The sequences of these numbers differ for $k_i = 0$ and $k_i = 1$ which is inherent to the Montgomery *kP* algorithm. The key-dependent addressing of the components is shown by the turquoise marked numbered rectangles in Fig. 2. In most cases, i.e. in 42 out of 54 clock cycles, the addressing of the blocks does not depend on k_i. For example, in the 1st clock cycle the multiplier writes its result to the ***BUS*** (its address is 6) independent of the key bit value.

The value at the ***BUS*** will be read by one of the registers/blocks. The addressing of the blocks ***ALU*** and ***MULT*** for the read-from-bus operation does not depend on the key. The addressing of registers depends on the key except of the second write-to-register operation (see clock cycles 5 and 6 in Fig. 2, the operation is marked with a circle) Thus, the small green squares for the addressing of the registers/blocks show which of the blocks – ***ALU***, ***MULT*** or one of the registers – reads from the ***BUS*** in the corresponding clock cycle. For example in clock cycle 3, depending on the currently processed key bit value, either the register Z_2 (see turquoise rectangle with the number 3) or the register Z_1 (see turquoise rectangle with the number 2) writes its output to the ***BUS***.

At the same time the block ***ALU*** reads the value from the ***BUS*** (see small green square in the ***ALU***), squares it (***ALU*** is marked yellow when performing this operation) and writes the calculated value to its internal register in the next clock cycle (see the small grey square in the ***ALU*** in the clock cycle 4). In the clock cycle 5, the block ***ALU*** writes the result of the squaring operation to the ***BUS*** i.e. the block ***ALU*** is addressed for the write-to-bus operation in clock cycle 5 (see Fig. 2, address 7 in clock cycle 5) independent of the processed key bit value.

In many cases the block ***ALU*** writes its result to the ***BUS*** directly after the calculation, for example, in clock cycles 5, 16, 32, 43 and 50. Sometimes we use ***ALU*** as a register for saving intermediate values. In such cases ***ALU*** is addressed a few clock cycles after being active, see for example clock cycles 7, 12, 30, 35.

Please note that in Fig. 2 not all clock cycles are marked as potentially dangerous with respect to the addressing of the blocks. There are unmarked clock cycles that nevertheless lead to a high correctness of the extracted key i.e. 8 and 9, but we do not cope with those issues in this paper.

There exist clock cycles in our design in which none of the blocks reads from or writes to the bus, see for example the 2nd and the 4th clock cycle. In these clock cycles any or none of the blocks can write to the ***BUS***

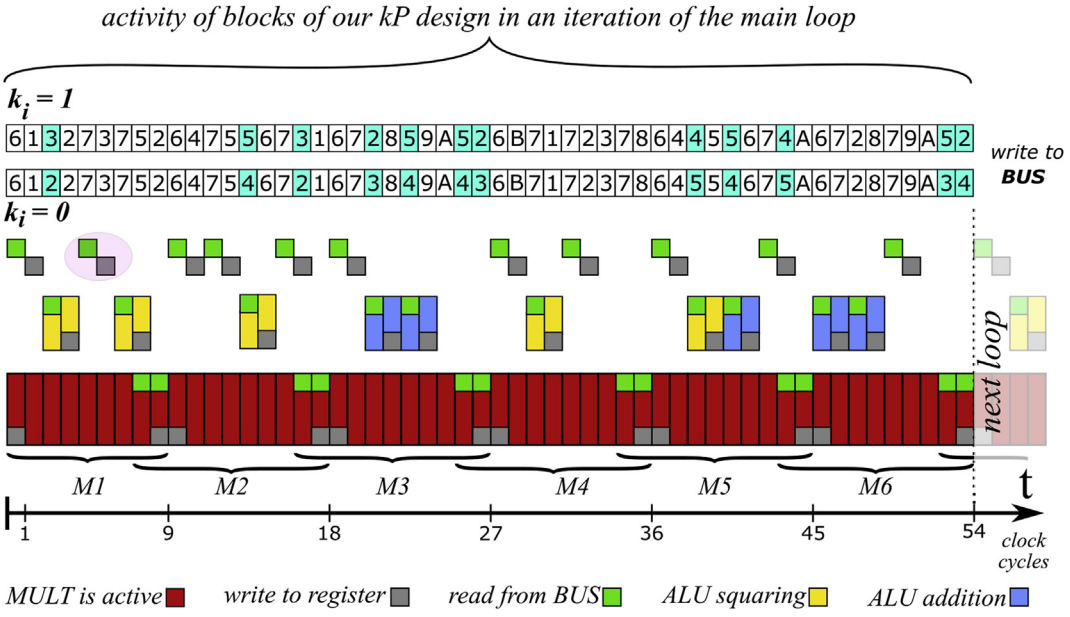


Fig. 2. Processing sequence of the main loop in our *kP* design: details about activity of each block.

without influencing the functionality of the *kP* implementation. In our implementation it is the block with address 0, i.e. the scalar k is always written to the ***BUS***. Even though this solution is a reasonable one from the perspective of reliability it can simplify microprobing based attacks. So, in order to get a tamper-proof implementation, all types of attacks need to be taken into account. We mention this to show how complex the implementation of cryptographic operations is.

Due to the fact that the addressing of registers in the Montgomery *kP* algorithm depends on the processed bit value k_i of the scalar k, horizontal differential SCA attacks can be successful. We discuss this fact in detail in section II-*G*.

2.3. Field multiplier

The multiplication is the most complex field operation in our designs. The polynomial multiplication (i.e. the first step of the multiplication of elements of $GF(2^n)$) can be realized by applying the classical multiplication method. Its gate complexity (GC) can be given as a number of Boolean AND and XOR operations, i.e. as the number of used AND and XOR gates. To implement the multiplication of n-bit long polynomials using the classical multiplication method n^2 AND and $(n-1)^2$ XOR gates are necessary. This results in an expensive implementation with respect to area and energy since the length of multiplicands is typically large (more than 200 bit). In order to tackle this complexity issue many optimizations, i.e. new multiplication formulae, have been proposed in the past. Many multiplication methods apply segmentation of both multiplicands into the same number of parts. The product is then calculated as a sum of smaller partial products. Historically, the first optimization was the Karatsuba multiplication method published in 1962 [31]. This method uses the segmentation of polynomials into two terms. The next multiplication formula was proposed by Winograd in 1980 [32]. This method uses the segmentation of polynomials into three terms. At the moment there exist a lot of multiplication methods exploiting different multiplication formulae or their combinations. Multi-segment-Karatsuba MM (MSK) [33] and enhanced MSK [34] are examples of such combinations. In Ref. [35,36] different multiplication methods were combined with the goal to find the optimal combination, i.e. the combination with minimal gate complexity and energy consumption. Each combination of multiplication methods (MM) has its own gate complexity. Additionally, the number of XOR gates can be significantly reduced by calculating the product iteratively [37]. Combinations of multiplication methods with reduced XOR-complexity are investigated in Ref. [38] for different length of operands and their segmentation.

In our basic *kP* design, the field multiplier is implemented using the 4-segment Karatsuba multiplication method [37] according to a fixed calculation plan. Formula (1) represents the 4-segment Karatsuba Multiplication Method (MM) for two binary polynomials $A(t)$ and $B(t)$.

$$
\begin{aligned}
&A(t)\cdot B(t) = A_3A_2A_1A_0\cdot B_3B_2B_1B_0 = \\
&= \left(A_3\cdot 2^{3m}\oplus A_2\cdot 2^{2m}\oplus A_1\cdot 2^{m}\oplus A_0\right)\cdot\left(B_3\cdot 2^{3m}\oplus B_2\cdot 2^{2m}\oplus B_1\cdot 2^{m}\oplus B_0\right) = \\
&= S_6\cdot 2^{6m}\oplus S_5\cdot 2^{6m}\oplus S_4\cdot 2^{4m}\oplus S_3\cdot 2^{3m}\oplus S_2\cdot 2^{2m}\oplus S_1\cdot 2^{1m}\oplus S_0 = \\
&= A_0B_0\cdot 2^{0}\oplus (A_0B_0\oplus A_1B_1\oplus (A_0\oplus A_1)(B_0\oplus B_1))\cdot 2^{1m} \\
&\oplus (A_0B_0\oplus A_1B_1\oplus A_2B_2\oplus (A_0\oplus A_2)(B_0\oplus B_2))\cdot 2^{2m} \\
&\oplus \begin{pmatrix} A_0B_0\oplus A_1B_1\oplus A_2B_2\oplus A_3B_3\oplus \\ (A_0\oplus A_2)(B_0\oplus B_2)\oplus (A_0\oplus A_1)(B_0\oplus B_1)\oplus \\ (A_1\oplus A_3)(B_1\oplus B_3)\oplus (A_2\oplus A_3)(B_2\oplus B_3)\oplus \\ (A_0\oplus A_1\oplus A_2\oplus A_3)(B_0\oplus B_1\oplus B_2\oplus B_3) \end{pmatrix}\cdot 2^{3m} \\
&\oplus (A_1B_1\oplus A_2B_2\oplus A_3B_3\oplus (A_1\oplus A_3)(B_1\oplus B_3))\cdot 2^{4m} \\
&\oplus (A_2B_2\oplus A_3B_3\oplus (A_2\oplus A_3)(B_2\oplus B_3))\cdot 2^{5m}\oplus A_3B_3\cdot 2^{6m} = \\
&= p_1\cdot 2^{0}\oplus (p_1\oplus p_2\oplus p_7)\cdot 2^{1m}\oplus (p_1\oplus p_2\oplus p_3\oplus p_5)\cdot 2^{2m}\oplus \\
&\oplus (p_1\oplus p_2\oplus p_3\oplus p_4\oplus p_5\oplus p_7\oplus p_6\oplus p_8\oplus p_9)\cdot 2^{3m}\oplus \\
&\oplus (p_2\oplus p_3\oplus p_4\oplus p_6)\cdot 2^{4m}\oplus (p_3\oplus p_4\oplus p_8)\cdot 2^{5m}\oplus p_4\cdot 2^{6m} = \\
&= C_7C_6C_5C_4C_3C_2C_1C_0, \qquad (1)
\end{aligned}
$$

with partial products:

$p_1 = A_0B_0,\quad p_2 = A_1B_1, p_3 = A_2B_2,\quad p_4 = A_3B_3,$
$p_5 = (A_0\oplus A_2)(B_0\oplus B_2),\quad p_6 = (A_1\oplus A_3)(B_1\oplus B_3),$
$p_7 = (A_0\oplus A_1)(B_0\oplus B_1),\quad p_8 = (A_2\oplus A_3)(B_2\oplus B_3),$
$p_9 = (A_0\oplus A_1\oplus A_2\oplus A_3)(B_0\oplus B_1\oplus B_2\oplus B_3)$
each partial product p_i is $2m-1$ bit long,
easch sum S_i is $2m-1$ bit long,
each segment of the product $C_6,\ldots,C_0$ is m bit long,
the segment C_7 is $m-1$ bit long.

For the EC B-233 the maximal length of the multiplicands is 233 bit. Thus, two up to 233 bit long operands $A(t)$ and $B(t)$ are segmented into four parts: A_3, A_2, A_1, A_0 and B_3, B_2, B_1, B_0 respectively. The parts A_3 and B_3 are 56 bit long. All other parts are 59 bit long i.e. we implemented the field multiplier using a Partial Multiplier for 59 bit long operands ($m = 59$). The structure of our field multiplier is shown in Fig. 3.

The field multiplier takes 9 clock cycles (clk^j, $j = 0,1,2, \ldots,8$) to calculate the product of the 233 bit long operands. In comparison to the classical MM the 4-segment Karatsuba MM has a significantly reduced execution time – only 9 clock cycles instead of 16 i.e. the time reduction is about 44%. In our implementation according to the 4-segment Karatsuba MM (1) only one of 9 partial products of the 59 bit long operands is calculated per clock cycle. In our implementation the partial products are calculated in the sequence $p_1, p_2, \ldots p_9$ as shown in (1).

The signals $cntr^j_i$ with $0 \le i \le 3$ and $0 \le j \le 8$ organize the calculation of operands for the Partial Multiplier clock-by-clock. For example; in the first of the nine clock cycles clk^0 of a field multiplication the signals $cntr^j_i$ are: $cntr^0_0 = 1$; $cntr^0_1 = 0$; $cntr^0_2 = 0$; $cntr^0_3 = 0$. Using these values the operands for the Partial Multiplier are calculated as follows:

$$
\begin{aligned}
A^0 = \bigoplus_{i=0}^{3}\left(A_i\cdot cntr_i^0\right) = A_0\cdot cntr_0^0\oplus A_1\cdot cntr_1^0\oplus A_2\cdot cntr_2^0\oplus A_3\cdot cntr_3^0 = \\
A_0\cdot 1\oplus A_1\cdot 0\oplus A_2\cdot 0\oplus A_3\cdot 0 = A_0
\end{aligned}
\qquad (2)
$$

$$
\begin{aligned}
B^0 = \bigoplus_{i=0}^{3}\left(B_i\cdot cntr_i^0\right) = B_0\cdot cntr_0^0\oplus B_1\cdot cntr_1^0\oplus B_2\cdot cntr_2^0\oplus B_3\cdot cntr_3^0 = \\
B_0\cdot 1\oplus B_1\cdot 0\oplus B_2\cdot 0\oplus B_3\cdot 0 = A_0
\end{aligned}
\qquad (3)
$$

Thus, in clk^j with $j = 0$ the Partial Multiplier calculates the partial product $A^{j=0}\cdot B^{j=0} = A_0\cdot B_0$. All partial products $A^j\cdot B^j$ are accumulated in the output register of the field multiplier iteratively, step-by-step (or clock-by-clock), corresponding to (1). The reduction is also performed in each clock cycle. The reduction can be performed only once, i.e. after all 9 partial multiplications, but the reduction consumes additional energy i.e. it can be used to increase the energy consumption of the field multiplier and – as the consequence – hides partially the activity of other blocks of the *kP* design. Thus, performing the reduction in each clock cycle can increase the resistance of the *kP* design against SCA attacks, but we did not yet evaluate this effect.

As already mentioned, all blocks of our design may run in parallel. So, the processing time of a key bit in the main loop iteration in our implementation is equal to the time needed to execute 6 field multiplications only. I.e. the multiplier is always active and is a strong noise source due to its big area and high energy consumption. The resulting effect is beneficial as the multiplier is resistant to the performed horizontal DPA attack which we show later. Thus, the activity of the multiplier increases the effort needed in the attacks i.e. the multiplier can be considered as a kind of an inherently implemented countermeasure against attacks.

2.4. Partial multiplier in our basic design: the implemented combination of multiplication formulae

This Partial Multiplier of 59 bit long operands A^j and B^j was implemented using the 2-segment iterative Karatsuba multiplication formula [38] for 60-bit long multiplicands. The gate complexity of this multiplier is $GC_{2m} = 3\cdot GC_m + (7m-3)_{XOR}$. Here m is the length of segments $m = 60/2$

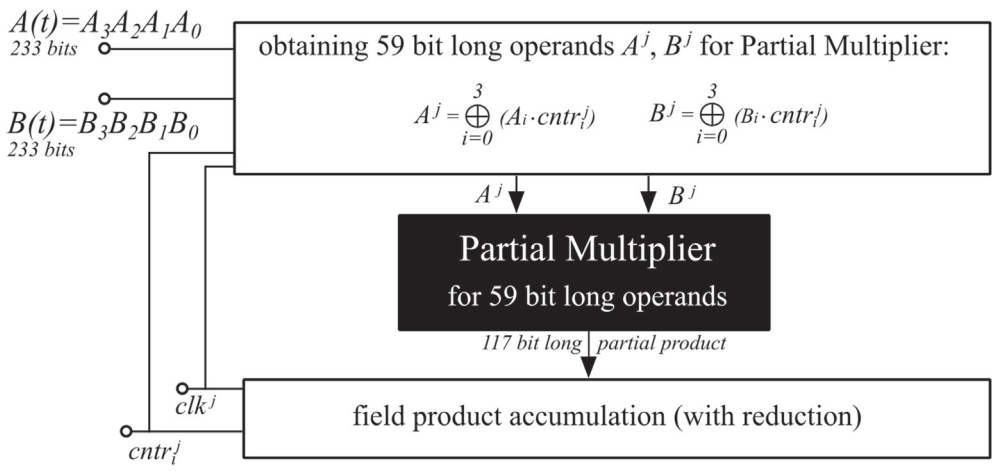


Fig. 3. Structure of the field multiplier.

$= 30$ and GC_m is the gate complexity of the internal *m-bit* partial multipliers. The Partial Multiplier contains 3 internal 30-bit partial multipliers. All these internal partial multipliers were implemented identically, using the 6-segment iterative Winograd multiplication formula [38], with a gate complexity of: $GC_{6m} = 18 \cdot GC_m + (72m\text{-}19)_{XOR}$, with $m = 30/6 = 5$ bits. Corresponding to the 6-segment iterative Winograd multiplication formula the 30-bit multiplier consists of 18 internal multipliers of 5-bit long operands. Each of these small multipliers was implemented using the classical multiplication formula (3) with $n = 5$.

$$C = A \cdot B = \left(a_{n-1}t^{n-1} + \ldots + a_0 t^0\right) \cdot \left(b_{n-1}t^{n-1} + \ldots + b_0 t^0\right) = \sum_{i=0}^{2n-2} ci \cdot ti, \text{with}$$
$$ci = \bigoplus_{i=k+l} a_k \cdot b_l, \forall k, l < n \tag{4}$$

Fig. 4 shows the structure of the Partial Multiplier implemented in our basic ECC design and the gate complexity for each kind of its internal multipliers.

Corresponding to Ref. [38] the combination of the multiplication methods for the implementation of a 59-bit Partial Multiplier – the 2-segment Karatsuba, the 6-segment Winograd and the classical multiplication formulae – results in one of smallest areas not only for the IHP 130 nm but also for the IHP 250 nm technology. Due to this fact, this combination was implemented in our basic ECC design.

2.5. Simulation of PTs of a kP execution

In order to implement our basic ECC design in hardware we described the *kP* algorithm using Very high-speed integrated circuits Hardware Description Language (VHDL). We synthesized our basic design using the IHP 130 nm and the IHP 250 nm technology libraries. The next steps in the design flow are the placement and routing processes. Power Traces can be simulated after the synthesis of the design as well as after the placement and routing processes. Simulated PTs are a powerful means to detect side channel leakages early. Algorithmic issues such as number of operations depending on the key bit value will become obvious. Also "quick-fixes" of this issue i.e. the use of dummy operations can be evaluated as parameters of the gates are taken into account during simulation. For a first evaluation it is reasonable to perform an attack using PTs simulated before place and route as the latter can take many hours.

We simulated the PT of the *kP* execution using the Synopsys PrimeTime suite [39] for a randomly generated EC point $P = (x, y)$ and for a randomly generated 232 bit long private key k. The processed values are in hexadecimal:

$x = 181856ADC1E7DF1378491FA736F2D02E8ACF$
$1B9425EB2B061FF0E9E8246$

$y = 9FED47B796480499CBAA86D8EB39457C49D$
$5BF345A0757E46E2582DE6$

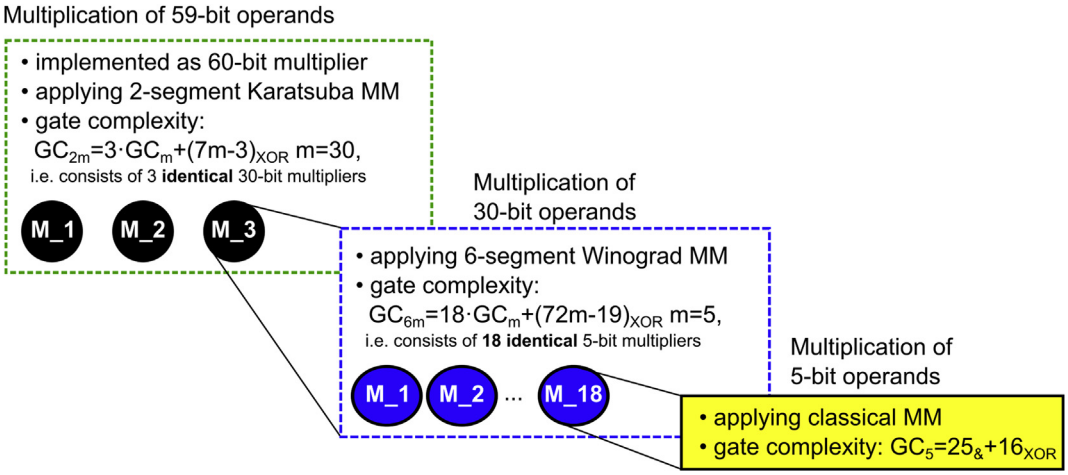


Fig. 4. Structure of the Partial Multiplier in our basic ECC design.

$k = 93919255FD4359F4C2B67DEA456EF70A545A9C$
$44D46F7F409F96CB52CC$

The PrimeTime PX tool uses the gate-level simulation activity stored in the activity file to estimate the averaged power consumption of the design. It can also precisely analyse the power consumption of the design for each clock cycle within a given time interval. Because the simulated traces are noiseless and no data are lost in simulations, we set the time interval for the simulation equal to the clock cycle period i.e. we represented each clock cycle using only one power value – the average power value of the clock cycle. This is reasonable, as it simplifies the statistical analysis of the trace without any loss of information relevant for the analysis.

To apply statistical analysis to simulated traces we use our own program. It extracts the information for each single block/component and stores it as a separate file i.e. we can analyse not only the PT of the whole *kP* design but also the PTs of single blocks.

Fig. 5 shows a part of the PTs of the whole *kP* design for processing of the following 6 bits of the scalar *k*: $k_{230}k_{229} \ldots k_{221}$ = '001001', simulated for both IHP gate libraries and the processed key bit values. The graph shows power in W over time that is shown in clock cycles. The solid line is the PT simulated for the 130 nm technology; the dotted line represents the PT simulated for the 250 nm technology.

Corresponding to the implemented Algorithm 2 the key bit $k_{l-2} = k_{230} = 0$ is processed before the main loop of the algorithm starts. The key bits k_i, with $229 \geq i \geq 0$ are processed in the main loop of the algorithm. We denote the time for processing a key bit in the main loop further as *slot*. The duration of each slot is 54 clock cycles. Due to the fact that in our simulation each clock cycle is represented using only one power value, each slot consists of 54 power values.

2.6. Performed horizontal attack

The goal of our attack is to reveal the key bit values processed in the main loop of the *kP* algorithm. There are only two key bits not processed in the main loop i.e. the most significant key bit $k_{l-1} = 1$ (see Algorithm 1 and Algorithm 2) and the key bit k_{l-2}. The attacker doesn't know the value k_{l-2} but he can easily reveal it via brute force, after successfully revealing of the key bits processed in the main loop. Thus, we concentrated on the analysis of the part of the PT corresponding to the main loop iterations. We fragmented the simulated PT into the slots and selected 230 time slots of the processing of key bits k_i with $229 \geq i \geq 0$ for the statistical analysis. We decided to apply the *comparison to the mean* for statistical analysis of the traces as this method is very effective [29].

Each value in the part of the PT selected for the analysis can be represented as the value $v_i{}^j$, where i is the number of the slot and j is the number of the clock cycle within the slot, with $0 \leq i \leq l\text{-}3$, $1 \leq j \leq 54$.

Each slot i is then a sequence of 54 values $v_i^1, v_i^2, v_i^3, \ldots, v_i^{54}$. We calculated the *mean* slot, i.e. the slot that is the sequence of values $\overline{v^1}, \overline{v^2}, \ldots, \overline{v^{54}}$ with:

$$\overline{v^1} = \frac{1}{l-2}\sum_{i=0}^{l-3} v_i^1 = \frac{1}{230}\sum_{i=0}^{229} v_i^1,$$

$$\overline{v^{54}} = \frac{1}{230}\sum_{i=0}^{229} v_i^{54}.$$

I.e. each value $\overline{v^j}$ in the mean slot is the arithmetical mean of all values with the same number j and different number i.

As a result of this attack we obtained $j = 54$ key candidates:

$$k^1_{candidate} = d^1_{229}d^1_{228}\ldots d^1_2 d^1_1 d^1_0$$
$$\ldots$$
$$k^{54}_{candidate} = d^{54}_{229}d^{54}_{228}\ldots d^{54}_2 d^{54}_1 d^{54}_0$$

For each key candidate $k_{candidate}{}^j$ we obtained its ith bit comparing $\overline{v^j}$ (the jth value of the mean slot) with $v_i{}^j$ (the jth value of the ith slot) as follows:

$$d_i^j = \begin{cases} 1, & \text{if } \overline{v^j} \geq v_i^j \\ 0, & \text{if } \overline{v^j} < v_i^j \end{cases} \quad (5)$$

To evaluate the success of the attack we calculated for each key candidate its *relative correctness* δ_1. This is the number of the correctly revealed bits in the key candidate divided by all revealed bits i.e.:

$$\delta_1^j = \frac{NumberOfCorrectRevealedBits\left(k^j_{candidate}\right)}{l-2} \cdot 100\% \quad (6)$$

The relative correctness δ_1 is a value between 0% and 100%. If a key candidate has the correctness $\delta_1 = 100\%$ it means that the key candidate is equal to the real key k. A correctness close to 0% means that all bits of the key candidates revealed are wrong. This means also that our assumption (5) is wrong and the opposite assumption needs to be correct. In this case we can easily obtain the key performing a bitwise inversion of the key candidates. Taking this fact into account we can calculate the correctness as a value between 50 and 100% as follows:

$$\delta^j = 50\% + \left|50\% - \delta_1^j\right| \quad (7)$$

Please note that the worst-case of the attack result from the attacker's point of view is a correctness of all key candidates of 50%. This means that the *comparison to the mean* method cannot even provide a slight hint whether the key bit processed is more likely a '1' or a '0', i.e. this means that the attack was not successful at all. The worst-case from the attacker's point of view is the ideal case from the designer's point of view. We denote it as the "ideal case" in the rest of the paper. Fig. 6 shows the calculated correctness of the key candidates obtained by analysing the traces simulated for our basic ECC design for the IHP 130 nm technology

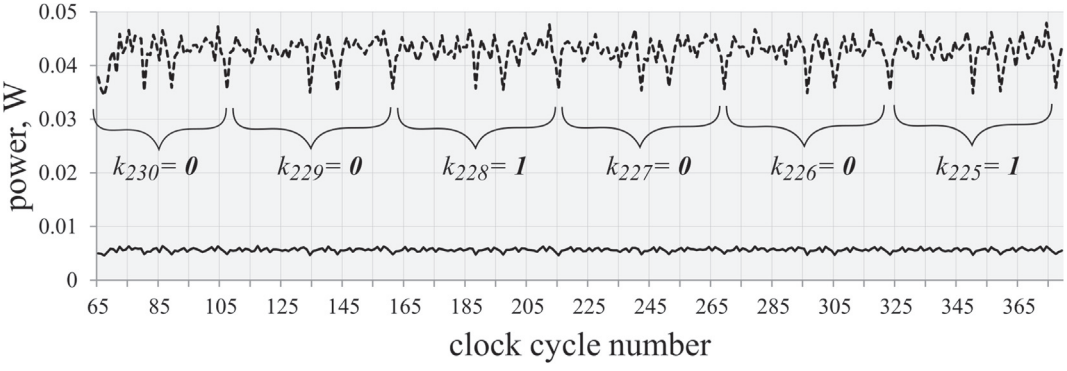


Fig. 5. A part of the power traces simulated for the execution of the *kP* operation.

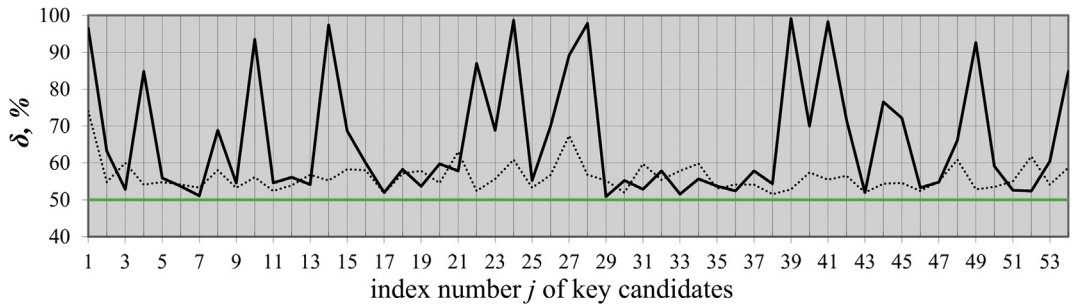


Fig. 6. Horizontal attack using *the comparison to the mean method*: the analysis results are obtained using the power trace of the *kP* design synthesized for the IHP 130 nm (dotted line) and 250 nm (solid line) technologies.

(see solid black line) and for the IHP 250 nm technology (see dotted black line). The green line represents the ideal case.

Fig. 6 shows that the *kP* design, that was synthesized for the 130 nm technology, is more resistant against the performed attack than the design synthesized for the 250 nm technology. Only four key candidates – $k^{1}_{candidate}$, $k^{21}_{candidate}$, $k^{27}_{candidate}$ and $k^{52}_{candidate}$ – were extracted with a relative high correctness of more than 60% i.e. 74%, 63%, 67% and 62% respectively (see doted black line). The correctness of the other 50 key candidates is in the range between 50% and 60%.

Although we implemented the *kP* design strongly balanced and the field multiplier is always active, the key was revealed successfully using a simple statistical analysis method. The analysis of the power trace of the design synthesized for the 250 nm technology reveals that 20 of 54 key candidates have a correctness between 70% and 100% (see black solid line in Fig. 6). In the next subsection, we discuss this vulnerability and show which component in our design is the SCA leakage source.

2.7. Discussion of the vulnerabilities

As shown in Fig. 6 the success of the same attack differs significantly for the same VHDL code if synthesized for different technologies. To understand and explain this fact we analysed the power traces of each component in both *kP* designs i.e. in the basic design for 250 nm and in the basic design for 130 nm. We started our analysis with design synthesized for 130 nm technology.

Using our software we stored the traces of components as separate files and analysed them in the same way as described in section II-F. Fig. 7 shows the results of the analysis for the whole *kP* design (see dotted black line; this is the same dotted black line as in Fig. 6) and for the components: field multiplier ***MULT*** (see green line), registers X_1, X_2, Z_1, Z_2 (see orange line), ***ALU*** (see yellow line), ***Controller*** (see dotted red line) and ***BUS*** (see violet line).

Fig. 7 clearly shows that:

- ***ALU***, registers, ***Controller*** and ***BUS*** are SCA leakage sources;
- The activity of the block ***MULT*** doesn't cause the high success rate of the attack. The correctness of all key candidates is between 50 and 60% i.e. only the block ***MULT*** can be considered as resistant against the performed horizontal attack.

Fig. 8 shows the results attacking the trace *rest* that we obtained via subtraction of the ***MULT*** trace from the trace of the whole *kP* design.

The results of the attack against the difference trace *kP*-***MULT*** show that the rest of the *kP* design is not resistant against the attack. The correctness of the best key candidate $k^{1}_{candidate}$ is 99.6% i.e. 229 of 230 key bits were revealed successful. 5 further key candidates have a correctness higher than 90%.

MULT is the most power and area consuming component in our *kP* design. Table 2 shows the total power and area of selected components in our basic designs for the IHP 130 nm technology and their contributions to the power and area of the whole *kP* design.

The activity of the big and resistant ***MULT*** hides the activity of other components that are strong SCA leakage sources.

We analysed the 250 nm design in the same way as the 130 nm design with the goal to determine which component in the 250 nm design causes the high success of the performed attack and why the resistance of both designs differs significantly. Fig. 9 shows the results of the analysis for the whole *kP* design (see solid black line; this is the same solid black line as in Fig. 6) and for the components: field multiplier ***MULT*** (see green line), registers X_1, X_2, Z_1, Z_2 (see orange line), ***ALU*** (see yellow line), ***Controller*** (see dotted red line) and ***BUS*** (see violet line).

Fig. 9 shows that:

- ***ALU***, registers, ***Controller*** and ***BUS*** are strong SCA leakage sources;
- The activity of the block ***MULT*** doesn't cause a success rate of the attack as high as the other components but two key candidates have a correctness between 70% and 80%. They are the candidates

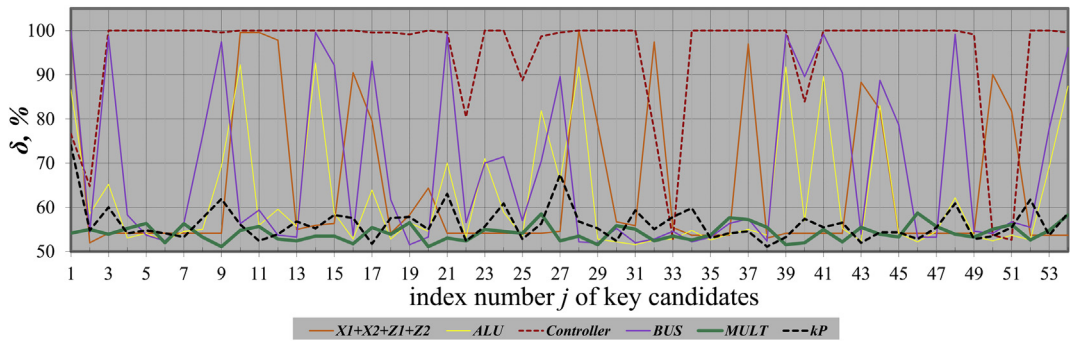


Fig. 7. Horizontal attack using *the comparison to the mean method*: results of the analysis for the whole 130 nm *kP* design and its components.

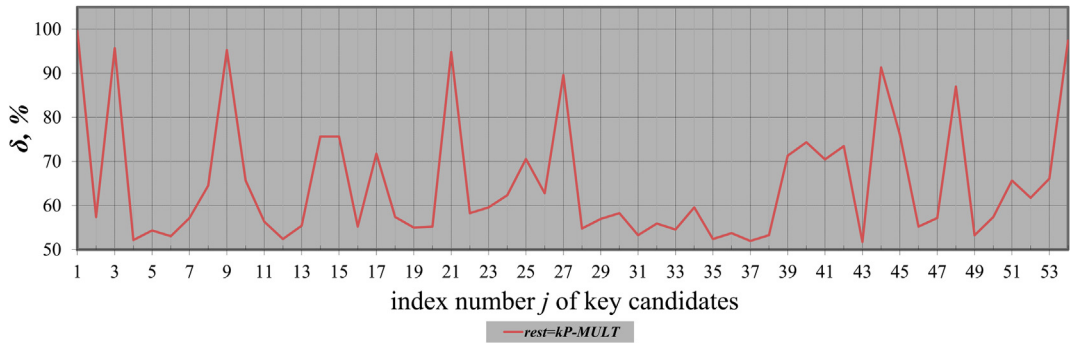


Fig. 8. Horizontal attack analysing the *rest*-trace, i.e. the difference of the traces: *rest* = *kP*-***MULT***.

Table 2
Power of selected components of the 130 nm *kP* design.

130 nm		*kP*	*MULT*	*ALU*	4 registers	*Controller*	*BUS*
power	*total, mW*	5.40	3.13	0.36	0.62	0.1	0.33
	relative to kP design, %	100	57.9	6.6	11.6	1.9	6.1
area	*total, mm²*	0.28	0.11	0.02	0.04	0.01	0.02
	relative to kP design, %	100	39.2	7.2	15.2	3.6	6.5

Table 3
Power of selected components of the 250 nm *kP* design.

250 nm		*kP*	*MULT*	*ALU*	4 registers	*Controller*	*BUS*
power	*total, mW*	40.9	22.40	2.51	5.00	0.83	3.6
	relative to kP design, %	100	54.7	6.1	12.2	2	8.8
area	*total, mm²*	1.38	0.50	0.09	0.23	0.05	0.12
	relative to kP design, %	100	36.1	6.4	16.4	3.4	8.5

$k^{27}_{candidate}$ and $k^{45}_{candidate}$ i.e. the ***MULT*** in the 250 nm technology design is not really resistant against the performed horizontal attack;
– The attack results using the PT of the whole *kP* design and those using the PT of ***BUS*** are very similar, compare black and violet lines.

Table 3 shows the total power and area of selected components in our basic design for the IHP 250 nm technology and their contributions to the power and area of the whole *kP* design.

The similarity of the attack results using the *kP* and the ***BUS*** traces shows that the activity of the ***BUS*** can be the strong SCA leakage source that causes the high success of the horizontal attacks. To be sure that this assumption is true we calculated the correlation coefficients between the PTs of the whole *kP* design and its components for both technologies. Table 4 shows the correlation coefficients.

For both technologies, the ***BUS*** is an SCA leakage source and influences the shape of PT of the whole *kP* design significantly. The Pearson coefficients are 0.75 and 0.80 for the 250 nm and 130 nm technology, respectively. But the relative power consumption of the ***BUS*** in the *kP* design synthesized for the 130 nm technology is less than the one for the 250 nm design: 4% and 9%, respectively (see Tables 2 and 3). Thus, the activity of the ***BUS*** is better hidden or compensated by the other components in the 130 nm technology than in the 250 nm design.

Table 4
Pearson Coefficients calculated for the whole *kP* design and its components.

technology		*MULT*	*ALU*	4 registers together: X_1, X_2, Z_1, Z_2	*Controller*	*BUS*
250 nm	*kP*	0.51	0.42	0.10	0.42	0.75
130 nm	*kP*	0.66	0.47	0.20	0.23	0.80

Based on the results shown in Figs. 7–9, Table 2 - Table 4 we assumed that the activity of the ***BUS*** is the main SCA leakage source. If this assumption is true, the reason why the performed horizontal attack was successful, is the key-dependent addressing of the components for the write-to-bus and for the read-from-bus operations. Fig. 10 shows the key-dependent addressing of components in our basic *kP* design for the write-to-bus operation for the main loop of the implemented Montgomery *kP* algorithm. The addressing was already shown in Fig. 2.

In sections III and IV we discuss two different strategies to increase the inherent resistance of the *kP* design: the possibility of scheduling the block addressing regularly and reducing the relative contribution of the ***BUS*** to the PT of the whole design by increasing the field multiplier's contribution.

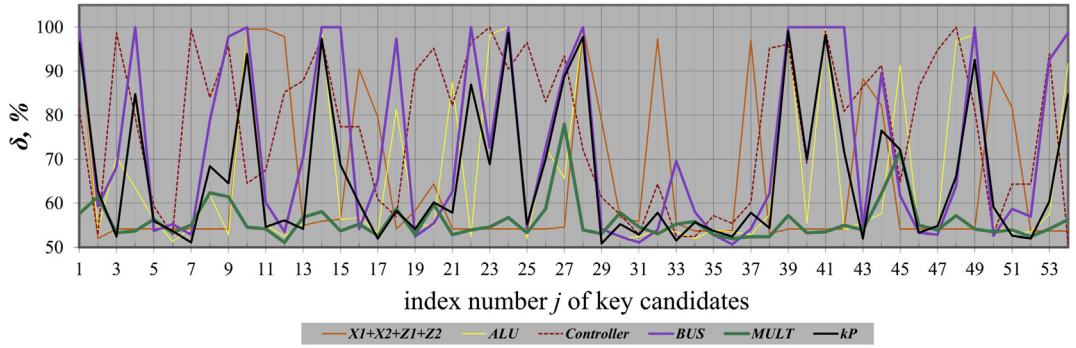


Fig. 9. Horizontal attack using *the comparison to the mean method*: results of the analysis for the whole 250 nm *kP* design and its components.

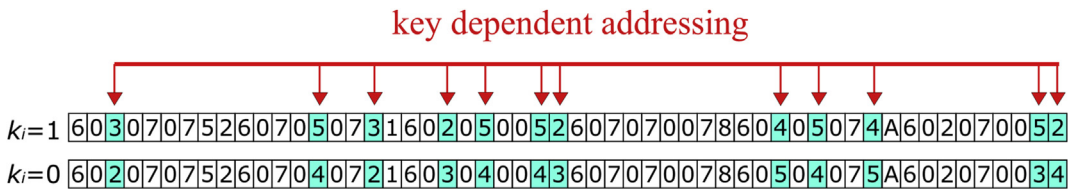


Fig. 10. Key dependent addressing of components in our basic *kP* design for the write-to-bus operation.

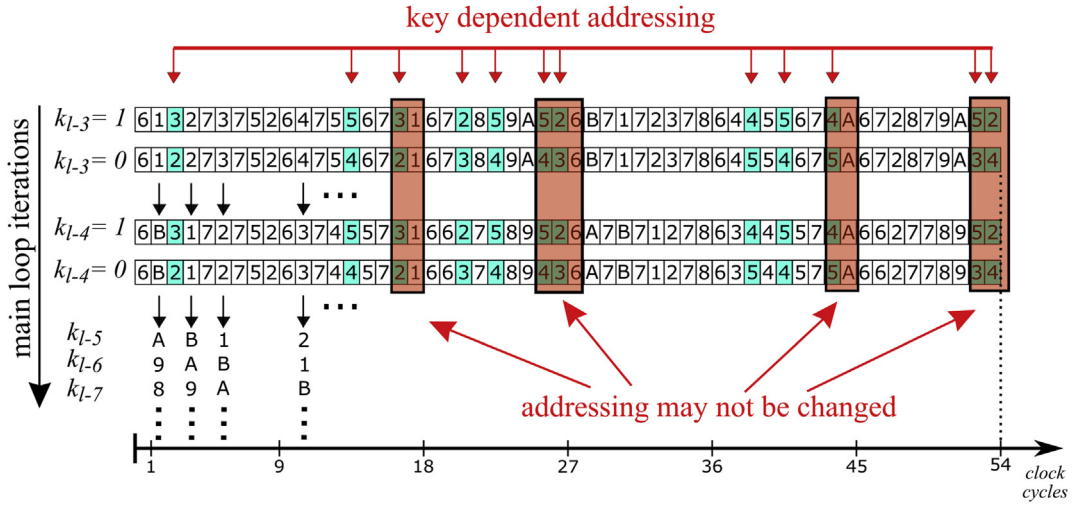
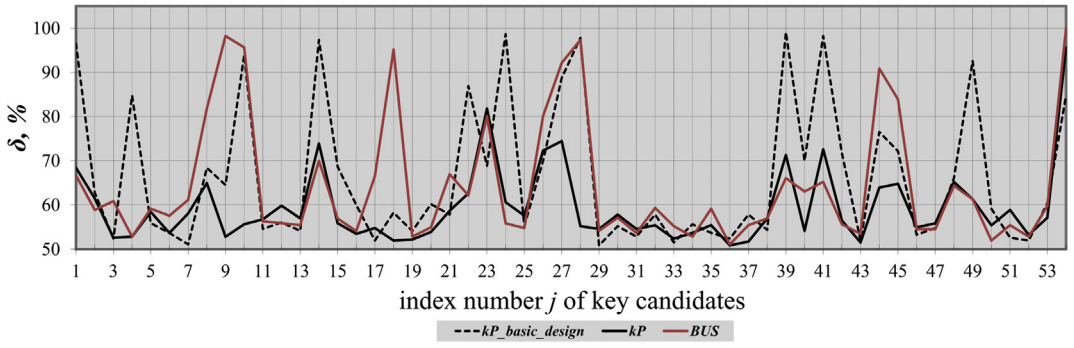
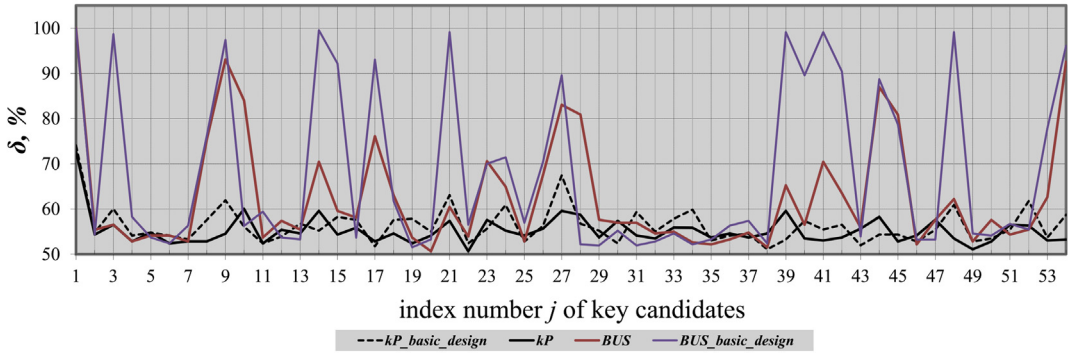


Fig. 11. Details about the main loop implementation in our *kP* design with the regular schedule: the sequences of the numbers show which component writes to the ***BUS*** in each clock cycle of the slots using the processing of key bits k_i, with $l\text{-}3 \geq i \geq l\text{-}7$ as examples.



(a) 250 nm technology



(b) 130 nm technology

Fig. 12. Horizontal attack using *the comparison to the mean method*: results of the analysis for the whole *kP* design and its ***BUS*** for both IHP technologies (design with the regularized addressing).

Table 5
Power of selected components of *kP* design with the regular schedule.

	power	*kP*	*MULT*	*ALU*	4 registers	*Controller*	*BUS*
250 nm	*total, mW*	40.7	22.4	2.48	5.02	1.03	3.35
	relative to kP design, %	100	55	6.1	12.4	2.5	8.2
130 nm	*total, mW*	5.40	3.13	0.35	0.62	0.13	0.31
	relative to kP design, %	100	58	6.5	11.6	2.5	5.7

Table 6
Analysis results for the design with the regularized schedule.

design		Number of key candidates extracted with a correctness $\delta \geq 90\%$		Pearson coefficient	
		kP design	*BUS*	*kP* to *BUS*	*kP* to *MULT*
250 nm	basic design	7	19	0.75	0.51
	design with the regular schedule	1	7	0.69	0.57
130 nm	basic design	0	18	0.80	0.66
	design with the regular schedule	0	3	0.66	0.68

3. Regular scheduling for the block addressing

In this section we describe shortly the results of our experiments with the regularization of the component addressing for the *kP* design. The new *kP* design evaluated here differs from our basic design only in its sequence of addressing components for the write-to-bus operation. Fig. 11 shows the addressing sequence for the new design. More designs implementing the regular addressing of components are described detailed in Ref. [29]. Our basic design implements the addressing sequence corresponding to Fig. 10. In our new design the component with address 0 no longer writes to the ***BUS***. We implemented within a main loop iteration (i.e. in a slot) an addressing sequence in which instead of the block with the address 0 the blocks with addresses from 1 up to 11 (1 to B in hexadecimal) are selected always in a fixed sequence, i.e. one after the other. There are 21 clock cycles with address 0 in our basic design. This means that in the first main loop iteration that corresponds to the processing of key bit k_{l-3} the sequence of the block addresses starts with 1 and ends with 10 (hexadecimal A). In the second main loop iteration that corresponds to the processing of key bit k_{l-4} the sequence is continued starting with 11 (hexadecimal B) and ending with 9. In the third main loop iteration it starts with 10 (hexadecimal A) and ends with 8, and so on. Thus, in the new design all blocks are addressed equally often within a *kP* execution. The addressing sequences for the first two key bits processed in the main loop (i.e. for different values k_{l-3} and k_{l-4}) are shown completely in Fig. 11. For the key bits k_{l-5}, k_{l-6}, and k_{l-7} the beginning of the main loop iteration is shown to illustrate the implemented regular addressing sequence.

In [29] the *kP* design with the regular schedule of the component addressing was ported to an FPGA and the measured traces of the whole *kP* design were analysed. Here we analyse the PT simulated for our 250 nm and 130 nm technologies. The analysis of the simulated trace allows to perform the attack for the individual components of the design as we have shown in the previous section. We use this to evaluate our assumptions about the reasons of the attack success.

We analysed the traces of the whole *kP* design and the ***BUS*** after applying the regular schedule and synthesizing it for both IP technologies. Fig. 12 shows the results of this analysis for the whole *kP* design (see solid black line) and for the ***BUS*** (see red line).

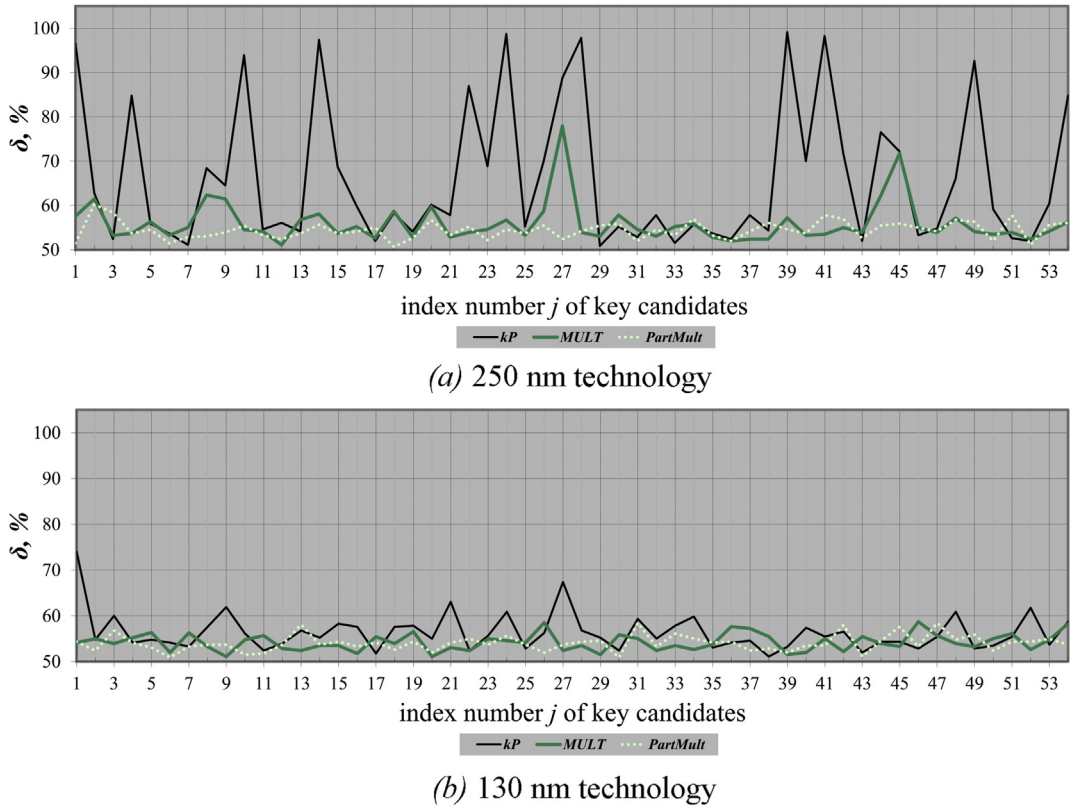


Fig. 13. Success rate attacking our basic design synthesized for both IHP technologies: traces of whole *kP* design, its component ***MULT***, and for the *Partial Multiplier*.

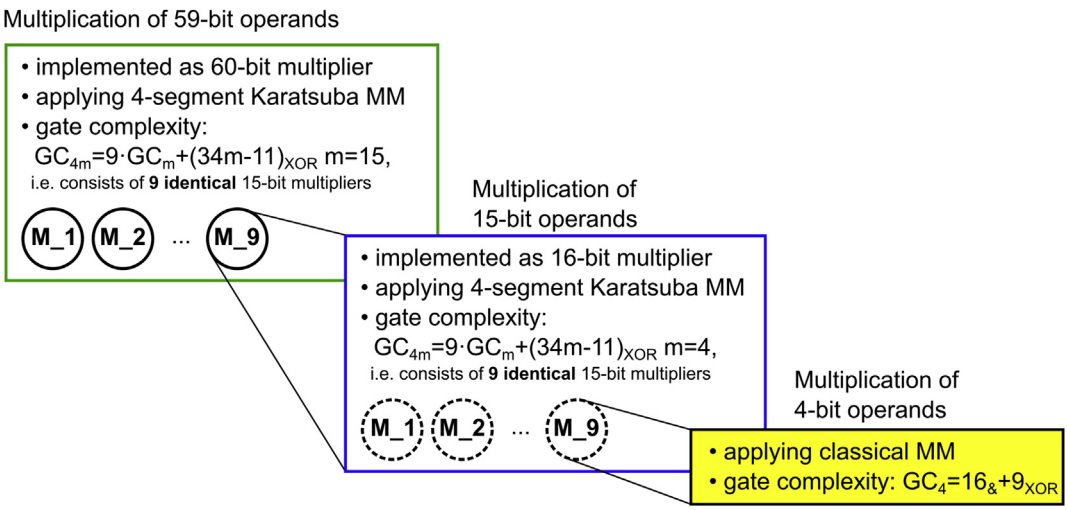


Fig. 14. Partial Multiplier ***ikm***.

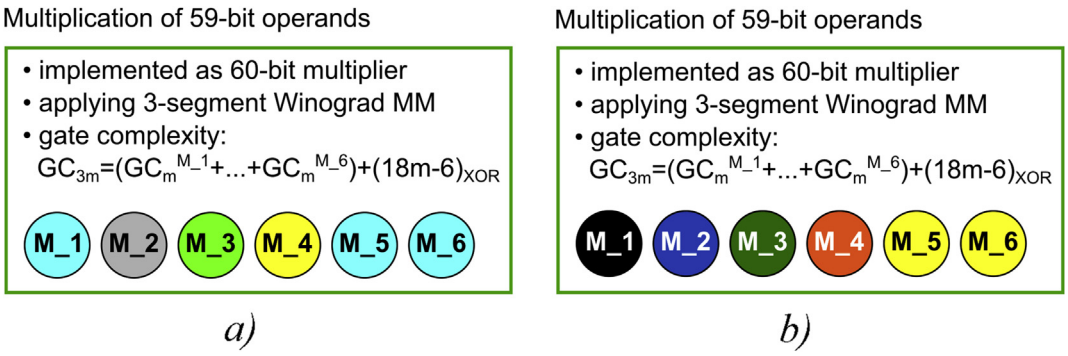


Fig. 15. Partial multipliers implemented using a combination of different multiplication formulae: *a)* - ***combi1***; *b)* – ***combi2***.

Fig. 12 clearly shows that the success of the performed horizontal attack for the 250 nm technology is significantly reduced for the design with the regular schedule of addresses in comparison to our basic design (compare the black dotted and black solid lines in Fig. 12-*(a)*). Also for the design in the 130 nm technology, the success rate of the attack analysing the trace of the ***BUS*** is significantly reduced (compare the red and violet lines in Fig. 12-*(b)*).

Table 5 shows the total power of selected components and their contributions to the power of the *kP* design with the regular schedule for both IHP technologies.

In comparison to our basic design the absolute and relative power of the ***BUS*** is slightly reduced, compare Table 5 with Tables 2 and 3. Table 6 gives an overview of the number of key candidates, extracted with a correctness of more than 90% for each design and the Pearson correlation coefficients between the PT of the whole *kP* design and the PT of the component ***BUS***.

Comparing the Pearson coefficients calculated for *kP* and ***BUS*** traces shows that the regular schedule reduces the influence of the ***BUS*** on the shape of the *kP* trace, whereby the influence of the ***MULT*** is slightly increased for both technologies, see Table 6. The impact of the change in the influences is manifested in the reduced success of the performed horizontal attack.

Please note that in our designs due to the implemented logic not the complete sequence of the addressing of components can be adapted. This holds true for the following sequence of clock cycles: 7–10, 16–19, 26–28, 35–37, 43–46, 53-54-1. The 5 key candidates with the highest correctness obtained using the PT of the new design are 10, 17, 18, 28 and 45, i.e. they are part of the clock cycle sequences that cannot be modified. Due to this fact, a significant reduction of the success rate of the attacks for these clock cycles was not expected. Increasing noise can reduce the SCA leakage. The noise can be increased using the activity of a big and resistant component. In our case only the field multiplier can be such a strong noise source due to the following facts:

- The multiplier is the component that consumes the most energy in the design and influences the shape of the *kP* trace significantly.
- The multiplier is always active i.e. consumes energy in each clock cycle.
- The multiplier's energy consumption in each clock cycle depends on the multiplication formula applied for the implementation of its main component i.e. of the Partial Multiplier. There are many MMs and their combinations for implementing the Partial Multiplier.

In the next section we evaluate this idea.

4. The “most resistant” multiplier

The field multiplier in our basic design synthesized for the IHP 130 nm technology is resistant against the performed horizontal differential power analysis attack. The resistance of the design obtained using the same VHDL code synthesized for the IHP 250 nm technology is smaller than the one of the 130 nm technology. Fig. 13 shows the success rate of the attack analysing the power trace of the whole *kP* design, its field multiplier (i.e. the component ***MULT***) and the Partial Multiplier that is a

Table 7
Parameters of *kP* designs with different Partial Multipliers.

Technology	Investigated *kP* designs			Field multiplier				
	name	area, mm^2	Power mW	Partial Multiplier	area		power	
					Total mm^2	relative to *kP* design, %	totalmW	relative to *kP* design, %
130 nm	design1	0.28	5.40	**basic**	0.11	39.2	3.13	57.9
	design2	0.31	7.76	**clas**	0.14	45.8	5.49	70.8
	design3	0.28	5.60	**ikm**	0.11	39.8	3.33	59.5
	design4	0.29	6.01	**combi1**	0.13	43.1	3.74	62.3
	design5	0.29	6.1	**combi2**	0.13	43.0	3.79	62.5
250 nm	design1	1.38	40.9	**basic**	0.50	36.1	22.4	54.7
	design2	1.50	51.1	**clas**	0.61	41.0	32.6	63.7
	design3	1.40	42.7	**ikm**	0.51	36.7	24.2	56.5
	design4	1.46	45.1	**combi1**	0.58	39.5	26.6	58.8
	design5	1.46	46.0	**combi2**	0.57	39.4	27.4	59.6

component of the ***MULT*** for both technologies, for comparison.

It can be seen in Fig. 13 that the Partial Multiplier is resistant against the performed attack for the designs synthesized in both technologies. It processes relatively long operands – the 59 bit long partial multiplicands – in each clock cycle implemented as a combination of the 2-segment Karatsuba MM, the 6-segment Winograd MM and the classical MM for 5 bit long operands as described in section II-*D*. Due to the fact that other combinations of the MM result in a different area and energy consumption, we assumed that the combination of the MM used for implementing the Partial Multiplier can have a significant impact on the resistance of the whole *kP* design. In this section we investigate this assumption with the goal do determine the combination of the MM resulting in the "most secure" field multiplier and *kP* design.

We implemented 4 additional designs to investigate the impact of the MM on the success of the horizontal attack. All designs differ from our basic design only in their Partial Multiplier. In the next subsections we explain the implementation details of our partial multipliers, give their parameters such area and average power consumption, and discuss the results of our attacks.

4.1. Partial multiplier **clas**

This partial multiplier was implemented using only the classical multiplication formula, i.e. it implements the following formula:

$$C = A \cdot B = \sum_{i=0}^{2n-2} c_i \cdot t^i, \quad with \quad c_i = \bigoplus_{i=k=l} a_k \cdot b_l, \forall k, l < n \tag{8}$$

here $n = 59$ is the length of the partial multiplicands.

The gate complexity of this multiplier, i.e. the amount of AND and XOR gates which are necessary to implement its functionality corresponding to formula (8) is n^2 AND gates and $(n\text{-}1)^2$ XOR gates, i.e.: $GC_{n=59} = 3481_{\&} + 3364_{XOR}$.

4.2. Partial multiplier **ikm**

This partial multiplier processing 59 bit long operands was implemented using the 4-segment iterative Karatsuba multiplication formula for 60-bit long multiplicands. The gate complexity of this multiplier is $GC_{4m} = 9 \cdot GC_m + (34m\text{-}11)_{XOR}$. Here m is the length of the segments $m = 60/4 = 15$ and GC_m is the gate complexity of the internal *m-bit* partial multipliers. In the Partial Multiplier ***ikm*** we implemented all 9 internal 15-bit partial multipliers identically. Each of them was again implemented using the 4-segment iterative Karatsuba multiplication formula. Thus, the 15-bit multiplier consists of 9 internal 4-bit multipliers, each implemented using the classical multiplication formula (8) with $n = 4$. Fig. 14 shows the structure of ***ikm***.

4.3. Partial multipliers **combi1** and **combi2**

These Partial Multipliers of 59 bit long operands were implemented as a combination of 3 multiplication formulae: the 3-segment iterative Winograd MM, the 4-segment iterative Karatsuba MM and the classical MM. At first the 3-segment iterative Winograd MM was applied for 60-bit long operands. This MM defines the number of the internal multipliers: ***combi1*** and ***combi2*** contain 6 internal multipliers *M_1, …,M_6* for 20-bit long operands. The multipliers *M_1*, *M_5* and *M_6* in ***combi1*** are identical (see blue marked multipliers in Fig. 15-(a)). The multiplier *M_5* and *M_6* in ***combi2*** are identical to the multiplier M_1 in ***combi1*** and are implemented using the classical MM only (see yellow marked multipliers in Fig. 15-(a) and b)). The combination of MMs for the implementation of the multipliers *M_1*, …,*M_4* in ***combi1*** and all multipliers of ***combi2*** was chosen randomly.

Details about the iterative 4-segment Karatsuba MM and the 3-segment Winograd MM are given in Ref. [36–38,40]. We do not give details here for simplifying the reading. Important is only the fact, that the complexity of the Partial Multipliers is different.

We synthesized the *kP* designs using the Partial Multipliers described here for the IHP 130 nm and 250 nm technologies. Table 7 gives a short overview of the parameters of the *kP* designs with the different partial multipliers described here. The goal is to show that the main parameters of the investigated *kP* designs such as their area and power are different.

4.4. Analysis results

For both technologies, the 5 implemented Partial Multipliers can be considered as resistant against our attack due to the fact that the correctness of all key candidates is smaller than 60%. The graph representing the success of the attack is similar to the green dotted line in Fig. 13.

Figs. 16 and 17 show the results of the analysis attacking traces of all implemented *kP* designs and their field multipliers.

Fig. 16 –(a) clearly shows that the *kP* design with the Partial Multiplier implemented using the classical multiplication formula is the best: no key candidates are extracted with a correctness higher than 60% (see dotted black line). Designs with other Partial Multipliers have at least one key candidate with a correctness of 70%. The resistance of all implemented field multipliers is very high: there are no key candidates with a correctness higher than 60%, see Fig. 16 –(b).

The correctness of the revealed scalar using traces of the whole *kP* design is very high in all investigated cases (Fig. 17-(a)), but the *kP* design with the Partial Multiplier that was implemented using the classical MM is the most resistant one (see dotted black line). Results in Fig. 17-(b) show that the use of the classical MM for implementing the Partial Multiplier results in the most resistant field multiplier: the success rate of the attack is reduced from 78% correctness of the best key candidates

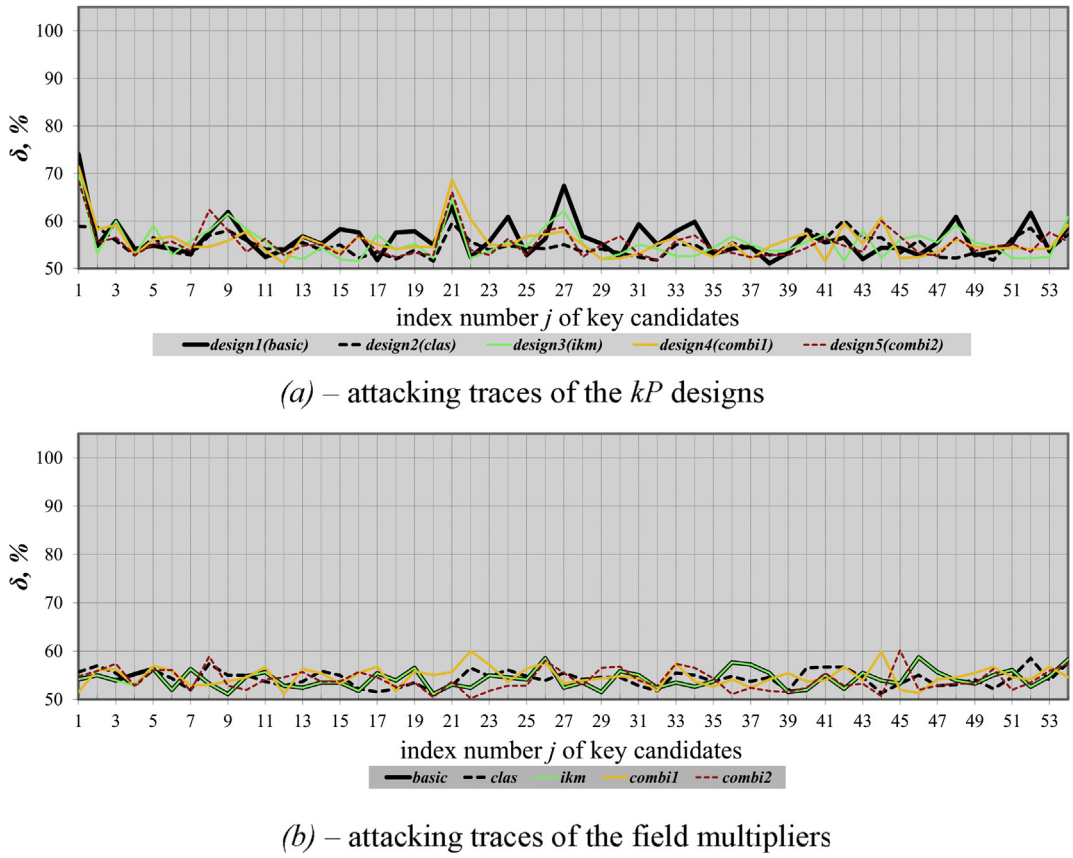


Fig. 16. Horizontal DPA attack using the traces of the different *kP* designs and their field multipliers synthesized for the IHP 130 nm technology.

down to 67%, see correctness of the $k^{27}_{candidate}$.

Thus, for both technologies, the *kP* design with the partial multiplier implemented using the classical multiplication formula is the best, i.e. the correctness of the extracted key candidates for this design (i.e. for the *design3*) is significantly smaller than for all other designs.

Additionally, we calculated the Pearson correlation coefficients for the trace of the whole *kP* design and traces of the two components: ***BUS*** and ***MULT***. Table 8 shows the correlation coefficients for all 5 designs for both technologies.

Only for the design with the Partial Multiplier implementing the classical MM is the correlation coefficient for the whole *kP* design and its field multiplier significantly higher than the correlation coefficient of the *kP* design and its ***BUS*** (see column marked with light green). Thus, the influence of the resistant component ***MULT*** in the design is higher than the influence of the component ***BUS*** that is an SCA leakage source. So, the activity of the field multiplier hides the activity of the ***BUS***. Fig. 18 shows the mean slot values to explain this. The power profile of the ***BUS*** is the same for all 5 designs within the same technology. The power profile of the ***MULT***, its fluctuation as well as its relative contribution to the power consumption of the whole design is different for each design. Thus, the component ***MULT*** is a kind of inherent countermeasure against SCA attacks if the Partial Multiplier is implemented using the classical MM.

The results of the analysis show that the implemented formula for the partial multiplication has a significant impact on the resistance of the *kP* design against horizontal DPA. The *kP* design with the partial multiplier that implemented the classical multiplication formula shows the highest resistance. This is due to the fact that compared to the other designs investigated, not only the average power consumption of designs with the classical partial multiplier, but also the amplitude range of the power trace, is the highest one. Please note that none of the designs with the different Partial Multipliers has the regular schedule of the addressing of the components (see Fig. 10).

5. The "most resistant" multiplier and regular scheduling: impact of the combination

In this section we describe the results of the analysis attacking our last *kP* design. In this design we combined the regular schedule of the block addressing as described in section III with the most resistant field multiplier i.e. the Partial Multiplier using the classical multiplication formula. Table 9 shows the total power of selected components and their contributions to the power of the *kP* design for both IHP technologies.

Table 10 gives an overview of the number of key candidates, extracted with a correctness of more than 70% for each design and the Pearson correlation coefficients between the PT of the whole *kP* design and the PT of the component ***BUS***. Please note that no key candidate with a correctness higher than 75% was extracted for the design synthesized for the 250 nm technology. This means that even if the attacker knows on which positions the key bits can be detected wrong he needs to perform $2^{(100\%-75\%)\cdot 233\text{bits}} \approx 2^{59} \approx (2^{10})^6 \approx 10^{18}$ *kP* operations for revealing the correct values on these positions via brute force. Using our specialized hardware *kP* accelerator in the IHP 130 nm technology that can calculate about 10000 *kP* operations per second[3] it needs 10^{14} s i.e. 3000000 years. Using 1 billion of such fast devices this task can be performed in 1 day only. But the energy required for this task is 2×10^{12} J that is the energy produced by the largest power producing station during 3 min.[4] We give here these estimations to show how costly brute forcing of only a quarter of the bit values of a 233 bit long key is.

Fig. 19 shows the results of the analysis for all 4 *kP* designs investigated here. The white dotted line corresponds to our basic design; the

[3] at the maximal clock frequency of 200 MHz our *kP* accelerator can calculate about 10 000 *kP* operations per second (values after synthesis).

[4] The annual generation of the Three Gorges Dam power station is 93.5 TWh [41].

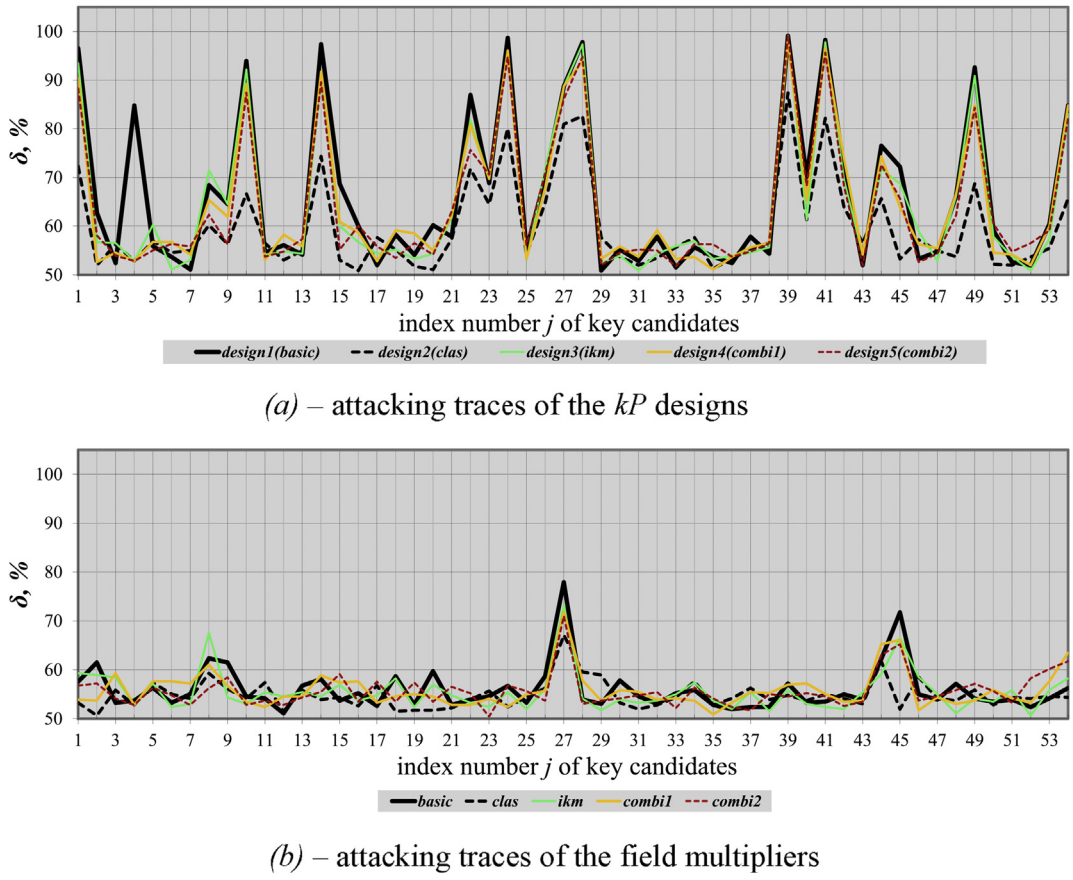


Fig. 17. Horizontal DPA attack using the traces of the different *kP* designs and their field multipliers synthesized for the IHP 250 nm technology.

blue dotted line to the design with the regular schedule and the green line to the design with the "most resistant" multiplier. The solid black line shows the success of the attack for our last design i.e. for the design combining the regular schedule with "most resistant" multiplier.

Fig. 19-(a) clearly shows that the success of the performed horizontal attack for the 250 nm technology is significantly reduced for our last design, see black line. Only 3 key candidates with a correctness of 71%, 69% and 73% are extracted, see $k^{20}_{candidate}$, $k^{27}_{candidate}$, $k^{54}_{candidate}$. The correctness of other key candidates is between 50% and 63%. For the design in 130 nm technology, the success rate of the attack is really small: the correctness of all key candidates is between 50% and 60%, see black line in Fig. 19 -(b).

Table 10 shows the number of the key candidates extracted with a correctness of more than 70% for the *kP* design in both technologies and Pearson coefficients between the whole *kP* design and its ***BUS*** and its ***MULT*** respectively.

The comparison of the correlation coefficients for all designs investigated here shows that in our last design the influence of the ***BUS*** is the smallest one and the influence of the multiplier is the highest one (compare Table 10, Tables 8 and 6) for both IHP technologies.

6. Conclusions

In this paper we investigated two different approaches to increase the inherent resistance of hardware implementations of the scalar point multiplication for elliptic curve cryptography using the Montgomery algorithm. The Montgomery algorithm is known to be resistant against simple side channel analysis attacks as the sequence of operations is independent of the key bit value. But the use of the registers depends on the key bit value. This dependency can be exploited to reveal the key. Here we introduced and evaluated the idea of a regular schedule when addressing different components and registers in the design. Such a regular schedule helps to reduce the success rate of horizontal differential power analysis attack significantly i.e. the number of key candidates with a correctness of more than 90% was reduced from 7 to 1 for the design synthesized in our 250 nm technology. When only the ***BUS*** i.e. the communication unit inside our design is investigated the regular schedule reduces the number key candidates with a correctness of more than 90% from 19 to 1 and from 17 to 0 for the 250 nm and the 130 nm technology respectively.

In addition, we investigated the impact of the implemented formula

Table 8
Pearson Coefficients calculated for the whole *kP* design and its components BUS and MULT.

technology	trace of the component	trace of the *kP* design				
		design1	design2	design3	design4	design5
130 nm	***BUS***	0.80	0.65	0.79	0.80	0.79
	MULT	0.66	0.88	0.70	0.71	0.73
250 nm	***BUS***	0.75	0.64	0.74	0.74	0.72
	MULT	0.51	0.76	0.56	0.61	0.63

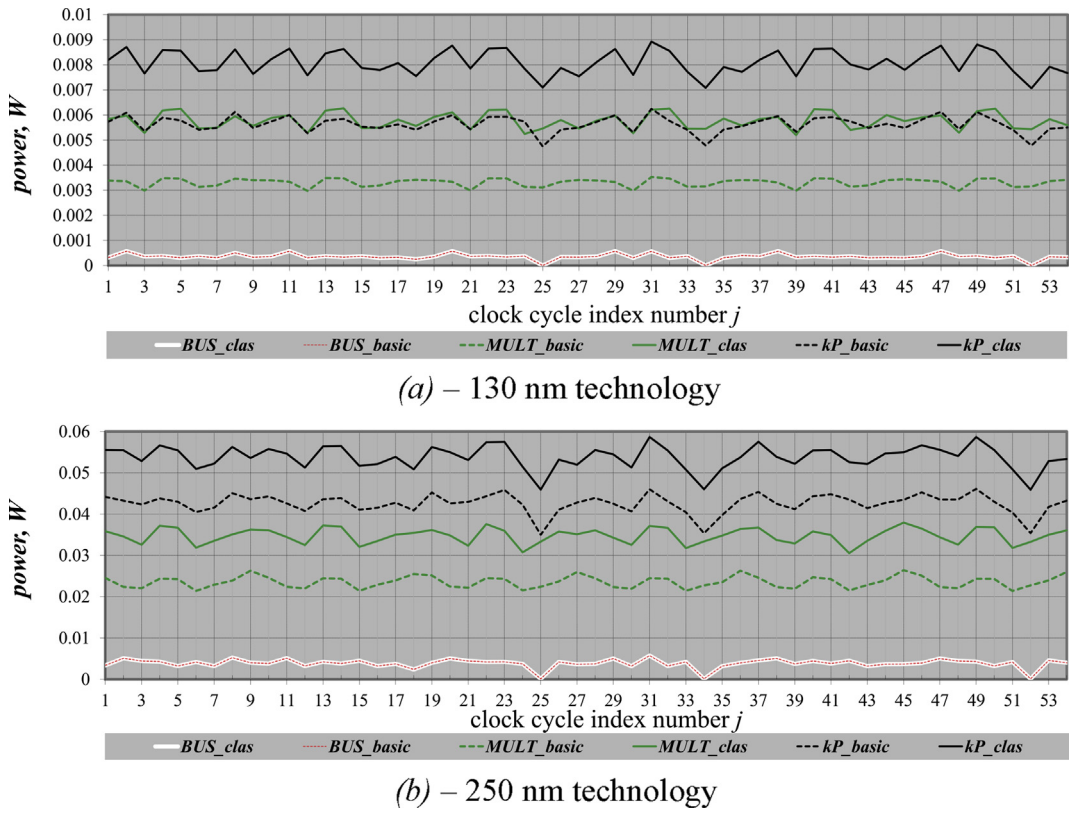


Fig. 18. Mean slot of the traces: *kP* design, field multiplier ***MULT***, ***BUS***.

for the partial multiplication on the results of a horizontal DPA attack. We investigated 5 *kP* designs synthesized for our two technologies. The designs differ only in their partial multipliers: each multiplier was implemented using a different multiplication formula or a combination of multiplication methods. Thus, each partial multiplier has its own gate complexity, structure and circuit. Our experiments show clearly that the field multiplier with the partial multiplier implemented using the classical multiplication formula only is the most resistant one against the performed DPA attack for both gate technologies. Using this partial multiplier reduces the success rate of the best key candidate from 78% down to 67% attacking traces of the field multiplier.

Combining these two approaches increases the resistance of the *kP* design significantly. The combination of the regular schedule with the most resistant version of the field multiplier ensures that there are only 2 key candidates with a success rate of about 70% for our 250 nm technology. For the 130 nm technology no key candidate could be revealed with a correctness of more than 60%.

In order to explain the above-reported effects, we investigated the contribution of the different components i.e. the partial multipliers as well as the one of the ***BUS*** to the power consumption of the whole *kP* design. The regular schedule reduces the influence of the ***BUS*** on the whole *kP* design represented as Pearson correlation coefficients from 0.75 to 0.69 for the 250 nm technology and from 0.80 to 0.66 for the 130 nm technology. In addition, the influence of the field multiplier is increased from 0.51 to 0.57 in the 250 nm technology and from 0.66 to 0.68 in 130 nm technology. When the field multiplier is implemented using the classical multiplication formula the influence of the ***BUS*** is further reduced down to 0.64 and 0.65 in 250 nm and 130 nm technologies, respectively whereby the influence of the field multiplier is further increased up to 0.76 and 0.88 in 250 nm and 130 nm technologies, respectively. The best results are obtained combining of both approaches: for the 250 nm technology the influence of the ***BUS*** is reduced down to 0.53 by increasing of the influence of the field multiplier up to 0.80; for the 130 nm technology the influence of the ***BUS*** is reduced down to 0.48 by increasing of the influence of the field multiplier up to 0.90. As the field multiplier is by far more resistant against side channel analysis attacks than the whole design, i.e. there is only one key candidate with a highest correctness of 67% for the 250 nm technology and zero key candidates with a correctness higher than 60% for the 130 nm technology it can be considered as an appropriate means to improve the inherent resistance of the whole *kP* design.

Table 9
Power of selected components of the *kP* designs.

	power	*kP*	*MULT*	*ALU*	4 registers	*Controller*	*BUS*
250 nm	total, mW	50.9	32.6	2.48	5.02	1.03	3.36
	relative to *kP* design, %	100	64	4.9	9.8	2	6.6
130 nm	total, mW	7.76	5.49	0.35	0.62	0.13	0.31
	relative to *kP* design, %	100	70.8	4.6	8	1.7	4

Table 10
Analysis results for the *kP* design with the regularized schedule and the "most resistant" multiplier.

technology	Number of key candidates extracted with a correctness $\delta \geq 70\%$	Pearson coefficient	
		kP to BUS	*kP to MULT*
250 nm	2	0.53	0.80
130 nm	0	0.48	0.90

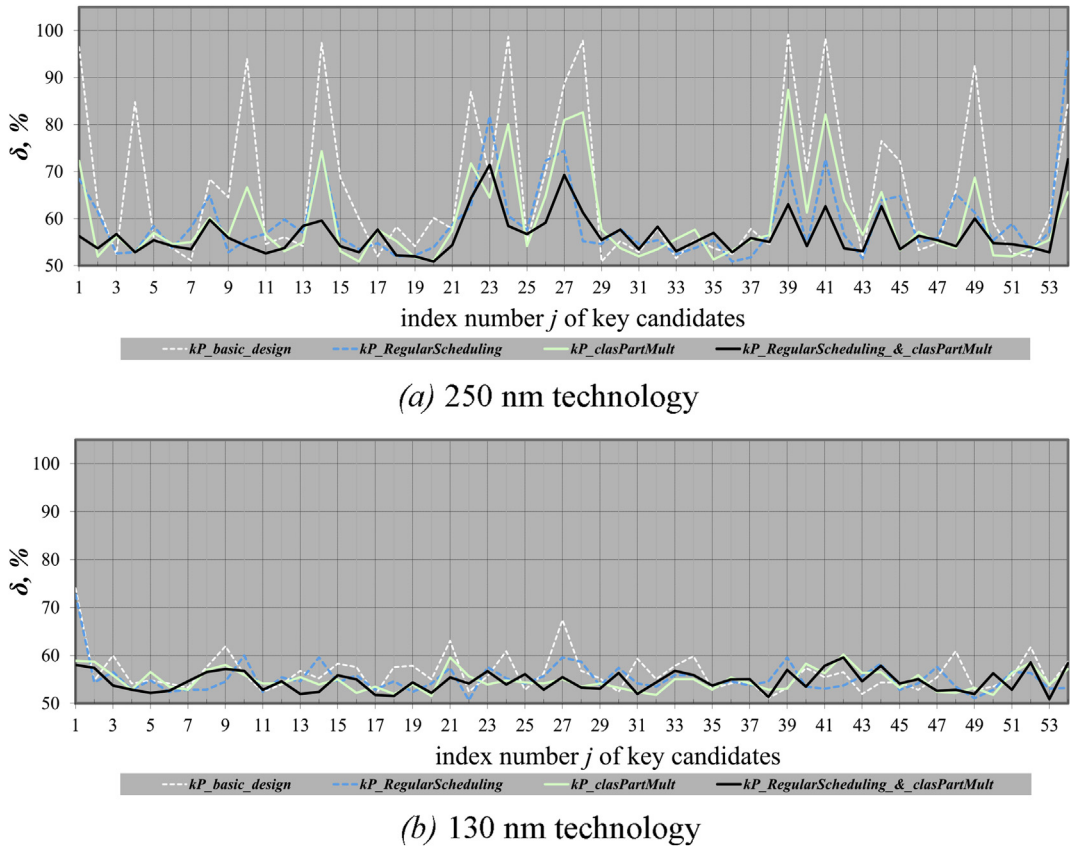


Fig. 19. Horizontal attack using the comparison to the mean method: results of the analysis of all investigated here *kP* designs for both IHP technologies.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Ievgen Kabin: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing - original draft, Writing - review & editing, Visualization. **Zoya Dyka:** Conceptualization, Methodology, Validation, Resources, Writing - review & editing, Supervision, Funding acquisition. **Dan Klann:** Software, Writing - review & editing, Visualization. **Peter Langendoerfer:** Conceptualization, Writing - review & editing, Supervision, Project administration, Funding acquisition.

References

[1] D. Halperin, T.S. Heydt-Benjamin, B. Ransford, S.S. Clark, B. Defend, W. Morgan, K. Fu, T. Kohno, W.H. Maisel, Pacemakers and Implantable Cardiac Defibrillators: Software Radio Attacks and Zero-Power Defenses, in: 2008 IEEE Symposium on Security and Privacy (Sp 2008), 2008, pp. 129–142, https://doi.org/10.1109/SP.2008.31.

[2] The Telegraph Hackers Could Kill Patients by Attacking Their Pacemakers, warns Royal Academy of Engineering, 2018 available at: https://www.telegraph.co.uk/science/2018/03/14/hackers-could-kill-patients-attacking-pacemakers-warns-royal/. (Accessed 3 May 2019).

[3] F. Sommer, J. Dürrwang, R. Kriesten, Survey and classification of automotive security attacks, Information 10 (2019) 148.

[4] E. Schoch, F. Kargl, On the efficiency of secure beaconing in VANETs, in: 3rd ACM Conference on Wireless Network Security, 2010.

[5] Information Technology Laboratory, Digital Signature Standard (DSS), National Institute of Standards and Technology, Jul. 2013, https://doi.org/10.6028/NIST.FIPS.186-4. NIST FIPS 186–4.

[6] J. Coron, Resistance against differential power analysis for elliptic curve cryptosystems, Proc.of CHES 1717 (1999) 292–302. LNCS.

[7] E. Oswald, M. Aigner, Randomized addition-subtraction chains as a countermeasure against power attacks, Proc. CHES 2162 (2001) 39–50. LNCS.

[8] C. Clavier, B. Feix, G. Gagnerot, M. Roussellet, V. Verneuil, Horizontal correlation analysis on exponentiation 6476, Springer, 2010, pp. 46–61. Proc. ICICS, LNCS.

[9] N. Homma, Atsushi miyamoto, takafumi aoki, akashi satoh, adi shamir: collision-based power analysis of modular exponentiation using chosen-message, CHES 5154 (2008) 15–29. LNCS.

[10] P.-A. Fouque, F. Valette, The doubling attack – why upwards is better than downwards, CHES 2779 (2003) 269–280. LNCS.

[11] T.S. Messerges, E.A. Dabbish, R.H. Sloan, Power analysis attacks of modular exponentiation in smartcards, CHES 1717 (1999) 144–157. LNCS.

[12] Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Advances in Cryptology — CRYPTO' vol. 99. pp. 388–397.

[13] E. Brier, C. Clavier, F. Olivier, Correlation power analysis with a leakage model, in: Proc. CHES, 2004, pp. 16–29.

[14] Z. Dyka, E.A. Bock, I. Kabin, P. Langendoerfer, Inherent Resistance of Efficient ECC Designs against SCA Attacks, in: 2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS), 2016, pp. 1–5, https://doi.org/10.1109/NTMS.2016.7792457.

[15] I. Kabin, Z. Dyka, D. Kreiser, P. Langendoerfer, Evaluation of resistance of ECC designs protected by different randomization countermeasures against horizontal DPA attacks, in Proc. EWDTS (2017) 1–7.

[16] I. Kabin, E.A. Bock, C. Wittke, D. Kreiser, Z. Dyka, P. Langendoerfer, On the influence of hardware technologies on the vulnerability of protected ECC implementations, in: Proc. DSD (Work in Progress Session), 2016.

[17] S.A. Kadir, A. Sasongko, Simple power analysis attack against elliptic curve cryptography processor on FPGA implementation, Proc. ICEEI (2011) 1–4.

[18] E. De Mulder, P. Buysschaert, S.B. Ors, P. Delmotte, B. Preneel, G. Vandenbosch, I. Verbauwhede, Electromagnetic analysis attack on an FPGA implementation of an elliptic curve cryptosystem, EUROCON (2005) 1879–1882.

[19] C.D. Walter, Sliding windows succumbs to big Mac Attack, in: Proc. CHES, 2001, pp. 286–299.

[20] J. Heyszl, S. Mangard, B. Heinz, F. Stumpf, G. Sigl, Localized electromagnetic analysis of cryptographic implementations, in: Topics in Cryptology – CT-RSA, 2012, pp. 231–244.

[21] A. Bauer, E. Jaulmes, E. Prouff, J. Wild, Horizontal and vertical side-channel attacks against secure RSA implementations, in: Topics in Cryptology – CT-RSA, 2013, pp. 1–17.

[22] A. Bauer, E. Jaulmes, E. Prouff, J. Wild, Horizontal collision correlation attack on elliptic curves, in: SAC, 2013, pp. 553–570.
[23] I. Kabin, Z. Dyka, D. Kreiser, P. Langendoerfer, Horizontal address-bit DPA against Montgomery kP implementation, in: Proc. Of the ReConFig, 2017.
[24] I. Kabin, Z. Dyka, D. Kreiser, P. Langendoerfer, Horizontal address-bit DEMA against ECDSA, in: Proc. Of the NTMS, 2018.
[25] L. Goubin: A refined power-analysis attack on elliptic curve cryptosystems. In: Desmedt, Y.G. (ed.) Public Key Cryptography – PKC-2003, LNCS, vol. vol. 2567, pp. 199-211.
[26] P. Fouque, D. Real, F. Valette, M. Drissi, The carry leakage on the randomized exponent countermeasure, in: Proc. CHES, vol. 5154, LNCS, 2008, pp. 198–213.
[27] M. Dugardin, S. Guilley, J.-L. Danger, Z. Najm, O. Rioul, Correlated extra-reductions defeat blinded regular exponentiation, in: Cryptographic Hardware and Embedded Systems, CHES 2016, Berlin, Heidelberg, 2016, pp. 3–22.
[28] I. Kabin, Z. Dyka, D. Klann, P. Langendoerfer, Horizontal DPA attacks against ECC: impact of implemented field multiplication formula, in: 2019 14th International Conference on Design Technology of Integrated Systems in Nanoscale Era, DTIS, 2019, pp. 1–6.
[29] I. Kabin, D. Kreiser, Z. Dyka, P. Langendoerfer, FPGA implementation of ECC: low-cost countermeasure against horizontal bus and address-bit SCA, in: 2018 International Conference on ReConFigurable Computing and FPGAs, ReConFig, 2018, pp. 1–7.
[30] D. Hankerson, J. Lopez, A. Menezes, Software implementation of elliptic curve cryptography over binary fields, in: Proc. Of CHES, vol. 1965, LNCS, 2000.
[31] A. Karatsuba, Y. Ofman, Multiplication of many-digital numbers by automatic computers, Dokl. Akad. Nauk SSSR 145 (1962) 293–294. Translation in Physics-Doklady, 7 , 1963, pp. 595–596.
[32] S. Winograd, Arithmetic Complexity of Computations, SIAM, 1980.
[33] M. Ernst, M. Jung, F. Madlener, S.A. Huss, R. Bliimel, A reconfigurable system on chip implementation for elliptic curve cryptography over GF(2^n), in: Proc. Of CHES, 2002.
[34] F. Madlener, M. Stoettinger, S.A. Huss, Novel hardening techniques against differential power analysis for multiplication in GF(2^n), Proc.FPT (2009) 328–334.
[35] J. Von zur Gathen, J. Shokrollahi, Efficient FPGA-based Karatsuba multipliers for polynomials over F2, Proc. of SAC 3897 (2005) 359–369. LNCS.
[36] Z. Dyka, P. Langendoerfer, F. Vater: Combining multiplication methods with optimized processing sequence for polynomial multiplier in GF(2^k), Proc. of WEWoRC-2011, LNCS 7242, pp. 137-151
[37] Z. Dyka, P. Langendoerfer, Area efficient hardware implementation of elliptic curve cryptography by iteratively applying Karatsuba's method, Proc. of the DATE 3 (2005) 70–75.
[38] Z. Dyka, Analysis and Prediction of Area- and Energy-Consumption of Optimized Polynomial Multipliers in Hardware for Arbitrary GF(2^n) for Elliptic Curve Cryptography, Dissertation, 2012.
[39] Synopsys, DC compiler, PrimeTime. http://www.synopsys.com/Tools/.
[40] Z. Dyka, P. Langendoerfer, F. Vater, S. Peter, Towards strong security in embedded and pervasive systems: energy and area optimized serial polynomial multipliers in GF(2^k), Proc. NTMS- (2012) 1–6.
[41] List of largest power stations. https://en.wikipedia.org/wiki/List_of_largest_power_stations.