Increasing the Robustness of the Montgomery kP-Algorithm against SCA by Modifying its Initialization

Estuardo Alpirez Bock, Zoya Dyka, and Peter Langendoerfer

IHP
Im Technologiepark 25, Frankfurt (Oder), Germany
{alpirez,dyka,langendoerfer}@ihp-microelectronics.com
http://www.ihp-microelectronics.com

Abstract. *The Montgomery kP-algorithm using Lopez-Dahab projective coordinates is a well-known method for performing the scalar multiplication in elliptic curve crypto-systems (ECC). It is considered resistant against simple power analysis (SPA) since each key bit is processed by the same type, amount and sequence of operations, independently of the key bit's value. Nevertheless, its initialization phase affects this algorithm's robustness against side channel analysis (SCA) attacks. We describe how the first iteration of the kP processing loop reveals information about the key bit being processed, i.e. bit k_{l-2}. We explain how the value of this bit can be extracted with SPA and how the power profile of its processing can reveal details about the implementation of the algorithm. We propose a modification of the algorithm's initialization phase and of the processing of bit k_{l-2}, in order to hinder the extraction of its value using SPA. Our proposed modifications increase the algorithm's robustness against SCA and even reduce the time needed for the initialization phase and for processing k_{l-2}. Compared to the original design, our new implementation needs only 0.12% additional area, while its energy consumption is almost the same, i.e. we improved the security of the design at no cost.*

Keywords: elliptic curve cryptography, Montgomery kP-algorithm, power analysis

1 Introduction

Side channel analysis (SCA) attacks have been a popular research topic in the last years. Parameters like power consumption, electromagnetic radiation and execution time of a cryptographic implementation can be analysed for identifying implementation details and based on this, extracting the private key. The Montgomery kP-algorithm using Lopez-Dahab projective coordinates [1] is an efficient method for performing the scalar multiplication kP in elliptic curve crypto-systems (ECC). This algorithm is a bitwise processing of the l-bit long scalar $k = k_{l-1}, k_{l-2}, \dots, k_1, k_0$; which is the private key used for performing decryption in ECC. It is considered resistant against simple power analysis (SPA).

Nevertheless its first loop iteration (performed for processing the key bit k_{l-2}) reveals information about the value of the key bit being processed. This key bit can be extracted with SPA. Besides this, the power profile of the processing of k_{l-2} can be used for understanding implementation details of the kP-algorithm and thus for the preparation of further attacks.

In this paper we describe how the initialization phase of the Montgomery kP-algorithm affects the algorithm's resistance against SCA attacks. We use simulated power traces (PTs) to show how the power profile of the processing of k_{l-2} differs from the power profiles of the processing of all other key bits. Moreover, we demonstrate that this power profile differs significantly for the cases $k_{l-2} = 1$ and $k_{l-2} = 0$. This leads to an easy extraction of bit k_{l-2} using SPA and exposes details of the implementation of the algorithm, which can be useful for the preparation of further attacks. As a countermeasure against this vulnerability, we propose to process key bit k_{l-2} outside of the algorithm's main loop, with a different operation flow. We show that with this modification, the power profiles of the processings of $k_{l-2} = 1$ and $k_{l-2} = 0$ look similar to each other and similar to the processing of all remaining bits of the key, i.e. the value of the key bit k_{l-2} cannot be extracted using SPA. The initialization phase of the algorithm is shortened, as well as the processing of k_{l-2}. The execution time of a kP-operation using our modified design was reduced by 11 clock cycles. Our modifications did not imply an increase on the energy consumption needed for the calculation of kP, which remains by 2.09 μJ, and our implementation's chip area was increased by only 0.12%.

The rest of this paper is structured as follows. In section 2 we describe the Montgomery kP-algorithm using Lopez-Dahab projective coordinates and discuss its resistance against SCA. Section 3 explains how the processing of k_{l-2} reveals information about the key bit being processed, as well as information regarding the implementation details. In section 4 we present our modifications of the Montgomery kP-algorithm regarding its initialization phase and the processing of k_{l-2}. Section 5 shows results regarding the power profiles, area and energy consumption of our implementation of the original kP-algorithm and our modified version.

2 Montgomery *kP*-Algorithm

The Montgomery kP-algorithm using Lopez-Dahab projective coordinates was introduced in 1999 [1]. The work presented in [2] shows a possible way of implementing this algorithm (see Algorithm 1). Only the value of the x-coordinate of point P is used. No division operations and no operations with the y-coordinates of the EC points need to be performed in the main loop. This reduces the execution time and energy consumption of the calculation of kP. Due to this fact, the algorithm is often implemented for energy constrained devices such as wireless sensor nodes.

Algorithm 1 Montgomery algorithm for the kP-operation using projective coordinates

```
Input: $k = (k_{l-1}, ..., k_1, k_0)_2$ with $k_{l-1} = 1, P = (x, y) \in E(GF(2^m))$.
Output: $kP = (x_1, y_1)$.
 1: $X_1 \leftarrow x$, $Z_1 \leftarrow 1$, $X_2 \leftarrow x^4 + b$, $Z_2 \leftarrow x^2$.
 2: for $i$ from $l-2$ downto 0 do
 3:   if $k_i = 1$ then
 4:     $T \leftarrow Z_1$, $Z_1 \leftarrow (X_1 Z_2 + X_2 Z_1)^2$, $X_1 \leftarrow xZ_1 + X_1 X_2 T Z_2$,
 5:     $T \leftarrow X_2$, $X_2 \leftarrow X_2^4 + bZ_2^4$, $Z_2 \leftarrow T^2 Z_2^2$.
 6:   else
 7:     $T \leftarrow Z_2$, $Z_2 \leftarrow (X_2 Z_1 + X_1 Z_2)^2$, $X_2 \leftarrow xZ_2 + X_1 X_2 T Z_1$,
 8:     $T \leftarrow X_1$, $X_1 \leftarrow X_1^4 + bZ_1^4$, $Z_1 \leftarrow T^2 Z_1^2$.
 9:   end if
10: end for
11: $x_1 \leftarrow X_1 / Z_1$.
12: $y_1 \leftarrow y + (x + x_1)[X_1 + xZ_1)(X_2 + xZ_2) + (x^2 + y)(Z_1 Z_2)]/(xZ_1 Z_2)$.
13: return $((x_1, y_1))$.
```

The Montgomery kP-algorithm is a bitwise processing of the scalar k. The scalar k is the private key used for performing decryption in ECC. Each bit of k, except its most significant bit (MSB), is processed with the same type, amount and sequence of operations, independently of the key bit's value. Due to this fact, the Montgomery kP-algorithm is in the literature referred to as resistant against some SCA attacks, such as SPA and simple electromagnetic analysis [3]. The algorithm consist of three parts. The first part is the initialization phase (see line 1 in Algorithm 1). During this phase, the conversion of affine EC point coordinates to Lopez-Dahab projective coordinates takes place and the MSB of the scalar k, the key bit $k_{l-1} = 1$, is processed. The second part corresponds to the processing of all remaining bits of the scalar k, i.e. bits $k_{l-2}, k_{l-3}, \ldots, k_1, k_0$ (see lines 2 to 10 in Algorithm 1). This is the main loop of the algorithm. Depending on the value of the key bit k_i the operations in lines 4 and 5 or the operations in lines 7 and 8 are executed. Both possible loop iterations, i.e. in case $k_i = 1$ and in case $k_i = 0$, are executed in exactly the same way. In both cases 6 multiplications,[1] 5 squarings, 3 additions and 6 register write operations are performed. The two loops only differ in the interchangeable use of the registers as input and output parameters. The third part of Algorithm 1 corresponds to the conversion of the multiplicaiton result $kP = (X, Z)$ back to affine coordinates (see lines 11 and 12).

[1] For example if the product $X_1 X_2 T Z_2$ in line 4 is calculated as $X_1 X_2 T Z_2 = (X_1 Z_2) \cdot (X_2 T)$, this calculation corresponds to only one multiplication since the products $X_1 \cdot Z_2$ and $X_2 \cdot T$ are already calculated.

2.1 Initialization Phase as Loop Iteration

In [4] the initialization phase of Algorithm 1 is simplified. Only the values given in (1) are assigned to the registers and no calculations are performed in this phase.

$$X_1 \leftarrow 1, Z_1 \leftarrow 0, X_2 \leftarrow x, Z_2 \leftarrow 1. \tag{1}$$

Then, the first iteration of the main loop is executed according to Algorithm 1, but for the MSB $k_{l-1} = 1$. Thus, the initialization phase in Algorithm 1 is performed as a regular loop. After processing key bit k_{l-1}, the registers have the following values, which are the same as those shown in line 1 of Algorithm 1:

$$X_1 \leftarrow x, Z_1 \leftarrow 1, X_2 \leftarrow x^4 + b, Z_2 \leftarrow x^2. \tag{2}$$

The purpose of this modification was to avoid the design of any additional modules, eventually needed for the calculations performed during the initialization phase of the algorithm. Recent publications such as [5] and [6] also implement the initialization phase of the Montgomery kP-algorithm in this way, i.e. as a regular loop with special inputs.

2.2 Implementation of the Montgomery *kP*-Algorithm and SCA

A lot of research has been done on efficient implementations of the Montgomery kP-algorithm. A possible way of achieving efficiency is through the parallel execution of the operations in the algorithm. [7], [5] and [8] presented efficient implementations of the Montgomery kP-algorithm based on architectures that consist of one multiplier only. In these implementations the arithmetic and register write operations are performed in parallel to the multiplications during the executions of the main loop. In this case, the execution time of one loop iteration is defined by the time needed for performing all 6 multiplications in the loop. This is the minimum execution time for one iteration of the loop.

The focus of many research publications is only on the efficiency of the algorithm's implementation, while resistance against SCA is not considered (for example [7], [5] and [6]). Other papers discuss only the resistance of the Montgomery kP-algorithm against SCA attacks, for example [9]. The resistance against timing, simple power analysis and simple electromagnetic analysis attacks is claimed based on the fact that the algorithm performs the same type, sequence and number of operations on every iteration, independent of the key bit value [3]. Implementations resistant to SPA attacks can still be attacked using differential power analysis (DPA). The randomization of the key k or of the EC projective coordinates, as well as blinding of the EC point P [10] are well known countermeasures against DPA attacks.

In the following section, we show that the value of k_{l-2} can be extracted through SPA if the Montgomery kP-algorithm is implemented using Lopez-Dahab projective coordinates and if no special countermeasures have been implemented. In section 4 we show how we modified Algorithm 1 to avoid the easy extraction of key bit k_{l-2} through SPA.

3 Vulnerabilities due to the Initialization Phase

In line 1 of Algorithm 1 the registers X_1, Z_1, X_2 and Z_2 are initialized. The registers are used with these initial values as inputs for the first iteration of the algorithm's main loop, i.e. for the processing of key bit k_{l-2}. Register Z_1 is initialized with the value 1. This means that for the processing of k_{l-2}, all operations performed with register Z_1 are operations performed with an operand with value 1:

if $k_{l-2} = 1$

$$\begin{aligned} &T \leftarrow 1, Z_1 \leftarrow (X_1Z_2 + X_2 \cdot 1)^2, X_1 \leftarrow xZ_1 + (X_1Z_2)(X_2 \cdot 1), \\ &T \leftarrow X_2, X_2 \leftarrow (X_2^2)^2 + b(Z_2^2)^2, Z_2 \leftarrow T^2Z_2^2. \end{aligned} \tag{3}$$

if $k_{l-2} = 0$

$$\begin{aligned} &T \leftarrow Z_2, Z_2 \leftarrow (X_2 \cdot 1 + X_1Z_2)^2, X_2 \leftarrow xZ_2 + (X_1T)(X_2 \cdot 1), \\ &T \leftarrow X_1, X_1 \leftarrow (X_1^2)^2 + b(1^2)^2, Z_1 \leftarrow T^2 \cdot 1^2. \end{aligned} \tag{4}$$

This fact has the following consequences regarding the processing of k_{l-2}:

- Any multiplication performed with $Z_1 = 1$ as operand[2] will result in the value of the other operand.
- Any squaring operation performed with $Z_1 = 1$ as input will result in 1.
- The power consumption of such operations is significantly low in comparison to the power consumed by operations performed using operands with values higher than 1.

Thus, the power profile of the processing of k_{l-2} differs significantly from the power profile of the processing of all other key bits. Moreover, the power profiles in the cases $k_{l-2} = 1$ and $k_{l-2} = 0$ differ significantly from each other. Thus, the value of k_{l-2} can be extracted through SPA.

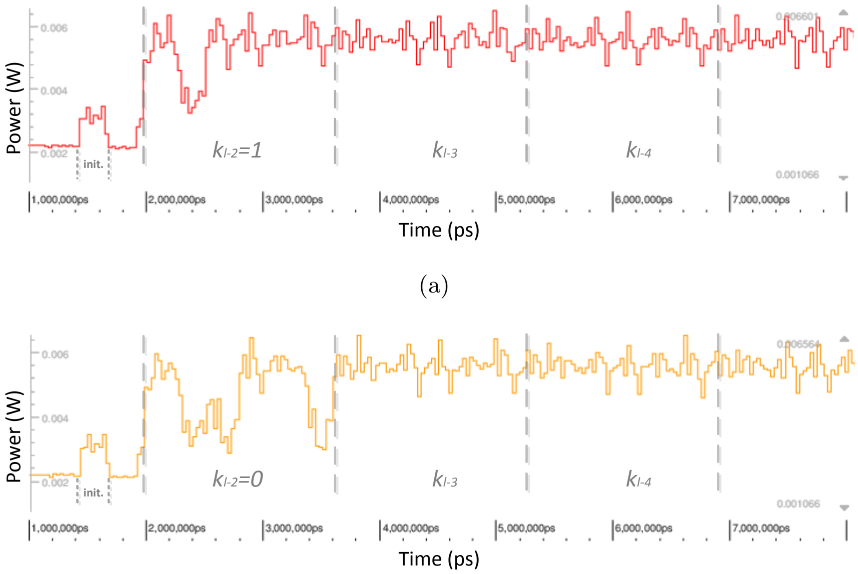
3.1 Easy Extraction of the Key Bit k_{l-2}

In the first loop iteration of Algorithm 1, a different amount of operations using register $Z_1 = 1$ as operand are performed depending on the value of k_{l-2} (compare (3) and (4)). If $k_{l-2} = 1$, register T is overwritten with $Z_1 = 1$ and only one multiplication uses $Z_1 = 1$ as operand. If $k_{l-2} = 0$, two squarings and three multiplications are performed using $Z_1 = 1$ as operand. This means that the power profile of the processing of k_{l-2} is different in case $k_{l-2} = 1$ and in case $k_{l-2} = 0$. In case $k_{l-2} = 1$ the corresponding power profile should have one dip, which corresponds to the multiplication $X_2 \cdot Z_1 = X_2 \cdot 1$. In case $k_{l-2} = 0$, the corresponding power profile should have three of such dips, corresponding to $X_2 \cdot Z_1 = X_2 \cdot 1; b \cdot Z_1^4 = b \cdot 1$, and $T^2 \cdot Z_1^2 = T^2 \cdot 1$. In this context, the value

[2] Here, 1 is the integer value.

of k_{l-2} can be easily identified.

Figure 1 shows simulated PTs of an execution of the kP-operation with our implementation of the Montgomery kP-algorithm [8] using the IHP 130 nm technology [11]. Each trace is divided into slots, whereby one slot corresponds to the processing of one key bit k_i. Each simulation was made using a different key.[3] The trace in Figure 1(a) was simulated using key $k1$, whereby the value of the key bit $k1_{l-2} = 1$. The trace in Figure 1(b) was simulated using key $k2$, whereby the value of key bit $k2_{l-2} = 0$. Our simulation results were obtained using the Synopsis PrimeTime suite [12].



(a)

(b)

Fig. 1: Two PTs simulated using our implementation of the Montgomery kP-algorithm according to Algorithm 1. The trace in (a) was simulated for the point multiplication $k1{\cdot}P$ with $k1_{l-2} = 1$. Only one dip can be seen during the processing of k_{l-2} in this trace. The trace in (b) was simulated for the point multiplication $k2{\cdot}P$ with $k2_{l-2} = 0$. Three dips can be seen during the processing of k_{l-2} in this trace.

[3] $k1 = cd\ ea65f6dd\ 7a75b8b5\ 133a70d1\ f27a4d95\ 06ecfb6a\ 50ea526e\ b3d426ed$
$k2 = 93\ 919255fd\ 4359f4c2\ b67dea45\ 6ef70a54\ 5a9c44d4\ 6f7f409f\ 96cb52cc$

Figure 1(a) shows only one dip in the slot corresponding to the processing of k_{l-2}. Figure 1(b) shows three dips in the slot corresponding to the processing of k_{l-2}. Thus, it can be easily concluded that $k_{l-2} = 1$ has been processed in the first slot of the curve in Figure 1(a). The same way it is easily observable that $k_{l-2} = 0$ has been processed in the first slot of the curve in Figure 1(b). This means that the key bit k_{l-2} can be extracted through SPA.

3.2 Vulnerabilities to Other Attacks

In section 3.1 we demonstrated that the key bit k_{l-2} can be extracted with SPA. The extraction through SPA can be done for only one bit of the key, but the power profile of the processing of k_{l-2} can be helpful for the preparation of other physical attacks.

For a successful extraction of the complete key, the attacker needs to know which operands are processed in operations within a certain clock cycle, i.e. he needs knowledge about the implementation details of the algorithm's main loop. If the kP operation is implemented according to Algorithm 1, the power profile of the processing of k_{l-2} is helpful to understand the implementation details. This power profile reveals details about the implemented operation execution sequence and the time needed for processing one key bit, i.e. for performing one loop iteration. This information is very useful for preparing, for example, DPA attacks [13], template attacks [14] or fault analysis attacks. For the processing of k_{l-2}, the attacker knows exactly which data is being processed, i.e. he knows the input values for this loop iteration (see line 1 of Algorithm 1) and can easily extract the key bit value being processed (see section 3.1). Since he knows as well *how* this data is being processed, the processing of k_{l-2} can be used as a reference for creating templates for other attacks.

If the initialization phase of the Montgomery kP-algorithm is implemented according to [4], the implementation becomes even more vulnerable to other power analysis, template or fault analysis attacks. The attacker knows the value of the key bit and the input data that has been processed not only in the first loop iteration, i.e. $k_{l-1} = 1$, but also in the second loop iteration, i.e. k_{l-2}. The attacker has the processing of two bits as a reference for creating templates.

4 Countermeasure for Protecting the Key Bit k_{l-2}

To avoid the extraction of key bit k_{l-2} through SPA and to hinder the use of the processing of k_{l-2} for preparing other attacks, we suggest to process the key bit k_{l-2} outside of the main loop of Algorithm 1 using a simplified sequence of operations. Key bit k_{l-2} can be processed with a simplified operation sequence since each operation performed with an operand with value 1, i.e. each operation performed with register Z_1, can be skipped.

The initialization phase of Algorithm 1 can be simplified as well. The initialization for register $Z_1 \leftarrow 1$ (see line 1 of Algorithm 1) can be skipped since no operations will be performed using this value as an operand. This reduces the time and energy consumption needed for processing the key bits k_{l-1} and k_{l-2}.

By skipping all operations performed with operand 1 in both cases, $k_{l-2} = 1$ and $k_{l-2} = 0$, the value of k_{l-2} can also be easily extracted through SPA, because of the different number of operations performed in each case. In case $k_{l-2} = 1$, one register write operation and one multiplication can be skipped, while if $k_{l-2} = 0$, two squarings and three multiplications can be skipped. Thus, the execution time and power profiles of the processing of $k_{l-2} = 1$ and $k_{l-2} = 0$ differ significantly. To prevent the SPA in this case, the same operation flows should be performed independently of the value of k_{l-2}. Thus in case $k_{l-2} = 0$, in which two squarings and three multiplications can be skipped, both squarings and two of these multiplications should be replaced by dummy operations (all operands $\neq 1$), whose results can be ignored. In case $k_{l-2} = 1$, one dummy register write operation should be performed. The details of our modification are discussed in the rest of this section.

4.1 Shortened Initialization Phase

Since no operations using register $Z_1 = 1$ as input will be executed during the processing of k_{l-2}, the initialization of register Z_1 can be skipped. This makes the initialization phase of the algorithm shorter, consisting of only the following operations:

$$X_1 \leftarrow x, X_2 \leftarrow x^4 + b, Z_2 \leftarrow x^2. \tag{5}$$

4.2 New Sequence for Processing of Key Bit k_{l-2}

Algorithm 2 shows our modified version of Algorithm 1. The initialization phase and the processing of k_{l-2} are simplified. The operation flow for k_{l-2} (see lines 2-8) differs from the operation flow in the main loop (see lines 9-17).

Algorithm 2 Modified Montgomery algorithm for the kP-operation

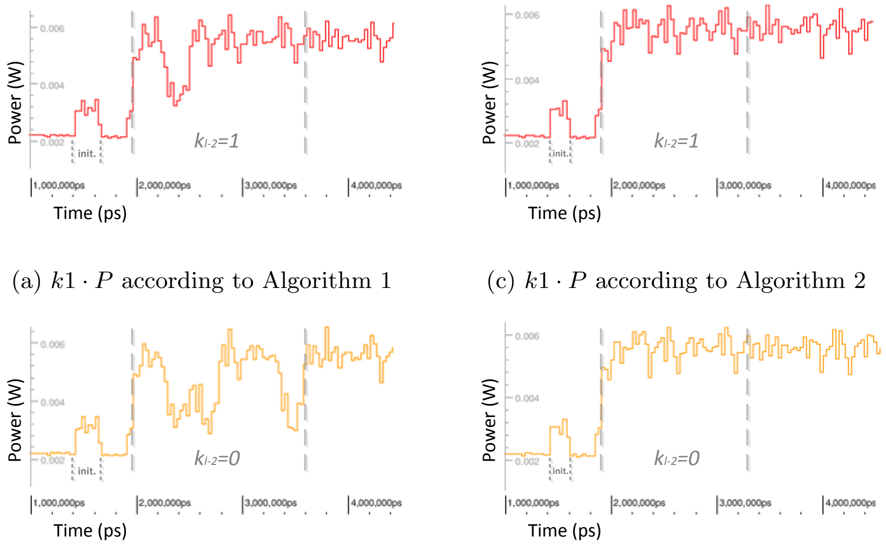
```
Input: $k = (k_{l-1}, ..., k_1, k_0)_2$ with $k_{l-1} = 1, P = (x, y) \in E(GF(2^m))$.
Output: $kP = (x_1, y_1)$.
 1: $X_1 \leftarrow x$, $X_2 \leftarrow x^4 + b$, $Z_2 \leftarrow x^2$.
 2: if $k_{l-2} = 1$ then
 3:     $T \leftarrow Z_2$, $Z_1 \leftarrow (X_1Z_2 + X_2)^2$, $X_1 \leftarrow X_1Z_2X_2 + Z_1x$,
 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:     $T \leftarrow Z_2$, $Z_2 \leftarrow (X_1Z_2 + X_2)^2$, $X_2 \leftarrow X_1X_2T + Z_2x$,
 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
 9: for $i$ from $l - 3$ downto 0 do
10:     if $k_i = 1$ then
11:         $T \leftarrow Z_1$, $Z_1 \leftarrow (X_1Z_2 + X_2Z_1)^2$, $X_1 \leftarrow xZ_1 + X_1X_2TZ_2$,
12:         $T \leftarrow X_2$, $X_2 \leftarrow X_2^4 + bZ_2^4$, $Z_2 \leftarrow T^2Z_2^2$.
13:     else
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:         $T \leftarrow X_1$, $X_1 \leftarrow X_1^4 + bZ_1^4$, $Z_1 \leftarrow T^2Z_1^2$.
16:     end if
17: end for
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 $((x_1, y_1))$.
```

The processing of key bit k_{l-2} consists of 5 multiplications, 5 squarings, 3 additions and 8 register write operations, independently of the value of k_{l-2}. Two dummy multiplications and two dummy squarings are performed for the case $k_{l-2} = 0$ (see line 8, operations $U \leftarrow bX_2^4$ and $U \leftarrow TX_2$). In case $k_{l-2} = 1$, one dummy register write operation is necessary (see line 4 the operation $T \leftarrow Z_2$). No operations are performed with an operand with integer value 1.

5 Results

With the goal of evaluating our proposed modification of the Montgomery kP-algorithm, we implemented the kP-operation according to Algorithm 1 and 2 and synthesized both using the IHP 130 nm technology. The main difference between both designs is the processing of the key bits k_{l-1} and k_{l-2}.

Figures 2(a) and 2(b) show simulated PTs of the kP-operation executed with our implementation of the Montgomery kP-algorithm according to Algorithm 1. Figures 2(c) and 2(d) show simulated PTs of the kP-operation executed with our implementation of the Montgomery kP-algorithm according to Algorithm 2. The traces in Figures 2(a) and 2(c) were simulated using key $k1$, whereby $k1_{l-2} = 1$. The traces in Figures 2(b) and 2(d) were simulated using key $k2$, whereby $k2_{l-2} = 0$. The power profiles of the slots corresponding to key bit k_{l-2} in Figures 2(c) and 2(d) look similar and show no dips in contrast to the power profiles of the first slots in Figures 2(a) and 2(b).



(a) $k1 \cdot P$ according to Algorithm 1

(c) $k1 \cdot P$ according to Algorithm 2

(b) $k2 \cdot P$ according to Algorithm 1

(d) $k2 \cdot P$ according to Algorithm 2

Fig. 2: PTs simulated using our implementations of the Montgomery kP-algorithm according to Algorithm 1 (see PTs (a) and (b)) and to Algorithm 2 (see PTs (c) and (d)). The traces in (a) and (c) correspond to the simulations made for key $k1$ with $k1_{l-2} = 1$. The traces in (b) and (d) correspond to the simulations made for key $k2$ with $k2_{l-2} = 0$. The power profiles of the processing of k_{l-2} look similar for the two traces simulated using the implementation of Algorithm 2.

Table 1 shows a comparison of both implementations. We compare the execution times and energy consumption of both implementations with special focus on the execution times and energy consumption demanded for the initialization phase of the algorithm and the processing of k_{l-2}.

Table 1: Comparison of our implementation of Algorithm 1 with our implementation of Algorithm 2.

ECC implementation			**Algorithm 1**	**Algorithm 2**
Initialization Phase	*Clock cycles*		7	5
	Energy		0.63 nJ	0.46 nJ
Processing of k_{l-2}	*Clock cycles*		54	45
	Energy	$k_{l-2} = 1$	8.60 nJ	7.45 nJ
		$k_{l-2} = 0$	7.60 nJ	7.45 nJ
Extraction of k_{l-2} *through SPA*			Yes	No
Revealed implementation details			Yes	No
kP	*Clock cycles*		12915	12904
	Energy		2.10 μJ	2.09 μJ
	Area		0.274503 mm^2	0.274843 mm^2

The time and energy consumption needed for processing the key bits k_{l-1} and k_{l-2} has been reduced in our implementation of Algorithm 2. Thus, the complete implementation of the Montgomery kP-algorithm according to Algorithm 2 consumes slightly less energy for the complete calculation of kP. Moreover, protection for the key bit value of k_{l-2} against SPA has been reached through Algorithm 2. Since key bits k_{l-1} and k_{l-2} are processed in a different way as the rest of the bits of k, our implementation of the Montgomery kP-algorithm does not give the opportunity to learn/understand implementation details of the main loop of the kP calculation. Thus, it no longer helps preparing other PA or fault analysis attacks. The modifications made for Algorithm 2 only demanded an increase in the chip area of 0.12% in comparison to our implementation of Algorithm 1.

6 Conclusions

The Montgomery kP-algorithm using Lopez-Dahab projective coordinates is considered to be an SPA resistant method for performing the kP-operation. We showed using simulated PTs that the power profile of the processing of k_{l-2} differs significantly in the cases $k_{l-2} = 1$ and $k_{l-2} = 0$. This leads to an easy extraction of the value of k_{l-2} with SPA and reveals information about the analysed implementation of the algorithm. We proposed a modification of the algorithm's initialization phase and of the processing of bit k_{l-2} as a countermeasure (see Algorithm 2). We showed that our modifications of the algorithm provide protection of the key bit k_{l-2} against SPA.
In comparison to the original implementation, the execution time of the kP-operation has been slightly reduced by 11 clock cycles with our modification of

the Montgomery kP-algorithm. Our modifications did not demand an increase on our implementation's energy consumption needed for the calculation of kP, which remained by 2.09 μJ and only demanded a very small increase of the implementation's chip area by 0.12%. Thus, we achieved to increase the robustness of our implementation against selected SCA attacks without any additional costs.

Acknowledgments. The research leading to these results has received funding from the European Commissions Horizon 2020 under grant agreement from project myAirCoach No. 643607.

References

1. Lopez, J., Dahab, R.: Fast Multiplication on Elliptic Curves over $GF(2^m)$ without Precomputation. In: Ko, .K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 316-327. Springer, Heidelberg (1999)
2. Hankerson, D., Lopez Hernandez, J., Menezes, A.: Software Implementation of Elliptic Curve Cryptography over Binary Fields. In: Ko, .K., Paar, C. (eds.) CHES 2000. LNCS, vol. 1965, pp. 1-24. Springer, Heidelberg (2000)
3. Joye, M., Yen, S.: The Montgomery Powering Ladder. In: Kaliski, B. S., Ko, .K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 291-302. Springer, Heidelberg (2002)
4. Mahdizadeh, H., Masoumi, M.: Novel Architecture for Efficient FPGA Implementation of Elliptic Curve Cryptographic Processor Over $GF(2^{163})$. In: IEEE Transactions on Very Large Scale Integration (VLSI) Systems (Volume: 21, Issue: 12), pp. 2330-2333, IEEE, (2013)
5. Liu, S., Ju, L., Cai, X., Jia, Z., Zhang, Z.: High Performance FPGA Implementation of Elliptic Curve Cryptography over Binary Fields. In: 13th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp 148-155, IEEE, (2014)
6. Li, L., Li, S.: High-Performance Pipelined Architecture of Elliptic Curve Scalar Multiplication over $GF(2^m)$. In: IEEE Transactions on Very Large Scale Integration (VLSI) Systems (Volume:PP, Issue: 99), pp 1-10, IEEE, (2015)
7. Ansari, B., Hasan, A.: High-Performance Architecture of Elliptic Curve Scalar Multiplication. In: IEEE Transactions on Computers (Volume: 57, Issue: 11), pp. 1443-1453, IEEE, (2008)
8. Alpirez Bock, E.: SCA Resistent Implementation of the Montgomery kP-Algorithm. Master Thesis, BTU Cottbus-Senftenberg, (2015)
9. Fan, J., Verbauwhede, I.: An Update Survey on Secure ECC Implementations: Attacks, Countermeasures and Cost, Cryptography and Security. In: Naccache, D. (ed.) From Theory to Applications, pp. 265-282, Springer, Heidelberg (2012)
10. Coron, J.: Resistance Against Differential Power Analysis For Elliptic Curve Cryptosystems. In: Ko, .K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 292-302. Springer, Heidelberg (1999)
11. IHP, `http://www.ihp-microelectronics.com/en/start.html`
12. Synopsis, *PrimeTime* `http://www.synopsys.com/Tools/Implementation/SignOff/Pages/PrimeTime.aspx`

13. Clavier, C., Feix, B., Gagnerot, G., Roussellet, M., Verneuil, V.: Horizontal Correlation Analysis on Exponentiation. In: Soriano, M., Qing, S., Lpez, J. (eds.) Information and Communication Security. LNCS, vol. 6476, pp. 46-61. Springer, Heidelberg (1999)
14. Chari, S., Rao, J., Rohatgi, P.: Template Attacks. In: Kaliski, B. S., Ko, .K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13-28. Springer, Heidelberg (2002)