# Fresh Re-Keying: Security against Side-Channel and Fault Attacks for Low-Cost Devices

Marcel Medwed[1], François-Xavier Standaert[2],
Johann Großschädl[3], and Francesco Regazzoni[2]

[1] Graz University of Technology, Austria.
[2] Université catholique de Louvain, Belgium.
[3] University of Luxembourg, Luxembourg.

**Abstract.** The market for RFID technology has grown rapidly over the past few years. Going along with the proliferation of RFID technology is an increasing demand for secure and privacy-preserving applications. In this context, RFID tags need to be protected against physical attacks such as Differential Power Analysis (DPA) and fault attacks. The main obstacles towards secure RFID are the extreme constraints of passive tags in terms of power consumption and silicon area, which makes the integration of countermeasures against physical attacks even more difficult than for other types of embedded systems. In this paper we propose a fresh re-keying scheme that is especially suited for challenge-response protocols such as used to authenticate tags. We evaluate the resistance of our scheme against fault and side-channel analysis, and introduce a simple architecture for VLSI implementation on RFID tags. In addition, we estimate the cost of our scheme in terms of area and execution time for various security/performance trade-offs. Our experimental results show that the proposed re-keying scheme provides better security (and does so at less cost) than other state-of-the-art countermeasures.

## 1 Introduction

Radio-Frequency Identification (RFID) is an emerging technology that enables identification of non-line-of-sight objects or subjects. Based on cheap RF microcircuits - called tags - apposed on or incorporated in the items to identify, the RFID technology is now widely deployed in our everyday life. In view of the large variety of applications that are targeted, the security and privacy that can be provided by these tags has become a primary concern in the past few years. Hence, solving these security issues has raised a need for cryptography-enabled devices with a strong constraint on area cost and power consumption.

Unfortunately, while designing private-key or public-key cryptographic computing devices that hold in a few thousands of logic gates is already challenging, the cost criteria is not the only one to be fulfilled by cryptographic RFID tags. Indeed, these devices are frequently manipulated in hostile environments in which different types of implementation (aka physical) attacks can be applied. That is, rather than targeting the cryptographic algorithms and protocols at the mathematical level, an adversary can decide to directly target the hardware to break the system. Since the mid-nineties, a large number of experiments have witnessed

the low cost nature of certain categories of physical attacks, most prominently side-channel attacks (e.g. Simple Power Analysis, Differential Power Analysis [14]) and fault attacks [1]. Unsurprisingly, the very same attacks that can break unprotected smart cards also apply to RFID (once slight adaptations of the measurement setup have been performed [10]). This is natural since both devices are generally based on the same CMOS technology. As a consequence, the need of low-cost protections against such physical attacks arises. This requirement of physical security is a particularly challenging issue since most countermeasures that have been published to increase the resistance against side-channel or fault attacks are in fact quite expensive [19]. And the situation gets even worse if one tries to prevent both threats with the same piece of hardware. On the other hand, developing and implementing security protocols with large mathematical security margins is not very relevant if the devices running these protocols do not show a similar (or at least reasonable) level of physical security.

In order to solve this problem, we describe a fresh re-keying scheme that is expected to be secure against Differential Fault Attacks and a large category of side-channel attacks (namely the standard SPA and DPA attacks that will be described in Section 2). This scheme, that is pictured in Figure 1, can be used in a challenge-response protocol for RFID or more generally for physically secure encryption. It contains an encryption function $f$ (typically, a block cipher - the AES Rijndael will be our running example) to encrypt every message block $m$ with a fresh session key $k^*$. This session key $k^*$ is obtained thanks to a function $g$ from a master key $k$ and a public nonce $r$ that is chosen by the tag. That is, one computes the session key $k^* = g_k(r)$ first and then the ciphertext $c = f_{k^*}(m)$. On the first sight, it may seem that such a scheme just shifts the problem of protecting the block cipher $f$ against physical attacks to the problem of protecting the function $g$ against the same attacks. Interestingly, we argue in this paper that it can have significant advantages both in terms of security and performance. First, and quite simply, it makes the application of Differential Fault Analysis unpractical, because the same key is never used twice to encrypt with the block cipher. Second, it allows separating the requirements for the two functions. On the one hand, $g$ has to be low-cost and easy to protect against side-channel attacks, but does not have to be cryptographically strong. On the other hand, $f$ only needs to be secure against side-channel attacks with a data complexity bounded to one single query (i.e. SPA, essentially).

Concretely, we provide a list of desired properties for the function $g$ and propose an instance that we analyze in detail. We then investigate a large design space, trading performance for different side-channel countermeasures. Relying on previous results of evaluations for protected devices, our implementation figures show that a significant level of physical security can be obtained at a reasonable cost. In particular, we quantify the computational difficulty of performing attacks based on the traditional "divide-and-conquer" strategy in which different parts of the master key $k$ are recovered separately. This complexity is shown to be prohibitive for the targeted applications. Regarding fault analysis, we discuss the protection against differential fault attacks. Simple fault attacks

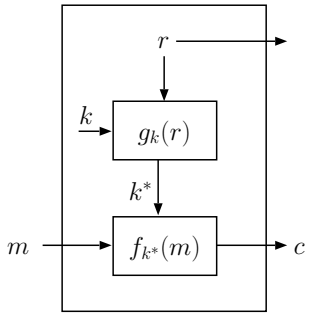



**Fig. 1.** Fresh re-keying: basic principle.

which reduce the number of rounds of a block cipher or output the key instead of the ciphertext are not in the scope of this work. They present a more general, scheme-independent threat and are usually prevented by other means like loop invariants or code signatures. Finally, we briefly discuss the resistance of our scheme against the recently introduced algebraic side-channel attacks [29]. While the exact evaluation of such advanced techniques is left as a scope for further research, we also propose solutions to prevent them.

### 1.1 Related work

A large number of countermeasures have been proposed in the literature to prevent side-channel attacks. In this section, we list a number of them with their advantages and limitations. We then argue how our proposal can be seen as a new tradeoff between security and performance issues.

First, masking (e.g. [7, 31]) is a very frequently considered solution to protect a device against side-channel attacks. It has the advantage of being quite well understood. Its main drawback is that the performance overheads can be important because of the need to compute a correction term on-the-fly, during the encryption process. Masking can be defeated by higher-order attacks [21] or because of technological issues such as glitches [18]. Overall, it is usually considered as one useful part of the solution for protecting cryptographic hardware. The permutation tables that are analyzed in [3] have quite similar properties, both in terms of performance overheads and security [27].

Next to masking, hiding is another frequently considered countermeasure. Numerous hiding schemes have been proposed in the literature, e.g. different time randomization tools and side-channel resistant logic-styles of which the goal is to have the leakage as close to constant as possible. Such logic-styles can be based on standard CMOS cell libraries (e.g. WDDL [36]), or require full-custom design (e.g. SABL [35]). Again, there is a security vs. performance tradeoff since designing full-custom hardware is more expensive (at least in development time) than using standard libraries, but the security of the latter ones is generally lower, mainly because they offer less fine tuning possibilities [16].

Closer to our present proposal, different protocol-level solutions have also been investigated. For example, the idea of regular key updates, first described in [13], has recently attracted a significant attention, as witnessed, e.g. by [22, 23]. Such re-keying schemes have the advantage of being quite formally analyzed which leads to a good evaluation of the security level they provide. On the other hand, they still rely on certain physical assumptions that must be achieved by the hardware and they can be quite inefficient when a chip has to be re-initialized regularly (which is typically the case of challenge-response authentication protocols). More precisely, and as detailed in [33], a secure initialization process for such constructions using block ciphers would require to implement a tree-based structure with up to $n$ applications of, e.g. the AES Rijndael, where $n$ is the bit-size of the initialization vector. This hardly fits to the RFID scenario.

Eventually, the use of "all-or-nothing" transforms to prevent certain classes of side-channel attacks is discussed in [15]. Here the idea is to transform the plaintexts and ciphertexts with a low-cost mapping that is easy to protect against physical avdersaries and makes the guessing strategy, exploited in most standard DPA, hardly applicable. As this proposal is quite recent, its careful security analysis is still an open problem. Interestingly, it also shifts the problem of protecting a complete cipher to the one of protecting a simpler transform. But as for re-keying schemes, the initialization and synchronization of an encryption protected with such all-or-nothing transforms can be expensive. Another drawback is the need of an additional secret shared between the two parties.

Our fresh re-keying is in fact in close connection with the idea of all-or-nothing transforms and standard re-keying schemes. Its main advantage is to propose a low-cost solution to the initialization problem (since a fresh session key is used to encrypt every block of plaintext). Also, since we apply a transform on the key, we avoid the need to share an additional secret as in the all-or-nothing transforms' case. Finally, the proposed solution is low-cost, because we only need one transform to protect the key rather than two transforms to protect the plaintext and ciphertext in the all-or-nothing case. For the rest, we share the advantages of these protocol level countermeasures. In particular, we do not need to compute correction terms during the encryption process. Also, and as will be detailed in the following sections, we can take advantage of masking and hiding to protect our re-keying transform. And because of its relatively small gate count, we can even consider using a full-custom circuit for this purpose.

Note that, because we consider the RFID scenario, this paper has a strong focus on implementation efficiency, as will be detailed in Sections 4 and 5. In particular, we show that for a similar gate count, our solution allows implementing more masking and shuffling than if one directly attempts to protect a block cipher implementation with similar countermeasures. We believe that this is an important first step in order to motivate further research on fresh re-keying schemes. In particular, our security analysis is based on an important class of practical attacks. But generalizing it towards more abstract and general models of computation and leakage (e.g. the ones surveyed in [24]) and evaluating the performance penalties that this would imply is an interesting open question.

## 2 Background

As detailed in the introduction, the countermeasure proposed in this paper mainly splits the problem of physical security in different subproblems.

Some parts of our design are only required to be protected against SPA, other parts also require DPA-resistance. Since these are all standard notions in the field of cryptographic hardware, this section only summarizes them and points towards different references for more formal definitions. Additionally, we describe the particular type of DPA attacks exploiting a standard divide-and-conquer strategy that we consider in our security analysis. Finally, we discuss how our re-keying scheme can be used in an authentication protocol.

### 2.1 SPA and DPA

In terms of side-channel resistance, the main requirement for our following protocol to be secure can be summarized as follows:

1. The function $f$ needs to be secure against SPA.
2. The function $g$ needs to be secure against both SPA and DPA.

SPA stands for Simple Power Analysis and corresponds to attacks in which an adversary directly recovers key material from the inspection of a single measurement trace (i.e. power consumption or electromagnetic radiation, typically). DPA stands for Differential Power Analysis and corresponds to more sophisticated attacks in which the leakage corresponding to different measurement traces (i.e. different plaintexts encrypted under the same key) is combined. As a matter of fact, in the absence of an efficient solution to guess the session key $k^*$ from the master key $k$, such attacks can only be applied to the function $g$ in the scheme of Figure 1. Indeed, for the block cipher $f$, every plaintext will be encrypted with a different $k^*$. For more details about such attacks, we refer to [19].

### 2.2 Divide-and-conquer strategies

Divide-and-conquer attacks such as the standard DPA detailed in [20], are attacks in which the adversary recovers small parts of a master key (also called subkeys) one by one. Most side-channel attacks published in the open literature fall under this category. In such a setting, an important feature of the adversary is the need to predict some (key-dependent) intermediate computations during the encryption process (e.g. the first round S-boxes' outputs in a block cipher). As will be detailed in Section 6.3, this is typically what is made difficult by our fresh re-keying scheme. If $g$ has good enough diffusion, it should be hard to guess the intermediate computations of $f$ in function of the master key $k$. As a result, only SPA attacks can be performed against the session key $k^*$.

### 2.3 Challenge-response protocol

In a challenge-response authentication, one party sends a challenge and the other party responds with the encrypted challenge together with some additional information. Then, the response is verified. Depending on the cases, this process can be repeated with swapped roles. Since our re-keying scheme is designed for physically secure encryption, it can be straightforwardly used in any symmetric-key challenge-response authentication. The tag simply has to implement the fresh-rekeying exactly as in Figure 1. The reader implements the same scheme, except that the portion $r$ is provided from outside by the tag.

As for the communication overhead, the distribution of the $r$ values does not necessarily imply that the number of passes in the protocol increases. In a three-pass mutual authentication protocol (as for instance described in ISO/IEC 9798-2 [11]) the $r$'s can be included in data transported during the passes. Thus, the number of passes increases at most by one, depending on who starts the protocol. Note that an important property of the fresh re-keying is that the adversary should not gain an advantage when resetting the device. That is, after each reset the tag should compute a fresh new nonce and session key, *e.g.* in a passive RFID scenario, the tag is reset any time it is taken out of the reader field.

## 3 Choice of the function $g$

In order to investigate the security of the fresh re-keying scheme of Figure 1, one first needs to determine the functions $f$ and $g$. As previously mentioned, a convenient choice for the function $f$ is a block cipher, e.g. the AES Rijndael. Hence, it remains the choice of the function $g$ that is in fact the most critical both for security and performance. In this section, we list the required properties for $g$ and select an appropriate candidate according to those properties.

### 3.1 Desired properties

The following properties for $g$ are motivated by a combination of side-channel security aspects and hardware implementation aspects.

*P1: Diffusion.* One bit of the session key $k^*$ should depend on many bits of the master key $k$. In other words, guessing one bit of the session key must be computationally difficult. This property ensures that the divide-and-conquer approach, usually applied in DPA, cannot be easily carried out.

*P2: No need for synchronization.* The function $g$ should not have a variable inner state which needs to be kept synchronous among the parties. The only inner state should be the static portion $k$ (contrary to [22, 23]).

*P3: No additional key material.* The symmetric key material which needs to be preliminary distributed among the parties and stored within the devices should not be larger than that of classical block encryption. That is, the master key $k$ should suffice to evaluate both functions $f$ and $g$ (contrary to [15]).

*P4: Little hardware overhead.* Deriving the session key in hardware must be cheaper than protecting the original circuit (i.e. the function $f$) by means of secure logic-styles and other countermeasures.

*P5: Easy to protect against SCA.* $g$ should have a suitable algebraic structure that makes its protection against SCA easier than, *e.g.* block ciphers. Combined with the previous property, it means that deriving the session key with a protected $g$ should also be lower in cost than protecting $f$.

*P6: Regularity.* If possible, the function $g$ should have a high regularity in order to facilitate its implementation in a full-custom design. This is motivated by the good security properties that the fine-tuning of such designs allows.

### 3.2 Candidate

From a cryptographic point of view the most obvious choice for $g$ would be either a hash function or an encryption function. However, they would not be useful in the present context since they are just as complex to implement and protect as the original block cipher $f$. By contrast, from an engineering point of view, a bitwise XOR function would be best. In fact, a XOR fulfills many of the above properties, but the diffusion remains extremely weak. Combining these two extremes led us to select $g$ as the following modular multiplication:

$$g : \left(\mathrm{GF}(2^8)[y]/p(y)\right)^2 \rightarrow \mathrm{GF}(2^8)[y]/p(y) : (k, r) \rightarrow k * r.$$

In the remaining of the paper, the polynomial $p(y)$ will be defined as $y^d + 1$ with $d \in \{4, 8, 16\}$. The actual choice of $d$ will be used as a parameter to improve the diffusion (i.e. *P1*), as will be discussed in Section 6.3. As for the other properties, *P2* is fulfilled because the function only depends on the public but random nonce $r$ and the secret key $k$; *P3* is fulfilled because only one master key $k$ is needed for $g$'s evaluation; finally *P4-P6* are discussed in the next section.

Note that the diffusion property of this modular multiplication significantly depends on the choice of $r$. Since it is randomly generated on-chip by the tag, it allows arguing about the physical security of the tag by showing that the diffusion is high enough on average. By contrast, on the reader side, the nonce can be generated by an adversary. Hence, the re-keying will not ensure diffusion (and physical security) on that side. Consequently, the (more expensive) reader is expected to be protected against implementation attacks by other means.

## 4 Implementation of the function $g$

In this section we discuss the implementation of $g$. We start from a general description of the multiplication algorithm, extend it to a blinded version and eventually discuss the use of secure logic for a hardware implementation.

### 4.1 Unprotected implementation

The unprotected implementation of the multiplication follows Algorithm 1, in which the complexity mainly depends on $p(y)$. Thus, the degree of this polynomial can be used to trade performance for diffusion.

For example, if $d = 16$ (resp. $d = 8$), every bit of the session key $k^*$ will depend on 64 (resp. 32) bits of the master key on average (details are in Section 6.3). Note that if $d < 16$, the multiplication is simpler but has to be applied several times to cover all the key bytes (e.g. 2 times if $d = 8$, 4 times if $d = 4$).

**Algorithm 1.** Product-scan algorithm for multiplication

**Require:** $a, b \in GF(2^8)[y]/y^d + 1$
**Ensure:** $c = a * b \in GF(2^8)[y]/y^d + 1$
1: $\rho \leftarrow$ rand()
2: $i \leftarrow 0, j \leftarrow \rho, k \leftarrow \rho, l \leftarrow 0$
3: **while** $k \neq \rho - 1 \pmod d$ **do**
4: ACCU $\leftarrow 0$
5: **for** $l = 0$ to $d - 1$ **do**
6: ACCU $\leftarrow$ ACCU $+ a_i \cdot b_j$
7: **if** $l < d$ **then**
8: $i \leftarrow i - 1 \pmod d$
9: **end if**
10: $j \leftarrow j + 1 \pmod d$
11: **end for**
12: $c_k \leftarrow$ ACCU
13: $k \leftarrow k + 1 \pmod d$
14: **end while**
15: **return** $c$

We opted for a product-scan algorithm [8], in which the result is calculated digit-wise. That is, in each iteration of the outer loop (lines 3-14), all partial products which add to the same digit of the final product are computed and accumulated. Usually, a disadvantage of this algorithm is the out-of-order processing of the operands. However, the special choice of $p(y)$ allows to overcome this problem. This choice will be justified in Section 4.4.

### 4.2 Improving $g$'s SPA/DPA resistance with shuffling

Due to the structure of the ring we are operating on, the digits of the product are independent (i.e. carry-free). This implies that the order in which the multiplication algorithm operates on the product digits can be randomized. Therefore, *shuffling* can be applied as a SCA countermeasure [19]. Shuffling has the effect that an adversary who observes a side-channel trace cannot directly infer the operations carried out in different samples (i.e. in our case: which part of the product is processed at what time). This makes the application of SPA difficult.

Shuffling also increases the data complexity of a DPA by $d^2$ [9]. Eventually, the countermeasure comes for free in our case, because only the starting index of the outer loop has to be initialized with a random value (Algorithm 1, line 1).

### 4.3 Improving $g$'s SPA/DPA resistance with blinding

Usually, DPA attacks against a multiplication algorithm target the partial products. This is because a partial product depends only on one digit of each operand, thus allowing the application of a divide-and-conquer strategy. A common countermeasure to SCA that is also applicable in our context is to use a redundant representation for the variables. Sharing a variable over $(m + 1)$ variables is referred to as $m^{th}$-order *blinding* (also called masking in the symmetric setting [31]). Blinding is a powerful countermeasure, but it is only efficient if the computational overhead due to operating on the redundant representation is small. Since addition and multiplication are distributive in our algebra, this condition is nicely respected. Algorithm 2 implements an $m^{th}$-order blinded version of the function $g$. In line 3, $m$ random blinds $b_i$ are added to $k$ before the multiplication is carried out in line 5. Afterwards, each product $b_i * r$ has to be removed again from the result in line 7. It can be easily verified that this does not change the result. However, it ensures that any adversary who wants to mount a DPA on $g$ needs to exploit the joint information of $m$ partial products, thus perform an $m^{th}$-order DPA. The number $m$ presents the second parameter in our design space for $g$. The time and space complexity of $g$ increases linearly with $m$, whereas the complexity of a DPA of $g$ increases exponentially with $m$ [2].

**Algorithm 2.** Blinded session key generation

**Require:** $k, r, b_i$ with $i = 1$ to $m$ the masking order
**Ensure:** $k^* = k * r$
1: $bk \leftarrow k$
2: **for** $i = 1$ to $m$ **do**
3: $\quad bk \leftarrow bk + b_i$
4: **end for**
5: $k^* \leftarrow bk * r$
6: **for** $i = 1$ to $m$ **do**
7: $\quad k^* \leftarrow k^* + b_i * r$
8: **end for**
9: **return** $k^*$

### 4.4 Improving $g$'s SPA/DPA resistance with protected logic-styles

Finally, the main reason why we opted for the product-scan algorithm is that it enables the use of secure logic-styles. This is because the part of the memory which holds sensitive data is small in this case. Indeed, whereas a DPA can be applied to the partial products when performing the multiplication algorithm, it is difficult to attack a byte of the final result directly, as they depend on many

bytes of both operands. Since our product-scan algorithm works only on one product byte at a time, only this byte and the corresponding GF($2^8$) multiplication have to be protected. In general, secure logic is expensive compared to standard CMOS and efficient implementations require to use it sparingly. This is typically what our proposed re-keying scheme allows. Hence, a third parameter in our design space will be the use of such protected hardware.

## 5 Global architecture

Following the algorithmic description in the previous section, we now look at concrete hardware cost and performance issues. First, we illustrate the design space of $g$ with a block diagram. Then, we present the area and cycle-count results for the different hardware design choices that we implemented.

### 5.1 Block diagram and design space for the function $g$

Figure 2 depicts a hardware architecture for $g$. The diagram contains all components necessary to generate $k^*$, except for a random number generator that we can reasonably assume to be available on the tag. The three main components of the circuit are the controller, the memory and the multiply-accumulate (MAC) unit. Determined by the use of the AES for the function $f$, the memory consists of 128-bit registers. Note that the register for $k^*$ would be shared with $f$, thus does not contribute to the size of $g$'s circuit. The actual size of the memory is directly related to the second design parameter: the blinding order. If the blinding order is zero, only two registers are needed. If $m^{th}$-order blinding is implemented, $(m+1)$ additional registers are required.

The control unit is essentially invariant throughout the design space. Changing the degree of the polynomial only changes loop constants within the controller and affects the cycles needed to carry out an operation. Also, the successive executions of an operation (as needed for $d \in \{4, 8\}$) is managed by the software. A similar statement holds when changing the blinding order. The core of the architecture is the MAC unit. It features a GF($2^8$) multiplier, a GF($2^8$) adder and an 8-bit register. Since the MAC unit is the target for secure-logic implementations, its size is crucial, as will be evaluated in the next section.

### 5.2 Implementation results and performance evaluation

We evaluated the post-synthesis area occupation for an ASIC implementation of $g$. The synthesis tool was Design Compiler 2008.09 from Synopsys, and the library was the Free Standard Cell library from Faraday Technology Corporation, using the UMC L180 CMOS technology. We used typical corner values and reasonable assumptions for the constraints. Additionally, we varied the frequency between 1 and 20 MHz, since these are reasonably boundaries for the target application (typical frequencies are in fact 6 MHz for HF tags and 1 MHz for UHF tags). After performing several runs with different constraints and optimization options, we selected the results with the smallest area.

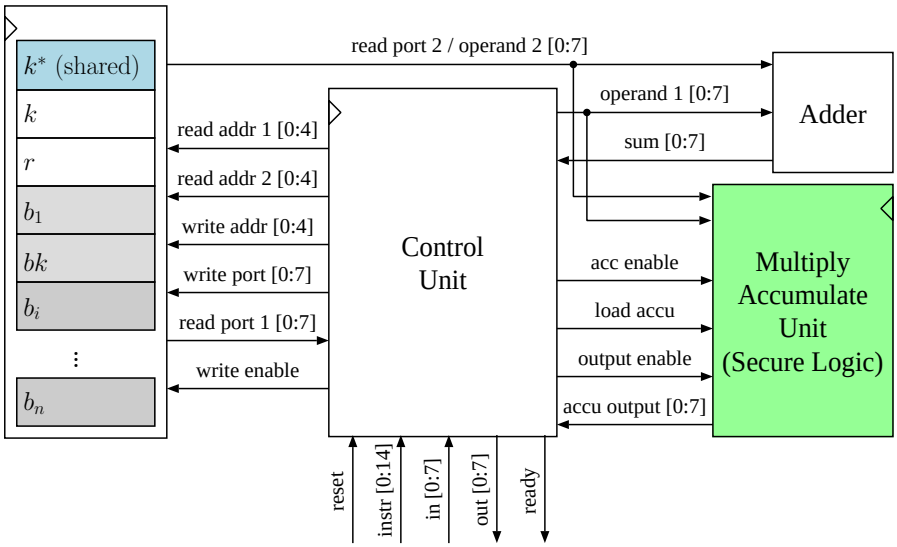



**Fig. 2.** Block diagram of the random transformation circuit

Our results are reported in Table 1 in which we find:

1. The gate equivalent estimation for $g$ using different blinding orders.
2. The cost of a MAC unit in a SCA-resistant logic-style ($g$-pMAC). We leveraged on iMDPL [26] and used a scaling factor of 18 for our estimations [12].
3. The total area ($g$ + AES) needed to protect the AES core by Feldhofer et al. [5] using our fresh re-keying scheme, with the same parameters.

**Table 1.** Post synthesis results for an ASIC implementation of $g$, $g$-pMAC and the full re-keying scheme. The protected MAC-unit is estimated with the iMDPL secure logic.

| Implementation | w/o blinding | $1^{st}$-order | $2^{nd}$-order | $3^{rd}$-order |
|---|---|---|---|---|
| function $g$ | 4.5kG | 7.3kG | 8.7kG | 10.2kG |
| $g$-pMAC | 11.7kG | 14.6kG | 16.0kG | 17.5kG |
| $g$ + AES[1] | 7.9kG | 10.7kG | 12.1kG | 13.6kG |
| $g$-pMAC + AES[1] | 15.1kG | 18.0kG | 19.4kG | 20.9kG |

[1] AES implementation taken from [5].

We compared our design to the protected circuit presented by Feldhofer et al. in [6]. Their implementation requires approximately 19.5kG and, to the best of our knowledge, is the smallest protected implementation of the AES that targets RFID applications. These results show that our fresh re-keying scheme has smaller area requirements than a direct protection of its underlying block cipher, i.e. that properties *P4, P5* and *P6* in Section 3.1 are indeed fulfilled.

More precisely, the implementation in [6] has a level of security comparable to the one of $g$ featuring either $1^{st}$-order blinding or a protected MAC. This is because their implementation protects parts with masking and other parts with secure logic. In the first case, our implementation requires approximately 8.8kG less than theirs and in the second case the difference is approximately 4.4kG. A combination of the two countermeasures would still be around 1.5kG smaller. We could also implement a blinding order of up to 5 at the same cost. Table 2 finally reports the number of clock cycles needed for the generation of a fresh session key $k^*$. Contrary to the area requirements, this number is strongly affected by the polynomial selected for the diffusion. For example, when this polynomial equals $y^{16}+1$, the time required to complete the computation is almost three times larger than when using $y^4+1$. The corresponding security levels will be discussed in the next section. As a consequence, the designer can easily tune the desired diffusion level towards the requirements of the application, even if for RFID, the throughput is generally not a strict constraint. As a comparison, the performance overhead of the the implementation Feldhofer et al. [6] is ranging between 32 and 1005 clock cycles, depending on the number of dummy instructions inserted.

**Table 2.** Cycle count for re-keying with different diffusion levels and blinding orders.

| Blinding order | $y^{16}+1$ | $y^8+1$ | $y^4+1$ |
|---|---|---|---|
| w/o blinding | 290 | 162 | 98 |
| $1^{st}$-order | 562 | 360 | 178 |
| $2^{nd}$-order | 834 | 504 | 258 |
| $3^{rd}$-order | 1160 | 648 | 338 |

## 6 Security analysis

In this section, we provide the security analysis of our re-keying scheme in different parts. We start with a note on the choice of $k$. Next, we discuss the security of the complete scheme against Differential Fault Analysis. Then, we investigate its resistance against side-channel attacks in three parts. First, we argue about the security of the function $g$ against SPA and DPA. Second, we discuss the security of the function $f$ against SPA only. Finally and most importantly, we analyze the difficulty of mounting divide-and-conquer attacks against the complete re-keying scheme. We conclude the section with some open questions related to advanced attack techniques exploiting algebraic cryptanalysis.

### 6.1 The choice of $k$

Due to the structure of the used ring, there exist zero divisors. If $k$ takes the value of a zero divisor, there exist several $r$ which lead to the same $k^*$. To avoid such collisions we have to reduce the key space $K$ to elements $k$ which are co-prime to $y^d+1$. The resulting loss of entropy can be stated as $\Delta H = 128 - H(K)$. For $d=16$, we get $\Delta H = 128 - \log_2(255 * 256^{15}) \approx 0.0056$ bits. For $d=8$, $\Delta H$ doubles and for $d=4$ it becomes four times as large respectively. In any of the three cases, the reduction of $K$ can be neglected.

### 6.2 Resistance against fault attacks

Even in its most powerful variants, Differential Fault Analysis requires at least one pair of correct and erroneous outputs to attack cryptographic algorithms [25]. From such a pair, information about the secret key can be recovered. This means that an adversary requires to encrypt the same plaintext (at least) twice with the same secret key, which is prevented by our scheme. In other words, the combination of re-keying with an initialization process using fresh $r$'s for every plaintext block provides a solid protection against Differential Fault Attacks.

### 6.3 Resistance against standard side-channel attacks

As mentioned in Section 2.1, the security of our fresh re-keying scheme requires two main properties: (1) the function $f$ needs to be secure against SPA; (2) the function $g$ needs to be secure against both SPA and DPA. In this section, we discuss how our architectural choices allow fulfilling these requirements in function of different security parameters. Then, we analyze in detail the security of the complete scheme against divide-and-conquer attacks. That is, we show that if the two previous conditions are respected, then it is computationally hard to mount such DPA attacks against the functions $f$ and $g$ taken as a whole.

**Security of $g$ against SPA and DPA.** The design space of the proposed architecture allows to deploy three well studied countermeasures against DPA attacks: shuffling, blinding and protection by secure logic. For an extensive discussion of those countermeasures see for instance [19], or [30] for a more theoretical approach. We note that in addition to the large design space, our architecture has specific advantages compared to the straightforward protection of a block cipher. For example, how to design a masking scheme for a software implementation of a block cipher that has an order higher than 3 is an open problem, as pointed out in [30]. In our case, thanks to the algebraic structure of the function $g$, such a generalization to high orders is as easy as for asymmetric encryption. Furthermore and as detailed in the previous section, the low-cost nature of $g$ makes it possible to combine several types of countermeasures against side-channel attacks at a lower cost than if they had to be applied to the original AES.

**Security of the AES against SPA.** Although not as difficult to obtain as DPA resistance, security against SPA is an important issue for the AES. In particular and since our re-keying scheme implies to run the key scheduling algorithms for every new encryption, it is important to avoid attacks such as [17]. In order to limit cost overheads, our strategy is to apply the same shuffling that is described in Section 4.2 to the 16 state bytes of our AES implementation as well as to the key expansion. As detailed in [6], this can be done with negligible overhead (we do not even need addionnal memory since we do not make use of dummy cycles).

**Security of the complete scheme against divide-and-conquer attacks.** The previous paragraphs described solutions for achieving SPA/DPA resistance for $g$ and SPA-only resistance for $f$. They show that the level of security against

these attacks can be easily tuned at the cost of some performance overheads that are at least lower than those of protecting a stand-alone AES. It now remains to argue about the security against the combined functions. That is: can an adversary directly perform a DPA on the function $f$ by guessing the master key $k$.

In order to show that such attacks are computationally hard, we argue as follows. Following [20], a DPA attack against the AES requires to guess some intermediate computation during the encryption process. In the simplest case, one bit can be guessed (e.g. one bit after the first key addition layer). In this context, the number of master key bits on which each bit of the session key $k^*$ depends only depends on the Hamming Weight (HW) of $r$. This is because every bit of $k^*$ is a sum of all bits of $k$ weighted by the bits of $r$. Since all $n$ bits of $r$ are uniformly distributed, the probability that $\mathrm{HW}(r) \leq X$ is given by:

$$P = \Pr[\mathrm{HW}(r) \leq X] = \sum_{i=0}^{X} \frac{\binom{n}{i}}{2^n}.$$

This probability can be directly related to the data complexity of an attack. That is, a small multiple of $1/P$ traces have to be collected to observe an $r$ with the desired properties. Figure 3(a) illustrates this relationship between the Hamming weight of $r$ and the number of traces that have to be collected to observe such an $r$. It can be seen that even for a Hamming weight as large as 30 the data complexity is significant with one million traces.

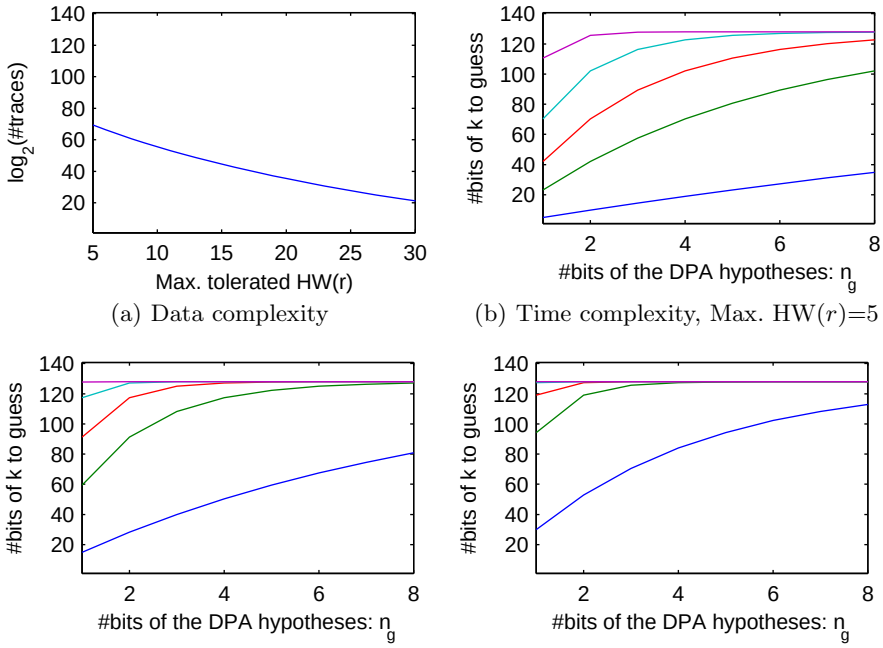
Then, in a typical DPA, there are two effects that will improve the diffusion. First, the adversary will usually not predict the key addition's output but the first S-box layer's (or even MixColumn's) output, in order to better discriminate the different key candidates. This requires to guess eight bits of session key (32 for MixColumns). This number of bits of session key to guess is denoted as $n_g$. Second, several traces corresponding to several plaintexts will generally be combined in a DPA, each one giving rise to a new random $r$. We denote at $n_t$ this number of traces. Overall, the percentage of bits on which an attack depends can be described in function of the maximum tolerated Hamming weight $X$ as:

$$1 - \left(\frac{n - X}{n}\right)^{n_t \cdot n_g},$$

Figures 3(b)-3(d) show the number of bits of $k$ to guess as a function of the hypothesis' size $n_g$. The different curves show the complexity for $n_t = 1$ (lowest curve), 5, 10, 20, and 50 (topmost curve). Since between 10 and 50 traces are usually required to recover an AES key byte with reasonable confidence in unprotected devices [32], it directly implies that the diffusion and hence time complexity will generally be sufficient to protect RFID tags.

We end this section with two specific examples to give an idea about how our countermeasure influences the data and time complexities of a divide-and-conquer attack. First we consider an AES implementation for which the attacker needs to predict $n_g = 8$ bits of the session key (a usual context). Furthermore, we assume that he needs $n_t = 10$ traces to mount a successful DPA. Even if

the attacker waits for $r$ values with a Hamming weight of 5 (as in Figure 3(b)), he needs to guess close to 128 bits of the master key to predict 10 times those 8 bits of the session key. Thus the time complexity of such an attack would be close to $2^{128}$. To get the data complexity we additionally look at the probability of observing such $r$ values. From Figure 3(a) it can be seen that they occur every $2^{70}$ traces on average. For the second example we assume that guessing $n_g = 1$ bit for $n_t = 5$ traces is enough. Even in this (unlikely) case, the data and time complexities are still prohibitive. In order to meet a more reasonable data complexity, we wait for $r$ values with Hamming weight 15. That means, that we have to observe (and acquire the power traces for) $5 \times 2^{44}$ encryptions on average. And the attack time complexity in this case would be $2^{60}$. Note that high data complexities may be hard to reach in practical side-channel attacks since even a fast measurement setup is limited to approximately 20 traces per second.




(a) Data complexity

(b) Time complexity, Max. HW($r$)=5

(c) Time complexity, Max. HW($r$)=15

(d) Time complexity, Max. HW($r$)=30

**Fig. 3.** Data vs. time complexity of a standard DPA against the combined $f$ and $g$.

### 6.4 Resistance against algebraic side-channel attacks

By clearly separating the properties of the functions $f$ and $g$, our re-keying scheme has pushed the security against side-channel attacks towards one extreme direction. On the one hand, standard side-channel attacks are prevented, as discussed in this paper. On the other hand, the function $g$ that is protected against such attacks is not strong from a cryptographic point of view. As a consequence, it appears as an interesting target for the recently introduced algebraic

side-channel attacks. Such attacks are not based on a divide-and-conquer strategy. They rather write the encryption of a plaintext into a ciphertext as a big system of low degree boolean equations in which the key bits are unknown variables. Then, the information leakage corresponding to this encryption is added to the system, also in the form of some low degree equations. As demonstrated in [29], information leakages such as Hamming weights can be sufficient to solve the system in practical time limits (minutes, typically). Quite naturally, the complexity of solving such a system of equations strongly depends on the algebraic structure of the target algorithm. For example, the AES Rijndael is more robust than the low-cost cipher PRESENT in this context [28]. Looking at our proposal for $g$, things are even worse since this function is linear. Taking an analogy with stream ciphers, one could see the side-channel leakage as a filtering function at the output of a linear number generator $g$. However, thanks to our flexible architecture, we also have positive arguments to prevent such attacks. Mainly, algebraic attacks can hardly deal with erroneous information. Hence, the shuffling that we can perform for free, both on the implementation of $f$ and $g$ will most likely make their application much harder. Because of their recent nature, we leave the exact quantification of these attacks as a scope for further research.

## 7 Conclusions

In this paper we discussed a new approach to re-keying. We explored its use as a countermeasure against physical attacks. The proposed scheme is tailored towards the security and resource needs of RFID applications.

We have evaluated the scheme's architecture and security, discussing its robustness against DFA, SPA, and DPA. The flexible configurability of the proposed circuits allows reaching a high level of security for an implementation cost that is close to the most efficient solutions available in the literature.

Open problems are in two main directions. First, it would be useful to extend the present proposal in order to protect the reader side (that is needed to be protected against physical attacks by other means than the fresh re-keying in the present proposal). Second, our analysis relies on a simple candidate of $g$ function. Investigating alternative ones, possibly trading some performance overheads for security is an interesting scope for further research. In this respect, it is worth noting that there exist simple ways to improve the diffusion properties of our scheme. As an illustration, one can generate two random nonces $r_1$ and $r_2$ and then compress the resulting $k * r_1$ and $k * r_2$ (e.g. by XORing their two halves together, producing $n/2$ bits twice) and then use the concatenation of these two compressed strings as $k^*$. Pushing such a diffusion vs. performance tradeoff further, one could also consider randomness extractors as function $g$ (that are of independent interest in leakage-resilient cryptography [4, 34]).

Summarizing, we hope that our new countermeasure and instantiation for $g$ makes an interesting case compared to traditional approaches to prevent physical attacks and raise interesting (theoretical and practical) research questions.

**Acknowledgements.** This work is funded in part by the European Commission's ECRYPT NoE phase II project, by the Belgian State IAP program P6/26 BCRYPT, by the Walloon region E.USER project and by the Austrian Government funded project ARTEUS. François-Xavier Standaert is a Research Associate of the Belgian Fund for Scientific Research (FNRS - F.R.S.).

## References

1. E. Biham, A. Shamir, *Differential Fault Analysis of Secret Key Cryptosystems*, In the proceedings of Crypto 97, Lecture Notes in Computer Science, vol 1294, pp 513-525, Santa Barbara, California, August 1997.
2. S. Chari, C.S. Jutla, J.R. Rao, P. Rohatgi, *Towards Sound Approaches to Counteract Power Analysis Attacks*, in the proceedings of Crypto 1999, Lecture Notes in Computer Science, vol 1666, pp 398-412, Santa Barbara, CA, USA, August 1999.
3. J.-S. Coron, *A New DPA Countermeasure Based on Permutation Tables*, proceedings of SCN 2008, LNCS, vol 5229, pp 278-292, Amalfi, Italy, September 2008.
4. S. Dziembowski, K. Pietrzak, *Leakage-Resilient Cryptography*, in the proceedings of FOCS 2008, pp 293-302, Washington, DC, USA, October 2008.
5. M. Feldhofer, J. Wolkerstorfer, V. Rijmen, *AES Implementation on a Grain of Sand*, IEE Proceedings on Information Security **152** (2005), no. 1, 13–20.
6. M. Feldhofer, T. Popp, *Power Analysis Resistant AES Implementation for Passive RFID Tags*, Proceedings of Austrochip 2008, October 8, 2007, Linz, Austria, October 2008, ISBN 978-3-200-01330-8, pp. 1–6.
7. L. Goubin, J. Patarin, *DES and Differential Power Analysis: the Duplication Method*, in the proceedings of CHES 1999, Lecture Notes in Computer Science, vol 1717, pp 158-172, Worcester, Massachussets, USA, August 1999.
8. D. Hankerson, A.J. Menezes, S. Vanstone, *Guide to Elliptic Curve Cryptography*, Springer, Berlin, Germany / London, UK, 2004.
9. C. Herbst, E. Oswald, S. Mangard, *An AES Smart Card Implementation Resistant to Power Analysis Attacks*, in the proceedings of ACNS 2006, Lecture Notes in Computer Science, vol 3989, pp 239-252, Singapore, June 2006.
10. M. Hutter, M. Medwed, D. Hein, J. Wolkerstorfer, *Attacking ECDSA-enabled RFID Devices*, in the proceedings of ACNS 2009, Lecture Notes in Computer Science, vol 5536, pp 519-534, Paris, France, June 2009.
11. International Organisation for Standardization (ISO), *ISO/IEC 9798-2: Information technology – Security techniques – Entity authentication – Mechanisms using symmetric encipherment algorithms*, 1999.
12. M. Kirschbaum, T. Popp, *Private Communication*, 2009.
13. P. Kocher, *Leak Resistant Cryptographic Indexed Key Update*, US Patent 6539092.
14. P. Kocher, J. Jaffe, B. Jun, *Differential Power Analysis*, Crypto 1999, LNCS, vol 1666, pp 398-412, Santa-Barbara, California, USA, August 1999.
15. R.P. McEvoy, M. Tunstall, C. Whelan, C.C. Murphy, W.P. Marnane, *All-or-Nothing Transforms as a Countermeasure to Differential Side-Channel Analysis*, Cryptology ePrint Archive, Report 2009/185, http://eprint.iacr.org/2009/185.
16. F. Macé, F.-X. Standaert, J.-J. Quisquater, *Information Theoretic Evaluation of Side-Channel Resistant Logic Styles*, CHES 2007, Lecture Notes in Computer Science, vol 4727, pp 427-442, Vienna, Austria, September 2007.
17. S. Mangard, *A Simple Power-Analysis (SPA) Attack on Implementations of the AES Key Expansion*, in the proceedings of ICISC 2002, Lecture Notes in Computer Science, vol 2587, pp 343-358, Seoul, Korea, November 2002.

18. S. Mangard, T. Popp, B.M. Gammel, *Side-Channel Leakage of Masked CMOS Gates*, in the proceedings of CT-RSA 2005, Lecture Notes in Computer Science, vol 3376, pp 351-365, San Francisco, California, USA, February 2005.
19. S. Mangard, E. Oswald, T. Popp, *Power Analysis Attacks*, Springer, 2007.
20. S. Mangard, E. Oswald, F.-X. Standaert, *One for All, All for One: Unifying Standard DPA Attacks*, Cryptology ePrint Archive, Report 2009/449.
21. T.S. Messerges, *Using Second-Order Power Analysis to Attack DPA Resistant Software*, in the proceedings of CHES 2000, Lecture Notes in Computer Science, vol 2523, pp 238-251, Worcester, Massachussets, USA, August 2000.
22. C. Petit, F.-X. Standaert, O. Pereira, T.G. Malkin, M. Yung, *A Block Cipher based PRNG Secure Against Side-Channel Key Recovery*, in the proceedings of ASIACCS 2008, pp 56-65, Tokyo, Japan, March 2008.
23. K. Pietrzak, *A Leakage-Resilient Mode of Operation*, Eurocrypt 2009, Lecture Notes in Computer Science, vol 5479, pp 462-482, Cologne, Germany, April 2009.
24. K. Pietrzak, *Provable Security for Physical Cryptography* , invited talk, in the proceedings of WEWORC 2009, Graz, Austria, July 2009.
25. G. Piret, J.-J. Quisquater, *A Differential Fault Attack Technique against SPN Structures, with Application to the AES and KHAZAD*, in the proceedings of CHES 2003, LNCS, vol 2779, pp 77-88, Cologne, Germany, September 2003.
26. T. Popp, M. Kirschbaum, T. Zefferer, S. Mangard, *Evaluation of the Masked Logic Style MDPL on a Prototype Chip*, in the proceedings of CHES 2007, Lecture Notes in Computer Science, vol 4727, pp 81-94, Vienna, Austria, September 2007.
27. E. Prouff, R.P. McEvoy, *First-Order Side-Channel Attacks on the Permutation Tables Countermeasure*, in the proceedings of CHES 2009, Lecture Notes in Computer Science, vol 5747, pp 81-96, Lausanne, September 2009.
28. M. Renauld, F.-X. Standaert, *Algebraic Side-Channel Attacks*, Cryptology ePrint Archive, Report 2009/279, http://eprint.iacr.org/2009/279.
29. M. Renauld, F.-X. Standaert, N. Veyrat-Charvillon, *Algebraic Attacks on the AES: Why Time also Matters in DPA*, in the proceedings of CHES 2009, Lecture Notes in Computer Science, vol 5747, pp 97-111, Lausanne, Switzerland, September 2009.
30. M. Rivain, E. Prouff, J. Doget, *Higher-Order Masking and Shuffling for Software Implementations of Block Ciphers*, in the proceedings of CHES 2009, Lecture Notes in Computer Science, vol 5747, pp. 171-188, Lausanne, Switzerland, September 2009.
31. K. Schramm, C. Paar, *Higher Order Masking of the AES*, CT-RSA 2006, LNCS, vol 3860, pp 208-225, San Jose, CA, USA, February 2006.
32. F.-X. Standaert, B. Gierlichs, I. Verbauwhede, *Partition vs. Comparison Side-Channel Distinguishers: an Empirical Evaluation of Statistical Tests for Univariate Side-Channel Attacks against Two Unprotected CMOS Devices*, in the proceedings of ICISC 2008, LNCS, vol 5461, pp 253-267, Seoul, Korea, December 2008.
33. F.-X. Standaert, O. Pereira, Y. Yu, J.-J. Quisquater, M. Yung, E. Oswlad, *Leakage Resilient Cryptography in Practice*, Cryptology ePrint Archive, Report 2009/341, http://eprint.iacr.org/2009/341.
34. F.-X. Standaert, *How Leaky is and Extractor?*, workshop on Provable Security against Side-Channel Attacks, Leiden, The Netherlands, February 2010.
35. K. Tiri, M. Akmal, I. Verbauwhede, *Dynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand DPA on Smart Cards*, in the proceedings of ESSCIRC 2002, pp 403-406, Florence, Italy, September 2002.
36. K. Tiri, I. Verbauwhede, *A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation*, in the proceedings of DATE 04, vol 1, pp 10246 - 10251, Paris, France, February 2004.