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. 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 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. Considering the plethora of applications that are targeted, it is little surprising that security and privacy issues related to RFID technology have become an important concern in recent years. Solving these issues has created a need for cryptography-enabled RFID tags that fulfill stringent constraints on area and power consumption.

Unfortunately, while designing secret-key or public-key cryptographic hardware on an area budget of a few thousand logic gates is challenging enough, the cost criteria is not the only one to be met by secure RFID tags. In particular, as these tags often operate in insecure or even hostile environments, they can be subject to different types of implementation (i.e. physical) attacks. This means that, instead of targeting the cryptographic algorithms and/or protocols at the mathematical level, an adversary may directly attack their implementation in

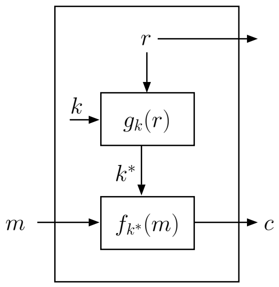

Fig. 1. Fresh re-keying: basic principle

order to break a system. A considerable number of experiments reported in the literature during the past ten years demonstrate the low-cost nature of certain classes of physical attacks, most prominently side-channel attacks (e.g. Simple Power Analysis, Differential Power Analysis [14]) and fault attacks [1]. The very same attacks that can break unprotected smart cards also apply to RFID tags (just slight adaptations of the measurement setup are necessary [10]), which is no surprise given that both types of devices are manufactured using the same or similar VLSI technology. As a consequence, the need for low-cost protections against such physical attacks emerges. The requirement of physical security is a particularly challenging issue since most published approaches to increase the resistance against side-channel or fault attacks are rather expensive [19]. Even worse, there exist only very few combined countermeasures against both types of attacks. On the other hand, developing and implementing security protocols with large mathematical security margins is not very worthwhile if the devices executing these protocols do not provide a similar (or at least reasonable) level of physical security.

In order to solve these problems, we introduce a simple re-keying scheme to protect RFID tags and other low-cost devices against Differential Fault Attacks and a large category of side-channel attacks (namely the “standard” SPA and DPA attacks as described in Section 2). This scheme, depicted in Figure 1, can be used in a challenge-response authentication protocol for RFID tags or, more generally, for physically secure encryption. It contains an encryption function f (typically, f is a block cipher; we will use the AES as example throughout this paper) to encrypt every challenge (or, more generally, every message block m) with a fresh session key k^*. This session key k^* is obtained with the help of a function g that uses a master key k and an on-tag generated public nonce r as input. That is, the tag computes the session key $k^* = g_k(r)$ first and then the ciphertext $c = f_{k^*}(m)$. At a first glance, 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. However, we argue in this paper that fresh re-keying provides significant advantages, both in terms of security and performance, over existing countermeasures. Firstly, and quite simply, it makes attacks based on Differential Fault Analysis unpractical because the same key is never used twice to encrypt a challenge or message. Secondly, it allows for separating the requirements on 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).

In the following sections of this paper, we define a number of desired properties for the function g and propose a concrete implementation that we analyze in detail. We then explore the design space for g and discuss different trade-offs between performance and resistance against side-channel attacks. Relying on previous results of evaluations of protected devices, our implementation figures show that a significant level of physical security can be obtained at reasonable cost. In particular, we assess 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. We show that the cost of such an attack is prohibitive for the targeted applications. Regarding fault analysis, we discuss the protection against differential fault attacks, but do not consider any of the simple fault attacks that reduce the number of rounds of a block cipher or output the key instead of the ciphertext. These attack scenarios represent a more general, scheme-independent threat and are commonly prevented by other means, e.g. 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 topic for further research, we sketch solutions to prevent them.

1.1 Related Work

There exists a significant literature on countermeasures to prevent side-channel attacks. In this section, we summarize the most common countermeasures and discuss their advantages and limitations. We then argue how our proposal can be seen as a new trade-off between security and performance.

First, masking (e.g. [7,31]) is a commonly applied and thus well understood technique to protect a device against side-channel attacks. Its main drawback is that the performance penalty can be significant because of the need to compute a correction term on-the-fly, during the encryption process [7]. Masking can be defeated by higher-order attacks [21] or due to hardware issues such as glitches [18]. Nonetheless, it is usually considered as an effective ingredient for the protection of cryptographic hardware. The permutation tables analyzed in [3] have quite similar properties, both in terms of performance and security [27].

Next to masking, hiding is another frequently applied countermeasure. Many hiding schemes have been proposed in the literature recently, e.g. different time randomization tools or side-channel resistant logic-styles featuring an (almost) data-independent power consumption profile. 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/cost trade-off since designing full-custom hardware is more expensive than using standard cells, but provides also more options for fine-tuning and, hence, better security [16].

Closer to our proposed solution are different protocol-level countermeasures such as regular key updates. The idea of regular key updates was first described in [13] and has recently attracted significant attention, see e.g. [22,23]. These re-keying schemes have the advantage of being formally analyzed, which allows for a good evaluation of the security level they provide. On the other hand, they still rely on certain physical assumptions that must be fulfilled by the hardware and they can be quite inefficient when a chip has to be re-initialized regularly (which is typically the case in a challenge-response setting). More precisely, as detailed in [33], a secure initialization process for such constructions using e.g. the AES would require a tree-based structure with up to n applications of the AES, where n is the size of the initialization vector. This hardly fits into the RFID realm.

The use of all-or-nothing transforms to prevent certain types of side-channel attacks is discussed in [15]. Here the idea is to transform the plain- and ciphertexts with a low-cost mapping that is easy to protect against physical adversaries and makes the guessing strategy, exploited in most standard DPA attacks, hard to apply. 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 problem 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 related to the idea of all-or-nothing transforms and the standard re-keying schemes. Its big advantage is to provide a low-cost solution to the initialization problem because a fresh session key is used to encrypt every block of plaintext. In addition, since we also apply a transform on the key, we avoid the need of sharing an additional secret as is the case with all-or-nothing transforms. 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. On the other hand, we share the advantages of these protocol-level countermeasures. In particular, we do not need to compute correction terms during the encryption process and can take advantage of masking and hiding to protect our re-keying scheme, as will be detailed in the following sections. Due to its high regularity, we can even consider a full-custom implementation.

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

2 Background

As detailed in the introduction, the countermeasure proposed in this paper splits the problem of physical security into different subproblems. Some parts of our design are only required to be protected against SPA, whereas other parts also require DPA-resistance. Since these are all standard notions in the field of cryptographic hardware, we only summarize them and point to references for a more formal treatment. In addition, we describe DPA attacks exploiting the 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 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 an attack in which an adversary directly recovers key material from the inspection of a single measurement trace (i.e. power consumption or EM 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 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 depicted in Figure 1. Indeed, for the block cipher f, every plaintext will be encrypted using 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 described 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 into this category. In such a setting, an important skill of the adversary is the ability to predict some (key-dependent) intermediate computations during the encryption process (e.g. the first round S-box 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 depending on the master key k. Therefore, only SPA attacks can be performed against the session key k^*.

2.3 Challenge-Response Protocol

In a challenge-response authentication protocol, one entity sends a challenge and the other party responds with the encrypted challenge together with some additional information. Then, the response is checked. Depending on the use case, this process can be repeated with swapped roles. As 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 as shown in Figure 1. The reader implements the same scheme, except that the nonce r is provided from outside by the tag.

Regarding the communication overhead, the transmission of the r values does not necessarily imply that the number of passes in the protocol increases. In a three-pass mutual authentication protocol (such as described in, e.g., ISO/IEC 9798-2 [11]) the r values 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 our 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 nonce r and session key (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 in Figure 1, one first needs to determine the two functions f and g. As previously mentioned, a natural choice for the function f is a block cipher, e.g. AES. Hence, it remains the choice of the function g, which is, in fact, the most critical both for security and performance. In this section, we specify 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 technique, 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 distributed in advance 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 k^* 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, this means that deriving the session key k^* 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 either be 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. In contrast, from an engineering point of view, a bitwise XOR function would be best. In fact, an XOR fulfills many of the above properties, but the diffusion remains very 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 later sections of this 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 parameter to improve the diffusion (i.e. *P1*), as will be discussed in Section 6.3. Regarding 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 to evaluate g. *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 r is randomly generated on-chip by the tag, it allows arguing about the tag's physical security by showing that the diffusion is high enough on average. By contrast, on the reader side, the nonce might be generated by an adversary. Hence, the re-keying will not ensure diffusion (and physical security) on that side. As a consequence, 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 elaborate on 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, the complexity of which mainly depends on $p(y)$. Thus, the degree of this polynomial can be used to trade off 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 (refer to Section 6.3 for details). Note that if $d < 16$, the

Algorithm 1. Product-scan algorithm for multiplication.

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

multiplication is simpler, but needs to be applied several times to cover all key bytes (e.g. twice if $d = 8$, four times if $d = 4$).

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 4–10), all partial products which add to the same digit of the final product are computed and accumulated. The main 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 individual digits of the product are independent (i.e. "carry-free"), which allows one to randomize the order in which these digits are accumulated. Therefore, *shuffling* can be applied as a side-channel countermeasure [19]. Shuffling has the effect that an adversary who observes a side-channel trace can not directly infer the operations carried out in different samples (i.e. in our case: which part of the product is processed at what time), which makes SPA difficult. In addition, shuffling also increases the data complexity of a DPA attack by d^2 [9]. Note that this countermeasure comes for free in our case because only the starting index of the outer loop has to be initialized with a random value (line 1 in Algorithm 1).

4.3 Improving g's SPA/DPA Resistance with Blinding

DPA attacks against a multiplication algorithm usually target the partial products. This is simply because a partial product depends only on one digit of each operand, which allows for applying a divide-and-conquer strategy. A common

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:     $bk \leftarrow bk + b_i$
4: end for
5: $k^* \leftarrow bk * r$
6: for $i = 1$ to $m$ do
7:     $k^* \leftarrow k^* + b_i * r$
8: end for
9: return $k^*$
```

side-channel countermeasure, which 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 only efficient when the computational overhead due to operating on such a redundant representation is small. Since addition and multiplication are distributive in our algebra, this condition is nicely satisfied.

Algorithm 2 shows 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. Both the time and space complexity of g increases linearly with m, while the complexity of a DPA on g grows exponentially with m [2].

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 because it facilitates the use of secure logic-styles. More precisely, the part of the memory that holds sensitive data is small when multiplying operands according to the product-scan method. While a DPA can be performed on the partial products during the execution of the multiplication, it is difficult to attack a byte of the final result directly, as they depend on many bytes of both operands. Since the product-scan algorithm operates only on one product-byte at a time, only this byte and the corresponding multiplication in $GF(2^8)$ have to be protected. In general, secure logic is expensive compared to standard CMOS; thus it should be used sparingly. However, this is exactly what our proposed re-keying scheme allows a designer to achieve. A third parameter in our design space will be the use of such protected hardware and the resulting impact on area.

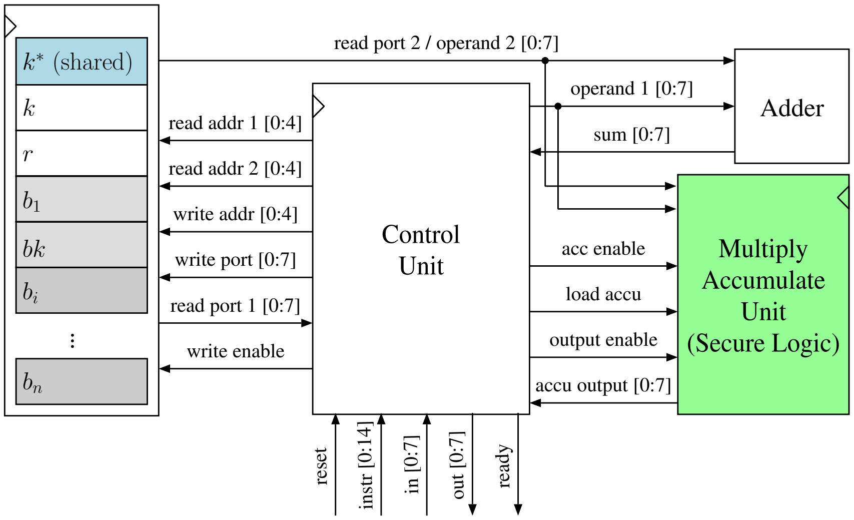

Fig. 2. Block diagram of the random transformation circuit

5 Global Architecture

Following the algorithmic description in the previous section, we now focus on concrete hardware cost and performance issues. We investigate the design space of g and present the results of different hardware implementations.

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 a random number generator, which we can reasonably assume to be available on the tag. The three main components of the architecture 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 holding k^* would be shared with f, hence it does not contribute to the cost 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 basically invariant across the design space. Changing the degree of the polynomial only changes loop constants within the controller and affects the clock cycles needed to carry out an operation. Also the successive executions of an operation (such as needed for $d \in \{4, 8\}$) are managed by the controller. A similar statement holds when changing the blinding order. The core of the architecture is the MAC unit featuring a GF(2^8) multiplier, a GF(2^8) adder, and an 8-bit register. Since the MAC unit targeted for implementation in secure logic, its size is crucial, as will be analyzed in the next section.

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

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

[1]AES implementation taken from [5].

5.2 Implementation Results and Performance Evaluation

We evaluated the post-synthesis silicon area of an ASIC implementation of the function g. The synthesis tool we used was the Design Compiler 2008.09 from Synopsys, and the library was the Free Standard Cell library from Faraday Technology for the UMC L180 CMOS technology. Our evaluation is based on typical corner values and reasonable assumptions for the constraints. Additionally, we varied the frequency between 1 and 20 MHz, which are reasonably boundaries for the considered application (typical frequencies are 6 MHz for HF tags and 1 MHz for UHF tags). After several synthesis runs with different constraints and optimization options, we selected the results with the smallest area. Our results are summarized in Table 1, including the following information:

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

We compared our design to the protected AES circuit presented by Feldhofer et al. in [6]. Their implementation has an area of approximately 19.5 kG and is, to the best of our knowledge, the smallest protected AES core that targets RFID applications. These results show that our fresh re-keying scheme requires less area than a direct protection of its underlying block cipher, i.e. 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 1st-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 8.8 kG less than theirs, and in the second case, the difference is approximately 4.4 kG. A combination of the two countermeasures would still be around 1.5 kG smaller. Note that we could also implement a blinding order of up to 5 at the same cost.

Table 2 presents the number of clock cycles needed for the generation of a fresh session key k^*. Contrary to the area requirements, this number is strongly determined by the polynomial selected for the diffusion. For example, when this polynomial equals $y^{16} + 1$, the time required to complete the computation is

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
1st-order	562	360	178
2nd-order	834	504	258
3rd-order	1160	648	338

almost three times longer 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 needs of the application (in RFID the throughput is generally not a strict constraint). For comparison, the performance overhead of the implementation of Feldhofer et al. [6] is between 32 and 1005 clock cycles, depending on the number of dummy instructions.

6 Security Analysis

In this section, we provide a security analysis of our re-keying scheme. 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 aim to analyze the difficulty of mounting divide-and-conquer attacks against the complete re-keying scheme. We will 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 ring we use, 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 $k \in K$ that are co-prime to the polynomial 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 $d=8$, ΔH doubles, while for $d=4$ it becomes four times as large. 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 a fresh r 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 is based on two requirements: (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 elaborate on how our architectural choices allow for fulfilling these requirements (depending on different security parameters). Thereafter, we analyze in detail the security of the complete scheme against divide-and-conquer attacks, i.e. we show that, if the two previous conditions are met, 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 proposed architecture for g allows us to deploy three well-studied countermeasures against DPA attacks, namely shuffling, blinding and protection via secure logic. For an extensive discussion of these countermeasures we refer to e.g. [19]; a more theoretical treatment can be found in [30]. We note that, in addition to the large design space, our scheme has some specific advantages compared to the straightforward protection of a block cipher. For example, designing 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, a generalization to higher orders is as easy as for asymmetric encryption. In addition, as detailed in the previous section, the low-cost nature of g allows one to combine several types of countermeasures against side-channel attacks at a lower cost than if they would be directly applied to the original AES.

Security of the AES against SPA. Although not as difficult to achieve as DPA resistance, security against SPA is crucial for the AES. In particular, and since our re-keying scheme implies to run the key scheduling algorithm for any new encryption, it is important to avoid attacks like the one in [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 little overhead (we do not even need additional memory as we do not make use of dummy cycles).

Security of the Overall Scheme against Divide-and-Conquer Attacks. The previous paragraphs described solutions for achieving SPA/DPA resistance for g and SPA 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 of the combined functions, i.e. can an attacker directly perform a DPA on the function f by guessing the master key k?

In order to evidence that such attacks are computationally hard, we argue in the following way. According to [20], a DPA attack against the AES requires guessing some intermediate computation during the encryption process. In the simplest case, one bit may 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

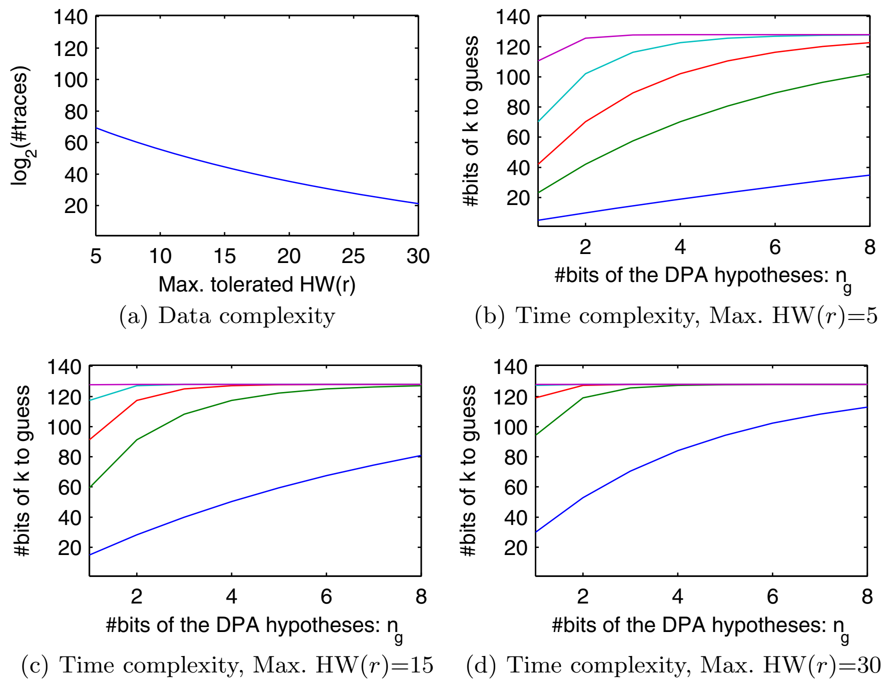

Fig. 3. Data vs. time complexity of a standard DPA against the combined f and g

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

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

This probability can be directly related to the data complexity of a concrete 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 as one million traces would be required.

In a typical DPA, there are two effects that improve the diffusion. First, the adversary will usually not predict the output of the key addition, but rather the output of the first S-box layer (or even MixColumns), in order to better distinguish between the different key candidates. This requires to guess 8 bits of the session key (or 32 for MixColumns); we denote the number of session key bits to guess 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 this number of traces as n_t. Overall, the percentage of bits on which a

DPA attack depends can be described as a function of the maximally tolerated Hamming weight X as follows:

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

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 subsection with two basic examples to illustrate 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 quantity). 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 almost 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 assess the data complexity, we additionally look at the probability of observing such r values. In 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 arrive at a more reasonable data complexity, we wait for r values with Hamming weight 15. This means that we have to observe (and acquire the traces for) $5 \cdot 2^{44}$ encryptions on average. The time complexity in this case would be 2^{60}. Note that large data complexities may be hard to cope with in practical side-channel attacks as even an effective measurement setup is limited to approximately 20 traces per second.

6.4 Resistance against Algebraic Side-Channel Attacks

By clearly separating the properties of the functions f and g, our proposed re-keying scheme has pushed the security against side-channel attacks towards an extreme direction. On the one hand, standard side-channel attacks are thwarted (as discussed in this paper). On the other hand, the function g that is protected against such attacks is not very strong from a cryptographic point of view. As a consequence, it appears to be an interesting target for the recently introduced algebraic side-channel attacks [28,29]. These attacks are not based on a divide-and-conquer strategy; they rather interpret 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 low-degree equations. As demonstrated in [29], leaking information about the Hamming weights may 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 is more robust than the low-cost cipher PRESENT in this context [28].

Looking at our proposal for g, the situation is 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 can also offer positive arguments to prevent such attacks. Most importantly, algebraic attacks can hardly deal with erroneous information. Hence, the shuffling that we can perform for free on the implementation of both f and g will most likely make mounting these attacks much harder. Because of their recent nature, we leave the exact quantification of algebraic side-channel attacks as a scope for further research.

7 Conclusions

In this paper, we discussed a new approach for re-keying and explored its use as a countermeasure against physical attacks. The proposed scheme is tailored to the security requirements and resource constraints of RFID applications. We evaluated the architecture and security of the scheme, including its robustness against DFA, SPA, and DPA. The flexibility and configurability of the proposed architecture allows for reaching a high level of security at an area 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 (which 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 for the function g. Investigating alternative ones, possibly trading some performance overhead for security, is a promising topic for future research. In this context, 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 XOR-ing the two halves together, producing $n/2$ bits twice), and use the concatenation of the two compressed strings as k^*. Pushing such a diffusion/performance trade-off even further, one could also consider randomness extractors as function g (which 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 raises interesting (theoretical and practical) research questions.

Acknowledgements. This work was funded in part by the European Commission's ECRYPT II Network of Excellence, 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. Biham, E., Shamir, A.: Differential Fault Analysis of Secret Key Cryptosystems. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 513–525. Springer, Heidelberg (1997)
2. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards Sound Approaches to Counteract Power Analysis Attacks. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 398–412. Springer, Heidelberg (1999)
3. Coron, J.-S.: A New DPA Countermeasure Based on Permutation Tables. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 278–292. Springer, Heidelberg (2008)
4. Dziembowski, S., Pietrzak, K.: Leakage-Resilient Cryptography. In: Proceedings of FOCS 2008, Washington, DC, USA, October 2008, pp. 293–302 (2008)
5. Feldhofer, M., Wolkerstorfer, J., Rijmen, V.: AES Implementation on a Grain of Sand. IEE Proceedings on Information Security 152(1), 13–20 (2005)
6. Feldhofer, M., Popp, T.: Power Analysis Resistant AES Implementation for Passive RFID Tags. In: Proceedings of Austrochip 2008, Linz, Austria, October 8, 2007, pp. 1–6 (October 2008), ISBN 978-3-200-01330-8
7. Goubin, L., Patarin, J.: DES and Differential Power Analysis: the Duplication Method. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 158–172. Springer, Heidelberg (1999)
8. Hankerson, D., Menezes, A.J., Vanstone, S.: Guide to Elliptic Curve Cryptography. Springer, Berlin (2004)
9. Herbst, C., Oswald, E., Mangard, S.: An AES Smart Card Implementation Resistant to Power Analysis Attacks. In: Zhou, J., Yung, M., Bao, F. (eds.) ACNS 2006. LNCS, vol. 3989, pp. 239–252. Springer, Heidelberg (2006)
10. Hutter, M., Medwed, M., Hein, D., Wolkerstorfer, J.: Attacking ECDSA-enabled RFID Devices. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 519–534. Springer, Heidelberg (2009)
11. International Organisation for Standardization (ISO), ISO/IEC 9798-2: Information technology – Security techniques – Entity authentication – Mechanisms using symmetric encipherment algorithms (1999)
12. Kirschbaum, M., Popp, T.: Private Communication (2009)
13. Kocher, P.: Leak Resistant Cryptographic Indexed Key Update, US Patent 6539092
14. Kocher, P., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 398–412. Springer, Heidelberg (1999)
15. McEvoy, R.P., Tunstall, M., Whelan, C., Murphy, C.C., Marnane, W.P.: 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. Macé, F., Standaert, F.-X., Quisquater, J.-J.: Information Theoretic Evaluation of Side-Channel Resistant Logic Styles. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 427–442. Springer, Heidelberg (2007)
17. Mangard, S.: A Simple Power-Analysis (SPA) Attack on Implementations of the AES Key Expansion. In: Lee, P.J., Lim, C.H. (eds.) ICISC 2002. LNCS, vol. 2587, pp. 343–358. Springer, Heidelberg (2003)
18. Mangard, S., Popp, T., Gammel, B.M.: Side-Channel Leakage of Masked CMOS Gates. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 351–365. Springer, Heidelberg (2005)
19. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks. Springer, Heidelberg (2007)

20. Mangard, S., Oswald, E., Standaert, F.-X.: One for All, All for One: Unifying Standard DPA Attacks, Cryptology ePrint Archive, Report 2009/449 (2009)
21. Messerges, T.S.: Using Second-Order Power Analysis to Attack DPA Resistant Software. In: Paar, C., Koç, Ç.K. (eds.) CHES 2000. LNCS, vol. 1965, pp. 238–251. Springer, Heidelberg (2000)
22. Petit, C., Standaert, F.-X., Pereira, O., Malkin, T.G., Yung, M.: A Block Cipher based PRNG Secure Against Side-Channel Key Recovery. In: The Proceedings of ASIACCS 2008, Tokyo, Japan, March 2008, pp. 56–65 (2008)
23. Pietrzak, K.: A Leakage-Resilient Mode of Operation. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 462–482. Springer, Heidelberg (2009)
24. Pietrzak, K.: Provable Security for Physical Cryptography. In: The Proceedings of WEWORC 2009, Graz, Austria (July 2009) (invited talk)
25. Piret, G., Quisquater, J.-J.: A Differential Fault Attack Technique against SPN Structures, with Application to the AES and KHAZAD. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 77–88. Springer, Heidelberg (2003)
26. Popp, T., Kirschbaum, M., Zefferer, T., Mangard, S.: Evaluation of the Masked Logic Style MDPL on a Prototype Chip. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 81–94. Springer, Heidelberg (2007)
27. Prouff, E., McEvoy, R.P.: First-Order Side-Channel Attacks on the Permutation Tables Countermeasure. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 81–96. Springer, Heidelberg (2009)
28. Renauld, M., Standaert, F.-X.: Algebraic Side-Channel Attacks, Cryptology ePrint Archive, Report 2009/279, `http://eprint.iacr.org/2009/279`
29. Renauld, M., Standaert, F.-X., Veyrat-Charvillon, N.: Algebraic Attacks on the AES: Why Time also Matters in DPA. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 97–111. Springer, Heidelberg (2009)
30. Rivain, M., Prouff, E., Doget, J.: Higher-Order Masking and Shuffling for Software Implementations of Block Ciphers. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 171–188. Springer, Heidelberg (2009)
31. Schramm, K., Paar, C.: Higher Order Masking of the AES. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 208–225. Springer, Heidelberg (2006)
32. Standaert, F.-X., Gierlichs, B., Verbauwhede, I.: Partition vs. Comparison Side-Channel Distinguishers: an Empirical Evaluation of Statistical Tests for Univariate Side-Channel Attacks against Two Unprotected CMOS Devices. In: Lee, P.J., Cheon, J.H. (eds.) ICISC 2008. LNCS, vol. 5461, pp. 253–267. Springer, Heidelberg (2009)
33. Standaert, F.-X., Pereira, O., Yu, Y., Quisquater, J.-J., Yung, M., Oswlad, E.: Leakage Resilient Cryptography in Practice, Cryptology ePrint Archive, Report 2009/341 (2009), `http://eprint.iacr.org/2009/341`
34. Standaert, F.-X.: How Leaky is and Extractor? In: Workshop on Provable Security against Side-Channel Attacks, Leiden, The Netherlands (February 2010)
35. Tiri, K., Akmal, M., Verbauwhede, I.: Dynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand DPA on Smart Cards. In: The Proceedings of ESSCIRC 2002, Florence, Italy, September 2002, pp. 403–406 (2002)
36. Tiri, K., Verbauwhede, I.: A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation. In: The Proceedings of DATE 2004, Paris, France, February 2004, vol. 1, pp. 10246–10251 (2004)