How Far Should Theory be from Practice?
– Evaluation of a Countermeasure –

Amir Moradi and Oliver Mischke

Horst Görtz Institute for IT Security, Ruhr University Bochum, Germany
{moradi,mischke}@crypto.rub.de

Abstract. New countermeasures aiming at protecting against power analysis attacks are often proposed proving the security of the scheme given a specific leakage assumption. Besides the classical power models like Hamming weight or Hamming distance, newer schemes also focus on other dynamic power consumption like the one caused by glitches in the combinational circuits. The question arises if with the increasing downscale in process technology and the larger role of static leakage or other harder to model leakages, the pure theoretical proof of a countermeasure's security is still good practice. As a case study we take a new large ROM-based masking countermeasure recently presented at CT-RSA 2012. We evaluate the security of the scheme both under the leakage assumptions given in the original article and using a more real-world approach utilizing collision attacks. We can demonstrate that while the new construction methods of the schemes provide a higher security given the assumed leakage model, the security gain in practice is only marginal compared to the conventional large ROM scheme. This highlights the needs for a closer collaboration of the different disciplines when proposing new countermeasures to provide better security statements covering both the theoretical reasoning and the practical evaluations.

1 Introduction

Security-enabled devices like smartcards play a larger and larger role in our everyday lives. From a mathematical point of view these can easily be protected by modern ciphers which are secure in a black-box scenario where only the inputs and outputs can be observed. Unfortunately, with the discovery of side-channel attacks in the late 90s the security of a device no longer relies only on the use of a secure cryptographic algorithm, but especially on how this algorithm is implemented. In unprotected implementations sensitive information like encryption keys can be recovered by observing so called side channels.

Many different kinds of countermeasures have since been proposed either for protection of software and/or hardware platforms (see [11] for instance). While the masking countermeasures for software are relatively limited mainly to the algorithmic level, dedicated hardware circuits further allow the use of special logic styles and gate-level countermeasures. Preventing side-channel leakage in hardware is especially intricate since glitches in the circuit can cause otherwise

theoretically secure schemes to leak [12]. Because of their wide versatility the community has shown a huge interest to different aspects of masking countermeasures, e.g., [5, 6, 16, 17, 19, 20]. More recent schemes trying to take the aforementioned problem of glitches into account are schemes relying on multi-party computation, e.g., [17, 19]. Most articles dealing with side-channel countermeasures either propose a new scheme and try proving its security in theory under a given leakage model [19] or evaluating the scheme in practice [15]. Because these tasks normally require different backgrounds – math vs. engineering – there have only been few attempts at a joint approach providing theoretically proven security in addition to practical investigations.

In this work we focus on the practical evaluation of a masking scheme recently presented at CT-RSA 2012 [10]. It is based on boolean masking and large Look-Up-Tables (LUTs), and tries to avoid some known shortcomings caused by updating the mask and masked data at the same time. The idea behind the scheme is to store and update the masks in a way that the leakage caused by updating the mask and masked data are not directly related. According to the given proofs the scheme should prevent any kind of univariate leakage if the leakage characteristics of the target device fits to a Hamming distance (HD) model. Taking AES Rijndael in hardware as our case study, we compare the efficiency and side-channel leakage of this scheme with the conventional way of using global look-up-tables (GLUT) [18]. Following the guidelines in [10], we have implemented the SubBytes transformation using the target masking scheme while facing difficulties when dealing with the linear parts especially MixColumns.

Using an FPGA-based platform we practically examine and compare the side-channel leakage of an exemplary design made considering different masking schemes including the conventional one and those proposed in [10]. We show that the proposed constructions indeed increase the resistance against power analysis attacks, but only when restricting the attack to the model assumed in the original article. Analyzing the resistance of the scheme under more realistic assumptions about the leakage model shows that the new constructions are as vulnerable as the conventional one.

We should stress that our practical results do not show any shortcomings of the scheme considering the assumed leakage model. However, we show that the model taken into account when designing and proving the security of the proposed scheme does not comprehensively consider all the possible leakages which are available in practice.

In Section 2 we define the notations and explain the basics of the target masking schemes. Our design architectures and adaptions to fit to the schemes' assumptions are described in Section 3, and the corresponding evaluation results are provided in Section 4. Finally, Section 5 concludes our research.

2 Preliminaries

When using boolean masking as a side-channel countermeasure, a secret value x, which contributes to the computations of a cryptographic device, is represented

by a randomized variable z as $x \oplus \bigoplus_{i=1}^{d} r_i$, where each r_i is an independent random variable with a uniform distribution. Each of z, r_1, ...,r_d is considered as a separate share in this secret sharing scheme, which is called d^{th}-order boolean masking. Since each share contributes to the computations separately, the scheme is supposed to provide security at most against d^{th}-order power analysis attacks. In the literature there exist two distinct definitions for what is the order of an attack. Some previous work define the order via the number of different leakage points considered simultaneously mainly because of the sequential processing in software. Others define the order via the statistical moment applied. In this work we use the following definition:

- An attack which combines v different time instances – usually in v different clock cycles – of each power trace is called v-variate attack.
- The order of an attack – regardless of v – is defined by the order of the statistical moments which are considered in the attack.

For instance, a CPA [4] which combines two points of each power trace by summing them up is a bivariate 1^{st}-order attack, and a CPA which applies the squared values of each point of each trace is a univariate 2^{nd}-order attack. Those attacks where no specific statistical moment is applied, e.g., mutual information analysis (MIA) [2], are distinguished only by v like univariate or bivariate MIA [7].

The focus of this article is 1^{st}-order boolean masking by considering one random value for each secret. The main goal of different hardware implementations of these schemes is to counteract univariate attacks of any order. Therefore, we consider only these attacks in our evaluations.

While the linear operations of a cryptographic algorithm are easy to adjust to the underlying boolean masking, providing solutions to adjust the non-linear parts of different algorithms suitable either for software or hardware platforms still is not trivial. This has indeed taken huge interest by the community and several schemes and techniques have been proposed. One which is the focus of this article is Global Look-up Table (GLUT) introduced in [18]. In the following we briefly explain the scheme w.r.t to the specific case considered in [10].

Suppose that the desired non-linear part of the target algorithm is a bijection and denoted by an $n \times n$ S-box $S : \mathbb{F}_2^n \mapsto \mathbb{F}_2^n$. The secret value x represented by two shares $z = x \oplus r$ and r is mapped to a shared representation of $S(x)$, i.e., $z' = S(x) \oplus r'$ and r'. This mapping is done by a pre-computed look-up table T as $(z, r) \mapsto (z', r')$, where output mask r' is made by a deterministic function U over input mask r. Therefore, the look-up table T maps a $2n$-bit input to a $2n$-bit output and is of a size of $2n \cdot 2^{2n}$ bits. This scheme, which is a usual way of realizing a masked S-box if the table fits into the memory space, is called "conventional" scheme in the rest of this article.

The problem which is observed in [10], is a univariate leakage caused when the registers containing the look-up table input (z, r) are updated by its output (z', r'). The bit flips in the registers are represented by $(\Delta z, \Delta r) = (x \oplus S(x) \oplus r \oplus r', r \oplus r')$. Therefore, a univariate MIA or a univariate 2^{nd}-order CPA can

reveal the relation between the bit flips and x. In order to overcome this problem a new scheme, which is shortly restated below, has been introduced in [10].

In the new scheme, x is represented by two shares $z = x \oplus F(r)$ and r, where the bit length of r (denoted by $p < 2n$) is no longer equal to n, and F is a deterministic function from $\mathbb{F}_2^p$ to $\mathbb{F}_2^n$. The look-up table T^* maps a $(n+p)$-bit input (z, r) to an n-bit output $z' = S(x) \oplus F(r')$, and the output mask is computed easily as $r' = r \oplus \alpha$, where α is a non-zero p-bit constant. The table T^* needs $n \cdot 2^{n+p}$ storage bits and still is comparable to the size of T for values p close to n. In this case the register update – same scenario as before – leads to $(\Delta z, \Delta r) = (x \oplus S(x) \oplus F(r) \oplus F(r'), r \oplus r')$. The mask difference is constant, i.e., $\Delta r = \alpha$, but in order to appropriately choose the function F two constructions have been proposed in [10] (we simplified the conditions for clarity):

- $p = n+1$, $\alpha = (\{1\}, \{0\}^n)$, and $F(r) = G(r)$ if $r \in \{0\} \times \mathbb{F}_2^n$ and $F(r) = \{0\}^n$ otherwise. G is an arbitrary bijective function from $\{0\} \times \mathbb{F}_2^n$ to $\mathbb{F}_2^n$.
- $p = n + n'$, and for $r = (r_h, r_l) \in \mathbb{F}_{2^{n'}} \times \mathbb{F}_{2^n}$, $F(r_h, r_l) = G(r_h) \bullet r_l$, where $\bullet$ denotes multiplication over the finite field $\mathbb{F}_{2^n}$. G is an arbitrary injective function from $\mathbb{F}_2^{n'}$ to $\mathbb{F}_2^n - \{0\}$. The constant α is also made by an arbitrary non-zero constant $\alpha' \in \mathbb{F}_2^{n'}$ as $(\alpha', \{0\}^n)$.

Both constructions satisfy the required conditions, i.e., Δr to be constant and distribution of $F(r) \oplus F(r')$ to be uniform. However, in the first construction $F(r)$ is zero for half of its input space. In other words, in half of the cases the secret x is represented by $z = x$ leading to a very high bias in the distribution of z for a given x and uniformly distributed random r. This issue is not seen in the second construction, and the function F is uniformly distributed over $\mathbb{F}_2^n$. We refer to these two constructions as "first CT-RSA" and "second CT-RSA" schemes in the rest of this article.

3 Case Study: AES

Hereafter we consider AES Rijndael as the target algorithm, and try to realize the encryption function using the aforementioned masking scheme. The first step toward our goal is to make the masked S-box. Because of the byte-wise computations ($n = 8$) $16 \cdot 2^{16} = 1$M bits storage space is required to realize the table T of the conventional scheme. For the first CT-RSA scheme, the same amount of space i.e., $8 \cdot 2^{17} = 1$M bits, is required to make the T_1^* table. The second CT-RSA scheme also leads to an $8 \cdot 2^{16+n'}$-bit table T_2^*, which is still possible in practice for some small n'.

Therefore, the SubBytes transformation can be easily realized. However, a question arises when trying to rewrite the linear parts of the algorithm under either the first or the second CT-RSA scheme. If the key is not masked, as is assumed in [10], the implementation of AddRoundkey is straightforward. On the other hand, if the key should also be masked this would push the space requirements to $(8 \cdot 2^{26}) = 512$M-bit for the look-up table to map the xored data and key and both their masks to a masked result. This problem becomes worse

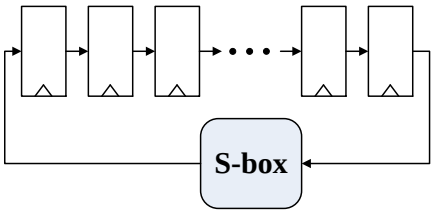

Fig. 1. The serialized design technique

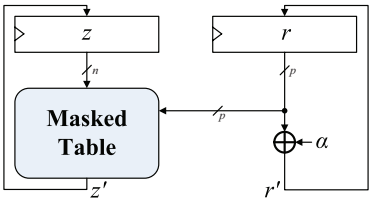

Fig. 2. CT-RSA masking architecture [10]

when trying to maintain the masking schemes while implementing mixcolums e.g., by a T-table approach.

In order to follow the scheme presented in [10] we suppose that only the SubBytes transformation is performed using the CT-RSA masking schemes, and for instance – by applying the F function – the masks are transformed to the conventional scheme to perform the rest of the encryption operations. In the following we discuss on issues arising when designing a circuit to solely perform the SubBytes transformation.

3.1 Our Design

When e.g., because of area constraint, there is only one instance of a circuit, e.g., S-box, in a design, the hardware designers usually take advantage of the serialized design methodology. In this technique, as shown in Fig. 1, a rotate shift register with an S-box circuit as the feedback function is employed. As a reference we can mention [3, 8, 9, 21] where this design methodology is used.

In either conventional or CT-RSA masking schemes the look-up tables, i.e., T or T^*, are quite large that integrating more than one table does not seem to be practical. However, the CT-RSA scheme has been designed and its security has been analyzed according to an assumption depicted in Fig. 2. It is assumed that both shares of a state byte are simultaneously replaced by the corresponding shares of the substitution value. This means that if a serialized architecture is considered for implementation, our target masking scheme does not provide the desired proven security. One solution is to use several multiplexers to select each state byte as the S-box input and save its output in the same register. This is usually not a designers' preferred choice because of its higher area overhead and slightly bigger control unit compared to the serialized one. The solution that we have applied to realize the serialized architecture and satisfy the assumption of the CT-RSA masking scheme is shown in Fig. 3. The rotate byte-wise shift register is active between each two table lookups (S-box). Therefore, the S-box output is saved at the same register which contains its input. Compared to both aforementioned architectures this design leads to time overhead since we have separated the shift and the save operations. While there are ways to improve the throughput in this scenario, we deliberately chose to keep it in this simple way since we are mainly interested in evaluating the side-channel leakage of

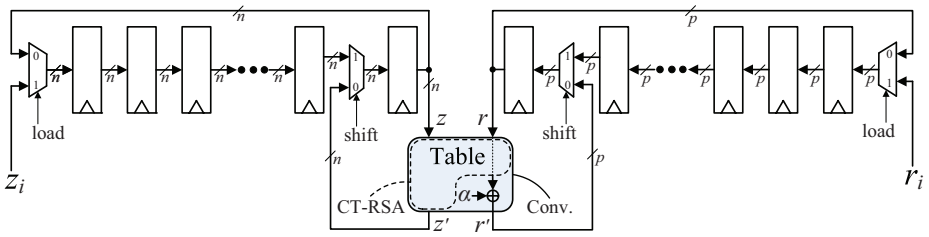

Fig. 3. Our exemplary design to examine conventional and CT-RSA masking schemes

the register updates and there is no need to risk introducing unwanted leakage sources for throughput reasons.

The target platform we selected to implement the schemes is a Virtex-5 FPGA (XC5VLX50) embedded in a SASEBO-GII board [1]. We selected three cases of masking schemes in our experiments to make the corresponding look-up tables:

- **Conventional**, look-up table T needs – as stated before – 1M bits space. The deterministic mask-update function U (see Section 2) is selected as $r' = U(r) = r^4 \oplus 56_{\mathrm{h}}$ in $\mathbb{F}_{2^8}$ and using the Rijndael irreducible polynomial.
- **First CT-RSA**, look-up table T_1^* also needs a 1M-bit space. $\alpha = 100_{\mathrm{h}}$ and the G function is randomly selected (see Appendix for a table representation).
- **Second CT-RSA**, where $n' = 1$, i.e., look-up table T_2^* also needs 1M bits space. $\alpha = 100_{\mathrm{h}}$, $G(0) = \mathrm{b2}_{\mathrm{h}}$, and $G(1) = 5\mathrm{f}_{\mathrm{h}}$.

There are a couple of different ways to implement a large look-up table in FPGAs. We selected two versions:

- **LUT**, a combination of 6-input 1-output small look-up tables (LUT6 [22]) which allows realizing a large ROM, and
- **BRAM**, a combination of 18k-bit block RAMs (RAMB18 [22]) and a few number of LUT6 which allows implementing a large RAM.

In order to make the T table in the **LUT** version, we required 1365 LUT6 instances in six depth levels for each output bit, i.e., in sum 21 840 LUT6 instances which perfectly fits to the number of available LUT6 instances in our target FPGA, i.e., 28 800. Making each the T_1^* and T_2^* tables similarly we needed 2731 LUT6 instances in seven depth levels for each output bit and in sum 21 848.

The **BRAM** version of table T needs four BRAM18 and one LUT6 for each output bit, i.e., 64 BRAM18 and 16 LUT6 for whole of the T table. Each of tables T_1^* and T_2^* also needs eight BRAM18 and three LUT6 and in sum 64 BRAM18 and 24 LUT6. We should note that 96 BRAM18 instances are available in our target FPGA, and $n' = 1$ for the second CT-RSA scheme is the only option which could fit into the available resources either in the **LUT** or **BRAM** version.

We should emphasize that we omitted using the *Architecture Wizard IP* tool of Xilinx to make the aforementioned look-up tables. Instead, we hard-instanced

the BRAM18 and LUT6 instances with our desired contents preventing any optimizations by the synthesizer. Also important to mention is the architectural difference between the **LUT** and **BRAM** versions. The tables made by LUT6 can be seen as a combinational circuit (clockless) which provides output for any value that appears at its input. However, the block RAMs need one clock cycle to provide the desired output of the given input. Therefore, our design (Fig. 3) in **LUT** version needs one clock cycle to save the table output before each shift. It means 32 clock cycles for whole of the SubBytes transformation. But one more clock cycle per state byte is required in the **BRAM** version leading to 48 clock cycles in total.

4 Practical Results

As stated before, we used a SASEBO-GII board as the evaluation platform, and implemented all our experimental designs on its target FPGA (XC5VLX50) running at a frequency of 3MHz. We also measured power consumption traces of the target FPGA using a LeCroy WP715Zi 1.5GHz oscilloscope at the sampling rate of 1GS/s. A 1Ω resistor in the VDD path, a DC blocker, a passive probe, an amplifier, and restricting the bandwidth of the oscilloscope to 20MHz helped to obtain clear and low-noise measurements.

We provided 6 design profiles made as **Conventional**, **First CT-RSA**, and **Second CT-RSA** each in both **LUT** and **BRAM** versions. Each design profile gets 16 plaintext bytes $p_{i\in\{1,\dots,16\}}$ and according to the target masking scheme makes a masked plaintext byte p'_i of each by means of 16 independent random values $r_{i\in\{1,\dots,16\}}$ (each 8-bit for the **Conventional** and 9-bit for the **CT-RSA** profiles). 16 secret key bytes $k_{i\in\{1,\dots,16\}}$, which are fix inside the design, are each XORed with the corresponding masked plaintext byte as $z_{i\in\{1,\dots,16\}} = p'_i \oplus k_i$. In 16 clock cycles z_i and r_i are serially given to the design (see Fig. 3) to completely fill the shift registers. Depending on the profile after 32 or 48 clock cycles the SubBytes transformation is completed.

We provided a clear trigger signal for the oscilloscope which indicates the start and end of the SubBytes transformation, thereby perfectly aligning the measured power traces. We also restricted the measurements to cover only the S-box computations. We fixed the number of measurements for all profiles to 1 000 000. In the experiments shown below we kept the secret key bytes fix and randomly selected the input plaintext bytes. Moreover, we made sure of the uniform distribution of internal random values r_i.

The technique we used to evaluate the side-channel leakage of these profiles is the *correlation-collision attack* [14]. This attack examines the leakage of one circuit instance that is used in different time instances. It originally considers only the first-order leakage, but according to [13] it can be adapted to use higher-order moments. Since our design profiles realize different 1^{st}-order masking schemes, we restrict our evaluations to consider only the first- and second-order univariate leakage of the profiles. We should again emphasize the goal of the **CT-RSA** pro-

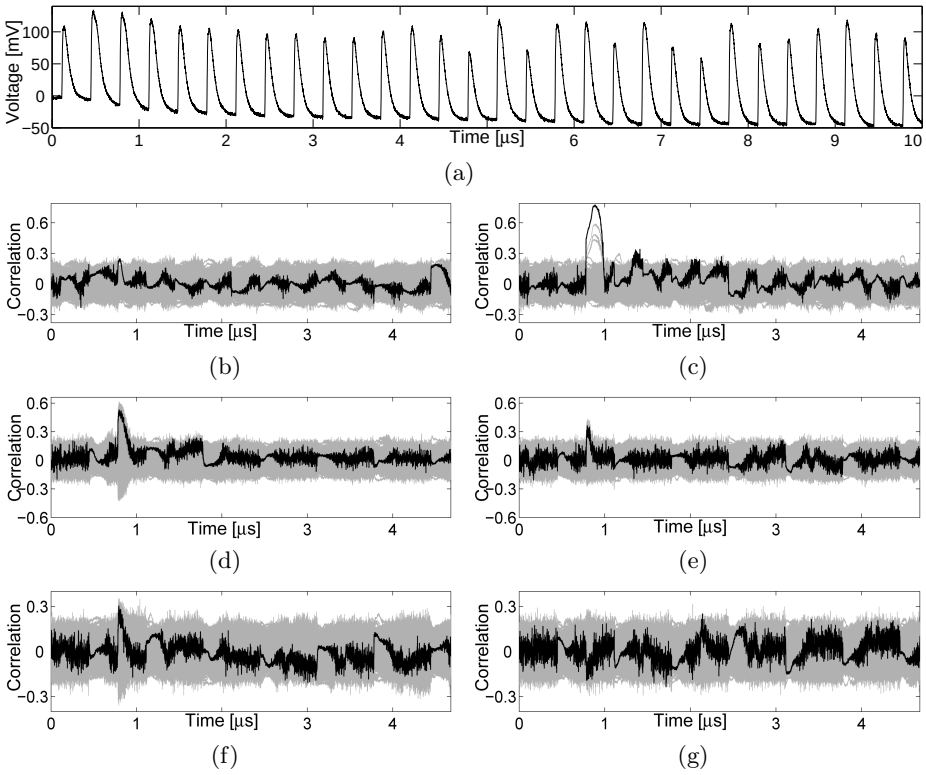

Fig. 4. LUT version, (a) sample power trace, collision attack results by register update model, (left) first-order (right) second-order: (b) and (c) **Conventional**, (d) and (e) **First CT-RSA**, (f) and (g) **Second CT-RSA** profiles, each using 1 000 000 traces

files which is preventing the univariate side-channel leakage of any order given the register update leakage model.

We start our evaluations with the **LUT** version of the **Conventional** profile. An exemplary power trace, which shows the S-box table lookup of the first few bytes, is depicted in Fig. 4(a). We consider the leakage caused by register updates when the S-box input is overwritten by its output, i.e., $v_i = (p_i \oplus k_i) \oplus S(p_i \oplus k_i)$ (not considering the masks in the formula). It is indeed the same model which the security of CT-RSA schemes are based on. In order to perform the aforementioned collision attack we need to compare the corresponding leakages of register updates of two different state bytes. We consider the second and the third state bytes, i.e., v_2 and v_3, and search for the correct (k_2, k_3) in a 2^{16} space by comparing each of the first- and second-order univariate leakages (means and variances) of corresponding parts of the measured power traces. The result of these attacks, which are not unexpected, are shown by Fig. 4(b) and Fig. 4(c). The same scenario is considered for the **First CT-RSA** profile, and performing the same attack with the same settings led to the results shown in Fig. 4(d) and

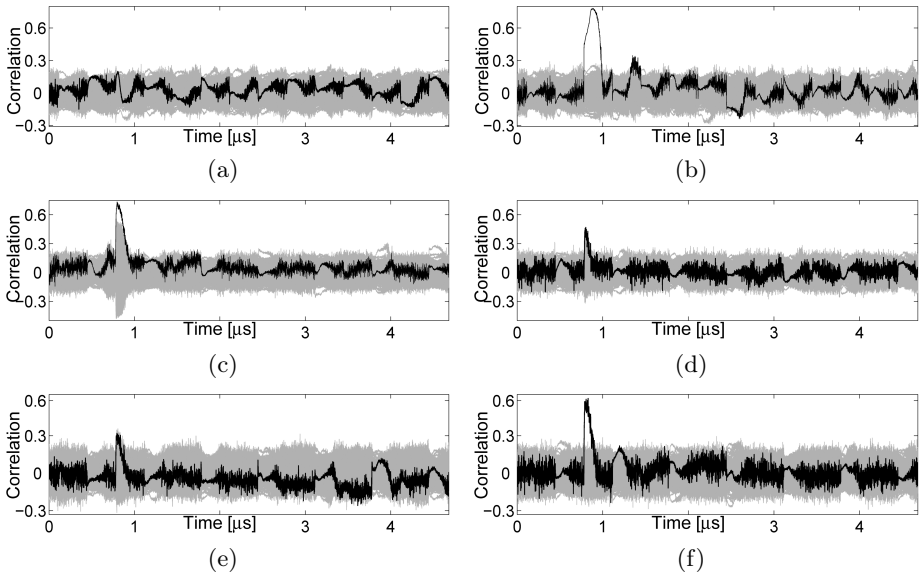

Fig. 5. LUT version, collision attack results by S-box input model, (left) first-order (right) second-order: (a) and (b) **Conventional**, (c) and (d) **First CT-RSA**, (e) and (f) **Second CT-RSA** profiles, each using 1 000 000 traces

Fig. 4(e). Comparing those attack results which are based on the second-order moments (Fig. 4(c) vs. Fig. 4(e)) shows the efficiency of the first CT-RSA scheme to counteract those attacks which use the register update model. The same holds for the **Second CT-RSA** profile, and the attack results depicted in Fig. 4(f) and Fig. 4(g) confirm the efficiency of the second CT-RSA scheme as well.

However, the register update (usually simplified by the HD model) is not the unique source of leakage in hardware. The value of e.g., S-box input or its output also affects the power consumption of the device and hence contributes in information leakage. In our designs the masked S-box input $(p_{i\in\{1,\ldots,16\}} \oplus k_i) \oplus F(r_i)$ and the mask r_i at the same time appear at the look-up table input, and the distribution of leakages which depend on the masked input and the mask is not independent of the unmasked input $p_i \oplus k_i$. Therefore, considering e.g., the S-box input a univariate attack is expected to be successful.

In order to consider such model in our attacks we take the second and the third S-box inputs, $p_2 \oplus k_2$ and $p_3 \oplus k_3$, and search for correct key difference $k_2 \oplus k_3$ amongst 2^8 candidates by comparing each first- and second-order moments of the corresponding parts of the power traces. The result of all six attacks are depicted in Fig. 5, where for each of the profiles exists a successful first- and/or second-order attack. It in fact confirms our claim that the register update is not the sole source of leakage, and in contrast to what is argued in Section 3.2 of [10] the leakage of the combinational logic – including look-up tables – cannot be separated from the leakage of the register update. Indeed, the leakage

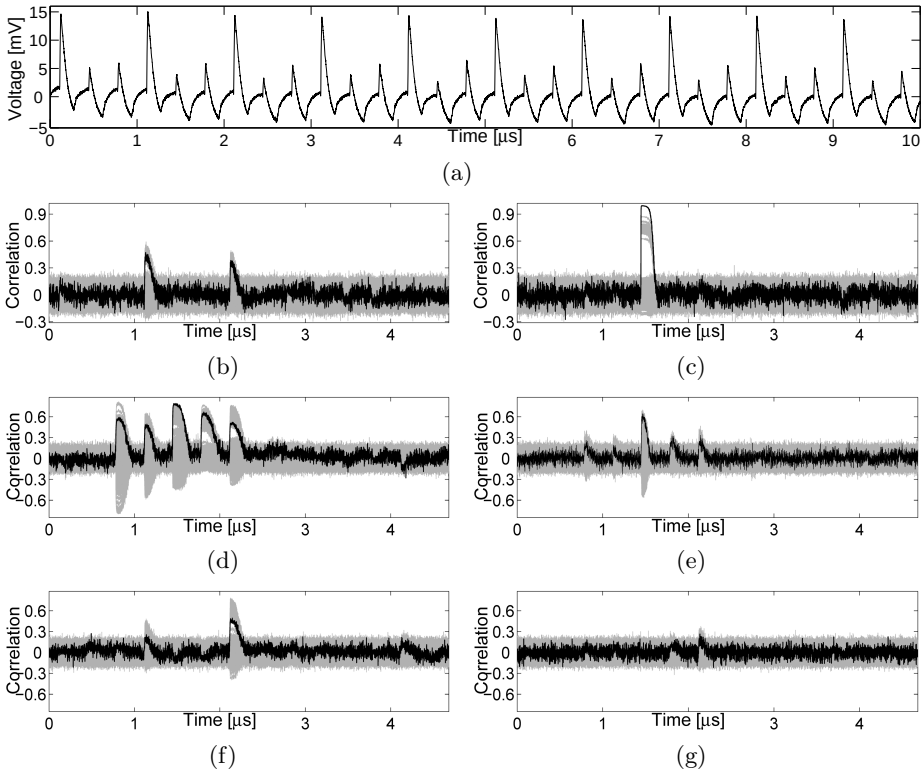

Fig. 6. BRAM version, (a) sample power trace, collision attack results by register update model, (left) first-order (right) second-order: (b) and (c) **Conventional**, (d) and (e) **First CT-RSA**, (f) and (g) **Second CT-RSA** profiles, each using 1 000 000 traces

which can be observed by currently available measurement setups is a mixture of both leakages caused by inherent low-pass filters of the device internals, PCBs, measurement tools, etc [11].

We also should stress the difference between the first-order leakage of the **First CT-RSA** and the **Second CT-RSA** profiles (Fig. 5(c) vs. Fig. 5(e)). The **First CT-RSA** profile has clear first-order leakage in contrast to the other profile. The reason behind this – as stated in Section 2 – is the F function of the first CT-RSA scheme, where the actual mask used to mask the secret, i.e., $F(r)$, is zero for half of the space of r. This results in having the S-box input – and consequently its output – unmask in the computations with the probability of 50%.

Repeating the same scenario of considering the register update as well as S-box input on profiles in the **BRAM** version led to the same results as depicted in Fig. 6 and Fig. 7. Although it needs much less power compared to the **LUT** version (Fig. 6(a) vs. Fig. 4(a)), the **First CT-RSA** and **Second CT-RSA** pro-

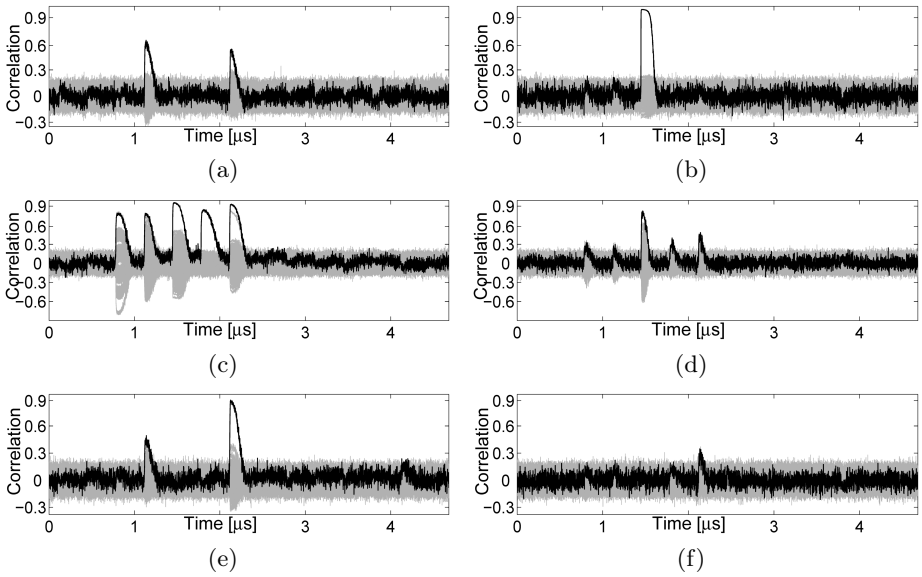

Fig. 7. BRAM version, collision attack results by S-box input model, (left) first-order (right) second-order: (a) and (b) **Conventional**, (c) and (d) **First CT-RSA**, (e) and (f) **Second CT-RSA** profiles, each using 1 000 000 traces

files also provide robustness against the attack using the register update model. However, they both – similar to the **Conventional** profile – show vulnerability against a collision attack utilizing a straightforward S-box input model.

4.1 Discussions

As mentioned before, the profiles of the **LUT** version can be seen as a huge combinational circuit which sees a masked value and the corresponding mask at its input signals. Therefore, the glitches happening inside the combinational circuit – similar to the results reported in [12] and [14] – are the main source of leakage. Their dependency to the unmasked values cause the designs to be vulnerable.

We should explain an architectural difference between the profiles of the **LUT** and the **BRAM** versions. As stated before, the profiles of the **BRAM** version need one more clock cycle per state byte compared to the **LUT** version. However, according to the results of the **BRAM** version (Fig. 7) the leakage, which depends on one S-box table lookup, appears at more than one clock cycle. Figure 7(a) shows that the input-output of a table lookup affects the power consumption not only at the clock cycle in which the block RAM is active but also at the next time when the block RAM is activated for the next table lookup (a distance of 3 clock cycles in our design profiles). Moreover, the biased distribution of the masks in the **First CT-RSA** profile becomes more problematic in the

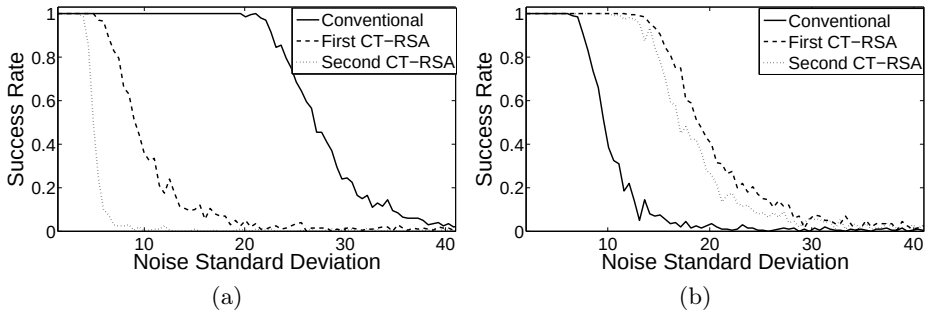

Fig. 8. Success rate of collision attacks by S-box input model in presence of noise using 1 000 000 traces, (a) **LUT** version and (b) **BRAM** version of all profiles

BRAM version, where the leakage related to a table lookup appears in five consecutive clock cycles (see Fig. 7(c)).

The internal architecture of block RAMs of our target FPGA is not publicly available, and in contrast to LUT6 it cannot be simply guessed. Therefore, we can only speculate on the actual reasons behind the strange leakage appearing in the profiles of the **BRAM** version. For instance, there exist additional input and output registers in the block RAMs which can be activated or bypassed. Also, each block RAM contains some cascading registers to be used when combining several small block RAMs to a bigger one. It is ambiguous whether all these registers still get triggered when they are bypassed in the settings. Additionally, the data and address bitwidth of each block RAM can be arbitrarily selected by the settings. This means that there exist several multiplexers and additional logics to provide all possible options. All these unclear issues prevent us from providing a certain reason for the observed leakage in the block RAMs.

At the end we compare the vulnerability of all profiles in presence of noise. Since only the attacks using the S-box input model are successful, we omitted the register update model in this evaluation. With a certain standard deviation we artificially added Gaussian random noise to the specific points of all the 1 000 000 measured power traces, and performed the same attacks as before. We repeated the noise addition and the attack 200 times for each step of the noise standard deviation, and reported the average of the attack success rate in Fig. 8. According to curves shown in Fig. 8(a), the **CT-RSA** profiles of the **LUT** version make the attacks harder compared to the **Conventional** profile. The threshold of the noise standard deviation for a successful attack on **CT-RSA** profiles is considerably lower than that of the **Conventional** one. However, a similar experiment on the profiles of the **BRAM** version shows different results (see Fig. 8(b)). The attack on the **Conventional** profile can be unsuccessful while with the same amount of noise the **CT-RSA** profiles of the **BRAM** version are still vulnerable to the aforementioned attack. The reason is most likely related to the obscure internal architecture of the block RAMs.

5 Conclusions

In this work we have implemented the scheme recently proposed in CT-RSA 2012 [10], and have evaluated its security under the given leakage assumption in the original paper as well as using an approach more close to a real-world scenario. We pointed out the practical issues when realizing this masking schemes for linear functions. Moreover, we addressed the difficulties of the GLUT technique caused by their extremely large resource consumption on FPGAs. For instance, two thirds of the available BRAMs or three quarters of all available LUTs in a Xilinx Virtex-5 LX50 are required for a single masked S-box look-up table.

Nevertheless, we could show that the newly proposed constructions indeed provide a higher level of security when only considering the register update model as the leakage source. On the other hand, our results show that this leakage assumption is still not close enough to practice even when using large ROM-style tables instead of pure combinational circuits to implement the masked S-boxes. By pointing out exploitable univariate leakages of all the design profiles we showed that just stating the security of the scheme under a register update assumption (simplified by HD) is not a valid choice in any kind of masking realization, being it in combinational logic or large ROMs.

A closer collaboration of the different fields of countermeasure creation and practical evaluations would help to increase the impact of new proposals. It indeed allows a better adaption of new schemes in real-world applications. This way the industry sector would benefit not only of a theoretical proof but would appreciate the demonstration of the consistency of the theoretical claims with practice.

Acknowledgment

The authors would like to thank Elisabeth Oswald for her valuable input while shepherding this paper and the anonymous reviewers of CHES 2012 for their constructive feedbacks.

In this project Oliver Mischke has been partially funded by the European Union, Investing in your future, European Regional Development Fund.

References

1. Side-channel Attack Standard Evaluation Board (SASEBO). Further information are available via `http://www.risec.aist.go.jp/project/sasebo/`.
2. L. Batina, B. Gierlichs, E. Prouff, M. Rivain, F.-X. Standaert, and N. Veyrat-Charvillon. Mutual Information Analysis: a Comprehensive Study. *J. Cryptology*, 24(2):269–291, 2011.
3. A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In *CHES 2007*, volume 4727 of *LNCS*, pages 450–466. Springer, 2007.

4. E. Brier, C. Clavier, and F. Olivier. Correlation Power Analysis with a Leakage Model. In *CHES 2004*, volume 3156 of *LNCS*, pages 16–29. Springer, 2004.
5. L. Genelle, E. Prouff, and M. Quisquater. Secure Multiplicative Masking of Power Functions. In *ACNS 2010*, volume 6123 of *LNCS*, pages 200–217, 2010.
6. L. Genelle, E. Prouff, and M. Quisquater. Thwarting Higher-Order Side Channel Analysis with Additive and Multiplicative Maskings. In *CHES 2011*, volume 6917 of *LNCS*, pages 240–255. Springer, 2011.
7. B. Gierlichs, L. Batina, B. Preneel, and I. Verbauwhede. Revisiting Higher-Order DPA Attacks: Multivariate Mutual Information Analysis. In *CT-RSA 2010*, volume 5985 of *LNCS*, pages 221–234. Springer, 2010.
8. J. Guo, T. Peyrin, A. Poschmann, and M. J. B. Robshaw. The LED Block Cipher. In *CHES 2011*, volume 6917 of *LNCS*, pages 326–341. Springer, 2011.
9. P. Hämäläinen, T. Alho, M. Hännikäinen, and T. D. Hämäläinen. Design and Implementation of Low-Area and Low-Power AES Encryption Hardware Core. In *DSD 2006*, pages 577–583. IEEE Computer Society, 2006.
10. H. Maghrebi, E. Prouff, S. Guilley, and J.-L. Danger. A First-Order Leak-Free Masking Countermeasure. In *CT-RSA 2012*, volume 7178 of *LNCS*, pages 156–170. Springer, 2012.
11. S. Mangard, E. Oswald, and T. Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, 2007.
12. S. Mangard, N. Pramstaller, and E. Oswald. Successfully Attacking Masked AES Hardware Implementations. In *CHES 2005*, volume 3659 of *LNCS*, pages 157–171. Springer, 2005.
13. A. Moradi. Statistical Tools Flavor Side-Channel Collision Attacks. In *EUROCRYPT 2012*, LNCS. Springer, 2012. to appear, available at the author's webpage.
14. A. Moradi, O. Mischke, and T. Eisenbarth. Correlation-Enhanced Power Analysis Collision Attack. In *CHES 2010*, volume 6225 of *LNCS*, pages 125–139. Springer, 2010.
15. A. Moradi, A. Poschmann, S. Ling, C. Paar, and H. Wang. Pushing the Limits: A Very Compact and a Threshold Implementation of AES. In *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 69–88. Springer, 2011.
16. S. Nikova, C. Rechberger, and V. Rijmen. Threshold Implementations Against Side-Channel Attacks and Glitches. In *ICICS 2006*, volume 4307 of *LNCS*, pages 529–545. Springer, 2006.
17. S. Nikova, V. Rijmen, and M. Schläffer. Secure Hardware Implementation of Nonlinear Functions in the Presence of Glitches. *J. Cryptology*, 24(2):292–321, 2011.
18. E. Prouff and M. Rivain. A Generic Method for Secure SBox Implementation. In *WISA 2007*, volume 4867 of *LNCS*, pages 227–244. Springer, 2007.
19. E. Prouff and T. Roche. Higher-Order Glitches Free Implementation of the AES Using Secure Multi-party Computation Protocols. In *CHES 2011*, volume 6917 of *LNCS*, pages 63–78. Springer, 2011.
20. M. Rivain and E. Prouff. Provably Secure Higher-Order Masking of AES. In *CHES 2010*, volume 6225 of *LNCS*, pages 413–427. Springer, 2010.
21. K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, and T. Shirai. Piccolo: An Ultra-Lightweight Blockcipher. In *CHES 2011*, volume 6917 of *LNCS*, pages 342–357. Springer, 2011.
22. Xilinx, Inc. Virtex-5 Libraries Guide for HDL Designs. `http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/virtex5_hdl.pdf`, 2009.

Appendix

		y															
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**a**	**b**	**c**	**d**	**e**	**f**
x	**00**	a8	f8	f0	00	d9	62	fd	39	4a	bd	af	06	a9	35	e1	df
	01	14	5b	82	0d	9b	d4	29	17	b9	02	f7	95	3e	65	79	d7
	02	7d	e4	ba	8b	cc	dc	1d	b5	87	71	07	fa	ef	d5	48	2f
	03	a7	e3	b2	6f	aa	ed	4d	a0	81	c0	8c	15	e0	19	9e	f1
	04	84	6b	4c	da	93	eb	58	2b	d3	27	33	76	b8	51	96	a3
	05	f4	c5	75	ae	d2	30	85	fb	64	38	3f	5c	9c	66	98	c1
	06	bb	63	a4	73	52	fc	9d	8d	24	25	31	cf	e2	57	9f	c3
	07	8f	f3	20	7f	3b	bc	bf	1a	54	03	91	0a	67	a5	16	10
	08	c7	e8	b3	21	13	72	0f	7a	01	88	e5	d1	f5	7c	40	ee
	09	97	4e	83	94	ad	5d	04	c4	32	a1	e7	92	43	b7	1e	e9
	0a	12	70	50	1f	a6	36	05	77	f6	ea	46	28	56	7b	55	db
	0b	61	34	b4	2e	9a	a2	6d	86	4f	cb	ab	ce	8a	6c	99	42
	0c	6a	5a	3d	ca	59	11	53	3c	ac	74	b6	c8	3a	89	2d	47
	0d	2a	cd	de	0c	5f	26	23	4b	c6	b0	7e	6e	f2	c2	b1	fe
	0e	18	37	0b	d6	e6	d0	49	8e	41	c9	69	44	d8	90	be	5e
	0f	0e	f9	60	ff	1b	ec	09	78	80	1c	dd	08	45	68	22	2c

Fig. 9. The G function selected in our experiments in the **First CT-RSA** profiles, values for the input as $(x, y) \in \mathbb{F}_2^5 \times \mathbb{F}_2^4$ (in hexadecimal format)