Side-Channel Analysis Protection and Low-Latency in Action

– case study of PRINCE and Midori –

Amir Moradi and Tobias Schneider

Horst Görtz Institute for IT-Security, Ruhr-Universität Bochum, Germany
{amir.moradi, tobias.schneider-a7a}@rub.de

Abstract. During the last years, the industry sector showed particular interest in solutions which allow to encrypt and decrypt data within one clock cycle. Known as low-latency cryptography, such ciphers are desirable for pervasive applications with real-time security requirements. On the other hand, pervasive applications are very likely in control of the end user, and may operate in a hostile environment. Hence, in such scenarios it is necessary to provide security against side-channel analysis (SCA) attacks while still keeping the low-latency feature.
Since the single-clock-cycle concept requires an implementation in a fully-unrolled fashion, the application of masking schemes – as the most widely studied countermeasure – is not straightforward. The contribution of this work is to present and discuss about the difficulties and challenges that hardware engineers face when integrating SCA countermeasures into low-latency constructions. In addition to several design architectures, practical evaluations, and discussions about the problems and potential solutions with respect to the case study PRINCE (also compared with Midori), the final message of this paper is a couple of suggestions for future low-latency designs to – hopefully – ease the integration of SCA countermeasures.

1 Introduction

The need for integration of side-channel analysis (SCA) [29] countermeasures into pervasive security-enabled devices is known to both academia and industry. Such a demand has also been motivated by several practical key-recovery attacks on commercial applications, e.g., [2,22,31,34,41,54]. From another perspective, there are several important applications for which a low-latency encryption and instant response time is highly desirable, such as read/write access to encrypted memory modules, which should be preferably conducted in a single clock cycle (initially motivated by [27]). It is also expected that given the ongoing growth of pervasive computing, there will be many more future embedded systems that require low-latency encryption, especially applications with real-time requirements, e.g., in the automotive domain. Hence, such pervasive applications, where low-latency cryptography is required, should be protected against SCA threats.

Here in this work, we present the challenges one may face by integrating SCA countermeasures into implementations with low-latency target. Insertion of *hiding*

techniques [32] in this scenario is either straightforward (such as noise generation or dual-rail logic) or ineffective (such as time randomization or shuffling) due to the fully unrolled architecture of low-latency implementations. Therefore, our focus is the integration of *masking* schemes into such designs. In particular, we concentrate on threshold implementation (TI) [40] as a provably-secure scheme against first-order SCA attacks. It should be noted that integration of ad hoc approaches, e.g., random pre-charging of [6], are out of our focus since we target solutions with provable security.

We should point out that it has previously been supposed that unrolled circuits – also the case of low-latency concept – are inherently secure against SCA attacks (see [6]). However, other practical results, e.g., in [36] and [51], showed that unrolling may make the attacks complicated since the common hypothetical power models (Hamming weight/distance) may not fit to the circuit's leakage anymore, but sophisticated yet first-order leakages can be exploited for key recovery.

As a known case study, PRINCE [13] (particularly designed as a low-latency cipher) is targeted in our investigations. We demonstrate design architectures and practical results with respect to the power consumption as well as SCA protection of different variants of implementations of PRINCE. In addition to several discussions about the SCA protection versus low-latency concept, we present a mixture of asynchronous circuit design methodology with threshold implementation which is expected to realize an SCA-protected self-timed design. Finally, having the PRINCE case study in mind, we give a couple of suggestions for the future low-latency cipher designs with the goal of mitigating the challenges, where SCA protection is desirable.

Furthermore, we consider the cipher Midori [3] which was designed with the goal of minimizing energy consumption. Since energy consumption and latency – to some extent – are proportional, we also provide a comparison between PRINCE and Midori with respect to latency when both are equipped with similar masking countermeasure.

2 Preliminaries

2.1 PRINCE

PRINCE [13] is a 64-bit block cipher that uses a 128-bit secret key k. The key expansion divides k into two 64-bit parts as $k = (k_0||k_1)$, and derives k'_0 from k_0 by a linear function as $(k_0 \ggg 1) \oplus (k_0 \gg 63)$. The subkeys k_0 and k'_0 are used as input and output whitening keys respectively, while k_1 is used as the round key for the core block cipher PRINCE$_{core}$ (see Figure 1).

Each of the first five round functions $\mathcal{R}_i$ consists of **S-Layer** (by a 4-bit Sbox), **M′-Layer** (multiplication with a 64×64 matrix M'), **ShiftRows** (the same as the AES one but on 4-bit cells), $\boldsymbol{RC_i}$**-add** (XORing the state with a 64-bit constant $\boldsymbol{RC_i}$), and $\boldsymbol{k_1}$**add** (XORing k_1 into the 64-bit state).

The last five inverse round functions $\mathcal{R}_i^{-1}$ are formed by the inverse of the corresponding operations. It is noteworthy that M' matrix is an involution, hence the inverse of **M′-Layer** is itself. Further, due to its underlying FX-construction [26]

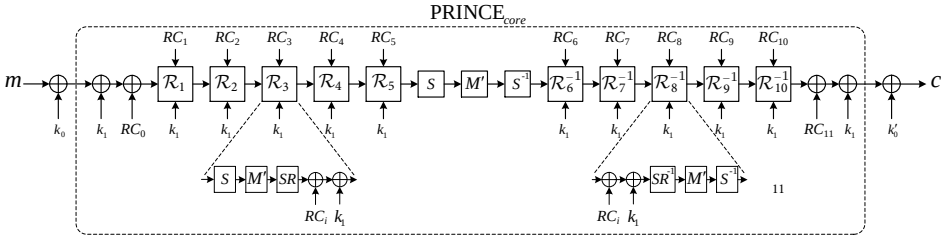


Fig. 1. A schematic view of PRINCE

as well as the α-reflection, i.e., $RC_{i\in\{0,\ldots,11\}} \oplus RC_{11-i} = \alpha$, the PRINCE encryption can turn to its decryption by swapping the whitening keys and XORing α to k_1,

$$\text{PRINCE}^{\text{Dec}}{}_{(k_0,k'_0,k_1)} = \text{PRINCE}^{\text{Enc}}{}_{(k'_0,k_0,k_1\oplus\alpha)}.$$

Note that $RC_0 = 0$, and $RC_{i\in\{1,\ldots,5\}}$ as well as α are derived from the fraction part of $\pi = 3.141\ldots$.

2.2 Threshold Implementation

Let us denote a 4-bit intermediate value of PRINCE, e.g., the Sbox input, as $\boldsymbol{x} = \langle x_1, \ldots, x_4\rangle$. Under the $n-1$ order Boolean masking concept, $\boldsymbol{x}$ is represented by $(\boldsymbol{x}^1, \ldots, \boldsymbol{x}^n)$, where $\boldsymbol{x} = \bigoplus_{i=1}^{n} \boldsymbol{x}^i$ and each $\boldsymbol{x}^i$ similarly denotes a 4-bit vector $\langle x_1^i, \ldots, x_4^i\rangle$.

The linear functions, such as **M′-Layer**, can be simply applied to the shares of $\boldsymbol{x}$ as $\mathsf{L}(\boldsymbol{x}) = \bigoplus_{i=1}^{n} \mathsf{L}(\boldsymbol{x}^i)$. Clearly, the non-linear functions, e.g., Sbox, cannot be trivially shared. Following the TI concept [8, 40], the minimum number of shares to realize an Sbox to be secure against first-order attacks is $n = t + 1$, where t denotes the algebraic degree of the Sbox. The shared Sbox should provide the output also in a shared form $(\boldsymbol{y}^1, \ldots, \boldsymbol{y}^m)$, where $m \geq n$ when the Sbox is a bijection. Obviously, to ensure the *correctness* of the computation, we should have $\mathsf{S}(\boldsymbol{x}) = \boldsymbol{y} = \bigoplus_{i=1}^{m} \boldsymbol{y}^i$.

Each output share $\boldsymbol{y}^{j\in\{1,\ldots,m\}}$ is given by a component function $\mathsf{f}^j(\cdot)$ over a subset of input shares. Defined as *non-completeness*, for first-order security each component function $\mathsf{f}^{j\in\{1,\ldots,m\}}(\cdot)$ must be independent of at least one input share. The security of masking schemes (to some extent) depends on the uniform distribution of the masks. Therefore, the output of a TI Sbox must be also uniform, since it supplies other non-linear functions. For example, the Sbox output of one PRINCE round is given to the next **S-Layer** after being processed by the linear diffusion layers. In case of the bijective PRINCE Sbox ($n = m$), each $(\boldsymbol{x}^1, \ldots, \boldsymbol{x}^n)$ should be mapped to a unique $(\boldsymbol{y}^1, \ldots, \boldsymbol{y}^n)$ to satisfy the *uniformity*. In other words, it is enough to check whether the TI Sbox also forms a bijection with $4\,n$ input (and output) bit length.

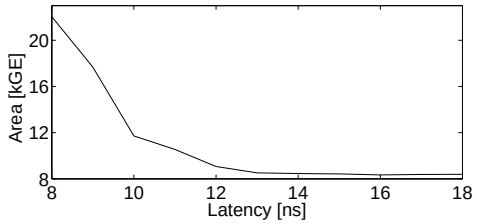


Fig. 2. Area versus latency of unrolled PRINCE

The PRINCE Sbox has an algebraic degree of $t = 3$. Hence, the number of input and output shares $n = m > 3$ what directly affects the complexity of the circuit and its associated area overhead. Therefore, it is preferable to decompose the Sboxes into smaller non-linear functions each with maximum algebraic degree of 2, which enables staying with the minimum number of shares $n = m = 3$. Note that in this case, registers must be placed between the shared decomposed functions. Otherwise, the glitches propagate into cascaded shared non-linear circuits, and violate the *non-completeness* property. As an example, the authors of [42] presented a decomposition of the PRESENT [12] Sbox into two quadratic bijections `g` and `f`.

Above we briefly reviewed the TI concept. For detailed information, the interested reader is referred to the original articles [8, 40].

3 Design Architectures

As stated before, PRINCE cipher has been designed with respect to low-latency feature. The goal was to achieve a short latency when the cipher is implemented in a fully-unrolled fashion. In other words, the implementation contains no sequential elements, e.g., register/flip-flop, and hence no clock.

In our investigations, in order to synthesize for an ASIC platform, we made use of Synopsys Design Compiler using the UMCL18G212T3 [49] ASIC standard cell library, i.e., UMC 0.18μm. As a side note, such a standard library has not been covered by the original article [13], where Nangate 45 nm, UMC 90 nm, and UMC 130 nm technologies have been considered. Therefore, the performance figures which we report here are based on our syntheses. Since the area requirement, i.e., Gate Equivalence (GE), of an implementation varies depending on the desired latency, we give in Figure 2 a curve of GE of the unrolled PRINCE implementation over the latency. We should stress that similar to the target of the seminal work [13], all our design architectures support both encryption and decryption. For the threshold implementations, the syntheses have been performed by keeping the hierarchy to avoid the combination of different shares (otherwise, first-order leakage is probable), and for the unrolled (unprotected) designs the hierarchy is avoided which allows the synthesizer to combine the cascaded circuits and reach the desired latency.

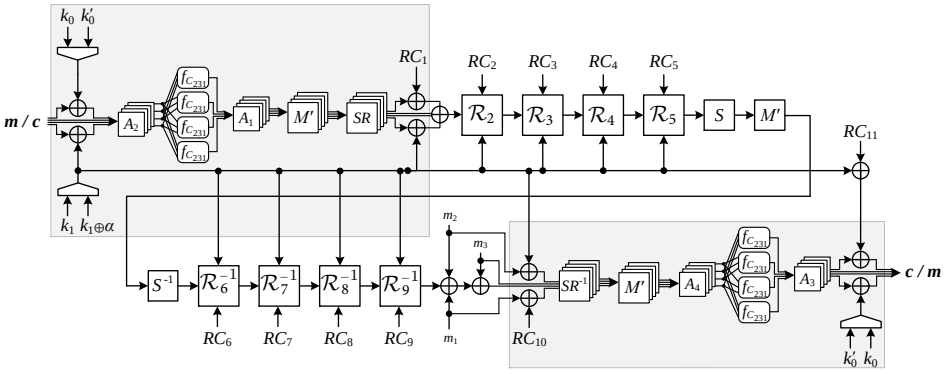


Fig. 3. Unrolled TI of PRINCE, only first and last round masked

As stated in Section 2.2, in order to realize a masked hardware implementation, the masked non-linear functions (Sboxes) should be separated from each other by means of registers to avoid the propagation of glitches. Therefore, an unrolled architecture can never be properly masked. It is noteworthy that unrolled architectures already change the leakage characteristics of the device (see [6, 51]). Hence, one may suppose that integration of masking into unrolled architectures may complicate the device leakage in such a way that it becomes unexploitable. However, such a combination would definitely lead to first-order leakage detectable e.g., by t-test [18, 24, 44]. As a heuristic-based example, it can be supposed that masking the first and last rounds of PRINCE should suffice to protect against SCA attacks[1]. The PRINCE Sbox is a cubic 4-bit bijection, i.e., algebraic degree $t = 3$, and at least $n = m = 4$ shares are required. The PRINCE Sbox belongs to the class $\mathcal{C}_{231}$ (with respect to the category given in [10]), which needs three decomposition stages to be uniformly shared with 3 as well as 4 shares, while it can be uniformly shared in one stage with $n = m = 5$ shares. It has been given in Section 2.2 that the *uniformity* is required because the output of the shared Sbox feeds the next non-linear functions. Hence, the non-uniform output of the first cipher round does not play any role, if the second cipher round is not masked. Therefore, we can stay with $n = m = 4$ shares and make the (non-uniform) shared PRINCE Sbox in one stage by direct sharing [11]. We have implemented such a design, whose block diagram is shown in Figure 3 and all the corresponding formulations are given in Appendix A. Further, its timing and area overheads are listed in Table 1. It turned out that this design is 3-6 times larger than the unprotected unrolled design and 2-3 times slower. We deal with its practical SCA evaluations in Section 3.4. We should emphasize that except for the first and last (i.e., masked) rounds, the hierarchy is not kept. This allows the optimization of the middle rounds, while the functions over the shared signals (in the first and

[1] In general, it is not a true statement since *i*) the unmasking at the end of the first round (see Figure 3) would anyway lead to (although hard-to-exploit) first-order leakage, and *ii*) the adversary can set certain plaintext bits to a fixed value and target the second cipher round.

the last rounds) must be kept separate to avoid any combination over the shares. Otherwise, the design would exhibit first-order leakage at the first and/or last rounds.

3.1 Round-based Architecture

Alternatively, we can consider the round-based architecture, although it obviously needs a fast clock, and the setup- and hold-time of the registers increase the whole latency. A round-based design has been given in the original article [13] which is also depicted in Figure 4(a). In this design two separate modules for the Sbox and its inverse are considered. It has been reported in a couple of works [9,37,42] that shared Sboxes are the most area consuming part. Therefore, one of our attempts with respect to this issue is to combine these two modules. Indeed, we have realized that the PRINCE Sbox and its inverse are affine equivalent. In other words, we can write

$$\forall \boldsymbol{x}, S(\boldsymbol{x}) = A_2 \left(S^{-1} \left(A_1 \left(\boldsymbol{x} \right) \right) \right),$$

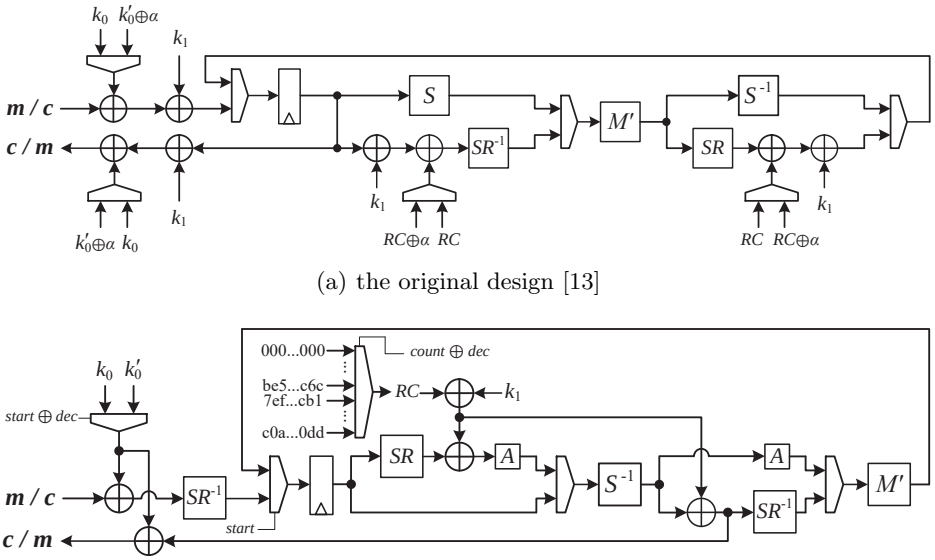
with A_1 and A_2 input- and output-affine transformations. In case of the PRINCE Sbox, there exists only one pair (A_1, A_2), and A_1 and A_2 are the same. Hence, we can write $S = A \circ S^{-1} \circ A$, with A: `B8A93021EDFC6574` as[2]

$$e = 1 + a + b + d, \qquad f = 1 + a, \qquad g = d, \qquad h = 1 + c,$$

with $\langle a, b, c, d \rangle$ the 4-bit input, $\langle e, f, g, h \rangle$ the 4-bit output, and a and e the least significant bits. Based on this findings, we developed another round-based architecture, shown in Figure 4(b), where only one **S-Layer** module is instantiated.

More detailed information about the active data path of both round-based designs at each cipher round is given in Appendix B. Table 1 lists the differences between these two designs. Note that we constrained the syntheses of both designs with different latencies to obtain both fastest and smallest designs for fair comparisons. As stated, the objective is to make use of only one **S-Layer**, hence our design utilizes more multiplexers compared to the original round-based design. Further, we optimized the way that whitening keys k_0 and k'_0 are added to the state considering the fact that it should support both encryption and decryption. Another issue is how to deal with the round constants. As given in Section 2.1, the round constants have been randomly selected, hence a combinatorial circuit should realize the selection of RC_i at each round. We have examined several cases, and the most optimized design (with respect to area) has been achieved by employing a multiplexer which selects one of the RC_0 to RC_{11} by the round counter. The role of optimization was to assign `0111` as the round counter to the round number $i = 5$. Therefore, the order of the round counter `0010`, `0011`, ..., `0111`, `1000`, ..., `1100`, `1101` can be reversed (required for decryption) by inverting the round counter bits (see Figure 4(b)). We should also point out that in both original and our round-based designs, the state register is placed right after the multiplexer. That allows the synthesizer to combine them and make use of scan flip-flop, which is smaller than a sum of a multiplexer and a flip-flop [43].

[2] It also holds for $S^{-1} = A' \circ S \circ A'$, with A': `5764FDCE1320B98A`.



(a) the original design [13]

(b) our developed design

Fig. 4. Round-based designs

As a side note, our round-based design is not necessarily the most optimized design. The tricks, that we used in our design, can be also applied in the original one (Figure 4(a)). However, since our target is to instantiate one **S-Layer** to ease the threshold implementations, we consider our round-based architecture as the basis of the further designs.

3.2 Uniform Sharing of the Sbox

The uniform sharing of the cubic 4-bit bijection $\mathcal{C}_{231}$, to which the PRINCE Sbox belongs, with 3 shares can only be achieved by a three-stage quadratic decomposition [11]. As listed in [11], there exist 5 quadratic classes, $\mathcal{Q}_4$, $\mathcal{Q}_{12}$, $\mathcal{Q}_{293}$, $\mathcal{Q}_{294}$, and $\mathcal{Q}_{299}$, that can be uniformly shared in one stage[3]. With respect to their size, i.e., the number of 2-input AND and XOR gates in their Algebraic Normal Form (ANF), $\mathcal{Q}_4$, $\mathcal{Q}_{294}$, $\mathcal{Q}_{12}$, $\mathcal{Q}_{293}$, and $\mathcal{Q}_{299}$ are respectively the smallest to the largest functions. We tried to decompose $\mathcal{C}_{231}$ by a set of the smallest quadratic functions. Indeed, several decompositions exist (see Appendix C for a complete list). If $\mathcal{Q}_4$ is involved in the decomposition, the other quadratic functions are a combination of $(\mathcal{Q}_{293}, \mathcal{Q}_{299})$ or $(\mathcal{Q}_{293}, \mathcal{Q}_{293})$ or $(\mathcal{Q}_{299}, \mathcal{Q}_{299})$. None of these combinations lead to a small design since $\mathcal{Q}_{293}$ and $\mathcal{Q}_{299}$ are amongst the largest classes. Instead, we can do the decomposition by $\mathcal{Q}_{294}$ in three stages. To this end, we first extracted

[3] One more quadratic class $\mathcal{Q}_{300}$ exists, but needs two stages for a uniform sharing with 3 shares.

all affine transformations A_1, A_2, and A_3 in such a way that

$$S^{-1} = A_3 \circ \mathcal{C}_{223} \circ A_2 \circ \mathcal{Q}_{294} \circ A_1.$$

There exist 2048 such (A_1, A_2, A_3) triples[4], and several solutions exist to decompose $\mathcal{C}_{223}$ (see Appendix C). One of the smallest ones is $\mathcal{C}_{223} = A_6 \circ \mathcal{Q}_{294} \circ A_5 \circ \mathcal{Q}_{294} \circ A_4$, and we found 262 144 affine triples (A_4, A_5, A_6) for such a decomposition. At the last step, we combined these two decompositions as

$$S^{-1} = \underbrace{A_3 \circ A_6}_{A_{out}} \circ \mathcal{Q}_{294} \circ \underbrace{A_5}_{A_{m2}} \circ \mathcal{Q}_{294} \circ \underbrace{A_4 \circ A_2}_{A_{m1}} \circ \mathcal{Q}_{294} \circ \underbrace{A_1}_{A_{in}}, \tag{1}$$

and examined all $2048 \times 262\,144$ cases[5]. With respect to the size of the resulting affines, we considered the number of 2-input XOR gates as well as the Hamming weight of the constants. The smallest combination has been achieved as

$$\begin{aligned}
A_{in} &: \texttt{8293C6D70A1B4E5F},\ e = b, && f = a, && g = c, && h = 1 + a + d,\\
A_{m1} &: \texttt{C480E6A2D591F7B3},\ e = d, && f = c, && g = 1 + b, && h = 1 + a,\\
A_{m2} &: \texttt{08C43BF72AE619D5},\ e = c, && f = c + d, && g = b, && h = a + b,\\
A_{out} &: \texttt{21748BDE6530CF9A},\ e = a + b, && f = 1 + a + c, && g = b + d, && h = c.
\end{aligned} \tag{2}$$

In order to share $\mathcal{Q}_{294} : \texttt{0123456789BAEFDC}$ as

$$e = a + bd, \qquad f = b + cd, \qquad g = c, \qquad h = d,$$

we can follow the direct sharing [11], which has been applied in [38]. The component function $f^{i,j}_{\mathcal{Q}_{294}}(\langle a^i, b^i, c^i, d^i\rangle, \langle a^j, b^j, c^j, d^j\rangle) = \langle e, f, g, h\rangle$ has been defined in [38] as

$$\begin{aligned}
e &= a^i + b^i d^i + d^i b^j + b^i d^j & g &= c^i\\
f &= b^i + c^i d^i + d^i c^j + c^i d^j & h &= d^i,
\end{aligned} \tag{3}$$

and it has been given that the three 4-bit output shares provided by $f^{2,3}_{\mathcal{Q}_{294}}(.,.)$, $f^{3,1}_{\mathcal{Q}_{294}}(.,.)$ and $f^{1,2}_{\mathcal{Q}_{294}}(.,.)$ make a uniform first-order sharing of $\mathcal{Q}_{294}$.

Since the affine functions applied on all shares do not change the uniformity, our construction – given in Equation (1) – in addition to the set of affines (Equation (2)) and component function $f_{\mathcal{Q}_{294}}$ (Equation (3)) form a uniform first-order sharing of the PRINCE Sbox inverse. To the best of our knowledge, this is amongst the smallest construction which fulfills all the TI properties with $n = m = 3$ shares. Note that the shared quadratic functions should be separated by registers to avoid the propagation of glitches.

3.3 Implementation

Our construction of the first-order TI of PRINCE is depicted in Figure 5. All operations except key and constant additions (and the **S-Layer**) are repeated

[4] It is the same for $S^{-1} = A_3 \circ \mathcal{Q}_{294} \circ A_2 \circ \mathcal{C}_{223} \circ A_1$.

[5] The result is a multiset, i.e., with repeated elements.

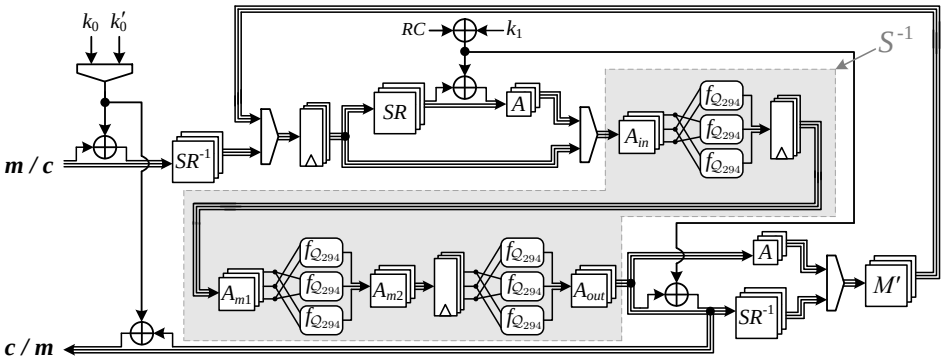


Fig. 5. Round-based first-order threshold implementation of PRINCE

three times. It suffices if the constant of the affine functions A, A_{in}, A_{m1}, A_{m2}, and A_{out} are applied on only one share[6]. The key and the constants are not shared, which is the same scenario applied in several works, e.g., [7–9, 37, 42], and is adequate to resist against first-order attacks. Hence, the keys and constant are also applied on only one share.

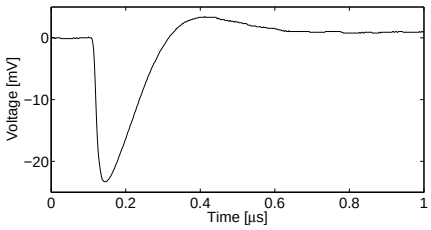
Due to the registers integrated into the shared Sbox, the design realizes a pipeline with three stages. In other words, three consecutive (shared) inputs (plaintexts/ciphertexts) can be fed into the design, and after 40 clock cycles three outputs (ciphertexts/plaintexts) are consecutively given out. Thanks to the uniform sharing of the Sbox, excluding the masks required to share the input, the design does not require any fresh randomness during the computations. The performance figures of this design are also given in Table 1 for comparison purposes.

3.4 Practical Evaluations

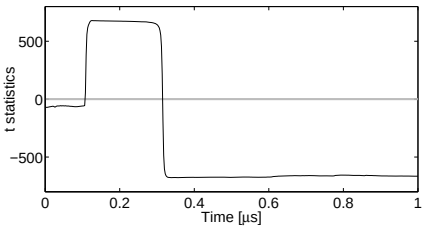
For the practical investigations – rather than ASIC-based experiments or simulation – we ported the designs to an FPGA-based platform. We have used a SAKURA-X board [1] with a Kintex-7 FPGA, particularly designed for SCA evaluations. In order to monitor the power consumption, we measured the voltage drop over a shunt resistor placed at the Vdd path of the Kintex FPGA. The power traces have been collected by means of a digital LeCroy oscilloscope at the sampling rate of 500 MS/s. Because of the low amplitude of the measured signal (due to the underlying low-power technology of Xilinx 7 series FPGAs), we employed an AC amplifier ZFL-1000LN+ from Mini-Circuits with 10 dB gain.

For SCA evaluation purposes, we applied the non-specific t-test (also known as *fixed versus random* t-test). This test procedure, originally called TVLA, has been proposed in [18], extended in [44] and [20, 21], and applied in e.g., [7, 8, 38]. The test – which compares the leakages associated to random inputs with that to a fixed input – can examine the existence of a detectable leakage, but cannot give any impression whether the leakage is exploitable. Hence, in case the t-test reports

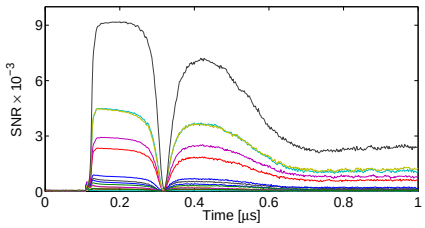
[6] It does not affect either the functionality or the uniformity.



(a) sample trace

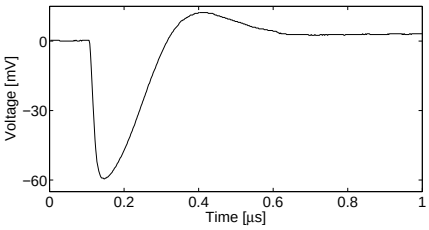


(b) t-test, 1 million traces

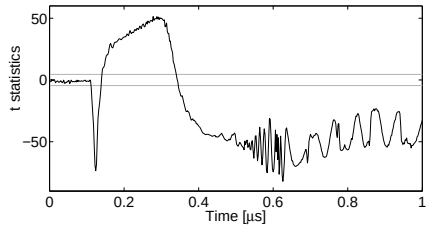


(c) SNR, plaintext nibbles, 1 million traces

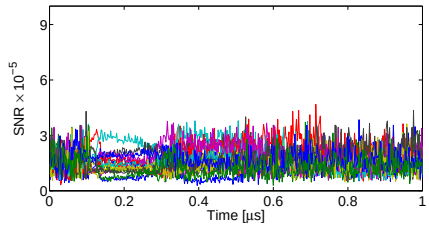
Fig. 6. Evaluation results, unrolled unprotected design



(a) sample trace

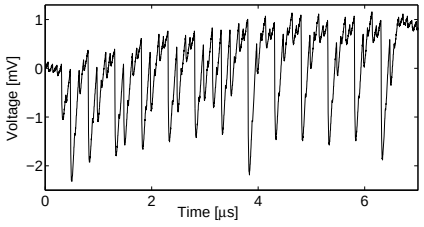


(b) t-test, 100 million traces

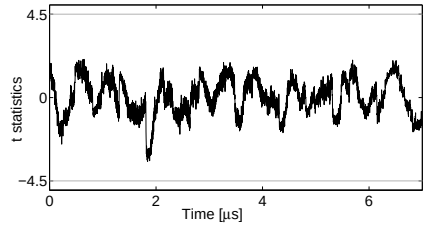


(c) SNR, plaintext nibbles, 1 million traces

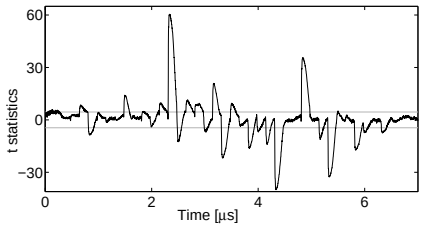
Fig. 7. Evaluation results, unrolled TI design (first and last rounds masked)



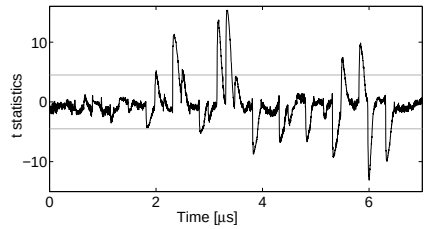
(a) sample trace



(b) 1st-order t-test, 100 million traces



(c) 2nd-order t-test, 100 million traces



(d) 3rd-order t-test, 100 million traces

Fig. 8. Evaluation results, round-based TI design

a first-order detectable leakage, we perform a signal-to-noise ratio (SNR) check. In such a check, the variance of the average leakage traces classified by e.g., the value of a plaintext nibble (divided by the variance of the noise) is examined [32]. It indeed can give an overview about the dependency of the average (first-order) leakages to the processed data. Here, we do not show any attack results, and only discuss about the existence of detectable leakages, and compare the amount of dependency of the leakages to the processed data.

For the unrolled unprotected design, Figure 6 shows a sample power trace, the t-test result as well as the SNR over all 16 plaintext nibbles. Since the design is not masked, the t-test as expected shows a pretty strong first-order leakage. Along the same lines, the corresponding SNR exhibit a clear dependency between the traces and the plaintext nibbles. Hence, a successful key-recovery attack is expected (e.g., in [51]). We should here note that 23.3 mV power peak is relatively large[7] for this low-power FPGA. Since several gates are packed into one LUT, the equivalent design in ASIC can be more glitchy, and hence (probably) more energy consuming[8]. This **may** harden the development of fully unrolled (even low-latency) designs into low-energy, e.g., battery-powered, applications (see simulation-based results in [3] and [4]).

We have shown the corresponding results of the unrolled TI design in Figure 7. As stated before, only the first and the last rounds are masked by means of four shares (see Figure 3). During the measurements, the 4-share input as well as other 3 independent fresh random masks (m_1, m_2, m_3 for the last round) are given to the Kintex-7 FPGA. The output is also provided in a 4-share masked form. As discussed before, since the middle rounds are not masked, we expected that the t-test exhibits first-order leakage. However, the SNR over plaintext nibbles shows a significant reduction, a factor of about 0.03, compared to the unrolled unprotected design (Figure 6(c) vs. Figure 7(c)). It indeed gives an impression that the first-order attacks on the first round are expect to be challenging. However, if the attacker fixes certain plaintext parts, he can target the second cipher round, which is not masked.

Compared to these, the round-based TI design, whose results are depicted in Figure 8, exhibits no first-order leakage with 100 million traces confirming the correctness of our TI construction. We should here emphasize that the design was operating at a frequency of 6 MHz during the measurements, and we filled the 3-stage pipeline with the same data. In other words, this way we have reduced the algorithmic noise of the pipeline architecture (see [46]), that allows us to mitigate the side effects into the evaluations. As expected, the higher-order t-tests report detectable leakages through higher-order moments. Regardless of the difference in their SCA-resistance levels, the power peak of the round-based TI architecture is significantly smaller than that of the unrolled designs, i.e., 10 times and 30 times compared to the unprotected and the TI unrolled designs respectively.

[7] We showed real voltage values, i.e., output of the amplifier divided by 10.
[8] This is a guess by the authors and should be examined in practice

Table 1. Performance figures of different PRINCE implementations. For each design, the first and the second row represents the smallest and the fastest variant respectively.

Design	Area [GE]	Crit. Path [ns]	Clock #	Latency [ns]	Throughput [Gbps]	Power[a] Peak [mV]	DPA res.
Unrolled	8 512	13	1	13	4.923	23.3	
	22 040	8		8	8.000		
Unrolled TI	48 012	38	1	38	1.684	59.5	∼[b]
	77 921	13.2		13.2	4.848		
Round-based [13]	2 809	5.6	13	72.8	0.879	1.7[c]	
	4 698	1.5		19.5	3.282		
Round-based ours	2 286	4.9	14	68.6	0.933	1.5[c]	
	4 663	2		28	2.285		
Round-based TI	9 292	4	40	160	1.143[d]	2.3[c]	✓
	11 275	1.9		76	2.406[d]	21[e]	
Round-based TI Asynchronous (simple `Ack`)	25 701	11	40×2[f]	800	0.208[d]	53.7	✓
	31 936	5.4		432	0.423[d]		
Midori64 Round-based TI	7 297	4	31	124	1.000[g]	20.2[e]	✓
	9 237	1.9		58.9	2.105[g]		

[a] measured from the FPGA implementations
[b] only at the first and the last rounds
[c] @ 6 MHz
[d] considering the 3-stage pipeline
[e] by controlled ring-oscillator clock
[f] doubled due to pre-charge/evaluation phases
[g] considering the 2-stage pipeline

4 Asynchronous Design

We have already discussed in Section 1 that low-latency concept is closely connected to the unrolled (single-cycle) architecture since the high clock rates (needed to rapidly run register-based designs) are not available or supported by many systems. For instance, in many FPGA designs clock rates above 200 MHz are often difficult to realize. In this settings, asynchronous circuits seem to be an alternative to this issue. With asynchronous circuit design, also known as self-timed and clock-less design, it is possible to realize circuits with high performance parameters in terms of their power, throughput, electromagnetic emissions, etc. [39, 45]. Asynchronous design is not as well-established and widely-used as synchronous design methodology. Hence, the standard tools for asynchronous design are not available, or not widely known, or particularly customized for certain technologies.

Because the field of asynchronous circuit design covers a wide range, we focus only on certain concepts which are relevant to our case studies. In terms of PRINCE, consider the round-based synchronous architecture in Figure 4(a). The maximum clock frequency is defined by the longest critical path (most likely when both Sbox and its inverse are active). However, such a path is not always active. In

other words, in all clock cycles except the middle one (see Figure 17 in Appendix B) the design can be clocked faster. If this design is realized by asynchronous design methodology, the end of the computation of one cipher round initiates the start of the next round. Hence, the design operates at its maximum speed, or let say with its lowest latency. In this case – similar to the unrolled architectures – the time when the computations are finished, i.e., the ciphertext is ready to be read, depends on the given inputs, but the maximum latency can be estimated.

As shown in Section 3, the round-based TI design can provide the first-order resistance, but it needs a clock with a frequency between 250 MHz and 500 MHz to achieve the highest throughput (see Table 1). Hence, our objective in this section is to realize the round-based TI design with an asynchronous design methodology.

State of the art. We should emphasize that the asynchronous design has been previously applied as a sole SCA countermeasure. One of the earlier works [33] describes a smartcard chip which relies on self-timed circuits to provide protection against physical attacks. The authors proposed to solely use dual-rail encoding to reduce the threat of data-dependent power consumption but also noted the obvious difficulties of this approach, e.g., varying wire lengths. Furthermore, they highlighted the problem of timing leakage of asynchronous circuits and advise to minimize data dependent gate delays coupled with the insertion of dummy delays to reduce this leakage. Later in [23] the security of a similar self-timed circuit has thoroughly been tested in practice. The authors found that small imbalances in the dual-rail circuits cause data-dependent leakage which enables an attacker to perform a successful DPA on the asynchronous circuit. They showed that their asynchronous design alone is not sufficient to prevent SCA attacks, and that these imbalances need to be eliminated during the design process to increase the level of security. This is in line with [30, 53] where some of the difficulties, e.g., no global clock, with respect to performing DPA on asynchronous designs are described.

One of the first clock-less implementation of AES was presented in [52]. It also relies on power-balancing capability of dual-rail and the absence of a global clock to thwart DPA. The dual-rail circuits were found to be more secure than the single-rail one, however this is only based on simulation results and a thorough practical evaluation is missing.

Another approach to secure AES using clock-less circuits is presented in [14]. It again relies on an asynchronous style called quasi delay insensitive (QDI) which has a range of supply voltages. The authors noted the above-mentioned limitation of this implementation style with respect to SCA resistance [15]. Therefore, they proposed to lower the supply voltage to reduce the SNR and thwart DPA. However, [14] does not include practical experiments related to this approach. Further techniques [16, 17] have been proposed to harden QDI against DPA based on the introduction of random timing and path swapping. However, their efficiency was only evaluated using electrical simulations.

More recently, an AES round function in Null Convention logic - another delay insensitive logic paradigm - has been proposed in [50] in which the SCA resistance has again been only evaluated with simulations.

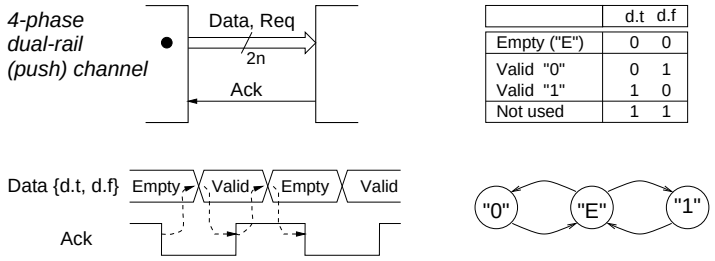


Fig. 9. A delay-insensitive 4-phase dual-rail protocol (taken from [45])

It should be noted that in a majority of the aforementioned articles SCA resistance was not the sole motivation for asynchronous circuits. Other beneficial properties include a low-power consumption for embedded devices and some form of an integrated fault tolerant scheme.

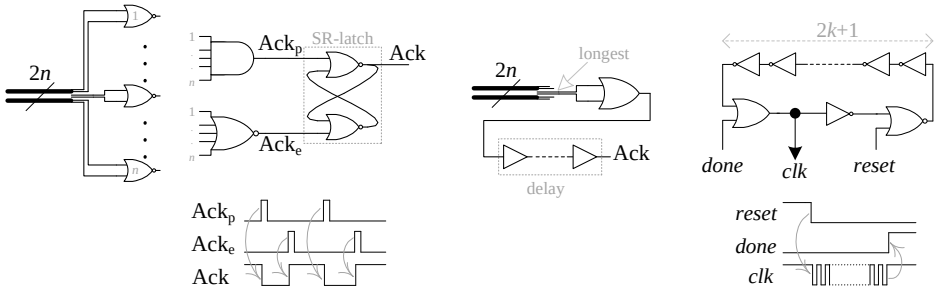
What we want to examine here is not the application of asynchronous design to prevent SCA leakages. In short, we do not aim at e.g., realizing the round-based **unprotected** architecture with asynchronous methodology and examine its SCA resistance. Instead, our goal is to investigate the challenges and outcomes of implementing a correctly-masked design, e.g., round-based threshold implementation (Figure 5), under the concept of asynchronous designs. Such an investigation is conducted with the goal of achieving a clock-less design while it is expected to still satisfy the desired first-order SCA protection due to its underlying uniform TI construction.

4.1 Fundamentals

Different parts of an asynchronous circuit need to communicate with each other. For example, the finish of one PRINCE round should initiate the next round. A couple of different handshaking protocols exist to establish such a communication.

The 4-phase dual-rail protocol encodes the data signals into two wires per bit (see Figure 9). Each logical '1' or '0' is represented by {1,0} or {0,1} respectively, while {0,0} is known as "no data" (or "empty") and {1,1} as invalid. A transition from one valid coding to another is not allowed, unless an "empty" value is transmitted in-between, that forms a *return-to-zero* protocol. This protocol is very robust; two parties can communicate reliably regardless of delays in the wires, i.e., it is *delay-insensitive* [45].

This concept is very similar to the WDDL logic style [48], which has been designed to mitigate SCA leakages. The underlying dual-rail pre-charge logic is the same encoding; the valid encodings {0,1} and {1,0} are known as *evaluation phase* and the empty value {0,0} as *pre-charge phase*. A WDDL circuit is usually a synchronous design, where the evaluation/pre-charge phases are controlled by the clock signal (the same concept as in Figure 9 by replacing the `Ack` signal with clock). This protocol is familiar to most digital designers, and avoids any



(a) `Ack` over all dual rails (b) Simple `Ack` (c) Clock Generator

Fig. 10. Exemplary circuits to generate the `Ack` and *clk* signals
(a) and (b) for asynchronous designs, (c) for a synchronous design

glitches in the circuit hence achieving a low-power construction. However, it has a disadvantage due to the extra return-to-zero transitions that cost time and energy.

We can implement the combinatorial parts of a design based on the WDDL concept, and add extra logic to detect the end of the pre-charge as well as evaluation phase. This allows us to form the `Ack` signal (see Figure 9). As shown in Figure 10(a), we can integrate n 2-input NOR gates, each of which for a dual-rail signal, and by means of an n-input AND and an n-input NOR gate[9] we can generate $\texttt{Ack}_\texttt{p}$ and $\texttt{Ack}_\texttt{e}$ respectively. When all n dual-rail signals are in pre-charge phase (resp. in evaluation phase), it can be detected by observing $\texttt{Ack}_\texttt{p}$ (resp. $\texttt{Ack}_\texttt{e}$). These two signals can drive an SR-latch to generate the desired `Ack` signal.

4.2 Asynchronous Round-based TI

WDDL combinatorial circuits (generally asynchronous circuits) are glitch free, i.e., each dual-rail signal changes only once at each pre-charge/evaluation phase. Threshold implementation has been developed mainly for glitchy circuits, and the registers should be placed between the non-linear shared functions to avoid the propagation of the glitches [40]. Hence, at the first glance it seems that it is not essential anymore to instantiate such registers if the circuit is glitch free.

Following this concept, we have implemented the round-based TI design presented in Figure 11, and did not integrate registers between the shared $\mathcal{Q}_{294}$ functions. The state register is moved to the end of the round function, and the `Ack` signal is generated based on the state register input. By a couple of engineering tricks the design is mapped to our FPGA platform. We should here emphasize that Xilinx FPGAs are developed yet only for synchronous designs, and integration of asynchronous circuits is neither straightforward nor efficient. For example, each dual-rail WDDL gate should be implemented by a LUT [5].

Our design is a self-timed circuit, i.e., it does not require an external clock, and once the `reset` signal goes LO, the circuit starts the first evaluation phase, which is the first PRINCE round. Controlled by the internally-generated `Ack` signal, the

[9] Such large gates are made by cascading the smaller gates.

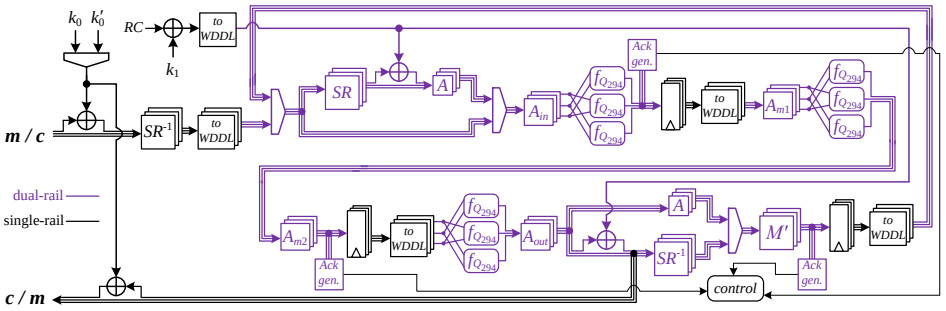
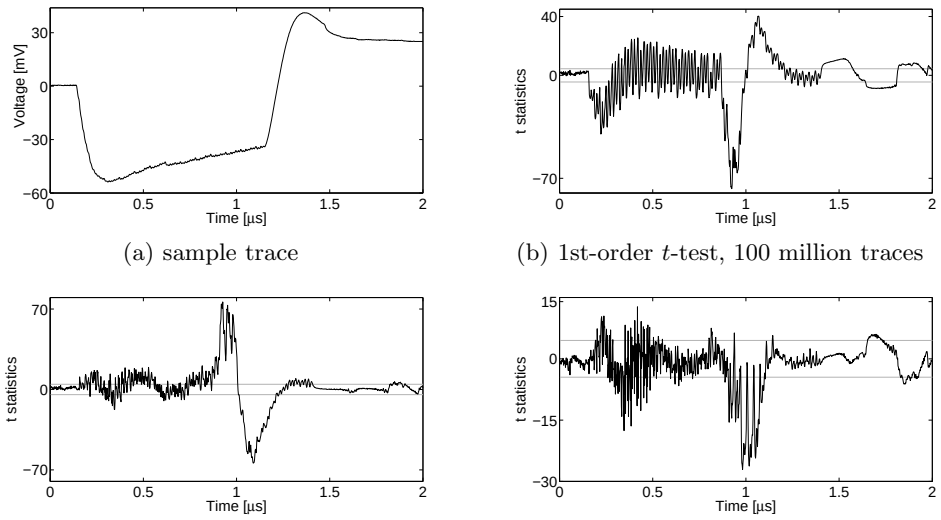


Fig. 11. Asynchronous round-based first-order threshold implementation of PRINCE

end of the evaluation phase triggers the state register to save the cipher state and simultaneously the start of the pre-charge phase. As stated before, a disadvantage of such a concept is its required interleaved pre-charge/evaluation phases. Because we avoided the extra registers within the Sbox, the design does not form a pipeline anymore. Therefore, a full PRINCE is performed by 14 (pre-charge, evaluation) cycles. Figure 12 shows a sample power trace of such a design, where the cipher rounds can be identified. However, the t-test indicates a pretty strong first-order leakage. Note that the design still realizes a uniform threshold implementation with 3 shares, and we have not used WDDL as an SCA countermeasure, rather as a 4-phase dual-rail protocol to enable detection of the end of the evaluation (and pre-charge) of the combinatorial circuit.

A more careful investigation about the detected first-order leakage clarified that although the circuit is glitch free, the non-linear circuits are cascaded. One of the component functions of the second non-linear circuit (the second shared $\mathcal{Q}_{294}$ in Figure 11) starts to evaluate when two output shares of the first non-linear circuit are both evaluated. Further, these two shares depend on all three shares of the Sbox input. Therefore, the start of the evaluation of the second non-linear circuit depends on all three input shares of the Sbox. This, which is a non-linear condition (i.e., when both two output shares of the first non-linear circuit are evaluated) is the reason for such a detectable first-order leakage (see [23] for a similar experience on an unmasked design). Although placing registers between the shared non-linear functions was initially introduced to avoid the propagation of glitches, it also synchronizes the start of their evaluation to be independent of the timing of the previous stage. As a result, the shared non-linear functions should also be isolated from each other even in asynchronous circuits.

If we isolate the shared non-linear circuits by means of registers, and trigger the registers to store when all 3 shares are evaluated, again the time of triggering the registers as well as the circuit which generates the `Ack` signals (Figure 10(a)) depends on all 3 shares and leak through first-order moments. As a proof of concept, we have examined this issue by realizing the asynchronous round-based TI design with registers in the Sbox module where the combinatorial parts are made by WDDL gates. In this case, the `Ack` signals are generated by observing the input of all three registers, i.e., when the pre-charge/evaluation of the entire circuit



(a) sample trace

(b) 1st-order t-test, 100 million traces

(c) 2nd-order t-test, 100 million traces

(d) 3rd-order t-test, 100 million traces

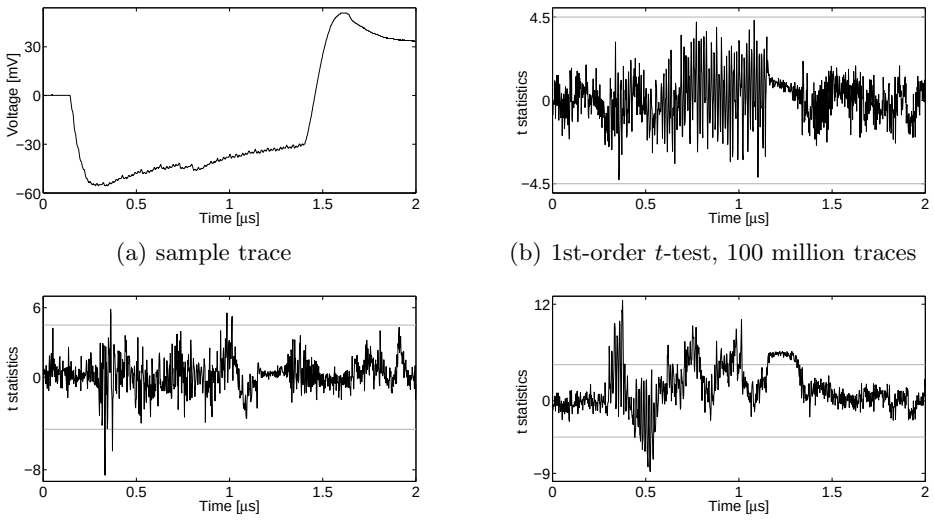
Fig. 12. Evaluation results, asynchronous round-based TI design

– pipeline with 3 stages – is completed. The evaluation of this construction has also showed detectable first-order leakage. So, we omit the corresponding results.

As a side note, the early propagation effect [47] of WDDL aggravates this issue. In the above explained experiments we have used the noEE version [5] of WDDL (available only for FPGAs), that avoids early propagation only in evaluation phase. We have also made use of its successor, AWDDL [35] (also only for FPGAs) which avoids early propagation in both phases. Regardless of its double area requirements, its utilization in our case slightly reduced the first-order leakage, but could not avoid it due to the known imbalances between the delay of dual rails. In other words, the time required for full pre-charge/evaluation phase of non-linear circuits still depends on three shares and hence on unshared input.

Therefore, the only solution which we could consider for a secure design is to simplify the `Ack` generator circuit. It means that if we generate the `Ack` signal based on only one share of one of the state registers, the start time of the next pre-charge/evaluation phase should be independent from the unshared values. However, such a circuit cannot guarantee that the pre-charge/evaluation of the other parts of the circuit are also finished. Therefore, we have found a path with the largest delay and connected the `Ack` generator circuit accordingly. To ensure the end of the pre-charge/evaluation of the other circuits, the generated `Ack` signal is delayed (see Figure 10(b)).

A sample trace as well as the t-test results are shown in Figure 13, which confirms the prevention of first-order leakages. This construction is still a self-timed asynchronous circuit without external clock, but it is vastly customized. For instance, it does not operate at its maximum speed, and controlled mainly by a delayed periodic signal. Hence, we do not benefit from all the features of



(a) sample trace

(b) 1st-order t-test, 100 million traces

(c) 2nd-order t-test, 100 million traces

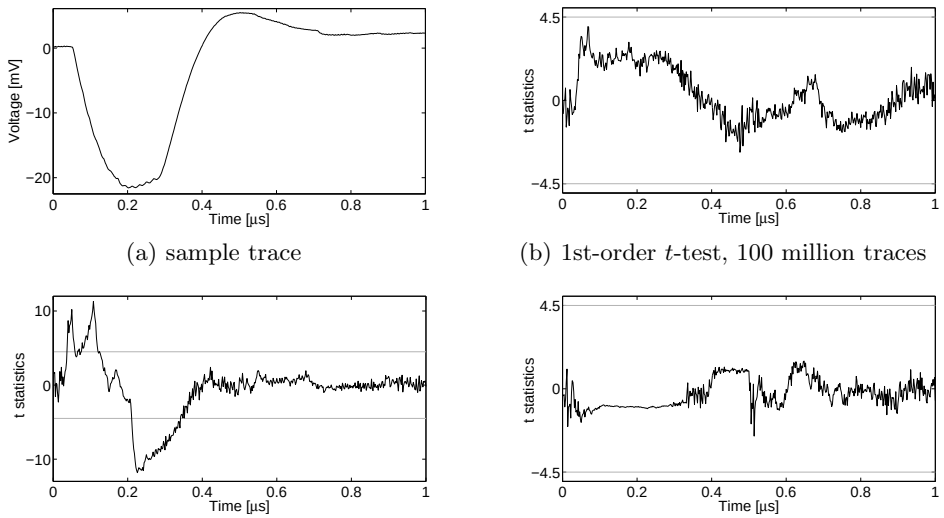
(d) 3rd-order t-test, 100 million traces

Fig. 13. Evaluation results, asynchronous round-based TI design with simple `Ack`

asynchronous methodology. If we ignore the low-power feature of this construction, it is not significantly different from the corresponding synchronous design with a high speed clock. As we listed in Table 1, the asynchronous design is much larger than its synchronous variant. Further, due to the interleaved (pre-charge, evaluation) phases, the latency of the asynchronous design is also not convincing.

The interleaved (pre-charge, evaluation) phases of 4-phase dual-rail protocols (e.g., WDDL which we used here) doubles the latency of the design. Alternatively, one can utilize a *2-phase dual-rail* protocol [45], where '1' and '0' values are encoded as signal transitions. Such protocols lead to faster but much more complex circuits. We have applied Level-Encoded Dual-Rail (LEDR) [19] concept and designed and evaluated the corresponding circuit, but due to the similarity of the results to that of the WDDL, their presentation is omitted. In short, the design was much bigger than its WDDL variant, but slightly faster. However, all issues with respect to isolation of non-linear functions as well as the `Ack` generator circuit hold true.

In this situation, where the operation of non-linear circuits **must** be isolated and independent of other non-linear parts, we believe that the synchronous design is favorable. For the remaining issue, i.e., absence of a fast clock in many applications where low-latency cryptography is required, we suggest to generate such a clock by means of a ring oscillator. Since the energy consumption of large clock-trees (operated at a high frequency) is not desirable in many applications, the ring oscillator can be controlled by the start and end of e.g., the encryption module. A schematic view of an exemplary circuit is depicted in Figure 10(c). Obviously the ring oscillator should be adjusted based on the critical path delay of the circuit.



(a) sample trace

(b) 1st-order t-test, 100 million traces

(c) 2nd-order t-test, 100 million traces

(d) 3rd-order t-test, 100 million traces

Fig. 14. Evaluation results, round-based TI design clocked by a controlled ring oscillator

We have practically evaluated such a construction as well, whose results are shown by Figure 14. As expected, higher power consumption peak compared to the same design operated at 6 MHz (see Figure 8(a)) is observed. However, the first-order leakage is still avoided, and more interestingly the higher-order leakages are mitigated (Figure 14 vs. Figure 8). The reason is due to the overlap between the adjacent power peaks, which leads to higher amount of noise, and consecutively harder higher-order leakages to detect, e.g., in [38].

5 Discussion

We have discussed and shown that SCA-protected designs (by means of masking) should involve registers even in case of asynchronous designs. Therefore, the low-latency concept – with a perspective of unrolled architectures – is in contradiction with masking in hardware. As a result, round-based architectures are the only possible solution for applications, where provably-secure SCA protection is required. In this scenario, in order to achieve a low latency two parameters play the most important role: *i*) the latency of each cipher round, and *ii*) the number of rounds.

Obviously, the most challenging issue, which we faced, was uniform realization of the shared Sbox with 3 shares. In the seminal article [13], 8 different Sboxes (up to affine equivalent) are suggested for the PRINCE-family. However, all of them need at least a 3-stage decomposition to be able to uniformly shared with 3 shares. Such a decomposition, as shown in Section 3.3, leads to a pipeline round-based architecture with 3 stages. This – as stated above – increases the number of clock cycles required for each cipher round, and negatively affect the latency.

For the future designs, our first suggestion is to select Sboxes, whose uniform sharing needs a low number of stages. The extreme case is to apply quadratic Sboxes, which can be shared in one stage, but such a choice leads to higher number of rounds (see PrintCipher [28]), which affect the low-latency target as well. Hence, the trade-off here is to select either a quadratic Sbox, which needs more number of rounds, or a cubic TI-friendly Sbox which forms a pipeline, hence more number of clock cycles per round.

The second challenging issue was to deal with round constants. In case of PRINCE, the round constants have been selected from a semi-random source (fraction of $\pi = 3.141\ldots$). This design decision does not have any performance penalty in case of unrolled architecture, since a round constant just turns some XOR gates of the prior AddRoundKey to XNOR, i.e., for free[10]. However, for a round-based design, this leads to a relatively large combinatorial circuit since each round constant should be selected at each round based on the round counter[11]. Hence, it is advisable to systematically generate the round constants, e.g., by means of an LFSR. Note that if a large LFSR is chosen, the area required to save its state (by registers) has also a negative impact on the area overhead.

In case of PRINCE, due to its underlying α-reflection structure, encryption and decryption circuits are very similar. **$\mathbf{M}'$-Layer** of PRINCE is self-inverse, and the Sbox is affine equivalent to its inverse, but it consists of two different round functions. Such a construction makes the round-based architecture (required for SCA protection) more complicated as both round functions need to be implemented (see Figure 4), which obviously increases the area requirements. Hence, it is preferred to have a design with a unique round function. In this case, achieving highly-similar encryption and decryption might be challenging.

5.1 Comparison to Midori

The Midori cipher has been introduced in [3] with the main goal of reducing the energy consumption. Based on the simulation results and the discussions given in [3,4], a round-based architecture is targeted to achieve the minimum energy consumption per bit. Further, it has been shown that the full latency of a round-based implementation of Midori outperforms that of other considered ciphers including PRINCE. Therefore, we considered Midori64 for comparison purposes[12].

Midori64 state is a 64-bit block, and its 4-bit Sbox (applied on all state nibbles) is an involution. Its linear layer includes an involutional MixColumn operation (made of a couple of XORs), and a ShiftCell which swaps the 4-bit cells of the state. It consists of 15 rounds, and respectively 15 round constants (each 16 bits) which are added to the LSB of the state nibbles. The 128-bit key is divided into two parts which are alternatively added to the sate at each round, and their XOR is used as a pre- and post-whitening key.

[10] 2-input XOR and XNOR gates need the same area [43].

[11] In our round-based designs, the selection of the round constant followed by 64-bit XOR need an area of 265 GE.

[12] We are aware of the weakness reported in [25], but to be compatible with PRINCE, i.e., 64-bit block size, we excluded Midori128 in our investigations.

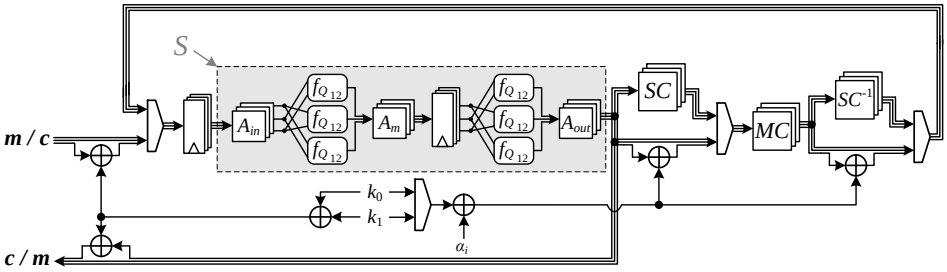


Fig. 15. Round-based first-order threshold implementation of Midori64

Figure 15 shows a round-based implementation of Midori64, which supports both encryption and decryption. Note that the authors of [3] proposed to apply the inverse of the linear operations, i.e., $\mathsf{ShiftCell}^{-1} \circ \mathsf{MixColumn}$, over the round keys and round constants for the decryption. However, we found our solution (see Figure 15) which needs 64 extra 2-input XOR gates, cheaper than the original suggestion.

In order to realize its threshold implementation, the linear layers are simply repeated over the 3 shares, and a uniform representation of its Sbox is constructed. The Midori64 Sbox is affine equivalent to $\mathcal{C}_{266}$ class [11], which can be decomposed to two quadratic bijections with uniform TI. Amongst many possible solutions we selected $\mathcal{Q}_{12} \times \mathcal{Q}_{12}$ and found affine functions as

$$S = A_{out} \circ \mathcal{Q}_{12} \circ A_m \circ \mathcal{Q}_{12} \circ A_{in}.$$

There exist 147 456 such (A_{in}, A_m, A_{out}) triples, and we selected the following settings (with respect to the same criteria explained in Section 3.2):

$$\begin{aligned}
A_{in} &: \texttt{0A1B82934E5FC6D7},\ e = b, && f = a, && g = d, && h = a + c,\\
A_m &: \texttt{84B70C3F95A61D2E},\ e = b + d, && f = b, && g = a, && h = 1 + a + c,\\
A_{out} &: \texttt{8A02DF57CE469B13},\ e = c, && f = a, && g = c + d, && h = 1 + b.
\end{aligned} \tag{4}$$

The sharing of $\mathcal{Q}_{12} : \texttt{0123456789CDEFAB}$ with

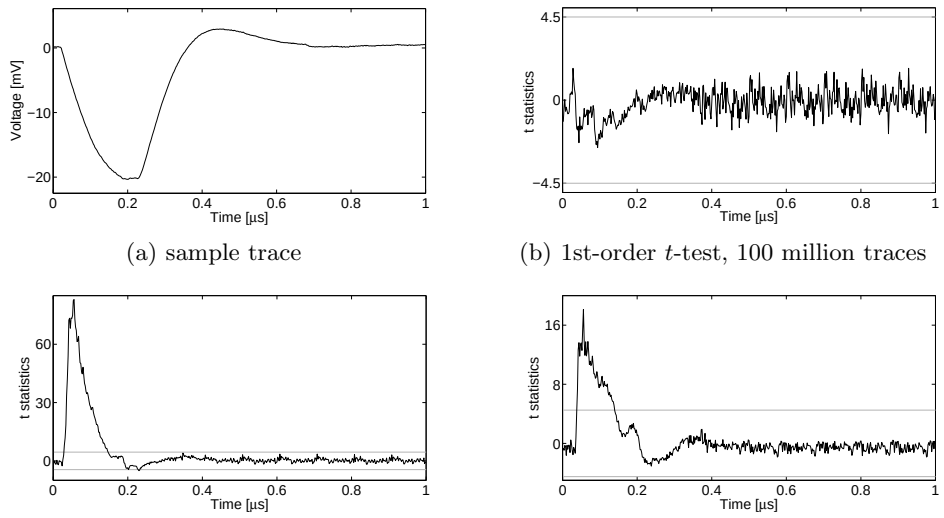
$$e = a, \qquad f = b + bd + cd, \qquad g = c + bd, \qquad h = d$$

can be derived by direct sharing [11]. If we define the component function $f^{i,j}_{\mathcal{Q}_{12}}(\langle a^i, b^i, c^i, d^i\rangle, \langle a^j, b^j, c^j, d^j\rangle) = \langle e, f, g, h\rangle$ as

$$\begin{aligned}
e &= a^i, && g = b^i d^i + d^i b^j + b^i d^j,\\
f &= b^i + b^i d^i + d^i b^j + b^i d^j + c^i d^i + d^i c^j + c^i d^j, && h = d^i,
\end{aligned} \tag{5}$$

we can form a uniform shared representation of $\mathcal{Q}_{12}$ by $f^{2,3}_{\mathcal{Q}_{12}}(.,.)$, $f^{3,1}_{\mathcal{Q}_{12}}(.,.)$ and $f^{1,2}_{\mathcal{Q}_{12}}(.,.)$, as shown in Figure 15.

We have also practically examined its SCA resistance by the FPGA prototype. For comparison purposes we considered only a synchronous version, where



(a) sample trace

(b) 1st-order t-test, 100 million traces

(c) 2nd-order t-test, 100 million traces

(d) 3rd-order t-test, 100 million traces

Fig. 16. Evaluation results, round-based TI design of **Midori64** clocked by a controlled ring oscillator

the clock is provided by a controlled ring oscillator (with the same number of inverters as in the corresponding PRINCE design). The results (indicating first-order resistance and stronger leakage through higher-order moments compared to its corresponding PRINCE) are shown in Figure 16, and the performance results are listed in Table 1.

5.2 Conclusions

We have presented the results of an extensive study on application of masking, particularly TI, on PRINCE considering its low-latency goal. As given in Table 1, the asynchronous design is around 2.8 times larger and around 2.6 times slower than its synchronous variant. Further, an overview about its power consumption (FPGA prototype) shows no advantage, even compared to the case when the synchronous design operates at a high frequency[13]. More importantly, we faced several issues regarding its detectable first-order leakage. Finally, the design, which could prevent the leakages, was not much structurally different to a synchronous design, whose clock is internally generated.

Based on Table 1, the fastest synchronous round-based TI needs 11 275 GE which is in the range of the unprotected unrolled design (8 512 - 22 040 GE). Although its critical path with 1.9 ns delay is more than 4 times shorter than that of the fastest unrolled design, its 40 clock cycle latency leads to 76 ns which is around 10 times more than 8 ns latency of the unrolled design. However, its underlying

[13] Note here the difference between power consumption of equivalent FPGA and ASIC circuits.

pipeline architecture compensates in terms of throughput to be between 2 and 3 times less than the unprotected unrolled designs.

Compared to the synchronous round-based TI of PRINCE, Midori64 is smaller and achieves lower latency (58.9 ns vs. 76 ns for the fastest designs), but their throughput are comparable considering the full capacity of the pipelines. We should emphasize that most of the suggestions (given above) can be seen in the design of Midori: *i*) the Sbox is an involution and TI friendly, *ii*) MixColumn is an involution, *iii*) it consists of only one type of round function, and *iv*) the round constants are short (16 bits per round) although they cannot be generated systematically. However, with respect to [25] our observation is that: there is still a gap to fill, i.e., a low-latency cipher, which in addition to the desired cryptographic strength, can easily deal with the challenges addressed in this article. In short, the candidate should still achieve a low latency when fully unrolled as well as in a round-based fashion, and at the same time its masked (TI) round-based variant is efficient in terms of area and latency for the applications, where provably-secure SCA protection is required.

Acknowledgment

The authors would like to acknowledge Alexander Kühn for his help with implementation of different asynchronous variants of PRINCE on FPGA.

References

1. Side-channel AttacK User Reference Architecture. `http://satoh.cs.uec.ac.jp/SAKURA/index.html`.
2. J. Balasch, B. Gierlichs, R. Verdult, L. Batina, and I. Verbauwhede. Power Analysis of Atmel CryptoMemory - Recovering Keys from Secure EEPROMs. In *Topics in Cryptology - CT-RSA 2012*, volume 7178 of *Lecture Notes in Computer Science*, pages 19–34. Springer, 2012.
3. S. Banik, A. Bogdanov, T. Isobe, K. Shibutani, H. Hiwatari, T. Akishita, and F. Regazzoni. Midori: A Block Cipher for Low Energy. In *ASIACRYPT 2015*, volume 9453 of *Lecture Notes in Computer Science*, pages 411–436. Springer, 2015.
4. S. Banik, A. Bogdanov, and F. Regazzoni. Exploring Energy Efficiency of Lightweight Block Ciphers. In *SAC 2015*, volume 9566 of *Lecture Notes in Computer Science*, pages 178–194. Springer, 2016.
5. S. Bhasin, S. Guilley, F. Flament, N. Selmane, and J. Danger. Countering early evaluation: an approach towards robust dual-rail precharge logic. In *Workshop on Embedded Systems Security - WESS 2010*, page 6. ACM, 2010.
6. S. Bhasin, S. Guilley, L. Sauvage, and J. Danger. Unrolling Cryptographic Circuits: A Simple Countermeasure Against Side-Channel Attacks. In *CT-RSA 2010*, volume 5985 of *Lecture Notes in Computer Science*, pages 195–207. Springer, 2010.
7. B. Bilgin, B. Gierlichs, S. Nikova, V. Nikov, and V. Rijmen. A More Efficient AES Threshold Implementation. In *Progress in Cryptology - AFRICACRYPT 2014*, volume 8469 of *Lecture Notes in Computer Science*, pages 267–284. Springer, 2014.
8. B. Bilgin, B. Gierlichs, S. Nikova, V. Nikov, and V. Rijmen. Higher-Order Threshold Implementations. In *Advances in Cryptology - ASIACRYPT 2014, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 326–343. Springer, 2014.

9. B. Bilgin, B. Gierlichs, S. Nikova, V. Nikov, and V. Rijmen. Trade-Offs for Threshold Implementations Illustrated on AES. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 34(7):1188–1200, 2015.
10. B. Bilgin, S. Nikova, V. Nikov, V. Rijmen, and G. Stütz. Threshold Implementations of All 3×3 and 4×4 S-Boxes. In *Cryptographic Hardware and Embedded Systems - CHES 2012*, volume 7428 of *Lecture Notes in Computer Science*, pages 76–91. Springer, 2012.
11. B. Bilgin, S. Nikova, V. Nikov, V. Rijmen, N. Tokareva, and V. Vitkup. Threshold implementations of small S-boxes. *Cryptography and Communications*, 7(1):3–33, 2015.
12. 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 *Cryptographic Hardware and Embedded Systems - CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.
13. J. Borghoff, A. Canteaut, T. Güneysu, E. B. Kavun, M. Knezevic, L. R. Knudsen, G. Leander, V. Nikov, C. Paar, C. Rechberger, P. Rombouts, S. S. Thomsen, and T. Yalçin. PRINCE - A Low-Latency Block Cipher for Pervasive Computing Applications - Extended Abstract. In *ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 208–225. Springer, 2012.
14. G. Bouesse, M. Renaudin, A. Witon, and F. Germain. A clock-less low-voltage aes crypto-processor. In *Solid-State Circuits Conference, 2005. ESSCIRC 2005. Proceedings of the 31st European*, pages 403–406. IEEE, 2005.
15. G. F. Bouesse, M. Renaudin, S. Dumont, and F. Germain. DPA on quasi delay insensitive asynchronous circuits: Formalization and improvement. In *DATE*, pages 424–429. IEEE Computer Society, 2005.
16. G. F. Bouesse, M. Renaudin, and G. Sicard. Improving DPA resistance of quasi delay insensitive circuits using randomly time-shifted acknowledgment signals. In *VLSI-SoC*, volume 240 of *IFIP*, pages 11–24. Springer, 2005.
17. G. F. Bouesse, G. Sicard, and M. Renaudin. Path swapping method to improve DPA resistance of quasi delay insensitive asynchronous circuits. In *CHES*, volume 4249 of *Lecture Notes in Computer Science*, pages 384–398. Springer, 2006.
18. J. Cooper, E. Demulder, G. Goodwill, J. Jaffe, G. Kenworthy, and P. Rohatgi. Test Vector Leakage Assessment (TVLA) Methodology in Practice. International Cryptographic Module Conference, 2013.
19. M. E. Dean, T. E. Williams, and D. L. Dill. Efficient self-timing with level-encoded 2-phase dual-rail (ledr). In *Conference on Advanced Research in VLSI*, pages 55–70. MIT Press, 1991.
20. A. A. Ding, C. Chen, and T. Eisenbarth. Faster, and More Robust T-test Based Leakage Detection. In *COSADE 2016*, volume ?? of *Lecture Notes in Computer Science*, pages ??–?? Springer, 2016. to appear.
21. F. Durvaux and F. Standaert. From Improved Leakage Detection to the Detection of Points of Interests in Leakage Traces. In *EUROCRYPT 2016*, volume ?? of *Lecture Notes in Computer Science*, pages ??–?? Springer, 2016. to appear.
22. T. Eisenbarth, T. Kasper, A. Moradi, C. Paar, M. Salmasizadeh, and M. T. M. Shalmani. On the Power of Power Analysis in the Real World: A Complete Break of the KeeLoqCode Hopping Scheme. In *Advances in Cryptology - CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 203–220. Springer, 2008.
23. J. J. A. Fournier, S. W. Moore, H. Li, R. D. Mullins, and G. S. Taylor. Security evaluation of asynchronous circuits. In *CHES*, volume 2779 of *Lecture Notes in Computer Science*, pages 137–151. Springer, 2003.

24. G. Goodwill, B. Jun, J. Jaffe, and P. Rohatgi. A testing methodology for side channel resistance validation. In *NIST non-invasive attack testing workshop*, 2011. `http://csrc.nist.gov/news_events/non-invasive-attack-testing-workshop/papers/08_Goodwill.pdf`.
25. J. Guo, J. Jean, I. Nikolić, K. Qiao, Y. Sasaki, and S. M. Sim. Invariant Subspace Attack Against Full Midori64. Cryptology ePrint Archive, Report 2015/1189, 2015. `http://eprint.iacr.org/`.
26. J. Kilian and P. Rogaway. How to Protect DES Against Exhaustive Key Search. In *CRYPTO 1996*, volume 1109 of *Lecture Notes in Computer Science*, pages 252–267. Springer, 1996.
27. M. Knezevic, V. Nikov, and P. Rombouts. Low-Latency Encryption - Is "Lightweight = Light + Wait"? In *CHES 2012*, volume 7428 of *Lecture Notes in Computer Science*, pages 426–446. Springer, 2012.
28. L. R. Knudsen, G. Leander, A. Poschmann, and M. J. B. Robshaw. PRINTcipher: A Block Cipher for IC-Printing. In *CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 16–32. Springer, 2010.
29. P. C. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *CRYPTO 1999*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.
30. K. J. Kulikowski, M. Su, A. B. Smirnov, A. Taubin, M. G. Karpovsky, and D. MacDonald. Delay insensitive encoding and power analysis: A balancing act. In *ASYNC*, pages 116–125. IEEE Computer Society, 2005.
31. J. Liu, Y. Yu, F. Standaert, Z. Guo, D. Gu, W. Sun, Y. Ge, and X. Xie. Small Tweaks Do Not Help: Differential Power Analysis of MILENAGE Implementations in 3G/4G USIM Cards. In *ESORICS 2015*, volume 9326 of *Lecture Notes in Computer Science*, pages 468–480. Springer, 2015.
32. S. Mangard, E. Oswald, and T. Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, 2007.
33. S. W. Moore, R. D. Mullins, P. A. Cunningham, R. J. Anderson, and G. S. Taylor. Improving smart card security using self-timed circuits. In *ASYNC*, pages 211–218. IEEE Computer Society, 2002.
34. A. Moradi, A. Barenghi, T. Kasper, and C. Paar. On the vulnerability of FPGA bitstream encryption against power analysis attacks: extracting keys from xilinx Virtex-II FPGAs. In *ACM Conference on Computer and Communications Security - CCS 2011*, pages 111–124. ACM, 2011.
35. A. Moradi and V. Immler. Early Propagation and Imbalanced Routing, How to Diminish in FPGAs. In *CHES 2014*, volume 5984 of *Lecture Notes in Computer Science*, pages 598–615. Springer, 2014.
36. A. Moradi, O. Mischke, and C. Paar. Practical evaluation of DPA countermeasures on reconfigurable hardware. In *HOST 2011*, pages 154–160. IEEE, 2011.
37. A. Moradi, A. Poschmann, S. Ling, C. Paar, and H. Wang. Pushing the Limits: A Very Compact and a Threshold Implementation of AES. In *Advances in Cryptology - EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 69–88. Springer, 2011.
38. A. Moradi and A. Wild. Assessment of Hiding the Higher-Order Leakages in Hardware - What Are the Achievements Versus Overheads? In *CHES 2015*, volume 9293 of *Lecture Notes in Computer Science*, pages 453–474. Springer, 2015.
39. C. J. Myers. *Asynchronous Circuit Design*. Wiley, 2001.
40. 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.
41. D. Oswald and C. Paar. Breaking Mifare DESFire MF3ICD40: Power Analysis and Templates in the Real World. In *CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 207–222. Springer, 2011.

42. A. Poschmann, A. Moradi, K. Khoo, C. Lim, H. Wang, and S. Ling. Side-Channel Resistant Crypto for Less than 2, 300 GE. *J. Cryptology*, 24(2):322–345, 2011.
43. A. Y. Poschmann. *Lightweight Cryptography: Cryptographic Engineering for a Pervasive World*. PhD thesis, Ruhr University Bochum, 2009.
44. T. Schneider and A. Moradi. Leakage Assessment Methodology - A Clear Roadmap for Side-Channel Evaluations. In *CHES*, volume 9293 of *Lecture Notes in Computer Science*, pages 495–513. Springer, 2015.
45. J. Spars and S. Furber. *Principles of Asynchronous Circuit Design: A Systems Perspective*. Springer Publishing Company, Incorporated, 1st edition, 2010.
46. F. Standaert, S. B. Örs, and B. Preneel. Power Analysis of an FPGA: Implementation of Rijndael: Is Pipelining a DPA Countermeasure? In *CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 30–44. Springer, 2004.
47. D. Suzuki and M. Saeki. Security Evaluation of DPA Countermeasures Using Dual-Rail Pre-charge Logic Style. In *Cryptographic Hardware and Embedded Systems - CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, pages 255–269. Springer, 2006.
48. K. Tiri and I. Verbauwhede. A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation. In *Design, Automation and Test in Europe - DATE 2004*, pages 246–251. IEEE Computer Society, 2004.
49. Virtual Silicon Inc. 0.18 μm VIP Standard Cell Library Tape Out Ready, Part Number: UMCL18G212T3, Process: UMC Logic 0.18 μm Generic II Technology: 0.18μm, July 2004.
50. J. Wu, Y. Kim, and M. Choi. Low-power side-channel attack-resistant asynchronous s-box design for AES cryptosystems. In *ACM Great Lakes Symposium on VLSI*, pages 459–464. ACM, 2010.
51. V. Yli-Mäyry, N. Homma, and T. Aoki. Improved Power Analysis on Unrolled Architecture and Its Application to PRINCE Block Cipher. In *LightSec 2015*, volume 9542 of *Lecture Notes in Computer Science*, pages 148–163. Springer, 2016.
52. A. Yu and D. S. Brée. A clock-less implementation of the AES resists to power and timing attacks. In *ITCC (2)*, pages 525–532. IEEE Computer Society, 2004.
53. Z. C. Yu, S. B. Furber, and L. A. Plana. An investigation into the security of self-timed circuits. In *ASYNC*, pages 206–215. IEEE Computer Society, 2003.
54. Y. Zhou, Y. Yu, F. Standaert, and J. Quisquater. On the Need of Physical Security for Small Embedded Devices: A Case Study with COMP128-1 Implementations in SIM Cards. In *Financial Cryptography and Data Security - FC 2013*, volume 7859 of *Lecture Notes in Computer Science*, pages 230–238. Springer, 2013.

A Masked Unrolled Design (only first and last rounds)

To share the Sbox and its inverse with 4 shares, we represented the Sbox as $S = A_2 \circ \mathcal{C}_{231} \circ A_1$ and its inverse as $S^{-1} = A_4 \circ \mathcal{C}_{231} \circ A_3$ with
A_1 : `EF548932AB10CD76`, A_2 : `08192A3B4C5D6E7F`, A_3 : `92386DC7F45E0BA1`, A_4 : `51736240FBD9C8EA`, and $\mathcal{C}_{231}$: `0123468B59CEDA7F` as

$$\begin{aligned} e =& a + d + ac + ad + bd + abc + bcd \\ f =& b + ac + bc + bd + abd \\ g =& c + d + bc + ad + cd + abd + bcd \\ h =& bc + ad + bd + cd + abd + acd + bcd. \end{aligned}$$

By applying direct sharing on $\mathcal{C}_{231}$ we reach the component function $f^{i,j,k}_{\mathcal{C}_{231}}(\langle a^i, b^i, c^i, d^i\rangle, \langle a^j, b^j, c^j, d^j\rangle, \langle a^k, b^k, c^k, d^k\rangle) = \langle e, f, g, h\rangle$ as

$$\begin{aligned}
e =& a^i + d^i + a^i c^i + a^i c^j + a^i c^k + a^j c^i + a^i d^i + a^i d^j + a^i d^k + a^j d^i + b^i d^i + \\
& b^i d^j + b^i d^k + b^j d^i + a^i b^i c^i + a^i b^j c^k + a^i b^k c^j + a^j b^i c^k + a^j b^k c^i + \\
& a^k b^i c^j + a^k b^j c^i + a^i b^i c^j + a^i b^j c^j + a^i b^i c^k + a^i b^k c^k + a^j b^j c^i + \\
& a^j b^i c^i + a^i b^j c^i + a^i b^k c^i + a^j b^i c^j + b^i c^i d^i + b^i c^j d^k + b^i c^k d^j + \\
& b^j c^i d^k + b^j c^k d^i + b^k c^i d^j + b^k c^j d^i + b^i c^i d^j + b^i c^j d^j + b^i c^i d^k + \\
& b^i c^k d^k + b^j c^j d^i + b^j c^i d^i + b^i c^j d^i + b^i c^k d^i + b^j c^i d^j
\end{aligned}$$

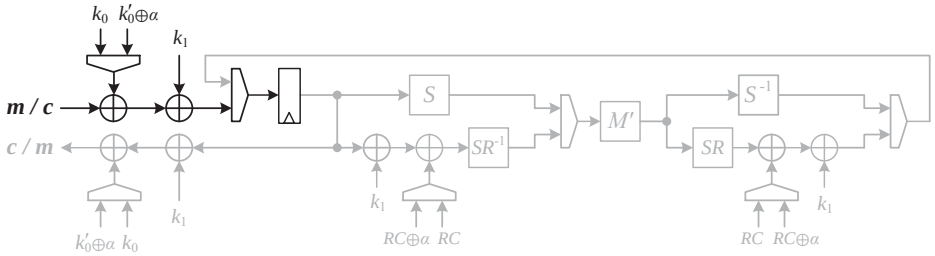
$$\begin{aligned}
f =& b^i + a^i c^i + a^i c^j + a^i c^k + a^j c^i + b^i c^i + b^i c^j + b^i c^k + b^j c^i + b^i d^i + b^i d^j + \\
& b^i d^k + b^j d^i + a^i b^i d^i + a^i b^j d^k + a^i b^k d^j + a^j b^i d^k + a^j b^k d^i + a^k b^i d^j + \\
& a^k b^j d^i + a^i b^i d^j + a^i b^j d^j + a^i b^i d^k + a^i b^k d^k + a^j b^j d^i + a^j b^i d^i + \\
& a^i b^j d^i + a^i b^k d^i + a^j b^i d^j
\end{aligned}$$

$$\begin{aligned}
g =& c^i + d^i + b^i c^i + b^i c^j + b^i c^k + b^j c^i + a^i d^i + a^i d^j + a^i d^k + a^j d^i + c^i d^i + c^i d^j + \\
& c^i d^k + c^j d^i + a^i b^i d^i + a^i b^j d^k + a^i b^k d^j + a^j b^i d^k + a^j b^k d^i + a^k b^i d^j + \\
& a^k b^j d^i + a^i b^i d^j + a^i b^j d^j + a^i b^i d^k + a^i b^k d^k + a^j b^j d^i + a^j b^i d^i + a^i b^j d^i + \\
& a^i b^k d^i + a^j b^i d^j + b^i c^i d^i + b^i c^j d^k + b^i c^k d^j + b^j c^i d^k + b^j c^k d^i + b^k c^i d^j + \\
& b^k c^j d^i + b^i c^i d^j + b^i c^j d^j + b^i c^i d^k + b^i c^k d^k + b^j c^j d^i + b^j c^i d^i + b^i c^j d^i + \\
& b^i c^k d^i + b^j c^i d^j
\end{aligned}$$

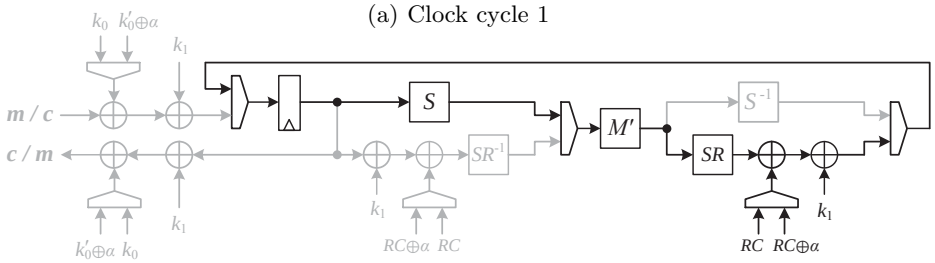
$$\begin{aligned}
h =& b^i c^i + b^i c^j + b^i c^k + b^j c^i + a^i d^i + a^i d^j + a^i d^k + a^j d^i + b^i d^i + b^i d^j + b^i d^k + \\
& b^j d^i + c^i d^i + c^i d^j + c^i d^k + c^j d^i + a^i b^i d^i + a^i b^j d^k + a^i b^k d^j + a^j b^i d^k + \\
& a^j b^k d^i + a^k b^i d^j + a^k b^j d^i + a^i b^i d^j + a^i b^j d^j + a^i b^i d^k + a^i b^k d^k + a^j b^j d^i + \\
& a^j b^i d^i + a^i b^j d^i + a^i b^k d^i + a^j b^i d^j + a^i c^i d^i + a^i c^j d^k + a^i c^k d^j + a^j c^i d^k + \\
& a^j c^k d^i + a^k c^i d^j + a^k c^j d^i + a^i c^i d^j + a^i c^j d^j + a^i c^i d^k + a^i c^k d^k + a^j c^j d^i + \\
& a^j c^i d^i + a^i c^j d^i + a^i c^k d^i + a^j c^i d^j + b^i c^i d^i + b^i c^j d^k + b^i c^k d^j + b^j c^i d^k + \\
& b^j c^k d^i + b^k c^i d^j + b^k c^j d^i + b^i c^i d^j + b^i c^j d^j + b^i c^i d^k + b^i c^k d^k + b^j c^j d^i + \\
& b^j c^i d^i + b^i c^j d^i + b^i c^k d^i + b^j c^i d^j
\end{aligned}$$

By implementing four instances of this component function $f^{2,3,4}_{\mathcal{C}_{231}}(.,.,.)$, $f^{3,4,1}_{\mathcal{C}_{231}}(.,.,.)$, $f^{4,1,2}_{\mathcal{C}_{231}}(.,.,.)$, and $f^{1,2,3}_{\mathcal{C}_{231}}(.,.,.)$ we reach a correct, non-complete, but non-uniform sharing of $\mathcal{C}_{231}$. Note that the 64-bit masks m_1, m_2, and m_3 required to share the last input round are independent of the masks used to share the cipher input.

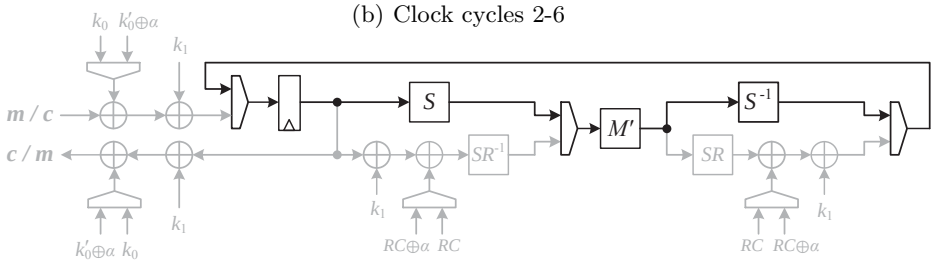
B Round-based Designs



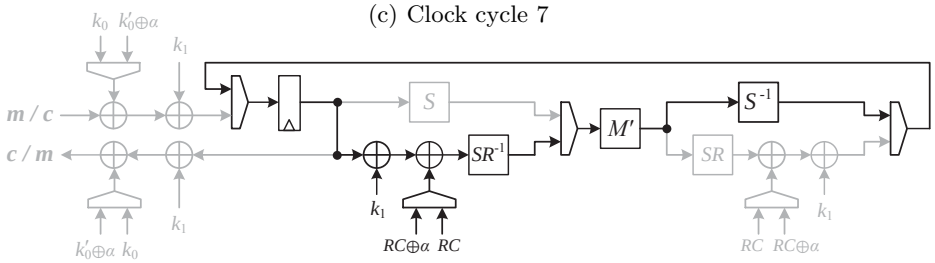
(a) Clock cycle 1



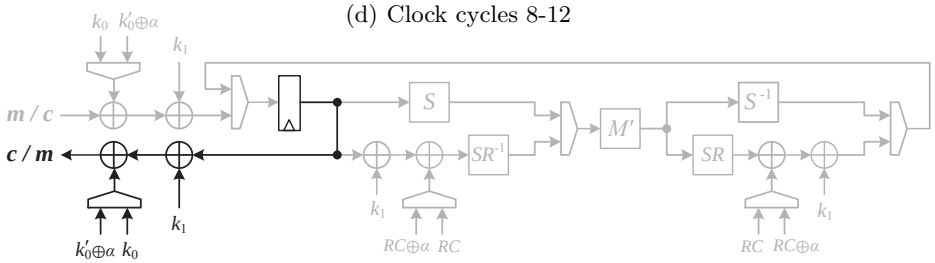
(b) Clock cycles 2-6



(c) Clock cycle 7

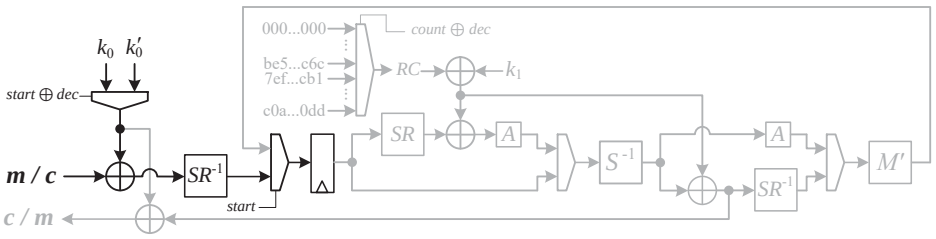


(d) Clock cycles 8-12

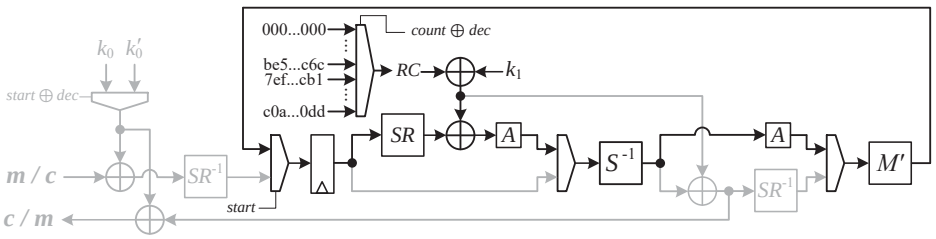


(e) Clock cycle 13

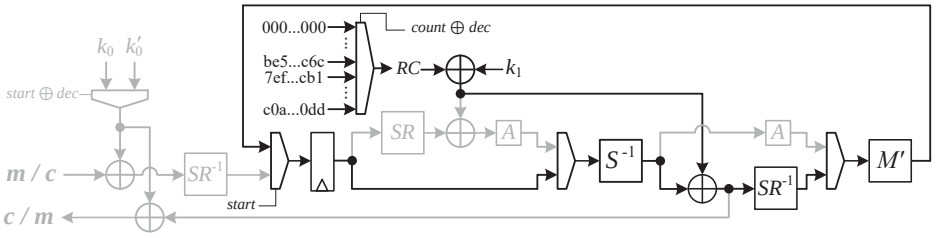
Fig. 17. Detailed active parts of the original round-based architecture [13]



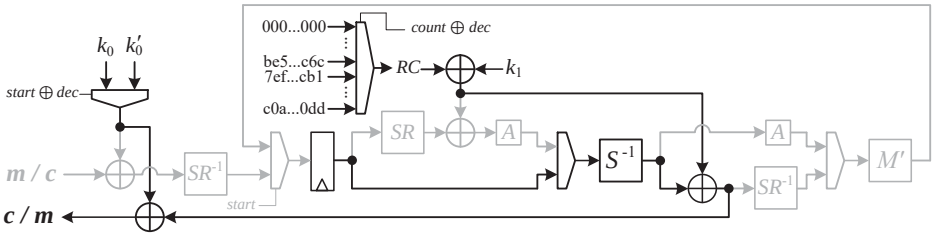
(a) Clock cycle 1



(b) Clock cycles 2-7



(c) Clock cycles 8-13



(d) Clock cycle 14

Fig. 18. Detailed active parts of our round-based architecture

C Decomposition of $\mathcal{C}_{231}$

Table 2. All possible ways to decompose $\mathcal{C}_{231}$ by selected quadratic bijections in three stages

$\mathcal{C}_{231}$	$\mathcal{C}_{150}$	$\mathcal{C}_{151}$	$\mathcal{C}_{158}$	$\mathcal{C}_{159}$	$\mathcal{C}_{168}$	$\mathcal{C}_{171}$	$\mathcal{C}_{172}$	$\mathcal{C}_{214}$	$\mathcal{C}_{215}$	$\mathcal{C}_{223}$	$\mathcal{C}_{262}$	$\mathcal{C}_{266}$	$\mathcal{C}_{296}$	$\mathcal{C}_{297}$
$\mathcal{Q}_4$			×	×										
$\mathcal{Q}_{12}$	×	×	×	×	×	×	×			×	×	×	×	×
$\mathcal{Q}_{293}$	×	×	×	×	×	×	×	×		×	×	×	×	×
$\mathcal{Q}_{294}$	×	×	×	×	×	×	×			×	×	×		×
$\mathcal{Q}_{299}$	×	×	×	×	×	×	×		×	×	×	×	×	×

$\mathcal{C}_{150}$:	$\mathcal{Q}_{12} \times \mathcal{Q}_{293}$					
$\mathcal{C}_{151}$:	$\mathcal{Q}_{293} \times \mathcal{Q}_{12}$					
$\mathcal{C}_{158}$:	$\mathcal{Q}_{299} \times \mathcal{Q}_{293}$					
$\mathcal{C}_{159}$:	$\mathcal{Q}_{293} \times \mathcal{Q}_{299}$					
$\mathcal{C}_{168}$:	$\mathcal{Q}_{293} \times \mathcal{Q}_{293}$					
$\mathcal{C}_{171}$:	$\mathcal{Q}_{293} \times \mathcal{Q}_{12}$	$\mathcal{Q}_{294} \times \mathcal{Q}_{293}$				
$\mathcal{C}_{172}$:	$\mathcal{Q}_{12} \times \mathcal{Q}_{293}$	$\mathcal{Q}_{293} \times \mathcal{Q}_{294}$				
$\mathcal{C}_{214}$:	$\mathcal{Q}_4 \times \mathcal{Q}_{299}$	$\mathcal{Q}_{12} \times \mathcal{Q}_{12}$	$\mathcal{Q}_{12} \times \mathcal{Q}_{294}$	$\mathcal{Q}_{12} \times \mathcal{Q}_{299}$	$\mathcal{Q}_{293} \times \mathcal{Q}_4$	$\mathcal{Q}_{293} \times \mathcal{Q}_{12}$
	$\mathcal{Q}_{293} \times \mathcal{Q}_{294}$	$\mathcal{Q}_{293} \times \mathcal{Q}_{299}$	$\mathcal{Q}_{294} \times \mathcal{Q}_{12}$	$\mathcal{Q}_{294} \times \mathcal{Q}_{294}$	$\mathcal{Q}_{294} \times \mathcal{Q}_{299}$	
$\mathcal{C}_{215}$:	$\mathcal{Q}_4 \times \mathcal{Q}_{293}$	$\mathcal{Q}_{12} \times \mathcal{Q}_{12}$	$\mathcal{Q}_{12} \times \mathcal{Q}_{293}$	$\mathcal{Q}_{12} \times \mathcal{Q}_{294}$	$\mathcal{Q}_{294} \times \mathcal{Q}_{12}$	$\mathcal{Q}_{294} \times \mathcal{Q}_{293}$
	$\mathcal{Q}_{294} \times \mathcal{Q}_{294}$	$\mathcal{Q}_{299} \times \mathcal{Q}_4$	$\mathcal{Q}_{299} \times \mathcal{Q}_{12}$	$\mathcal{Q}_{299} \times \mathcal{Q}_{293}$	$\mathcal{Q}_{299} \times \mathcal{Q}_{294}$	
$\mathcal{C}_{223}$:	$\mathcal{Q}_{12} \times \mathcal{Q}_{299}$	$\mathcal{Q}_{293} \times \mathcal{Q}_{293}$	$\mathcal{Q}_{293} \times \mathcal{Q}_{294}$	$\mathcal{Q}_{294} \times \mathcal{Q}_{293}$	$\mathcal{Q}_{294} \times \mathcal{Q}_{294}$	$\mathcal{Q}_{299} \times \mathcal{Q}_{12}$
	$\mathcal{Q}_{299} \times \mathcal{Q}_{299}$					
$\mathcal{C}_{262}$:	$\mathcal{Q}_{12} \times \mathcal{Q}_{299}$	$\mathcal{Q}_{294} \times \mathcal{Q}_{299}$	$\mathcal{Q}_{299} \times \mathcal{Q}_{12}$	$\mathcal{Q}_{299} \times \mathcal{Q}_{294}$		
$\mathcal{C}_{266}$:	$\mathcal{Q}_{12} \times \mathcal{Q}_{12}$	$\mathcal{Q}_{294} \times \mathcal{Q}_{299}$	$\mathcal{Q}_{299} \times \mathcal{Q}_{294}$	$\mathcal{Q}_{299} \times \mathcal{Q}_{299}$		
$\mathcal{C}_{296}$:	$\mathcal{Q}_{12} \times \mathcal{Q}_{299}$	$\mathcal{Q}_{293} \times \mathcal{Q}_{293}$	$\mathcal{Q}_{294} \times \mathcal{Q}_{12}$	$\mathcal{Q}_{299} \times \mathcal{Q}_{294}$	$\mathcal{Q}_{299} \times \mathcal{Q}_{299}$	
$\mathcal{C}_{297}$:	$\mathcal{Q}_{12} \times \mathcal{Q}_{294}$	$\mathcal{Q}_{293} \times \mathcal{Q}_{293}$	$\mathcal{Q}_{294} \times \mathcal{Q}_{299}$	$\mathcal{Q}_{299} \times \mathcal{Q}_{12}$	$\mathcal{Q}_{299} \times \mathcal{Q}_{299}$	