Assessment of Hiding the Higher-Order Leakages in Hardware

what are the achievements versus overheads?

Amir Moradi and Alexander Wild

Horst Görtz Institute for IT-Security, Ruhr-Universität Bochum, Germany
{firstname.lastname}@rub.de

Abstract. Higher-order side-channel attacks are becoming amongst the major interests of academia as well as industry sector. It is indeed being motivated by the development of countermeasures which can prevent the leakages up to certain orders. As a concrete example, threshold implementation (TI) as an efficient way to realize Boolean masking in hardware is able to avoid first-order leakages. Trivially, the attacks conducted at second (and higher) orders can exploit the corresponding leakages hence devastating the provided security. Hence, the extension of TI to higher orders was being expected which has been presented at ASIACRYPT 2014. Following its underlying univariate settings it can provide security at higher orders, and its area and time overheads naturally increase with the desired security order.
In this work we look at the feasibility of higher-order attacks on first-order TI from another perspective. Instead of increasing the order of resistance by employing higher-order TIs, we realize the first-order TI designs following the principles of a power-equalization technique dedicated to FPGA platforms, that naturally leads to hardening higher-order attacks. We show that although the first-order TI designs, which are additionally equipped by the power-equalization methodology, have significant area overhead, they can maintain the same throughput and more importantly can avoid the higher-order leakages to be practically exploitable by up to 1 billion traces.

1 Introduction

Side-channel attacks are a major threat to the security of modern embedded devices. If no particular attention is paid, the exploitation of physical leakages such as the power consumption and the electromagnetic radiation of a cryptographic implementation can lead to successful key recoveries, e.g., [2, 16, 27, 43, 57]. As a consequence, the topic has been followed by a vast literature on potential solutions to defeat such attacks.

The countermeasures against side-channel attacks range from ad hoc to formal, and are defined to be applied at various abstraction levels. For instance, time randomizations (based on random delay insertion [14] or shuffling [53]) are frequently-used low-overhead heuristic-based approaches (mainly) for software-based applications. These *hiding* schemes are not limited to only those which randomize the computations in time, but covers the approaches that add noise resources [18,24] as well as those aiming to equalize the power consumption [50,52]. The time randomizations can be overcome by preprocessing the leakage traces (e.g., combing [24]), and the effect of the noise additions can be mitigated by increasing the number of traces [24]. In contrast, the power-equalization techniques usually fail due to wrong assumptions (e.g., ignoring early propagation [49]) or overestimating the ability of the tools (e.g., balanced dual-rail routing [51]). Apart from [29, 52], dual-rail precharge logic styles, which have been initially designed for ASIC-based applications (e.g., [13, 39, 40, 50]), cannot be easily integrated into the FPGAs. Instead, other approaches like [3, 19–21, 23, 35, 46, 56] have particularly been developed with respect to the resources available in certain FPGAs. However, each of such techniques suffers from a flaw that prevents them to be considered as a potential solution (see [55] for details of each flaw). Further, a design methodology which combines a dual-rail logic style and duplication in FPGAs [56] has also been shown to be flawed [54]. As an alternative, the technique presented in [55] (so-called GliFreD) seems to avoid the known pitfalls. It has been designed particularly for Xilinx FPGAs, and aims at avoiding early propagation, preventing the glitches, and relaxing the necessity of a dual-rail routing tool. It seems that GliFreD can satisfy its goals toward equalizing the power consumption, but an ideally-equal situation cannot still be achieved due to the process variation violating the balance between the cloned routes.

On the other hand, probably the most investigated and best understood protection against side-channel attacks is *masking* [12, 15, 45]. The underlying principle of masking is to represent any sensitive variable in the implementation by d shares in such a way that the computations are performed only on these shares. Assuming that the leakage of the shares are independent of each other, a successful key-recovery attack needs to observe – at least – the dth-order statistical moment of the leakage distributions, where the corresponding complexity increases exponentially with d.

However, the independence of leakages associated to the shares is an assumption which is usually violated in hardware applications. As an example, the masked AES Sbox designs [11,38], where the glitches are ignored, failed in practice to satisfy the desired security level, i.e., first-order resistance [25,32]. Instead, based on Boolean masking and multiparty computation, threshold implementations (TI) [36, 37] can ensure first-order resistance in the presence of glitches. Indeed, not only its underlying principles are sound and realistic but also practical investigations confirmed its effectiveness [4,33]. Trivially, higher-order attacks are feasible on TI designs [4,26], which motivated the work presented in [5] where the concept of higher-order TI is demonstrated that extends its definitions to any order. Regardless of its significant overhead (e.g., requiring at least $d = 5$ for a second-order security) the note given in [44] and later practically confirmed in [48] made clear that the definitions of the higher-order TI stand valid only in univariate scenarios.

Our Contribution. Indeed, it is known to the community that hiding techniques (in particular power-equalizing approaches) are not *solely* capable to prevent key-recovery attacks. It is always suggested that such techniques should be combined with other countermeasures, but the benefit of such a combination has never truly been examined for a hardware platform. More precisely, exploiting higher-order leakages becomes extremely hard in practice when the leakage traces are sufficiently noisy [42]. Along the same lines, power-equalization schemes are also expected to reduce the signal (versus the noise) and have the same effect. To the best of our knowledge, the only work which tried to proceed toward this goal is [30], where a flawed masking scheme [11] has been implemented in a glitch-free setting. No particular attention has been payed on equalizing the power hence not a concrete hiding technique.

Our contribution in this work is to examine the benefit of combining two sound hardware-based countermeasures. More precisely, we aim at considering a provably (first-order) secure masking scheme (TI) and realize it under the principles of a proper power-equalizing technique (GliFreD). We pursue an investigation of our combined construction compared with:

– the same masking design (first-order TI) without employing any hiding technique, and
– the second-order TI of the same design excluding any power-equalization scheme.

Such comparisons with respect to the data complexity of leakage detection as well as time and area overheads of the designs allows us to have an overview on the tradeoff between the gains and overheads of different countermeasures as well as their combination.

Since the design overheads are application specific, we consider two design methodologies: first, a fully serialized architecture for lightweight applications with KATAN-32 cipher and second, a parallelized architecture for high-speed applications with PRESENT cipher. Amongst our achievements in this work – including a second-order TI of PRESENT – we can refer to the designs we developed with a combination of GliFreD and the first-order TI (of both KATAN-32 and PRESENT) which showed to be secure by up to 1 billion power traces measured from a Spartan-6 FPGA platform.

2 GliFreD

Dual-rail Precharge Logic (DPL) schemes are popular side-channel countermeasures for hardware circuits and assigned to the group of hiding techniques. Each DPL scheme places two contrary working (*true* and *false*) circuits on a device to *ideally* decorrelate the power consumption from the processed data. In common, DPL schemes have to deal with some implementation challenges. The three major challenges that the FPGA-based DPL designers face are: early propagation, glitches and different wire capacitance of coupled signals. GliFreD is a DPL scheme exclusively designed for FPGAs, and is amongst the few schemes which address all these three problems [55].

To overcome the aforementioned problems GliFreD defines the following design methodology. Each Look-Up Table (LUT) instance is connected to two global control signals: `CLK` and `active`; the later one toggles with half of the other one's frequency. These control signals determine whether the LUTs reside in precharge or in evaluation phase. Hence, the regulated LUT transitions overcome the definition of early evaluation [49]. To prevent the propagation of the LUT output transition, a register is connected to each LUT output. However, a single register stage in a DPL circuit contradicts the requirement of a constant gate and register transition per clock cycle [28] as inconstant and data-dependent transitions would result in data-dependent leakage. Therefore, the GliFreD principles require to place an even number of register stages between each two LUTs connected in the circuit. Consequently, GliFreD forms a pipeline architecture which prevents glitches by halting the propagation of a signal after each LUT. Figure 1(a) shows the timing diagram of a GliFreD circuit.

Similar to many DPL schemes, GliFreD also needs to place a dual of the circuit. Copying the routing structure is currently the best known way in FPGAs to keep the wire capacitances of the *false* circuit as equivalent as those of the *true* circuit. Hence, to perform the circuit dualization, i.e., placing the *false* circuit, a second horizontally-moved instance of the *true* circuit is placed on the FPGA. The copy process is performed on netlist level to pass on the routing information to the *false* circuit.

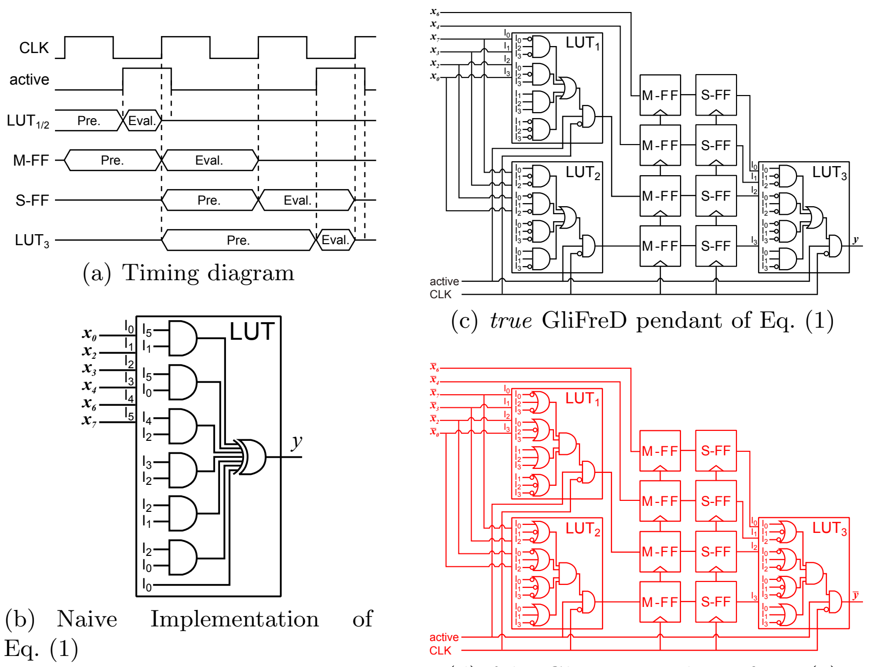
GliFreD allows an arbitrary LUT configuration; since both control signals `CLK` and `active` should be connected to each LUT, the function f each LUT can realize is limited to a 4-to-1 look-up table. The output of each LUT can be seen as $\mathrm{O} = \texttt{active} \cdot \overline{\texttt{CLK}} \cdot f(\mathrm{I}_2, \ldots, \mathrm{I}_5)$[1], while the corresponding dual function (of the *false* circuit) becomes $\overline{\mathrm{O}} = \texttt{active} \cdot \overline{\texttt{CLK}} \cdot \overline{f(\overline{\mathrm{I}_2}, \ldots, \overline{\mathrm{I}_5})}$. Figure 1 shows the GliFreD pendant of an exemplary function

$$y = x_0 + x_0x_3 + x_2x_3 + x_3x_4 + x_3x_6 + x_0x_7 + x_2x_7, \tag{1}$$

whose standard implementation is shown in Figure 1(b).

Since the output of each LUT is buffered by a register, the critical path in a GliFreD circuit is minimized allowing to run the circuit at high frequencies. To this end the delay between the `CLK` and `active` signals should be kept minimum (see Figure 1(a)), that can be achieved by forcing `active` signal to be routed through the clock trees. The GliFreD design methodology offers the ability to transfer a design into a fully-pipelined architecture, hence achieving a high throughput in combination with a high clock frequency. In general, large combinatorial circuits cause glitches which propagate through the whole circuit. Since GliFreD prevents those glitches, it may also reduce the power consumption. In small combinatorial circuits this benefit is faded and dominated by the increased amount of resources the GliFreD circuit utilizes. Nevertheless, GliFreD

[1] I_0 and I_1 are reserved for `CLK` and `active`.



(a) Timing diagram

(b) Naive Implementation of Eq. (1)

(c) *true* GliFreD pendant of Eq. (1)

(d) *false* GliFreD pendant of Eq. (1)

Fig. 1. An exemplary function implemented in a standard 6-to-1 LUT architecture and its GliFreD representation including the timing diagram

is a resource-costly solution. The LUT overhead (at most 8) required to form a GliFreD circuit strongly depends on the original design structure. Compared to the LUT utilization GliFreD causes a massive register overhead and hence an increased latency. The register overhead cannot be trivially estimated and depends on the LUT depth, width and the amount of registers in the original design.

3 Case Studies

Before giving the details of our case studies, we briefly restate the concept behind threshold implementation.

3.1 Threshold Implementation

As stated before, the masking scheme which we consider in this work is threshold implementation (TI) introduced and extended in [4, 5, 36, 37]. Let us denote an intermediate value of a cipher by $\boldsymbol{x}$ made of s single-bit signals $\langle x_1, \ldots, x_s \rangle$. The underlying concept of TI is to use Boolean masking to represent $\boldsymbol{x}$ in a shared form $(\boldsymbol{x}^1, \ldots, \boldsymbol{x}^n)$, where $\boldsymbol{x} = \bigoplus \boldsymbol{x}^i$ and each $\boldsymbol{x}^i$ similarly denotes a vector of

s single-bit signals $\langle x_1^i, \ldots, x_s^i \rangle$. A linear function $l(.)$ can be trivially applied over the shares of $\boldsymbol{x}$ as $l(\boldsymbol{x}) = \bigoplus l(\boldsymbol{x}^i)$. However, the realization of non-linear functions, e.g., an Sbox, over Boolean masked data is challenging. Following the concept of TI, if the algebraic degree of the underlying Sbox is denoted by t and the desired security order by d, the minimum number of shares to realize the Sbox under the TI settings is $n = t\,d + 1$. Further, such a TI Sbox provides the output $\boldsymbol{y} = S(\boldsymbol{x})$ in a shared form $(\boldsymbol{y}^1, \ldots, \boldsymbol{y}^m)$ with at least $m = \binom{n}{t}$ shares. Note that the bit length of $\boldsymbol{x}$ and $\boldsymbol{y}$ (respectively of their shared forms) are not necessary the same since $S(.)$ might be not a bijection, e.g., in case of DES.

Each output share $\boldsymbol{y}^{j \in \{1,\ldots,m\}}$ is given by a component function $f^j(.)$ over a subset of the input shares. To achieve the dth-order security, any d selection of the component functions $f^{j \in \{1,\ldots,m\}}(.)$ should be independent of at least one input share.

Since the security of masking schemes is based on the uniform distribution of the masks, the output of a TI Sbox must be also uniform as it is used as input in further parts of the implementation. To express the *uniformity* under the TI concept suppose that for a certain input $\mathbf{x}$ all possible sharings $\mathcal{X} = \left\{ (\boldsymbol{x}^1, \ldots, \boldsymbol{x}^n) | \mathbf{x} = \bigoplus \boldsymbol{x}^i \right\}$ are given to a TI Sbox. The set made by the output shares, i.e., $\left\{ \left(f^1(.), \ldots, f^m(.)\right) | (\boldsymbol{x}^1, \ldots, \boldsymbol{x}^n) \in \mathcal{X} \right\}$, should be drawn uniformly from the set $\mathcal{Y} = \left\{ (\boldsymbol{y}^1, \ldots, \boldsymbol{y}^m) | \mathbf{y} = \bigoplus \boldsymbol{y}^i \right\}$ as all possible sharings of $\mathbf{y} = S(\mathbf{x})$.

This uniformity check process should be individually performed for $\forall\ \mathbf{x} \in \{0,1\}^s$. We should note that for $d > 1$ where $m > n$ the uniformity cannot be achieved. Hence, some of the registered output shares should be combined to reduce the number of output shares to n. Afterward the uniformity can be examined. For more detailed information we refer to the original articles [5, 37].

3.2 KATAN-32

As stated in Section 2, the overhead and performance of a GliFreD circuit depends on the nature of the underlying application. If the target design is made of small combinatorial circuits, the overhead of the resulting GliFreD circuit is minimal. Therefore, KATAN [10] which benefits from a serialized architecture with very small combinatorial logics is a suitable candidate for our investigations. Further, both first- and second-order uniform TI representation of its non-linear functions are given in [5], allowing us to develop the design with minimal efforts.

The architecture of our designs are based on those given in [5]. Figure 2(a) shows an overview of such a serialized architecture considering KATAN-32 encryption engine with 32-bit plaintext and 80-bit symmetric key. The plaintext and key are serially loaded into the registers, and after 254 clock cycles the ciphertext can be taken from the state register[2]. The first-order TI of KATAN-32 with 3 shares (the minimum settings) needs the state (shift) registers to be

[2] For more detailed information on the construction of functions f_a and f_b in Figure 2(a) see [10] and [5].

tripled. Similar to that of [5], we do not represent the key (and the corresponding shift register) in a shared form. The XOR operations are easily repeated for each share, and the non-linear functions which are limited to the AND/XOR module (involved in function f_a and f_b of Figure 2(a)) need to be realized under the concept of the first-order TI. An AND/XOR function receives a 3-bit input (a, b, c) and gives a single-bit output y as

$$y = a + bc.$$

Following the concept of *direct sharing* [6] the component functions (given in [5]) which realize a uniform first-order TI can be derived as

$$f^{i,j}(\langle a^i, b^i, c^i \rangle, \langle a^j, b^j, c^j \rangle) = a^j + b^j c^j + b^i c^j + b^j c^i, \tag{2}$$

where each output share is made by an instance of such a component function as

$$y^1 = f^{1,2}(.,.), \qquad y^2 = f^{2,3}(.,.), \qquad y^3 = f^{3,1}(.,.).$$

The same procedure is followed to realize the second-order TI of KATAN-32. First, the minimum number of shares is increased to 5, and all state registers and linear functions need to be repeated accordingly. Further, a second-order TI representation of AND/XOR module (given in [5]) can be derived from Equation (2) and the following component function

$$g^{i,j}(\langle a^i, b^i, c^i \rangle, \langle a^j, b^j, c^j \rangle) = b^i c^j + b^j c^i. \tag{3}$$

In such a case, the output shares are made as

$$y^1 = f^{1,2}(.,.), \quad y^2 = f^{1,3}(.,.), \quad y^3 = f^{1,4}(.,.), \quad y^4 = f^{5,1}(.,.), \quad y^5 = f^{2,5}(.,.),$$

and

$$y^6 = g^{2,3}(.,.), \quad y^7 = g^{2,4}(.,.), \quad y^8 = g^{3,4}(.,.), \quad y^9 = g^{3,5}(.,.), \quad y^{10} = g^{4,5}(.,.).$$

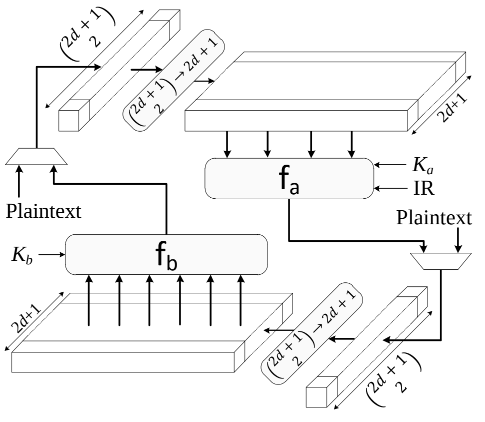
As mentioned before, in a second-order case the output shares should be combined after being registered in order to reduce the number of shares back to 5. In this case, the reduction is done as

$$z^{i \in \{1,\ldots,4\}} = y^i, \qquad z^5 = y^5 + y^6 + y^7 + y^8 + y^9 + y^{10},$$

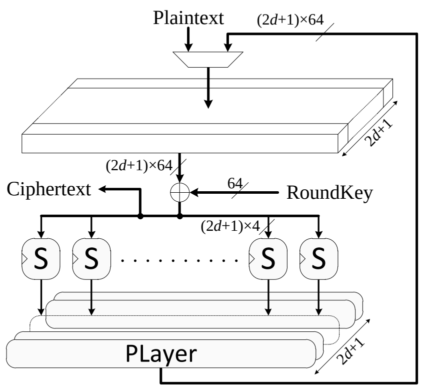
thereby achieving a uniform second-order TI of the AND/XOR module [5]. For more clarification the formula for all the component functions are given in Appendix A.

3.3 PRESENT

As the second target we selected the PRESENT cipher [9] to be implemented in a round-based fashion. As Figure 2(b) shows, 16 instances of the Sbox in



(a) KATAN-32



(b) PRESENT

Fig. 2. Architecture of the case studies, first ($d = 1$) and second ($d = 2$) order TI

addition to the `PLayer` operate in parallel to compute one cipher round. The reason for choosing such a target is to have an application for GliFreD with large combinatorial circuit compared to that of KATAN. Also, due to a possibility to decompose the PRESENT Sbox – as we express below – we are able to develop its uniform first- and second-order TI representations. We should note that we have not selected the AES as a target because its first-order TI (in [4] and [33]) can only be realized by remasking (requiring multiple fresh mask bits per clock cycle) and furthermore there is not yet a clear roadmap how to realize its second-order TI.

Similar to the case of KATAN, the first-order (respectively second-order) TI of the targeted PRESENT architecture employs a 3-share (respectively 5-share) Boolean masking. The `PLayer` (realized by routing in the round-based architecture) is repeated on each share, and the key XOR is applied on only one share as the 80-bit key is not represented in a shared form. Clearly the remaining part is the TI representation of the PRESENT Sbox. Previously Poschmann et al. [41] have shown a decomposition and a uniform first-order TI of such an Sbox. However, below we represent another decomposition allowing us to develop its both first- and second-order uniform TI representations.

The PRESENT Sbox $S(\boldsymbol{x}) = \boldsymbol{y}$ is a cubic bijection (i.e., with algebraic degree $t = 3$) leading to minimum $n = 4$ and $n = 7$ shares in the first- and second-order TI settings respectively. Therefore, it is preferable to decompose the Sbox into two (at most) quadratic bijections F and G, in such a way that $S(\boldsymbol{x}) = F(G(\boldsymbol{x}))$ (i.e., $S = F \circ G$). If so, each F and G can be shared with $n = 3$ and $n = 5$ (for first- and second-order TI). According to the classifications given in [7], the PRESENT Sbox belongs to the cubic class $\mathcal{C}_{266}$. It means that there exist affine transformations A and B, where $S(\boldsymbol{x}) = B(\mathcal{C}_{266}(A(\boldsymbol{x})))$. In other words, S and $\mathcal{C}_{266}$ are affine equivalent. To find the affine functions the algorithm given in [8] can be used; indeed there exist 4 such two affine functions. Also, as stated in [7]

$\mathcal{C}_{266}$ can be decomposed into two quadratic bijections. One of the possibilities is $\mathcal{Q}_{294} \times \mathcal{Q}_{299}$. It means that there exist three affine functions A_1, A_2, A_3, where $\mathcal{C}_{266} = A_3 \circ \mathcal{Q}_{299} \circ A_2 \circ \mathcal{Q}_{294} \circ A_1$. Since $\mathcal{C}_{266}$ and S are affine equivalent, there exist also three affine functions to decompose the PRESENT Sbox as

$$S(\boldsymbol{x}) = A_3\Bigg(\mathcal{Q}_{299}\bigg(A_2\Big(\mathcal{Q}_{294}\big(A_1(\boldsymbol{x})\big)\Big)\bigg)\Bigg). \tag{4}$$

We have found 229,376 such 3-tuple affine bijections, and we have selected one of the most simplest solutions with respect to the number of terms in their Algebraic Normal Form (ANF) directly affecting the size of the corresponding circuit.

The next step is to provide the uniform first-order TI of the quadratic bijections $\mathcal{Q}_{294}$ and $\mathcal{Q}_{299}$ which can be easily achieved by direct sharing [7]. For $\mathcal{Q}_{294}$:`0123456789BAEFDC` we can write

$$e = a + bd, \qquad f = b + cd, \qquad g = c, \qquad h = d, \tag{5}$$

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. The component functions of the first-order TI of $\mathcal{Q}_{294}$ can be derived by $f^{i,j}_{\mathcal{Q}_{294}}(\langle a^i, b^i, c^i, d^i\rangle, \langle a^j, b^j, c^j, d^j\rangle) = \langle e, f, g, h\rangle$ as

$$\begin{aligned} e &= a^i + b^i d^i + d^i b^j + b^i d^j \qquad & g &= c^i \\ f &= b^i + c^i d^i + d^i c^j + c^i d^j \qquad & h &= d^i \end{aligned} \tag{6}$$

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 TI of $\mathcal{Q}_{294}$.

Following the same principle for $\mathcal{Q}_{299}$:`012345678ACEB9FD` as

$$e = a + ad + cd, \quad f = b + ad + bc + cd, \quad g = c + bd + cd, \quad h = d, \tag{7}$$

we can define the component function $f^{i,j}_{\mathcal{Q}_{299}}(\langle a^i, b^i, c^i, d^i\rangle, \langle a^j, b^j, c^j, d^j\rangle) = \langle e, f, g, h\rangle$ as

$$\begin{aligned} e &= a^i + (a^i d^i + d^i a^j + a^i d^j) + (c^i d^i + d^i c^j + c^i d^j) \\ f &= b^i + (a^i d^i + d^i a^j + a^i d^j) + (b^i d^i + d^i b^j + b^i d^j) + (c^i d^i + d^i c^j + c^i d^j) \\ g &= c^i + (b^i d^i + d^i b^j + b^i d^j) + (c^i d^i + d^i c^j + c^i d^j) \\ h &= d^i. \end{aligned} \tag{8}$$

Similarly, three 4-bit output shares provided by $f^{2,3}_{\mathcal{Q}_{299}}(.,.)$, $f^{3,1}_{\mathcal{Q}_{299}}(.,.)$ and $f^{1,2}_{\mathcal{Q}_{299}}(.,.)$ make a uniform first-order TI of $\mathcal{Q}_{299}$.

Since the affine transformations A_1, A_2, A_3 do not change the uniformity and should be applied on each 4-bit share separately, the decomposition in Equation (4) provides a 3-share uniform first-order TI of the PRESENT Sbox. It should be noted that registers are required to be placed between the component

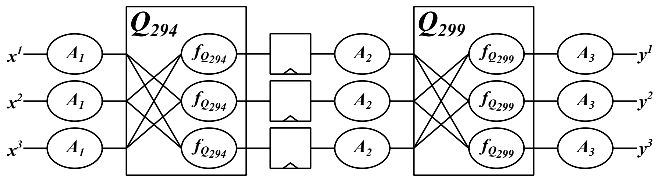


Fig. 3. A first-order TI of the PRESENT Sbox: $S(\boldsymbol{x}) = \boldsymbol{y}$

functions of $\mathcal{Q}_{294}$ and $\mathcal{Q}_{299}$ to avoid the propagation of the glitches (see Figure 3). Note that the affine function A_2 can be freely placed before or after the intermediate register.

For the second-order TI representations in addition to the above expressed component functions, we define $g^{i,j}_{\mathcal{Q}_{294}}(\langle a^i, b^i, c^i, d^i\rangle, \langle a^j, b^j, c^j, d^j\rangle) = \langle e, f, g, h\rangle$ as

$$\begin{aligned} e &= d^i b^j + b^i d^j & \qquad g &= 0 \\ f &= d^i c^j + c^i d^j & h &= 0. \end{aligned} \tag{9}$$

The 4-bit output shares $\boldsymbol{y}^{i\in\{1,\ldots,10\}}$ are provided by

$$\begin{aligned} &\boldsymbol{y}^1 = f^{2,3}_{\mathcal{Q}_{294}}(.,.), \quad \boldsymbol{y}^2 = f^{3,4}_{\mathcal{Q}_{294}}(.,.), \quad \boldsymbol{y}^3 = f^{4,5}_{\mathcal{Q}_{294}}(.,.), \quad \boldsymbol{y}^4 = f^{5,1}_{\mathcal{Q}_{294}}(.,.), \\ &\boldsymbol{y}^5 = f^{1,2}_{\mathcal{Q}_{294}}(.,.), \quad \boldsymbol{y}^6 = g^{2,4}_{\mathcal{Q}_{294}}(.,.), \quad \boldsymbol{y}^7 = g^{3,5}_{\mathcal{Q}_{294}}(.,.), \quad \boldsymbol{y}^8 = g^{1,4}_{\mathcal{Q}_{294}}(.,.), \\ &\qquad\qquad \boldsymbol{y}^9 = g^{2,5}_{\mathcal{Q}_{294}}(.,.), \quad \boldsymbol{y}^{10} = g^{1,3}_{\mathcal{Q}_{294}}(.,.). \end{aligned} \tag{10}$$

After a clock cycle, when $\boldsymbol{y}^{i\in\{1,\ldots,10\}}$ are stores in dedicate registers, the output shares should be combined as

$$\boldsymbol{z}^{i\in\{1,\ldots,5\}} = \boldsymbol{y}^i + \boldsymbol{y}^{i+5}, \tag{11}$$

which provides the uniform second-order TI of $\mathcal{Q}_{294}$.

The same procedure is valid in case of $\mathcal{Q}_{299}$ considering the component function $g^{i,j}_{\mathcal{Q}_{299}}(\langle a^i, b^i, c^i, d^i\rangle, \langle a^j, b^j, c^j, d^j\rangle) = \langle e, f, g, h\rangle$ as

$$\begin{aligned} e &= d^i a^j + d^i c^j + a^i d^j + c^i d^j \\ f &= d^i a^j + d^i b^j + d^i c^j + a^i d^j + b^i d^j + c^i d^j \\ g &= d^i b^j + d^i c^j + b^i d^j + c^i d^j \\ h &= 0. \end{aligned} \tag{12}$$

By changing the indices from ${}_{\mathcal{Q}_{294}}$ to ${}_{\mathcal{Q}_{299}}$ in Equations (10) and later applying the reduction in Equation (11), a uniform second-order TI of $\mathcal{Q}_{299}$ is achieved. Hence by means of these component functions in addition to the affine transformations, we can realize a uniform second-order TI of the PRESENT Sbox. Figure 4 shows the graphical view of such a construction, and all the required formulas are given in Appendix B. Note that the registers after the affine function A_2 can instead be place before A_2 right after the reduction from 10 to 5 shares.

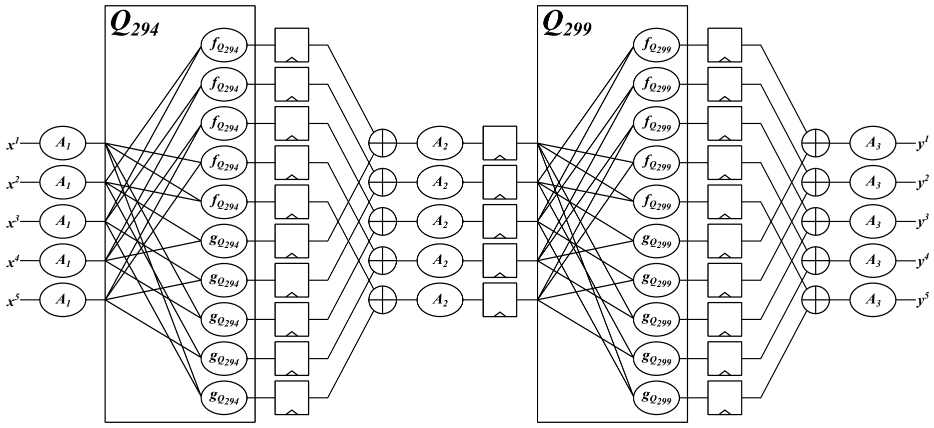


Fig. 4. A second-order TI of the PRESENT Sbox: $S(\boldsymbol{x}) = \boldsymbol{y}$

3.4 Implementation

Based on the specifications given above and considering a Spartan-6 FPGA (indeed the XC6SLX75 of SAKURA-G [1]) we implemented six designs. The first three ones are different profiles of KATAN-32, and the next three designs realize the encryption of PRESENT with a round-based architecture. For each of the targeted cipher we implemented

- the first-order TI, i.e., `KATAN-1st` and `PRESENT-1st` profiles,
- the second-order TI, i.e., `KATAN-2nd` and `PRESENT-2nd` profiles, and
- the first-order TI with GliFreD, i.e., `KATAN-1st-G` and `PRESENT-1st-G` profiles.

Although we did not consider any constraints on placement and routing of the four non-GliFreD profiles, following the principles of GliFreD the corresponding profiles have been realized by first defining an area on the target FPGA, where the component of the *true* part of the GliFreD circuit should be placed. After finishing the placement and routing, the corresponding dual circuit, i.e., the *false* part of the GliFreD circuit, has been cloned and dualized by means of the RapidSmith tool [22]. As a reference, the circuits shown in Figure 1 are the normal and GliFreD realizations of the least significant bit e of Equation (8).

Due to its serialized ring architecture, the `KATAN-1st-G` profile does not form a pipeline. The most important difference between such a profile and its original one (`KATAN-1st`) is on the one hand the number of required clock cycles to finish an encryption (i.e., latency) which is doubled and on the other hand the raised achievable clock frequency due to the minimal LUT depth. The max LUT depth in GliFreD circuits is 1, hence a very short critical path. However, the `PRESENT-1st-G` profile is implemented in a fully-pipelined way, so that the round-based architecture is able to hold 11 different cipher states. Hence, after $32 \times 11 \times 2 = 704$ clock cycles, 11 encryptions with the same key are performed.

Table 1. Details about the implemented profiles. The values given in this table are taken from the post route synthesis report of Xilinx ISE 14.7.

Profile	**Resources**		**Frequency**	**Latency**	**Pipeline**	**Throughput**
	LUT	FF	(MHz)	(#clock)	(stage)	(Mbit/s)
`KATAN-1st`	34	96	225.38	273	1	26.42
`KATAN-2nd`	65	180	321.54	273	1	37.69
`KATAN-1st-G`	114	548	438.21	546	1	25.68
`PRESENT-1st`	808	384	206.61	64	2	413.22
`PRESENT-2nd`	2245	1680	203.46	128	4	406.92
`PRESENT-1st-G`	5442	12672	458.09	704	11	458.09

The pipelined architecture naturally increases the register utilization of the components but provides a much higher throughput.

Table 1 compares the overhead and performance of different design profiles. It indeed gives an overview on the disadvantage (area and time overheads) as well as the advantage (throughput) of employing GliFreD with respect to two different design architectures, i.e., a fully-serialized one which is register oriented (`KATAN-1st-G`) and a round-based one which is combinatorial oriented (`PRESENT-1st-G`). As shown by Table 1, although the resource utilization and the latency of the GliFreD profiles are drastically increased, the throughput is still kept comparable with the original design profiles. Such achievements are mainly due to the naturally-minimized critical paths in the GliFreD designs allowing a high clock frequency.

4 Empirical Results

In addition to the performance and overhead figures given in Section 3.4, we practically examined the ability of each of our six developed designs to avoid side-channel leakages.

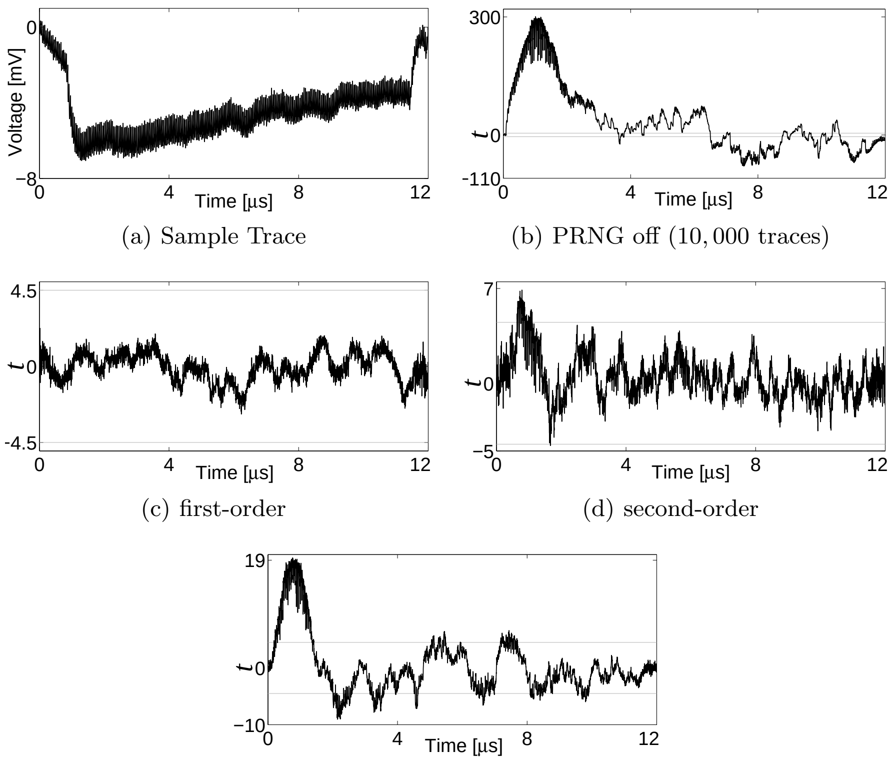
Setup. The experimental platform is a SAKURA-G [1] equipped with a Xilinx Spartan-6 FPGA. The side-channel leakages have been measured by collecting power consumption traces of the underlying FPGA by means of a Teledyne LeCroy HRO 66Zi digital oscilloscope at a sampling frequency of 500 MS/s and a limited bandwidth of 20 MHz. Due to the low peak-to-peak amplitude of the signals we also made use of the amplifier embedded on the SAKURA board. For all six design profiles, the target FPGA operated at a frequency of 24 MHz during the collection of the power traces. Our intuition on the measured power traces from our platform is that the traces are heavily filtered by the measurement setup including the shunt resistor, chip packaging, printed circuit board (PCB), and probes. Measuring the power traces with high bandwidth ($>$ 20 MHz) leads to higher electrical noise. We have examined this behavior and observed leakages easier when the bandwidth is limited. Note that this intuition does not hold true in case of EM measurements.

It is noteworthy that such a frequency of operation has intentionally been taken in order to : i) cover the full power trace length in the measurements as the KATAN profiles need 254 clock cycles after data being loaded (respectively 508 for `KATAN-1st-G`), and ii) cause the power peaks of adjacent clock cycles slightly overlap each other. The later has been considered with respect to the note given in [44] that the second-order TI can still be vulnerable to a second-order bivariate attack. Recalling the techniques introduced in [31], employing certain amplifiers or running the device at a high clock frequency leads to converting multivariate leakages to univariate. It has been shown in [48] that a second-order TI design actually can exhibit a univariate second-order leakage if the measurement setup is employed by certain components, e.g, DC blockers and/or amplifiers. Hence, operating the device at 24 MHz allows us to easily cover the long traces in the measurements and provide particular situations, where second-order TI profiles may demonstrate second-order leakage.

Evaluation. As the evaluation metric we employed the leakage assessment methodology of [17, 47] which is based on the Student's t-test. The reason for such a choice is twofold. First, the t-test can examine the existence of detectable leakages without performing any key-recovery attack, which significantly eases the evaluation process particularly where higher-order leakages using millions of traces should be examined. Moreover, the efficiency of the state-of-the-art key-recovery attacks strongly depends on the targeted intermediate value and the underlying (power) model. Second, the same leakage assessment technique (more precisely the non-specific t-test also known as *fixed vs. random* test) has been used to examine the resistance of different threshold implementations (for example see [5] and [48]). In order to keep our evaluations comparable with the former ones, we trivially employed the same evaluation method.

In a non-specific t-test the leakages associated to a fixed input (plaintext in case of encryption) are compared to that of random inputs while the key in all the measurements is kept constant. Such a test gives a level of confidence to conclude that the leakages related to the process of the fixed input are different to those of the random inputs. If so, an attack is expected to be feasible to exploit the leakage and recover the secrets. For more detailed information we refer the interested reader to [17] and [5].

It is noteworthy that all the tests we performed here are based on a univariate scenario. In other words, we did not run any combination function on different sample points of each collected power trace. Further, we followed the same principle explained in [5,47] to conduct the tests at higher orders. It means that we made the power traces mean-free squared (at each sample point independently), i.e., $(X-\mu)^2$ for the second-order evaluations, and standardized cubed, i.e., $\left(\frac{X-\mu}{\sigma}\right)^3$ for the third-order evaluations. In general, the pre-processing is done by $\left(\frac{X-\mu}{\sigma}\right)^d$ for the analyses at order $d > 2$, with X as a random variable denoting the power traces (at a particular sample point), μ and σ^2 as the sample mean and sample variance (at the same sample point) respectively.



(a) Sample Trace (b) PRNG off (10, 000 traces)

(c) first-order (d) second-order

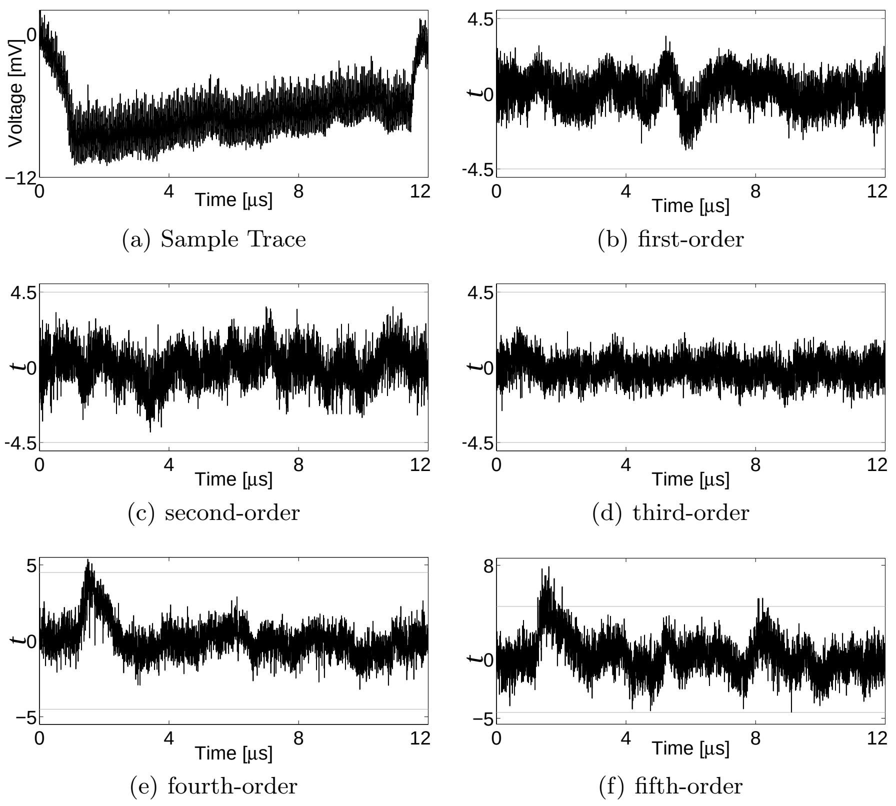
(e) third-order

Fig. 5. `KATAN-1st` profile, sample trace and non-specific t-test results using $1,000,000$ traces

Indeed, these pre-processes required for higher-order evaluations are with the respect to the centered and standardized higher-order statistical moments (for more information see [26, 34]).

We start our evaluations with `KATAN-1st` profile. Figure 5(a) shows a corresponding sample power trace. Note that the collected power traces do not cover a time period, when plaintext and key are serially loaded into the shift registers. In order to have an overview about the quality of the measurement setup and verify the employed evaluation metric, for the first analysis we turned the PRNG off thereby forcing all masks to zero, used for sharing the plaintexts. As shown by Figure 5(b), the first-order t-test shows clear detectable leakages using a few $10,000$ traces. By keeping the PRNG active and conducting the same non-specific t-tests up to third-order using $1,000,000$ traces we observed the curves shown by Figure 5, which indeed confirm the first-order resistance and vulnerability at the second and third orders, as expected.

For the `KATAN-2nd` profile we had to collect much more traces to be able to observe the higher-order leakages. It is due to the high order of sharing, i.e., at



(a) Sample Trace (b) first-order (c) second-order (d) third-order (e) fourth-order (f) fifth-order

Fig. 6. `KATAN-2nd` profile, sample trace and non-specific t-test results using $100,000,000$ traces

least 5 shares (see Section 3.1) in case of a second-order TI. In fact, we observed the fourth- and fifth-order leakages using approximately $100,000,000$ traces, as shown in Figure 6. However, in order to examine the issue reported in [44] (by operating the target at 24 MHz) we continued the collection of the traces up to $500,000,000$, but we have not observed any second-order leakage while the fourth- and fifth-order leakages became detectable – expectedly – with higher confidence. We should here refer to the issue addressed in [44] and the detectable second-order leakage reported in [48]. Based on the explanations of [44] a second-order bivariate leakage should be detectable, but such a bivariate leakage is not necessarily detectable from the consecutive clock cycles, that can additively be combined by means of an amplifier or running the device at a high clock frequency [31]. In case of the application of [48] apparently the consecutive clock cycles exhibit such a bivariate leakage, but it is not hold true for the serialized KATAN architecture. Further, compared to our design profiles the constructions in [48] make use of a kind of remasking which is a different methodology to ensure the uniformity.

Following the same scenario we performed the evaluations on the `KATAN-1st-G` profile and collected $1,000,000,000$ traces to perform the same t-tests at up to third order. The corresponding results which are depicted in Figure 7 indeed confirm the effectiveness of the underlying hiding technique to significantly harden the higher-order attacks. The result of this profile can be compared to that of the `KATAN-1st` profile (Figure 5), where $1,000,000$ traces are adequate to observe the second- and third-order leakages.

The same leakage assessment technique has been conducted on the three profiles of the round-based PRESENT architecture, and the corresponding results are shown in Figure 8, Figure 9 and Figure 10. For the `PRESENT-1st` profile we required $10,000,000$ trace to observe the second- and third-order leakages. Respectively $300,000,000$ traces were necessary for the `PRESENT-2nd` profile to exhibit fourth- and fifth-order leakages. We should again bring the reader's attention to the infeasibility to observe a second-order leakage from the `PRESENT-2nd` profile. We indeed continued our evaluations on this profile by measuring $1,000,000,000$ traces as well as with different fixed inputs (with respect to the non-specific t-tests), but in none of the tests we observed a detectable second-order leakage. As an example, we give the results of one of such tests with $1,000,000,000$ traces in Appendix C, where the third-order leakage also becomes detectable. Finally, similar to the KATAN GliFreD design we collected $1,000,000,000$ traces and conducted the same non-specific t-tests on the `PRESENT-1st-G` profile, which still shows robustness to avoid the leakages to be detectable at first, second, and third orders.

Discussion. Comparing the presented practical results, at the first glance it can be noticed that the GliFreD profiles consume more energy than the other corresponding profiles. They also increase the number of required clock cycles (latency) particularly in case of the PRESENT design as its combinatorial circuit has a longer depth compared to the KATAN design. However, their achievement, i.e., hiding the higher-order leakages to make the higher-order attacks *practically* infeasible, is confirmed. Hence, it can be concluded that the combination of such a power-equalization technique and a proper masking scheme (i.e, first-order TI) gives a high level of confidence to argue the practical infeasibility of the key-recovery attacks.

Our comparisons are limited to the second-order TI of KATAN and PRESENT, which can be extended to higher-order TI designs. However, by increasing the desired order of security the number of shares and the required internal PRNGs respectively increase (e.g., at least 7 and 9 shares for third- and fourth-order TI). Note that the numbers given in Table 1 exclude the area required for the PRNGs.

Nonetheless, due to the local separation of false and true parts in GliFreD circuits, the resistance of our proposed method against higher-order EM attacks is still an open question and should be addressed in the future. Further, GliFreD is exclusively designed for FPGAs and uses the fixed LUT structure to realize Boolean functions of a circuit. Transforming this logic style naively to ASIC may not lead to the expected results especially with respect to the area overhead. The idea of combining TI with DPL styles can be adopted for ASICs by employing one of the logic styles designed for ASICs in addition to a customized router.

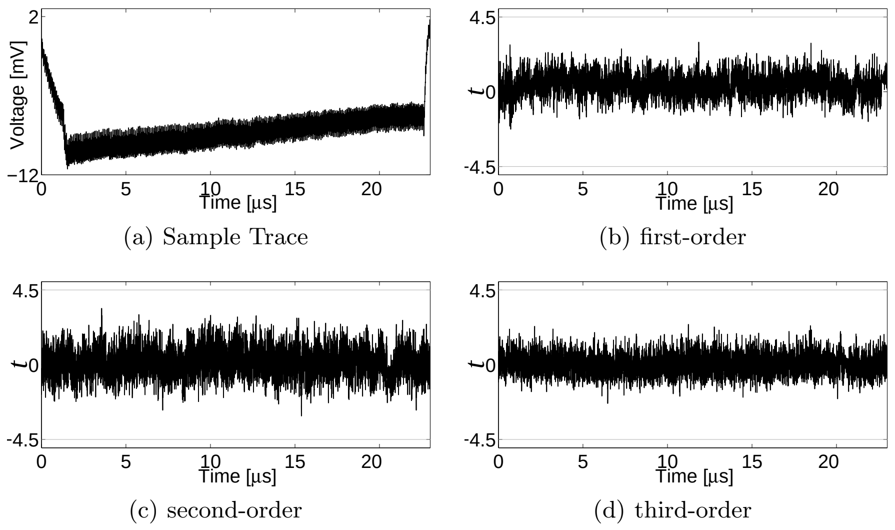


Fig. 7. `KATAN-1st-G` profile, sample trace and non-specific t-test results using $1,000,000,000$ traces

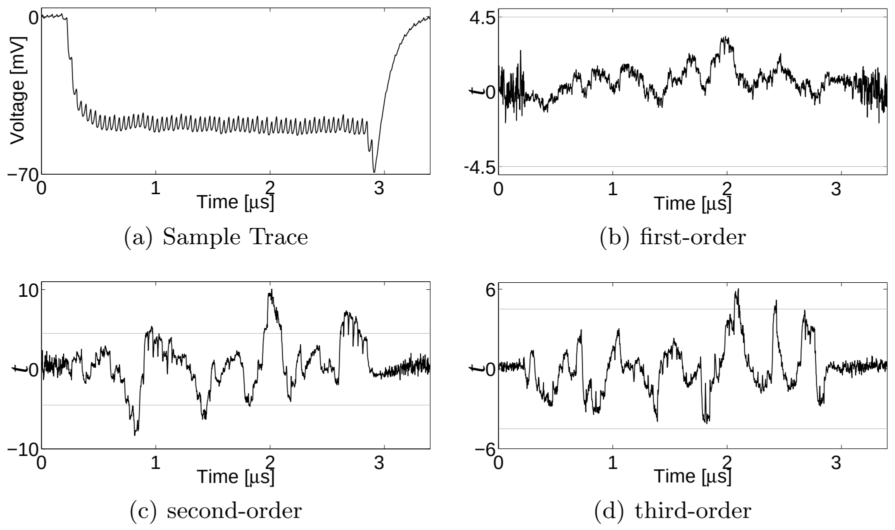


Fig. 8. `PRESENT-1st` profile, sample trace and non-specific t-test results using $10,000,000$ traces

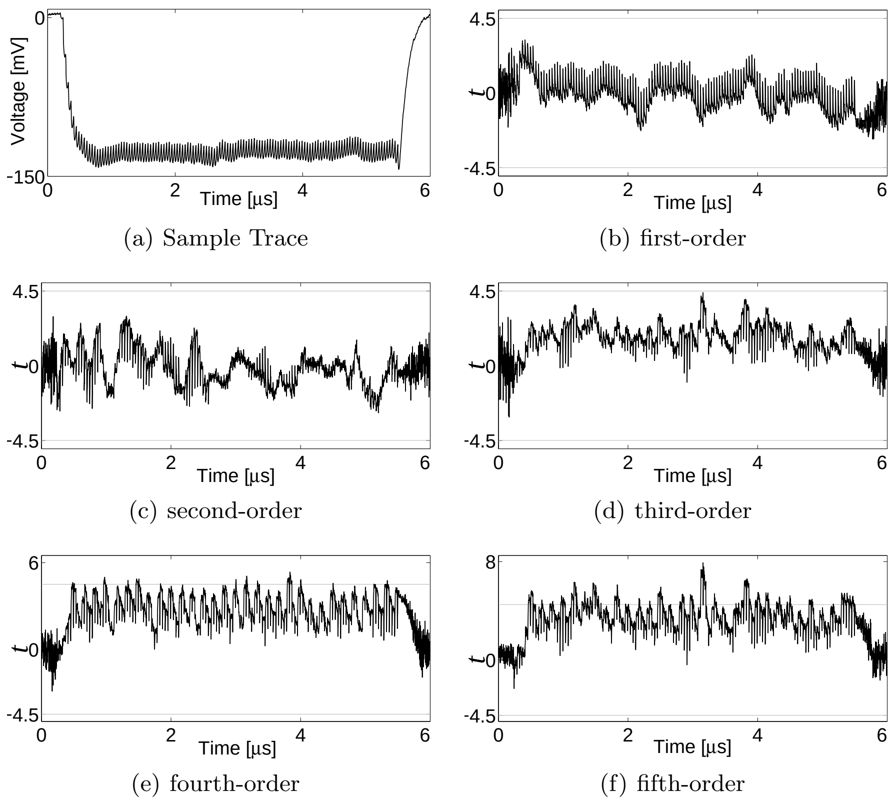


Fig. 9. `PRESENT-2nd` profile, sample trace and non-specific *t*-test results using 300, 000, 000 traces

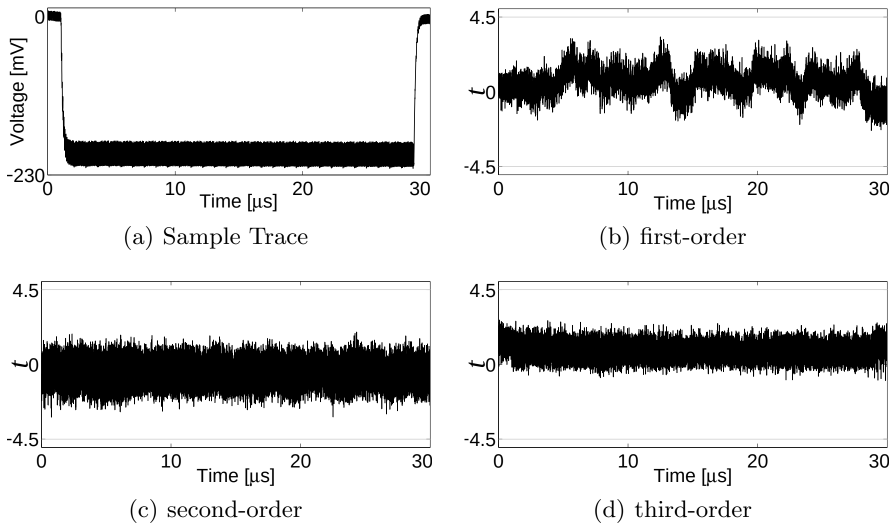


Fig. 10. `PRESENT-1st-G` profile, sample trace and non-specific *t*-test results using 1, 000, 000, 000 traces

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. 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.
4. 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.
5. 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.
6. 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.
7. 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.
8. A. Biryukov, C. D. Cannière, A. Braeken, and B. Preneel. A Toolbox for Cryptanalysis: Linear and Affine Equivalence Algorithms. In *Advances in Cryptology - EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 33–50. Springer, 2003.
9. 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.
10. C. D. Cannière, O. Dunkelman, and M. Knezevic. KATAN and KTANTAN - A Family of Small and Efficient Hardware-Oriented Block Ciphers. In *Cryptographic Hardware and Embedded Systems - CHES 2009*, volume 5747 of *Lecture Notes in Computer Science*, pages 272–288. Springer, 2009.
11. D. Canright and L. Batina. A Very Compact "Perfectly Masked" S-Box for AES. In *Applied Cryptography and Network Security - ACNS 2008*, volume 5037 of *Lecture Notes in Computer Science*, pages 446–459, 2008.
12. S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi. Towards Sound Approaches to Counteract Power-Analysis Attacks. In *Advances in Cryptology - CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.
13. Z. Chen and Y. Zhou. Dual-Rail Random Switching Logic: A Countermeasure to Reduce Side Channel Leakage. In *Cryptographic Hardware and Embedded Systems - CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, pages 242–254. Springer, 2006.
14. J. Coron and I. Kizhvatov. An Efficient Method for Random Delay Generation in Embedded Software. In *Cryptographic Hardware and Embedded Systems - CHES 2009*, volume 5747 of *Lecture Notes in Computer Science*, pages 156–170. Springer, 2009.

15. A. Duc, S. Dziembowski, and S. Faust. Unifying Leakage Models: From Probing Attacks to Noisy Leakage. In *Advances in Cryptology - EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 423–440. Springer, 2014.
16. 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.
17. 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`.
18. T. Güneysu and A. Moradi. Generic Side-Channel Countermeasures for Reconfigurable Devices. In *Cryptographic Hardware and Embedded Systems - CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 33–48. Springer, 2011.
19. W. He, E. de la Torre, and T. Riesgo. A Precharge-Absorbed DPL Logic for Reducing Early Propagation Effects on FPGA Implementations. In *Reconfigurable Computing and FPGAs - ReConFig 2011*, pages 217–222. IEEE Computer Society, 2011.
20. W. He, A. Otero, E. de la Torre, and T. Riesgo. Automatic generation of identical routing pairs for FPGA implemented DPL logic. In *Reconfigurable Computing and FPGAs - ReConFig 2012*, pages 1–6. IEEE Computer Society, 2012.
21. J. Kaps and R. Velegalati. DPA Resistant AES on FPGA Using Partial DDL. In *Field-Programmable Custom Computing Machines - FCCM 2010*, pages 273–280. IEEE Computer Society, 2010.
22. C. Lavin, M. Padilla, J. Lamprecht, P. Lundrigan, B. Nelson, B. Hutchings, and M. Wirthlin. RapidSmith – A Library for Low-level Manipulation of Partially Placed-and-Routed FPGA Designs. Technical report, Brigham Young University, September 2012.
23. V. Lomné, P. Maurine, L. Torres, M. Robert, R. Soares, and N. Calazans. Evaluation on FPGA of triple rail logic robustness against DPA and DEMA. In *Design, Automation and Test in Europe - DATE 2009*, pages 634–639. IEEE Computer Society, 2009.
24. S. Mangard, E. Oswald, and T. Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards.* Springer, 2007.
25. S. Mangard, N. Pramstaller, and E. Oswald. Successfully Attacking Masked AES Hardware Implementations. In *Cryptographic Hardware and Embedded Systems - CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*, pages 157–171. Springer, 2005.
26. A. Moradi. Statistical Tools Flavor Side-Channel Collision Attacks. In *Advances in Cryptology - EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 428–445. Springer, 2012.
27. 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.
28. A. Moradi, T. Eisenbarth, A. Poschmann, and C. Paar. Power Analysis of Single-Rail Storage Elements as Used in MDPL. In *Information, Security and Cryptology - ICISC 2009*, volume 5984 of *Lecture Notes in Computer Science*, pages 146–160. Springer, 2009.

29. A. Moradi and V. Immler. Early Propagation and Imbalanced Routing, How to Diminish in FPGAs. In *Cryptographic Hardware and Embedded Systems - CHES 2014*, volume 5984 of *Lecture Notes in Computer Science*, pages 598–615. Springer, 2014.
30. A. Moradi and O. Mischke. Glitch-free implementation of masking in modern FPGAs. In *Hardware-Oriented Security and Trust - HOST 2012*, pages 89–95. IEEE, 2012.
31. A. Moradi and O. Mischke. On the Simplicity of Converting Leakages from Multivariate to Univariate - (Case Study of a Glitch-Resistant Masking Scheme). In *Cryptographic Hardware and Embedded Systems - CHES 2013*, volume 8086 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2013.
32. A. Moradi, O. Mischke, and T. Eisenbarth. Correlation-Enhanced Power Analysis Collision Attack. In *Cryptographic Hardware and Embedded Systems, CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 125–139. Springer, 2010.
33. 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.
34. A. Moradi and F.-X. Standaert. Moments-Correlating DPA. Cryptology ePrint Archive, Report 2014/409, 2014. `http://eprint.iacr.org/`.
35. M. Nassar, S. Bhasin, J. Danger, G. Duc, and S. Guilley. BCDL: A high speed balanced DPL for FPGA with global precharge and no early evaluation. In *Design, Automation and Test in Europe - DATE 2010*, pages 849–854. IEEE Computer Society, 2010.
36. S. Nikova, V. Rijmen, and M. Schläffer. Secure Hardware Implementation of Nonlinear Functions in the Presence of Glitches. In *Information Security and Cryptology - ICISC 2008*, volume 5461 of *Lecture Notes in Computer Science*, pages 218–234. Springer, 2009.
37. 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.
38. E. Oswald, S. Mangard, N. Pramstaller, and V. Rijmen. A Side-Channel Analysis Resistant Description of the AES S-Box. In *Fast Software Encryption - FSE 2005*, volume 3557 of *Lecture Notes in Computer Science*, pages 413–423. Springer, 2005.
39. T. Popp, M. Kirschbaum, T. Zefferer, and S. Mangard. Evaluation of the Masked Logic Style MDPL on a Prototype Chip. In *Cryptographic Hardware and Embedded Systems - CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 81–94. Springer, 2007.
40. T. Popp and S. Mangard. Masked Dual-Rail Pre-charge Logic: DPA-Resistance Without Routing Constraints. In *Cryptographic Hardware and Embedded Systems - CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*, pages 172–186. Springer, 2005.
41. 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.
42. E. Prouff, M. Rivain, and R. Bevan. Statistical Analysis of Second Order Differential Power Analysis. *IEEE Trans. Computers*, 58(6):799–811, 2009.
43. J. R. Rao, P. Rohatgi, H. Scherzer, and S. Tinguely. Partitioning Attacks: Or How to Rapidly Clone Some GSM Cards. In *IEEE Symposium on Security and Privacy*, pages 31–41. IEEE Computer Society, 2002.
44. O. Reparaz. A note on the security of Higher-Order Threshold Implementations. Cryptology ePrint Archive, Report 2015/001, 2015. `http://eprint.iacr.org/`.

45. M. Rivain and E. Prouff. Provably Secure Higher-Order Masking of AES. In *Cryptographic Hardware and Embedded Systems, CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2010.
46. L. Sauvage, M. Nassar, S. Guilley, F. Flament, J. Danger, and Y. Mathieu. DPL on Stratix II FPGA: What to Expect? In *Reconfigurable Computing and FPGAs - ReConFig 2009*, pages 243–248. IEEE Computer Society, 2009.
47. T. Schneider and A. Moradi. Leakage Assessment Methodology - a clear roadmap for side-channel evaluations. In *Cryptographic Hardware and Embedded Systems - CHES 2015*, Lecture Notes in Computer Science. Springer, 2015.
48. T. Schneider, A. Moradi, and T. Güneysu. Arithmetic Addition over Boolean Masking - Towards First- and Second-Order Resistance in Hardware. In *Applied Cryptography and Network Security - ACNS 2015*, Lecture Notes in Computer Science, pages 517–536. Springer, 2015.
49. 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.
50. K. Tiri, M. Akmal, and I. Verbauwhede. A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards. In *ESSCIRC 2002*, pages 403–406, 2002.
51. K. Tiri, D. D. Hwang, A. Hodjat, B. Lai, S. Yang, P. Schaumont, and I. Verbauwhede. Prototype IC with WDDL and Differential Routing - DPA Resistance Assessment. In *Cryptographic Hardware and Embedded Systems - CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*, pages 354–365. Springer, 2005.
52. 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.
53. N. Veyrat-Charvillon, M. Medwed, S. Kerckhof, and F. Standaert. Shuffling against Side-Channel Attacks: A Comprehensive Study with Cautionary Note. In *Advances in Cryptology - ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 740–757. Springer, 2012.
54. A. Wild, A. Moradi, and T. Güneysu. Evaluating the Duplication of Dual-Rail Precharge Logics on FPGAs. In *Constructive Side-Channel Analysis and Secure Design - COSADE 2015*, volume 9064 of *Lecture Notes in Computer Science*. Springer, 2015.
55. A. Wild, A. Moradi, and T. Güneysu. GliFreD: Glitch-Free Duplication - Towards Power-Equalized Circuits on FPGAs. Cryptology ePrint Archive, Report 2015/124, 2015. `http://eprint.iacr.org/`.
56. P. Yu and P. Schaumont. Secure FPGA circuits using controlled placement and routing. In *Hardware/Software Codesign and System Synthesis - CODES+ISSS 2007*, pages 45–50, 2007.
57. 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 AND/XOR Module

A.1 First-order Component Functions

$$y^1 = f^{1,2}(\langle a^1, b^1, c^1\rangle, \langle a^2, b^2, c^2\rangle) = a^2 + b^2c^2 + b^1c^2 + b^2c^1 \tag{13}$$
$$y^2 = f^{2,3}(\langle a^2, b^2, c^2\rangle, \langle a^3, b^3, c^3\rangle) = a^3 + b^3c^3 + b^2c^3 + b^3c^2 \tag{14}$$
$$y^3 = f^{3,1}(\langle a^3, b^3, c^3\rangle, \langle a^1, b^1, c^1\rangle) = a^1 + b^1c^1 + b^1c^3 + b^3c^1 \tag{15}$$

A.2 Second-order Component Functions

$$y^1 = f^{1,2}(\langle a^1, b^1, c^1\rangle, \langle a^2, b^2, c^2\rangle) = a^2 + b^2c^2 + b^1c^2 + b^2c^1 \tag{16}$$
$$y^2 = f^{1,3}(\langle a^1, b^1, c^1\rangle, \langle a^3, b^3, c^3\rangle) = a^3 + b^3c^3 + b^1c^3 + b^3c^1 \tag{17}$$
$$y^3 = f^{1,4}(\langle a^1, b^1, c^1\rangle, \langle a^4, b^4, c^4\rangle) = a^4 + b^4c^4 + b^1c^4 + b^4c^1 \tag{18}$$
$$y^4 = f^{5,1}(\langle a^5, b^5, c^5\rangle, \langle a^1, b^1, c^1\rangle) = a^1 + b^1c^1 + b^5c^1 + b^1c^5 \tag{19}$$
$$y^5 = f^{2,5}(\langle a^2, b^2, c^2\rangle, \langle a^5, b^5, c^5\rangle) = a^5 + b^5c^5 + b^2c^5 + b^5c^2 \tag{20}$$
$$y^6 = g^{2,3}(\langle a^2, b^2, c^2\rangle, \langle a^3, b^3, c^3\rangle) = b^2c^3 + b^3c^2 \tag{21}$$
$$y^7 = g^{2,4}(\langle a^2, b^2, c^2\rangle, \langle a^4, b^4, c^4\rangle) = b^2c^4 + b^4c^2 \tag{22}$$
$$y^8 = g^{3,4}(\langle a^3, b^3, c^3\rangle, \langle a^4, b^4, c^4\rangle) = b^3c^4 + b^4c^3 \tag{23}$$
$$y^9 = g^{3,5}(\langle a^3, b^3, c^3\rangle, \langle a^5, b^5, c^5\rangle) = b^3c^5 + b^5c^3 \tag{24}$$
$$y^{10} = g^{4,5}(\langle a^4, b^4, c^4\rangle, \langle a^5, b^5, c^5\rangle) = b^4c^5 + b^5c^4 \tag{25}$$

B PRESENT Sbox

B.1 Decomposition

The PRESENT Sbox $S(\langle a, b, c, d\rangle) = \langle e, f, g, h\rangle$:`c56b90ad3ef84712` with ANF

$$\begin{aligned} e &= a + c + d + bc \\ f &= b + d + bd + cd + abc + abd + acd \\ g &= 1 + c + d + ab + ad + bd + abd + acd \\ h &= 1 + a + b + d + bc + abc + abd + acd \end{aligned}$$

is decomposed to $A_3 \circ \mathcal{Q}_{299} \circ A_2 \circ \mathcal{Q}_{294} \circ A_1$ with
A_1:`04bf8c37269dae15` as

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

A_2:`91e63b4cd5a27f08` as

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

and A_3:`0145efab89cd6723` as

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

B.2 First-order Component Functions

The first stage $\mathcal{Q}_{294}$

$$\begin{aligned}
\boldsymbol{y}^1 =& f^{2,3}_{\mathcal{Q}_{294}}(\langle a^2,b^2,c^2,d^2\rangle,\langle a^3,b^3,c^3,d^3\rangle)=\langle e,f,g,h\rangle\\
&e=a^2+b^2d^2+d^2b^3+b^2d^3 \qquad g=c^2\\
&f=b^2+c^2d^2+d^2c^3+c^2d^3 \qquad h=d^2 \qquad (26)\\
\boldsymbol{y}^2 =& f^{3,1}_{\mathcal{Q}_{294}}(\langle a^3,b^3,c^3,d^3\rangle,\langle a^1,b^1,c^1,d^1\rangle)=\langle e,f,g,h\rangle\\
&e=a^3+b^3d^3+d^3b^1+b^3d^1 \qquad g=c^3\\
&f=b^3+c^3d^3+d^3c^1+c^3d^1 \qquad h=d^3 \qquad (27)\\
\boldsymbol{y}^3 =& f^{1,2}_{\mathcal{Q}_{294}}(\langle a^1,b^1,c^1,d^1\rangle,\langle a^2,b^2,c^2,d^2\rangle)=\langle e,f,g,h\rangle\\
&e=a^1+b^1d^1+d^1b^2+b^1d^2 \qquad g=c^1\\
&f=b^1+c^1d^1+d^1c^2+c^1d^2 \qquad h=d^1 \qquad (28)
\end{aligned}$$

The second stage $\mathcal{Q}_{299}$

$$\begin{aligned}
\boldsymbol{y}^1 =& f^{2,3}_{\mathcal{Q}_{299}}(\langle a^2,b^2,c^2,d^2\rangle,\langle a^3,b^3,c^3,d^3\rangle)=\langle e,f,g,h\rangle\\
&e=a^2+(a^2d^2+d^2a^3+a^2d^3)+(c^2d^2+d^2c^3+c^2d^3)\\
&f=b^2+(a^2d^2+d^2a^3+a^2d^3)+(b^2d^2+d^2b^3+b^2d^3)+(c^2d^2+d^2c^3+c^2d^3)\\
&g=c^2+(b^2d^2+d^2b^3+b^2d^3)+(c^2d^2+d^2c^3+c^2d^3) \qquad h=d^2 \qquad (29)\\
\boldsymbol{y}^2 =& f^{3,1}_{\mathcal{Q}_{299}}(\langle a^3,b^3,c^3,d^3\rangle,\langle a^1,b^1,c^1,d^1\rangle)=\langle e,f,g,h\rangle\\
&e=a^3+(a^3d^3+d^3a^1+a^3d^1)+(c^3d^3+d^3c^1+c^3d^1)\\
&f=b^3+(a^3d^3+d^3a^1+a^3d^1)+(b^3d^3+d^3b^1+b^3d^1)+(c^3d^3+d^3c^1+c^3d^1)\\
&g=c^3+(b^3d^3+d^3b^1+b^3d^1)+(c^3d^3+d^3c^1+c^3d^1) \qquad h=d^3 \qquad (30)\\
\boldsymbol{y}^3 =& f^{1,2}_{\mathcal{Q}_{299}}(\langle a^1,b^1,c^1,d^1\rangle,\langle a^2,b^2,c^2,d^2\rangle)=\langle e,f,g,h\rangle\\
&e=a^1+(a^1d^1+d^1a^2+a^1d^2)+(c^1d^1+d^1c^2+c^1d^2)\\
&f=b^1+(a^1d^1+d^1a^2+a^1d^2)+(b^1d^1+d^1b^2+b^1d^2)+(c^1d^1+d^1c^2+c^1d^2)\\
&g=c^1+(b^1d^1+d^1b^2+b^1d^2)+(c^1d^1+d^1c^2+c^1d^2) \qquad h=d^1 \qquad (31)
\end{aligned}$$

B.3 Second-order Component Functions

The first stage $\mathcal{Q}_{294}$

$$\begin{aligned}
\boldsymbol{y}^1 =& f^{2,3}_{\mathcal{Q}_{294}}(\langle a^2,b^2,c^2,d^2\rangle,\langle a^3,b^3,c^3,d^3\rangle) = \langle e,f,g,h\rangle \\
& e = a^2 + b^2d^2 + d^2b^3 + b^2d^3 \qquad g = c^2 \\
& f = b^2 + c^2d^2 + d^2c^3 + c^2d^3 \qquad h = d^2
\end{aligned} \tag{32}$$

$$\begin{aligned}
\boldsymbol{y}^2 =& f^{3,4}_{\mathcal{Q}_{294}}(\langle a^3,b^3,c^3,d^3\rangle,\langle a^4,b^4,c^4,d^4\rangle) = \langle e,f,g,h\rangle \\
& e = a^3 + b^3d^3 + d^3b^4 + b^3d^4 \qquad g = c^3 \\
& f = b^3 + c^3d^3 + d^3c^4 + c^3d^4 \qquad h = d^3
\end{aligned} \tag{33}$$

$$\begin{aligned}
\boldsymbol{y}^3 =& f^{4,5}_{\mathcal{Q}_{294}}(\langle a^4,b^4,c^4,d^4\rangle,\langle a^5,b^5,c^5,d^5\rangle) = \langle e,f,g,h\rangle \\
& e = a^4 + b^4d^4 + d^4b^5 + b^4d^5 \qquad g = c^4 \\
& f = b^4 + c^4d^4 + d^4c^5 + c^4d^5 \qquad h = d^4
\end{aligned} \tag{34}$$

$$\begin{aligned}
\boldsymbol{y}^4 =& f^{5,1}_{\mathcal{Q}_{294}}(\langle a^5,b^5,c^5,d^5\rangle,\langle a^1,b^1,c^1,d^1\rangle) = \langle e,f,g,h\rangle \\
& e = a^5 + b^5d^5 + d^5b^1 + b^5d^1 \qquad g = c^5 \\
& f = b^5 + c^5d^5 + d^5c^1 + c^5d^1 \qquad h = d^5
\end{aligned} \tag{35}$$

$$\begin{aligned}
\boldsymbol{y}^5 =& f^{1,2}_{\mathcal{Q}_{294}}(\langle a^1,b^1,c^1,d^1\rangle,\langle a^2,b^2,c^2,d^2\rangle) = \langle e,f,g,h\rangle \\
& e = a^1 + b^1d^1 + d^1b^2 + b^1d^2 \qquad g = c^1 \\
& f = b^1 + c^1d^1 + d^1c^2 + c^1d^2 \qquad h = d^1
\end{aligned} \tag{36}$$

$$\begin{aligned}
\boldsymbol{y}^6 =& g^{2,4}_{\mathcal{Q}_{294}}(\langle a^2,b^2,c^2,d^2\rangle,\langle a^4,b^4,c^4,d^4\rangle) = \langle e,f,g,h\rangle \\
& e = d^2b^4 + b^2d^4 \qquad f = d^2c^4 + c^2d^4 \qquad g = 0 \qquad h = 0
\end{aligned} \tag{37}$$

$$\begin{aligned}
\boldsymbol{y}^7 =& g^{3,5}_{\mathcal{Q}_{294}}(\langle a^3,b^3,c^3,d^3\rangle,\langle a^5,b^5,c^5,d^5\rangle) = \langle e,f,g,h\rangle \\
& e = d^3b^5 + b^3d^5 \qquad f = d^3c^5 + c^3d^5 \qquad g = 0 \qquad h = 0
\end{aligned} \tag{38}$$

$$\begin{aligned}
\boldsymbol{y}^8 =& g^{1,4}_{\mathcal{Q}_{294}}(\langle a^1,b^1,c^1,d^1\rangle,\langle a^4,b^4,c^4,d^4\rangle) = \langle e,f,g,h\rangle \\
& e = d^1b^4 + b^1d^4 \qquad f = d^1c^4 + c^1d^4 \qquad g = 0 \qquad h = 0
\end{aligned} \tag{39}$$

$$\begin{aligned}
\boldsymbol{y}^9 =& g^{2,5}_{\mathcal{Q}_{294}}(\langle a^2,b^2,c^2,d^2\rangle,\langle a^5,b^5,c^5,d^5\rangle) = \langle e,f,g,h\rangle \\
& e = d^2b^5 + b^2d^5 \qquad f = d^2c^5 + c^2d^5 \qquad g = 0 \qquad h = 0
\end{aligned} \tag{40}$$

$$\begin{aligned}
\boldsymbol{y}^{10} =& g^{1,3}_{\mathcal{Q}_{294}}(\langle a^1,b^1,c^1,d^1\rangle,\langle a^3,b^3,c^3,d^3\rangle) = \langle e,f,g,h\rangle \\
& e = d^1b^3 + b^1d^3 \qquad f = d^1c^3 + c^1d^3 \qquad g = 0 \qquad h = 0
\end{aligned} \tag{41}$$

The second stage $\mathcal{Q}_{299}$

$$\begin{aligned}\boldsymbol{y}^1 =& f^{2,3}_{\mathcal{Q}_{299}}(\langle a^2,b^2,c^2,d^2\rangle,\langle a^3,b^3,c^3,d^3\rangle)=\langle e,f,g,h\rangle\\ &e=a^2+(a^2d^2+d^2a^3+a^2d^3)+(c^2d^2+d^2c^3+c^2d^3)\\ &f=b^2+(a^2d^2+d^2a^3+a^2d^3)+(b^2d^2+d^2b^3+b^2d^3)+(c^2d^2+d^2c^3+c^2d^3)\\ &g=c^2+(b^2d^2+d^2b^3+b^2d^3)+(c^2d^2+d^2c^3+c^2d^3)\qquad h=d^2\end{aligned}\tag{42}$$

$$\begin{aligned}\boldsymbol{y}^2 =& f^{3,4}_{\mathcal{Q}_{299}}(\langle a^3,b^3,c^3,d^3\rangle,\langle a^4,b^4,c^4,d^4\rangle)=\langle e,f,g,h\rangle\\ &e=a^3+(a^3d^3+d^3a^4+a^3d^4)+(c^3d^3+d^3c^4+c^3d^4)\\ &f=b^3+(a^3d^3+d^3a^4+a^3d^4)+(b^3d^3+d^3b^4+b^3d^4)+(c^3d^3+d^3c^4+c^3d^4)\\ &g=c^3+(b^3d^3+d^3b^4+b^3d^4)+(c^3d^3+d^3c^4+c^3d^4)\qquad h=d^3\end{aligned}\tag{43}$$

$$\begin{aligned}\boldsymbol{y}^3 =& f^{4,5}_{\mathcal{Q}_{299}}(\langle a^4,b^4,c^4,d^4\rangle,\langle a^5,b^5,c^5,d^5\rangle)=\langle e,f,g,h\rangle\\ &e=a^4+(a^4d^4+d^4a^5+a^4d^5)+(c^4d^4+d^4c^5+c^4d^5)\\ &f=b^4+(a^4d^4+d^4a^5+a^4d^5)+(b^4d^4+d^4b^5+b^4d^5)+(c^4d^4+d^4c^5+c^4d^5)\\ &g=c^4+(b^4d^4+d^4b^5+b^4d^5)+(c^4d^4+d^4c^5+c^4d^5)\qquad h=d^4\end{aligned}\tag{44}$$

$$\begin{aligned}\boldsymbol{y}^4 =& f^{5,1}_{\mathcal{Q}_{299}}(\langle a^5,b^5,c^5,d^5\rangle,\langle a^1,b^1,c^1,d^1\rangle)=\langle e,f,g,h\rangle\\ &e=a^5+(a^5d^5+d^5a^1+a^5d^1)+(c^5d^5+d^5c^1+c^5d^1)\\ &f=b^5+(a^5d^5+d^5a^1+a^5d^1)+(b^5d^5+d^5b^1+b^5d^1)+(c^5d^5+d^5c^1+c^5d^1)\\ &g=c^5+(b^5d^5+d^5b^1+b^5d^1)+(c^5d^5+d^5c^1+c^5d^1)\qquad h=d^5\end{aligned}\tag{45}$$

$$\begin{aligned}\boldsymbol{y}^5 =& f^{1,2}_{\mathcal{Q}_{299}}(\langle a^1,b^1,c^1,d^1\rangle,\langle a^2,b^2,c^2,d^2\rangle)=\langle e,f,g,h\rangle\\ &e=a^1+(a^1d^1+d^1a^2+a^1d^2)+(c^1d^1+d^1c^2+c^1d^2)\\ &f=b^1+(a^1d^1+d^1a^2+a^1d^2)+(b^1d^1+d^1b^2+b^1d^2)+(c^1d^1+d^1c^2+c^1d^2)\\ &g=c^1+(b^1d^1+d^1b^2+b^1d^2)+(c^1d^1+d^1c^2+c^1d^2)\qquad h=d^1\end{aligned}\tag{46}$$

$$\begin{aligned}\boldsymbol{y}^6 =& g^{2,4}_{\mathcal{Q}_{299}}(\langle a^2,b^2,c^2,d^2\rangle,\langle a^4,b^4,c^4,d^4\rangle)=\langle e,f,g,h\rangle\\ &e=d^2a^4+d^2c^4+a^2d^4+c^2d^4\\ &f=d^2a^4+d^2b^4+d^2c^4+a^2d^4+b^2d^4+c^2d^4\\ &g=d^2b^4+d^2c^4+b^2d^4+c^2d^4\qquad h=0\end{aligned}\tag{47}$$

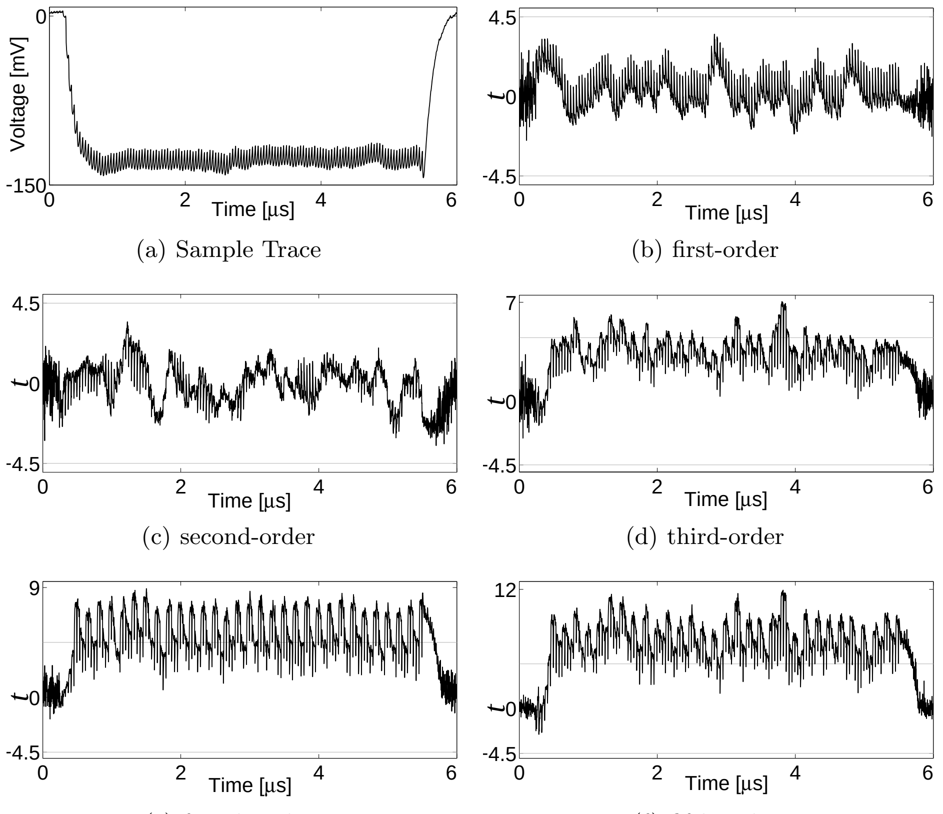
$$\begin{aligned}\boldsymbol{y}^7 =& g^{3,5}_{\mathcal{Q}_{299}}(\langle a^3,b^3,c^3,d^3\rangle,\langle a^5,b^5,c^5,d^5\rangle)=\langle e,f,g,h\rangle\\ &e=d^3a^5+d^3c^5+a^3d^5+c^3d^5\\ &f=d^3a^5+d^3b^5+d^3c^5+a^3d^5+b^3d^5+c^3d^5\\ &g=d^3b^5+d^3c^5+b^3d^5+c^3d^5\qquad h=0\end{aligned}\tag{48}$$

$$\begin{aligned}\boldsymbol{y}^8 =& g^{1,4}_{\mathcal{Q}_{299}}(\langle a^1,b^1,c^1,d^1\rangle,\langle a^4,b^4,c^4,d^4\rangle)=\langle e,f,g,h\rangle\\ &e=d^1a^4+d^1c^4+a^1d^4+c^1d^4\\ &f=d^1a^4+d^1b^4+d^1c^4+a^1d^4+b^1d^4+c^1d^4\\ &g=d^1b^4+d^1c^4+b^1d^4+c^1d^4\qquad h=0\end{aligned}\tag{49}$$

$$\begin{aligned}
\boldsymbol{y}^9 =& g^{2,5}_{\mathcal{Q}_{299}}(\langle a^2, b^2, c^2, d^2\rangle, \langle a^5, b^5, c^5, d^5\rangle) = \langle e, f, g, h\rangle \\
& e = d^2a^5 + d^2c^5 + a^2d^5 + c^2d^5 \\
& f = d^2a^5 + d^2b^5 + d^2c^5 + a^2d^5 + b^2d^5 + c^2d^5 \\
& g = d^2b^5 + d^2c^5 + b^2d^5 + c^2d^5 \qquad h = 0
\end{aligned} \tag{50}$$

$$\begin{aligned}
\boldsymbol{y}^{10} =& g^{1,3}_{\mathcal{Q}_{299}}(\langle a^1, b^1, c^1, d^1\rangle, \langle a^3, b^3, c^3, d^3\rangle) = \langle e, f, g, h\rangle \\
& e = d^1a^3 + d^1c^3 + a^1d^3 + c^1d^3 \\
& f = d^1a^3 + d^1b^3 + d^1c^3 + a^1d^3 + b^1d^3 + c^1d^3 \\
& g = d^1b^3 + d^1c^3 + b^1d^3 + c^1d^3 \qquad h = 0
\end{aligned} \tag{51}$$

C Evaluation results of PRESENT-2nd by 1 billion traces



(a) Sample Trace

(b) first-order

(c) second-order

(d) third-order

(e) fourth-order

(f) fifth-order

Fig. 11. PRESENT-2nd profile, sample trace and non-specific t-test results using 1,000,000,000 traces