Higher-Order Threshold Implementations

Begül Bilgin[1,2], Benedikt Gierlichs[1], Svetla Nikova[1],
Ventzislav Nikov[3], and Vincent Rijmen[1]

[1] KU Leuven, ESAT-COSIC and iMinds, Belgium
{name.surname}@esat.kuleuven.be
[2] University of Twente, EEMCS-SCS, The Netherlands
[3] NXP Semiconductors, Belgium
venci.nikov@gmail.com

Abstract. Higher-order differential power analysis attacks are a serious threat for cryptographic hardware implementations. In particular, glitches in the circuit make it hard to protect the implementation with masking. The existing higher-order masking countermeasures that guarantee security in the presence of glitches use multi-party computation techniques and require a lot of resources in terms of circuit area and randomness. The Threshold Implementation method is also based on multi-party computation but it is more area and randomness efficient. Moreover, it typically requires less clock-cycles since all parties can operate simultaneously. However, so far it is only provable secure against 1st-order DPA. We address this gap and extend the Threshold Implementation technique to higher orders. We define generic constructions and prove their security. To illustrate the approach, we provide 1st, 2nd and 3rd-order DPA-resistant implementations of the block cipher KATAN-32. Our analysis of 300 million power traces measured from an FPGA implementation supports the security proofs.

1 Introduction

Differential power analysis (DPA) attacks as introduced by Kocher et al. [19] exploit unintentional information leakage of a device's internal processing through its power consumption. Over the years, many types of countermeasures against DPA attacks have been proposed. One family of countermeasures is called masking and consists in computing the algorithm on a randomized representation of the data. For this purpose, the data is split in several shares that are processed sequentially or in parallel. A DPA attack that exploits the information leakage of several shares jointly, be it by combining the leakage from several points in time or by analyzing higher-order statistical moments of the leakage at one point in time, is a higher-order DPA (HO-DPA) attack [6,24].

It is preferable to protect the implementation of a cryptographic algorithm with a higher-order masking countermeasure, where $d > 1$ random masks are used to generate $d + 1$ shares of a variable, since 2nd-order DPA attacks can be relatively inexpensive to mount. It is well known that the number of measurements required for a HO-DPA attack to succeed scales exponentially in the noise standard deviation, the exponent being $d + 1$ [6,33].

P. Sarkar and T. Iwata (Eds.): ASIACRYPT 2014, PART II, LNCS 8874, pp. 326–343, 2014.

In a secure masking, all $d+1$ shares are necessary to re-construct the variable. Such a secure masking is called d^{th}-order masking and leads to a d^{th}-order secure implementation in software. An implementation of the same secure masking in CMOS-like glitchy hardware, on the other hand, will typically be insecure [22,23,27].

Related Work. Several masking schemes that are secure against HO-DPA have been proposed so far, e.g. [8,9,14,17,18,34,35]. However the scheme in [17] is shown to be insecure in [10] and [9] discovers and proposes a fix to a leak in [35]. Only one scheme claims to be secure against HO-DPA even in the presence of glitches [34], based on separation of the operations in the time domain. Nevertheless, implementing this scheme within the defined models is a challenging task. Moradi and Mischke [26] provided practical evidence that a simple separation of the operations in the time domain alone is not sufficient when the shares of a sensitive variable are processed in consecutive clock cycles.

Threshold Implementation (TI) is a masking scheme based on secret sharing and multi-party computation [29,30,31]. It provides provable security against 1^{st}-order DPA even in the presence of glitches. The only requirement is that the shares leak independently, but this requirement holds for all masking schemes.

So far, several S-boxes and symmetric-key algorithms have been implemented with this method and the security claim has been confirmed in practical experiments [2,28,32]. However, it has also been confirmed that a TI is vulnerable to HO-DPA [2,25].

Contribution. So far, the theory of TI and its practical security is limited to counteract 1^{st}-order DPA. In this work, we define *Higher-Order Threshold Implementation (HO-TI)* to thwart HO-DPA. We define generic constructions and use results from CHES 2010, EUROCRYPT 2010 and 2014 to prove their security. We provide a relation between 1^{st}-order DPA secure implementations of 4 × 4 S-boxes in the alternating group with 5-shares provided in [4] and 2^{nd}-order DPA secure implementations of these S-boxes. To illustrate the HO-TI approach in a comprehensible example, we provide 1^{st}, 2^{nd} and 3^{rd}-order DPA-resistant implementations of the block cipher KATAN-32. Our analysis of 300 million power traces measured from an FPGA implementation supports the security proofs.

2 Theory of HO-TI

We use lower case characters to refer to elements of a finite field and upper case characters to describe vectors and vector functions. Stochastic variables are described by the superscript $^{\$}$. The probability that $x^{\$}$ takes the value x is $Pr(x^{\$}=x)$. In order to implement a function $f(x)=y$ from $\mathcal{F}^n$ to $\mathcal{F}^m$ with TI, we first split each variable x into s shares x_i where $i \in \{1,2,\ldots,s\}$ by means of Boolean masking, such that the XOR sum of these shares is equal to the variable itself ($x = \sum_i x_i$). For all values x with $Pr(x^\$ = x) > 0$, let $Sh(x)$ denote the set of valid share vectors X for x:

$$Sh(x) = \{X \in \mathcal{F}^{ns} \mid x_1 \oplus x_2 \oplus \cdots \oplus x_s = x\}.$$

We use the terms sharing or masking interchangeably for a valid share vector X and use the term s-sharing of x to emphasize the number of shares. $Pr(X^\$ = X|x^\$ = x)$ denotes the probability that $X^\$ = X$ when the unmasked value equals x, taken over all auxiliary inputs of the masking.

In a TI, f is implemented as a vector of functions F that takes X as input. Each function in this vector is called a component function and represented by f_i where $i \in \{1, \ldots, s\}$. From now on, we use the term sharing of the function to describe F and s-sharing of f to emphasize the number of component functions of F. F must satisfy the following property for a correct implementation.

Property 1 (Correctness). $\forall y \in \mathcal{F}^m$, $\forall X \in Sh(x)$ and $\forall Y \in Sh(y)$; $F(X) = Y \iff f(x) = y$.

We call each share of X an input share and each share of Y an output share.

2.1 HO-TI of an Arbitrary Function

Like for other masking schemes, the masking of the input of a shared function F must be uniform. We call a masking X of a variable x uniform if and only if $Pr(X^\$ = X|x^\$ = x)$ is equal to the same constant p for each X and $\sum_X Pr(X^\$ = X|x^\$ = x) = Pr(x^\$ = x)$. If the input is a uniform masking, then the shared function F must satisfy the following property in order to achieve security against d^{th}-order DPA.

Property 2 (d^{th}-order non-completeness). Any combination of up to d component functions f_i of F must be independent of at least one input share.

One can see that this d^{th}-order non-completeness property is equivalent to the non-completeness property defined in [31] for 1^{st}-order DPA resistance when $d = 1$. We define a TI that satisfies Property 1 and Property 2 as d^{th}-order TI.

In 2010, two different works at Eurocrypt and Ches [13,35] show a correspondence between the HO-DPA attack model and the so-called "probing model" where the d probing model considers an adversary that is allowed to observe the value of up to d intermediate wires of the circuit during the computation. Moreover, at Eurocrypt 2014, this probing model is used by [12] to prove security against HO-DPA, and a relation between the probing model and the noisy leakage model is provided in [8]. We make use of the following results.

Lemma 1. *The attack order in a higher-order DPA corresponds to the number of wires that are probed in the circuit (per unmasked bit).*

This lemma implies that if a circuit is perfectly secure against d probes, then combining d power consumption points as in a d^{th}-order DPA will reveal no information. Since TI operates on the component functions in parallel and does not separate these operations in the time domain, this is equivalent to security against DPA exploiting the d^{th}-order statistical moment. However, it should be noted that the models considered in the mentioned papers do not take glitches into account. Thanks to the TI separation of the component functions we are able to use their models and results, and prove stronger security in the presence of glitches.

Theorem 1. *If the input masking X of the shared function F is a uniform masking and F is a d^{th}-order TI then the d^{th} statistical moment of the power consumption of a circuit implementing F is independent of the unmasked input value x even if the inputs are delayed or glitches occur in the circuit.*

Proof. By Lemma 1, it is sufficient to prove that an adversary who can probe d wires does not get any information about x. By construction, if F satisfies Property 1 and Property 2, an adversary who probes d or less wires will get information from all but at least one input share, which is independent of the input. □

2.2 On the Number of Shares

The storage of the state of a symmetric key algorithm and hence the storage of the sharing of the state is typically the most expensive part in terms of area in a hardware implementation. In all masking schemes, the number of shares required increases with the order of DPA to protect against. Considering that DPA is a powerful attack especially against constrained devices, defining a higher-order masking that has a small area footprint, therefore with the minimum number of shares, becomes important.

An affine function $f(x) = y$ can be implemented with $s \geq d + 1$ component functions to thwart d^{th}-order DPA. One possible way to generate F is to define the first component function to be $f_1(x_1) = y_1 = f(x_1)$ and the rest of the component functions to be $f_i(x_i) = y_i$ where f_i is equal to f without constant terms and $2 \leq i \leq s$. To give an example $f(x) = 1 + x$ can be implemented with the following component functions:

$$f_1(x_1) = 1 + x_1 \text{ and } f_i(x_i) = x_i, \text{ where } i \in \{2, \ldots, s\}.$$

However, the minimum number of shares required increases together with the nonlinearity. When the whole cryptographic algorithm is considered, one way to construct a TI is to adapt the number of shares to be minimum for each component of the algorithm, and to decrease or increase the number of shares as required. This approach is partially applied in [2]. Even though this method may lead to a relatively small circuit, it raises the problem to generate fresh randomness to be able to increase the number of shares. Another method is to keep the number of shares constant as much as possible as in [29,32] to avoid using fresh randomness. We adopt the second idea and try to keep the number of input shares that are used by a sharing of a nonlinear operation as small as possible. It is also possible to have different numbers of input and output shares to a nonlinear operation. This idea was already mentioned for 1st-order TI in [3]. Unlike that particular case where the number of input shares s_{in} is greater than the number of output shares s_{out}, we require $s_{out} \geq s_{in}$ to avoid using fresh randomness for increasing the number of shares back to s_{in}. A way to decrease the number of shares without using extra randomness will be discussed in the following subsection.

Theorem 2. *There always exist a d^{th}-order TI of a function of degree t that requires $s_{in} \geq t \times d + 1$ input and $s_{out} \geq \binom{s_{in}}{t}$ output shares.*

Proof. Consider, without loss of generality, the product $x^1 x^2 x^3 \dots x^t$ of first t variables where $\mathcal{F}^n \ni x = (x^1, x^2, \dots x^n)$ and $x^j \in \mathcal{F}$. We represent the sharing of each variable x^j as x^j_i where $i \in \{1, \dots, s_{in}\}$. Then,

$$\begin{aligned} x^1 x^2 x^3 \dots x^t &= (x^1_1 + x^1_2 + \dots + x^1_{s_{in}}) \dots (x^t_1 + x^t_2 + \dots + x^t_{s_{in}}) \\ &= (x^1_1 x^2_1 \dots x^t_1) + (x^1_1 x^2_1 \dots x^t_2) + \dots + (x^1_{s_{in}} x^2_{s_{in}} \dots x^t_{s_{in}}). \end{aligned}$$

To satisfy the correctness each term in the above sum should exist in (or belong to) at least one component function. This can be done in the following way. Let each component function use only t different shares such that any t combination of s_{in} shares is used by only one component function. Hence any combination of up to d component functions carries information from at most $t \times d$ shares. To achieve the non-completeness property, $s_{in} > t \times d$ which implies the equation $s_{in} \geq t \times d + 1$ for the number of input shares. With the given sharing, there exist $\binom{s_{in}}{t}$ different ways of choosing t combinations of s_{in} shares and placing them in component functions. Hence, this sharing needs $s_{out} \geq \binom{s_{in}}{t}$ component functions. The proof can be extended to all degree t terms. □

Theorem 2 shows that the number of input shares of a function depends linearly on the order of security for a TI. Moreover, the required number of input and output shares given in Theorem 2 corresponds to the number of shares for $d = 1$ in [31].

We point out that a TI using the number of shares defined in the previous theorem is not the only possible construction. Moreover, the theorem does not imply that the number of output shares or the total number of input and output shares $(s_{in} + s_{out})$ are minimized. As an example, consider $y = f(a, b, b) = 1 + a + bc$ where $y, a, b, c \in \mathcal{F}$. For a 2nd-order TI of f, by Theorem 2, one requires $s_{in} = 5$ input shares which implies $s_{out} = 10$ output shares. One of the many alternatives for constructing the component functions for that scenario is

$$\begin{aligned} y_1 &= 1 + a_2 + b_2 c_2 + b_1 c_2 + b_2 c_1 & y_2 &= a_3 + b_3 c_3 + b_1 c_3 + b_3 c_1 \\ y_3 &= a_4 + b_4 c_4 + b_1 c_4 + b_4 c_1 & y_4 &= a_1 + b_1 c_1 + b_1 c_5 + b_5 c_1 \\ y_5 &= b_2 c_3 + b_3 c_2 & y_6 &= b_2 c_4 + b_4 c_2 \\ y_7 &= a_5 + b_5 c_5 + b_2 c_5 + b_5 c_2 & y_8 &= b_3 c_4 + b_4 c_3 \\ y_9 &= b_3 c_5 + b_5 c_3 & y_{10} &= b_4 c_5 + b_5 c_4 \,. \end{aligned} \qquad (1)$$

If we do not fix $s_{in} = 5$, we can also construct a 2^{nd}-order TI with $s_{in} = 6$ input and $s_{out} = 7$ output shares as described in Appendix A.2. It is still an open question to find a lower bound for $s_{in} + s_{out}$.

The component functions provided in Equation (1) for a 2^{nd}-order TI of a degree two function are constructed in a systematic way following the proof of Theorem 2. Namely, they are constructed with $s_{in} = 2 \times 2 + 1 = 5$ input shares and each component function uses one of the $\binom{5}{2} = 10$ possible combinations of $t = 2$ shares exactly. When this construction is reduced to achieve 1^{st}-order DPA security, one gets the equation given in [29] which is repeated in Appendix A.1 for completeness.

Component functions for functions of higher degrees and/or other security levels can be derived with the same construction. We provide an example of a 3^{rd}-order TI of f in Appendix A.3.

2.3 Decreasing the Number of Shares

With the construction described in the previous section, we see that the number of output shares becomes greater than the number of input shares when $d > 1$. To avoid further increase in shares and hence in area, we need to decrease the number of shares. This decrease can be done by combining different shares with an affine function as described in the following theorem.

Theorem 3. *Given $s_{in} \geq d+r$ input shares where $r \geq 1$ that are not necessarily uniform masking but secure against $(d+r-1)^{st}$-order DPA, any sharing G that combines any r of the input shares linearly in one component function and keeps the rest of the input shares unchanged, is secure against d^{th}-order DPA.*

Proof. We represent the variable a with $s_{in} \geq d+r$ shares for a given d, that are not necessarily a uniform masking. Assume that this initial masking of a is secure against $(d+r-1)^{\text{st}}$-order DPA. That implies that combining any $d+r-1$ shares does not reveal the unmasked value a. Consider $s_{in}-1$ component functions: the first component function combines the first two input shares linearly, without loss of generality, the other component functions each take one share as input and output it unchanged, i.e. $g_1 = a_1 + a_2$ and $g_{i-1} = a_i$ for $3 \leq i \leq s_{in}$. This construction satisfies both Property 1 and Property 2 for $(d + r - 2)^{\text{nd}}$-order security and one needs $s_{in}-1 \geq d+r-1$ shares to reveal the unmasked variable. Moreover, the component function g_1 only uses a balanced gate. Namely, a 2×1 XOR gate whose output changes with probability 1 for any input bit change, independent of the input value. Hence, even though the input is not uniform, this sharing of g will not leak information. A mere $r - 1$ repetition of this procedure gives a sharing with $d + 1$ shares that satisfies Property 1 and Property 2 and that is hence d^{th}-order DPA secure. Moreover, since there are only balanced gates involved, one can combine this repetitive construction in one step. □

Remark 1. To satisfy Property 2, the nonlinear operation generating the sharing for a mentioned in the proof of Theorem 3 and the operation to decrease the number of shares should be separated by registers.

Given Theorem 3, one can decrease the number of shares from s_{out} to s_{in} as follows. Let the nonlinear function we want to share be $f(x) = y$ with d^{th}-order TI sharing $F(X) = Y$ such that X is an s_{in}-sharing and Y is an s_{out}-sharing. Consider another sharing $G(Y) = Z$ of a function g as defined in Theorem 3 where Z is again an s_{in}-sharing, and G is a d^{th}-order TI. It is not necessarily required that the input sharing of G is uniform. As an example, for Equation (1) which represents a 2^{nd}-order TI of a quadratic function, one possible way to decrease the shares such that X and Z are represented with the same number of shares is given below.

$$z_i = y_i, \text{ where } i < 5 \text{ and } z_5 = y_5 + y_6 + y_7 + y_8 + y_9 + y_{10}. \tag{2}$$

With this TI, it is important to make sure that Remark 1 is applied by using registers after the nonlinear operation F.

2.4 On Uniformity

We have proved that a function f can be implemented in a way that is secure against d^{th}-order DPA if Property 1 and Property 2 are satisfied and the masking of the input is uniform (Theorem 1). Hence, we need to make sure that the input to a shared function K of a nonlinear function k which follows $H = G \circ F$ is also a uniform masking unless it is equal to the exceptional linear case defined in Theorem 3. This is equivalent to saying that H should be a uniform sharing of the function h as defined by the following property.

Property 3 (Uniform sharing of functions). The sharing H of h is uniform if and only if $\forall z \in \mathcal{F}^m, \forall (z_1, z_2, \ldots, z_{s_{out_z}}) \in Sh(z), \forall x \in \mathcal{F}^n$ with $h(x) = z$ and $s_{out_z} \geq d + 1$:

$$|\{(x_1, x_2, \ldots, x_{s_{in}}) \in Sh(x) | H(x_1, x_2, \ldots, x_{s_{in}}) = (z_1, z_2, \ldots, z_{s_{out_z}})\}| = \frac{\mathcal{F}^{n(s_{in}-1)}}{\mathcal{F}^{m(s_{out_z}-1)}}.$$

We call a d^{th}-order TI that is a uniform sharing, a uniform d^{th}-order TI. Unfortunately, we do not know a straight forward way to generate the component functions with s_{min} input and s_{out} output shares provided in Theorem 2 so that this property holds (unlike the other two properties) for any Boolean function. Hence, a sharing should be explicitly checked to satisfy Property 3. In this paper, we recall a uniform sharing of an AND and an XOR gate that is secure against 1^{st}-order DPA in Equation (6) which is equal to the formula derived in [31]. Moreover, we provide uniform sharings that are secure against 2^{nd} and 3^{rd}-order DPA by the sharings of $H = G \circ F$ generated from Equation (1) and Equation (8) together with Equation (2). Note that Equations (1) and (8) alone are not uniform. We found these sharings with a guided computer search. In the following section, we will also provide a way to construct uniform 2^{nd}-order TI of 4×4 S-boxes in the alternating group with 5 input and 10 output shares.

2.5 Constructing 2^{nd}-Order TI of Some 4 × 4 S-Boxes

The majority of the S-boxes used in lightweight implementations are 4 × 4 S-boxes, therefore it is important to provide hardware implementations of these S-boxes secure against HO-DPA.

In [4], it is shown that all 4 × 4 S-boxes that are in the alternating group (S-boxes that can be represented as an even number of transpositions, e.g. PRESENT [5], KLEIN [15] and NOEKEON S-boxes and half of the Optimal S-boxes [21]) can be decomposed into quadratic S-boxes. Moreover, these S-boxes (represented as $s(x)$ or one of their affine equivalents $s'(x) = a(s(b(x)))$ s.t. a and b are affine permutations) have a uniform 1^{st}-order TI with 5 input and output shares with *direct sharing*. To be more precise, if the sharing in Equation (3) (given for $f(a, b, c) = 1 + a + bc$) is applied to each term of the vectorial Boolean functions, the resulting TI is 1^{st}-order DPA-resistant and uniform.

$$\begin{aligned}
y_1 &= 1 + a_2 + b_2c_2 + b_2c_3 + b_3c_2 + b_2c_4 + b_4c_2\\
y_2 &= a_3 + b_3c_3 + b_3c_4 + b_4c_3 + b_3c_5 + b_5c_3\\
y_3 &= a_4 + b_4c_4 + b_4c_5 + b_5c_4 + b_4c_1 + b_1c_4\\
y_4 &= a_5 + b_5c_5 + b_5c_1 + b_1c_5 + b_5c_2 + b_2c_5\\
y_5 &= a_1 + b_1c_1 + b_1c_2 + b_2c_1 + b_1c_3 + b_3c_1\,.
\end{aligned} \tag{3}$$

Generating the component functions as in Equation (4) for any of the mentioned S-boxes would lead to 2^{nd}-order TI with $s_{in} = 5$ and $s_{out} = 10$.

$$\begin{aligned}
y_1 &= 1 + a_2 + b_2c_2 + b_2c_3 + b_3c_2 & y_2 &= b_2c_4 + b_4c_2\\
y_3 &= a_3 + b_3c_3 + b_3c_4 + b_4c_3 & y_4 &= b_3c_5 + b_5c_3\\
y_5 &= a_4 + b_4c_4 + b_4c_5 + b_5c_4 & y_6 &= b_4c_1 + b_1c_4\\
y_7 &= a_5 + b_5c_5 + b_5c_1 + b_1c_5 & y_8 &= b_5c_2 + b_2c_5\\
y_9 &= a_1 + b_1c_1 + b_1c_2 + b_2c_1 & y_{10} &= b_1c_3 + b_3c_1\,.
\end{aligned} \tag{4}$$

If the sharing G of $g(y) = z$ described in Section 2.3 is generated as $g_i = y_{2i-1} + y_{2i}$ for $i \leq 5$, the overall sharing $H(X) = G(F(X))$ of the S-box (or one of its affine equivalent) is uniform since the sharing H is equivalent to the sharing given in Equation (3). Hence, we can construct uniform 2^{nd}-order TI of all 4 × 4 S-boxes in the alternating group.

3 Implementation

We recall the block cipher KATAN and propose HO-TIs of it. We provide the area requirements of these implementations in the Faraday Standard Cell Library FSA0A_C_Generic_Core which is based on UMC 0.18μm GenericII Logic Process with 1.8V voltage. We verify the functionality of the implementations with ModelSim and synthesize using Synopsys Design Vision D-201-.03-SP4 without any optimization.

3.1 KATAN

KATAN [11] is a family of block ciphers that is designed to be efficient in hardware. The family has three variants with 32, 48 or 64-bit state size. All these variants use an 80-bit key, hence have the same security level. A plaintext block, of the same size as the state, is loaded into the state to start an encryption. After 254 rounds, the content of the state is taken as the ciphertext. The round operation is very similar for all variants and has only a few AND and XOR gates.

Our main consideration is to show how to instantiate a higher-order TI of a simple algorithm and to analyze its side channel leakage. For this reason, we implement the smallest variant of KATAN with 32-bit state size and focus on encryption. A description of one round of KATAN-32 is provided in Appendix B.

3.2 TI of KATAN

We describe a general TI that has s_{in} input shares, the same number of shares in the state and s_{out} output shares for nonlinear operations. An example of a 2^{nd}-order TI of one round KATAN-32 where $s_{in} = 5$ and $s_{out} = 10$ is depicted in Figure 1 (z coordinate refers to s_{in} different shares of the state). In all these versions, we use the same unshared key schedule for simplicity.

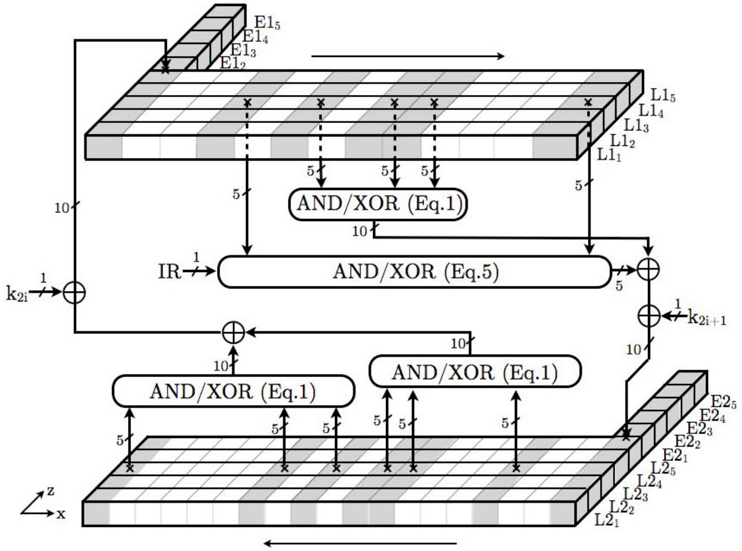

Fig. 1. Description of 2^{nd}-order TI of one round of KATAN-32

We assume that the plaintext has a uniform masking with s_{in} shares that is provided as input. Each share will be split into two chunks of 13 and 19 bits that will be written to the registers $L1_j$ and $L2_j$ respectively where $j \leq s_{in}$. Since we already know how to implement an AND and an XOR gate with a uniform TI, we split the operations of the round update accordingly. For all the AND/XOR blocks except the one that receives IR, we use the TI in Equation (1) (resp. Eqn. (6) and Eqn.(8) for 1st and 3rd-order TI) which takes s_{in} input shares. For the AND/XOR block that receives IR we use the sharing

$$y_i = a_i + IR \times b_i \text{ where } i \leq s_{in} \tag{5}$$

because we do not share the round counter (and hence IR).

The XOR of two AND/XOR blocks is applied over s_{out} shares or over the first s_{in} shares if the output of the AND/XOR block that receives IR is involved. Similarly, the key is introduced only in the first of the s_{out} shares. This s_{out}-sharing is written to the first bit of the $L1$ and $L2$ registers respectively which have s_{out} shares only for the first bits. One can think of it as having an extension of $s_{out} - s_{in}$ shares for those bits in addition to the s_{in} shares of the state. In the next clock cycle, the s_{out} shares in the first bits of the $L1$ and $L2$ registers are reduced to s_{in} shares as described in Section 2.3 and written as the second bits. This implementation does not increase the number of clock cycles compared to the unprotected KATAN-32 implementation. In Table 1, we show the area requirements of these implementations in NAND gate equivalents. The gate counts for the round function include the decrease of the number of shares by means of Equation (2). The key register is included in the gate count of the key schedule together with the LFSR update.

Table 1. Synthesis results for plain and TI of KATAN-32

	State Array	Round Function	Key Schedule	Control	Other	Total
Plain	170	54	444	64	270	1002
1st-order TI	510	135	444	64	567	1720
2nd-order TI	900	341	444	64	807	2556
3rd-order TI	1330	760	444	64	941	3539

4 Analysis

We implement our 2nd-order TI of Katan-32 on a SASEBO-G board [1] using Xilinx ISE version 10.1 to evaluate its leakage characteristics in practice. The board features two Xilinx Virtex-II Pro FPGA devices: we implement the 2nd-order TI of Katan-32 in the crypto FPGA (xc2vp7) while the control FPGA (xc2vp30) handles I/O with the measurement PC and other equipment includingn the random number generation. We use the "keep hierarchy" constraint when we generate the bitstream for the crypto FPGA to prevent the tools from optimizing over module boundaries. This is to prevent the tools from merging component functions and to reduce the chance for crosstalk. The key is hardcoded in the Katan-32 implementation. The PRNG on the control FPGA is implemented as AES-128 in CTR mode. To start an encryption, we share the plaintext in 5 shares using random numbers from the PRNG and send the shares to the Katan-32 implementation. When the PRNG is turned off, it outputs zeros.

We measure the power consumption of the crypto FPGA during the first 12 rounds of Katan-32 encryption as the voltage drop over a 1Ω resistor in the FPGA core GND line. The output of the passive probe is sampled with a Tektronix DPO 7254C digital oscilloscope at 1GS/s sampling rate and 1mV/div amplitude resolution. We provide the FPGA with a stable 3 MHz clock signal and use synchronized clocks to obtain high-quality measurements.

The main goal of our evaluation is not to demonstrate that the implementation resists state-of-the-art attacks that exploit the 1^{st} or 2^{nd} statistical moment of the leakage distributions, but beyond that to demonstrate that there is no evidence of leakage in these moments of the leakage distributions, exploitable by state-of-the-art attacks or not. Obviously achieving this goal is much more demanding than resistance to known attacks, but it directly corresponds to our claims regarding provable security. We narrow the evaluation to univariate attacks because our implementation processes all component functions in parallel.

We use *leakage detection* to evaluate our implementation. Contrary to the classical approach of testing whether a given attack is successful, this approach decouples the detection of leakage from its exploitation. For our purpose we use the *non-specific* t-test based fixed versus random leakage detection methodology of [7,16], see Appendix C for a brief introduction.

For all tests we obtain two sets of measurements. For the first set, we fix the plaintext to some chosen value. We denote this set $\mathcal{S}_0$. For the second set, the plaintexts are uniformly distributed and random. We denote this set $\mathcal{S}_{random}$. We obtain the measurements for both sets interleaved and in a random order, i.e. before each measurement we flip a coin, to avoid any deterministic or time-dependent external and internal influences on the test result.

We compute Welch's (two-tailed) t-test

$$t = \frac{\mu(\mathcal{S}_0) - \mu(\mathcal{S}_1)}{\sqrt{\frac{\sigma^2(\mathcal{S}_0)}{|\mathcal{S}_0|} + \frac{\sigma^2(\mathcal{S}_1)}{|\mathcal{S}_1|}}}$$

(where $\mu()$ is the sample mean, $\sigma^2()$ is the sample variance and $|\cdot|$ denotes the sample size) to determine if the samples in both sets were drawn from populations with the same mean (or from the same population). The null hypothesis is that the samples in both sets were drawn from populations with the same mean. In our context, this means that the TI is effective. The alternative hypothesis is that the samples in both sets were drawn from populations with different means. In our context, this means that the TI is not effective.

At each point in time, the test statistic t together with the degrees of freedom ν, computed with the Welch-Satterthwaite equation

$$\nu = \frac{(\sigma^2(\mathcal{S}_0)/|\mathcal{S}_0| + \sigma^2(\mathcal{S}_1)/|\mathcal{S}_1|)^2}{(\sigma^2(\mathcal{S}_0)/|\mathcal{S}_0|)^2/(|\mathcal{S}_0|-1) + (\sigma^2(\mathcal{S}_1)/|\mathcal{S}_1|)^2/(|\mathcal{S}_1|-1)},$$

allow to compute a p value to determine if there is sufficient evidence to reject the null hypothesis at a particular significance level $(1-\alpha)$. The p value expresses the probability of observing the measured (or a greater) difference by chance if the null hypothesis was true. In other words, small p values give evidence to reject the null hypothesis.

While this evaluation methodology relieves us from choosing certain parameters such as targeted intermediate value, power model and distinguisher, it does not resolve all such issues. As in any evaluation, the tests are limited to the number of measurements at hand and one has to choose a threshold to decide if an observed difference is statistically significant or not. Nevertheless, as we demonstrate below this type of evaluation is very data-efficient, i.e. a small number of measurements is required to provide evidence of leakage, and a decision threshold can be motivated with some basic experiments.

To calibrate our threshold value we apply the test methodology to two groups of 10 000 measurements each for which we know that the null hypothesis is true. For the first group of measurements we switch off the PRNG and use the same fixed plaintext for both sets, i.e. all measurements in both sets are samples from the same population and the only cause of variance is noise. We compute the t statistic, record its greatest absolute value and repeat the experiment 100 times on a random split of the measurements in this group. The highest absolute t value we observed was 4.7944. For the second group we switch on the PRNG and use random plaintexts for both sets, i.e. the measurements in both sets are samples from distributions with the same mean and high variance. We repeat the analysis and the highest absolute t value we observed was 4.8608. Based on these results and the recommendation in [7] we select the significance threshold ±4.5. For large sample sizes, observing a single t value greater/smaller than ±4.5 roughly corresponds to a 99.999% probability of the null hypothesis being false.

To confirm that our setup works correctly and to get some reference values we first evaluate the implementation with the PRNG switched off. Figure 2 shows the t values of fixed versus random tests with two different fixed plaintexts (left and right) and for the $1^{\text{st}}, 2^{\text{nd}}$ and 3^{rd} statistical moment of the distributions (for the higher-order moments we pre-process the traces to expose the desired standardized moment before we apply the t-test, e.g. for the 2^{nd} moment we center and then square the traces). Horizontal lines mark the ±4.5 thresholds.

The plots clearly show that there is sufficient evidence of leakage in all cases, as there are multiple and systematic crossings of the thresholds. Comparing the plots on the left hand side with the plots on the right hand side, we see that the "shape" of the t curve depends on the fixed plaintext value. This is no longer true when we switch on the PRNG, because all shares of the input are random. We used 1 000 measurements (500 for fixed and 500 for random plaintext) to generate these plots, but less than 100 measurements are required to see evidence of leakage in the 1^{st} statistical moment.

Now we switch on the PRNG and repeat the evaluation with a randomly chosen fixed plaintext using 300 million measurements (150M for fixed, 150M for random, done in a temperature controlled environment). Figure 3 (top left and

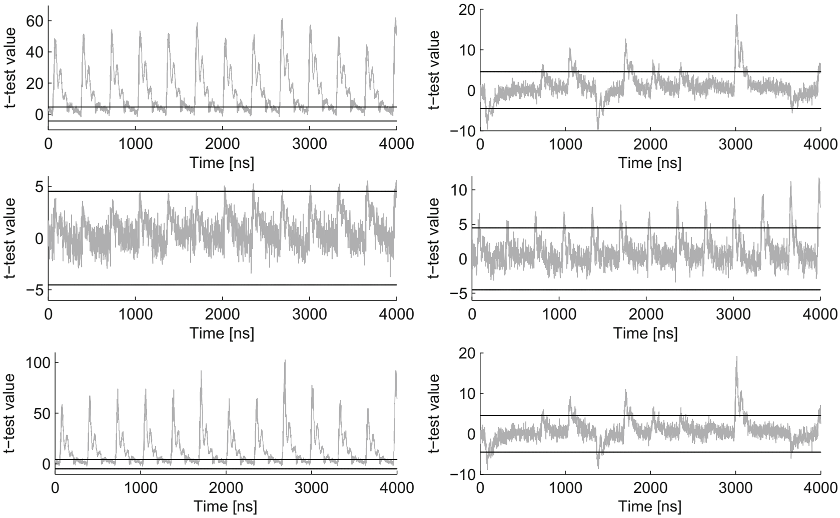

Fig. 2. Fixed versus random t-test evaluation results with PRNG switched off; left: for fixed plaintext 0x00000000, right: for a randomly chosen fixed plaintext; from top to bottom: $1^{st}, 2^{nd}$ and 3^{rd}-order statistical moment; 1 000 measurements

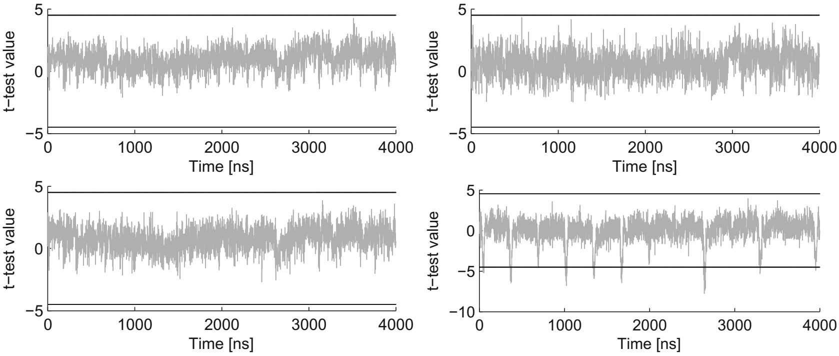

Fig. 3. Fixed versus random t-test evaluation results with PRNG switched on for a randomly chosen fixed plaintext; from top left, top right, to bottom right: $1^{st}, 2^{nd}, 3^{rd}$ and 5^{th} statistical moment; 300 million measurements

right) shows plots of the t values for the 1^{st} and 2^{nd} moment. As expected there is not sufficient evidence of leakage. But as mentioned earlier, one may always wonder if the number of measurements at hand is sufficient. For completeness, we also provide evaluation results of the 3^{rd} and 5^{th} moment. The 3^{rd} moment is the smallest moment for which our implementation does not provide provable security in the combinational logic (Property 2) and the 5^{th} moment is the smallest moment for which our implementation does not provide provable security in the memory elements (the state is shared in at least 5 shares). Therefore we may be able to detect leakage in these moments. Figure 3 (bottom left and right) shows plots of the t values.

While there is not sufficient evidence of leakage also in the 3rd moment, we can see multiple and systematic crossings of the threshold in the 5th moment. This result suggests that we use enough measurements, and that we should be able to detect leakage in the lower-order moments, if there was any. Together, the results support our claim regarding provable 2nd-order DPA resistance.

One may wonder why we do not detect leakage in the 3rd moment. Several explanations are possible but their careful investigation is beyond the scope of this paper.

5 Conclusion

Research on HO-DPA attacks shows that these attacks are realistic threats, and advances in the field can only increase the attack potential. It is therefore desirable to have masking schemes that can be implemented securely at any order. In hardware implementations, glitches make this a challenging task. TI is a masking technique that provides provable security even in the presence of glitches, but the method is limited to 1st-order DPA resistance. We address this gap and extend the technique to higher orders. We define generic constructions, prove their security and provide exemplary 1st, 2nd and 3rd-order DPA-resistant implementations of the block cipher KATAN-32. Our analysis of 300 million power traces from a 2nd-order DPA-resistant implementation in an FPGA with a leakage detection test does not show significant evidence of leakage and supports the security proofs. We also show that this method can be straightaway applied to generate 2nd-order TI of 4 × 4 S-boxes in the alternating group.

Acknowledgements. This work has been supported in part by the Research Council of KU Leuven (OT/13/071), by the FWO (G.0550.12), by the Hercules foundation (AKUL/11/19) and by GOA (tense). B. Bilgin was partially supported by the FWO project G0B4213N and Benedikt Gierlichs is a Postdoctoral Fellow of the Research Foundation - Flanders (FWO).

References

1. AIST. Side-channel Attack Standard Evaluation BOard, `http://staff.aist.go.jp/akashi.satoh/SASEBO/en/`
2. Bilgin, B., Gierlichs, B., Nikova, S., Nikov, V., Rijmen, V.: A more efficient AES threshold implementation. In: Pointcheval, D., Vergnaud, D. (eds.) AFRICACRYPT. LNCS, vol. 8469, pp. 267–284. Springer, Heidelberg (2014)
3. Bilgin, B., Nikova, S., Nikov, V., Rijmen, V., Stütz, G.: Threshold implementations of all 3 ×3 and 4 ×4 S-boxes. Cryptology ePrint Archive, `http://eprint.iacr.org/`
4. Bilgin, B., Nikova, S., Nikov, V., Rijmen, V., Stütz, G.: Threshold implementations of all 3 ×3 and 4 ×4 S-boxes. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 76–91. Springer, Heidelberg (2012)

5. Bogdanov, A.A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M., Seurin, Y., Vikkelsoe, C.: PRESENT: An ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
6. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 398–412. Springer, Heidelberg (1999)
7. Cooper, J., De Mulder, E., Goodwill, G., Jaffe, J., Kenworthy, G., Rohatgi, P.: Test vector leakage assessment (TVLA) methodology in practice. In: International Cryptographic Module Conference (2013), http://icmc-2013.org/wp/wp-content/uploads/2013/09/goodwillkenworthtestvector.pdf
8. Coron, J.-S.: Higher order masking of look-up tables. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 441–458. Springer, Heidelberg (2014)
9. Coron, J.-S., Prouff, E., Rivain, M., Roche, T.: Higher-order side channel security and mask refreshing. In: FSE 2013 (2013) (to appear)
10. Coron, J.-S., Prouff, E., Roche, T.: On the use of shamir's secret sharing against side-channel analysis. In: Mangard, S. (ed.) CARDIS 2012. LNCS, vol. 7771, pp. 77–90. Springer, Heidelberg (2013)
11. De Cannière, C., Dunkelman, O., Knežević, M.: KATAN and KTANTAN — A family of small and efficient hardware-oriented block ciphers. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 272–288. Springer, Heidelberg (2009)
12. Duc, A., Dziembowski, S., Faust, S.: Unifying leakage models: From probing attacks to noisy leakage. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 423–440. Springer, Heidelberg (2014)
13. Faust, S., Rabin, T., Reyzin, L., Tromer, E., Vaikuntanathan, V.: Protecting circuits from leakage: the computationally-bounded and noisy cases. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 135–156. Springer, Heidelberg (2010)
14. Genelle, L., Prouff, E., Quisquater, M.: Thwarting higher-order side channel analysis with additive and multiplicative maskings. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 240–255. Springer, Heidelberg (2011)
15. Gong, Z., Nikova, S., Law, Y.W.: KLEIN: A new family of lightweight block ciphers. In: Juels, A., Paar, C. (eds.) RFIDSec 2011. LNCS, vol. 7055, pp. 1–18. Springer, Heidelberg (2012)
16. Goodwill, G., Jun, B., Jaffe, J., Rohatgi, P.: A testing methodology for side channel resistance validation. NIST non-invasive attack testing workshop (2011), http://csrc.nist.gov/news_events/non-invasive-attack-testing-workshop/08_Goodwill.pdf
17. Goubin, L., Martinelli, A.: Protecting AES with Shamir's Secret Sharing Scheme. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 79–94. Springer, Heidelberg (2011)
18. Kim, H., Hong, S., Lim, J.: A fast and provably secure higher-order masking of AES S-box. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 95–107. Springer, Heidelberg (2011)
19. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)

20. Mather, L., Oswald, E., Bandenburg, J., Wójcik, M.: Does my device leak information? An *a priori* statistical power analysis of leakage detection tests. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part I. LNCS, vol. 8269, pp. 486–505. Springer, Heidelberg (2013)
21. Leander, G., Poschmann, A.: On the classification of 4 bit S-boxes. In: Carlet, C., Sunar, B. (eds.) WAIFI 2007. LNCS, vol. 4547, pp. 159–176. Springer, Heidelberg (2007)
22. Mangard, S., Popp, T., Gammel, B.M.: Side-channel leakage of masked CMOS gates. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 351–365. Springer, Heidelberg (2005)
23. Mangard, S., Pramstaller, N., Oswald, E.: Successfully attacking masked AES hardware implementations. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 157–171. Springer, Heidelberg (2005)
24. Messerges, T.S.: Using second-order power analysis to attack DPA resistant software. In: Paar, C., Koç, Ç.K. (eds.) CHES 2000. LNCS, vol. 1965, pp. 238–251. Springer, Heidelberg (2000)
25. Moradi, A.: Statistical tools flavor side-channel collision attacks. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 428–445. Springer, Heidelberg (2012)
26. Moradi, A., Mischke, O.: On the simplicity of converting leakages from multivariate to univariate. In: Bertoni, G., Coron, J.-S. (eds.) CHES 2013. LNCS, vol. 8086, pp. 1–20. Springer, Heidelberg (2013)
27. Moradi, A., Mischke, O., Eisenbarth, T.: Correlation-enhanced power analysis collision attack. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 125–139. Springer, Heidelberg (2010)
28. Moradi, A., Poschmann, A., Ling, S., Paar, C., Wang, H.: Pushing the limits: A very compact and a threshold implementation of AES. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 69–88. Springer, Heidelberg (2011)
29. Nikova, S., Rechberger, C., Rijmen, V.: Threshold implementations against side-channel attacks and glitches. In: Ning, P., Qing, S., Li, N. (eds.) ICICS 2006. LNCS, vol. 4307, pp. 529–545. Springer, Heidelberg (2006)
30. Nikova, S., Rijmen, V., Schläffer, M.: Secure hardware implementation of non-linear functions in the presence of glitches. In: Lee, P.J., Cheon, J.H. (eds.) ICISC 2008. LNCS, vol. 5461, pp. 218–234. Springer, Heidelberg (2009)
31. Nikova, S., Rijmen, V., Schläffer, M.: Secure hardware implementation of nonlinear functions in the presence of glitches. J. Cryptology 24(2), 292–321 (2011)
32. Poschmann, A., Moradi, A., Khoo, K., Lim, C.-W., Wang, H., Ling, S.: Side-channel resistant crypto for less than 2300 GE. J. Cryptology 24(2), 322–345 (2011)
33. Prouff, E., Rivain, M.: Masking against side-channel attacks: A formal security proof. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 142–159. Springer, Heidelberg (2013)
34. Prouff, E., Roche, T.: Higher-order glitches free implementation of the AES using secure multi-party computation protocols. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 63–78. Springer, Heidelberg (2011)
35. Rivain, M., Prouff, E.: Provably secure higher-order masking of AES. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 413–427. Springer, Heidelberg (2010)
36. Standaert, F.-X., Malkin, T., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 443–461. Springer, Heidelberg (2009)

A Component Functions of the Sharing F of $f(a,b,c) = 1 + a + bc$

A.1 1st-Order TI and $s_{in} = 3$

$$\begin{aligned}
y_1 &= 1 + a_2 + b_2c_2 + b_1c_2 + b_2c_1 \\
y_2 &= a_3 + b_3c_3 + b_2c_3 + b_3c_2 \\
y_3 &= a_1 + b_1c_1 + b_1c_3 + b_3c_1
\end{aligned} \tag{6}$$

A.2 2nd-Order TI and $s_{in} = 6$

$$\begin{aligned}
y_1 &= 1 + a_2 + b_2c_2 + b_1c_2 + b_2c_1 + b_1c_3 + b_3c_1 + b_2c_3 + b_3c_2 \\
y_2 &= a_3 + b_3c_3 + b_3c_4 + b_4c_3 + b_3c_5 + b_5c_3 \\
y_3 &= a_4 + b_4c_4 + b_2c_4 + b_4c_2 + b_2c_6 + b_6c_2 \\
y_4 &= a_5 + b_5c_5 + b_1c_4 + b_4c_1 + b_1c_5 + b_5c_1 \\
y_5 &= b_2c_5 + b_5c_2 + b_4c_5 + b_5c_4 \\
y_6 &= a_6 + b_6c_6 + b_3c_6 + b_6c_3 + b_4c_6 + b_6c_4 \\
y_7 &= a_1 + b_1c_1 + b_1c_6 + b_6c_1 + b_5c_6 + b_6c_5
\end{aligned} \tag{7}$$

A.3 3rd-Order TI and $s_{in} = 7$

$$\begin{aligned}
y_1 &= 1 + a_2 + b_2c_2 + b_1c_2 + b_2c_1 & y_2 &= a_3 + b_3c_3 + b_1c_3 + b_3c_1 \\
y_3 &= a_4 + b_4c_4 + b_1c_4 + b_4c_1 & y_4 &= a_5 + b_5c_5 + b_1c_5 + b_5c_1 \\
y_5 &= a_6 + b_6c_6 + b_1c_6 + b_6c_1 & y_6 &= a_1 + b_1c_1 + b_1c_7 + b_7c_1 \\
y_7 &= b_2c_3 + b_3c_2 & y_8 &= b_2c_4 + b_4c_2 \\
y_9 &= b_2c_5 + b_5c_2 & y_{10} &= b_2c_6 + b_6c_2 \\
y_{11} &= a_7 + b_7c_7 + b_2c_7 + b_7c_2 & y_{12} &= b_3c_4 + b_4c_3 \\
y_{13} &= b_3c_5 + b_5c_3 & y_{14} &= b_3c_6 + b_6c_3 \\
y_{15} &= b_3c_7 + b_7c_3 & y_{16} &= b_4c_5 + b_5c_4 \\
y_{17} &= b_4c_6 + b_6c_4 & y_{18} &= b_4c_7 + b_7c_4 \\
y_{19} &= b_5c_6 + b_6c_5 & y_{20} &= b_5c_7 + b_7c_5\,. \\
y_{21} &= b_6c_7 + b_7c_6
\end{aligned} \tag{8}$$

B Unmasked KATAN-32

The description of one round of KATAN-32 is provided in Fig. 4 where each block represents one bit. 32-bit plaintext is divided into two chunks of 13 and 19 bits and written to the registers $L1$ and $L2$ respectively. In every round, several bits are used to update the first bits of the registers together with a one bit shift to the right for $L1$ and to the left for $L2$. The bit depicted by IR is the last bit of a round counter that decides irregularly if the fourth bit of $L1$ is used for the round update or not. k_{2i} and k_{2i+1} are the $2i^{\text{th}}$ and $(2i+1)^{\text{st}}$ bits of the 80-bit key for rounds $i \leq 40$. For the rest of the rounds they are generated from the original key by an LFSR.

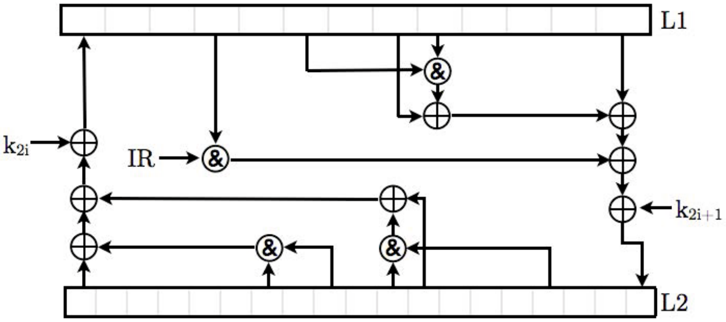

Fig. 4. Description of one round of KATAN-32

C T-test Based Fixed versus Random Leakage Detection

The t-test based fixed versus random leakage detection methodology has two main ingredients: first, chosen inputs allow to generate two sets of measurements for which intermediate values in the implementation have a certain difference. Without making an assumption about how the implementation leaks, a safe choice is to keep the intermediate values fixed for one set of measurements, while they take random values for the second set. The test is *specific*, if particular intermediate values or transitions in the implementation are targeted (e.g. S-box input, S-box output, Hamming distance in a round register, etc.). This type of testing requires knowledge of the device key and carefully chosen inputs. On the other hand, the test is *non-specific* if *all* intermediate values and transitions are targeted at the same time. This type of testing only requires to keep all inputs to the implementation fixed for one set of measurements, and to choose them randomly for the second set. Obviously, the non-specific test is extremely powerful. The second ingredient is a simple, robust and efficiently computable statistical test to determine if the two sets of measurements are significantly different. Contrary to the information-theoretic metric of Standaert et al. [36] and the mutual-information-based leakage detection tests explored in [20] the t-test based approach evaluates a specific statistical moment of the measured distributions.