Bitsliced Masking and ARM: Friends or Foes? [⋆]

Wouter de Groot[2], Kostas Papagiannopoulos[1], Antonio de La Piedra[1], Erik Schneider[2] , Lejla Batina[1]

[1] Radboud University Nijmegen, The Netherlands
[2] Eindhoven University of Technology, The Netherlands

Abstract. Software-based cryptographic implementations can be vulnerable to side-channel analysis. Masking countermeasures rank among the most prevalent techniques against it, ensuring formally the protection vs. value-based leakages. However, its applicability is halted by two factors. First, a masking countermeasure involves a *computational overhead* that can render implementations inefficient. Second, physical effects such as glitches and distance-based leakages can cause the *reduction of the security order* in practice, rendering the masking protection less effective. This paper, attempts to address both factors. In order to reduce the computational cost, we implement a high-throughput, bitsliced, 2nd-order masked implementation of the PRESENT cipher, using assembly in ARM Cortex-M4. The implementation outperforms the current state of the art and is capable of encrypting a 64-bit block of plaintext in 6,532 cycles (excluding RNG), using 1,644 bytes of data RAM and 1,552 bytes of code memory. Second, we analyze experimentally the effectiveness of masking in ARM devices, i.e. we examine the effects of distance-based leakages on the security order of our implementation. We confirm the theoretical model behind distance leakages for the first time in ARM-based architectures.

Keywords: PRESENT, ARM Cortex-M, bitslicing, masking, SCA

1 Introduction

Nowadays, everyday devices, sensors, vehicles and other items are embedded with electronics, allowing network connectivity and information exchange. Often, these fairly simple devices need to maintain a high level of security against powerful adversaries with passive monitoring, as well as active tampering capabilities.

For instance, side-channel attacks (SCA) allow attackers to learn sensitive data by observing physical characteristics of a cryptographic implementation. Their discovery in 1999 by Kocher et al. [32] exposed a blind spot in theoretical, proof-driven cryptography and has motivated researchers to find efficient

[⋆] The work described in this paper has been supported by the Netherlands Organization for Scientific Research NWO under project ProFIL (628.001.007).

countermeasures. A very common option for provably secure software countermeasures is masking [18], which uses secret-sharing techniques to hinder key recovery.

However, the masking countermeasure can imply a severe *performance overhead* in terms of processing speed due to the quadratic computational complexity required [30]. Moreover, masking can formally ensure protection against a theoretical leakage model, namely the value-based model. As a result, device-specific divergence from the assumed model can lead to security order reduction. For instance, software devices often exhibit *distance-based leakages*, which have been theorized to reduce the order of a masked scheme by 50% [2].

This paper attempts to answer whether masking countermeasures and ARM devices are *friends or foes.* The contribution is twofold and extends to both the performance factor as well as the security order factor.

First, we improve the current state of the art by creating an efficient, bitsliced, 2nd-order implementation of PRESENT. The PRESENT cipher was selected due to its widespread applicability in the Internet of Things context. Our implementation requires 1,644 bytes of RAM, 1,552 bytes of code memory and encrypts 32 blocks of data in 209,023 clock cycles, achieving a throughput of 6,532 clock cycles per block, excluding the cost of random number generation. Thus, we demonstrate that ARM-based architectures can host masked implementations efficiently, given that the implementors opt for full-scale assembly programs and use efficient state representations.

Second, we examine potential distance-based leakages in ARM architectures. That is, we perform side-channel experiments in order to test whether our ARM Cortex-M4 device is prone to causing order reduction in our 2nd-order implementation. In addition, we confirm that the observed order reduction follows the theorized reduction established by Balasch et al. [2]. That is, we confirm the *order-reduction theorem* (Section 5) in ARM-based architectures for the first time.

In the next section, we describe the work of other practitioners who implemented PRESENT and relate their performance figures with our work. In Section 3 we offer a brief description of the PRESENT cipher. Section 4 discusses the design options and optimizations w.r.t. the masked ARM implementation, as well as the performance results. Section 5 links the order reduction model suggested by Balasch et al. [2] to our ARM-based device. Finally, Section 6 concludes and discusses future work.

2 Related work

In this section, we describe the work of those implementors that addressed the implementation of PRESENT in software. We do this in ascending order of word size according to the architecture.

4-bit architectures Poschmann implemented PRESENT in different software platforms [39]. In a 4-bit μC, particularly an Atmel ATAM893-D at 2 MHz he obtained a performance figure of 55,734 cycles per block. He also implemented PRESENT in an 8-bit ATmega μC clocked at 4 MHz, obtaining a performance of 10,089 cycles.

8-bit architectures Papagiannopoulos presented a bitsliced implementation of PRESENT on the 8-bit ATtiny85 μC. He applied bitslicing to the permutation and substitution layers using a bitslice factor of 8 [38]. That work relied on the PRESENT Sboxes resulting from the application of 2-stage Boyar-Peralta heuristic in tandem with SAT solvers [12]. He obtained a throughput (cycles per block) of 2,967 using 3,816 bytes of Flash and 256 bytes of SRAM. In this work, we use the same Sbox. Dinu et al. also analyzed the suitability of a wide range of lightweight block ciphers in sensor-based applications in three different architectures: an 8-bit ATmega, 16-bit MSP430 and 32-bit ARM processor. They do not apply bitslicing and implemented the Cipher Block Chaining (CBC) and counter (CTR) modes of operation [23]. The CBC implementation requires 121,906 cycles on the ATmega processor whereas the CTR implementation can obtain one block of ciphertext in 15,239 cycles. Furthermore, the authors from [25] implement PRESENT in an ATiny 8-bit μC, using 80 bits keys the required 11,343 cycles, 1,000 bytes of code and 18 bytes of RAM. Using the same platform Papagiannopoulos decreased the amount of cycles to 8,712 cycles in [37] by using a merged SP layer, squared and compact representations of the Sbox and minimal key register rotations. Finally Rauzy et al. presented a design methodology for inserting Dual-rail with Precharge Logic (DPL) in a software implementation of PRESENT in an automatic way [41]. They relied on an 8-bit AVR ATmega 163 implementation (bitsliced). They require 235,427 cycles for obtaining a single block of ciphertext.

16-bit architectures Poschmann also implemented PRESENT on an 16-bit Infineon C167CR processor, obtaining a performance figure 19,460 cycles per block [39]. On the other hand, Dinu et al. relied on the MSP430 of 16-bit for implementing both the CBC and CTR modes of PRESENT, obtaining a performance of 100,786 and 12,226 cycles respectively. In [17], Cazorla et al. evaluated a variety of lightweight primitives on the 16-bit MSP430 μC that sensor nodes usually equip due to its low-power and cost. Clocked at 8 MHz, their performance figures are 364,587 cycles and 45,573 cycles/byte (they do not employ bitslicing).

32-bit architectures Dinu et al. implemented PRESENT on the ARM Cortex architecture [23]. Their CBC implementation requires 138,947 cycles on the ARM processor whereas their CTR implementation can obtain one block of ciphertext in 16,919 cycles.

64-bit architectures Benadjila et al. explored the software implementation of the LED, Piccolo and PRESENT block ciphers [5, 29, 43]. They relied on table-based implementations, vector permutations and bitslice approaches. The best results for bitsliced PRESENT-80 are 18.7 cycles/byte for 16 plaintexts in 2,221 cycles in an Intel Core i3 2367M clocked at 1.4 GHz. Matsuda et al. proposed in [34] the utilization of PRESENT in sensor-related applications for processing a high-amount of data gathered by nodes. They relied on 3 Intel architectures, particularly on the Core 45 nm and Nehalem (equipped with the Streaming SIMD extensions (SSE) 4.1 and 128-bit XMM registers) and on the Sandy Bridge, equipped with the Advanced Vector Extension (AVX). Executing 32 plaintexts simultaneously via a bitsliced implementation, the require 4.73 cycles/byte on the Sandy Bridge architecture.

Contribution In this manuscript we present a very fast and 2nd-order protected implementation of the PRESENT block cipher by combining bitslicing and 2nd-order masking. We rely on the 32-bit ARM Cortex-M4 CPU [3]. The analytical results can be seen in section 4.3. Our implementation can encrypt one PRESENT plain text in 6,532 cycles using 1,644 bytes of RAM and 1,552 bytes of ROM. To our knowledge, this is the first high-order protected implementation of PRESENT that includes side-channel evaluation. We have evaluated our implementation against first, second and third-order security using state-of-the art techniques (Section 5). None of the works described in this section performed such exhaustive evaluation on their implementation while protecting it against second-order attacks [5, 17, 23, 25, 34, 37–39, 41]. Our performance figures suggest that our implementation is between 2.5 and 21.2 times faster than prior art relying on the same architecture (Section 4.4). Further, we have made our implementation available under the General Public License (GPL)[4].

Finally, since the constructions found in PRESENT are also used on the hash functions SPONGENT and H-PRESENT [8, 10], the same approaches we present in this manuscript can be applied to their implementation.

3 PRESENT

Given the need of alternative cryptographic primitives aimed at low-power and compact applications such as RFID and sensor networks, a variety of lightweight primitives such as PRESENT has been proposed in the last few years [9]. Standardized in ISO/IEC 29192-2:2012 [5], it consists of a substitution-permutation (SP) network, 80/128-bit key sizes and 64-bit data blocks. PRESENT applies the following layers during 31 rounds to a 64-bit state b:

[3] In particular, we used an STM32F417IG SoC by ST clocked at 168 MHz with 1,024 Kbytes of Flash and 196 Kbytes of RAM.

[4] `http://tinyurl.com/zw7zlkv` (Accessed 24 June 2016)

[5] `http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=56552` (Accessed 24 June 2016)

1. **addRoundKey**: During the execution of PRESENT, 32 round keys (K_i w.r.t. $1 \leq i \leq 32$) are generated via a key schedule using the encryption key K as an input. The last subkey, K_{32} is used for post-whitening. Each round key K_i has a size of 64 bits. Thus, each execution of **addRoundKey** is comprised of the XOR operation between the state and the round key, i.e. $b' \leftarrow b \oplus K_i$.
2. **sBoxLayer**: This layer is a non-linear substitution operation that relies on a 4-bit Sbox ($\mathbb{F}_2^4 \rightarrow \mathbb{F}_2^4$), applied 16 times per round to the state. The 64-bit state is divided in 16 groups of 4 bits that feed the PRESENT Sbox (Table 1).

Table 1: 4-bit PRESENT Sbox

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S[x]$	c	5	6	b	9	0	a	d	3	e	f	8	4	7	1	2

3. **pLayer**: This layer consists of a linear bit-wise permutation where each bit i of the state (b_i) is moved to another position $P(i)$ according to Table 2.

Table 2: Permutation table of PRESENT

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(i)$	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(i)$	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P(i)$	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P(i)$	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

Finally, the round subkeys are generated as follows. Given a key K of 80 bits s.t. $K_{79}, K_{78}, ..., K_0$, a round key i of 64 bits is the 64 left most bits of K updated via the following operations:

1. 61 bits rotations to the left of K.
2. The left most 4 bits are processed in the PRESENT Sbox.
3. The round counter is exclusive-ored with the bits $K_{19}, ..., K_{15}$ of K.

4 Bitsliced Masking of PRESENT for ARM Cortex-M4

The current section describes the design choices investigated in order to develop a protected, high-throughput, assembly-based PRESENT implementation. Sections 4.1, 4.2 describe the logic-level optimizations performed, while Section 4.3 discusses the instruction-level improvements.

4.1 Bitslicing and Efficient Sbox Representation

CPU architectures tend to operate best on their native word size or half-words and they encounter performance issues with bit-level manipulation. To deal with this issue, the Cortex-M4 features bit-banding support[6], as well as a wide selection of bit-field instructions. However, applying them in the context of PRESENT requires extensive use of load and store instructions or numerous bit extractions/insertions, often resulting in poor performance.

Bitslicing is a technique introduced by Biham to tackle this inefficiency for DES [6]. Instead of using registers to store consecutive bits of a state, one uses them to hold one specific bit from several different states, effectively transforming bit-level operations into SIMD equivalents.

In our implementation, we employ a bitsliced representation of factor 32, i.e. we process in parallel 32 cipher blocks, 64 bits each, resulting in 256 bytes per bitsliced encryption. Doing so, allows us to efficiently compute both the substitution and the permutation layer of PRESENT. Analytically, the Sbox can be decomposed into $GF(2)$ operations which can be accelerated by via the SIMD-like instructions and it no longer requires the application of memory lookup tables.[7]. Similarly, the bit permutations can be accelerated by directly exchanging the memory contents of the corresponding bitsliced bits according to the permutation pattern, instead of relying on bit extraction, insertion and shifting.

The $GF(2)$ decomposition of the Sbox has sparked interest in the optimization of boolean circuits w.r.t. computational efficiency. In our implementation, we use the optimized boolean circuit suggested for PRESENT by Courtois et al. [21]. The optimized representation was generated by applying the Boyar-Peralta heuristic [12], which reduces the circuit's *gate complexity*, i.e. the number of AND, OR, XOR, NOT operations. The representation is shown below.

```
T1 = X2^X1;    T2 = X1&T1;    T3 = X0^T2;  Y4 = X3^T3;  T2 = T1&T3;
T1 ^= Y4;      T2 ^= X1;      T4 = X3|T2;  Y3 = T1^T4;  X3 =~ X3;
T2^ = X3;      Y1 = Y3^T2;    T2 |= T1;    Y2 = T3^T2;
```

Values X1–X4 represent an Sbox input, T1–T4 hold temporary values and Y1-Y4 are output values. The total cost is 14 operations, 4 non-linear (AND, OR) and 10 linear (XOR,NOT).

4.2 Boolean Masking

Chari et al. [18] were among the first to suggest that splitting intermediate values using a secret sharing scheme would force attackers to analyze joint distribution

[6] Bit-banding allows individual bits to be addressed as though they were bytes in RAM.

[7] Note that implementations based on lookup tables can be prone to timing side-channel attacks in the presence of memory caches.

functions on multiple points. That is, a dth-order masking scheme splits a sensitive value x into $d+1$ shares $(x_0, x_1, \ldots, x_d)$ as follows:

$$x = x_0 \oplus x_1 \cdots \oplus x_d \tag{1}$$

Assuming sufficient noise, it has been shown that the number of traces required for a successful attack grows exponentially w.r.t. the order d [18, 40].

Masking involves several implemenation angles, e.g. Goubin et al. [26], Messerges [35] and recently Coron [19] applied the masking principle in *lookup tables* used in Sbox computation. Adopting a different implementation angle, Trichina [47], Canright [14], Akkar et al. [1] and Blömer et al. [7] applied masking in the context of *GF operations* used in Sbox computation. This operation-based approach was formalized by Ishai, Sahai, and Wagner's shared secret approach (ISW), which introduced the notion of private boolean circuits [30]. ISW provided implementors with a provably secure method to mask operations in $GF(2)$ for any masking order d.

This work employs a bitsliced representation of PRESENT and enhances the implementation using a 2nd-order protection scheme. As demonstrated in Section 4.1, the Sbox is decomposed into $GF(2)$ operations. Thus, ISW is our technique of choice in order to apply 2nd-order protection on the boolean operations required for the Sbox computation.

Table 3 shows the ISW equivalent of common boolean operations when applied to bitsliced operands a and b, as well as the computational cost involved for each operation. The values $z_{i,j}$ where $1 \leq i < j \leq (d+1)$ are drawn from a uniform random distribution and the remaining $z_{i,j}$ are computed using $(z_{i,j} \oplus a_i b_j) \oplus a_j b_i$. Note that the cost of the NOT operation is a single negation, the cost of the XOR operation is linear and the cost of the AND,OR operations is quadratic. In our implementation, the OR operation is converted to a single AND and three NOT operations in order to apply the ISW method.

Table 3: ISW equivalents of common boolean operations

Operation	**ISW Equivalent**	**Cost**
NOT(a)	$\neg a_0$	$\mathcal{O}(1)$
XOR(a,b)	$a_i \oplus b_i$	$\mathcal{O}(d)$
OR(a,b)	**NOT(AND(NOT(a),NOT(b)))**	$\mathcal{O}(d^2)$
AND(a,b)	$a_i b_i \oplus \bigoplus_{i \neq j} z_{i,j}$	$\mathcal{O}(d^2)$

The quadratic computational complexity of non-linear operations can result in a computationally demanding masked Sbox. To avoid this, several techniques [15, 16, 21, 27, 45] on reducing the *multiplicative complexity of an Sbox*, i.e the number of AND,OR operations. The decomposition that we currently use (shown in Section 4.1) is optimal w.r.t. multiplicative complexity, since brute-force techniques [28] demonstrate that the minimal complexity in $GF(2)$ of cryptographically relevant, 4-bit Sboxes is 4 non-linear operations.

4.3 ARM-based Optimizations

Our implementation targets the ARM Cortex-M4 microcontroller architecture using ARM assembly with Thumb2 encoding. Thus, we use a 32-bit architecture with 14 general purpose registers designed for low-cost, low-power applications. The implementation board is the Riscure Pinata [8] which is based on the STM32F417IG SoC by ST and embeds an ARM 32-bit Cortex-M4 CPU clocked at 168 MHz. It features 1,024 Kbytes of Flash and 196 Kbytes of RAM. The device is also equipped with a TRNG on the board in order to generate the random values associated to our masking implementation. In the case of the STM32F417IG, the TRNG generates 32-bit random numbers via an integrated analog circuit. Note that the computational penalty w.r.t. random number generation is particularly steep when implemented on-the-fly, amounting to roughly 25 percent of the total computation. Still, we note that the random numbers can be precomputed in advance, given that the application context allows for idle intervals between consecutive encryptions. Below, we discuss implementation details and efficiency improvements pertaining to the ARM architecture, memory organization and assembly instructions.

1. **Memory organization**: Our design requires two full bitsliced states in RAM, each comprising of three sub-states corresponding to the three-share masking scheme. The two full bitsliced states are needed because the permutation layer would otherwise overwrite unprocessed data. We optimize for cycles by integrating the permutation into the Sbox and writing words to their permuted destination immediately after the Sbox computation.
   Wherever the code operates on shares we organize our fetch and store data in batches so as to reduce overhead. In most cases we use the LDM and STM instructions to load or store three or four words at a time. This yields improvements in the Sbox computation when reading in the next four words to be substituted, in the key schedule, where three words at a time are read in for processing and also when converting a regular state representation from/to a bitsliced one.
2. **Loop Unrolling**: To improve the efficiency of our Sbox implementation, which encrypts twelve shares (four bit-sliced data blocks of three shares each), we unroll the substitution process to reduce the unnecessary read/write steps required for a looped construction. The unrolling adds considerable size to the code, yet we achieve trading code size for throughput. Note that unrolling is performed with memory access in mind. For example, we mentioned that adding the key schedule is performed in a loop of three words. This optimizes the key schedule operation and maximizes the amount of data we can bring from/to the RAM.
3. **Key Schedule**: The round key is not stored in a bitsliced fashion and the key schedule is computed on the fly. Note that round key precomputation is also a valid implementation option, assuming that the key does not need to

[8] `https://www.riscure.com/security-tools/hardware/pinata` (Accessed 24 June 2016)

be renewed often. Since, key refreshing can act as a side-channel countermeasure, we chose to retain the on-the-fly key updates. Updating the round key requires a push through the Sbox for four bits each round. To that purpose, we use Cortex-M4's `UBFX` instruction for extracting a contiguous series of bits from a word in an efficient manner. In addition, we used ARM's barrel shifter function, which allows the second operand to be shifted with no additional cost before an instruction is performed.

4.4 Performance Results

The current section summarizes the achieved performance results w.r.t. throughput and size. We depict in Tables 4, 5 the performance figures of the works described in Section 2. As mentioned, we outperform prior art on the same architecture between 2.5 and 21.2 times [23].

Table 4: PRESENT implementations, comparison with prior art (performance)

Work	**Implementation**	**Bitslicing**	**Bitslicing factor**	**Protected**	**Platform**	**No. cycles per block**
This work	PRESENT-80	**yes**	**32**	**yes**	**ARM Cortex–M4**	**6,532**
[23]	PRESENT-80, CBC	no	-	no	ATmega	121,906
[23]	PRESENT-80, CBC	no	-	no	MSP430	100,786
[23]	PRESENT-80, CBC	no	-	no	ARM Cortex-M3	138,947
[23]	PRESENT-80, CTR	no	-	no	ATmega	15,239
[23]	PRESENT-80, CTR	no	-	no	MSP430	12,226
[23]	PRESENT-80, CTR	no	-	no	ARM	16,919
[37]	PRESENT-80	no	-	no	ATiny	8,721
[41]	PRESENT-80	**yes**	8	no	ATMega163	78,403
[41]	PRESENT-80, DPL	**yes**	8	**yes**	ATMega163	235,427
[38]	PRESENT-80	**yes**	8	no	ATiny85	2,967
[5]	PRESENT-80, table	no	-	no	Corei3-2367M	988
[5]	PRESENT-80, vperm	**yes**	2	no	Corei3-2367M	890
[5]	PRESENT-80	**yes**	8	no	Corei3-2367M	2,039
[5]	PRESENT-80	**yes**	16	no	Corei3-2367M	3,138
[34]	PRESENT-80	**yes**	32	no	Xeon E3-1280	37.84
[34]	PRESENT-80	**yes**	16	no	Xeon E3-1280	52.16
[34]	PRESENT-80	**yes**	8	no	Xeon E3-1280	67.68
[17]	PRESENT-80	no	-	no	MSP430	364,587
[39]	PRESENT-80	no	-	no	ATAM893-D	55,734
[39]	PRESENT-80	no	-	no	ATMega163	10,089
[39]	PRESENT-80	no	-	no	C167CR	19,460

As expected, the ISW implementation of the Sbox dominated CPU time, accounting for 95,88 percent of all clock cycles within the encryption process. A complete breakdown of the memory and time overheads required for different modules is provided in Table 6.

5 Masking Effectiveness in ARM Cortex-M4

In this section, we assess experimentally the security level (masking order) provided by the ISW masking scheme, taking into account the possibility of distance-based leakages in ARM Cortext-M4. In addition, we investigate whether the

Table 5: PRESENT implementations, comparison with prior art (size)

Work	Implementation	Code (bytes)	RAM (bytes)
This work	PRESENT-80	1,548	1,644
[38]	PRESENT-80	3,816	256
[39]	PRESENT-80, ATMega	1,494	272
[39]	PRESENT-80, C167CR	$45.9 \cdot 10^3$	-
[23]	PRESENT-80, CBC, ATMega	1,388	56
[23]	PRESENT-80, CBC, MSP430	1,108	52
[23]	PRESENT-80, CBC, ARM	1,304	124
[23]	PRESENT-80, CTR, ATMega	1,416	54
[23]	PRESENT-80, CTR, MSP430	1,244	58
[23]	PRESENT-80, CTR, ARM	1,532	140
[37]	PRESENT-80	1,794	-
[41]	PRESENT-80, bitslicing	1,620	288
[41]	PRESENT-80, bitslicing + DPL	3,056	352

Table 6: SW transformations of common logical operations

Operation	Code Size (%)	No. Cycles (%)
main	208 (13.44)	3,807 (1.82)
sbox	892 (57.62)	200,404 (95.88)
updatekey	146 (9.43)	1,688 (0.81)
addroundkey	176 (11.37)	1,209 (0.58)
split data	60 (3.88)	1,292 (0.62)
unsplit data	66 (4.26)	623 (0.30)

theoretical repercussions of distance-based leakages can be confirmed experimentally. In other words, we examine whether the cost of "lazy engineering" as introduced by Balasch et al. [2] is applicable to an ARM-based microcontroller.

5.1 Experimental Pitfalls

The *effective* and *efficient* evaluation of the *actual mask order* of cryptographic implementations remains an open problem due to several evaluation pitfalls.

Effectivity-wise, when evaluating a masking scheme via the measured power consumption, we face the pitfall of the *limited attack scope.* That is, a particular attack technique in use may fail to exploit the available leakage due to e.g. an unsuitable choice of intermediate values or an incorrect power model assumption[9]. Moreover, introducing additional countermeasures on top of the

[9] Knowledge about the device can often be limited in the context of black-box evaluations.

masking scheme may render particular exploitation techniques ineffective, while the implementation remains vulnerable to different lines of attack.

In order to tackle this issue, the research community followed several approaches. Prior research established generic side-channel distinguishers such as Mutual Information Analysis (MIA) [4], the Kolmogorov-Smirnov and the Cràmer-von Mises tests [48,49], which require minimal assumptions about the noise and the power model of the device under test. On the other side of the spectrum, Standaert et al. [44] proposed an evaluation framework assuming the strongest possible adversary, equipped with extensive profiling capabilities and Bayesian templates.

While being effective, the aforementioned approaches focus on leakage *exploitation* and perform key recovery, which may require a large number of traces. Thus, they face the *efficiency* pitfall w.r.t. computational and storage requirements. Note that this increased demand for resources is magnified when inserting extra countermeasures in a masked implementation. Thus, it can be difficult to decide with confidence whether the masking order is reduced or not.

In order to evaluate the *effective masking order*, we opt for a more recent approach called *leakage detection methodology* [31]. This approach focuses on leakage *detection* and disregards exploitation. Thus, the acquisition and the computational cost is reduced while the methodology can retain its generic nature.

Despite the gain achieved via decoupling detection and exploitation, the leakage detection methodology still presents challenges w.r.t. efficiency. In the context of software masking, we need to combine multiple time samples in order to evaluate the masked implementation. Thus, we rely on the work by Schneider et al. [42], who extended the leakage detection methodology into higher-order evaluations by providing efficient, incremental formulas that can handle the computation involved with minimal memory requirements. In certain cases, we also resort to traditional evaluation techniques such as correlation-power analysis (CPA) [13], despite their limited attack scope, so as to enhance our discussion.

5.2 Bitsliced Masking and Distance-Based Leakages

In order to perform leakage detection and determine the actual masking order, we opt to use the *fixed vs. random*, *non-specific* t-test statistic. The process involves two steps: a custom acquisition of two trace sets (populations) and a population comparison based on statistical inference.

In the first step, we perform a *fixed vs. random* acquisition and obtain two distinct trace sets for comparison: S_{fixed} and S_{random}, under the same encryption key. For S_{fixed}, the input plaintext is set to a fixed value, while for S_{random}, the input is drawn from a uniformly random distribution. Following the suggestion from Shneider et al. [42], the implementation receives the fixed or random plaintext in a non-deterministic and randomly-interleaved manner. This type of acquisition is performed in order to randomize the implementation's internal state and avoid measurement-related variations over time, e.g. due to environmental parameters. The evaluation test to be performed is *non-specific*, i.e. we target all sensitive values computed during encryption. Thus, we maintain a wide attack scope, without any prior assumptions on the leakage model or intermediate values.

The acquisition is performed on the ARM-based Pinata device, using a Picoscope 5203 oscilloscope and the Riscure current probe [10]. The device clock operates on 168 MHz and the oscilloscope's sample rate is 1 GSample/sec. We also apply post-processing in the form of signal resampling.

For the second step, we model the sets S_{fixed} and S_{random} as independent random samples $\{S^1_{fixed} \dots S^n_{fixed}\}$ and $\{S^1_{random} \dots S^m_{random}\}$ drawn from normal distributions with means $\mu_{fixed}, \mu_{random}$, standard deviations $\sigma_{fixed}, \sigma_{random}$ and $\sigma_{fixed} \neq \sigma_{random}$. Subsequently, leakage detection methods will test the equality of means μ_{fixed}, μ_{random} (null hypothesis). Finding a statistic for this test is known as the Behrens-Fisher problem and an approximate solution is the Welch t-test [33] with υ degrees of freedom, as shown below.

$$\begin{aligned} H_{null}: &\quad \mu_{fixed} = \mu_{random} \\ H_{alt}: &\quad \mu_{fixed} \neq \mu_{random} \end{aligned} \tag{2}$$

$$w = \frac{\mu_{fixed} - \mu_{random}}{\sqrt{\frac{\sigma^2_{fixed}}{n} + \frac{\sigma^2_{random}}{m}}} \tag{3}$$

$$\upsilon = \frac{(\frac{\sigma^2_{fixed}}{n} + \frac{\sigma^2_{random}}{m})^2}{\frac{\sigma^4_{fixed}}{n^2(n-1)} + \frac{\sigma^4_{random}}{m^2(m-1)}} \tag{4}$$

The null hypothesis H_{null} is rejected at a given level of significance α, if $|w| > t_{\alpha/2,\upsilon}$, where $t_{\alpha/2,\upsilon}$ is the value of the Student t distribution with υ degrees of freedom[11]. In the evaluation context, rejecting H_{null} implies leakage detection, i.e. potential evidence of an ineffective masking scheme. A common rejection criterion that we also use in our analysis is $|w| > 4.5$, which corresponds to $\upsilon > 1000$ and $\alpha > 0.99999$ [22]. Note that that H_{null} rejection shouldn't be interpreted directly as an applicable vulnerability. Even after detection, the amount of traces required for exploitation may render an attack infeasible.

In this work, we need to evaluate the masking order provided by our ARM-based, 2nd-order masked cipher. From a theoretical point of view, a 2nd-order ISW masking countermeasure is capable of preventing value-based leakages of order 2 or less. However, practice has demonstrated that software implementations, including ARM microcontrollers, may exhibit leakages with large divergence from the value-based leakage abstraction. An exemplary case is the distance-based leakage model, observed by Daemen et al. [46], addressed by Coron et al. [20] and recently formalized by Balasch et al. [42]. This particular divergence leads in the reduction of the security order. Balasch et al. theorized

[10] `https://www.riscure.com/security-tools/hardware/current-probe` (Accessed 24 June 2016)

[11] Note that side-channel analysis usually employs two-tailed tests.

that a dth-order scheme can reduce to order $\lfloor \frac{d}{2} \rfloor$ and provided experimental validation using an AVR-based microcontroller. We will refer to this formalization as the *order-reduction theorem*. To address such leakage divergence issues in our implementation, we use the Welch t-test in order to verify *experimentally* the theoretical security claims.

We commence the evaluation by testing the 1st-order security of our masked cipher. We perform the 1st-order t-test on the first round of bitsliced PRESENT. The size of both S_{fixed} and S_{random} is 10k traces with 30k samples per trace. The trace waveform and t-test results are visible in Figures 1,2. We observe that that we remain well below the 4.5 threshold, indicating that our 2nd-order masked PRESENT implementation is able to maintain 1st-order security.

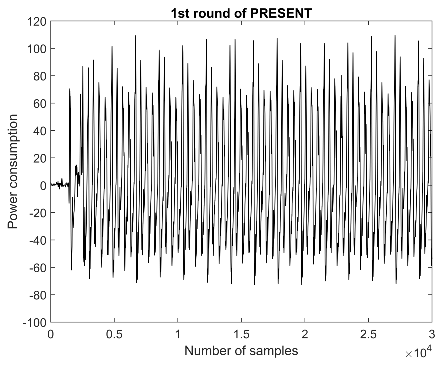

Fig. 1: Trace waveform of 1st round, masked, bitsliced PRESENT after resampling.

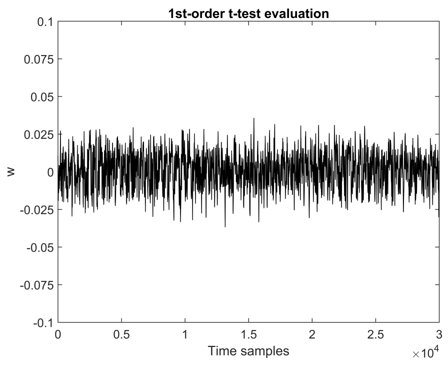

Fig. 2: 1st-order t-test evaluation for 2nd-order masked PRESENT cipher. The results suggest absence of 1st-order leakage.

To enhance our confidence, we also perform a 1st-order CPA attack, with a large amount of traces (800k) to exploit potential 1st-order leakages. We use the HW model and a custom-made selection fuction due to the bitsliced Sbox computation. Similarly to Balasch et al. [3], the selection function must take into account that not all Sbox output bits leak at the same time due to the $GF(2)$-oriented Sbox implementation. Thus, our selection function focuses on key bits from different registers that once combined through the Sbox, affect a single bit of the Sbox output. Attacking a section of the 1st round with 10k traces, while the RNG is disabled, is successful, confirming the validity of our choice w.r.t. the leakage model (HW) and selection function. The results are visible in Figure 3. We also perform the CPA attack with enabled RNG and the results are visible in Figure 4. In order to manage the computation required, we employ the techniques suggested by Bottinelli et al. [11], i.e. we partition the 800k traces, compute correlation coefficient per partition, then recombine in order to reduce the execution and memory workload.

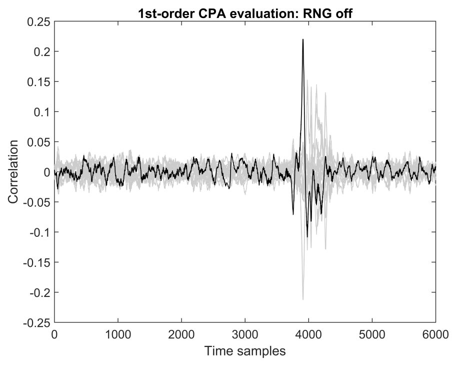

Fig. 3: 1st-order CPA attack results with RNG turned off, in selected section of the 1st round.

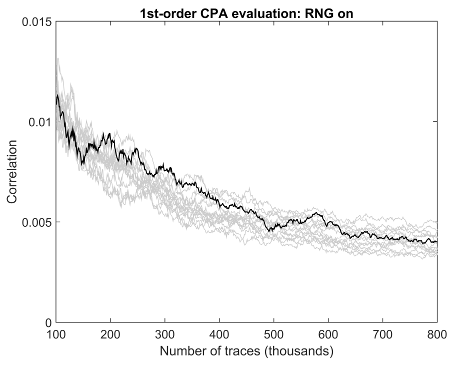

Fig. 4: 1st-order CPA attack results with RNG turned on, from 100k to 800k traces. The attack does not exploit any leakage.

The results demonstrate that no 1st-order leakage can be exploited in the presence of our 2nd-order scheme. Both the t-test and the CPA result is in accordance with the order-reduction theorem, since a 2nd-order masked implementation can maintain $\lfloor \frac{2}{2} \rfloor = 1$ order of security in the presence of distance-based leakages.

Assuming that our device exhibits distance-based leakage, it is of particular interest to prove experimentally that the order-reduction theorem holds when we test the 2nd-order security of our ARM-based masked implementation. Performing a 2nd-order evaluation requires pre-processing the acquired trace sets in order to generate all possible 2-tuples (pairs) of distinct samples via a combination function. Subsequently, the multivariate 2nd-order t-test is performed on the generated trace sets in order to determine the robustness of the 2nd order.

The main hindrance of this process is the computational complexity pertaining to generating and processing all $\binom{NoSamples}{2}$ sample pairs. Even with a small number of samples per trace, the evaluation cost can quickly become prohibitive. To address this issue, researchers have relied on intuitive selection of points of interest in conjunction with naive search [36] or they deployed heuristic techniques such as projection pursuits [24] to perform point of interest selection for higher-order attacks. In our evaluation, we follow the intuitive approach by focusing on a reduced version of the 1st round which contains the substitution layer. Inside this reduced round, we enumerate naively all possible pairs. Given the bitsliced nature of the implementation and the considerable RNG overhead, the reduced round has a length of 800 samples. In order to keep the processing cost manageable, we use the incremental formulas suggested by Schneider et al. which enable the efficient computation of the multivariate statistical moments required for 2nd-order t-tests. The memory-less feature of the computation yields significant improvement compared to straightforward computation techniques. In addition, we partition the reduced round into windows of 150 samples each and perform the attack in each window independently. Figure 6 shows the t-test results using 10k fixed input traces and 10k random input traces for the sample window with the largest detected leakage.

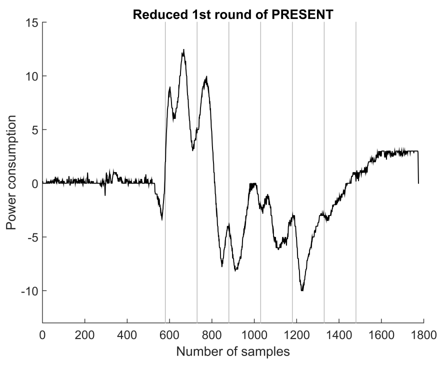

Fig. 5: Trace waveform of reduced 1st round masked, bitsliced PRESENT.

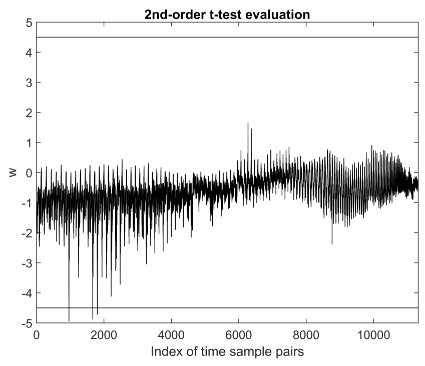

Fig. 6: 2nd-order t-test results. The rejection of H_{null} indicates potential leakage.

The test value slightly exceeds the threshold, indicating potential leakage. Thus, it hints the experimental verification of the order-detection theorem in our ARM-based device for 2nd-order ISW schemes. However, several concerns were raised over the t-test robustness, usually w.r.t. the exact threshold value (Appendix A from [2], [22]). As a result, it remains an open question whether 2nd-order leakages are practically exploitable in our context. To investigate this, we perform a 2nd-order CPA-based attack using the centered product combination function and the custom bitsliced selection function on the 1st round of PRESENT. The point selection window has size 100 samples and we use 100k traces. The results are visible in Figure 7 and show that the leakage is *exploitable* with roughly 60k traces.

As a result, we suggest that the order-reduction theorem remains applicable in software-based, masked implementations for the ARM Cortex-M4. However, we recommend that the exploitation is always verified in practice.

Moreover, we need to stress the fact that this type of behavior has been observed in a *specific* ARM-based device. Although it provides indications on the behavior of similar architectures, this experimental result should not be extrapolated as a hard fact w.r.t. all ARM Cortex-M devices. Naturally, a 3rd-order multivariate t-test is able to detect a large amount of leakage, as shown in Figure 8 and indicates that a 3rd-order attack is also applicable.

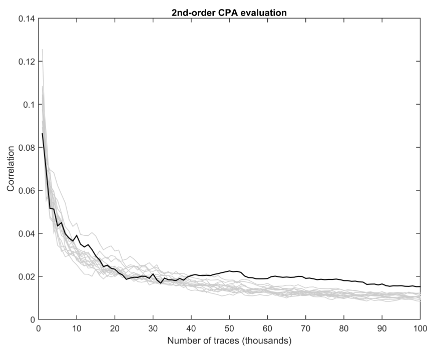

Fig. 7: 2nd-order CPA on section of the 1st round exploiting the available leakage.

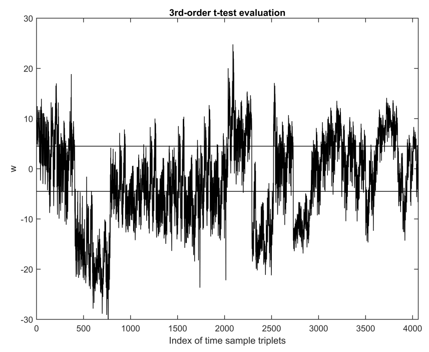

Fig. 8: 3rd-order t-test results on a section of the 1st round, indicating strong 3rd-order leakage.

6 Conclusions

This paper investigated the speed and space requirements of a bitsliced implementation of PRESENT on the ARM Cortex-M4 architecture, protected with 2nd-order ISW masking. In addition, we explore and confirm the applicability of the order-reduction theorem in the context of ARM-based devices. From the attacker point of view, future work can involve deciding on the optimal strategy to attack masked implementations, given the amount of leakage available in different security orders. From the defender's point of view, implementors need to also investigate the computational cost of the randomness required for masking, which itself may pose a bigger issue than the quadratic computational complexity of masking.

7 Acknowledgments

We would like to thank Rafael Boix–Carpi from *Riscure BV* for his advice and help.

References

1. Mehdi-Laurent Akkar, Régis Bevan, and Louis Goubin. Two power analysis attacks against one-mask methods. In Bimal K. Roy and Willi Meier, editors, *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers*, volume 3017 of *Lecture Notes in Computer Science*, pages 332–347. Springer, 2004.
2. Josep Balasch, Benedikt Gierlichs, Vincent Grosso, Oscar Reparaz, and François-Xavier Standaert. On the cost of lazy engineering for masked software implementations. In *Smart Card Research and Advanced Applications - 13th International*

Conference, CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers, pages 64–81, 2014.

3. Josep Balasch, Benedikt Gierlichs, Oscar Reparaz, and Ingrid Verbauwhede. Dpa, bitslicing and masking at 1 ghz. In *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, pages 599–619, 2015.
4. Lejla Batina, Benedikt Gierlichs, Emmanuel Prouff, Matthieu Rivain, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. Mutual information analysis: a comprehensive study. *J. Cryptology*, 24(2):269–291, 2011.
5. Ryad Benadjila, Jian Guo, Victor Lomné, and Thomas Peyrin. Implementing lightweight block ciphers on x86 architectures. In Tanja Lange, Kristin E. Lauter, and Petr Lisonek, editors, *Selected Areas in Cryptography - SAC 2013 - 20th International Conference, Burnaby, BC, Canada, August 14-16, 2013, Revised Selected Papers*, volume 8282 of *Lecture Notes in Computer Science*, pages 324–351. Springer, 2013.
6. Eli Biham. A fast new DES implementation in software. In Eli Biham, editor, *Fast Software Encryption, 4th International Workshop, FSE '97, Haifa, Israel, January 20-22, 1997, Proceedings*, volume 1267 of *Lecture Notes in Computer Science*, pages 260–272. Springer, 1997.
7. Johannes Blömer, Jorge Guajardo, and Volker Krummel. Provably secure masking of AES. In Helena Handschuh and M. Anwar Hasan, editors, *Selected Areas in Cryptography, 11th International Workshop, SAC 2004, Waterloo, Canada, August 9-10, 2004, Revised Selected Papers*, volume 3357 of *Lecture Notes in Computer Science*, pages 69–83. Springer, 2004.
8. Andrey Bogdanov, Miroslav Knezevic, Gregor Leander, Deniz Toz, Kerem Varici, and Ingrid Verbauwhede. SPONGENT: the design space of lightweight cryptographic hashing. *IEEE Trans. Computers*, 62(10):2041–2053, 2013.
9. Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: an ultra-lightweight block cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.
10. Andrey Bogdanov, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, and Yannick Seurin. Hash functions and RFID tags: Mind the gap. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, volume 5154 of *Lecture Notes in Computer Science*, pages 283–299. Springer, 2008.
11. Paul Bottinelli and Joppe W. Bos. Computational aspects of correlation power analysis. *IACR Cryptology ePrint Archive*, 2015:260, 2015.
12. Joan Boyar and René Peralta. A new combinational logic minimization technique with applications to cryptology. In Paola Festa, editor, *Experimental Algorithms, 9th International Symposium, SEA 2010, Ischia Island, Naples, Italy, May 20-22, 2010. Proceedings*, volume 6049 of *Lecture Notes in Computer Science*, pages 178–189. Springer, 2010.
13. Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, pages 16–29, 2004.

14. D. Canright and Lejla Batina. A very compact "perfectly masked" s-box for AES (corrected). *IACR Cryptology ePrint Archive*, 2009:11, 2009.
15. Claude Carlet, Louis Goubin, Emmanuel Prouff, Michaël Quisquater, and Matthieu Rivain. Higher-order masking schemes for s-boxes. In *Fast Software Encryption - 19th International Workshop, FSE 2012, Washington, DC, USA, March 19-21, 2012. Revised Selected Papers*, pages 366–384, 2012.
16. Claude Carlet, Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. Algebraic decomposition for probing security. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, pages 742–763, 2015.
17. Mickaël Cazorla, Sylvain Gourgeon, Kevin Marquet, and Marine Minier. Survey and benchmark of lightweight block ciphers for MSP430 16-bit microcontroller. *Security and Communication Networks*, 8(18):3564–3579, 2015.
18. Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, pages 398–412, 1999.
19. Jean-Sébastien Coron. Higher order masking of look-up tables. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 441–458, 2014.
20. Jean-Sébastien Coron, Christophe Giraud, Emmanuel Prouff, Soline Renner, Matthieu Rivain, and Praveen Kumar Vadnala. Conversion of security proofs from one leakage model to another: A new issue. In *Constructive Side-Channel Analysis and Secure Design - Third International Workshop, COSADE 2012, Darmstadt, Germany, May 3-4, 2012. Proceedings*, pages 69–81, 2012.
21. Nicolas Courtois, Daniel Hulme, and Theodosis Mourouzis. Solving circuit optimisation problems in cryptography and cryptanalysis. *IACR Cryptology ePrint Archive*, 2011:475, 2011.
22. A. Adam Ding, Cong Chen, and Thomas Eisenbarth. Simpler, faster, and more robust t-test based leakage detection. *IACR Cryptology ePrint Archive*, 2015:1215, 2015.
23. Daniel Dinu, Yann Le Corre, Dmitry Khovratovich, Léo Perrin, Johann Großschädl, and Alex Biryukov. Triathlon of lightweight block ciphers for the internet of things. *NIST Lightweight Cryptography Workshop 2015*, 2015:209, 2015.
24. François Durvaux, François-Xavier Standaert, Nicolas Veyrat-Charvillon, Jean-Baptiste Mairy, and Yves Deville. Efficient selection of time samples for higher-order DPA with projection pursuits. In *Constructive Side-Channel Analysis and Secure Design - 6th International Workshop, COSADE 2015, Berlin, Germany, April 13-14, 2015. Revised Selected Papers*, pages 34–50, 2015.
25. Thomas Eisenbarth, Zheng Gong, Tim Güneysu, Stefan Heyse, Sebastiaan Indesteege, Stéphanie Kerckhof, François Koeune, Tomislav Nad, Thomas Plos, Francesco Regazzoni, François-Xavier Standaert, and Loïc van Oldeneel tot Oldenzeel. Compact implementation and performance evaluation of block ciphers in attiny devices. In Aikaterini Mitrokotsa and Serge Vaudenay, editors, *Progress in Cryptology - AFRICACRYPT 2012 - 5th International Conference on Cryptology in Africa, Ifrance, Morocco, July 10-12, 2012. Proceedings*, volume 7374 of *Lecture Notes in Computer Science*, pages 172–187. Springer, 2012.
26. Louis Goubin and Jacques Patarin. DES and differential power analysis (the "duplication" method). In Çetin Kaya Koç and Christof Paar, editors,

Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, Proceedings, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 1999.

27. Dahmun Goudarzi and Matthieu Rivain. On the multiplicative complexity of boolean functions and bitsliced higher-order masking. *IACR Cryptology ePrint Archive*, 2016:557, 2016.
28. Vincent Grosso, Gaëtan Leurent, François-Xavier Standaert, and Kerem Varici. Ls-designs: Bitslice encryption for efficient masked software implementations. In Carlos Cid and Christian Rechberger, editors, *Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers*, volume 8540 of *Lecture Notes in Computer Science*, pages 18–37. Springer, 2014.
29. Jian Guo, Thomas Peyrin, Axel Poschmann, and Matthew J. B. Robshaw. The LED block cipher. In *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, pages 326–341, 2011.
30. Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.
31. Gilbert Goodwill Joshua Jaffe Gary Kenworthy Jeremy Cooper, Elke DeMulder and Pankaj Rohatgi. Test vector leakage assessment (tvla) methodology in practice.
32. Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, pages 388–397, 1999.
33. Richard J Larsen and Morris L Marx. *An introduction to mathematical statistics and its applications; 5th ed.* Prentice Hall, Boston, MA, 2012.
34. Seiichi Matsuda and Shiho Moriai. Lightweight cryptography for the cloud: Exploit the power of bitslice implementation. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*, pages 408–425. Springer, 2012.
35. Thomas S. Messerges. Securing the AES finalists against power analysis attacks. In Bruce Schneier, editor, *Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings*, volume 1978 of *Lecture Notes in Computer Science*, pages 150–164. Springer, 2000.
36. Elisabeth Oswald, Stefan Mangard, Christoph Herbst, and Stefan Tillich. Practical second-order DPA attacks for masked smart card implementations of block ciphers. In *Topics in Cryptology - CT-RSA 2006, The Cryptographers' Track at the RSA Conference 2006, San Jose, CA, USA, February 13-17, 2006, Proceedings*, pages 192–207, 2006.
37. Konstantinos Papagiannopoulos and Aram Verstegen. Speed and size-optimized implementations of the PRESENT cipher for tiny AVR devices. In Michael Hutter and Jörn-Marc Schmidt, editors, *Radio Frequency Identification - Security and Privacy Issues 9th International Workshop, RFIDsec 2013, Graz, Austria, July 9-11, 2013, Revised Selected Papers*, volume 8262 of *Lecture Notes in Computer Science*, pages 161–175. Springer, 2013.

38. Kostas Papapagiannopoulos. High throughput in slices: The case of present, PRINCE and KATAN64 ciphers. In Nitesh Saxena and Ahmad-Reza Sadeghi, editors, *Radio Frequency Identification: Security and Privacy Issues - 10th International Workshop, RFIDSec 2014, Oxford, UK, July 21-23, 2014, Revised Selected Papers*, volume 8651 of *Lecture Notes in Computer Science*, pages 137–155. Springer, 2014.
39. Axel Poschmann. Lightweight cryptography - cryptographic engineering for a pervasive world. Cryptology ePrint Archive, Report 2009/516, 2009. `http://eprint.iacr.org/`.
40. Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 142–159, 2013.
41. Pablo Rauzy, Sylvain Guilley, and Zakaria Najm. Formally proved security of assembly code against power analysis: A case study on balanced logic. *CoRR*, abs/1506.05285, 2015.
42. Tobias Schneider and Amir Moradi. Leakage assessment methodology - A clear roadmap for side-channel evaluations. In *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, pages 495–513, 2015.
43. Kyoji Shibutani, Takanori Isobe, Harunaga Hiwatari, Atsushi Mitsuda, Toru Akishita, and Taizo Shirai. Piccolo: An ultra-lightweight blockcipher. In *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, pages 342–357, 2011.
44. François-Xavier Standaert, Tal Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, pages 443–461, 2009.
45. Ko Stoffelen. Optimizing s-box implementations for several criteria using SAT solvers. *IACR Cryptology ePrint Archive*, 2016:198, 2016.
46. Keccak team. *Note on side-channel attacks and their countermeasures.*
47. Elena Trichina. Combinational logic design for AES subbyte transformation on masked data. *IACR Cryptology ePrint Archive*, 2003:236, 2003.
48. Nicolas Veyrat-Charvillon and François-Xavier Standaert. Mutual information analysis: How, when and why? In *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, pages 429–443, 2009.
49. Carolyn Whitnall, Elisabeth Oswald, and Luke Mather. An exploration of the kolmogorov-smirnov test as a competitor to mutual information analysis. In *Smart Card Research and Advanced Applications - 10th IFIP WG 8.8/11.2 International Conference, CARDIS 2011, Leuven, Belgium, September 14-16, 2011, Revised Selected Papers*, pages 234–251, 2011.