Low-Latency Hardware Masking of PRINCE

Nicolai Müller, Thorben Moos, and Amir Moradi

Ruhr University Bochum, Horst Görtz Institute for IT Security, Bochum, Germany
`firstname.lastname@rub.de`

Abstract. Efficient implementation of Boolean masking in terms of low latency has evolved into a hot topic due to the necessity of embedding a physically secure and at-the-same-time fast implementation of cryptographic primitives in e.g., the memory encryption of pervasive devices. Instead of fully minimizing the circuit's area and randomness requirements at the cost of latency, the focus has changed into finding optimal tradeoffs between the circuit area and the execution time. The main latency bottleneck in hardware masking lies in the need for registers to stop the propagation of glitches and maintain non-completeness. Usually, an exponentially growing number of shares (hence an extremely large circuit), as well as a high demand for fresh randomness, are the result of avoiding registers in a securely masked hardware implementation of a block cipher. In this paper, we present several first-order secure and low-latency implementations of PRINCE. In particular, we show how to realize the masked variant of round-based PRINCE with only a single register stage per cipher round. We compare the resulting architectures, based on the popular TI and GLM masking scheme based on the area, latency, and randomness requirements and point out that both designs are suited for specific use cases.

1 Introduction

Industry and academia are aware that the integration of cryptographic primitives into embedded devices is inevitable to ensure the confidentiality of processed data. From a mathematical point of view, cryptographic primitives must resist analytical attacks, for instance, linear and differential cryptanalysis. Moreover, from an economic perspective, the area overhead of the integrated cryptographic circuits forms an additional cost factor without providing new features for the user. Nevertheless, marked requirements or a security-aware target group make it impossible to avoid cryptographic protection of sensitive data. Hence, the research field of lightweight cryptography focuses on mathematically secure ciphers that result in a small circuit size when implemented in hardware. Besides cryptanalytical attacks, embedded devices are prone to side-channel analysis (SCA) attacks. The vulnerability results from the fact that embedded devices are handed over to, potentially malicious, owners. Hence, the owner can measure and manipulate the physical properties of his purchased device. In the past, attackers could recover the secret key of several commercial devices with physical

attacks [10,15].
To protect devices from SCA attacks, the integration of Boolean masking to the cryptographic algorithm is a common and well-studied countermeasure. Unfortunately, the realization of a cryptographic algorithm protected with Boolean masking requires significantly more area due to the processing of multiple shares. Hence, researchers focus on the area-efficient integration of masking to algorithms that potentially handle sensitive data. Usually, the generic approach for reducing the area is to minimize the number of processed shares which makes it necessary to integrate additional register stages into the circuit. As a result, minimizing the circuit area comes at the cost of latency.
The steadily increasing demand for embedded real-time applications, especially for fast memory encryption on smartcards and IoT microcontrollers [1] shifts the focus from the reduction of the circuit size to the minimization of latencies. The PRINCE block cipher [6] fills this gap since it is a lightweight cipher designed to perform an encryption or decryption operation within a single cycle. However, its low latency characteristic gets lost if Boolean masking is applied without considering the latency as shown in [20]. Several adaptions of the popular masking schemes show that accepting a higher circuit size reduces the latency significantly. The most prominent examples are given in [7,12]. The protection with this so-called "low-latency masking" is a special challenge since the resulting circuit must ensure low latency and has to be small enough to be practical. Hence, the main task is to find an acceptable tradeoff between latency on one side and area and randomness requirements on the other side.

Our Contribution. In this work, we show the integration of low-latency masking in PRINCE. In contrast to comparable publications [7,20], we focus on achieving as small latencies as possible whereby we allow larger, but realistic, circuit sizes. In particular, we reduce the number of register stages per round to one in order to process one cipher round per clock cycle. Hence, we achieve an up to three times lower number of clock cycles per encryption than competitive designs. We focus on first-order SCA security due to the integration of the provably secure GLM and TI masking schemes. We remark that other works also aim at SCA resistance of unrolled PRINCE but with the integration of hiding and by taking advantage of the high asynchronicity, glitch number, and parallelism of unrolled circuits [18]. While these works may achieve practical resistance against DPA and CPA, the provable security of hiding countermeasures is not given.

2 Preliminaries

2.1 PRINCE

PRINCE [6] is a 64-bit block cipher with a 128-bit secret key k and has been designed for fully-unrolled implementation. The key is composed of two con-

[1] The LPC55S series of IoT microcontrollers by NXP Semiconductors is one example where PRINCE is employed for memory encryption and where SCA resistance is relevant.

catenated 64-bit subkeys k_0 and k_1 and extended to 192 bits by applying the following linear mapping:

$$(k_0||k_1) \rightarrow (k_0||k_0'||k_1) \coloneqq (k_0||(k_0 \ggg 1) \oplus (k_0 \gg 63)||k_1)$$

To be secure against an exhaustive key search, PRINCE follows the FX construction [16] and applies k_0 and k_0' as whitening keys. The remaining subkey k_1 is fed to the actual cipher named PRINCE$_{core}$.

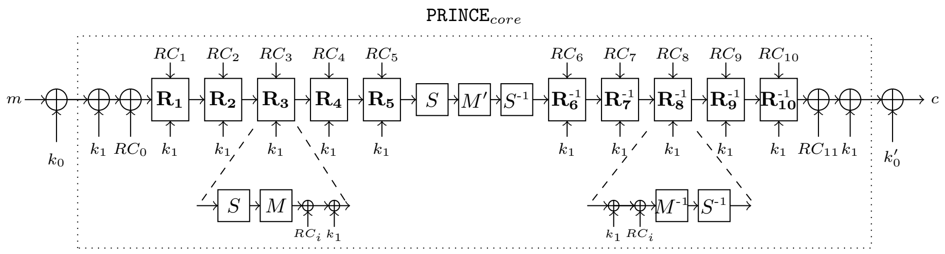

PRINCE$_{core}$
m
k_0
k_1
RC_0
RC_1 RC_2 RC_3 RC_4 RC_5
$\mathbf{R_1}$ $\mathbf{R_2}$ $\mathbf{R_3}$ $\mathbf{R_4}$ $\mathbf{R_5}$
S M' S^{-1}
RC_6 RC_7 RC_8 RC_9 RC_{10}
$\mathbf{R_6^{-1}}$ $\mathbf{R_7^{-1}}$ $\mathbf{R_8^{-1}}$ $\mathbf{R_9^{-1}}$ $\mathbf{R_{10}^{-1}}$
RC_{11}
k_0'
c
S M RC_i k_1
k_1 RC_i M^{-1} S^{-1}

Fig. 1. Schematic description of PRINCE$_{core}$ including the structure of a regular round R_i and an inverse round R_i^{-1}.

Each round performs a *substitution layer* which applies a 4-bit S-Box to each nibble of the state, followed by a linear *M-layer* and the XOR concatenation of k_1 and RC_i (referred to as *constant addition*) to the state. The M-layer is realized as a matrix multiplication of the state with a 64×64 matrix M followed by the *ShiftRows* operation of the AES applied to 4-bit nibbles. Note that the matrix multiplication without ShiftRows builds the *M'-layer* in the middle of PRINCE$_{core}$. After the middle part, all rounds are inverted.
Since M' is an involution and the round constants satisfy $RC_i \oplus RC_{11-i} = \alpha$, the whole PRINCE$_{core}$ can be inverted by changing the underlying key k_1 to $k_1 \oplus \alpha$. If the whitening keys are swapped, PRINCE satisfies :

$$D_{(k_0||k_0'||k_1)}(\cdot) = E_{(k_0'||k_0||k_1 \oplus \alpha)}(\cdot)$$

which is named *α-reflection property.* This feature allows encryption and decryption with the same underlying circuit and consequently saves an independent decryption routine overhead.

2.2 Probing Security

The d-probing model [14] is a widely used and accepted standard attacker model. To specify an attacker's capabilities, the model depends on the *security order* d. An attacker can place up to d different probes on freely chosen spots of the chip. Since a probe measures one spotted wire or gate's power consumption, the attacker can record intermediate signals of any chosen wire or gate. If a design

is d-probing secure, the attacker can not recover any sensitive information by probing d different intermediate values. Formally, this means that every possible d-tuple of intermediate values is independent of every secret value [19]. While this model allows an evaluation of a circuit, in theory, and with simulations, it does not fit for security proofs under real-world conditions. Real hardware is not idealized, so *glitches* can lead to the propagation of unexpected intermediate signals over a wire. Hence, an attacker can read the intermediate value of a probed wire and receive additional information by propagated toggles from previous parts of the circuit. The d-probing model does not reflect the unexpected additional information by propagating glitches, which cause leakage, despite the circuit's security in the d-probing model being proven.
To better match the conditions on real hardware, the d-probing model has been extended to the *glitch-extended d-probing model* [11]. This model allows the attacker to record up to d intermediate values and all possibly propagated values on the probed wires. The attacker can record every intermediate value of the connected combinatorial circuit back to the last register stage. While the glitch-extended d-probing model can simulate an attacker's information gain due to glitches, it does not reflect all other physical properties of the target architecture. Therefore, it is essential to measure each design on the real underlying hardware, even if the security in the glitch-extended d-probing model is proven.

2.3 Threshold Implementation

The concept of threshold implementation (TI) [22] is the oldest provably secure masking scheme which is also resistant to glitches. The core idea is to split any non-linear transformation with algebraic degree t, e.g. the cubic PRINCE Sbox $S(a, b, c, d) = (w, x, y, z)$ with $t = 3$, into a set of at least $t + 1$ component functions $S_{i \in 0,\dots,t}$. The set of component functions must satisfy *correctness*. Hence, it must hold that $S(a, b, c, d) = S_0 \oplus S_1 \oplus S_2 \oplus S_3$ to ensure the correct functionality of the Sbox.
To ensure first-order security of the Sbox TI, each Sbox input, e.g. the MSB a, is given as a set of four *shares* $(a_{i \in 0,\dots,t})$. Moreover, it holds that $a = a_0 \oplus a_1 \oplus a_2 \oplus a_3$. Each S_i can process a subset of shares that is independent of at least one share per variable. We refer to this security-critical condition as *first-order non-completeness*. Besides non-completeness, the security of the Sbox is only given if *uniformity* holds for all input sharings. Since the Sbox output builds the input of following Sboxes, the Sbox output sharing must also satisfy uniformity. Hence, we must ensure that each valid sbox output sharing occurs equally likely for all possible input sharings.
All linear functions, e.g. the PRINCE M-layer, are computed share-wise. Hence, all component functions compute the M-layer on one share independently. The area overhead of TI depends on the number of applied shares which is bounded by the algebraic degree. Since the Sbox consists of cubic coordinate functions, at least four shares are required to achieve non-completeness. To reduce t, one approach is to decompose the cubic functions into multiple chained quadratic functions ($t = 2$) and then separate them with register stages. While several

works show a significant improvement in the area due to *quadratic decomposition*, we avoid it because of two reasons:

1. It is shown in [20] that the decomposition of the cubic PRINCE Sbox into two consecutive quadratic functions is not possible. Hence, the decomposition of the PRINCE Sbox results in at least three quadratic functions.
2. The separation of the quadratic functions is necessary to stop the propagation of glitches. As a result, each subfunction requires an additional register stage which increases the latency by one clock cycle per subfunction. For the PRINCE Sbox, this means that the evaluation of a decomposed Sbox requires three clock cycles. The increment of the cycle count violates our goal of one clock cycle per round.

As a tradeoff, we accept the higher number of shares as long as the circuit size does not explode.

2.4 $d + 1$-Masking

In contrast to TI, another scheme, first published in [23] achieves the same security level as TI but with a fixed number of $d + 1$ input shares. Hence, the number of input shares no longer depends on the algebraic degree but only on the security order. This is achieved since the non-completeness also holds if each component function receives only d shares per variable. Hence, a set of component functions that achieves correctness and non-completeness with $(d+1)$ input shares must contain at least $(d + 1)^t$ component functions. Since the number of output shares grows exponentially with the algebraic degree, only the $d + 1$-masking of low degree functions is more efficient in area than TI. For the cubic PRINCE Sbox, eight component functions per coordinate function are necessary to achieve first-order security. The output sharing is usually not uniform. Hence, fresh randomness is added to the output shares to make them uniform. Right after that, registers store the remasked shares. As the last step, the eight shares are compressed into two shares by XORing two quadruples of shares.
Similar to quadratic decomposition, the circuit size and the amount of fresh randomness is decreased if the cubic Sbox is split into quadratic functions with a register stage and share compression at the output. As for the TI, we omit the decomposition to execute one round per clock cycle.
The concept of $d + 1$-masking builds the foundation for advanced masking schemes such as *domain oriented masking (DOM)* [13]. In particular, the *generic low latency masking (GLM)* [12] offers an interesting approach for low-latency masking. For the GLM scheme, the authors implement no share compression. Hence it allows the probing-secure evaluation of every Boolean function with a combinatorial circuit. While GLM is provably secure, the evaluation of high-degree functions is not practical due to the exponentially growing number of output shares. Nevertheless, GLM allows the designer to adjust every function in terms of area and latency.
We remark that skipping the share compression does not necessarily mean that

the output shares can be fed to the following modules without caution. If the shares are not compressed, the combination during the next modules can violate the non-completeness, although the non-completeness holds for each module independently. This is the case if one variable is part of two inputs of a non-linear gate. The authors of [12] refer to these flaws as *collisions*. To ensure that two inputs are independent, we must design the circuit in a way that the same intermediate value is not fed twice to a non-linear gate or that an intermediate value gets a reshared copy before the intermediate and copy are fed to a non-linear gate.

3 Low-Latency TI Architecture

The intended implementation strategy of PRINCE is an unrolled architecture. Hence, it consists of combinatorial logic only. As a first thought experiment, the whole combinatorial circuit can be shared with GLM without any register stage. As PRINCE encompasses 12 consecutive and cubic substitution layers, the overall algebraic degree for the whole encryption function is $t = 3^{12} = 531441$. Clearly, the number of $s_{out} = 2^{531441}$ output shares is not realizable. As a consequence, it is not possible with currently known methods to achieve single-cycle encryption and provable first-order security together.
Thus, we focus on the evaluation of one round per clock cycle. It is necessary to transform the unrolled design into a round-based architecture with one register stage per round. To the best of our knowledge, four round-based architectures of PRINCE exist in literature [6, 7, 20, 24]. Following the arguments of [20], we choose their design for the following TI experiments. We visualize the unprotected circuit in Figure 2. The authors present a design requiring only a single substitution layer and register stage if no decomposition is applied. Moreover, the authors eliminate the inverse substitution layer, so either the S-Box or its inverse is part of the circuit. Since S and S^{-1} are affine equivalent, there is a tuple of functions consisting of an affine input and output transformation. Applying affine transformations to the input and the substitution layer's output allows computing the inverse substitution layer from the regular substitution layer. In more detail, it holds that $S^{-1} = A \circ S \circ A$ with $A : 5764FDCE1320B98A$ so input and output transformation are equal. In an unmasked fashion, one complete encryption requires 12 clock cycles, e.g. one clock cycle per round.

3.1 TI Sharing of the Sbox

The Sbox of PRINCE, referred to as $S \in \mathbb{F}_2^4 \rightarrow \mathbb{F}_2^4$, consists of four coordinate functions $f_0, f_1, f_2, f_3 \in \mathbb{F}_2^4 \rightarrow \mathbb{F}_2$ which are all cubic. As discussed before, a first-order non-complete TI of S requires at least four shares per variable (for input and output sharing). While finding a correct and non-complete set of component functions is relatively easy, the uniformity is challenging. We apply the TI finder tools of Nikova et al. [5] to find a uniform sharing of S algorithmically. We find

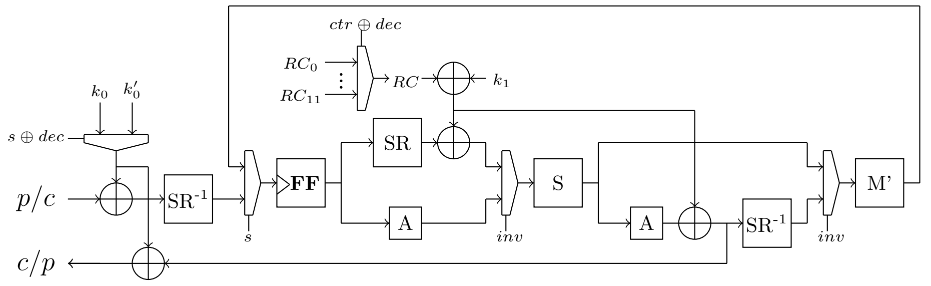


Fig. 2. The optimized round-based architecture of PRINCE with only one substitution layer and register stage.

a tuple of four-bit permutations (S', A) with $S = S' \circ A$ that builds a first-order secure TI of S if the sharing of all input and output variables encompass five shares per variable. Hence, a TI of (S', A) consists of five component functions. The corresponding lookup-tables of S' and A are given in the following:

$$S' = BF32AC918067D4E5 \quad A = 01234567AB89EFCD$$

Since A is an affine transformation on four-bit nibbles, its generated component functions compute A on all shared nibbles separately. The unmasked affine transformation $A(a, b, c, d) = (w, x, y, z) \in \mathbb{F}_2^4 \to \mathbb{F}_2^4$ is computed as:

$$w = a \quad x = b \quad y = a \oplus c \quad z = d$$

According to [5], it holds that S' is uniform if we create a non-complete sharing of S' with the concept of *direct sharing* [4]. Since the correctness is satisfied due to $S = S' \circ A$, the resulting TI is first-order secure. The full circuit, processing five shares, is given in Figure 3.

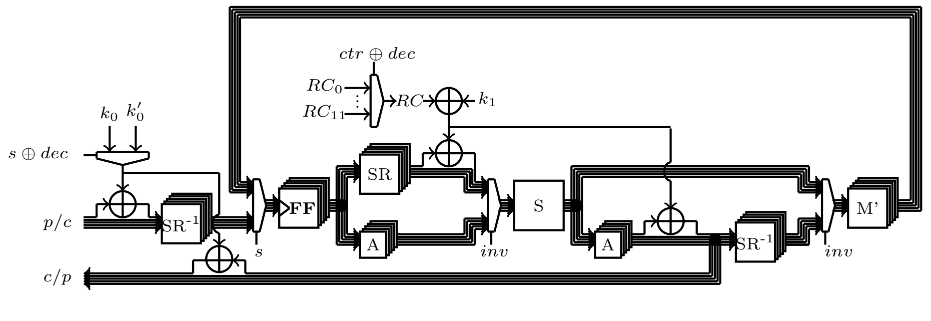


Fig. 3. Five-share uniform PRINCE TI architecture without any fresh randomness.

However a sharing with five input and output shares increases the circuit size significantly compared to a non-complete sharing with four input and out-

put shares. To keep the circuit size as small as possible, we decide to reduce the number of input shares to four and generate a non-complete set of component functions by following direct sharing. Compared to the five-share TI, we avoid one additional component function and hence about 20% of area. We remark that this change does not influence the number of required clock cycles. Regarding the critical path, we observe that the reduction of shares leads to component functions that combine fewer monomials, so the number of consecutive XOR gates is reduced in all component functions. Moreover, we implement all component functions in a way that they are almost balanced. Hence, the critical path is almost equal-sized for all component functions resulting in a shorter critical path and higher possible clock frequencies.
Obviously, the resulting component functions S_0, S_1, S_2, S_3 are first-order non-complete but not uniform anymore. We satisfy the uniformity of the output sharing manually by adding initial randomness according to the *changing of the guards* method [9] (referred in the Figure 4 as R). This technique guarantees uniformity at the price of a small overhead of 12 random initial bits per encryption. In particular, we require no fresh-randomness per clock cycle. This small amount of initial randomness is acceptable to reduce the number of shares by one as the five-share variant would require more area *and* a larger amount of initial randomness (due to the initial sharing into five shares instead of four).
After the remasking, all shares and the *guards* are stored in registers. All following operations, especially the M-layer, are linear and performed on each share separately. We show the resulting architecture of the full PRINCE cipher in Figure 4.

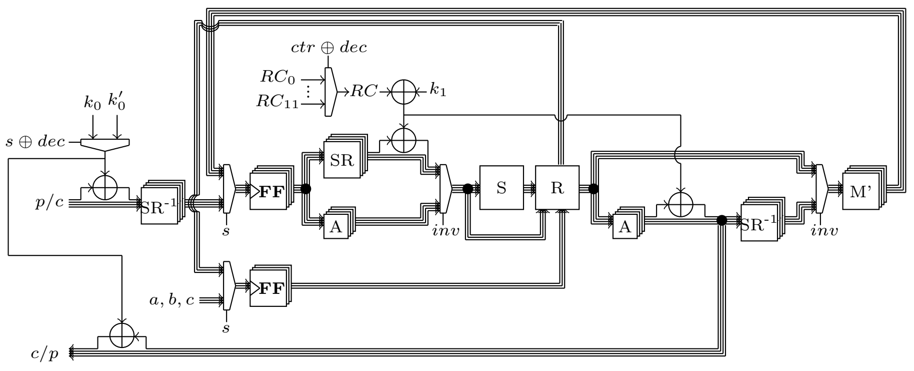


Fig. 4. Four-share uniform PRINCE TI architecture without fresh randomness.

4 Low-Latency GLM Architecture

In contrast to the TI scheme, the application of GLM is prone to the introduction of collisions during the linear layers. Hence, the underlying architecture

must satisfy that the non-completeness is not violated on a path through multiple layers. Unfortunately, the previous architecture contains a collision introduced during the affine layer A and leaking during S. It holds that the outputs of $A(a,b,c,d) = (w,x,y,z)$ are not independent of each other (both w and y depend on a). Therefore, a non-linear gate with the inputs w and y violates the non-completeness.
The remaining linear layers, in particular M', are collision-free. Each output bit of M' is a linear combination of three different input bits which are forwarded from different Sboxes. Moreover, M' keeps the structure of all four-bit Sbox outputs. Each output bit of M' inside a four-bit nibble $m_{i\in 0,\ldots,3}$ is a combination of three different Sbox output bits $s^0_{i\in 0,\ldots,3}$, $s^1_{i\in 0,\ldots,3}$, $s^2_{i\in 0,\ldots,3}$ from the same position inside its computing Sbox. The resulting four-bit output nibbles encompass four bits which are all independent of each other. Hence, the output of M' can be fed as an independent input to the following Sboxes. The shifting layers operate on four-bit nibbles independently, so the structure of the M' is not changed.
The investigations above point out that we can not apply the same underlying architecture as for the TI experiments. It is inevitable to remove the affine transformations and to implement an additional circuit for S^{-1} instead. The resulting architecture is mainly influenced by the architecture applied in [24]. We only remove the second register stage and we changed the position of the SR^{-1} layer so that SR and SR^{-1} operate in parallel. Since SR^{-1} is only wiring, the additional insertions of SR^{-1} on the output path and the key path do not increase the circuit size. On the other hand, the parallel processing of SR and SR^{-1} eliminates the additional data path that wires the unshifted state to the Sboxes. This is helpful if we consider the propagation of glitches. Note that the only register stage per round is placed right behind the substitution layer so that it can synchronize the state after the remasking. We present the resulting design in an unmasked form in Figure 5.

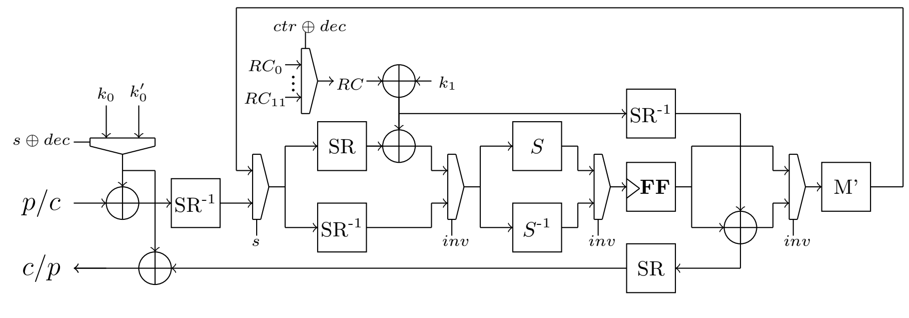


Fig. 5. The underlying target PRINCE architecture for the GLM experiments.

4.1 GLM Sharing of the Sbox

According to $d+1$ masking, we share the first-order secure Sbox input with two shares $(a_0, b_0, c_0, d_0), (a_1, b_1, c_1, d_1)$. Since all coordinate functions of S and S^{-1} are cubic, eight component functions per coordinate function are required to satisfy first-order non-completeness. We apply the sharing given in [24] to minimize the extra circuit due to an additional S^{-1} module. Hence, all component functions compute one share per coordinate function for S and S^{-1}, in total, eight output values per component function. Since we build all component functions based on the algebraic normal form (ANF), no collisions occur within the computations of the coordinate functions. To ensure the independence of the Sbox output we add fresh randomness to the shared values and store them in registers. The synchronized values are then compressed (in module C) from eight shares back to two. As all shares are refreshed, no collisions occur during the share compression. All linear components are instantiated twice to handle two shares separately. We give the full design in Figure 6.

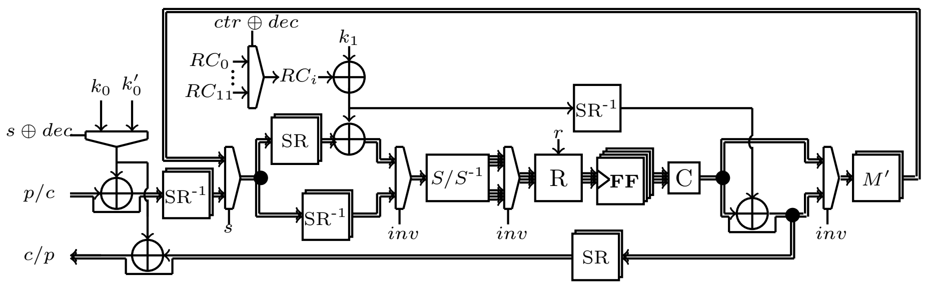


Fig. 6. The masked PRINCE GLM achitecture.

Remasking and Share Compression To ensure the independence of the following round's Sbox input, we apply a remasking to all output shares of the substitution layer. Since we integrate only one register stage, we must investigate glitches propagated through the whole circuit. Naturally, the required amount of randomness grows if the propagation of glitches stops only at a single point of the round function. We add four fresh random bits $r^{i\in 0,\ldots,3}$ to the shares $s_{i\in 0,\ldots,7}$ of each bit according to the following scheme:

$$\begin{array}{llll} g_0 = s_0 \oplus r^0 & g_1 = s_1 \oplus r^1 & g_2 = s_2 \oplus r^2 & g_3 = s_3 \oplus r^3 \\ g_4 = s_4 \oplus r^0 & g_5 = s_5 \oplus r^1 & g_6 = s_6 \oplus r^2 & g_7 = s_7 \oplus r^3 \end{array}$$

The remasking ensures that an attacker gets no information if she probes a combination of output shares computed during the share compression. Since both compressed output shares are never combined, we can apply the same random bits to both quadruples of output shares. After the remasking, all remasked shares $g_{i \in 0,\ldots,7}$ are synchronized in registers and the share compression XOR concatenates four shares with different randomness together. Hence, it compresses eight input shares to two output shares. Formally, the share compression is given as:

$$z_0 = g_0 \oplus g_1 \oplus g_2 \oplus g_3 \qquad z_1 = g_4 \oplus g_5 \oplus g_6 \oplus g_7$$

We remark that the remasking makes all shares independent of each other. Hence, glitches introduced within the share compression do not leak any sensitive information if we separate both additions into different modules.

Reduction of Online Randomness. As described before, we remask one shared output bit with four fresh random bits. In total, the remasking of the whole state leads to an amount of $64 \cdot 4 = 256$ random bits per clock cycle. To reduce the amount of online randomness, we refer to the structural analysis of PRINCE given in [7] and adapt it to a round circuit with a single register stage. In contrast to the architecture specified in [7], we place SR and SR^{-1} in parallel and choose the correct output via multiplexers. Moreover, we place no register stage in front of the substitution layer. Hence, glitches introduced by the multiplexers are propagated to the following substitution layer. According to the glitch-extended d-probing model, an attacker who places a probe on one Sbox input bit receives one output bit of SR and $SR^{\text{-}1}$. To omit information leakage, we remask all bits, wired to the same position, with different random bits. An example of such a probe propagation due to glitches through the combinatorial round circuit is visualized in Figure 7. In this example, the attacker places one probe on the LSB of the substitution layers output which results in various glitch-extended probes on single bits in the second and fourth 16-bit input block. It turns out that all output bits of the substitution layer depend either on the first and the third or on the second and the fourth 16-bit input block. As the values from the first 16-bit block are never combined with values from the second 16-bit block during one round, we can use the same randomness to mask the first and the second block. Analogously, we use the same randomness for blocks three and four. The repeated application of the same randomness halves the number of random bits per clock cycle from 256 to 128.

5 Synthesis Results

To give comparable circuit sizes and critical path lengths of our investigated architectures, we synthesize all presented unprotected (cf. Table 1) and first-order masked (cf. Table 2) designs against a real gate library. Moreover, we implement comparable designs from literature and synthesize them against the same gate library (cf. Table 3). We apply synopsis design compiler to synthesize against Nangate 15 nm standard cell library. The synthesizer generates a gate-level netlist and the corresponding area and timing reports. As our goal is to optimize latency, we force the synthesizer to generate the fastest possible design by setting the clock period to 0.1 picoseconds. Hence, the results can never

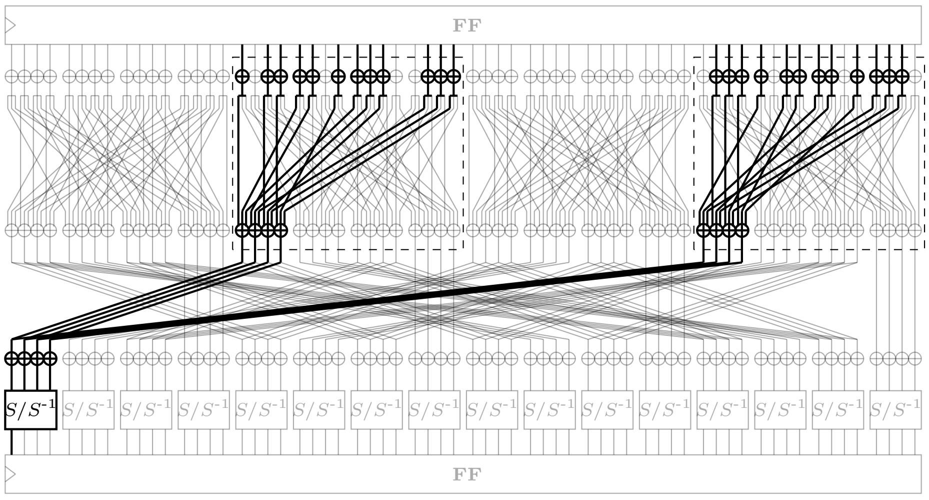


Fig. 7. Probe propagation in the glitch-extended probing model through one round of the unprotected PRINCE GLM architecture by probing the LSB of the round function output. The probed 16-bit input blocks (second and fourth) are bordered by a dashed rectangle.

fulfill the timing requirements and the synthesizer optimizes the design's timing as much as possible. For the area estimation, we only take the cipher core into account. Hence we ignore additional area overheads of the FSM or additional PRNGs. We give an overview of the specific requirements of each design in terms of latency, area, and randomness requirements. Note that the timing estimations are divided into the total number of clock cycles for one encryption and the critical path length. To take all design requirements into account we also show the randomness requirements, divided into initial and online randomness.

Table 1. Synthesis results of all investigated unprotected PRINCE designs.

Scheme	Area	Latency	Delay
	GE	cycles	ps
KECCAK-PRNG [3]	4930	1	80
unrolled (Fig. 1)	16352	1	412
round-based (Fig. 2)	3997	12	118
round-based (Fig. 5)	5045	12	98

We conclude that the latency improvement sourced by the application of one register stage per cipher round comes at the cost of area and randomness overheads. All investigated TI designs are at least 73% larger than the comparable TI architecture from [20] with a three-stage decomposed Sbox especially since

Table 2. Synthesis results of all investigated first-order masked PRINCE designs.

Scheme	#Shares	Area	Initial Rand	Online Rand	Latency	Delay
		GE	bits	bits/cycle	cycles	ps
TI (Fig. 3)	5	42158	256	0	12	153
TI (Fig. 4)	4	26158	204	0	12	145
GLM (Fig. 6)	2	20046	64	128	12	97

Table 3. Synthesis results of comparable first-order masked PRINCE designs from literature.

Scheme	#Shares	Area	Initial Rand	Online Rand	Latency	Delay
		GE	bits	bits/cycle	cycles	ps
TI [20]	3	15063	128	0	40	85
GLM [7]	2	16951	64	112	24	101
GLM [24]	2	14235	64	0	24	85

they operate on more than the minimum number of three shares. Since fewer shares are processed and combined, the circuit size and the critical path delay decrease if the number of shares is decreased. This is beneficial, for TIs with fewer shares as the changing of the guards technique allows a remasking without any online randomness. Due to these results, we recommend using the minimum number of shares and the changing of the guards technique for a low latency TI of PRINCE.

The further reduction to $d+1$ shares makes it necessary to use either online randomness affecting the circuit size in case that additional PRNGs are required or to place another register stage in the round circuit [24] which doubles the latency. Nevertheless, we achieve the shortest critical path and the smallest cipher core area for the GLM architecture. Compared to the similar design in [24] we half the number of clock cycles at the cost of 128 bit online randomness per round and an area overhead of 41%. We sum up, that the GLM architecture is the most suited design regarding low-latency but with the restriction that a small and fast PRNG is available. We apply a PRNG based on KECCAK-f[200] [3] with parameters $r = 128$ and $c = 72$ to generate 128 bits of randomness per clock cycle. Otherwise, the TI architecture is usable in absence of a PRNG. The area requirement for the first-order masking itself grows by at least a factor of four for the GLM architecture. Nevertheless, our smallest first-order secure low-latency architecture is only 23% larger than the originally proposed unrolled architecture. We, therefore, conclude that the area requirements are still acceptable.

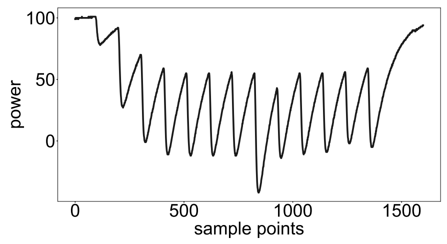
6 Security Analysis

We evaluate our designs' power-side-channel leakage based on physical measurements on an FPGA. We place both architectures on a SAKURA-G board [1], specifically designed to evaluate power-analysis attacks. The board integrates two Spartan-6 FPGAs. The smaller FPGA is a controller that communicates with the target FPGA and the measurement script. Moreover, a PRNG based on the AES in counter mode is placed on the control FPGA to generate masks and additional online randomness. We measure the power consumption at a shunt resistor inserted in the target FPGA's V_{dd} path and record the traces with a digital PicoScope oscilloscope of the 6000 series. The oscilloscope operates on a sampling rate of 625 MS/s while all target designs receive a 6 MHz clock. The recorded power trace is quantized with an 8-bit resolution and stored on the host PC. A sample trace of the five-share TI architecture (cf. Figure 8(a)), the four-share TI architecture (cf. Figure 9(a)), and the GLM architecture (cf. Figure 10(a)) is illustrated. We apply the non-specific t-test over 100 million traces, measured either by encrypting a fixed or a random plaintext. The t-test detects general information leakage by comparing the statistical properties of two groups of traces. We say that an implementation has general information leakage if an absolute t-value in a single sample point exceeds the 4.5 absolute threshold. We compute the t-values for the first and second statistical moment. The resulting plots are given in Figures 8, 9, and 10.

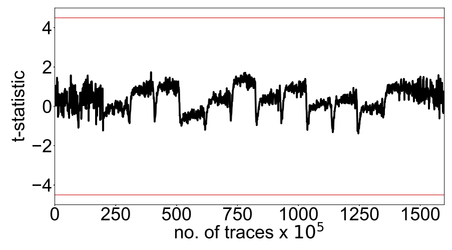
Figure 8(b) indicates that the evaluated PRINCE TI architecture with five shares is first-order secure since no absolute t-value surpasses the 4.5 threshold. Moreover, Figure 8(c) shows no growing progression of the maximum t-value. As expected, we observe strong second-order leakage after hundred thousands of traces (cf. Figures 8(d) and 8(e)).

The same holds for the evaluated TI architecture with four-shares and remasking. Figure 9(b) shows no first-order leakage indicating that the design is first-order secure. Figure 9(c) shows no growing progression of the maximum t-value and we observe strong second-order leakage after hundred thousands of traces (cf. Figures 9(d) and 9(e)). Compared to the five-share TI, the maximum second-order leakage is higher than for the four-share TI architecture and the threshold is exceeded in a larger set of sample points.

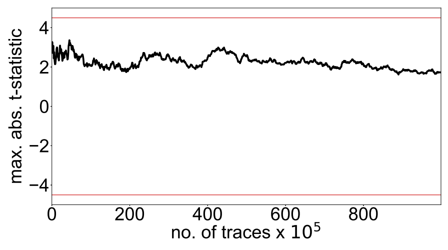
For the GLM architecture, the t-test results do not indicate perfect first-order security. Figure 10(c) shows that the maximum t-statistics value exceeds the threshold after around 50 million traces. However, several previous works have observed that the t-test may indicate the presence of detectable leakage in the first order due to coupling effects, despite the implementation being glitch-extended first-order probing secure [8]. This seems to be especially relevant for $d + 1$-masked implementations with $d = 1$. In particular, the authors of [2] observe first-order leakage when evaluating a first-order secure DOM protected implementation of the AES with the non-specific t-test. Since the GLM architecture utilizes a $d + 1$ masking scheme and is implemented using 2 shares, it is not overly surprising that the non-specific t-test indicates small amounts of detectable first-order leakage. However, to verify that the leakage stems from coupling effects and not from an implementation flaw we have used SILVER [17] to verify the implementation's glitch-extended probing security.
Additionally, we decided to evaluate the design with a moments-correlating DPA



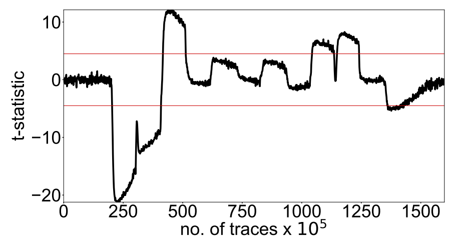
(a) A sample trace of the five-share PRINCE TI



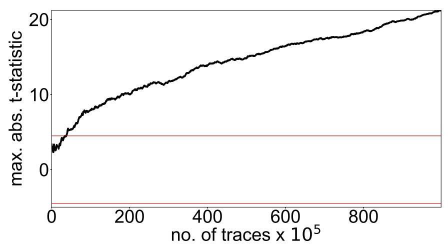
(b) First-order t-test over points



(c) First-order t-test over traces



(d) Second-order t-test over points

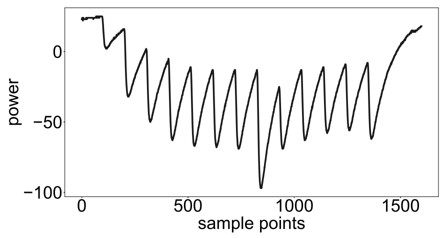


(e) Second-order t-test over traces

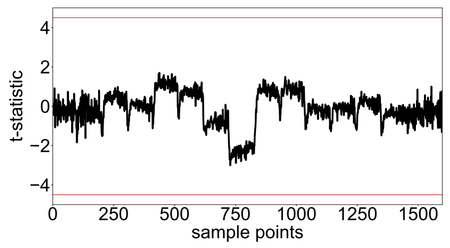
Fig. 8. Non-specific t-test results over 100 million traces encompassing the whole encryption with the PRINCE five-share TI architecture.

(MCDPA) [21]. Since the MCDPA is a collision-based attack we avoid restricting our evaluation to a specific power model. During the profiling phase, we create first and second-order models based on a set of 50 million traces for each Sbox. Another set of 50 million traces is used to perform the actual attack on each Sbox. As a result, the attack returns the correlation between the model and attack traces for all possible input differences. Since we correlate the same Sbox from the modeling set and the attack set, we expect the highest correlation for the input difference zero.

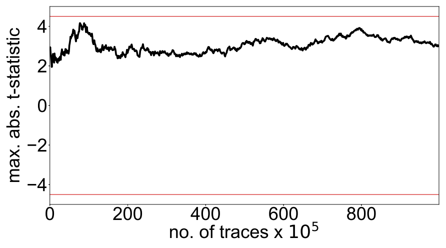
The first-order MCDPA results for each nibble are shown in Figure 11. As the correct difference (colored black) shows no higher correlation than all other differences, we again confirm that our implementation is indeed secure against first-order attacks. The second-order MCDPA results are given in Figure 12. As



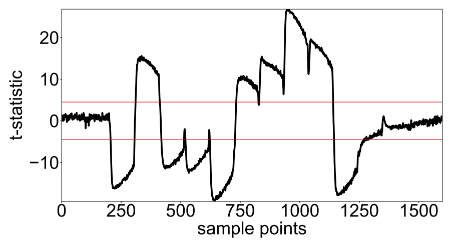
(a) A sample trace of the four-share PRINCE TI



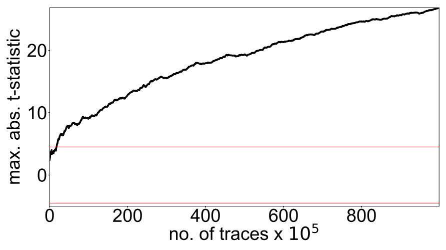
(b) First-order t-test over points



(c) First-order t-test over traces



(d) Second-order t-test over points



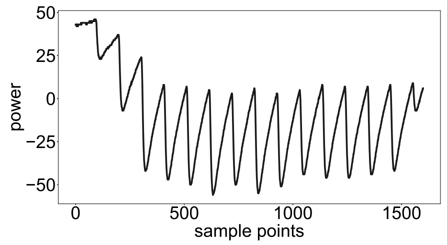
(e) Second-order t-test over traces

Fig. 9. Non-specific t-test results over 100 million traces encompassing the whole encryption with the PRINCE four-share TI architecture.

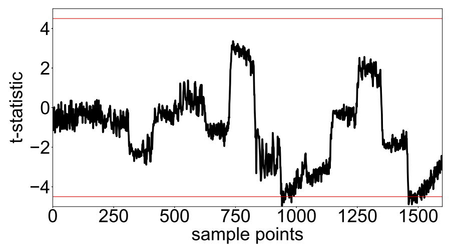
expected, we observe that the attack successfully recovered the key difference in the second order. In particular, the correct key difference is distinguishable from all other differences for all nibbles.

7 Conclusion

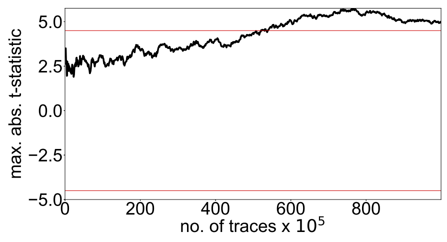
In this paper, we present several case studies on low-latency masked PRINCE architectures with the TI and GLM masking scheme. The comparison of all presented designs in Table 2 points out that every scheme comes with its characteristic requirements for area, latency, and randomness. The analyzed TI architectures offer an acceptable circuit size and require no fresh randomness. Moreover, we point out that the usage of remasking with changing of the guards is prefer-



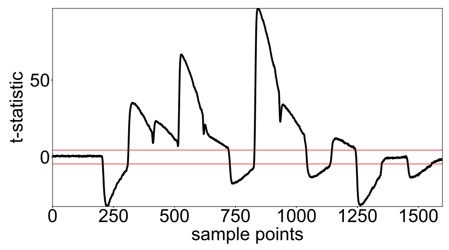
(a) A sample trace of PRINCE GLM



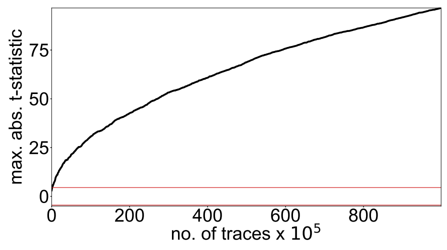
(b) First-order t-test over points



(c) First-order t-test over traces



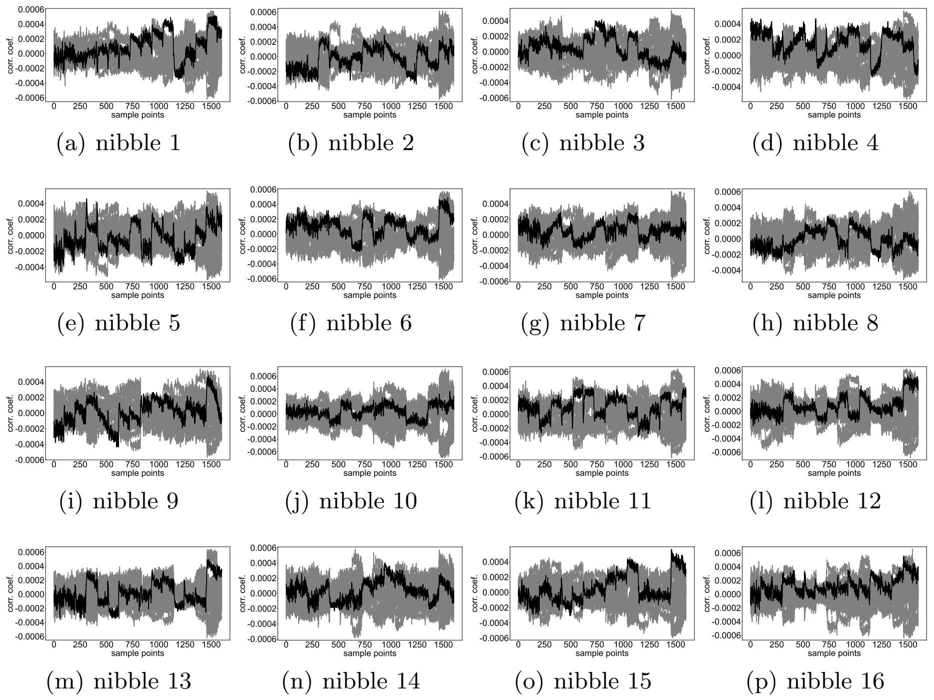
(d) Second-order t-test over points



(e) Second-order t-test over traces

Fig. 10. Non-specific t-test results over 100 million traces encompassing the whole encryption with the PRINCE GLM architecture.

able compared to the usage of more shares. The reduction of shares to $d + 1$ improves the circuit size significantly at the cost of online randomness. We also demonstrate all latency improvements result in at least an overhead in the area due to the processing of more shares. Hence, the lightweight property of every block cipher is decreased by applying low-latency masking to it. Nevertheless, all overheads are acceptable and applicable in practical devices.



(a) nibble 1 (b) nibble 2 (c) nibble 3 (d) nibble 4

(e) nibble 5 (f) nibble 6 (g) nibble 7 (h) nibble 8

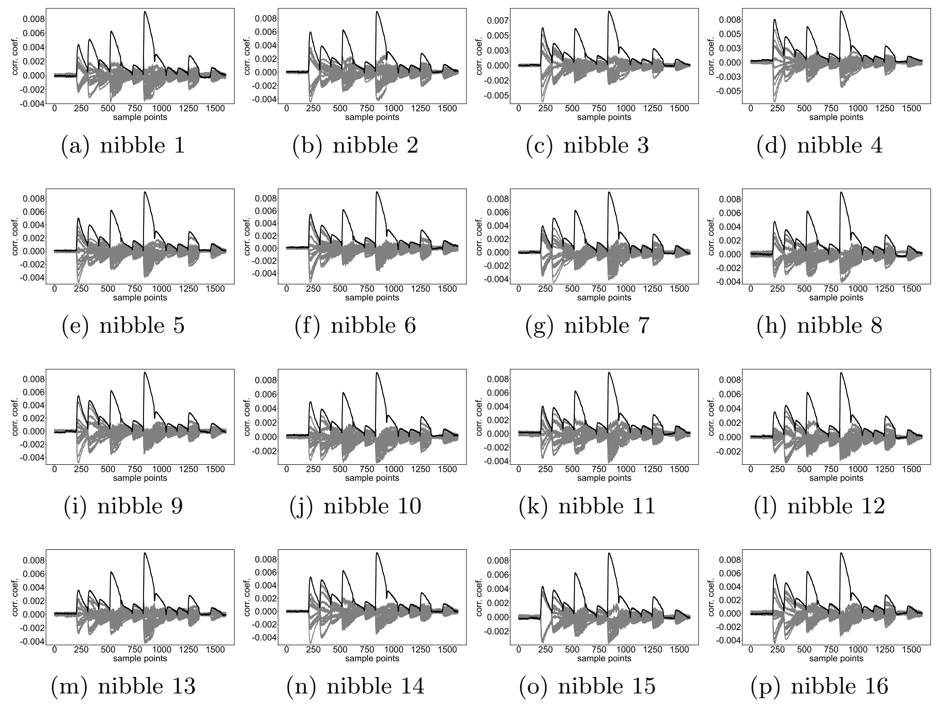
(i) nibble 9 (j) nibble 10 (k) nibble 11 (l) nibble 12

(m) nibble 13 (n) nibble 14 (o) nibble 15 (p) nibble 16

Fig. 11. 1st-order profiled MCDPA targeting the PRINCE GLM architecture using 50 million profiling traces and 50 million attack traces.

Acknowledgments

The work described in this paper has been supported in part by the German Research Foundation (DFG) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972, and through the project 435264177 'phySicAlly secUre reconfiguraBlE platfoRm (SAUBER)'.



(a) nibble 1 (b) nibble 2 (c) nibble 3 (d) nibble 4

(e) nibble 5 (f) nibble 6 (g) nibble 7 (h) nibble 8

(i) nibble 9 (j) nibble 10 (k) nibble 11 (l) nibble 12

(m) nibble 13 (n) nibble 14 (o) nibble 15 (p) nibble 16

Fig. 12. 2nd-order profiled MCDPA targeting the PRINCE GLM architecture using 50 million profiling traces and 50 million attack traces.

References

1. Side-channel attack user reference architecture. `http://satoh.cs.uec.ac.jp/SAKURA/index.html`
2. Bache, F., Plump, C., Güneysu, T.: Confident leakage assessment — a side-channel evaluation framework based on confidence intervals. In: 2018 Design, Automation Test in Europe Conference Exhibition (DATE). pp. 1117–1122 (2018). https://doi.org/10.23919/DATE.2018.8342178
3. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Sponge-based pseudo-random number generators. In: Mangard, S., Standaert, F.X. (eds.) Cryptographic Hardware and Embedded Systems, CHES 2010. pp. 33–47. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
4. Bilgin, B., Nikova, S., Nikov, V., Rijmen, V., Tokareva, N., Idrisova, V.: Threshold implementations of small s-boxes. Cryptography and Communications **7** (03 2015). https://doi.org/10.1007/s12095-014-0104-7
5. Bilgin, B., Nikova, S., Nikov, V., Rijmen, V., Stütz, G.: Threshold implementations of all 3x3 and 4x4 s-boxes. In: Prouff, E., Schaumont, P. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2012. pp. 76–91. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
6. Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knezevic, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., Rombouts, P., Thom-

sen, S.S., Yalçin, T.: PRINCE - A low-latency block cipher for pervasive computing applications - extended abstract. In: Wang, X., Sako, K. (eds.) Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7658, pp. 208–225. Springer (2012). https://doi.org/10.1007/978-3-642-34961-4_14, https://doi.org/10.1007/978-3-642-34961-4_14

7. Božilov, D., Knežević, M., Nikov, V.: Optimized threshold implementations: Minimizing the latency of secure cryptographic accelerators. In: Belaïd, S., Güneysu, T. (eds.) Smart Card Research and Advanced Applications. pp. 20–39. Springer International Publishing, Cham (2020)
8. Cnudde, T., Ender, M., Moradi, A.: Hardware masking, revisited. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2018**, 123–148 (2018)
9. Daemen, J.: Changing of the guards: a simple and efficient method for achieving uniformity in threshold sharing. Cryptology ePrint Archive, Report 2016/1061 (2016), https://eprint.iacr.org/2016/1061
10. Eisenbarth, T., Kasper, T., Moradi, A., Paar, C., Salmasizadeh, M., Shalmani, M.T.M.: On the power of power analysis in the real world: A complete break of the keeloq code hopping scheme. In: Wagner, D. (ed.) Advances in Cryptology – CRYPTO 2008. pp. 203–220. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
11. Faust, S., Grosso, V., Merino Del Pozo, S., Paglialonga, C., Standaert, F.X.: Composable masking schemes in the presence of physical defaults &; the robust probing model. IACR Transactions on Cryptographic Hardware and Embedded Systems **2018**(3), 89–120 (Aug 2018). https://doi.org/10.13154/tches.v2018.i3.89-120, https://tches.iacr.org/index.php/TCHES/article/view/7270
12. Groß, H., Iusupov, R., Bloem, R.: Generic low-latency masking in hardware. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2018**(2), 1–21 (2018). https://doi.org/10.13154/tches.v2018.i2.1-21, https://doi.org/10.13154/tches.v2018.i2.1-21
13. Gross, H., Mangard, S., Korak, T.: Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order. In: Proceedings of the 2016 ACM Workshop on Theory of Implementation Security. p. 3. TIS '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2996366.2996426, https://doi.org/10.1145/2996366.2996426
14. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: Securing hardware against probing attacks. In: Boneh, D. (ed.) Advances in Cryptology - CRYPTO 2003. pp. 463–481. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)
15. Kasper, M., Kasper, T., Moradi, A., Paar, C.: Breaking keeloq in a flash: On extracting keys at lightning speed. In: Preneel, B. (ed.) Progress in Cryptology – AFRICACRYPT 2009. pp. 403–420. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
16. Kilian, J., Rogaway, P.: How to protect des against exhaustive key search. In: Koblitz, N. (ed.) Advances in Cryptology — CRYPTO '96. pp. 252–267. Springer Berlin Heidelberg, Berlin, Heidelberg (1996)
17. Knichel, D., Sasdrich, P., Moradi, A.: Silver – statistical independence and leakage verification. In: Moriai, S., Wang, H. (eds.) Advances in Cryptology – ASIACRYPT 2020. pp. 787–816. Springer International Publishing, Cham (2020)
18. Moos, T.: Unrolled cryptography on silicon: A physical security analysis. IACR Transactions on Cryptographic Hardware and Embedded Systems **2020**(4), 416–

442 (Aug 2020). https://doi.org/10.13154/tches.v2020.i4.416-442, https://tches.iacr.org/index.php/TCHES/article/view/8689

19. Moos, T., Moradi, A., Schneider, T., Standaert, F.X.: Glitch-resistant masking revisited: or why proofs in the robust probing model are needed. IACR Transactions on Cryptographic Hardware and Embedded Systems **2019**(2), 256–292 (Feb 2019). https://doi.org/10.13154/tches.v2019.i2.256-292, https://tches.iacr.org/index.php/TCHES/article/view/7392
20. Moradi, A., Schneider, T.: Side-channel analysis protection and low-latency in action - case study of prince and midori. Cryptology ePrint Archive, Report 2016/481 (2016), https://eprint.iacr.org/2016/481
21. Moradi, A., Standaert, F.X.: Moments-correlating dpa. In: Proceedings of the 2016 ACM Workshop on Theory of Implementation Security. p. 5–15. TIS '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2996366.2996369, https://doi.org/10.1145/2996366.2996369
22. Nikova, S., Rechberger, C., Rijmen, V.: Threshold implementations against side-channel attacks and glitches. In: Ning, P., Qing, S., Li, N. (eds.) Information and Communications Security. pp. 529–545. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)
23. Reparaz, O., Bilgin, B., Nikova, S., Gierlichs, B., Verbauwhede, I.: Consolidating masking schemes. IACR Cryptology ePrint Archive **2015**, 719 (2015)
24. Shahmirzadi, A.R., Moradi, A.: Re-consolidating first-order masking schemes - nullifying fresh randomness. Cryptology ePrint Archive, Report 2020/890 (2020), https://eprint.iacr.org/2020/890