Analyzing the Leakage Resistance of the NIST's Lightweight Crypto Competition's Finalists

Corentin Verhamme[1], Gaëtan Cassiers[1,2,3], and François-Xavier Standaert[1]

[1] ICTEAM, Université catholique de Louvain, Louvain-la-Neuve, Belgium
[2] TU Graz, Graz, Austria
[3] Lamarr Security Research, Graz, Austria

Abstract. We investigate the security of the NIST Lightweight Crypto Competition's Finalists against side-channel attacks. We start with a mode-level analysis that allows us to put forward three candidates (Ascon, ISAP and Romulus-T) that stand out for their leakage properties and do not require a uniform protection of all their computations thanks to (expensive) implementation-level countermeasures. We then implement these finalists and evaluate their respective performances. Our results confirm the interest of so-called leveled implementations (where only the key derivation and tag generation require security against differential power analysis). They also suggest that these algorithms differ more by their qualitative features (e.g., two-pass designs to improve confidentiality with decryption leakage vs. one-pass designs, flexible overheads thanks to masking vs. fully mode-level, easier to implement, schemes) than by their quantitative features, which all improve over the AES and are quite sensitive to security margins against cryptanalysis.

1 Introduction

Security against side-channel attacks is explicitly mentioned by the NIST as a target in the ongoing standardization process for lightweight cryptography.[4] In this paper, we analyze the leakage resistance of 9 out of the 10 finalists of the competition. Our contributions in this respect are twofold.

First, we use a framework introduced by Bellizia et al. to evaluate the high-level leakage properties of the candidates' modes of operations [BBC+20]. (We exclude Grain-128AEAD from our study, which cannot be captured with such a mode vs. primitive granularity.) This high-level analysis allows us to observe that 6 candidates can mostly rely on (expensive) implementation-level countermeasures. By contrast, 3 candidates (namely Ascon [DEMS21], ISAP [DEM+20] and Romulus-T[5]) have leakage-resistant features enabling so-called leveled implementations, where different parts of the implementations require different (more or less expensive) implementation-level countermeasures.

[4] `https://csrc.nist.gov/Projects/lightweight-cryptography`.

[5] `https://romulusae.github.io/romulus/`. Note that Romulus comes with different modes of operation. In particular, the (single-pass) N version does not provide mode-level leakage-resistance guarantees while the (two-pass) T version does.

Second, we investigate the hardware performances of these 3 leakage-resistant modes of operation and evaluate their leveled implementation. In leveled implementations, we distinguish between Differential Power Analysis (DPA), where the adversary is able to collect an adversarially chosen number of measurements corresponding to fixed secret inputs to the target primitive, and Simple Power Analysis (SPA), where the number of such traces is small and bounded by design. The goal of mode-level protections is to minimize the amount of computations that must be protected against DPA and SPA. For Ascon and Romulus-T, we protect the Key Derivation Function (KDF) and Tag Generation Function (TGF) against DPA with Hardware Private Circuits (HPC), a state-of-the-art masking scheme that jointly provides resistance against physical defaults and composability [CGLS21, CS21]. For ISAP, the KDF and TGF are based on a leakage-resilient PRF that embeds a fresh re-keying mechanism such that they only require security against SPA [MSGR10,BSH+14]. The latter is natively (and efficiently) obtained thanks to parallelism in hardware. For all 3 candidates, the bulk of the computation contains an internal re-keying mechanism. Hence, guarantees of confidentiality with leakage essentially require its SPA security (again achieved with hardware parallelism). This part of the implementation can even leak in an unbounded manner if only integrity with leakage is required.

The hardware design space of Ascon (and ISAP, that relies on the same permutation) has already been quite investigated in the literature, both regarding unprotected and masked implementations [GWDE15, GWDE17]. Our implementations heavily build on this state-of-the-art. By contrast, to the best of our knowledge such evaluations are a bit sparser for Romulus-T [Kha22] and the Skinny block cipher it relies on [BJK+20], especially for higher-order masked implementations. Therefore, and as an additional technical contribution, we complete the study of masked Skinny implementations tailored for masking.

We conclude that more than the quantitative comparison of the finalists, the main criteria that should help the NIST in selecting a lightweight cryptography standard (if leakage is deemed important) are qualitative. The limited relevance of quantitative comparisons at this stage of the competition follows from two facts. For ciphers that rely on comparable countermeasures for their DPA security (like Ascon and Romulus-T, both leveraging masking), the performance gap is limited and quite sensitive to security margins against cryptanalysis. Both are nevertheless significantly easier to protect against leakage than the AES, as witnessed by simple proxies such as their number of AND gates or AND depth. For ciphers that rely on different countermeasures for their DPA security (like ISAP, that leverages re-keying), we currently lack (both theoretical and practical) tools that would allow a definitive comparison (e.g., with masking). By contrast, these three ciphers have different qualitative features, leading to at least two questions that could (and we think, should) guide the final selection:

– *Is confidentiality with decryption leakage wanted?* Ascon, ISAP and Romulus-T all reach the top of the hierarchy in [GPPS19] for integrity with leakage (coined CIML2). The leveled implementation of Ascon only provides confidentiality with encryption leakages and misuse-resilience (coined CCAmL1):

decryption queries of a ciphertext leak the underlying plaintext via a straightforward DPA. The leveled implementations of ISAP and Romulus-T can additionally provide confidentiality with decryption leakages and misuse-resilience (coined CCAmL2) at the cost of being two-pass for decryption (they are only CCAmL1 if a single-pass decryption is performed).
– *Flexibility or simplicity for the KDF and TGF?* Ascon and Romulus-T require DPA countermeasures like masking to protect their KDF and TGF. Implementing masking securely is a sensitive process that requires expertise [MPG05, CGP+12]. But it comes with a lot of flexibility: countermeasures do not always have to be deployed, different security vs. performance trade-offs can be considered and one can have different security levels in encryption and decryption. ISAP relies on a re-keying mechanism so that only SPA security is needed for the whole implementation, which is easy to obtain in hardware.[6] But it has no flexibility: the overheads of the leakage-resilient PRF have to be paid even if side-channel security is not a concern.

A slightly longer-term question relates to the choice between permutations and Tweakable Block Ciphers (TBCs). While the same leakage-resistant features can be obtained at somewhat similar costs from permutations and sponges, these two building blocks also come with some differences. On the one hand, TBC-based designs seem more amenable to security analyzes in the standard model [BGP+20, BGPS21], while permutations currently require idealized assumptions [DM19, GPPS20]. On the other hand, TBC-based schemes enable performing an inverse-based tag verification that can leak in full [BPPS17] while permutation-based schemes require masking [BMPS21] or additional computations [DM21] for securing this part of their design against leakage.

2 Mode-level analysis

Our mode-level analysis follows the framework of Bellizia et al. [BBC+20]. Because of place constraints, we do not detail the specifications of the NIST's Lightweight Crypto Competition's Finalists and refer to the webpage `https://csrc.nist.gov/Projects/lightweight-cryptography` for this purpose. We rather focus on the features of these modes that are relevant for leakage.

The high-level decomposition of the modes we will rely on is depicted in Figure 1. It includes a KDF that generates a fresh encryption key K^*, the bulk of the scheme that processes the message blocks, the TGF that generates the authentication tag T and the verification (that checks whether T is correct). Some parts may naturally be empty for some candidates.

The goal of this decomposition is to identify the parts of the modes that must be implemented in a DPA-resistant manner and the parts of the modes that can

[6] Security in low-end embedded software implementations is unclear both for masking and re-keying, which can be the target of strong attacks in low-noise contexts: see [BS21] for masking and [KPP20, BBC+20] for re-keying.

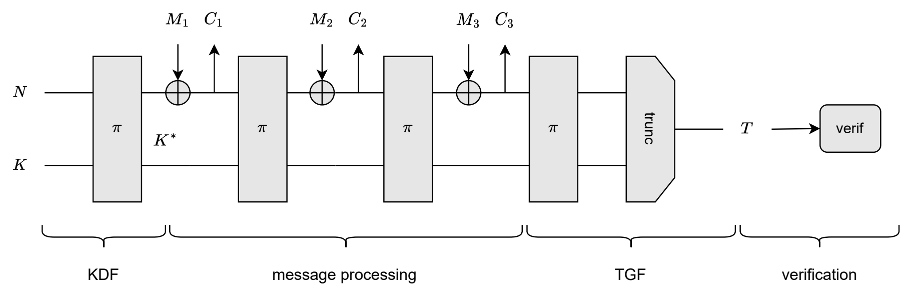


Fig. 1: Leakage-resistant modes of operation decomposition.

be implemented with weaker guarantees. When analyzing confidentiality, these weaker guarantees correspond to SPA security. When analyzing integrity, it is even possible to implement those parts without any guarantee (which is referred to as the unbounded leakage model in formal analyzes). We next classify the designs based on the amount of mode-level protections they embed. At high-level (details are given in [BBC+20]), Grade-0 designs do not provide mode-level leakage-resistance; Mode-1 designs can be leveled to preserve confidentiality and integrity as long as only encryption leakages are given to the adversary (i.e., CCAmL1 and CIML1); Mode-2 designs can be leveled even if integrity with decryption leakage is required (i.e., CIML2); Mode-3 designs complete the picture by allowing leveled implementations that preserve both confidentiality and integrity with decryption leakages (i.e., CCAmL2 and CIML2).

Grade-0 designs (no mode-level protections). A first way to design modes of operation for lightweight cryptography is to focus exclusively on performance and to ignore leakage. This is the case of modes where the long-term secret key is used by most of the underlying primitives. In the NIST lightweight crypto competition, it is for example what happens for Elephant, GIFT-COFB, Romulus-N, Romulus-M and TinyJambu. A protected implementation of Romulus-N targeting integrity with encryption leakage is illustrated in Figure 2, where the blue color is used to reflect that the corresponding computations must be protected against DPA. This requirement essentially holds for any security target (i.e., for confidentiality and integrity, with or without nonce-misuse and leakage available in encryption or decryption). We insist that being Grade-0 does not imply that these modes cannot be protected against leakage. It rather implies that this protection will be expensive because uniformly applied to all the components of the modes. The following (higher-level) designs gradually increase the mode-level protections, leading to different trade-offs between the efficiency of their unprotected implementations (that mildly decreases) and the efficiency of their protected implementations (that significantly increases for long messages).

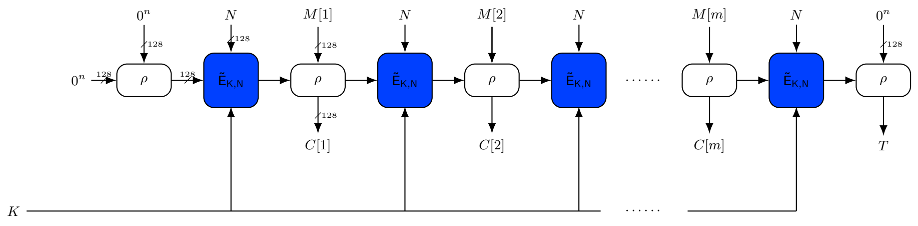


Fig. 2: Uniformly protected implementation of Romulus-N (integrity with encryption leakage). Blue blocks have to be secure against DPA.

Grade-1 designs (internal re-keying). A first step towards building modes of operation that cope better with leakage is to embed an internal re-keying mechanism. In this case, the mode first generates a fresh key K^* from the long-term key and the nonce, which is then updated after the processing of each message block. As a result, and as long as the adversary can only observe encryption leakage without nonce misuse, only the KDF needs security against DPA (as there is a DPA using the nonce) and all the other computations must only be protected against SPA. Such a leveled implementation is illustrated in Figure 3 for PHOTON-Beetle. Unfortunately, this guarantee vanishes as soon as nonce misuse or decryption leakage are granted to the adversary. In this case the adversary can target the processing of one message block with many different messages (while keeping the nonce and all the the other message blocks constant) and perform a DPA to recover the corresponding intermediate state. In the case of a P-sponge construction [BDPA07], it is then possible to invert the permutation and get back to the long-term key. In the NIST lightweight crypto competition, it is the case of PHOTON-Beetle, Sparkle and Xoodyak.

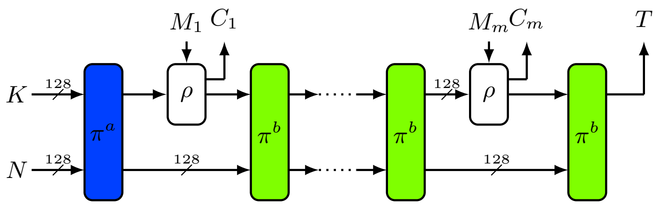
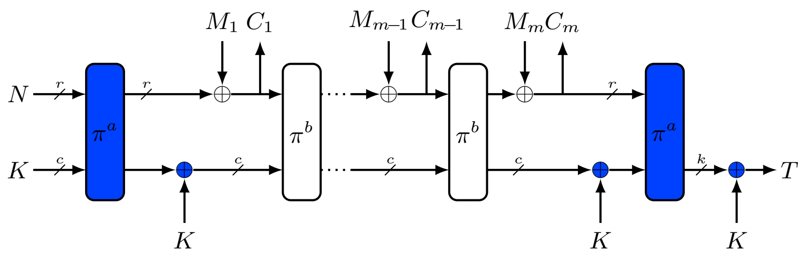
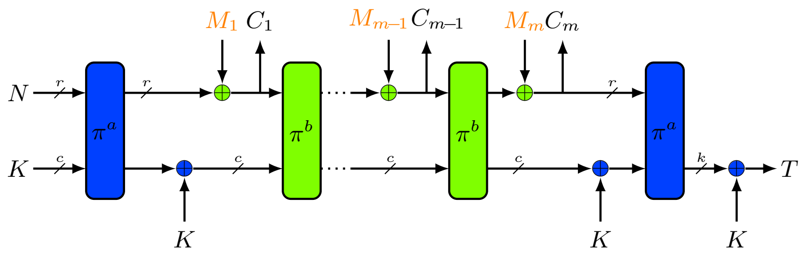


Fig. 3: Leveled implementation of PHOTON-Beetle (integrity with encryption leakage). Blue (resp., green) blocks have to be secure against DPA (resp., SPA).

Grade-2 (Grade-1 + strengthened KDF/TGF). The second step towards building modes of operation that cope better with leakage is to strengthen the KDF/TGF so that the recovery of an internal state of the mode cannot lead to long-term secrets. This is easily (and efficiently) done by making the KDF and the TGF non-invertible. In the case of sponges, it can be achieved by XORing the long-term key before and after the permutation used to generate the fresh key K^* and the tag T. For TBCs, it is a direct consequence of their PRP security. In the NIST lightweight crypto competition, it is for example the case of Ascon. For illustration, its leveled implementation is illustrated in Figure 4.



(a) Integrity requirements (with decryption leakage).



(b) Confidentiality requirements (without decryption leakage).

Fig. 4: Leveled implementation of Ascon. The blue blocks have to be protected against DPA and the green blocks have to be protected against SPA, while the white ones do not require protection against side-channel leakage.

The top of the figure depicts the integrity requirements. In this case, only the KDF and the TGF (in blue) must be protected against DPA and the rest of the computations (in white) can leak in full. This guarantee holds even when nonce misuse and leakage in decryption are granted to the adversary. It intuitively derives from the fact that the ephemeral secrets cannot be used to infer long-term ones, and corresponds to the top of the hierarchy introduced in [GPPS19]. The bottom of the figure depicts the confidentiality requirements. In this case, it is naturally not possible to tolerate unbounded leakage. Yet, as long as the adversary is not granted with decryption leakage, only SPA security (in green) is required for this part of the computation. (The orange color for the plaintexts is used to reflect that even their very manipulation may leak sensitive information). The main attack vector that remains against this construction happens with decryption leakage. Since the message is decrypted before verifying the tag, an adversary can then target the processing of one message block with many different messages (keeping the nonce and all the the other message blocks constant) and perform a DPA to recover the corresponding intermediate state. This reveals the ephemeral keystream, hence the message, but does not affect the confidentiality of messages encrypted with a different nonce.

Grade-3 (Grade-2 + two passes). The natural way to get rid of the last attack vector against Ascon is to consider 2-pass designs (such as encrypt-then-MAC constructions). In this case, the tag can be computed from the ciphertext blocks and tested before the decryption takes place. In the NIST lightweight crypto competition, it is for example the case of ISAP (which is permutation-based) and Romulus-T (which is TBC-based). Their main difference lies in the way they secure their KDF and TGF. Like Ascon, Romulus-T (which is based on the TEDT mode of operation [BGP+20]) relies on masking for this purpose. By contrast, ISAP relies on a leakage-resilient PRF. As illustrated in Figure 5, the leakage-resilient PRF can be viewed as a re-keying scheme where the nonce bits are absorbed one by one so that each of its intermediate keys is only used to process two permutation calls. As a result, this PRF essentially "reduces" DPA security to SPA security, at the cost of iterating the Ascon-p^1 permutation.[7]

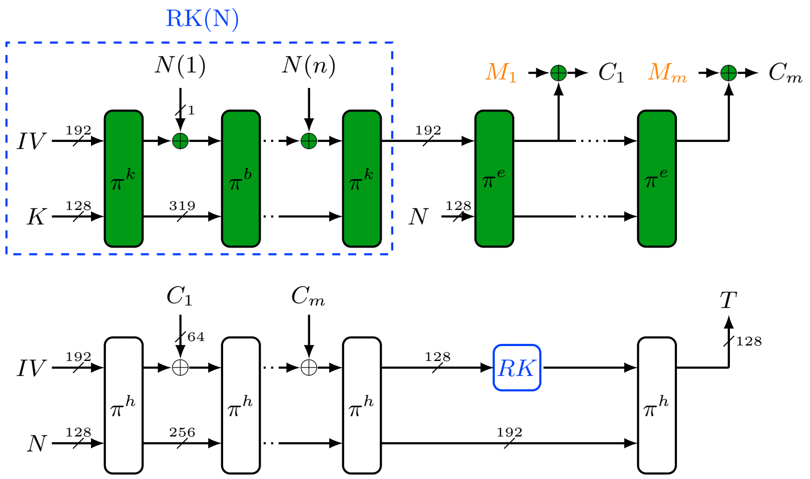


Fig. 5: Leveled implementation of ISAP (confidentiality with decryption leakage). The green blocks have to be protected against SPA (with averaging), while the white ones do not require any protection against side-channel leakage.

Overall, ISAP's confidentiality with decryption leakage requires two calls to the leakage-resilient PRF and a plaintext processing that is secure against SPA. We provide a similar picture for the leveled implementation of Romulus-T in Figure 6, where the KDF and TGF are directly instantiated with a masked

[7] Increasing the rate to absorb more bits and get a more efficient design is possible but it then opens a DPA attack vector (so we do not consider this option here).

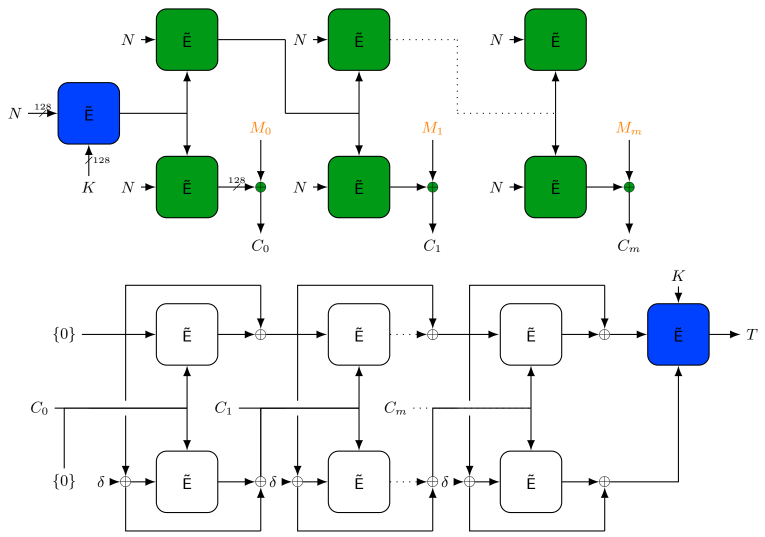


Fig. 6: Leveled implementation of Romulus-T (confidentiality with decryption leakage). The blue blocks have to be protected against DPA and the green blocks have to be protected against SPA (with averaging), while the white ones do not require any protection against side-channel leakage.

TBC. Note that the SPA-secure blocks of these two figures are in dark green to reflect the possibility that the adversary repeats the same measurements to average the noise. And as in the case of Ascon, integrity with decryption leakage only requires the two protected calls (to the leakage-resilient PRF or masked TBC) and can let all the other computations leak in full.

3 Hardware implementations

Given the previous analysis, it appears that Ascon, ISAP and Romulus-T are the most promising candidates of the NIST lightweight crypto competition for leakage-resistant implementations. In this section, we therefore investigate their hardware implementations. For this purpose, we focus on the security guarantees that they enable without a uniformly protected implementation. We first investigate their primitives, with a special focus on Romulus-T and its underlying TBC Skinny-384+ (for which, as mentioned in introduction, the literature is a bit scarcer). We then detail the implementation of the modes and report their performances (with ASIC synthesis). We use these results to confirm the relevance of leveled implementations and to discuss the respective interest of the three implemented ciphers in the context of the NIST competition.

Algorithm 1 HPC2 AND gadget with d shares (sync. registers are omitted).

Input: Sharings $\mathbf{x}$, $\mathbf{y}$
Output: Sharing $\mathbf{z}$ such that $z = x \cdot y$.

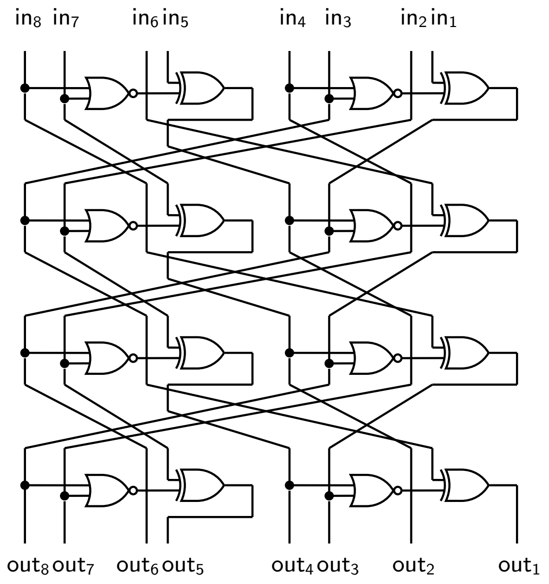
```
1: for $i = 0$ to $d - 1$ do
2:     for $j = i + 1$ to $d - 1$ do
3:         $r_{ij} \overset{\$}{\leftarrow} \mathbb{F}_2;\ r_{ji} \leftarrow r_{ij}$
4: for $i = 0$ to $d - 1$ do
5:     $\mathbf{z}_i \leftarrow \mathbf{x}_i \mathbf{y}_i$
6:     for $j = 0$ to $d - 1, j \neq i$ do
7:         $\mathbf{z}_i \leftarrow \mathbf{z}_i \oplus \mathsf{Reg}\left((\mathbf{x}_i \oplus 1)\, r_{ij}\right) \oplus \mathsf{Reg}\left(\mathbf{x}_i \mathsf{Reg}\left(\mathbf{y}_j \oplus r_{ij}\right)\right)$
```

3.1 Masked implementation of the primitives

We use the HPC2 masking scheme, as it allows almost arbitrary composition while ensuring security against both hardware glitches and transitions [CGLS21, CS21]. The main characteristics of this masking scheme are the following. The linear operations are very efficient, since they are made of purely combinational logic and have a linear overhead in the masking order. On the other hand, the non-linear operation, which is the 2-input AND gate (see Algorithm 1), has quadratic overhead and asymmetric latency: 2 cycles with respect to one input, and only 1 cycle with respect to the other input.

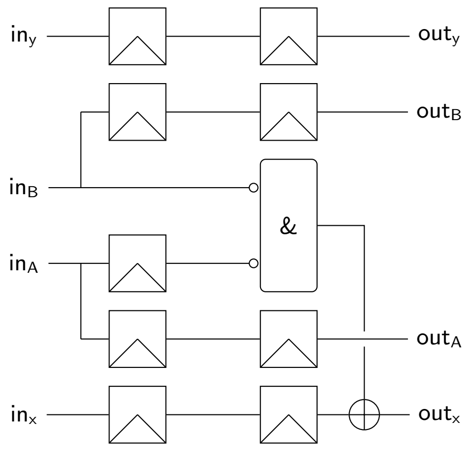
Skinny Sbox. The Skinny Sbox (depicted in Figure 7) is made of XOR gates (that are linear, and therefore easy to implement) and of NOR gates that we implement with a AND gate whose inputs are inverted.

We next propose two area-optimized architectures for this Sbox, that both instantiate two masked AND and two masked XOR gates. First, the high-throughput one is based on the observation that the Sbox can be decomposed into 4 applications of a simpler function (as visible in Figure 7), followed by a wire shuffling. This function has a latency of at least two cycles with our masking scheme, due to the AND gate. We therefore build the high-throughput Sbox by looping 4 times on a two-stage pipeline. The core of this pipeline is a simple function block B shown in Figure 8a, which is then used twice and connected to combinational logic to form the Sbox (Figure 8b). The two-stage pipeline can be used to perform simultaneously two Sbox operations. We finally add input and output synchronization registers such that the logic performs two Sbox evaluations in 9 clock cycles, without the need for any external synchronization mechanism. Second, we design a low-latency architecture with an ALU-style design: the Sbox inputs are stored in a register, as well as the outputs, and the data fed to the two AND-XOR blocks are selected from these states (and from the input wires) when needed (see Figure 10). This flexible architecture allows to benefit from the asymmetric latency of the HPC2 AND gadgets, leading to a latency of 6 cycles for one Sbox evaluation (see Figure 9).

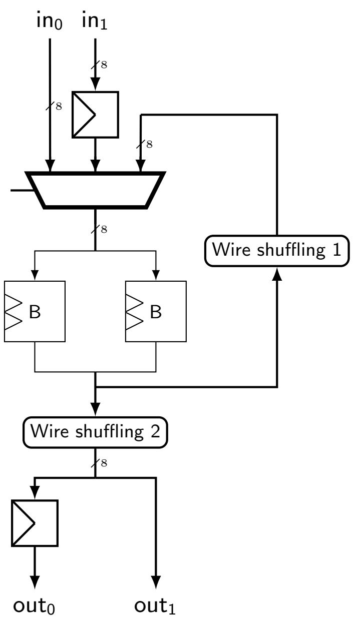


in8 in7 in6 in5 in4 in3 in2 in1
out8 out7 out6 out5 out4 out3 out2 out1

Fig. 7: Skinny Sbox circuit representation with NOR and XOR gates.

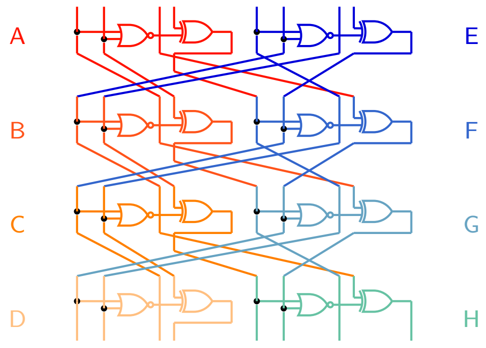


(a) Inner logic block "B" for Sbox implementation, with one AND and one XOR.

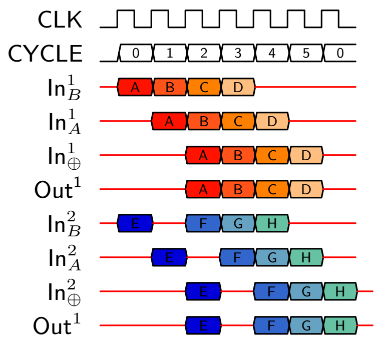


(b) High-throughput Sbox architecture with input and output sync. registers.

Fig. 8: High-throughput masked Skinny Sbox.



(a) Decomposition of the logic circuit in iterated logic blocks (A to H).



(b) Scheduling of the computations.

Fig. 9: Decomposition and scheduling of the computations for low-latency masked Skinny Sbox: 8 logic blocks are scheduled on 2 block instances.

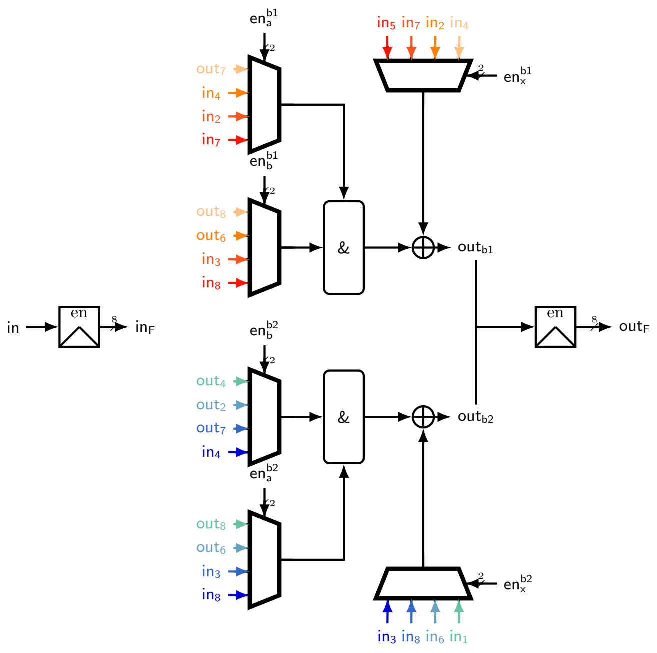

en_a^{b1}
en_b^{b1}
en_x^{b1}
en_b^{b2}
en_a^{b2}
en_x^{b2}
out_7 in_4 in_2 in_7
out_8 out_6 in_3 in_8
in_5 in_7 in_2 in_4
out_4 out_2 out_7 in_4
out_8 out_6 in_3 in_8
in_3 in_8 in_6 in_1
&
out_{b1}
out_{b2}
in
en
in_F
out_F
8
2

Fig. 10: Architecture of low-latency masked Skinny Sbox.

Lastly, we discuss fully pipeline architectures. While such architectures can achieve very high throughput that compensate for their large area, they are difficult to use in our case. Indeed, we are not interested in parallel Skinny evaluations (since this is not useful for encrypting a single message with a Romulus-T leveled implementation). Therefore, the latency overhead of filling the pipeline (of at least 6 cycles) is significant when it is used for only 16 Sbox evaluations (a Skinny round). We however note that the only other HPC Skinny implementation we know of uses that strategy, and uses a depth-12 pipeline [KB22].[8]

Skinny. Based on these Sbox architectures, we design three Skinny implementations with various area vs. latency trade-offs. The two first ones are simple round-based architectures, as shown in Figure 11, where either 16 low-latency Sboxes ("low-latency Skinny", S_{LL}) or 8 high-throughput ("balanced Skinny", S_{B}) Sboxes are used. The third implementation ("small Skinny", S_{S}) targets lower area: it is a serialized architecture that instantiates only one high-throughput Sbox (that is used 8 times per round), as shown in Figure 12.

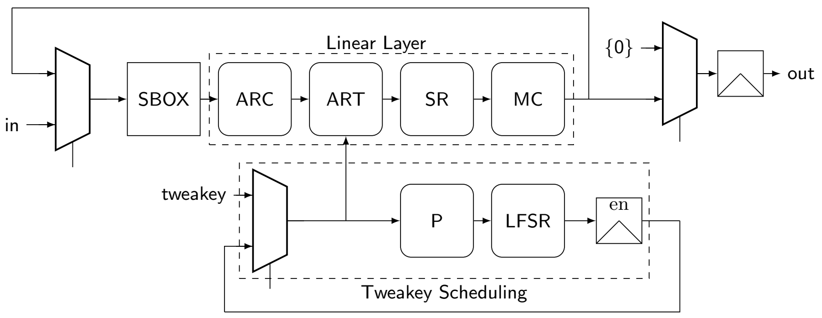


Fig. 11: Round-based masked Skinny architecture (S_{LL} and S_{B}).

Let us now discuss the performance of these implementations. We consider latency, randomness requirements and area as performance metrics, since the critical path will be similar in all cases (it lies in the linear layer). The latency and maximum randomness requirements per cycle of the implementations are shown in Table 1. We can see that the S_{S} implementation has 8 times the latency of S_{B} (due to 8x serialization), while S_{LL} reduces latency by 33 % compared to S_{B}. Regarding randomness, the maximum randomness throughput of S_{S} is 8 times lower than S_{B}, and the one of S_{LL} is twice the one of S_{B}.

Next, we look at area requirements in Figure 13. The Sbox logic area clearly reflects the architectural choices: 2 AND and XOR gadgets for S_{S}, 16 of each for S_{B}, and 32 of each for S_{LL}. Next, the remaining Sbox area is fairly high for S_{LL} due to the large number of Sbox instances and due to their large MUXes and registers. For S_{B} and S_{S}, the larger number MUXes and registers in the Sboxes

[8] Which can be improved to depth-6 thanks to the asymmetric latency of HPC2 ANDs.

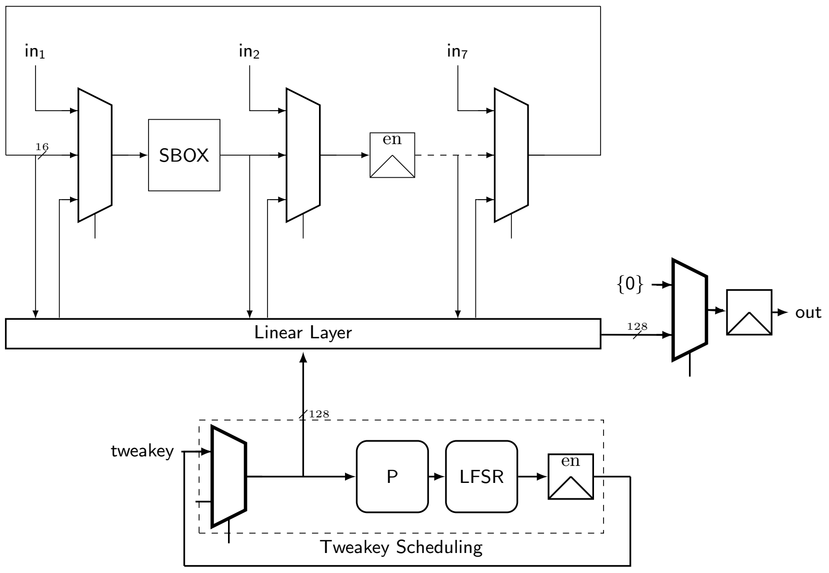


Fig. 12: Serialized masked Skinny architecture (S_{S}).

	Latency [cycle]	Randomness [bit]				
		$d = 2$	$d = 3$	$d = 4$	$d = 5$	$d = 6$
S_{S}	2880	2	6	12	20	30
S_{B}	360	16	48	96	160	240
S_{LL}	240	32	96	192	320	480

Table 1: Skinny-384+ masked implementations: total latency and maximum randomness consumption for a single clock cycle (where the total randomness consumption is $64 \cdot d \cdot (d - 1)$ bits for all three implementations).

of the former compensate for the more complex datapath of the latter, resulting in a similar "routing" area for both of them. The remaining parts of Skinny are the same for all three architectures. Overall, the difference in area between the architectures is small for low number of shares, and increases as the latter grows. For all considered number of shares ($d \leq 6$), the Sboxes do not dominate the area of neither S_{S} nor S_{B}, hence S_{S} brings a limited area gain at a large latency cost compared to S_{B}. On the other hand, S_{LL} has an area overhead of up to 39 % (for $d \leq 6$), and a latency gain of 33 % over S_{B}.

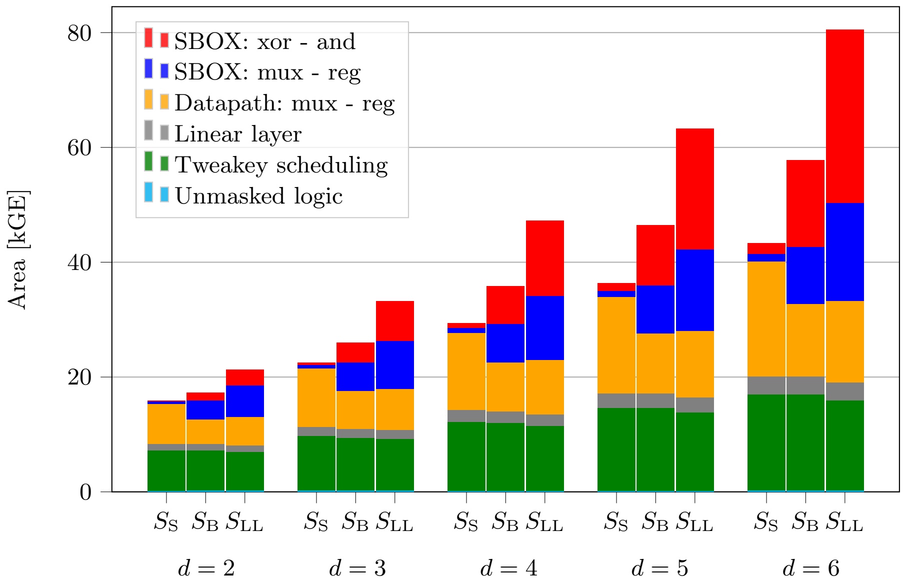


Fig. 13: Area requirements for the three masked Skinny hardware implementations in a 65 nm ASIC technology using the HPC2 masking scheme.

3.2 Implementation of the modes

We implemented side-channel protected hardware accelerators for Romulus-N, Romulus-T, Ascon and ISAP, using the primitives described in Table 2. The Romulus-N implementation is fully masked and uses one S_{LL} instance. Next, the implementation of Romulus-T is leveled with one masked instance of Skinny (we also used S_{LL}) and four non-masked Skinny instances (with a round-based architecture). Similarly, the Ascon implementation is also leveled. The masked Ascon-p primitive is based on the HPC2 masking scheme and is serialized with 16 Sbox instances (each Sbox is a 2-stage pipeline performing 4 Sbox evaluations per round)[9], while the non-masked permutation is round-based (1 cycle per round). Finally, ISAP uses two instances of the non-masked Ascon-p primitive.

[9] This choice is somewhat arbitrary: we took a serialization factor that gives a good latency versus area trade-off. It also happens to lead to a latency of 6 clock cycles per round, which is the same latency as a round of S_{LL}.

	Masked		Non-masked	
	Latency	Architecture	Latency	Architecture
Skinny-384+	240	S_{LL}	40	round-based
Ascon-p^6			6	round-based
Ascon-p^{12}	72	serialized 4x	12	round-based
ISAP RK			152	round-based

Table 2: Primitive implementations used in the AEAD cores: latency in clock cycles and architecture for masked and non-masked versions.

Let us first discuss the latency of these implementations with Figure 14. The encryption time of Romulus-N grows very quickly with the message size due to the need of masking all Skinny calls, which are slow compared to non-masked calls (as shown in Table 2). However, for very short messages, Romulus-N is fairly competitive thanks to its low number of Skinny calls it that case. On the other hand, Romulus-T has a larger upfront cost, due to the larger number of Skinny calls even for short messages, however the mode-level leakage resistance allows to use non-masked calls for the bulk processing, resulting in lower latency than Romulus-N for long messages. Next, Ascon enjoys lower initial latency and long-message latency than Romulus-T. This is due to the lower number of rounds in Ascon-p^6 and Ascon-p^{12} compared to Skinny (which has 40 rounds). Finally, the latency of ISAP is between the one of Ascon and Romulus-T. Indeed, ISAP's bulk processing is very similar to Ascon's, but uses more rounds to increase the security margin in presence of leakage. Moreover, the leakage-resilient PRF of ISAP uses many permutation rounds, which makes it slower than a masked Ascon for short messages, while still being faster than Romulus-T.

Let us now discuss the area usage with Figure 15. As a general trend, for implementations with a masked primitive, the area for that primitive dominates the overall area, with the exception of Romulus-T with $d = 2$ shares (where the area of the four non-masked Skinny instances dominates). These results therefore confirms the interest of leveled implementations. Next, the areas for all these modes is fairly similar, with a slight advantage to Ascon and Romulus-N at $d = 2, 3$ shares thanks to their lower unmasked area, while Romulus-T and Romulus-N are a bit better than Ascon for larger numbers of shares, thanks to using less masked AND gadgets. Lastly, the area for ISAP is similar to the area of the leveled implementations with $d = 2$ shares.

4 Conclusion

Even though the previous quantitative results should be interpreted with care, since they explore only a few points in the design space (e.g., we considered only round-based architectures for non-masked primitives and did not optimize the masking randomness usage), their comparison highlights a few general trends.

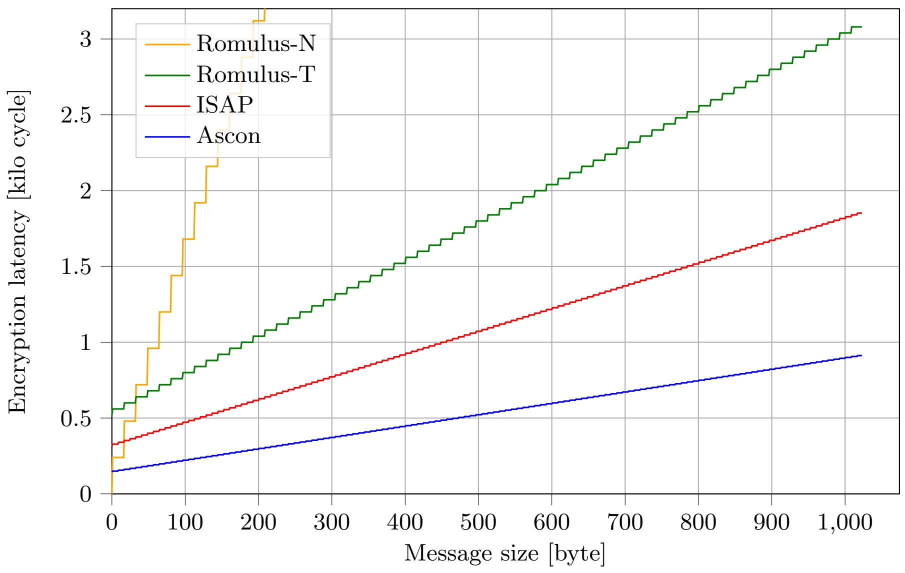


Fig. 14: Encryption latency as a function of the message size.

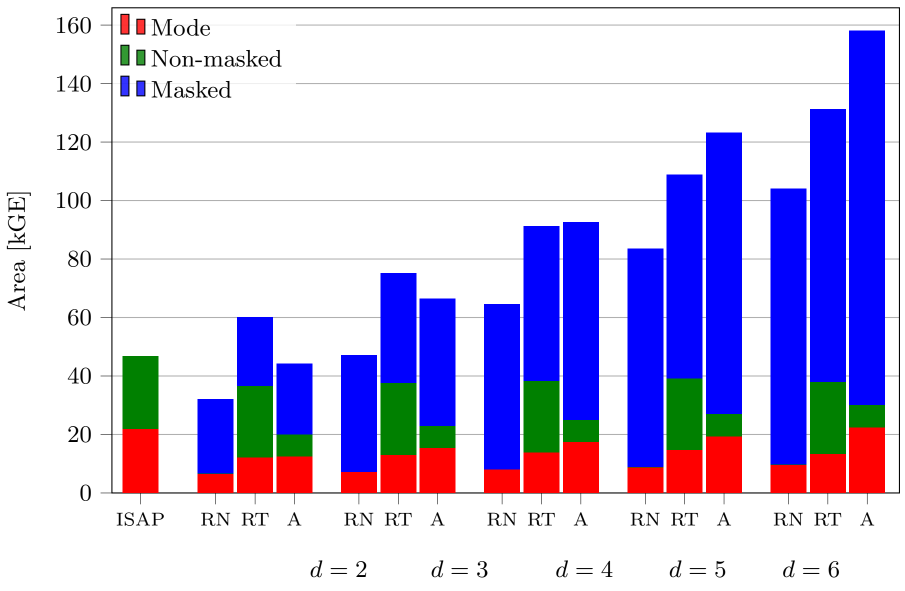


Fig. 15: Area requirements of leakage-resistant hardware AEAD cores in 65 nm ASIC technology. For leveled implementations, the area is split in three part: DPA-protected (i.e., masked) primitives, SPA-protected primitives (implemented in parallel), and mode (i.e., logic not in a primitive). RN/RT respectively stand for Romulus-N/Romulus-T and A stands for Ascon.

First, for long messages, leveled implementations bring large latency improvements while their area overheads remain small over non-leveled implementations. This is because unmasked primitives are small compared to the masked ones, especially when the number of shares is large. This smaller unmasked area as well as lower latency naturally translate into large energy savings. The candidates that can be implemented in such a way for both encryption and decryption (grade-2 and grade-3, see Table 3) will benefit most from these savings. However, having such mode-level characteristics usually implies more complex modes of operations, which leads to worse performance for small messages, as shown by the comparison between Romulus-N and the grade-2/grade-3 candidates.

Grade	Security	Candidates
0	CCA+CI	Elephant, GIFT-COFB, Romulus-M/N, TinyJambu
1	CCAL1+CIL1	PHOTON-Beetle, Sparkle, Xoodyak
2	CCAmL1+CIML2	Ascon
3	CCAmL2+CIML2	ISAP, Romulus-T

Table 3: NIST LWC finalists grouped by mode-level leakage resistance.

Second, the leakage-resilient PRF technique used by ISAP leads to low area implementation (similar to a leveled implementations with two shares), and its latency is comparable to leveled implementations. Such techniques therefore appear quite promising in hardware implementation setting. Yet, we note that on the side-channel security side, the formal security guarantees of such implementations have been much less analyzed than masking and their practical security evaluation can be more challenging as well (see, e.g., [UBS21]).

Finally, we observe that the different security margins of the algorithms we implemented can explain some of the observed performance differences. For example, the differences in latency between the leveled implementations of Ascon and ISAP are explained by their number of rounds: the hashing part of ISAP uses Ascon-p^{12} while the Ascon inner sponge part uses only Ascon-p^{6}. When considering CIML2, both should however withstand the same attacks.

Overall, these results backup our suggestion that from a side-channel security viewpoint, the finalists of the NIST's Lightweight Crypto competition differ more by their qualitative features than by their quantitative performances.

Acknowledgments. Gaëtan Cassiers and François-Xavier Standaert are respectively Research Fellow and Senior Associate Researcher of the Belgian Fund for Scientific Research (FNRS-F.R.S.). This work has been funded in part by the ERC consolidator grant number 724725 (acronym SWORD), by the Walloon region CyberExcellence project number 2110186 (acronym Cyberwal) and by the Horizon Europe project 1010706275 (acronym REWIRE).

References

BBC$^+$20. Davide Bellizia, Olivier Bronchain, Gaëtan Cassiers, Vincent Grosso, Chun Guo, Charles Momin, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Mode-level vs. implementation-level physical security in symmetric cryptography - A practical guide through the leakage-resistance jungle. In *CRYPTO (1)*, volume 12170 of *Lecture Notes in Computer Science*, pages 369–400. Springer, 2020.

BDPA07. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sponge functions. In *ECRYPT hash workshop*, 2007.

BGP$^+$20. Francesco Berti, Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Tedt, a leakage-resist AEAD mode for high physical security applications. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(1):256–320, 2020.

BGPS21. Francesco Berti, Chun Guo, Thomas Peters, and François-Xavier Standaert. Efficient leakage-resilient macs without idealized assumptions. In *ASIACRYPT (2)*, volume 13091 of *Lecture Notes in Computer Science*, pages 95–123. Springer, 2021.

BJK$^+$20. Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. SKINNY-AEAD and skinny-hash. *IACR Trans. Symmetric Cryptol.*, 2020(S1):88–131, 2020.

BMPS21. Olivier Bronchain, Charles Momin, Thomas Peters, and François-Xavier Standaert. Improved leakage-resistant authenticated encryption based on hardware AES coprocessors. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(3):641–676, 2021.

BPPS17. Francesco Berti, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. On leakage-resilient authenticated encryption with decryption leakages. *IACR Trans. Symmetric Cryptol.*, 2017(3):271–293, 2017.

BS21. Olivier Bronchain and François-Xavier Standaert. Breaking masked implementations with many shares on 32-bit software platforms or when the security order does not matter. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(3):202–234, 2021.

BSH$^+$14. Sonia Belaïd, Fabrizio De Santis, Johann Heyszl, Stefan Mangard, Marcel Medwed, Jörn-Marc Schmidt, François-Xavier Standaert, and Stefan Tillich. Towards fresh re-keying with leakage-resilient prfs: cipher design principles and analysis. *J. Cryptogr. Eng.*, 4(3):157–171, 2014.

CGLS21. Gaëtan Cassiers, Benjamin Grégoire, Itamar Levi, and François-Xavier Standaert. Hardware private circuits: From trivial composition to full verification. *IEEE Trans. Computers*, 70(10):1677–1690, 2021.

CGP$^+$12. 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 *COSADE*, volume 7275 of *Lecture Notes in Computer Science*, pages 69–81. Springer, 2012.

CS21. Gaëtan Cassiers and François-Xavier Standaert. Provably secure hardware masking in the transition- and glitch-robust probing model: Better safe than sorry. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(2):136–158, 2021.

DEM$^+$20. Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, Bart Mennink, Robert Primas, and Thomas Unterluggauer. Isap v2.0. *IACR Trans. Symmetric Cryptol.*, 2020(S1):390–416, 2020.

DEMS21. Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2: Lightweight authenticated encryption and hashing. *J. Cryptol.*, 34(3):33, 2021.

DM19. Christoph Dobraunig and Bart Mennink. Leakage resilience of the duplex construction. In *ASIACRYPT (3)*, volume 11923 of *Lecture Notes in Computer Science*, pages 225–255. Springer, 2019.

DM21. Christoph Dobraunig and Bart Mennink. Leakage resilient value comparison with application to message authentication. In *EUROCRYPT (2)*, volume 12697 of *Lecture Notes in Computer Science*, pages 377–407. Springer, 2021.

GPPS19. Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Authenticated encryption with nonce misuse and physical leakage: Definitions, separation results and first construction - (extended abstract). In *LATINCRYPT*, volume 11774 of *Lecture Notes in Computer Science*, pages 150–172. Springer, 2019.

GPPS20. Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Towards low-energy leakage-resistant authenticated encryption from the duplex sponge construction. *IACR Trans. Symmetric Cryptol.*, 2020(1):6–42, 2020.

GWDE15. Hannes Groß, Erich Wenger, Christoph Dobraunig, and Christoph Ehrenhöfer. Suit up! - made-to-measure hardware implementations of ASCON. In *DSD*, pages 645–652. IEEE Computer Society, 2015.

GWDE17. Hannes Groß, Erich Wenger, Christoph Dobraunig, and Christoph Ehrenhfer. Ascon hardware implementations and side-channel evaluation. In *Microprocessors & Microsystems*, volume 52 (C). Elsevier Science Publishers, 2017.

KB22. Mustafa Khairallah and Shivam Bhasin. Hardware implementations of romulus: Exploring nonce-misuse resistance and boolean masking. In *NIST Lightweight Cryptography Workshop*, 2022.

Kha22. Mustafa Khairallah. Hardware implementations of romulus: Exploring nonce misuse resistance and boolean masking. In *Lightweight Cryptography Workshop*, 2022.

KPP20. Matthias J. Kannwischer, Peter Pessl, and Robert Primas. Single-trace attacks on keccak. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(3):243–268, 2020.

MPG05. Stefan Mangard, Thomas Popp, and Berndt M. Gammel. Side-channel leakage of masked CMOS gates. In *CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 351–365. Springer, 2005.

MSGR10. Marcel Medwed, François-Xavier Standaert, Johann Großschädl, and Francesco Regazzoni. Fresh re-keying: Security against side-channel and fault attacks for low-cost devices. In *AFRICACRYPT*, volume 6055 of *Lecture Notes in Computer Science*, pages 279–296. Springer, 2010.

UBS21. Balazs Udvarhelyi, Olivier Bronchain, and François-Xavier Standaert. Security analysis of deterministic re-keying with masking and shuffling: Application to ISAP. In *COSADE*, volume 12910 of *Lecture Notes in Computer Science*, pages 168–183. Springer, 2021.