Mode-Level vs. Implementation-Level Physical Security in Symmetric Cryptography

A Practical Guide Through the Leakage-Resistance Jungle

Davide Bellizia[1], Olivier Bronchain[1], Gaëtan Cassiers[1], Vincent Grosso[2], Chun Guo[3], Charles Momin[1], Olivier Pereira[1], Thomas Peters[1], and François-Xavier Standaert[1](✉)

[1] Crypto Group, ICTEAM Institute, UCLouvain, Louvain-la-Neuve, Belgium
{davide.bellizia,olivier.bronchain,gaetan.cassiers,charles.momin, olivier.pereira,thomas.peters,fstandae}@uclouvain.be

[2] CNRS/Laboratoire Hubert Curien, Université de Lyon, Lyon, France
vincent.grosso@univ-st-etienne.fr

[3] School of Cyber Science and Technology and Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan, China
chun.guo.sc@gmail.com

Abstract. Triggered by the increasing deployment of embedded cryptographic devices (e.g., for the IoT), the design of authentication, encryption and authenticated encryption schemes enabling improved security against side-channel attacks has become an important research direction. Over the last decade, a number of modes of operation have been proposed and analyzed under different abstractions. In this paper, we investigate the practical consequences of these findings. For this purpose, we first translate the physical assumptions of leakage-resistance proofs into minimum security requirements for implementers. Thanks to this (heuristic) translation, we observe that (*i*) security against physical attacks can be viewed as a tradeoff between mode-level and implementation-level protection mechanisms, and (*ii*) security requirements to guarantee confidentiality and integrity in front of leakage can be concretely different for the different parts of an implementation. We illustrate the first point by analyzing several modes of operation with gradually increased leakage-resistance. We illustrate the second point by exhibiting leveled implementations, where different parts of the investigated schemes have different security requirements against leakage, leading to performance improvements when high physical security is needed. We finally initiate a comparative discussion of the different solutions to instantiate the components of a leakage-resistant authenticated encryption scheme.

1 Introduction

State-of-the-art. Since the introduction of side-channel attacks in the late nineties [55,57], securing cryptographic implementations against leakage has been a major research challenge. These attacks raise critical security concerns, as they enable recovering sensitive information such as long-term secret keys, and are virtually applicable to any type of implementation if no countermeasures are deployed [61]. As a result, various types of protection mechanisms have been introduced, working at different abstraction levels. Due to the physical nature of the leakage, the first countermeasures were typically proposed at low abstraction levels. For example, hardware countermeasures can target a reduction of the side-channel information by blurring the signal into noise in the time or amplitude domains [22,60], or by reducing this signal thanks to special (dual-rail) circuit technologies [81,82]. These hardware countermeasures can then be augmented by implementation-level randomization mechanisms aimed at amplifying the side-channel leakage reduction. Masking achieves this goal by exploiting data randomization (i.e., secret sharing) [20,40] and shuffling does it by randomizing the order of execution of the operations [48,85]. Steady progresses have been made in order to improve the understanding of these different countermeasures. For example, masking is supported by a strong theoretical background (see [5,31,32,49] to name a few). Yet, it remains that the secure implementation of low-level countermeasures (e.g., masking) is quite sensitive to physical defaults [3,23,62,66], and is expensive both in software and hardware contexts [42,43].

In view of the sensitive and expensive nature of hardware-level and implementation-level side-channel countermeasures, a complementary line of works initiated the investigation of cryptographic primitives with inherently improved security against physical leakage. In the case of symmetric cryptography, this trend started with heuristic proposals such as [37,56,63,69]. It was then formalized by Dziembowski and Pietrzak under the framework of leakage-resilient cryptography [33], which has been the inspiration of many follow up works and designs. Simple and efficient PRGs & stream ciphers were proposed in [70,78,86,87]. PRFs and PRPs can be found in [1,30,35,79]. Concretely, such leakage-resilient primitives typically require some type of bounded leakage assumption, which was found to be significantly easier to fulfill for PRGs and stream ciphers that are naturally amenable to key evolution schemes than for PRFs and PRPs for which each execution requires the manipulation of a long-term secret key [8].

The concept of leveled implementations introduced by Pereira et al. closely followed this finding: it aims at combining the minimum use of a PRF or PRP heavily protected against side-channel attacks thanks to (possibly expensive) hardware-level and implementation-level countermeasures with a mode of operation designed to cope with leakage, requiring much less protections (or even no protection at all) in order to process the bulk of the computation [68].

These seed results on basic cryptographic primitives (such as PRGs, PRFs and PRPs) next triggered analyzes of complete functionalities like encryption and authentication, and rapidly shifted the attention of designers to Authenticated Encryption (AE) schemes mixing both integrity and confidentiality guarantees. Following the seminal observation of Micali and Reyzin that indistinguish-ability-based notions are significantly harder to capture and ensure with leakage than unpredictability-based notions [64], strong integrity properties with leakage have then been investigated, first with leakage in encryption only [14], next with leakage in encryption and decryption [15]. It turns out they could indeed be satisfied with weak physical assumptions for the bulk of the computation. For example, ciphertext integrity can be achieved with full leakage of all the intermediate computations of an AE scheme and two manipulations of a long-term secret key with a strongly protected block cipher implementation. This is obviously insufficient for any type of confidentiality guarantee, as it would leak plaintexts immediately. This last conclusion motivated a systematic analysis of composite security definitions, enabling different physical requirements for integrity and confidentiality guarantees with leakage [46]. Eventually, various full-fledged AE schemes have been analyzed against leakage, based on Tweakable Block Ciphers (TBCs) [13], permutations [24,25,27] and combinations of both [47].

We note that our following investigations are restricted to symmetric cryptography, which is the most investigated target for practical side-channel attacks. Yet, security against leakage has also been considered for other important objects such as digital signatures schemes (e.g., [34,52,59]), public-key encryption schemes (e.g., [54,65]) and more advanced cryptographic functionalities like secure computation (e.g., [17,39]). We refer to [50] for a recent survey.

Contribution. The starting point of our investigations is the observation that the development of low-level side-channel countermeasures and the one of primitives and modes of operation to prevent leakage have for now followed quite independent paths. This can, in part, be explained by the very different abstractions used to analyze them. While low-level countermeasures are typically evaluated experimentally based on statistical metrics [77], proving the security of the aforementioned modes against leakage is rather based on cryptographic reductions leveraging some physical assumptions. In this respect, the quest for sound physical assumptions that are at the same time realistic (e.g., are falsifiable and can be quantified by evaluation laboratories) and sufficient for proofs has been and still is an important problem: see again [50]. To a large extent, the current situation therefore mimics the one of black box security proofs, with efficient schemes proven under idealized assumptions (e.g., the oracle-free leakages introduced in [87] and used for analyzing sponge constructions in [27,47]) and a quest for more standard analyses under weaker assumptions. Combined with the massive amount of definitions capturing all the possible combinations of security targets for confidentiality and integrity with leakage [46], the complexity of these theoretical investigations is therefore calling for a systematization effort towards the concrete interpretation of leakage security proofs, in order to determine how these results can help developers in the design of secure implementations.
Our main contributions in this direction are threefold:

1. We provide a simplifying framework that allows us (a) to identify a reduced number of relevant security targets (compared to the full zoo of definitions in [46]); and (b) to translate the physical assumptions used in leakage security proofs into practical implementation guidelines, stated in terms of security against Simple Power Analysis (SPA) and Differential Power Analysis (DPA),

as can be evaluated by evaluation laboratories with tools such as [21,76] and extensions thereof. Despite SPA and DPA requirements are only necessary conditions for the formal physical security assumptions to hold, this analysis allows us to pinpoint the minimum level of efforts that implementers must devote to protect different parts of an AE implementation in function of the security target.

2. We use this framework to illustrate that reasoning about security against leakage can be viewed as a tradeoff between mode-level and implementation-level (or hardware-level) protections. To give a simple example, a physical security property (e.g., the aforementioned ciphertext integrity with leakage) can theoretically be obtained from standard modes of operation like OCB [74], given that all the block cipher executions in the mode are strongly protected against DPA (which has high cost). Modes with better resistance against leakage can relax this DPA security requirement for certain parts of the implementations. Interestingly, we additionally observe that the literature already includes different modes of operation corresponding to different leakage security targets. This allows us to illustrate the physical security tradeoff based on actual algorithms, including candidates to the ongoing NIST standardization effort.[1] We will focus on OCB-Pyjamask [41], PHOTON-Beetle [4], Ascon [26], Spook [10], ISAP [25] and TEDT [13], but our analysis applies to many other similar ciphers.
3. Finally, we answer concrete questions that these analyzes suggest, namely:

   – We discuss the interest of leveled implementations compared to uniformly protected implementations based on a hardware implementation case study.
   – We compare the (masked) TBC-based and permutation-based initialization/finalization steps of two AE schemes (namely, Ascon [26] and Spook [10]) thanks to a software case study, and evaluate their respective advantages.
   – We compare a standard tag verification mechanism (that requires hardware-level or implementation-level DPA protections) with the inverse-based tag verification proposed in [15] that does not require DPA protections.
   – We evaluate the pros and cons of the two main solutions to implement a strongly protected component for AE, namely masking and fresh re-keying based on a leakage-resilient PRF [9], which is for example used in ISAP [25].

   For completeness, and despite the practical focus of the paper, we additionally provide a high-level view of the formal security guarantees that the analyzed AE schemes offer. For this purpose, we leverage the existing literature and tailor some existing generic results to the investigated cases studies.

[1] https://csrc.nist.gov/projects/lightweight-cryptography.

Related work. In an independent line of work, Barwell et al. analyzed the security of AE schemes in a model where the leakage is excluded from the challenge encryption [6]. This model corresponds to a context of leakage-resilience, where security guarantees may vanish in the presence of leakage, but are restored once leakage is removed from the adversary's view. It enables strong composition results similar to the ones obtained without leakage, strong security against nonce misuse (i.e., misuse-resistance in the sense of Rogaway and Shrimpton [75]) and has been instantiated with uniformly protected implementations (e.g., the scheme in [6] is based on masking and pairings). Here, we rather focus on the composite definitions introduced by Guo et al. [46] which consider integrity and confidentiality properties separately, rely on the setting of leakage-resistance (i.e., aim to resist against leakage even during the challenge encryption) and can only be combined with a weaker guarantee of nonce misuse-resilience (introduced by Ashur et al. [2]) for the confidentiality guarantees in the presence of leakage.[2] The motivations for this choice are twofold. First, composite definitions enable the identification of meaningful security targets matching existing AE schemes that a single "all-in-one" definition as the one of Barwell et al. cannot capture. Second, the security gains of some design tweaks we aim to analyze (e.g., ephemeral key evolution) cannot be reflected in the leakage-resilience setting. We refer to Sect. 2.2 for more discussion about this definitional choice.

Cautionary notes. By focusing on the qualitative requirements that masking security proofs imply for implementers, our next analyzes and discussions naturally elude some additional important points that should also be considered in the evaluation of a leakage-resistant AE scheme, namely:
(*i*) We do not consider the quantitative aspects: for example, do the modes provide beyond-birthday and/or multi-user security against leakage attacks?
(*ii*) We ignore the impact of the primitives (e.g., the internals of the block ciphers and permutations) on the cost of low-level side-channel countermeasures. Yet, it is for example a well-known fact that minimizing the multiplicative complexity and depth of a block cipher is beneficial for masking [38,44,71].

The first point is an important scope for further research. A complete analysis would indeed require having quantitative (and ideally tight) bounds for all the investigated schemes. For the second one, we believe it should have limited impact since most current lightweight ciphers aimed at security against side-channel attacks (e.g., the NIST Lightweight Cryptography candidates) are based on small S-boxes with minimum AND complexity and depth.

2 Simplifying Framework

In this section, we present the simplifying framework that we will use in order to reason about leakage-resistant AE modes based on concrete (SPA and DPA)

[2] This limitation only holds in the presence of leakage, so nothing prevents to ask for black box misuse-resistance as an additional requirement of a mode, which we do not consider in this paper but could be investigated as future work.

attacks. For this purpose, we first propose an informal decomposition of the modes that we will consider, and then list the design tweaks that such modes can leverage in order to reach different leakage security guarantees.

2.1 Leakage-Resistant AE Modes Decomposition

We decompose the AE modes of operation under investigation into four informal parts, as illustrated in Fig. 1 for a simple Inner Keyed Sponge (IKS) design [16]. An optional Key Generation Function (KGF) is used to generate a fresh key K^* based on a long-term master key K and a nonce N. Next, the message processing part uses the (optionally fresh) key in order to encrypt the message blocks. A Tag Generation Function (TGF) finally uses the result of the message processing part and outputs a tag for message authentication. The tag is only verified (i.e., compared to the genuine tag) in case of decryption.

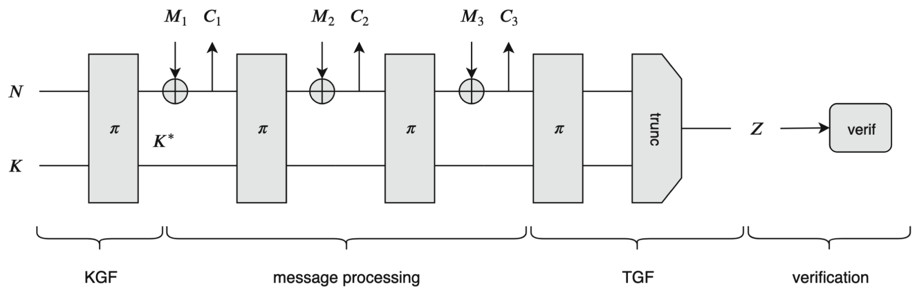

Fig. 1. Exemplary decomposition of an AE scheme.

We note that we make no claim regarding the generality of this decomposition. As will be clear next, it nicely matches a number of recent AE schemes with different levels of security against side-channel attacks, but other modes may not be directly adaptable to this simple framework. We believe this limitation is natural for a work aiming at specializing leakage-resistance analyzes to practical AE schemes. We note also that for simplicity, we ignore the associated data in our discussions, since it has limited impact on leakage analyzes.

2.2 Design Tweaks and Security Levels

The main design tweaks that enhance mode-based side-channel security are:

1. *Key evolution.* As formalized by Dziembowski and Pietrzak, updating the ephemeral keys of an implementation so that each of them is only used – and therefore leaks – minimally can improve confidentiality with leakage [33].
2. *Strengthened KGF and TGF.* As formalized by Berti et al., using key and tag generation functions so that, given their input (resp., output), it is not direct to compute their output (resp., input) can improve security with leakage [14, 15] – for example by preventing that recovering an ephemeral secret during message processing leads to the long-term master key.

3. *Two-pass designs.* As formalized by Guo et al. (resp., Barwell et al.) in the leakage-resistance setting [46] (resp., leakage-resilience setting [6]), two-pass modes can improve message confidentiality with decryption leakages, if the tag verification does not require the manipulation of sensitive plaintexts.

Based on these three main design tweaks, we select a number of practically-relevant security targets that reflect existing AE schemes from the leakage-resistance definition zoo of [46]. For this purpose, we first recall Guo et al.'s definitions of integrity and confidentiality with leakage in an abstracted way, which will be sufficient to guide our attack-based analysis in the next section. Their more formal introduction is additionally provided in the full paper [11].

Definition 1 (Ciphertext Integrity with Leakage [15], Informal.). *In the ciphertext integrity with leakage security game, the adversary can perform a number of queries to encryption and decryption oracles enhanced with leakage functions, that capture the implementation of an AE scheme. His goal is to produce a valid fresh ciphertext and the implementation is considered secure if the adversary cannot succeed with good probability. Variants that we will use next include: ciphertext integrity with (nonce-respecting adversary and) leakage in encryption only (CIL1) and ciphertext integrity with misuse-resistance (i.e., no constraint on nonces) and leakage in encryption and decryption (CIML2).*

Definition 2 (Confidentiality with Leakage [46], Informal.). *In the Chosen Ciphertext Attack (CCA) with leakage security game, the adversary can perform a number of queries to encryption and decryption oracles enhanced with leakage functions, that capture the implementation of an AE scheme. During a so-called "challenge query", he picks up two fresh messages X_0 and X_1 and receives a ciphertext Y_b encrypting X_b for $b \in \{0,1\}$, with the corresponding leakage. His goal is to guess b and the implementation is considered secure if the adversary cannot succeed with good advantage. Variants that we will use next include: chosen ciphertext security with nonce-respecting adversary and leakage in encryption only (CCAL1), chosen ciphertext security with nonce misuse-resilience (i.e., fresh challenge nonce) and leakage in encryption only (CCAmL1) and chosen ciphertext security with nonce misuse-resilience and leakage in encryption and decryption, including decryption of the challenge Y_b (CCAmL2).*[3]

In our notations, small caps are for resilience to misuse or leakage and capital letters for resistance. For integrity guarantees, it is possible to ensure misuse-resistance and leakage-resistance jointly. As discussed in [46], such a combination is believed to be impossible under reasonable leakage models for confidentiality guarantees and one has to choose between Barwell et al.'s CCAMl2 security or Guo et al.'s CCAL1, CCAmL1 or CCAmL2 security (see also Sect. 3.1).

Based on these definitions, we list our security targets and their link with the aforementioned design tweaks. We insist that these links hold for the AE schemes

[3] We focus on the single-challenge definition for simplicity. Multi-challenge extensions are treated in the extended version of [46] – see also the full paper [11].

investigated in the next section. We do not claim they are necessary to reach the security targets and admit other design ideas could be leveraged. We further reckon a finer-grain analysis may be useful in order to analyze other modes.

- **Grade-1a.** CIL1 and CCAL1 security thanks to key evolution.
- **Grade-1b.** CIML2 security thanks to strengthened KGF and TGF (and only black box security guarantees for message confidentiality).
- **Grade-2.** CIML2 and CCAmL1 security thanks to a combination of key evolution (i.e., Grade-1a) and strengthened KGF and TGF (i.e., Grade-1b).
- **Grade-3.** CIML2 and CCAmL2 security thanks to a combination of key evolution and strengthened KGF and TGF with a two-pass design.

We also denote as **Grade-0** the AE schemes without leakage-resistant features.

Definitional framework motivation. The grades and design tweaks that we just described motivate our choice of definitions. On the one hand, the different grades exploit Micali and Reyzin's seminal observation that integrity requirements may be significantly easier to fulfill with leakage than confidentiality requirements. For example, Grade-2 designs achieve stronger integrity guarantees (with decryption leakage) than confidentiality guarantees (without decryption leakage). In Barwell et al.'s all-in-one definition, removing decryption leakage could only be done jointly for confidentiality and integrity guarantees. On the other hand, the security gains of some design tweaks cannot be reflected in the leakage-resilience setting. For example, excluding the challenge leakage implies that an implementation leaking challenge messages (and ephemeral keys) in full is deemed secure according to Barwell et al.'s definition. Hence, ephemeral key evolution has no impact in this case (and the construction in [6] is indeed based on CFB). We insist that this observation does not invalidate the interest of the leakage-resilience setting: whether (stronger) leakage-resistance or (weaker) leakage-resilience is needed depends on application constraints. In general, our only claim is that the definitional framework of Guo et al. enables a fine-grain analysis that can capture various designs and application constraints which are not apparent when using an all-in-one definition of leakage-resilience.

3 From Leakage-Resistance to Side-Channel Security

In this section, we first discuss how the physical assumptions used in leakage security proofs can be translated into minimum requirements for implementers. Next, we illustrate what are these minimum requirements for concrete instances of existing AE schemes. From the current literature, we identify Grade-0, Grade-1a, Grade 2 and Grade-3 AE schemes, which suggests the design of an efficient Grade-1b instance as an interesting scope for further investigations. We show that the minimum requirements suggested by security proofs are (qualitatively) tight and that failing to meet them leads to realistic SPA and DPA attacks.

3.1 Translating Physical Assumptions into Implementation Goals

Leakage security proofs for AE schemes rely on physical assumptions. As mentioned in introduction, the quest for sound and realistic assumptions is still a topic of ongoing research. In this section, we observe that the (sometimes strong) assumptions needed in proofs can be translated into minimum security requirements, expressed in terms of security against practical side-channel attacks.

Integrity with leakage requirements can be limited to the KGF, TGF (and optionally verification functions) for AE schemes with good leakage properties, and are extended to all the components of a mode of operation without such good properties (see Sect. 5 for the details). The simplest assumption is to consider the underlying blocks to be leak-free [68]. A recent relaxation introduces a weaker requirement of unpredictability with leakage [12]. In both cases, these assumptions need that implementations manipulating a long-term key limit the probability of success of key-recovery DPA attacks, which we express as:

$$\Pr\left[\mathcal{A}_{\mathsf{kr}}^{\mathsf{L}(.,.)}\big(X_1, \mathsf{L}(X_1, K), \ldots, X_q, \mathsf{L}(X_q, K)\big) \rightarrow K | K \xleftarrow{\text{u}} \{0,1\}^n\right] \approx 2^{-n+q\cdot\lambda(r)}, \tag{1}$$

where $\mathcal{A}_{\mathsf{kr}}^{\mathsf{L}(.,.)}$ is the key recovery adversary able to make offline calls to the (unkeyed) leakage function $\mathsf{L}(.,.)$, $X_1, \ldots, X_q$ the different inputs for which the primitive is measured, K the long-term key of size n bits and $\lambda(r)$ the (informal) amount of information leakage obtained for a single input X_i measured r times (i.e., repeating the same measurement multiple times can be used to reduce the leakage noise). For security against DPA, it is required that this probability remains small even for large q values, since there is nothing that prevents the adversary to measure the target implementation for many different inputs. Such DPA attacks reduce the secrecy of the long-term key exponentially in q. Hence, preventing them requires a mechanism that counteracts this reduction. For example, masking can be used for this purpose and makes $\lambda(r)$ exponentially small in a security parameter (i.e., the number of masks or shares) [20,32].

Confidentiality with leakage requirements are significantly more challenging to nail down. For the KGF and TGF parts of the implementation, they at least require the same DPA security as required for integrity guarantees. For the message processing part, various solutions exist in the literature:

1. *Only computation leaks assumption and bounded leakage,* introduced by Dziembowski and Pietrzak [33]. By splitting a key in two parts and assuming that they leak independently, it allows maintaining some computational entropy in a key evolution scheme under a (strong) bounded leakage assumption.[4]

[4] Precisely, [33] assumes high min-entropy or high HILL pseudoentropy, which are quite high in the hierarchy of assumptions analyzed by Fuller and Hamlin [36]. Note also that for non-adaptive leakages, the "alternating structure" that splits the key in two parts can be replaced by alternating public randomness [35,86,87].

2. *Oracle-free and hard-to-invert leakage function,* introduced by Yu et al. [87]. The motivation for Dziembowski and Pietrzak's alternating structure is to limit the computational power of the leakage function, which can otherwise compute states of a leaking device before they are even produced in the physical world. A straightforward way to prevent such unrealistic attacks is to assume the underlying primitives of a symmetric construction to behave as random oracles and to prevent the leakage function to make oracle calls. This comes at the cost of an idealized assumption, but can be combined with a minimum requirement of hard-to-invert leakages.[5]
3. *Simulatability,* introduced by Standaert et al., is an attempt to enable standard security proofs with weak and falsifiable physical assumptions, without alternating structure [78]. It assumes the existence of a leakage simulator that can produce fake leakages that are hard to distinguish from the real ones, using the same hardware as used to produce the real leakages but without access to the secret key. The first instances of simulators were broken in [58]. It remains an open problem to propose new (stronger) ones.

Despite technical caveats, all these assumptions aim to capture the same intuition that an ephemeral key manipulated minimally also leaks in a limited manner, preserving some of its secrecy. As a result, they share the minimum requirement that the probability of success of a key-recovery SPA attack remains small. Such a probability of success has a similar expression to the one of Eq. 1, with as only (but paramount) difference that the number of inputs that can be measured is limited by design. Typically, $q = 2$ for leakage-resilient stream ciphers where one input is used to generate a new ephemeral key and the other to generate a key stream to be XORed with the plaintexts [70,78,86,87].

Note that because of these limited q values, the possibility to repeat measurements (by increasing the r of the leakage expression $\lambda(r)$) is an important asset for SPA adversaries. As will be detailed next, this creates some additional challenges when leakages can be obtained with nonce misuse or in decryption.

Besides the aforementioned requirements of security against key-recovery DPA and SPA, definitions of leakage-resistance provide the adversary with the leakage of the challenge query. In this setting, another path against the confidentiality of an AE scheme is to directly target the manipulation of the message blocks (e.g., in Fig. 1, this corresponds to the loading of the M_i blocks and their XOR with the rate of the sponge). Following [80], we express the probability of success of such Message Comparison (MC) attacks with:

$$\Pr\left[\mathcal{A}_{\mathsf{mc}}^{\mathsf{L}(.,.)}\big(X_0, X_1, \mathsf{L}(X_b, K)\big) \rightarrow b | K \overset{\text{u}}{\leftarrow} \{0,1\}^n, b \overset{\text{u}}{\leftarrow} \{0,1\}\right]. \qquad (2)$$

In this case, the adversary has to find out whether the leakage $\mathsf{L}(X_b, K)$ is produced with input X_0 or X_1. As discussed in [46], there are currently no

[5] The hard-to-invert leakage assumption was introduced beforehand [28,29], and is substantially weaker than entropic leakage requirements (see again [36]). For example, suppose $\mathsf{L}(K)$ is the leakage, where K is secret and L is a one-way permutation. Then the leakage is non-invertible, but the conditional entropy of K could be 0.

mode-level solutions enabling to limit this probability of success to negligible values. So implementers have to deal with the goal to minimize the message leakage with lower-level countermeasures. Yet, as will be discussed next, even in this case it is possible to leverage mode-level protection mechanisms, by trying to minimize the manipulation of sensitive messages (e.g., in decryption).

Discussion. We note that combining leakage-resistance with misuse-resistance would require to resist attacks similar to the one of Eq. 2, but with a "State Comparison" (SC) adversary $\mathcal{A}_{\mathsf{sc}}^{\mathsf{L}(.,K)}$ able to make offline calls to a keyed leakage function. As discussed in [46,80], this allows the adversary to win the game by simply comparing $\mathsf{L}(X_0, K)$ and $\mathsf{L}(X_1, K)$ with $\mathsf{L}(X_b, K)$, which is believed to be hard to avoid (unless all parts of the implementations are assumed leak-free). As a result, we next consider a combination of misuse-resilience with leakage-resistance as our strongest target for confidentiality with leakage.

Summary. The heuristic security requirements for the different parts of an AE scheme with leakage (following the decomposition of Sect. 2.1) include:

- **For integrity guarantees:**
  - *For the KGF and TGF:* security against (long-term key) DPA.
  - *For the message processing part:* security against (ephemeral key) DPA or no requirements (i.e., full leakage of ephemeral device states).
  - *For tag verification:* security against DPA or no requirements.
- **For confidentiality guarantees:**
  - *For the KGF and TGF:* security against (long-term key) DPA.
  - *For the message processing part:* security against (ephemeral key) DPA or (ephemeral key) SPA and security against MC attacks.

As detailed next, for some parts of some (leakage-resistant) AE schemes, different levels of physical security are possible, hence enabling leveled implementations. For readability, we will illustrate our discussions with the following color code: blue for the parts of an AE scheme that require DPA security, light (resp., dark) green for the parts of an AE scheme that require SPA security without (resp., with) averaging, light (resp., dark) orange for the parts of an AE scheme that require security against MC attacks without (resp., with) averaging and white for the parts of an AE scheme that tolerate unbounded leakages.[6] We draw the tag verification in light grey when it is not computed (i.e., in encryption).

We note that precisely quantifying the implementation overheads associated with SPA and DPA security is highly implementation-dependent, and therefore beyond the scope of this paper. For example, the number of shares necessary to reach a given security level in software (with limited noise) may be significantly higher than in (noisier) hardware implementations [32]. Yet, in order to provide

[6] For the MC attacks, SPA without averaging is only possible in the single-challenge setting. In case multiple challenges are allowed, all MC-SPA attacks are with averaging. This change is not expected to create significant practical differences when the adversary can anyway use challenge messages with many identical blocks.

the reader with some heuristic rule-of-thumb, we typically assume that preventing SPA implies "small" overheads (i.e., factors from 1 to 5) [85] while preventing DPA implies "significant" overheads (i.e., factors from 5 to > 100) [42,43]. Some exemplary values will be given in our more concrete discussions of Sect. 4. We insist that the only requirement for the following reasoning to be practically-relevant is that enforcing SPA security is significantly cheaper than enforcing DPA security, which is widely supported by the side-channel literature.

In the rest of this section, we illustrate the taxonomy of Sect. 2.2 with existing modes of operation. For each grade, we first exhibit how leakage-resistance (formally analysed in Sect. 5) translates into leveled implementations; we then show that this analysis is qualitatively tight and describe attacks breaking the proposed implementations for stronger security targets than proven in Sect. 5; we finally discuss the results and their applicability to other ciphers.

3.2 Grade-0 Case Study: OCB-Pyjamask

Mode-level guarantees. The OCB mode of operation does not provide mode-level security against leakage. This is due to the use of the same long-term key in all the (T)BC invocations of the mode. As a result, all the (T)BC calls must be strongly protected against DPA. For completeness, a uniformly protected implementation of OCB-Pyjamask is illustrated in the full paper [11].

Proofs' (qualitative) tightness. Not applicable (no leakage-resistance proofs).

Discussion. As mentioned in introduction (cautionary notes) and will be further illustrated in Sect. 3.6, the absence of mode-level leakage-resistance does not prevent a mode of operation to be well protected against side-channel attacks: it only implies that the protections have to be at the primitive, implementation or hardware level. In this respect, the Pyjamask cipher is better suited to masking than (for example) the AES and should allow efficient masked implementations, thanks to its limited multiplicative complexity. A similar comment applies to the NIST lightweight candidates SKINNY-AEAD and SUNDAE-GIFT.

3.3 Grade-1a Case Study: PHOTON-Beetle

Mode-level guarantees. As detailed in Sect. 5.2, PHOTON-Beetle is CCAL1 and CIL1 under physical assumptions that, as discussed in Sect. 3.1, translate into SPA security requirements for its message processing part. Therefore, it can be implemented in a leveled fashion as illustrated in Fig. 2. Note that nonce repetition is prevented in the CCAL1 and CIL1 security games, which explains the light grey and orange color codes (for SPA security without averaging).

Proofs' qualitative tightness. As soon as nonce misuse or decryption leakages are granted to the adversary, the following DPA becomes possible against the message processing part of PHOTON-Beetle: fix the nonce and the ephemeral key K^*; generate several plaintext (or ciphertext) blocks M_1 (or C_1); recover

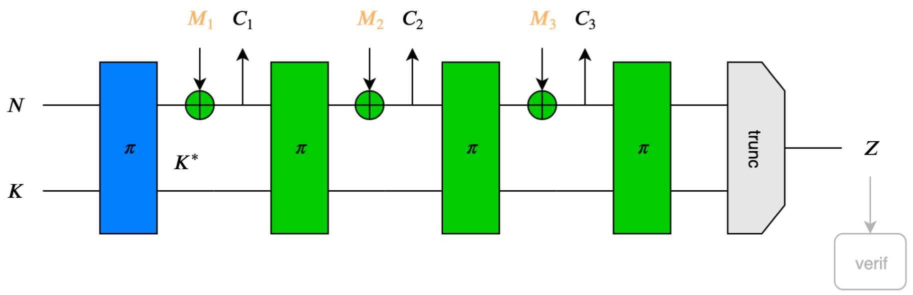

Fig. 2. PHOTON-Beetle, leveled implementation, CCAL1, CIL1.

the capacity part of the state including M_1 (or C_1) and finally inverse the permutation to recover the long-term key K. The number of plaintext/ciphertext blocks that can be measured in this way equals 2^r, where r (i.e., the rate of the sponge) equals 128 in the case of PHOTON-Beetle. This is (considerably) more than needed to perform a successful side-channel attack against a permutation without DPA protections [61]. Hence, we conclude that for any stronger security target than CCAL1 and CIL1, uniform protections are again needed.

Discussion. A similar analysis applies to many IKS designs in the literature, for example the NIST lightweight candidates Gimli and Oribatida and the CAESAR competition candidate Ketje. It formalizes the intuition that when encrypting without misuse, it is not necessary to protect the message processing part of IKS modes as strongly as their KGF. But this guarantee vanishes with nonce misuse or decryption leakage because it is then possible to control the ephemeral keys and the KGF is invertible. Hence, for stronger security targets, lower-level countermeasures have to be uniformly activated, the cost of which again depends on the structure (e.g., multiplicative complexity) of the underlying primitives.

3.4 Grade-2 Case Studies: Ascon and Spook

Mode-level guarantees. As detailed in Sect. 5.3, Ascon and Spook are CCAmL1 and CIML2 under different sets of physical assumptions for confidentiality and integrity guarantees. They represent interesting case studies where the composite nature of Guo et al.'s security definition enables different practical requirements for different parts of a mode and different security targets. We note that the previous requirement of DPA security for the KGF and TGF cannot be relaxed, so it will be maintained in this and the next subsection, without further discussion. By contrast, the security requirements for the message processing and tag verification parts can significantly vary, which we now discuss.

We start with Ascon's CCAmL1 requirements, illustrated in Fig. 3. They translate into SPA security (without averaging) for the message processing part even with nonce misuse.

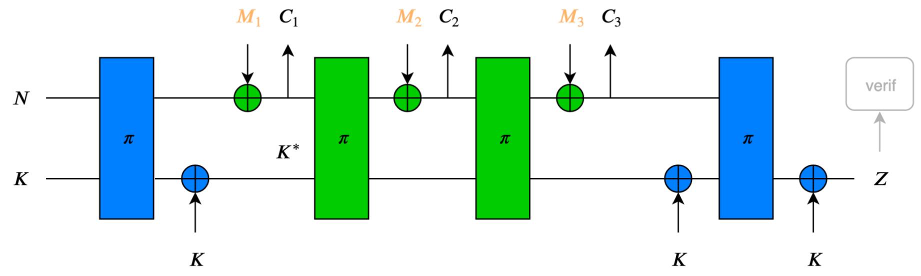

Fig. 3. Ascon, leveled implementation, CCAmL1.

This is because (i) in the misuse-resilience setting, the challenge query of Definition 2 comes with a fresh nonce (i.e., nonce misuse is only granted during non-challenge queries), and (ii) even a full permutation state leakage obtained for non-challenge queries (e.g., thanks to the same DPA as described against PHOTON-Beetle) does not lead to the long-term key K on which confidentiality relies (thanks to the strengthened KGF). A similar situation holds for Spook and is illustrated in the full paper [11].

We follow with Spook's CIML2 requirements, illustrated in Fig. 4. The main observation here is that integrity is proven in a weak (unbounded leakage) model where all the intermediate permutation states are given in full to the adversary. This is possible thanks to the strengthening of the KGF and TGF which prevents any of these ephemeral states to leak about long-term secrets and valid tags. In the case of Spook, even the tag verification can be implemented in such a leaky manner (thanks to the inverse-based verification trick analyzed in [15]). Optionally, a direct tag verification $\mathsf{verif}_\mathsf{B}$ can be used but requires DPA protections. A similar situation holds for the integrity of Ascon and is illustrated in the full paper [11], with as only difference that it can only be implemented with a direct DPA-secure tag verification. Note that without an inverse-based or DPA-secure verification, it is possible to forge valid messages without knowledge of the master key [15], which is for example critical for secure bootloading [67]. We will confirm the practical feasibility of such attacks in Sect. 4.3.

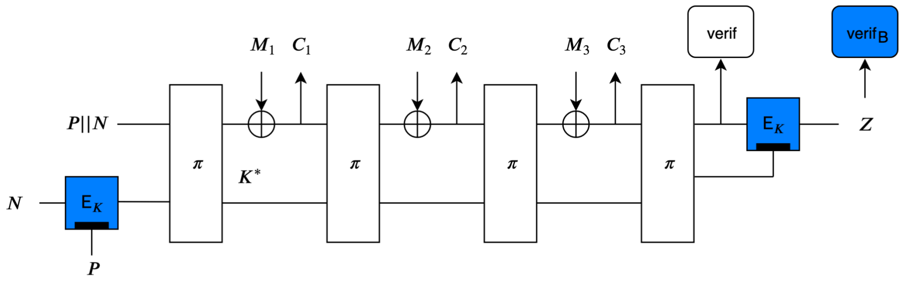

Fig. 4. Spook, leveled implementation, CIML2.

Proofs' qualitative tightness. From Fig. 3 (and 14 in the full paper [11]), it can be observed that as soon as decryption leakages are granted to the adversary, a DPA attack against the confidentiality of the messages becomes possible. The beginning of the attack is exactly the same as the one against PHOTON-Beetle: fix the nonce and the ephemeral key K^*; generate several plaintext (or ciphertext) blocks M_1 (or C_1) and recover the capacity part of the state including M_1 (or C_1). This time, the full state leakage cannot lead to the long-term key K but it still allows recovering all the decrypted messages in full. Note that this attack actually targets a weaker notion than CCAmL2 since it only exploits the decryption leakage, without access to the decryption oracle. Yet, as discussed in [15], it is a quite practical one in case of applications where IP protection matters (e.g., when a code or content can be decrypted and used but not copied).

Discussion. A similar analysis applies to other IKS designs with strengthened KGF and TGF, for example the NIST lightweight candidates ACE, WAGE and SPIX and the CAESAR competition candidate GIBBON. The TBC-based TET scheme also offers the same guarantees [13]. These designs reach the best integrity with leakage but due to their one-pass nature, cannot reach CCAmL2. The main concrete differences between Ascon and Spook are: (i) The KGF and TGF of Ascon are based on a permutation while Spook uses a TBC for this purpose, and (ii) the tag verification of Spook can be implemented in a leaky way with an inverse TBC call or in a DPA-protected way with a direct TBC call, while the tag verification of Ascon can only be implemented in the DPA-protected manner. The pros and cons of these options will be discussed in Sects. 4.2 and 4.3.

3.5 Grade-3 Case Studies: ISAP and TEDT

Mode-level guarantees. Leveled implementations of Ascon and Spook reach the highest security target for integrity with leakage (i.e., CIML2) but they are only CCAmL1 without uniform protections. ISAP and TEDT cross the last mile and their leveled implementations are proven CCAmL2 in Sect. 5.4, while also maintaining CIML2 security. The integrity guarantees of ISAP and TEDT follow the same principles as Ascon and Spook. Therefore, we only provide their CIML2 implementations in the full paper [11], Figures 16 and 17, and next focus on their practical requirements for confidentiality with decryption leakage.

We start with the ISAP design for which a leveled implementation is illustrated in Fig. 5. For now skipping the details of the re-keying function RK which aims at providing "out-of-the-box" DPA security without implementation-level countermeasures such as masking, the main observation is that ISAP is a two-pass design where the tag verification does not require manipulating plaintext blocks. Hence, as long as the KGF, TGF (instantiated with RK) and the default tag verification are secure against DPA, the only attack paths against the confidentiality of the message are a SPA against the message processing part and a MC attack against the manipulation of the plaintext blocks. In both cases, averaging is possible due to the deterministic nature of the decryption.

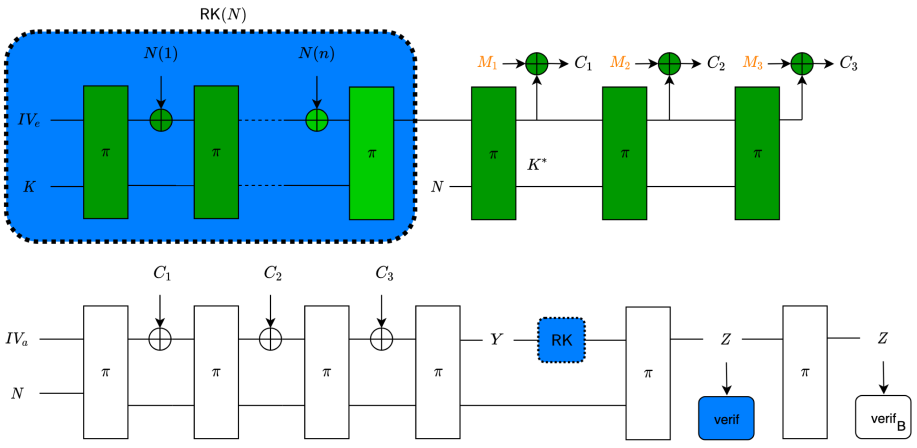

Fig. 5. ISAP, leveled implementation, CCAmL2.

The default tag verification of Fig. 5 must be secure against DPA. An exemplary attack path that becomes possible if this condition is not fulfilled is the following: given a challenge ciphertext (C, Z), flip some bits of C leading to related ciphertexts $C', C'', \ldots$ (which, due to the malleability of the encryption scheme, correspond to messages $M', M'', \ldots$ with single bits flipped compared to the target M); forge valid tags for these ciphertexts thanks to the leaking message comparison (as experimentally validated in Sect. 4.3) and finally perform a DPA against M thanks to the related messages' decryption leakage, which breaks the SPA requirements guaranteed by the proofs in Sect. 5.4 and leads to the same (practical) IP protection issue as mentioned for Ascon and Spook.

Alternatively, ISAP also comes with a tag verification that provides similar guarantees as Spook's inverse one at the cost of another permutation call.

TEDT's CCAmL2 requirements, illustrated in the full paper [11], Fig. 18, are mostly identical: the only difference is that the RK function is replaced by a TBC which must be secure against DPA thanks to masking or other low-level countermeasures, and optionally enables an inverse-based tag verification.

Discussion. TEDTSponge, a sponge-based variant of TEDT with similar guarantees, is proposed in [47]. Besides their DPA resistant tag verifications, the main difference between ISAP and TEDT is their KGF and TGF. The concrete pros and cons of both approaches will be discussed in Sect. 4.4. We also mention that TBC-based constructions allow proofs in the standard model (under the simulatability assumption), which is currently not possible with sponge-based constructions, for which idealized assumptions are frequently used even in black box security proofs. Whether this gap can be bridged (with the simulatability or a weaker physical assumption) is an interesting open problem.

3.6 Summary Table

The practical requirements that implementers must ensure for the different parts of the different modes of operation investigated in this section are summarized in Fig. 6, in function of the security target. It nicely supports the conclusion that security against side-channel attacks can be viewed as a tradeoff between mode-level and implementation-level (or hardware-level) protections.

In general, even the highest security targets (i.e., CCAmL2 and CIML2) can be reached by modes without any leakage-resistance features like OCB, but then require strong low-level countermeasures for all the implementation parts. As the security targets and the quantitative security levels needed by an application increase, it is expected that leveled implementations will lead to better performance figures, which will be further analyzed in the next section.

		CCAL1	CCAmL1	CCAmL2	CIL1	CIML1	CIML2
OCB-Pyjamask	KGF/TGF	DPA	DPA	DPA	DPA	DPA	DPA
	mess. proc	DPA	DPA	DPA	DPA	DPA	DPA
	verif.	NA	NA	unb.	NA	NA	DPA
	MC	SPA	SPA	SPA+avg	NA	NA	NA
PHOTON-Beetle	KGF/TGF	DPA	DPA	DPA	DPA	DPA	DPA
	mess. proc	SPA	DPA	DPA	SPA	DPA	DPA
	verif.	NA	NA	unb.	NA	NA	DPA
	MC.	SPA	SPA	SPA+avg	NA	NA	NA
Ascon	KGF/TGF	DPA	DPA	DPA	DPA	DPA	DPA
	mess. proc	SPA	SPA	DPA	unb.	unb.	unb.
	verif.	NA	NA	unb.	NA	NA	DPA
	MC.	SPA	SPA	SPA+avg	NA	NA	NA
Spook	KGF/TGF	DPA	DPA	DPA	DPA	DPA	DPA
	mess. proc	SPA	SPA	DPA	unb.	unb.	unb.
	verif.	NA	NA	unb.	NA	NA	unb.
	MC.	SPA	SPA	SPA+avg	NA	NA	NA
ISAP	KGF/TGF	DPA	DPA	DPA	DPA	DPA	DPA
	mess. proc	SPA	SPA	SPA+avg	unb.	unb.	unb.
	verif.	NA	NA	DPA	NA	NA	DPA
	MC.	SPA	SPA	SPA+avg	NA	NA	NA
TEDT	KGF/TGF	DPA	DPA	DPA	DPA	DPA	DPA
	mess. proc	SPA	SPA	SPA+avg	unb.	unb.	unb.
	verif.	NA	NA	DPA	NA	NA	unb.
	MC.	SPA	SPA	SPA+avg	NA	NA	NA

Fig. 6. Leveled implementations requirements (NA refers to attacks that cannot be mounted as they need access to leakage that is not available in the game).

4 Design Choices and Concrete Evaluations

The framework of Sect. 2 allowed us to put forward a range of AE schemes with various levels of leakage-resistance in Sect. 3. These modes of operation leverage a combination of design ideas in order to reach their security target. In this section, we analyze concrete questions related to these ideas and, when multiple options are possible, we discuss their pros and cons in order to clarify which one to use in which context. We insist that our goal is not to compare the performances of AE schemes but to better understand their designs. (For unprotected implementations, we refer to ongoing benchmarking initiatives for this purpose. For protected ones, this would require agreeing on security targets and evaluating the security levels that actual implementations provide).

4.1 Uniform vs. Leveled Implementations

Research question. One important pattern shared by several designs analyzed in the previous section is that they can enable leveled implementations where different parts of the designs have different levels of security against side-channel attacks. This raises the question whether and to what extent such leveled implementations can be beneficial. In software, it has already been shown in [13] that gains in cycles can be significant and increase with the level of security and message size. We next question whether the same observation holds in hardware, which is more tricky to analyze since enabling more speed vs. area tradeoffs. Precisely, we investigate whether leveled implementations can imply energy gains (which is in general a good metric to capture a design's efficiency [53]).

Experimental setting. In order to investigate the energy efficiency aspects of leveled implementations in presence of side-channel protections, we have designed FPGA and ASIC leveled implementations of Spook. We applied the masking countermeasure with $d = 2$ and $d = 4$ shares (with the HPC2 masking scheme [19]) to the Clyde 128-bit TBC (used as KGF and TGF in Spook), and no countermeasure to the Shadow 512-bit permutation. We used a 32-bit architecture for Clyde and a 128-bit architecture for Shadow. The FPGA implementations have been synthesized, tested and measured on a Sakura-G board, running on a Xilinx Spartan-6 FPGA at a clock frequency of 6 MHz. As a case study, we encrypted a message composed of one block of authentication data and six blocks of plaintext. ASIC implementations have been designed using *Cadence Genus 18* with a commercial 65 nm technology, and the power consumption has been estimated *post-synthesis*, running at a clock frequency of 333 MHz.

Experimental results. The power consumption versus time of the Spook FPGA implementation is shown in Fig. 7. Its main phases can be recognized: first, the Clyde KGF, then the Shadow executions and finally the Clyde TGF. We observe that a Shadow execution (48 clock cycles) is shorter than a masked Clyde execution (157 clock cycles). The power consumption of Shadow is independent of the masking order d, while the one of Clyde increases with d. The figure intuitively confirms the energy gains that leveled implementations enable. We note that larger architectures for Clyde would not change the picture: latency could be reduced down to 24 cycles (i.e., twice the AND depth of the algorithm) but this would cause significant area and dynamic power consumption increases.

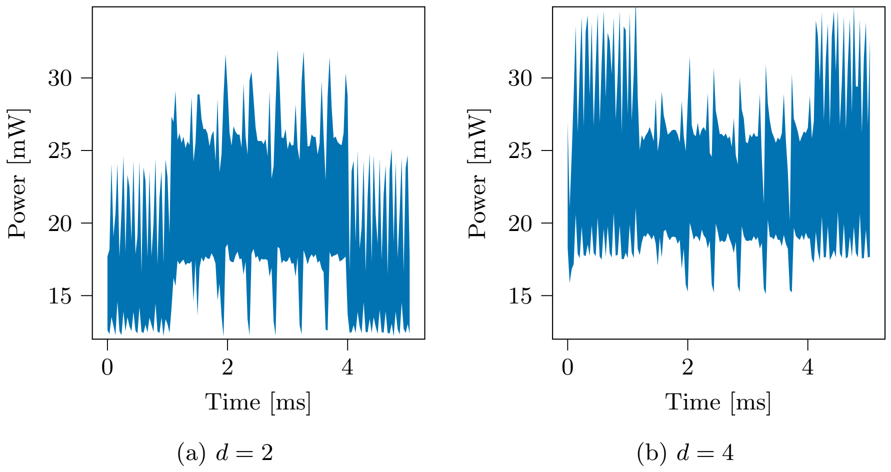

Fig. 7. Power consumption of a leveled implementation of Spook with a masked Clyde on a FPGA (Xilinx Spartan 6) at clock frequency $f_{CLK} = 6\,\text{MHz}$, for the encryption of one block of associated data and 5 blocks of message.

We confirm this intuitive figure with performance results for the ASIC implementations of Spook. For this purpose, we have extracted energy estimations for one execution of Clyde (about 3.4 nJ for $d = 2$ and 8.1 nJ for $d = 4$) and one execution of (unprotected) Shadow (about 1.2 nJ) independently, in order to easily study the contributions of the two primitives. Assuming that only the execution of the primitives consumes a significant amount of energy, we can then estimate the energy consumption per byte for both Spook (i.e., 2 Clyde executions and $n + 1$ Shadow executions where n is the number of 32-byte message blocks) and OCB-Clyde-A (resp., OCB-Clyde-B), assuming $n + 2$ (resp., $n + 1$) Clyde executions (where n is the number of 16-byte message blocks). The OCB mode was used as an example of Grade-0 mode, and we used the Clyde TBC in order to have a fairer comparison between the modes. The A (resp., B) variant models the case where the OCB initialization is not amortized (resp., amortized) over a larger number of encryptions. The estimated energy per byte encrypted on ASIC is shown in Fig. 8. For short messages (at most 16 bytes) and for both masking orders, OCB-Clyde-B consumes the least (with 2 Clyde executions), followed by Spook (2 Clyde and 2 Shadow executions), and OCB-Clyde-A is the most energy-intensive (with 3 Clyde executions). For long messages, both OCB-Clyde-A and -B converge to 1 Clyde execution per 16-byte block, while Spook converges to 1 Shadow execution per 32-byte block. In that scenario, a leveled implementation of Spook is therefore much more energy-efficient than OCB (e.g., 5 times more efficient for $d = 2$, and the difference increases with d).

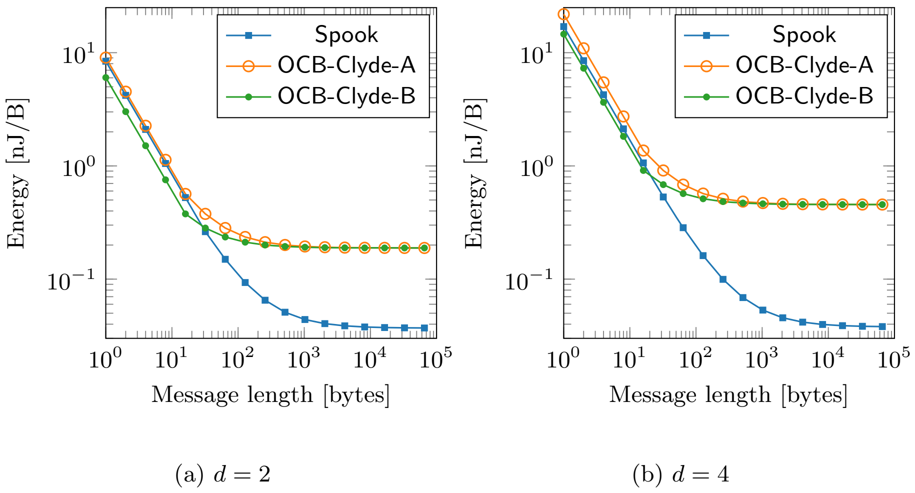

Fig. 8. Energy consumption of leveled Spook and uniform OCB-Clyde implementations on ASIC (65 nm technology), in function of the message length.

Discussion. Both the FPGA measurements and the ASIC synthesis results confirm that leveled implementations can lead to significant energy gains for long messages. This derives from the fact that the energy per byte of a protected primitive is larger than for a non-protected one. More interestingly, even for short messages our results show that leveled implementations can be beneficial. For example, Spook requires only two TBC executions. Hence, it is always more energy-efficient than OCB, excepted when the OCB initialization is perfectly amortized, and the message length is less than 16 bytes. Furthermore, even in this case, the Spook overhead is only 35% for $d = 2$ and 15% for $d = 4$.

These energy gains come at the cost of some area overheads since the leveled nature of the implementations limits the possibility to share resources between their strongly and weakly protected parts. In the case of Spook studied in this section, the total area requirements of a 2-share (resp., 4-share & 8-share) leveled implementation is worth 53,897 (resp., 90,311 & 191,214) μm^2. The Shadow-512 part of the implementation is only 22% of this total for the 2-share implementation and decreases to 13% (resp., 6%) with 4 shares (resp., 8 shares).[7]

4.2 TBC-Based vs. Sponge-Based KGF and TGF

Research question. The Grade-2 designs Ascon and Spook use strengthened KGF and TGF instantiated with a permutation and a TBC. This raises the question whether both approaches are equivalent or if one or the other solution is preferable in some application context. We next answer this question by

[7] All the results in this subsection include the cost of the PRNG used to generate the shares (we used a 128-bit LFSR) and the area costs include the interface.

leveraging recent results analyzing the (software) overheads that masked implementations of the Ascon permutation and Spook TBC imply.

Experimental setting. The Ascon permutation (used for the KGF and TGF) and Spook TBC have conveniently similar features: both are based on a quadratic S-box and both have 12 rounds. Hence, both have the same multiplicative depth and the different number of AND gates that these primitives have to mask (which usually dominates the overheads as soon as the number of shares increases) only depends on their respective sizes: 384 bits for the Ascon permutation and 128 bits for the Spook TBC. We next compare the cycle counts for masked implementations of these two primitives in an ARM Cortex-M4 device.

Experimental results. A work from Eurocrypt 2020 investigates the proposed setting. It uses a tool to automatically generate masked software implementations that are secure in the (conservative) register probing model [7]. The Ascon permutation and Spook TBC (denoted as Clyde) are among the analyzed primitives. As expected, the resulting cycle counts for the full primitive are roughly doubled for Ascon compared to Clyde (reflecting their state sizes). For example, with a fast RNG to generate the masks and $d = 3$ (resp., $d = 7$) shares, the Ascon permutation requires 42,000 (resp., 123,200) cycles and Clyde only 15,380 (resp., 56,480). When using a slower RNG, these figures become 53,600 (resp., 182,800) for the Ascon permutation and 30,080 (resp., 121,280) for Clyde.

Discussion. Assuming similar security against cryptanalysis, Spook's TBC allows reduced overheads for the KGF and TGF by an approximate factor two compared to Ascon's permutation. Since TBCs generally rely on smaller state sizes than the permutations used in sponge designs, we believe this conclusion generally holds qualitatively. That is, a DPA-protected KGF or TGF based on a TBC allows reduced multiplicative complexities compared to a (wider) permutation-based design. It implies performance gains, especially for small messages for which the execution of the KGF & TGF dominates the performance figures. This gain comes at the cost of two different primitives for Spook (which can be mitigated by re-using similar components for these two primitives). Based on the results of Sect. 4.1, we assume a similar conclusion holds in hardware. So overall, we conclude that the TBC-based KGF/TGF should lead to (mild) advantages when high security levels are required while the permutation-based design enjoys the simplicity of a single-primitive mode of operation which should lead to (mild) advantages in the unprotected (or weakly protected) cases.

4.3 Forward vs. Inverse Tag Verification

Research question. Another difference between Ascon and Spook is the possibility to exploit an inverse-based tag verification with unbounded leakage rather than a DPA-protected direct verification. We next question whether protecting the tag verification is needed and describe a DPA against an unprotected tag verification enabling forgeries for this purpose. We then estimate the cost of these two types of protections and discuss their benefits and disadvantages.

Experimental setting. We analyze a simple 32-bit tag verification algorithm implemented in a `Cortex-M0` device running at 48 [MHz], and measured the power side-channel at a sampling frequency of 500 [MSamples/s]. It computes the bitwise XNOR of both tags and the AND of all the resulting bits. The adversary uses multivariate Gaussian templates estimated with $100,000$ profiling traces corresponding to random tag candidates and ciphertexts [21]. During the online phase, she performs a template attack on each byte of the tag individually. To do so, she records traces corresponding to known random tag candidates.

Experimental results. Figure 9 shows the guessing entropy of the correct tag according to the number of attack measurements. It decreases with the number of traces meaning that the attack converges to the correct tag. After the observation of 300 tag candidates, the guessing entropy is already reduced below 2^{32}. The correct tag can then be obtained by performing enumeration (e.g., with [72]).

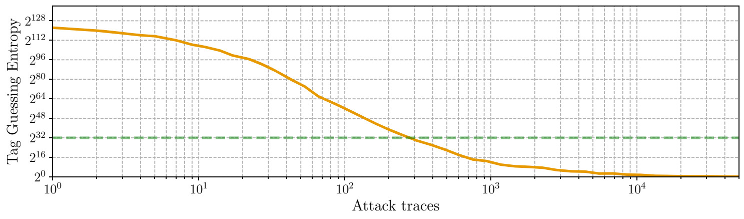

Fig. 9. Security of tag verification on ARM Cortex M0.

Discussion. The DPA of Fig. 9 is slightly more challenging than attacks targeting non-linear operations (e.g., S-box outputs) [73], but still succeeds in low number of traces unless countermeasures are implemented. As discussed in Sect. 3.4, two solutions can be considered to prevent it. First, protecting the tag verification against DPA with masking. Masking the XNOR operations is cheap since it is an affine operation. Masking the AND operations is more costly and implies performing 127 two-input secure AND gates (for a 128-bit tag), with a multiplicative depth of at least 7. The overall cost can be estimated as about 15% of a Clyde execution in cycle count/latency (in software and hardware), and corresponds to 20% in hardware area (for a 32-bit architecture similar to the one of Sect. 4.1). The second method is only applicable to TBC-based TGF. It computes the inverse of the TBC on the candidate tag, allowing a secure comparison with unbounded leakage. Being unprotected, the comparison is cheap but the inverse leads to overheads. For example, for the Clyde TBC, the inverse does not change the execution time, but increases the hardware area by 24% or the software code size by 23%.[8] We conclude that protecting the tag verification leads to limited overheads in front of other implementation costs, with a (mild) simplicity and performance advantage for the inverse-based solution.

[8] https://www.spook.dev/implementations.

We recall that ISAP also comes with a possibility to avoid the DPA-protected tag verification, at the cost of an additional call to its internal permutation. It implies an increase of the execution time (rather than area overheads).

4.4 Masked vs. Deterministic Initialization/Finalization

Research question. One important difference between ISAP and TEDT is the way they instantiate their KGF and TGF. TEDT relies on a TBC that has to resist DPA thanks to masking, for which we can rely on a wide literature. ISAP rather uses a re-keying mechanism which is aimed to provide out-of-the-box DPA security. In this case, the best attack is a SPA with averaging, which is a much less investigated context. We therefore question the security of ISAP against advanced Soft Analytical Side-Channel Attacks (SASCA) [84], and then discuss its pros and cons compared to a masked TBC as used in TEDT.

We insist that the following results do not contradict the security claims of ISAP since the investigated attacks are SPA, not DPA. Yet, they allow putting the strengths and weaknesses of ISAP and TEDT in perspective.

Experimental setting. In order to study the out-of-the-box side-channel security of ISAP, we target its reference implementation where the permutations of Fig. 5 are instantiated with the Keccak-400 permutation. We performed experiments against a `Cortex-M0` device running at 48 [MHz], and measured the power side-channel at a sampling frequency of 500 [MSamples/s]. Even though this target has a 32-bit architecture, the compiler generates code processing 16-bit words at a time. This is a natural approach since it is the size of a lane in Keccak-400. Therefore, the intermediate variables exploited are 16-bit wide. We used 10, 000 averaged traces corresponding to known (random) IV_e and K values for profiling. The profiling method is a linear regression with a 17-element basis corresponding to the 16 bits manipulated by the implementation and a constant [76]. We profiled models for all the intermediate variables produced when computing the permutation. During the attack phase the adversary is provided with a single (possibly averaged) leakage and optionally with IV_e (which is public in ISAP). She performs SASCA following the same (practical) steps as [45], but with 16-bit intermediate targets rather than 8-bit ones. For time & memory efficiency, she only exploits the first round of the full permutation.

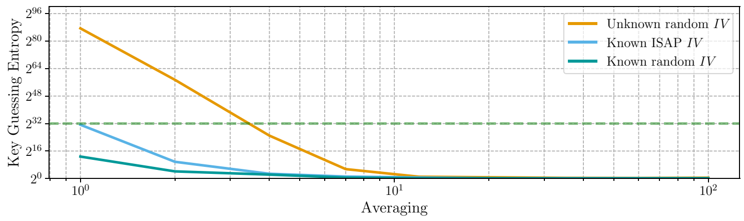

Fig. 10. Security of ISAP's re-keying in an ARM Cortex M0.

Experimental results. Figure 10 shows the guessing entropy of the 128-bit key K in function of the number of times the leakage is averaged. We note that the adversary can average her measurements for the permutations of ISAP's re-keying, since the first permutation has a fixed input and the next ones only integrate the nonce bit per bit (so for example, even without controlling the nonce, 50% of the second permutation calls are identical). Increased averaging allows reducing the noise and therefore reducing the key's guessing entropy.

Overall, if IV_e is known (as in the ISAP design) the guessing entropy of the key is already lower than 2^{32} without averaging. The correct key can then be retrieved by performing key enumeration. Interestingly, we observe that the value of IV_e has some impact on the attack success (i.e., the ISAP value, which has a lot of zeros, leads to slightly more challenging attacks than a random value). We further analyzed the unknown IV_e case and could successfully perform the same attack with a slight averaging (i.e., the attack trace measured ten times). The difference with the known IV_e case derives from the additional efforts that the adversary has to pay for dealing with more secret intermediate states.

Discussion. One important difference between the ISAP and TEDT approaches relates to the presence of a security parameter. While masking a TBC can use a number of shares as security parameter, there is no such security parameter for ISAP if used out-of-the-box. In this respect, the choice between one or the other approach can be viewed as a tradefoff between simplicity and expertise. On the one hand, implementations of ISAP deployed without specific countermeasures already enjoy security against a wide-class of (DPA) attacks; on the other hand, the deterministic nature of the (out-of-the-box implementation of the) re-keying function makes it susceptible to advanced (SPA) attacks that randomized countermeasures such as masking can prevent for TEDT implementations.

Admittedly, this conclusion is in part implementation-specific and the efficiency of SASCA generally degrades with the size of the implementation. In this respect, targeting 32-bit operations would be more challenging, which is an interesting scope for further investigations. Yet, we note that in case the size of the architecture makes power analysis attacks difficult, advanced measurement capabilities may be used [83], maintaining the intuition that for high-security levels, some algorithmic amplification of the noise is generally needed.

We mention that our results are in line with the recent investigations in [51] where similar attacks are reported. The authors conclude that "unprotected implementations should always be avoided in applications sensitive to side-channel attacks, at least for software targets", and suggest (low-order) masking and shuffling as potential solutions to prevent SPA. The concrete investigation of these minimum protections and their implementation cost is an interesting open problem given the difficulty to protect embedded software implementations [18].

A second important difference is that the performance overheads of ISAP are primitive-based while they are implementation-based for the TBC used in TEDT, which can therefore be masked in function of application constraints.

Overall, we conclude that in their basic settings, ISAP and TEDT target different goals: reduction from DPA security to SPA security with primitive-based performance oveaheads for ISAP and high-security against advanced adversaries with flexible overheads (e.g., if side-channel security is not needed) for TEDT.

5 Formal Qualitative Analysis

We conclude the paper with the security analysis of Beetle, Spook, Ascon, TEDT and ISAP, in the ideal permutation model. All the theorems below highlight that integrity requires weaker assumptions than confidentiality even in attack models where the adversary gets more leakage and nonce-misuse capabilities. The reader can find the formal definitions of the security notions in the full paper [11].

5.1 Background: Definitions and Assumptions

For leaking components, we follow [47] and enforce limitations on the leakages of the permutation calls as well as those of the XOR executions. Precisely:

– For the former, we define $(\mathsf{L}^{in}(U), \mathsf{L}^{out}(V))$ as the leakages of a permutation call $\pi(U) \rightarrow V$, where both L^{in} and L^{out} are probabilistic functions. Note that this means the leakage of a single permutation call is viewed as two independent "input" and "output" halves. And we assume the following non-invertibility: *given the leakages* $(\mathsf{L}^{out}(Y\|X), \mathsf{L}^{in}(Y'\|X))$ *of a secret c-bit value X and two adversarially chosen r-bit values Y, Y', the probability to guess X is negligible.* Note that this is a special case of Eq. (1).
– For the XOR executions, we define $\mathsf{L}_{\oplus}(Y, M)$ as the leakage of an XOR computation $Y \oplus M \rightarrow C$, where $\mathsf{L}_{\oplus}$ is also a probabilistic function. This time, we make an assumption on the following message distinguishing advantage: *given the leakages* $(\mathsf{L}^{out}(Y\|X), \mathsf{L}_{\oplus}(Y, M^b))$ *of a secret r-bit key Y and adversarially chosen r-bit values X, M^0, M^1, the probability to guess b is bounded to ε.* Note that this assumption is a special case of Eq. (2).

5.2 CCAL1 and CIL1 Security of PHOTON-Beetle

We first consider Grade-1a schemes and focus on PHOTON-Beetle. As mentioned in Sect. 3.3, the leakage properties of other IKS schemes are similar.

Theorem 1. *Assuming that a PHOTON-Beetle implementation satisfies (i) its KGF is leak-free, and (ii) the leakage of unprotected permutation calls are non-invertible as assumed in Sect. 5.1, the circuit ensure CIL1 integrity. Moreover, if this implementation also satisfies (iii) the leakages of XOR executions are bounded as assumed in Sect. 5.1, the circuit ensures CCAL1 confidentiality.*

The proof sketch is given in the full version of the paper [11].

5.3 CCAmL1 and CIML2 Security of Ascon/Spook

As the mode of Spook is analyzed in [47], we only focus on Ascon.

Theorem 2. *Assuming that an* Ascon *implementation satisfies (i) its KGF is leak-free, and (ii) the tag verification process is leak-free, the circuit ensures* CIML2 *integrity. Moreover, if the* Ascon *implementation also satisfies (iii) the leakages of unprotected permutation calls are non-invertible as assumed in Sect. 5.1, and (iv) the leakages of XOR executions are bounded as assumed in Sect. 5.1, then the* Ascon *implementation ensures* CCAmL1 *confidentiality.*

The proof sketch is given in the full version of the paper [11].

5.4 CCAmL2 and CIML2 Security of ISAP/TEDT

We finally consider 2-pass Encrypt-then-MAC designs. TEDT has been thoroughly analyzed in [13]. Hence we focus on (the 2.0 version of) ISAP.

Theorem 3. *Assuming that an ISAP implementation satisfies (i) its KGF is leak-free, and (ii) the tag verification process is leak-free, the circuit ensures* CIML2 *integrity. Moreover, if the ISAP implementation also satisfies (iii) the leakages of unprotected permutation calls are non-invertible as assumed in Sect. 5.1, and (iv) the leakages of XOR executions are bounded as assumed in Sect. 5.1, then the ISAP implementation ensures* CCAmL2 *confidentiality.*

The proof sketch is given in the full version of the paper [11].

6 Conclusion and Open Problems

The research in this work underlines that there is no single "right definition" of leakage-resistant AE. As the security targets (e.g., the grades of the designs we investigated) and the security levels required by an application increase, it becomes more interesting to exploit schemes that allow minimizing the implementation overheads of side-channel countermeasures. This observation suggests the connection of actual security targets with relevant application scenarios and the performance evaluation of different AE schemes to reach the same physical security levels as a natural next step of this study. Looking for security targets that are not captured by our taxonomy and improving existing designs to reach various targets more efficiency are other meaningful goals to investigate.

Acknowledgment. Gaëtan Cassiers, Thomas Peters and François-Xavier Standaert are respectively PhD student, postdoctoral researcher and senior research associate of the Belgian Fund for Scientific Research (F.R.S.-FNRS). Chun Guo was partly supported by the Program of Qilu Young Scholars (Grant Number 61580089963177) of Shandong University, the National Natural Science Foundation of China (Grant Number 61602276), and the Shandong Nature Science Foundation of China (Grant Number ZR2016FM22). This work has been funded in parts by the ERC project SWORD (Grant Number 724725), the Win2Wal project PIRATE and the UCLouvain ARC project NANOSEC.

References

1. Abdalla, M., Belaïd, S., Fouque, P.-A.: Leakage-resilient symmetric encryption via re-keying. In: Bertoni, G., Coron, J.-S. (eds.) CHES 2013. LNCS, vol. 8086, pp. 471–488. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40349-1_27
2. Ashur, T., Dunkelman, O., Luykx, A.: Boosting authenticated encryption robustness with minimal modifications. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10403, pp. 3–33. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63697-9_1
3. Balasch, J., Gierlichs, B., Grosso, V., Reparaz, O., Standaert, F.-X.: On the cost of lazy engineering for masked software implementations. In: Joye, M., Moradi, A. (eds.) CARDIS 2014. LNCS, vol. 8968, pp. 64–81. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-16763-3_5
4. Bao, Z., et al.: PHOTON-Beetle. Submission to the NIST Lightweight Cryptography Standardization Effort (2019)
5. Barthe, G., et al.: Strong non-interference and type-directed higher-order masking. In: ACM CCS, pp. 116–129. ACM (2016)
6. Barwell, G., Martin, D.P., Oswald, E., Stam, M.: Authenticated encryption in the face of protocol and side channel leakage. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10624, pp. 693–723. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_24
7. Belaïd, S., Dagand, P.É., Mercadier, D., Rivain, M., Wintersdorff, R.: Tornado: automatic generation of probing-secure masked bitsliced implementations. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12107, pp. 311–341. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45727-3_11
8. Belaïd, S., Grosso, V., Standaert, F.-X.: Masking and leakage-resilient primitives: one, the other(s) or both? Crypt. Commun. **7**(1), 163–184 (2014). https://doi.org/10.1007/s12095-014-0113-6
9. Belaïd, S., et al.: Towards fresh re-keying with leakage-resilient PRFs: cipher design principles and analysis. J. Cryptographic Eng. **4**(3), 157–171 (2014). https://doi.org/10.1007/s13389-014-0079-5
10. Bellizia, D., et al.: Spook: sponge-based leakage-resistant authenticated encryption with a masked tweakable block cipher. Submission to the NIST Lightweight Cryptography Standardization Effort (2019)
11. Bellizia, D., et al.: Mode-Level vs. implementation-level physical security in symmetric cryptography: a practical guide through the leakage-resistance jungle. IACR Cryptol. ePrint Arch., 2020:211 (2020)
12. Berti, F., Guo, C., Pereira, O., Peters, T., Standaert, F.-X.: Strong authenticity with leakage under weak and falsifiable physical assumptions. In: Liu, Z., Yung, M. (eds.) Inscrypt 2019. LNCS, vol. 12020, pp. 517–532. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-42921-8_31
13. Berti, F., Guo, C., Pereira, O., Peters, T., Standaert, F.: TEDT, a leakage-resist AEAD mode for high physical security applications. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2020**(1), 256–320 (2020)
14. Berti, F., Koeune, F., Pereira, O., Peters, T., Standaert, F.: Ciphertext integrity with misuse and leakage: definition and efficient constructions with symmetric primitives. In: AsiaCCS, pp. 37–50. ACM (2018)
15. Berti, F., Pereira, O., Peters, T., Standaert, F.: On leakage-resilient authenticated encryption with decryption leakages. IACR Trans. Symmetric Cryptol. **2017**(3), 271–293 (2017)

16. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Duplexing the sponge: single-pass authenticated encryption and other applications. In: Miri, A., Vaudenay, S. (eds.) SAC 2011. LNCS, vol. 7118, pp. 320–337. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28496-0_19
17. Boyle, E., Goldwasser, S., Jain, A., Kalai, Y.T.: Multiparty computation secure against continual memory leakage. In: STOC, pp. 1235–1254. ACM (2012)
18. Bronchain, O., Standaert, F.: Side-channel countermeasures' dissection and the limits of closed source security evaluations. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2020**(2), 1–25 (2020)
19. Cassiers, G., Grégoire, B., Levi, I., Standaert, F.: Hardware Private Circuits: From Trivial Composition to Full Verification (aka Repairing Glitch-Resistant Higher-Order Masking). IACR ePrint Archive (2020)
20. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 398–412. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_26
21. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Kaliski, B.S., Koç, K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36400-5_3
22. Clavier, C., Coron, J.-S., Dabbous, N.: Differential power analysis in the presence of hardware countermeasures. In: Koç, Ç.K., Paar, C. (eds.) CHES 2000. LNCS, vol. 1965, pp. 252–263. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44499-8_20
23. Coron, J.-S., Giraud, C., Prouff, E., Renner, S., Rivain, M., Vadnala, P.K.: Conversion of security proofs from one leakage model to another: a new issue. In: Schindler, W., Huss, S.A. (eds.) COSADE 2012. LNCS, vol. 7275, pp. 69–81. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29912-4_6
24. Degabriele, J.P., Janson, C., Struck, P.: Sponges resist leakage: the case of authenticated encryption. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019. LNCS, vol. 11922, pp. 209–240. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34621-8_8
25. Dobraunig, C., Eichlseder, M., Mangard, S., Mennink, F.M.B., Primas, R., Unterluggauer, T.: ISAP v2.0. Submission to the NIST Lightweight Cryptography Standardization Effort (2019)
26. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon v1.2. Submission to the NIST Lightweight Cryptography Standardization Effort (2019)
27. Dobraunig, C., Mennink, B.: Leakage resilience of the duplex construction. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019. LNCS, vol. 11923, pp. 225–255. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34618-8_8
28. Dodis, Y., Goldwasser, S., Tauman Kalai, Y., Peikert, C., Vaikuntanathan, V.: Public-key encryption schemes with auxiliary inputs. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 361–381. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11799-2_22
29. Dodis, Y., Kalai, Y.T., Lovett, S.: On cryptography with auxiliary input. In: STOC, pp. 621–630. ACM (2009)
30. Dodis, Y., Pietrzak, K.: Leakage-resilient pseudorandom functions and side-channel attacks on feistel networks. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 21–40. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_2

31. Duc, A., Dziembowski, S., Faust, S.: Unifying leakage models: from probing attacks to noisy leakage. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 423–440. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_24
32. Duc, A., Faust, S., Standaert, F.-X.: Making masking security proofs concrete. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 401–429. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_16
33. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: FOCS, pp. 293–302. IEEE Computer Society (2008)
34. Faust, S., Kiltz, E., Pietrzak, K., Rothblum, G.N.: Leakage-resilient signatures. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 343–360. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11799-2_21
35. Faust, S., Pietrzak, K., Schipper, J.: Practical leakage-resilient symmetric cryptography. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 213–232. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33027-8_13
36. Fuller, B., Hamlin, A.: Unifying leakage classes: simulatable leakage and pseudoentropy. In: Lehmann, A., Wolf, S. (eds.) ICITS 2015. LNCS, vol. 9063, pp. 69–86. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-17470-9_5
37. Gammel, B., Fischer, W., Mangard, S.: Generating a session key for authentication and secure data transfer. US Patent 8,861,722 (2014)
38. Gérard, B., Grosso, V., Naya-Plasencia, M., Standaert, F.-X.: Block ciphers that are easier to mask: how far can we go? In: Bertoni, G., Coron, J.-S. (eds.) CHES 2013. LNCS, vol. 8086, pp. 383–399. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40349-1_22
39. Goldwasser, S., Rothblum, G.N.: Securing computation against continuous leakage. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 59–79. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_4
40. Goubin, L., Patarin, J.: DES and differential power analysis the "Duplication" method. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 158–172. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48059-5_15
41. Goudarzi, D. et al.: Pyjamask v1.0. Submission to the NIST Lightweight Cryptography Standardization Effort (2019)
42. Goudarzi, D., Rivain, M.: How fast can higher-order masking be in software? In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10210, pp. 567–597. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56620-7_20
43. Gross, H., Mangard, S., Korak, T.: An efficient side-channel protected AES implementation with arbitrary protection order. In: Handschuh, H. (ed.) CT-RSA 2017. LNCS, vol. 10159, pp. 95–112. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-52153-4_6
44. Grosso, V., Leurent, G., Standaert, F.-X., Varıcı, K.: LS-designs: bitslice encryption for efficient masked software implementations. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 18–37. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46706-0_2
45. Grosso, V., Standaert, F.-X.: ASCA, SASCA and DPA with enumeration: which one beats the other and when? In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9453, pp. 291–312. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48800-3_12

46. Guo, C., Pereira, O., Peters, T., Standaert, F.-X.: Authenticated encryption with nonce misuse and physical leakage: definitions, separation results and first construction. In: Schwabe, P., Thériault, N. (eds.) LATINCRYPT 2019. LNCS, vol. 11774, pp. 150–172. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30530-7_8
47. Guo, C., Pereira, O., Peters, T., Standaert, F.: Towards low-energy leakage-resistant authenticated encryption from the duplex sponge construction. IACR Trans. Symmetric Cryptol. **2020**(1), 6–42 (2020)
48. Herbst, C., Oswald, E., Mangard, S.: An AES smart card implementation resistant to power analysis attacks. In: Zhou, J., Yung, M., Bao, F. (eds.) ACNS 2006. LNCS, vol. 3989, pp. 239–252. Springer, Heidelberg (2006). https://doi.org/10.1007/11767480_16
49. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: securing hardware against probing attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_27
50. Kalai, Y.T., Reyzin, L.: A survey of leakage-resilient cryptography. In: Providing Sound Foundations for Cryptography, pp. 727–794. ACM (2019)
51. Kannwischer, M.J., Pessl, P., Primas, R.: Single-trace attacks on Keccak. IACR Cryptol. ePrint Arch. 2020:371 (2020)
52. Katz, J., Vaikuntanathan, V.: Signature schemes with bounded leakage resilience. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 703–720. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10366-7_41
53. Kerckhof, S., Durvaux, F., Hocquet, C., Bol, D., Standaert, F.-X.: Towards green cryptography: a comparison of lightweight ciphers from the energy viewpoint. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 390–407. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33027-8_23
54. Kiltz, E., Pietrzak, K.: Leakage resilient elgamal encryption. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 595–612. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17373-8_34
55. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-68697-5_9
56. Kocher, P.C.: Leak-resistant cryptographic indexed key update. US Patent 6,539,092 (2003)
57. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_25
58. Longo, J., Martin, D.P., Oswald, E., Page, D., Stam, M., Tunstall, M.J.: Simulatable leakage: analysis, pitfalls, and new constructions. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 223–242. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45611-8_12
59. Malkin, T., Teranishi, I., Vahlis, Y., Yung, M.: Signatures resilient to continual leakage on memory and computation. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 89–106. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19571-6_7
60. Mangard, S.: Hardware countermeasures against DPA – a statistical analysis of their effectiveness. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 222–235. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24660-2_18

61. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks - Revealing the Secrets of Smart Cards. Springer, New York (2007). https://doi.org/10.1007/978-0-387-38162-6
62. Mangard, S., Popp, T., Gammel, B.M.: Side-channel leakage of masked CMOS gates. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 351–365. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30574-3_24
63. Medwed, M., Standaert, F.-X., Großschädl, J., Regazzoni, F.: Fresh re-keying: security against side-channel and fault attacks for low-cost devices. In: Bernstein, D.J., Lange, T. (eds.) AFRICACRYPT 2010. LNCS, vol. 6055, pp. 279–296. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12678-9_17
64. Micali, S., Reyzin, L.: Physically observable cryptography (extended abstract). In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 278–296. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24638-1_16
65. Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 18–35. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03356-8_2
66. Nikova, S., Rijmen, V., Schläffer, M.: Secure hardware implementation of nonlinear functions in the presence of glitches. J. Cryptol. **24**(2), 292–321 (2011)
67. O'Flynn, C., Chen, Z.D.: Side channel power analysis of an AES-256 bootloader. In: CCECE, pp. 750–755. IEEE (2015)
68. Pereira, O., Standaert, F., Vivek, S.: Leakage-resilient authentication and encryption from symmetric cryptographic primitives. In: ACM CCS, pp. 96–108. ACM (2015)
69. Petit, C., Standaert, F., Pereira, O., Malkin, T., Yung, M.: A block cipher based pseudo random number generator secure against side-channel key recovery. In: AsiaCCS, pp. 56–65. ACM (2008)
70. Pietrzak, K.: A leakage-resilient mode of operation. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 462–482. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01001-9_27
71. Piret, G., Roche, T., Carlet, C.: PICARO – a block cipher allowing efficient higher-order side-channel resistance. In: Bao, F., Samarati, P., Zhou, J. (eds.) ACNS 2012. LNCS, vol. 7341, pp. 311–328. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31284-7_19
72. Poussier, R., Standaert, F.-X., Grosso, V.: Simple key enumeration (and rank estimation) using histograms: an integrated approach. In: Gierlichs, B., Poschmann, A.Y. (eds.) CHES 2016. LNCS, vol. 9813, pp. 61–81. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53140-2_4
73. Prouff, E.: DPA attacks and S-boxes. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 424–441. Springer, Heidelberg (2005). https://doi.org/10.1007/11502760_29
74. Rogaway, P., Bellare, M., Black, J.: OCB: a block cipher mode of operation for efficient authenticated encryption. ACM Trans. Inf. Syst. Secur. **6**(3), 365–403 (2003)
75. Rogaway, P., Shrimpton, T.: A provable-security treatment of the key-wrap problem. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 373–390. Springer, Heidelberg (2006). https://doi.org/10.1007/11761679_23
76. Schindler, W., Lemke, K., Paar, C.: A stochastic model for differential side channel cryptanalysis. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 30–46. Springer, Heidelberg (2005). https://doi.org/10.1007/11545262_3

77. Standaert, F.-X.: Towards fair and efficient evaluations of leaking cryptographic devices - overview of the ERC project CRASH, Part I (invited talk). In: Carlet, C., Hasan, M.A., Saraswat, V. (eds.) SPACE 2016. LNCS, vol. 10076, pp. 353–362. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49445-6_20
78. Standaert, F.-X., Pereira, O., Yu, Yu.: Leakage-resilient symmetric cryptography under empirically verifiable assumptions. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 335–352. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_19
79. Standaert, F.X., Pereira, O., Yu, Y., Quisquater, J.J., Yung, M., Oswald, E.: Leakage resilient cryptography in practice. In: Sadeghi, A.R., Naccache, D. (eds.) Towards Hardware-Intrinsic Security. Information Security and Cryptography. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14452-3_5
80. Standaert, F.-X.: Towards and open approach to secure cryptographic implementations (invited talk). In: EUROCRYPT I. LNCS, vol. 11476, p. xv (2019). https://www.youtube.com/watch?v=KdhrsuJT1sE
81. Tiri, K., Verbauwhede, I.: Securing encryption algorithms against DPA at the logic level: next generation smart card technology. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 125–136. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45238-6_11
82. Tiri, K., Verbauwhede, I.: A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation. In: DATE, pp. 246–251. IEEE Computer Society (2004)
83. Unterstein, F., Heyszl, J., De Santis, F., Specht, R., Sigl, G.: High-resolution EM attacks against leakage-resilient PRFs explained. In: Smart, N.P. (ed.) CT-RSA 2018. LNCS, vol. 10808, pp. 413–434. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-76953-0_22
84. Veyrat-Charvillon, N., Gérard, B., Standaert, F.-X.: Soft analytical side-channel attacks. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 282–296. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45611-8_15
85. Veyrat-Charvillon, N., Medwed, M., Kerckhof, S., Standaert, F.-X.: Shuffling against side-channel attacks: a comprehensive study with cautionary note. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 740–757. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34961-4_44
86. Yu, Yu., Standaert, F.-X.: Practical leakage-resilient pseudorandom objects with minimum public randomness. In: Dawson, E. (ed.) CT-RSA 2013. LNCS, vol. 7779, pp. 223–238. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36095-4_15
87. Yu, Y., Standaert, F., Pereira, O., Yung, M.: Practical leakage-resilient pseudorandom generators. In: ACM CCS, pp. 141–151. ACM (2010)