IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2021, No. 3, pp. 641–676. DOI:10.46586/tches.v2021.i3.641-676

Improved Leakage-Resistant Authenticated Encryption based on Hardware AES Coprocessors

Olivier Bronchain, Charles Momin,
Thomas Peters and François-Xavier Standaert

Crypto Group, ICTEAM Institute, UCLouvain, Louvain-la-Neuve, Belgium.
{firstname.lastname}@uclouvain.be

Abstract. We revisit Unterstein et al.'s leakage-resilient authenticated encryption scheme from CHES 2020. Its main goal is to enable secure software updates by leveraging unprotected (e.g., AES, SHA256) coprocessors available on low-end microcontrollers. We show that the design of this scheme ignores an important attack vector that can significantly reduce its security claims, and that the evaluation of its leakage-resilient PRF is quite sensitive to minor variations of its measurements, which can easily lead to security overstatements. We then describe and analyze a new mode of operation for which we propose more conservative security parameters and show that it competes with the CHES 2020 one in terms of performances. As an additional bonus, our solution relies only on AES-128 coprocessors, and it halves the amount of key material needed in order to encrypt and authenticate.

Keywords: Leakage-Resilient Cryptography · Authenticated Encryption · Secure Software Updates · Ciphertext Integrity · ARM Cortex · Differental Power Analysis

1 Introduction

In a recent work from CHES 2020, Unterstein et al. propose a leakage-resilient Authenticated Encryption (AE) mode of operation that is targeted for implementation in low-end microcontrollers empowered with hardware (e.g., AES, SHA256) coprocessors [USS+20]. Their main target application is secure firmware update. The ingredients they use for this purpose are the leakage-resilient Pseudo-Random Function (PRF) of Medwed et al. for generating an ephemeral key and the tag [MSJ12], and a mode of operation adapted from Degabriele et al. to process the message [DJS19, KS20]. In this paper, we first exhibit some limitations of this work, both in the proposed design and in its security evaluation. We then describe mitigations that we analyze theoretically and validate experimentally.

The first limitation of [USS+20] relates to its motivation of enabling secure firmware updates by leveraging hardware coprocessors rather than countermeasures like masking. While this appears as a strong motivation given the challenge of implementing masking securely on low-end devices [BS20], we observe that the main security property that is required for this purpose is ciphertext integrity with leakage in decryption. As discussed in [BPPS17], this property can be reached with two calls to a strongly protected block cipher and letting most of the other parts of the implementation leak in an unbounded manner. Yet, it requires a careful instantiation of the tag verification. Namely, either this tag verification is protected at the implementation level (e.g., via masking), which contradicts the goal of relying only on hardware coprocessors, or it has to be designed such that leaking the verified value cannot lead to forgeries. It is shown in [BPPS17] that an inverse-based verification can remain secure with unbounded leakages. The variant

Received: 2020-10-15 Accepted: 2020-12-15 Published: 2021-07-09

in [DEM+20] achieves a similar result without inverse, but based on a slightly stronger assumption, which translates into security against Simple Power Analysis (SPA).

By contrast, the proposal of Unterstein et al. does not include such a careful design. Hence, we first show that a standard Differential Power Analysis (DPA) against its tag verification can lead to forgeries, completely breaking the integrity of a leaking software update using this scheme. We note that such an attack was already exhibited in [BBC+20], Section 4.3, for an ARM Cortex-M0. We repeat it here for slightly more noisy targets embedding hardware coprocessors. We next propose an adaptation of the tricks in [BPPS17, DEM+20] that we incorporate into an improved mode as a remedy.

A second limitation of [USS+20] relates to its security analysis of the leakage-resilient PRF used for ephemeral key and tag generation. The security of this construction is based on the use of "carefully chosen plaintexts" which are aimed to generate key-dependent algorithmic noise. As mentioned in [MSJ12], this (heuristic) security relies on parallelism and on the fact that (*i*) exploiting leakage after the AES MixColumns operation is difficult (e.g., because only transition-based leakages that are intensive to guess provide significant leakage), and (*ii*) the leakage models of different S-boxes are sufficiently similar.

We revisit these requirements by investigating two hardware coprocessors available on ARM Cortex-M33 and M4 devices, with respectively 32-bit and 128-bit architectures, that are similar (yet not identical) to the ones analyzed by Unterstein et al. We observe that while the leakage of the AES MixColumns operation is indeed difficult to exploit, a profiling using a linear subspace and up to $4 \cdot 10^9$ traces allow us to exploit the different leakage models of different S-boxes, leading to significant reductions of the data complexities reported at CHES 2020. Since the complexity of these attacks directly impacts the number of bits that can be "absorbed" per block cipher call in the tree-based PRF of Medwed et al., these results imply important updates on the efficiency of such constructions. More conceptually, these investigations put forward that evaluating leakage-resilient PRFs is a quite delicate task. That is, in contrast with standard countermeasures like masking and shuffling, which mostly depend on a sufficiently high level of noise in the measurements (that may be uneasy to obtain on low-end devices, but is at least easy to evaluate), leakage-resilient PRFs enable attacks with averaged observations so that they rather require a sufficiently low side-channel signal in the measurements. Our experiments show that evaluating such a signal is a quite sensitive process and that slight changes in the measurement setup or profiling methods can significantly modify the perceived level of security of such primitives. Hence, they suggest that considering sufficient security margins is needed in order to mitigate risks of overstated security. We insist that we do not claim leakage-resilient PRFs cannot be implemented securely: our only claim is that their security evaluation has been less thoroughly studied than for countermeasures like masking. So while such PRFs remain interesting candidates for secure implementation in low-end devices with hardware coprocessors, our results suggest a deeper understanding of their security assumptions is needed, which is an interesting research direction.

Based on these security analyses and keeping the motivation of providing efficient and secure firmware updates, a third observation is that the AE mode in [USS+20] requires two-passes in decryption. While the use of two passes is needed to provide strong confidentiality guarantees in the presence of decryption leakage (as for example witnessed by the TEDT mode of operation [BGP+20]), optimal integrity guarantees in the presence of leakage can already be obtained with a single-pass and online mode, as reflected by the so-called Grade-2 designs in [BBC+20]. We therefore provide a new mode of operation that is exclusively based on block ciphers and satisfies such integrity guarantees.[1]

[1] Strictly speaking, the Grade-1b designs of [BGP+20] that only provide integrity (so no confidentiality) in the presence of leakage would be sufficient for this purpose. Yet, since ensuring beyond-birthday integrity in the presence of leakage with n-bit block ciphers requires a $2n$-bit state, adding an ephemeral

An obvious direction to achieve this goal would be to use the TET mode of operation described in [BGP+20], that relies on Tweakable Block Ciphers (TBC). Yet, this solution would require to instantiate the TBC with block cipher calls. Using generic constructions like [Men15] would then require arguing about security in the presence of leakage during the intermediate computations (especially since our integrity guarantees are aimed to hold with unbounded leakages).[2] Besides, the tag verification of TET is based on inverting a TBC while our goal is to leverage a leakage-resilient PRF for this part.

We propose a solution avoiding these caveats which instantiates a Grade-2 design by leveraging a block cipher based hash function introduced by Mennink [Men17] slightly tailored for our purposes. It requires four block cipher calls per message block. We also propose a careful instantiation of the tag verification that does not rely on masking We show that this design is competitive with the CHES 2020 proposal and that it even improves it when a SHA256 coprocessor is not available (which is sometimes the case in low-end embedded devices). For completeness, we additionally describe solutions to ensure confidentiality in the presence of decryption leakage, which can be necessary in contexts where Intellectual Property (IP) must be protected [BPPS17, USS+20]. One heuristic option is to leverage a multi-user variant of our Grade-2 mode. Another option is to use a block cipher based variant of TEDT leveraging Mennink's hash function. As a bonus, our solutions also halve the amount of key material compared to [USS+20].

In summary, we show that (*i*) the AE scheme in [USS+20] does not enable secure firmware updates unless its tag verification is protected against side-channel attacks with implementation-level countermeasures (which contradicts its design goals); (*ii*) the evaluation of the LR-PRF used in [USS+20] is a sensitive task that hardly transfer from one device (or measurement setup) to another, hence suggesting the use of conservative security parameters when instantiating it; (*iii*) block cipher based modes of operation can be designed in order to mitigate the aforementioned integrity issue at the algorithmic level, in one or two passes depending on the type of protection against leakage-based IP theft required, and avoiding the use of SHA256 which can lead to performance gains.

We finally mention an application note from NXP on leakage-resilient primitives as evidence that there is industry demand for such concrete security solutions.[3]

Related works and terminology. The idea of limiting the power of side-channel attacks thanks to adapted cryptographic designs is an old one, already present in Kocher's first patents [Koc05]. Dziembowski and Pietrzak formalized this idea and proposed the first leakage-resilient primitive proven based on standard reductionist arguments [DP08]. Their work has then been the source of many leakage-resilient designs for various cryptographic primitives such as Pseudo-Random Number Generators (PRNGs) [Pie09, YSPY10], PRFs [SPY+10, FPS12] or Pseudo-Random Permutations [DP10]. These seed results on basic cryptographic primitives next triggered analyzes of complete functionalities like encryption and authentication [BPS15, PSV15], and rapidly shifted the attention of designers to Authenticated Encryption (AE) schemes mixing both integrity and confidentiality guarantees [BMOS17, BKP+18, BPPS17, GPPS19]. Two definitional frameworks exist for this purpose. Barwell et al. consider an all-in-one definition of leakage-resilience (excluding the leakage in the challenge query) [BMOS17]. Guo et al. rather consider a composite definition of leakage-resistance (including the leakage in the challenge query) [GPPS19]. Our investigations follow the approach of Guo et al. which, as discussed in [BBC+20],

key evolution mechanism that provides some confidentiality guarantees comes at very limited price and we therefore have limited incentives to design a Grade-1b mode in the context of this paper.

[2] A more efficient alternative would be to assume the AES-256 to provide a tweakable block cipher (e.g., as done in [BÖS11]). Yet, AES-256 coprocessors are less standard than AES-128 ones, and such an assumption is at least theoretically disputable in view of the related-key attacks described in [BK09]. In particular, such attacks show that formally, the AES-256 cannot instantiate an ideal cipher.

[3] https://www.nxp.com/docs/en/application-note/AN12304.pdf, see also [MSNF16].

is well-suited to analyze integrity and confidentiality guarantees separately and to take advantage of the different physical requirements they may lead to. We will use the term leakage-resilient for the PRF of Medwed at al. [MSJ12], the term leakage-resilient for the AE mode of [UHSS17] which is based on the definitions of Barwell et al., and the term leakage-resistant for our modes of operation which are based on the definitions of Guo et al. We further note that the leakage-resilient PRF of Medwed et al. essentially corresponds to the seminal construction of Goldreich, Goldwasser and Micali (GGM) [GGM86], the leakage-resilience of which has been analyzed by Faust et al. [FPS12]. Its originality mainly lies in a careful selection of the plaintexts to encryption to generate "key-dependent" algorithmic noise, which has only been analyzed heuristically so far. We finally mention that the issue of secure tag comparison is not a new one in cryptography. It has been frequently discussed in the context of authenticated encryption and has for example been put forward by both Barwell et al. [BPS15] and Berti et al. [BPPS17].

2 Background

In the following, we first recall the side-channel attack we use in the rest of the paper. Then, we continue with a description of the leakage-resilient PRF from [MSJ12].

2.1 Linear subspace template attacks

Profiled side-channel attacks can be performed by building templates in a linear subspace [APSQ06]. To do so, the adversary estimates the conditional probability density function of the leakage according to:

$$\hat{\mathsf{f}}[\vec{l}|x] = \frac{1}{\sqrt{(2\pi)^{n_d}|\hat{\mathbf{\Sigma}}|}} \exp^{\frac{1}{2}((\mathbf{W}\vec{l}-\hat{\mu}_x)'\hat{\mathbf{\Sigma}}(\mathbf{W}\vec{l}-\hat{\mu}_x)}, \tag{1}$$

where $\vec{l}$ is a leakage vector of length n_p and x a sensible variable. The matrix $\mathbf{W}$ with dimensions $n_d \times n_p$ is the linear projection from the leakage domain (directly sampled from a scope) to its linear subspace with n_d dimensions. The vector $\hat{\mu}_x$ of dimension n_d is the estimated mean of the leakage in the linear subspace conditioned to x. Finally, the matrix $\hat{\mathbf{\Sigma}}$ is the estimated pooled covariance matrix within the linear subspace. Thanks to this conditional distribution, the adversary can use Bayes' theorem and apply maximum likelihood to recover the secret, just like in a standard template attack [CRR02].

When dealing with subspace-based template attacks, a few parameters have to be carefully chosen such as n_p, n_d and a method to derive the projection $\mathbf{W}$. In this work, the dimensions of interest are selected to be the n_p most informative time samples according to Signal-to-Noise Ratio (SNR) [Man04]. The value of n_p as well as n_d are chosen in order to maximize the Perceived Information (PI) after projection [BHM+19]. Finally, the projection is build based on a Linear Discriminant Analysis (LDA) [SA08].

2.2 Medwed et al.'s leakage-resilient PRF

The LR-PRF introduced in [MSJ12] and used in [USS+20] aims at reducing the number of different plaintexts that can be encrypted with single a key, in effect leading to reducing DPA security to SPA security. As illustrated in Figure 1, it processes an input x with a key k using a tree-based construction: at each stage, a plaintext depending on n_b bits of x is encrypted with a key taken as the output (ciphertext) of the previous stage. The last stage of the LR-PRF is a whitening that encrypts a public plaintext. Therefore, the total number of calls to the block cipher (or stages) to run the LR-PRF is worth $1 + n/n_b$ where n is the size of the LR-PRF input. In order to mount a DPA against k, the adversary

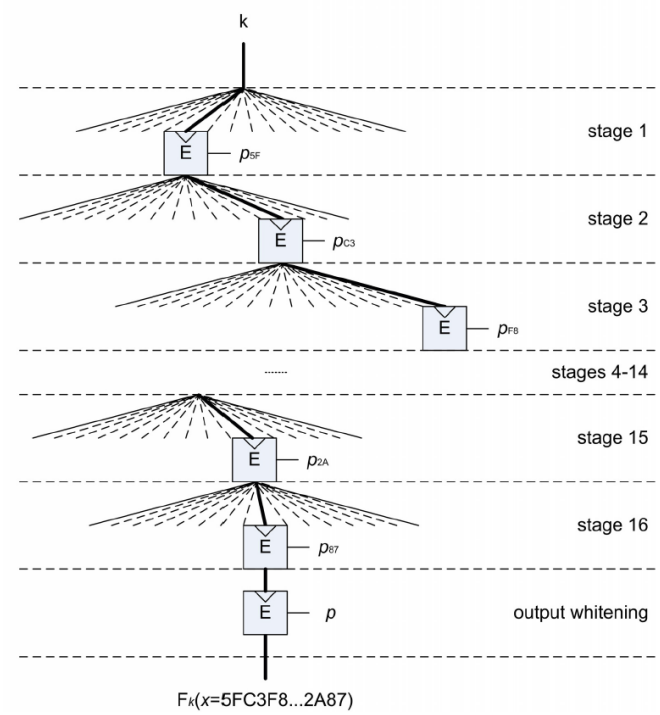


Figure 1: LR-PRF construction from [MSJ12] with $n_b = 8$ (illustration from [MSJ12]).

has access to 2^{n_b} different plaintexts. Since she has control of x, she can observe multiple leakage for the same input and so obtain averaged leakages with minimum noise.

In [MSJ12], the authors format the plaintext based on the n_b bits of x such that all the bytes have the same value (e.g., set to these n_b bits). As a result, for a byte-oriented cipher like the AES, the bytes of the plaintext that enter the key addition plus the S-box in the first cipher round will be equal. This makes the guesses performed by the DPA adversary equal for all the bytes. In a parallel implementation and assuming that the leakage model of each S-box is identical, an adversary will not be able to distinguish the leakage from the S-boxes. In case this attack vector is the best one (i.e., if attacking after the MixColumns operation is hard), the adversary will be forced to enumerate the permutation on all the key bytes. We will further elaborate on these hypotheses in Subsection 4.1.

3 Attack against the integrity of the [USS+20] LR-AE

In this section, we first describe the experimental setup used in the rest of the document. Then, we demonstrate a DPA against the tag verification that can be used to mount forgeries against the leakage-resilient AE proposed in [USS+20].

3.1 Targets and measurement setup

We perform experiments on two commercial MCUs with AES coprocessors. The first one is an ARM Cortex-M4 STM32F439ZI mounted on a (unmodified) demonstration board Nucleo-144.[4] The coprocessor has a 128-bit architecture meaning that all the S-boxes are processed in parallel. The device is running at its maximum clock frequency of 180[MHz].

[4] https://www.st.com/en/evaluation-tools/nucleo-f439zi.html

The second target is an ARM Cortex-M33 STM32L562QEI6QU with ARM TrustZone and a dual core. The MCU is also mounted on a (unmodified) demonstration board STM32L562E-DK.[5] The coprocessor has a 32-bit architecture meaning that a single AES column is processed in parallel. This target is also running at its maximum clock frequency of 110[MHz]. On both targets, the supply voltage is lowered to 1.8[V] which is the minimal specified value, in order to reduce the noise generated by the on-chip voltage regulator. The two investigated targets have similar but not identical coprocessors as the ones investigated in [USS+20], which are respectively based on 32-bit and 128-bit architectures, and are running on ARM Cortex-M4 and ARM Cortex-M3 devices. We evaluate the coprocessor of a more recent ARM Cortex-M33 in place of the ARM Cortex-M3.

Similarly to [USS+20], the side-channel leakage is measured with an electromagnetic (EM) probe. The EM signal is sampled with a PicoScope 6424E at 5[GSamples/sec] (with a 500[MHz] analog bandwidth), using a 10-bit resolution. This signal is amplified thanks to a PA 306 from Langer that offers 30[dB] of gain on the bandwidth 100[kHz] to 6[GHz]. For the ARM Cortex-M33, the probe RF-B 0.3-3 from Langer is used while for the ARM-Cortex-M4 the probe RS H2.5-2 from Rohde&Schwarz is selected. The quality of the equipment is similar to the one used in [USS+20], excepted that we did automatically position the probe instead of manually. For both targets, the probe was positioned thanks to a XYZ table used to scan the top of the (untouched) package. The SNR is estimated for all the S-boxes with 200,000 traces at each point of a grid. For the Cortex-M33, the grid is of size 16x16. For the Cortex-M4, we use a 64x64 grid.[6] The heatmaps obtained after the grid scan of Cortex-M4 are reported in Appendix A, Figure 15. For both targets, the heatmap for each S-box suggests to place the probe around the same area. Therefore, the probe is moved to the position maximizing the averaged SNR across the 16 S-boxes and we do not exploit spacial resolution as investigated in [UHSS17, UHS+18].

Overall, we were able to capture the traces at a rate larger than 30,000[measurements/sec] for the two investigated targets. Similarly to [SM16], the AES coprocessor is used as a PRG to generate the random plaintexts and keys, which saves most of the cost of communication with a control desktop. Namely, it encrypts in CBC mode a zero string with a known random IV. The SNRs are then computed on-the-fly thanks to a one-pass algorithm similar to [SM16], which totally saves the cost of disk writing. The estimations of SNR as well as most of the side-channel analyses have been performed with the open-source library SCALib.[7]

3.2 Tag recovery attack

In [BBC+20], the authors present a practical side-channel attack against a tag verification implemented on an ARM Cortex-M0. It can directly be used to forge valid messages under adversarial control [BPPS17], and therefore to break the integrity of a software update. We repeat that experiment with two differences: *(i)* we use an ARM Cortex-M4 which is more noisy, and *(ii)* we do template attacks in a linear subspace instead of directly in the leakage domain.[8] The target code is described in Listing 1, where a 128-bit tag candidate S is compared with the correct tag T by using 4 XOR instructions. After using 3 OR and 1 comparison, the returned value is 1 if all the bits of S and T are equal. We report the corresponding assembly code obtained with `objdump` in Appendix A, Listing 2.

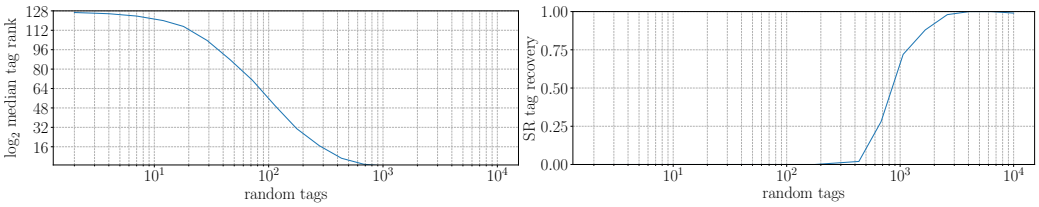
More precisely, considering the scheme from [USS+20], Algorithm 3, an adversary can try to forge a valid tag T for a ciphertext of her choice and a nonce. By having access

[5] https://www.st.com/en/evaluation-tools/stm32l562e-dk.html
[6] The Cortex-M33 package is smaller than the Cortex-M4, which explains the different resolutions.
[7] https://github.com/simple-crypto/SCALib
[8] The maximum SNR on the ARM Cortex-M4 we investigate is approximately 0.15 while it is approximately 0.8 on a similar ARM Cortex-M0 as investigated in [BBC+20].

```
1 uint32_t tag_verif(uint32_t *S,uint32_t *T){
2     uint32_t flag;
3     flag  = S[0] ^ T[0];
4     flag |= S[1] ^ T[1];
5     flag |= S[2] ^ T[2];
6     flag |= S[3] ^ T[3];
7     return (flag == 0);
8 }
```

Listing 1: Tag verification C code.

to a decryption oracle, she can attempt decryption with multiple random tags S. The oracle will first compute the valid tag T (lines 12 and 13) and then compare it with S (line 14) allowing one to mount a DPA on the XOR instructions in Listing 1.[9] In Figure 2a, we report the median rank of the 128-bit tag estimated with [PSG16], in function of the number of calls to the decryption oracle. In Figure 2b, we similarly report the success probability if no enumeration is possible. After $3,000$ calls, the adversary can recover the valid tag in full without enumeration with a probability almost equal to one. After 200 calls, she is already able to reduce the rank of the valid tag below 2^{32}. This would allow her to still succeed in the attack by enumerating these 2^{32} tags.



(a) Median tag ranks

(b) Tag recovery success rate

Figure 2: Median tag rank & success rate for various number of traces on ARM Cortex-M4. Metrics estimated on 100 independent random keys. $2 \cdot 10^6$ profiling traces used.

In the context of firmware updates investigated in [USS+20], this tag recovery could be exploited as follows. First, an adversary can anyway select a (garbage) ciphertext C and one nonce of her choice N. By targeting the verification, she is able to obtain the valid tag T and to decrypt the (garbage) plaintext M corresponding to the ciphertext C. This already results in a valid authenticated (garbage) firmware update which can lead to practical issues (e.g., denial-of-service) if no additional protection mechanisms are implemented. Second, if we additionally assume that the adversary can observe the garbage plaintext, she can then also recover the random string R output by the LR-PRG that depends only on N and a secret key, by adding C to M. In this case, she can choose a (malicious) plaintext of her choice M' and compute the corresponding ciphertext $C' = R \oplus M'$, which is valid for the nonce N. Afterwards, she only needs to repeat the attack against the tag verification to recover the corresponding valid tag T' This attack therefore allows the adversary to forge an authenticated firmware update (C', T', N) that she controls without recovering the long-term key. We note that it admittedly assumes that the adversary can observe the garbage plaintext, which may require some additional (possibly side-channel based) reverse engineering. But on the one hand, our following investigations will show that for a well instantiated mode, integrity can hold even with an unbounded leakage of ephemeral secrets (e.g., plaintexts), and on the other hand, this attack shows that if one single correct plaintext/ciphertext pair is ever recovered, then

[9] We note that by using template attacks, the adversary implicitly exploits all the leakage points that are bijectively related to a byte at the output of the target XOR operation.

forging any malicious software update of at most this size becomes possible.

Based on the above, a natural question is: where does this weakness arise from? That is, does this tag recovery attack result from problems with the model, the security proofs or a wrong instantiation of the construction? In this respect, we note that the model used to analyze [DJS19, KS20] is the one of Barwell et al. which considers leakage during verification [BMOS17]. They observe that a MAC whose verification algorithm recomputes the tag and checks for equality with the candidate tag cannot be strongly unforgeable with leakage. Therefore, they proposed an instance of leakage-resilient MAC based on pairings to mitigate this issue. The work of Degabriele et al. then suggested that such expensive pairings could be avoided in the (quite realistic) non-adaptive leakage setting, enabling simple MAC verifications that work by recomputing the tag on an input pair (nonce, ciphertext) and checking that it is identical to the given tag. Yet, it turns out their analysis ignores the leakage of the tag comparison. We show in this section that a simple DPA with non-adaptive leakages is sufficient to forge MACs in this case, as discussed in [BPPS17]. Such an issue was for example recognized in the ISAP v2 design [DEM+20], which comes with a discussion of secure tag comparison that was missing in ISAP v1 [DEM+17]. But it seems that it was ignored in the theoretical treatment of [DJS19, KS20], an error that was then propagated in [USS+20], as explicited by the sentence: "this does not add any additional attack vectors, as all sensitive operations are located within the LR-PRF and LR-PRG". Clearly, the comparison of the tags is another sensitive operation.

4 Evaluation of the [MSJ12, USS+20] LR-PRF

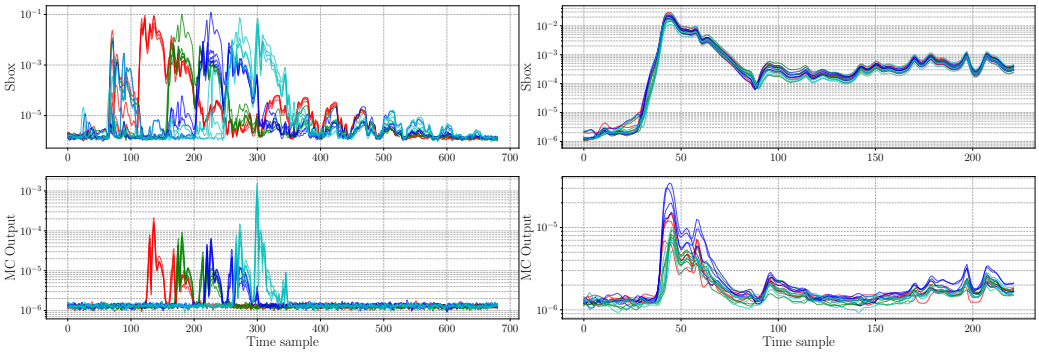
In this section, we detail our side-channel analysis of the LR-PRF. We first give a qualitative analysis aimed to highlight the relevant sources of leakage to target such a construction. Next, we describe our attack results quantitatively and conclude with a more general discussion on the evaluation of low-complexity DPA/SPA.[10]

4.1 LR-PRF: qualitative leakage analysis

The heuristic security provided by Medwed et al.'s LR-PRF is based on two main working principles. On the one hand, it bounds the amount of average leakage traces that an adversary can exploit to perform a DPA. Namely, she can only access 2^{n_b} traces to attack each stage of the LR-PRF. On the other hand, the plaintexts that are encrypted are not random since they are structured as vectors with 16 times the same value. This aims to generate "key-dependent algorithmic noise" as described in Subsection 2.2. For this purpose, it is additionally required that the implementation executes the S-boxes in parallel, that their leakage models are sufficiently similar and that targeting the MixColumns operation is hard. If these conditions are satisfied, it implies that the adversary will not be able to distinguish the leakage from the 16 S-boxes (due to their identical input), at least leading to the requirement to enumerate all the possible permutations of key bytes, which has a cost of $16! \approx 2^{44.25}$ for the AES (see [MSJ12], Section 5). As a result, the main qualitative questions when analyzing the physical security of such a primitive are whether attacking MixColumns is indeed hard, and whether the S-boxes' leakage models are sufficiently similar. In the rest of this section, we analyze these two points in detail.

Regarding the first question, Figure 3 reports the SNR observed on the S-boxes and MixColumns output bytes, for our two targets. On the ARM Cortex-M33, we first note that the 4 columns are processed serially, confirming that the underlying architecture is

[10] Unterstein et al. also analyze the additional leakage that is provided by key transportation. We do not repeat this part of their experiments since our analysis of the LR-PRF will already conclude that using a minimum n_b value is recommended for the two targets we consider in this paper.



(a) Cortex-M33 (32-bit arch) **(b)** Cortex-M4 (128-bit arch)

Figure 3: SNR on AES S-boxes and MixColumns outputs evaluated on $200 \cdot 10^6$ traces. The color mapping corresponds to bytes within the same AES column.

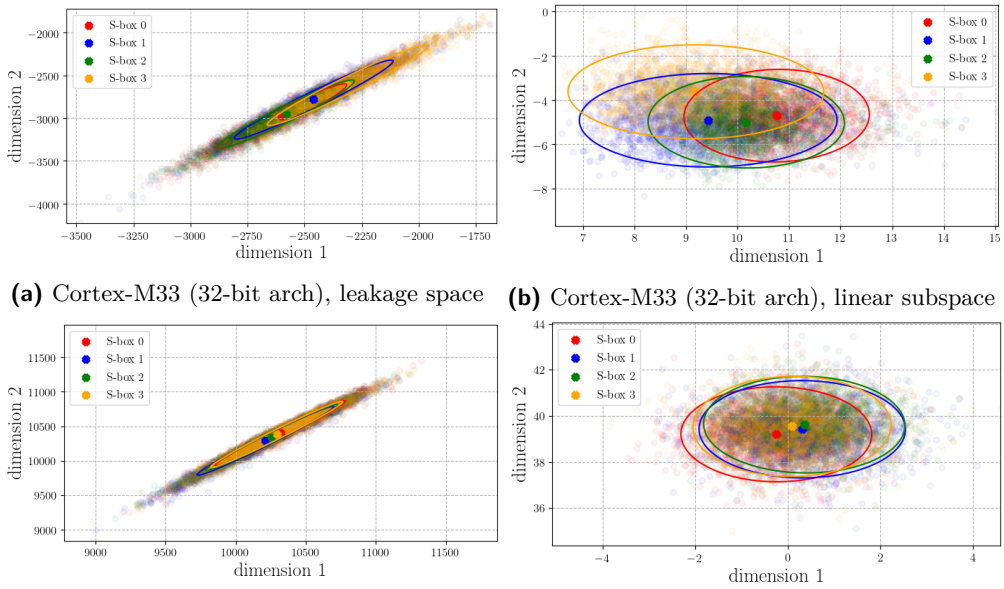
32-bit. For the ARM Cortex-M4, all the SNR peaks occur simultaneously, hinting towards a 128-bit architecture. We observe that the SNR on the MixColumns operation is at least two order of magnitudes lower than the one on the S-boxes. We assume the latter is due to MixColumns's being implemented by harder to target combinatorial logic and/or its leakages requiring wider key guesses to predict transitions. Hence, we conclude from this experiment that Medwed et al.'s first requirement is sufficiently satisfied.[11]

Regarding the second question, Figure 4 depicts the leakage distributions in a linear subspace and in the direct leakage space for a constant value at the input of each of the first AES column's S-boxes. The distributions are shown for the two first dimensions and the linear subspace is the one of the first S-box. Looking at the distributions on the ARM Cortex-M33 (resp., Cortex-M4), we observe in Figure 4a (resp., Figure 4c) that they are already different in the original leakage space. These differences in the first dimensions are enhanced by moving to the linear subspace, as depicted in Figure 4b (resp., Figure 4c).

In Figure 5, we then study the difference between the leakage distributions of the first vs. the other S-boxes, based on Hotelling's T^2-test [Hot31]. On these plots, the X-axis is the amount of averaging performed by the adversary and the Y-axis is the significance of the difference. First, we observe that by increasing the amount of averaging, the significance increases. This is expected since it reduces the noise on the distributions, making their means easier to estimate. Additionally for the ARM Cortex-M33 (resp., Cortex-M4), by comparing Figure 5a (resp., Figure 5b) with Figure 5c (resp., Figure 5d) we observe that the differences are of the same magnitude in the linear subspace and in the direct leakage space but that more dimensions are needed in the second case (200 dimensions instead of 6). We notice that the advantage of using additional dimensions is larger on the ARM Cortex-M33. The difference between the distributions on the ARM Cortex-M33 is also larger than on ARM Cortex-M4, presumably due to a more serial implementation.

While these results remain qualitative, they highlight that the leakage models of different S-boxes are different for our two targets. Hence, they contradict one of Medwed et al.'s assumptions. Informally, our results suggest that with a precise enough profiling, it should be possible to exploit these differences, enabling the adversary to escape the enumeration of a 16-permutation as expected from the LR-PRF design. In that context, projecting the leakages in a linear subspace allows "concentrating" these differences in a

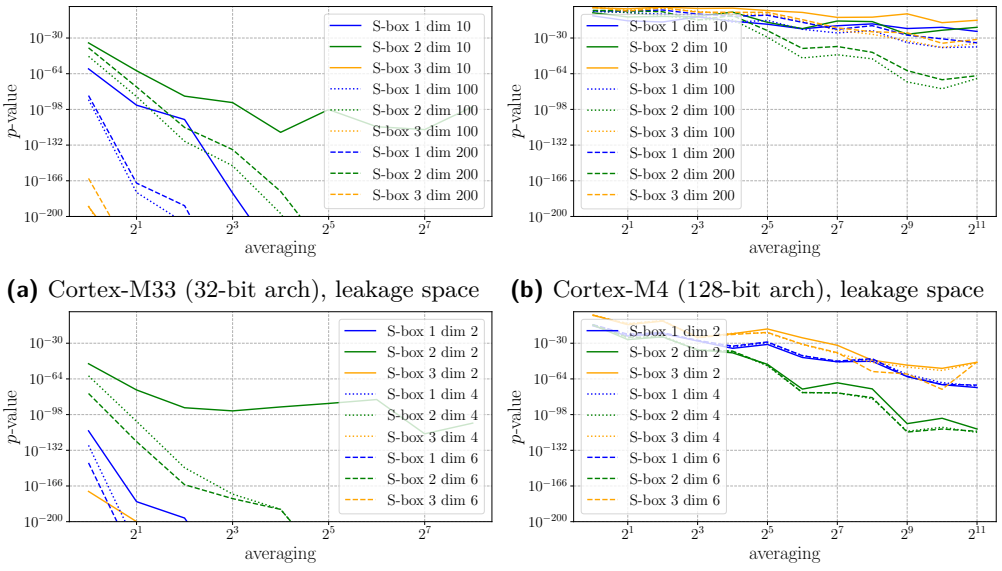
[11] Our SNR is admittedly based on 8-bit guesses. However, 32-bit linear models estimated with [SLP05] did not exhibit significantly better results. So given the wide gap with the S-box leakages, we assume that targeting MixColums will remain a worse strategy for our two target devices.



(a) Cortex-M33 (32-bit arch), leakage space **(b)** Cortex-M33 (32-bit arch), linear subspace

(c) Cortex-M4 (128-bit arch), leakage space **(d)** Cortex-M4 (128-bit arch), linear subspace

Figure 4: Leakage distributions of multiple S-boxes for a constant input value. Distributions after 256 (resp., 2048) averaging on the ARM Cortex-M33 (resp., Cortex-M4). The linear subspace is the one of the first S-box. The two dimensions of the original leakage space are the ones with highest SNR. Curves represent a Gaussian fit on the clusters.



(a) Cortex-M33 (32-bit arch), leakage space **(b)** Cortex-M4 (128-bit arch), leakage space

(c) Cortex-M33 (32-bit arch), linear subspace **(d)** Cortex-M4 (128-bit arch), linear subspace

Figure 5: Hotelling's T^2-test between leakage distributions of S-box 0 and other S-boxes for various number of dimensions, computed with (averaged) 2,000 samples per distribution.

few dimensions which makes it easier to exploit by an adversary/evaluator. In Appendix B, we illustrate that the amount of information extracted in the original leakage space and in the linear subspace is similar. But since the second solution uses less dimensions, it requires less profiling traces to accurately profile a model. We note that [UHSS17, UHS+18] already pointed out that linear subspaces and spacial resolution can be used to observe different leakage models for different S-boxes. Our experiments show that a similar effect can be obtained without spacial resolution for the targets we look at.

4.2 LR-PRF: quantitative attacks

We now report the results of attacks against the LR-PRF. We used template attacks in linear subspace, as described in Subsection 2.1, since they provided the best results. For the ARM Cortex-M33 (resp., Cortex-M4), the profiling is performed on $500 \cdot 10^3$ (resp., $2 \cdot 10^6$) averaged traces. For example, for 256 (resp., 2048) averaging, we collected a total of $256 \cdot 500 \cdot 10^3$ (resp., $2048 \cdot 2 \cdot 10^6$) measurements. We note that the profiling was performed using random keys and plaintexts in order to avoid biasing our models with key-dependent features that would not be exploitable during the online attack phase.

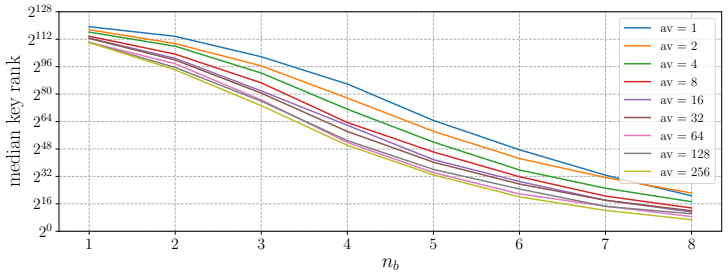
The median key rank estimated according to [PSG16] is reported in Figure 6, where the X-axis is the LR-PRF parameter n_b (see Subsection 2.2). We observe that by increasing n_b, and so the number of different averaged traces available to the adversary, the key rank significantly decreases. The fact that the key rank decreases below $16! \approx 2^{44.25}$ confirms that we are able to exploit the different leakage models of the different S-boxes. Averaging also allows decreasing the key rank but its impact saturates at some point, similarly to what can be observed in [USS+20], Figure 10. On the ARM Cortex-M33 (resp., Cortex-M4) starting from 64 (resp., 512), averaging does not improve the attacks.

Concretely, for the ARM Cortex-M33 our adversary reduces the median key ranks down to 2^9 if $n_b = 8$, and down to 2^{50} if $n_b = 4$. For keeping it larger than 2^{90}, we need $n_b \leq 2$. The ARM Cortex-M4 is more resistant thanks to its parallel architecture: with $n_b = 8$, the median key rank is equal to 2^{25} but with $n_b \leq 5$ it is already above 2^{96}.

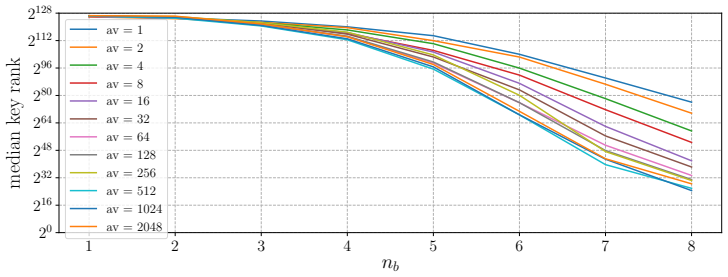
4.3 LR-PRF: the evaluation challenge

Our analyses imply that smaller n_b values than reported in [USS+20] should be used for our similar targets and measurement setup. For example, Figure 11 in this reference suggests that on both targets, $n_b = 6$ leads to more than 100 bits of security. In our evaluations, this value is reduced to 20 (resp., 70) on the ARM Cortex-M33 (resp., Cortex-M4).

Technically, these results are not in strong contradiction with previous works. For example, as illustrated in [UHSS17, UHS+18], the resistance of a LR-PRF highly depends on the adversary's measurement capabilities and the exploitable signal within the target. On an FPGA implementation of the LR-PRF, is has been showed that such schemes can be insecure even for $n_b = 1$ depending on the floorplanning. To some extent, this is typical for any side-channel countermeasure: eventually, their security depends on physical assumptions. Fulfilling such assumptions is always a challenge, whether being to ensure a sufficient noise level or a sufficiently low side-channel signal. Yet, one important conclusion of our experiments is that the "low signal" requirement of LR-PRF constructions ensuring low complexity DPA (or even SPA) to be hard is also more challenging to evaluate. The main reason of this fact is that such constructions generally enable averaging the physical noise. In this context, evaluating the signal becomes very dependent on the shape (i.e., the deterministic part) of the leakage function, while "noise amplification" countermeasures such as masking or shuffling only require that the ratio between signal and noise (SNR) is sufficiently low, independently of the signal shape. Since the shape of a leakage function is quite sensitive to (even minor) modifications of the target designs and measurement setups,



(a) Cortex-M33 (32-bit arch)



(b) Cortex-M4 (128-bit arch)

Figure 6: Median key rank of the Medwed et al. LR-PRF for various data complexities. Medians are estimated on 100 independent attacks with random keys.

constructions relying on limited signal (if implemented without additional countermeasures), inherently carry a higher risk of suboptimal (i.e., non worst-case) evaluation.

Such a risk is illustrated by the comparison between our analysis and the one of [USS+20] where mild variations of the measurement setup and a collection of 1,000 more profiling traces lead to different security parameters for similar targets (i.e., same architecture but not exactly the same MCU). They could be amplified if the adversary obtains a better a-priori knowledge of her target, or performs larger key guesses than the evaluator. Hence, we conclude that designers should be conservative when selecting the parameter n_b of a LR-PRF. As will be detailed in Section 6, low n_b values are anyway not overly detrimental from a performance viewpoint, since they can be amortized for long messages.

5 The LR-BC-2 mode of operation

In order to design a one-pass AE mode with Grade-2 leakage security, we need to combine an ephemeral key evolution scheme and a compression function. On the one hand, the ephemeral key evolution scheme allows iteratively processing each block of message with a fresh key, which is reminiscent of designs conferring confidentiality guarantees in the presence of encryption leakage [BBC+20]. On the other hand, the compression function is used to progressively absorb the blocks to make the computation more and more dependent on the already processed blocks, leading to a kind of digest that can then be authenticated using a fixed-length leakage-resilient Message Authentication Code (LR-MAC). While relying on a TBC is not only a convenient solution to implement the above principle but also a way to achieve high security bounds (e.g., with 2 TBC calls per block as in TET [BGP+20]), we do not have the same degree of freedom when only relying on a block cipher, as explained in the introduction. In this section, we show how we can still reach high security guarantees in the presence of decryption leakages when only block ciphers like the AES, are available. Our new mode, coined LR-BC-2, only requires four BC calls per

block, which is equivalent in performance to a variant of TET where each TBC call would be implemented by 2 BC calls as in [Men15], even if such a composition would be insecure in our (unbounded) leakage setting. We start with a general description of our design and its rationale. We follow with a formal quantitative analysis of its integrity in the presence of leakage, which is the most important property for our intended (secure firmware update) application. We finally provide a shorter (heuristic) discussion of confidentiality in the presence of leakage, for which LR-BC-2 does not exhibit particular innovations.

5.1 Specification and rationale

Our mode follows the general 3-step blueprint suggested in [BBC+20] for Grade-2. In the *initialization* step, we compute a random $2n$-bit state (K_1, L_1) from a key derivation function $\mathsf{KDF}_K(N)$, where K_1 is an ephemeral encryption key. For this first step, we borrow the BC-based bit-by-bit PRF of the previous section that we apply to the nonce N to get a seed K_0, allowing producing our initial state (K_1, L_1). As a second step, the *bulk* of the computation is made as a one-time encryption of $M_1 \| \cdots \| M_\ell$ where, at each successive processing of a message block M_i, the corresponding ciphertext block C_i is created and absorbed, and the state is refreshed. Eventually, in the *finalization* step, we use an LR-MAC, aka Tag Generation Function (TGF) to authenticate the final state with an appropriate verification mechanism. We now focus on the second step.

Our starting point is the BC-based compression function family due to Mennink [Men17], which achieves optimal collision security using only 3 calls to the underlying n-bit block cipher. In the sequel, we denote this function by MenH. Given a $2n$-bit state (K_i, L_i) and a n-bit block C_i to absorb, $\mathsf{MenH}(K_i, L_i, C_i)$ compresses the $3n$-bit input into $2n$-bit output state (K_{i+1}, L_{i+1}). The reason why we rely on MenH is twofold: (i) in the ideal-cipher model, the collision bound holds even if all the intermediate values are exposed, which comes in handy for our integrity with leakage purpose; (ii) by carefully choosing MenH in the family, we can use it to turn K_i into an ephemeral secret key allowing encrypting M_i, which only requires a single additional call to the BC. More precisely, if E is the underlying BC, given an i-th $2n$-bit state (K_i, L_i), we encrypt M_i as $C_i = E_{K_i}(L_i \oplus \theta) \oplus M_i$ and compute $\mathsf{MenH}(K_i, L_i, C_i) = (K_{i+1}, L_{i+1})$ in such a way that K_{i+1} can, in turn, be used as the next ephemeral key. The value θ is a non-zero constant, *e.g.* $\theta = 10^*$.

In Figure 7, we depict the encryption algorithm of our new mode, where (K_1, L_1) is the initial state. The 4 BC calls per block M_i can easily be parsed from the picture. For each processing of message block, the 3 BC calls in black as well as the black wires and the linear map $\mathcal{L}$ (computed over $\mathbb{F}_{2^n}$) represent MenH. In red, the additional BC call, M_i, and the wires correspond to our plug-in that turns MenH into a block encryption. We note that many $\mathcal{L}$ transforms are possible for the collision/preimage-resistant compression function. Our specific choice is illustrated in Figure 8(a). We additionally assume that the bit-length of any message and any ciphertext is a multiple of n since our goal is to show the leakage security of our design, which is orthogonal to (standard) padding issues.

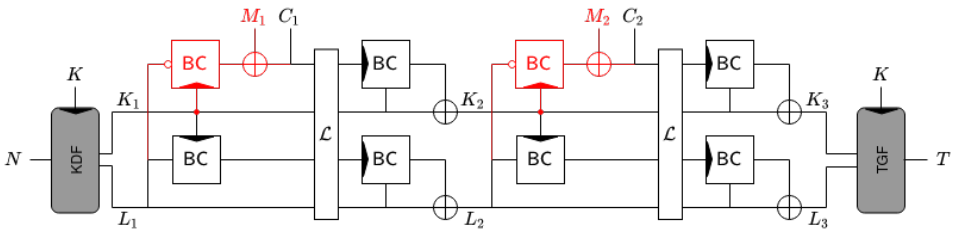
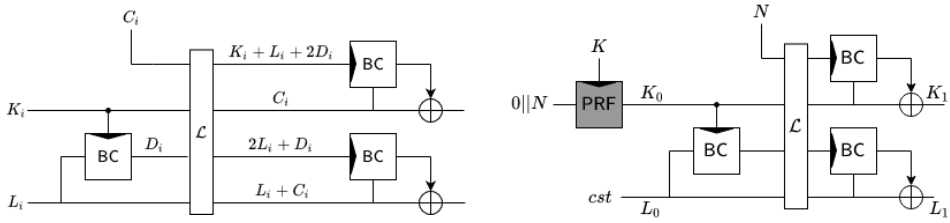


Figure 7: The LR-BC-2 mode of operation.

We now picture the design of KDF in Figure 8(b). It first uses the key K to set up an IV (K_0, L_0) for a constant cst so that $L_0 := cst$. The preimage resistance of the half state L-value as the image of a single-length compression function ensures that no internal state can collide with (K_0, L_0) because of this constant cst. To avoid IV-collisions between encryption and decryption, we simply apply MenH once with the nonce: we make the call $\mathsf{MenH}(K_0, L_0, N)$. This first call inside KDF forces the initial state (K_1, L_1) to diverge for distinct nonces. We also add a separation bit before the nonce. This is for reusing the same secret key both in the initialization and the finalization of our mode.



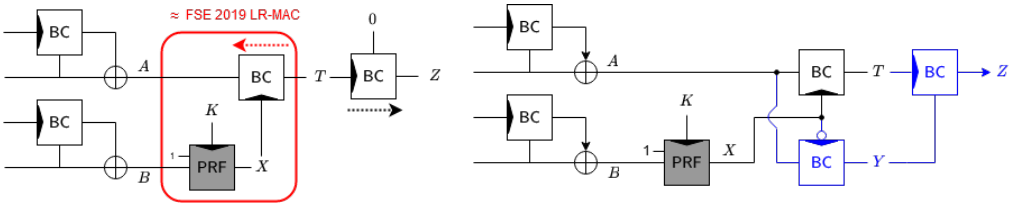
(a) The coefficients of $\mathcal{L}$.

(b) KDF for some constant cst.

Figure 8: The Key Derivation Function (KDF) and our linear map $\mathcal{L}$.

It remains to explain the design of TGF, which we would like to build without relying on hardware-level countermeasures such as masking, and to argue why the verification of maliciously chosen tags does not reveal enough information on a valid tag T during the decryption of fresh ciphertexts. A natural starting point for this purpose is the recent work of Guo et al. [GSWY19], which describes different such constructions with good bounds. We now argue why they cannot be used in our context and how to tweak them so that they lead to secure implementations based on unprotected hardware coprocessors.

For this purpose, let us start with the MAC illustrated in the red box of Figure 9(a), and integrated in the LR-BC-2 mode of operation. Guo et al. showed that it ensures beyond-birthday security if implemented with two strongly protected block ciphers (e.g., thanks to masking) and the verification is done by inverting the upper block cipher. The problem in our context is that we do not want to rely on masking. Hence, typically, the lower PRF will be strongly protected thanks to the leakage-resilience features discussed in Section 4, and the upper block cipher call will rely on an unprotected hardware AES implementation. As a result, if we apply this idea in our context we cannot assume anymore that the intermediate key X will be leak-free. In fact, a simple DPA against LR-BC-2's decryption is possible. By querying decryption on $(N, C_1, \ldots, C_\ell, S)$ for many candidate tags S and fixed nonce and ciphertext blocks, the last state (A, B) and X, are fixed. The DPA is then run against X in the verification when $E_X^{-1}(S)$ is performed on distinct S.



(a) Birthday security.

(b) Beyond-birthday security (1st attempt).

Figure 9: Towards our tag generation/verification functions.

A first step towards solving this issue, illustrated in Figure 9(a), is to use a tag verification similar to the one proposed for ISAP v2 [DEM+20]. That is, rather than

performing the verification by inverting the block cipher (which, as discussed in [BPPS17], enables the tag comparison to remain secure with unbounded leakages), use an additional call to the block cipher to encrypt a constant (e.g., zero) plaintext and compare the resulting Z values during verification. Such a tweak mitigates the previous attack since this time, the adversary cannot perform a DPA against $\mathsf{BC}_X(A)$ by decrypting multiple tag candidates S. She rather has to attack this block cipher in the forward direction. But in order to fix the key X for α different A values so that a DPA can be performed with data complexity α, she will have to find multi-collisions on B. For reasonable values of α, this task becomes computationally intensive as the probability of finding an α-collision for a random function roughly decreases to $2^{-n(\alpha-1)}\binom{q}{\alpha}$, which remains safe for around $q \approx 2^{n(\alpha-1)/\alpha}$ queries. Therefore, to extract the internal X the adversary is only left with an SPA. Yet, it remains that the tag T and the value Z are only 128-bit wide and a collision on Z indicates a collision on T with high probability, so that this tag verification is only birthday secure in the black box world.[12] Indeed, by making $2^{n/2}$ encryption queries as well as $2^{n/2}$ decryption queries, with good chance we will have that among the many tags T returned in the ciphertexts we will have as many corresponding Z (that we can compute) and a colliding Z' returned from the decryption. For that identified decryption query we simply guess that the valid tag is the one from which the colliding Z was generated.

Therefore, a second step to keep our TGF/TVF beyond-birthday secure is to ensure that Z depends on a wider state, which we achieve by generating an additional value Y that we use as a plaintext to produce Z, as illustrated in Figure 9(b). We note again that the security of this proposal cannot hold with unbounded leakages as the decryption-based proposal in [BPPS17]. In particular, it requires SPA security for the intermediate key X and the tag T which is internally computed during the verification. Yet, the latter is natural in our case, since we assume it for the leakage-resilient PRF anyway.

While this solution may already look strong, it however remains that a repetition of B implies a repetition of the ephemeral key X, while we have to reduce the number of times $\mathsf{PRF}[E](1\|\cdot)$ – our notation for the leakage-resilient PRF using E as underlying block cipher with a bit of separation at its input – is run on the same input to avoid SPA with (constant but) larger number of leakages against the tag verification. Since the MenH outputs are not random over $\{0,1\}^{2n}$ (Mennink showed they are up to a tight birthday bound, but we aim for beyond-birthday integrity), there may be up to α^2 repetitions of a B-value among all the final states (A, B) if there are some α-multi-collisions on the underlying n-bit output compression functions inside the computation of MenH. Indeed, even if there are at most α multi-collisions on a single value of B, each of these collisions may themselves come from previous α multi-collisions on the $2L+D$ value, which is the key-input value of the last lower BC call in MenH – see Figure 8(a).

Since the adversary controls C_ℓ, the corresponding plaintext-input of that BC call can be chosen equal between key-input collisions. We would thus have to deal with α^2 possible repetitions of B since B is the output of two successive iterations of some compression functions. In turn, the key X may be manipulated up to $2\alpha^2$ times in Figure 9(b).[13]

Concretely, this essentially implies that in order to guarantee security with up to $q \approx 2^{n(\alpha-1)/\alpha}$ queries, one would also require that the block cipher implementation used in the construction of Figure 9(b) resists SPA with $q_{\mathsf{spa}} = 2\alpha^2$ queries. For 96 bits of security, this gives $q_{\mathsf{spa}} = 32$ (which correponds to $n_b = 5$ in our empirical evaluation of Section 4.2). For 112 bits of security, this gives $q_{\mathsf{spa}} = 128$ (which corresponds to $n_b = 7$). Those values correspond to SPA requirements that are not satisfied by the devices we evaluated.

As a result, the third and the last step to build our TGF/TVF is to carefully add one more call to a n-bit output compression function on (A, B) before going into $\mathsf{PRF}[E](1\|\cdot)$.

[12] A similar concern happens with the tag verification of ISAP v2 [DEM+20].
[13] Where the factor 2 comes from the fact that X is used to encrypt twice.

While this obviously reduces the number of repetitions from α^2 to α since the final states are pairwise distinct, we also have to introduce one more feed-forward in order to avoid collisions on (A, X). We illustrate our proposed TGF/TVF functions in Figure 10. In this case, we get ≈96-bit security with $q_{\mathsf{spa}} = 8$ (which corresponds to $n_b = 3$) and ≈112-bit security with $q_{\mathsf{spa}} = 16$ (which corresponds to $n_b = 4$). Those values better match what our implementations offer and (we believe) correpond to an interesting security vs. performance tradeoff. More precise evaluations are given after the proof in the next section.

We finally note that a simple but expensive solution to remove this multi-collision SPA is to directly send A and B to a leakage-resilient PRF with $2n$ input bits.

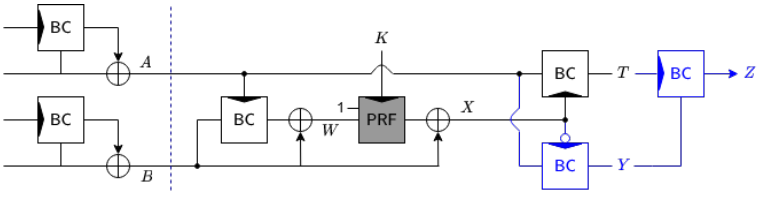


Figure 10: Our tag generation function with the tag comparison.

The full specification of our mode is given in Figure 11. To encrypt $\mathbf{M} = (M_1, \ldots, M_\ell)$ with nonce N, computing the ciphertext $\mathbf{C} = (C_1, \ldots, C_\ell, T)$, where T is the tag, requires $4\ell + 5$ calls to the block cipher E in addition to both evaluations of $\mathsf{PRF}[E]$.

Algorithm $\mathsf{Enc}[E](K, N, \boldsymbol{M})$
1. $M_1 \| \ldots \| M_\ell \xleftarrow{n} \boldsymbol{M}$
2. $(K_1, L_1) \leftarrow \mathsf{KDF}[E](K, N)$
3. **for** $i = 1$ **to** ℓ **do** // $\theta = 10^*$
4. $\quad C_i \leftarrow E_{K_i}(L_i \oplus \theta) \oplus M_i$
5. $\quad (K_{i+1}, L_{i+1}) \leftarrow \mathsf{MenH}[E](K_i, L_i, C_i)$
6. $T \leftarrow \mathsf{TGF}[E](K, K_{\ell+1}, L_{\ell+1})$
7. $\boldsymbol{C} \leftarrow C_1 \| \ldots \| C_\ell \| T$
8. **return** $\boldsymbol{C}$

Algorithm $\mathsf{KDF}[E](K, N)$
1. $K_0 \leftarrow \mathsf{PRF}[E](K, 0 \| N)$ // $L_0 = 0^n$
2. $(K_1, L_1) \leftarrow \mathsf{MenH}[E](K_0, L_0, N)$
3. **return** (K_1, L_1)

Algorithm $\mathsf{TGF}[E](K, A, B)$
1. $W \leftarrow E_A(B) \oplus B$
2. $X \leftarrow \mathsf{PRF}[E](K, 1 \| W) \oplus B$
3. **return** $T \leftarrow E_X(A)$

Algorithm $\mathsf{TVF}[E](K, A, B, S)$
1. $W \leftarrow E_A(B) \oplus B$
2. $X \leftarrow \mathsf{PRF}[E](K, 1 \| W) \oplus B$
3. $Y \leftarrow E_{X \oplus \theta}(A),\ T \leftarrow E_X(A)$
4. **return** $[\, E_S(Y) \stackrel{?}{=} E_T(Y) \,]$

Algorithm $\mathsf{Dec}[E](K, N, \boldsymbol{C})$
1. $C_1 \| \ldots \| C_\ell \| S \xleftarrow{n} \boldsymbol{C}$
2. $(K_1, L_1) \leftarrow \mathsf{KDF}[E](K, N)$
3. **for** $i = 1$ **to** ℓ **do** // $\theta = 10^*$
4. $\quad M_i \leftarrow E_{K_i}(L_i \oplus \theta) \oplus C_i$
5. $\quad (K_{i+1}, L_{i+1}) \leftarrow \mathsf{MenH}[E](K_i, L_i, C_i)$
6. $\beta \leftarrow \mathsf{TVF}[E](K, K_{\ell+1}, L_{\ell+1}, S)$
7. **if** $\beta = 0$ **then return** $\perp$
8. $\boldsymbol{M} \leftarrow M_1 \| \ldots \| M_\ell$
9. **return** $\boldsymbol{M}$

Algorithm $\mathsf{PRF}[E](K, b \| U)$
1. $\nu_1 \| \ldots \| \nu_n \xleftarrow{1} U$
2. $k_0 \leftarrow E_K(P_b)$
3. **for** $i = 1$ **to** n **do** // P_0, P_1, P_2
4. $\quad k_i \leftarrow E_{k_{i-1}}(P_{\nu_i})$
5. **return** $k_{n+1} \leftarrow E_{k_n}(P_2)$

Algorithm $\mathsf{MenH}[E](K, L, C)$
1. $D \leftarrow E_K(L)$
2. $K' \leftarrow E_{K+L+2D}(C) \oplus C$
3. $L' \leftarrow E_{2L+D}(L + C) \oplus (L + C)$
4. **return** (K', L')

Figure 11: Definition of the LR-BC-2 mode, using a BC E.

$\mathsf{PrivK}^{\mathsf{CIML2}}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n)$ experiment

Initialization:
1. $k \leftarrow \mathsf{Gen}(1^n)$, $\mathcal{S} \leftarrow \emptyset$

Finalization:
1. $(N, A, C) \leftarrow \mathcal{A}^{\mathsf{LEnc}_k, \mathsf{LDec}_k, \mathsf{L}}(1^n)$
2. If $(N, A, C) \in \mathcal{S}$, return 0
3. If $\mathsf{Dec}_k(N, A, C) = \perp$, return 0
4. Return 1

Leaking encryption: $\mathsf{LEnc}_k(N, A, M)$
1. $C \leftarrow \mathsf{Enc}_k(N, A, M)$
2. $\mathcal{S} \leftarrow \mathcal{S} \cup \{(N, A, C)\}$
3. Return $(C, \mathsf{L}_{\mathsf{Enc}}(k, N, A, M))$

Leaking decryption: $\mathsf{LDec}_k(N, A, C)$
1. Return $(\mathsf{Dec}_k(N, A, C), \mathsf{L}_{\mathsf{Dec}}(k, N, A, C))$

Figure 12: The CIML2 security game.

5.2 Integrity in the presence of leakage: formal analysis

Definition 1 (Nonce-Based AEAD [Rog02])**.** A *nonce-based authenticated encryption scheme with associated data* is a tuple $\mathsf{AEAD} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ such that, for any security parameter n, and keys in $\mathcal{K}$ generated from $\mathsf{Gen}(1^n)$:

- $\mathsf{Enc} : \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{M} \to \mathcal{C}$ deterministically maps a key in $\mathcal{K}$, a nonce in $\mathcal{N}$, some blocks of associated data in $\mathcal{AD}$, and a message in $\mathcal{M}$ to a ciphertext in $\mathcal{C}$.
- $\mathsf{Dec} : \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{C} \to \mathcal{M} \cup \{\perp\}$ deterministically maps a key in $\mathcal{K}$, a nonce in $\mathcal{N}$, some blocks of associated data in $\mathcal{AD}$, and a ciphertext in $\mathcal{C}$ to a message in $\mathcal{M}$ or to a special symbol $\perp$ if integrity checking fails.

The sets $\mathcal{K}, \mathcal{N}, \mathcal{AD}, \mathcal{M}, \mathcal{C}$ are completely specified by n. Given a key $k \leftarrow \mathsf{Gen}(1^n)$, $\mathsf{Enc}_k(N, A, M) := \mathsf{Enc}(k, N, A, M)$ and $\mathsf{Dec}_k(N, A, M) := \mathsf{Dec}(k, N, A, M)$ are deterministic functions whose implementations may be probabilistic.

Definition 2 (CIML2)**.** An authenticated encryption $\mathsf{AEAD} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with leakage function pair $\mathsf{L} = (\mathsf{L}_{\mathsf{Enc}}, \mathsf{L}_{\mathsf{Dec}})$ provides $(q_e, q_d, q_l, t, \varepsilon)$-ciphertext integrity with nonce misuse and leakages (both in encryption and decryption) given a security parameter n if, for all (q_e, q_d, q_l, t)-bounded adversaries $\mathcal{A}^{\mathsf{L}}$, we have:

$$\Pr\left[\mathsf{PrivK}^{\mathsf{CIML2}}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n) \Rightarrow 1\right] \leq \varepsilon,$$

with the $\mathsf{PrivK}^{\mathsf{CIML2}}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$ game of Figure 12 where $\mathcal{A}^{\mathsf{L}}$ makes at most q_e leaking encryption queries, q_d leaking decryption queries and q_l leakage evaluation queries.

For the sake of simplicity we do not deal with associated data. Therefore, when the adversary makes an encryption (resp., decryption) query for a nonce N and a plaintext M (resp., ciphertext C) we implicitly set the associated data to the empty string.

Idealized leakage assumptions. As a usual first step in the analysis of leakage-resistant modes of operation, we assume that the computation of the ideal cipher is leak-free as long as it is called a small constant number of times on each ephemeral secret key [PSV15]. This small number corresponds to the 2^{n_b} queries of the leakage-resilient PRF and to the α-multi-collisions. More precisely, we make the following simplifying assumption.

Given a block cipher $E : \{0,1\}^n \times \{0,1\}^n \mapsto \{0,1\}^n$, we say that E satisfies (α, u)-leakage independence if for any adversary $\mathcal{A}$ the leakage contains no information about the u ephemeral keys in the next experiment where they are explicitly used at most 2α times. Let $U_1, \ldots, U_u$ be random values over $\{0,1\}^n$, and $Q_1, \ldots, Q_u$ and R be initially empty sets.[14] In the experiment the adversary $\mathcal{A}$ can make the following query: on input

[14] These values correspond to the outputs of the PRF in TGF.

(i, A_{ij}, B_{ij}), where $1 \leq i \leq u$ and $A_{ij}, B_{ij} \in \{0,1\}^n$, (i) if $(A_{ij}, B_{ij}) \in Q_i$, abort;[15] (ii) if $(U_i \oplus B_{ij}, A_{ij})$ or $(U_i \oplus B_{ij} \oplus 0^*1, A_{ij}) \in R$, abort;[16] (iii) if $|Q_i| = \alpha$, abort; (iv) update $Q_i \leftarrow Q_i \cup \{(A_{ij}, B_{ij})\}$ and $R \leftarrow R \cup \{(U_i \oplus B_{ij}, A_{ij})\}$; and finally (v) compute and return $E_{U_i \oplus B_{ij}}(A_{ij})$ and $E_{U_i \oplus B_{ij} \oplus 0^*1}(A_{ij})$ as well as the leakage of these computations.

As clear from the previous discussions, the smaller is α, the weaker is the assumption. Depending on the number u of explicit random ephemeral keys, there might be many collisions among U-values. However, since no computation is repeated in this game, the adversary has no clue about which keys collide if she halts before making a forbidden query (i.e, those creating an abort). As a result, the only information the adversary can get from the experiment is many pairs of plaintext-input blocks and ciphertexts-output blocks corresponding to unknown keys. Hence, the leakage independence ensures that the leakage does not help her to guess collisions, for instance. In our mode of operation, we manage to create these non-aborting conditions in TGF and TVF.

Leakage function. By repeating nonces for different values of M_1 (and thus C_1) it is easy to mount a DPA on L_1 due to the computation of $L_1 + C_1$ in (the lower half of) the compression function of MenH – see Figure 8(a). By learning L_1, the adversary has the two plaintext-inputs of the last two BC calls in MenH and she can still change them at will. In turn, the key-inputs $K_1 + L_1 + 2D_1$ and $2L_1 + D_1$ will be leaked. We thus directly give the seed K_0 to the adversary in any leaking encryption or decryption query. For KDF and TGF, we simply give the leakage associated to the computation of the block cipher (and not the intermediate values). We already argued about the leakage of $\mathsf{PRF}[E]$ used in KDF and TGF in the previous sections and we also commented on why computing T does not leak it (except in encryption where T is the tag which is explicitly returned by the algorithm) since the ephemeral key X is refreshed given any final state (A, B), except a small constant number of times related to a maximum multi-collision parameter α that we control in the proof. We would have the same property for Y if it was not involved in the last verification $Z = E_S(Y)$ in decryption, where S is the candidate tag and $Z = E_T(Y)$. However, with many distinct S for a single prefix ciphertext, Z will leak by a DPA on the comparison check and thus $Y = E_S^{-1}(Z)$ can be discovered. Based on these premisces, we describe the *unbounded* leakage function pair $\mathsf{L} = (\mathsf{L}_{\mathsf{Enc}}, \mathsf{L}_{\mathsf{Dec}})$ used in our proofs as:

- $\mathsf{L}_{\mathsf{Enc}}(K, N, M)$: $\ell_0 = \mathsf{L}_{prf}(K, 0\|N)$ to compute K_0 and $\ell_1 = \mathsf{L}_{prf}(K, 1\|B)$ to compute X, the seed K_0 allowing to compute MenH offline on IV (K_0, L_0), where $L_0 = cst$ is a constant, and $\ell_{tgf} = \mathsf{L}_{tgf}(X, A)$ to compute T. Then, $\ell_{\mathsf{Enc}} = (\ell_0, \ell_1, K_0, \ell_{tgf})$.
- $\mathsf{L}_{\mathsf{Dec}}(K, N, C\|S)$: $\ell_0 = \mathsf{L}_{prf}(K, 0\|N)$ to compute K_0 and $\ell_1 = \mathsf{L}_{prf}(K, 1\|B)$ to compute X, the seed K_0 allowing to compute MenH offline on IV (K_0, L_0), and $\ell_{tvf} = \mathsf{L}_{tvf}(X, A)$ to compute T, Y and which includes Y, Z, where $E_T(Y) = Z$, to implicitly check whether $T = S$ without leaking T by verifying if $Z = E_S(Y)$, which can be done by the adversary. Then, $\ell_{\mathsf{Dec}} = (\ell_0, \ell_1, K_0, \ell_{tgf}, Y, Z)$.

The idea behind our integrity result with leakage is that the final state (A, B) is fresh throughout all the online queries, and so are the pairs (Y, Z). By showing that (Y, Z) is independent of all the other queries, the adversary is thus left with brute-forcing a compatible key-input S of the block cipher matching a plaintext-ciphertext pair (Y, Z).

Theorem 1 (CIML2)**.** *LR-BC2[E] with the leakage function pair* $\mathsf{L} = (\mathsf{L}_{\mathsf{Enc}}, \mathsf{L}_{\mathsf{dec}})$ *as defined above is CIML2 secure if the* $E : \{0,1\}^n \times \{0,1\}^n \mapsto \{0,1\}^n$ *can be modeled as an ideal cipher. More precisely, if* E *is an ideal cipher satisfying* (α, q_{on})*-leakage independence, and*

[15] These pairs correspond to final states which will all be distinct.
[16] In the CIML2 proof, we manage to always make fresh BC calls, which weakens the assumption.

$\mathsf{PRF}[E]$ *is leak-free,*[17] *then the CIML2 advantage is bounded as:*

$$\varepsilon_{\mathit{ciml2}} \leq \frac{4 \cdot 2^n}{(2^n - Q)^{\alpha+1}} \binom{Q}{\alpha+1} + \tilde{\varepsilon}_{\mathsf{MenH}} + \frac{2(\alpha^2+1)Q}{2^n - Q} \ ,$$

where α is a maximal multi-collision parameter, q is the number of offline ideal cipher queries, q_e is the number of leaking encryption queries, q_d is the number of leaking decryption queries, $q_{on} = q_e + q_d + 1$ is the total number of on-line queries, σ is the total number of message blocks involved in the online queries, and $Q = q + 7q_e + 9q_d + 4\sigma$ is the total number of ideal cipher calls excluding those implicitly computed in $\mathsf{PRF}[E]$.

To simplify the proof we give more power to the adversary by additionally returning Y and Z in any leaking encryption query. That is, even if the leakage differs in encryption and decryption, we decide to give both in the case of an encryption as the adversary may anyway make a leaking decryption query for the received valid ciphertext. That way, we can assume that the adversary never asks the decryption of a received valid ciphertext, and, conversely, the adversary never asks the encryption of the underlying message related to a ciphertext involved in a decryption query (since K_0 is equal in those cases with the same nonce, and it defines a bijection between the message blocks and the ciphertext blocks no matter what is the candidate tag); we will say the adversary never makes a *cross query*. As usual, we also assume that the adversary never repeats a query because it is useless (the leakage functions directly give all the information she can get with repetitions).

Proof. We use a sequence of modified CIML2 games where Game 0 is the proper CIML2 game and the last game is a dummy game where the adversary looses. In each game, Game i, W_i is the event that the adversary wins in that game.

In the state of MenH we call K-value (resp., L-value) the upper (resp., lower) half state value appearing in Figure 7. We also refer to $K+L+2D$-value (resp., $2L+D$-value) to denote the upper (resp., lower) key-input of the block cipher call made in MenH as shown in Figure 8(a). We use similar notations such as A-value (resp., B-value) for the upper (resp., lower) half final state, U-value for $U = \mathsf{PRF}[E]_K(1\|W)$, X-value for $X = U \oplus B$, Y-value, Z-value..., all of them denoted according to Figure 10.

Game 0: This is the actual CIML2 game as specified in Figure 12, but without associated data. Given a $(q, q_e, d_d \ldots)$-adversary $\mathcal{A}$ we select a random secret key $K \in \{0,1\}^n$. On an encryption query (N, M) with $M = (M_1, \ldots, M_\ell)$ in $\{0,1\}^{\ell n}$, for some non negative integer ℓ, we return $\mathsf{LEnc}_K(N, M) = (\mathsf{Enc}_K(N, M), \mathsf{L}_{\mathsf{Enc}}(K, N, M))$ as well as (Y, Z), where $\mathsf{Enc}_K(N, M) = C$ for some $C = (C_1, \ldots, C_\ell, T) \in \{0,1\}^{n(\ell+1)}$ with tag T. On a decryption query (N, C) with $C = (C_1, \ldots, C_\ell, S) \in \{0,1\}^{n(\ell+1)}$ with candidate tag S and for some for some non negative integer ℓ, we return $\mathsf{LDec}_K(N, C) = (\mathsf{Dec}_K(N, C), \mathsf{L}_{\mathsf{Dec}}(K, N, C))$, where $\mathsf{Dec}_K(N, C) = M$ for some $M = (M_1, \ldots, M_\ell)$ in $\{0,1\}^{\ell n}$ if the candidate tag S is valid, that is, it satisfies $E_S(Y) = Z$, where Y and $Z = E_T(Y)$ are computed in the TVF part of the decryption (and T is the real tag). Otherwise the decryption returns $\perp$ but $\mathcal{A}$ still gets the leakage ℓ_{Dec}. At the end of the game, $\mathcal{A}$ returns a nonce-ciphertext pair (N^*, C^*) and wins if it is valid and fresh, i.e., if the pair was never queried as an input to a leaking encryption. In that case, (N^*, C^*) is called a ciphertext forgery. We view the candidate ciphertext forgery as the (q_d+1)-th decryption query.

By definition we have $\Pr[W_0] = \Pr\left[\mathsf{PrivK}^{\mathsf{CIML2}}_{\mathcal{A},\mathsf{LR\text{-}BC\text{-}2,L}}(1^n) \Rightarrow 1\right]$.

Game 1: This is the CIML2 game except that we bring the following modification in the winning condition. Now, we also consider the adversary $\mathcal{A}$ to be unsuccessful if there is at least one $(\alpha+1)$-multi-collision on any of the PGV-like (n-bit output)

[17] The leakages ℓ_0, ℓ_1 are useless and can be removed from $\mathsf{L}_{\mathsf{Enc}}$ and $\mathsf{L}_{\mathsf{Dec}}$.

compression functions or in a (forward) BC output. If we call F_1 the event that the adversary wins in Game 0 but fails in this game we have $\Pr[W_1] - \Pr[W_0] \leq \Pr[F_1]$. Clearly, F_1 implies that there are some $(\alpha+1)$-multi-collisions in any underlying single-length compression function or the the block cipher in forward calls. Since we have the following compression functions and block cipher (forward) calls:

1. $f_1(h, m) = E_h(m) + m$ to compute (non-seed) K-values, L-values;
2. $f_2(h, m) = h + m + 2E_h(m)$ to compute $K+L+2D$-values;
3. $f_3(h, m) = 2m + E_h(m)$ to compute $2L+D$-values;
4. BC (including PRF) calls to compute K_0, U, Y, Z, T-values;

we can use for each case our multi-collision result of Appendix C, Theorem 2. We thus have $\Pr[F_1] \leq \frac{4 \cdot 2^n}{(2^n - Q)^{\alpha+1}} \binom{Q}{\alpha+1}$. We note that we do not consider cross-function collisions here. Also, an A-value (resp., a B-value) is a particular case of a K-value (resp.s an L-value). In the following games we have at most α-multi-collisions.

Game 2: This is as the previous game except that we deem the adversary A unsuccessful if a state-collision occurs during the game. If we denote F_2 the event that the adversary wins in Game 1 but fails in this game we have $\Pr[W_1] - \Pr[W_2] \leq \Pr[F_2]$.

Bounding $\Pr[F_2]$: We first consider initial state collision. Since the initial state $(K_1, L_1) = \mathsf{Menh}(K_0, L_0, N)$, distinct nonce N implies distinct (K_1, L_1) except if we have a MenH-collision. As a second case, we deal with the probability that an internal state (K', L') collides with an IV (K_0, cst). This can only happen if $L' = cst$ which implies finding a preimage of cst for the compression function f_1. We thus have the term $Q/(2^n - Q)$, where $Q \leq 2^{n-c}$ is the total number of calls to the ideal cipher. It remains to bound the probability that internal states collide, but this is directly related to MenH-collision. Therefore, $\Pr[F_2] \leq \tilde{\varepsilon}_{\mathsf{MenH}} + Q/(2^n - Q)$ since $\varepsilon_{\mathsf{MenH}} \leq \tilde{\varepsilon}_{\mathsf{MenH}} + \varepsilon_{\mathsf{mu\text{-}coll}}$, and the multi-collisions of Mennink's proof implies F_1.

Game 3: In this game we replace the output of $\mathsf{PRF}[E]_K(b||\cdot)$ used in KDF (with $b = 0$) and TGF/TVF (with $b = 1$) by an independent and secret random function in a leakage respectful manner. Since we rely on the idealized assumption that $\mathsf{PRF}[E]$ is leak-free and independent, we directly have $\Pr[W_2] = \Pr[W_3]$.

Game 4: In this game we bring yet another modification to the winning conditions. To be successful, the adversary must fulfil the winning conditions of Game 3 but she fails if she creates some final states (A, B) and (A, B') such that their respective X-value, denoted X and X', are such that $X + X' \in \{0, 1\}$ (using the finite field notation). That is, the adversary does not win if she forces the re-use of some computation $E_X(A)$ or $E_{X+1}(A)$ in an online query. If we denote F_4 the event that the adversary wins in Game 3 but fails in this game we have $\Pr[W_3] - \Pr[W_4] \leq \Pr[F_4]$.

Bounding $\Pr[F_4]$: Given a final upper state A, let consider all the final states $(A, B_1), \ldots, (A, B_a)$ with that A-value. Since A is the output of a consecutive computation of two compression functions $f_1(f_2(K_\ell, L_\ell), C_\ell)$ we know that $a \leq \alpha^2$. Indeed, since their is no state collision from Game 2, the previous states are all distinct except if the corresponding queries only differ on the last block C_ℓ (the prefix must be equal). In the latter case, that means that we must have a (proper) collision on f_1. In the former case, i.e., when the previous states differ, we must have a (proper) collision on f_2. That means that we have no more than α^2 repetitions of A (since the adversary does not repeat a query and never makes a cross query). With this in mind we can now consider several cases.

First, we deal with the case where some of these A pairs collide on a W-value. Consequently, those W-colliding pairs lead to a common U-value (the PRF output). But then the X-value differs since $X_i = U + B_i$ with necessarily distinct B_i for

the appropriate index. So, between the pairs (A, B_i) and (A, B_j) that collide on a same W, $X_i + X_j \neq 0$. Now it looks like we are stuck if $B_i + B_j = 1$ as it implies $X_i + X_j = 1$, but we next show that W-collision are very unlikely in that case.

Second, we bound the probability that there is a special f_1 collision of the form $f_1(h, m) = f_1(h, m')$ and $m + m' = 1$. That is $E_h(m) + E_h(m') = 1$. Assuming that the BC triple $(h, m, E_h(m))$ is already defined in the view, the probability that a forward ideal cipher query $(h, m + 1, \cdot)$ or a backward query $(h, \cdot, E_h(m) + 1)$ defines the triple $(h, m + 1, E_h(m) + 1)$ is bounded by $1/(2^n - Q)$. By taking the union bound on all the possible such queries, we find at most $Q/2 \cdot 1/(2^n - Q)$ (since each query defines a single triple which defines a single try).

Third, we show that among the respective X-values $X_1, \ldots, X_a$ associated to the pairs $(A, B_1), \ldots, (A, B_a)$, those related to non W-colliding pairs imply $X_i + X_j \notin \{0, 1\}$, except with a very low probability. To show this, we bound the probability that $X_{i+1} = \mathsf{PRF}[E](1\|W_{i+1}) + B_{i+1}$ falls either on X_j or on $X_j + 1$, for some $1 \leq j \leq i$, if $W_j \neq W_{i+1}$, for any $1 \leq i \leq a-1$. Given the index i, such a probability is lower than $2i/(2^n - Q)$ since we have at most $2i$ targets of the form $X_j + B_{i+1}$ or $X_j + B_{i+1} + 1$, with $1 \leq j \leq i$, for the random value $U_{i+1} = \mathsf{PRF}[E](1\|W_{i+1})$. Given A, by summing on the index i we get the lower bound $\sum_{i=1}^{a-1} 2i/(2^n - Q) = 2/(2^n - Q) \cdot a(a-1)/2$. Now, considering all the final upper half state A-values, we will show that the union probability is lower than $q_{on}/\alpha^2 \cdot \alpha^2(\alpha^2 - 1)/(2^n - Q)$ as the probability is maximal when all the A-values are repeated α^2 times with exactly q_{on}/α^2 distinct A-values. Let $A_1, \ldots, A_\nu$ be all the distinct A-values that are respectively repeated $a_1, \ldots, a_\nu$ times in a final state. We can compute the lower bound as $\sum_{i=1}^{\nu} a_i(a_i - 1)/(2^n - Q) = \sum_{j=1}^{\alpha^2} \sum_{i:a_i=j} j(j-1)/(2^n - Q) \leq (\alpha^2 - 1)/(2^n - Q) \sum_{j=1}^{\alpha^2} \sum_{i:a_i=j} j$, that is $q_{on}(\alpha^2 - 1)/(2^n - Q)$. Hence, $\Pr[F_4] \leq \frac{Q}{2(2^n-Q)} + \frac{q_{on}(\alpha^2-1)}{2^n-Q}$. From now on we can assume that the computations of $E_X(A), E_{X+1}(A)$ are fresh in the online queries.

Game 5: This game is as Game 4 except that we add one more failure event F_5 which causes the adversary $\mathcal{A}$ to loose the game if she wins in Game 4 but manages to make an (offline) ideal cipher query that (re)produces a triple (X, A, T) or $(X + 1, A, Y)$ for some pair (A, X) involved in an online query. We want to discard the case where $\mathcal{A}$ brute-forces X from offline queries and we have $\Pr[W_4] - \Pr[W_5] \leq \Pr[F_5]$.

Bounding $\Pr[F_5]$: First, we note that the leakage on U, X and $X+1$ is small so that it does not reveal anything useful to the adversary. Indeed, as there are at most α multi-collisions on a W-value, each $U = \mathsf{PRF}[E]_K(1\|W)$ is only explicitly (but secretly) used twice in the online computation of $E_{U+B_i}(A_i)$ and $E_{U+B_i+1}(A_i)$ (where $X_i = U + B_i$) and for at most α indexes of the final states. Each U-value is only explicitly called 2α times in the online queries. Then, the (α, q_{on})-leakage independence of E ensures that the leakage does not give more information than E does.

Second, considering (again) all the A-values $A_1, \ldots, A_\nu$ with their respective multiplicity $a_1, \ldots, a_\nu$, we can show that all the $2q_{on}$ pairs $(A_i, T_{ij}), (A_i, Y_{ij})$, where T_{ij}, Y_{ij} denotes all the T-values and the Y-values coming from A_i, for $1 \leq j \leq a_i$, are pairwise distinct except with probability $q_{on}(2\alpha^2 - 1)/(2^n - q_{on})$. Note that the adversary $\mathcal{A}$ only gets both pairs $(A_i, T_{ij}), (A_i, Y_{ij})$ if the corresponding query is a leaking encryption query. The leakage in decryption only reveals the pair (A_i, Y_{ij}). Now, if F_5 occurs $\mathcal{A}$ manages to find a key X_{ij} (even if it is $X_{ij} + 1$, at the first place) that might be involved in a decryption query and $\mathcal{A}$ then knows $E_{X_{ij}}(A_i) = T_{ij}$, which leads to a ciphertext forgery. To increase her advantage in finding X_{ij}, $\mathcal{A}$ should focus on (A_i, Y_{ij}) since for distinct i-indexes she must be lucky to find an X-collision (or more exactly, figure out the collision but the offline queries are independent from X-values before that time): it is more efficient to put all her power on the corresponding

i-index. Now, we show that $\mathcal{A}$ should focus on the A_i's with the highest multiplicities. Her goal is to make an offline query $(X'+1, A_i)$ for many keys X' and hope falling on Y_{ij}, for some $1 \leq j \leq a_i \leq \alpha^2$. In that case she can guess that the X' "hitting" Y_{ij} is actually X_{ij}. For each try where she has to bind the A-value in each ideal cipher query before getting the result, the probability of such a hit is maximal when a_i is maximal since a single try has more chance to hit more targets. More precisely, if q_i is the number of time $\mathcal{A}$ makes a query with A_i, then if $a_i \leq a_{i'}$, for a fixed value of $q_i + q_{i'}$, $q_i a_i/(2^n - Q) + q_{i'} a_{i'}/(2^n - Q) \leq (q_i - 1)a_i/(2^n - Q) + (q_{i'} + 1)a_{i'}/(2^n - Q)$. Since E is an ideal cipher, $\mathcal{A}$'s success is independent of the chosen i-index among those for which a_i is maximal because she does not know how many times the X-key she is looking for has already been used. Anyway, we assume that $a_1 = \alpha^2$ and that the keys X_{1j}, for $1 \leq j \leq \alpha^2$ are those that are already the most involved in the ideal cipher queries. Hence, the probability to find a good key X_{1j} is bounded by $q\alpha^2/(2^n - Q)$. We stress that making inverse ideal cipher call is less efficient for $\mathcal{A}$ as there are at most α multi-collisions on some Y-values. Finding an X-key is thus more likely in forward calls where A-values can be repeated up to α^2 times. From now on, forging a ciphertext cannot be made by guessing an X-key and thus by computing explicitly an honest tag T for the decryption of an invalid ciphertext.

Game 6: Game 6 is like game 5 except that a winning adversary in game 5 is no more considered successful if she finds a forgery by (purely) guessing a valid candidate tag S. More precisely, guessing S in that case means that $\mathcal{A}$ defines a block cipher triple (S, Y, Z), i.e., $E_S(Y) = Z$ where $Z = E_T(Y)$, for known Y, Z, by making an ideal cipher query either in a forward call of the form $(S, Y, \cdot)$ or in a backward call of the form $(S, \cdot, Z)$. Oviously, $\Pr[W_5] - \Pr[W_6] \leq \Pr[F_6]$.

Bounding $\Pr[F_6]$: as there are at most α multi-collisions on any Y-value and any Z-value there is no distinction to make between forward and backward calls as long as the adversary focuses on the values with the highest occurrence. A similar analysis used in the previous transition show that $\Pr[F_6] \leq q\alpha/(2^n - Q)$.

In game 6, the probability that adversary computes a ciphertext forgery is 0. That is, $\Pr[W_6] = 0$, and we have:

$$\Pr[W_0] \leq \frac{4 \cdot 2^n}{(2^n - Q)^{\alpha+1}} \binom{Q}{\alpha+1} + \tilde{\varepsilon}_{\mathsf{MenH}} + \frac{3Q/2 + q_{on}(\alpha^2 - 1) + q\alpha(\alpha+1)}{(2^n - Q)} \cdot$$

Eventually, $q_{on}(\alpha^2 - 1) + q\alpha(\alpha+1) \leq 2\alpha^2 Q$. Hence, the bound. □

Assuming that $Q \leq 2^{n-c}$, and $\alpha + 1 \leq n/c$, for some $1 \leq c \leq n$, the multi-collision term is bounded by:

$$\frac{4}{(\alpha+1)!} \left(\frac{2^c}{2^c - 1}\right)^{\alpha+1} \cdot \frac{Q^{\alpha+1}}{2^{n\alpha}} \cdot$$

For $n = 128$, and $c = 8$, the above expression is less than $8/(\alpha+1)! \cdot Q^{\alpha+1}/2^{n\alpha}$, for all $1 \leq \alpha \leq 15$. In Appendix D, we show $\varepsilon^*_{\mathsf{MenH}} \leq (2t_2^2 + 7t_2 + 18)Q/(2^n - Q)$, where we have to take $t_2 = \alpha$. This compresses the remaining terms of our bound to:

$$\frac{256}{255} \frac{(4\alpha^2 + 7\alpha + 20)\, Q}{2^n} \cdot$$

Now, let $Q = 2^{\beta n}$, with $0 < \beta < 1$ and $\beta n \leq n - 8$. For $\alpha = 2$, the advantage is bounded by $2^{3\beta n - 2n + 4} + 2^{\beta n - n + 5}$, so $\beta \leq 0.65$ and the mode supports up to $Q \leq 2^{83}$ queries. For $\alpha = 3$, the advantage is bounded by $2^{4\beta n - 3n - 2} + 2^{\beta n - n + 7}$, so $\beta \leq 3/4$ and the mode supports up to $Q \leq 2^{96}$ queries. Similarly, for $\alpha = 4$, we can go up to $Q \leq 2^{102}$ queries.

And for $\alpha \geq 5$, the above equation is bounded by $7\alpha^2 2^{\beta n - n}$ and the multi-collision term remains dominant. Therefore, the bit security for $5 \leq \alpha \leq 15$ is up to $\alpha/(\alpha+1)n$.

So overall, and as already outlined in the previous section, the integrity of our construction increases with the multi-collision parameter α, but increasing α also implies a stronger hardware assumption since we assume SPA security with $q_{\mathsf{SPA}} = 2\alpha$ queries. Interestingly, our detailed analysis is slightly better than the estimates of Section 5.1. For example, we reach 96-bit security with $\alpha = 3$ rather than $\alpha = 4$ (and for $\alpha = 4$ we reach 102-bit security). That is because, we actually tolerate α multi-collisions. We thus "forbid" $(\alpha+1)$-multi-collisions, and bound the probability that it eventually happens. These results therefore show that for actual security parameters as available on low-cost embedded devices, LR-BC-2 can provide strong concrete integrity guarantees.

Incidentally, we note that the LR-MAC of Figure 10 also improves the results of Guo et al. at FSE 2020 [GSWY19], which require two strongly protected block cipher calls where we only use one call to the LR-PRF (which could be replaced by a strongly protected block cipher, if not aiming for exploiting hardware co-processors as in this work).

5.3 Confidentiality in the presence of leakage: heuristic analysis

The confidentiality with leakage of LR-BC-2 follows the standard approach of previous leakage-resistent modes of operations. As a heuristic argument, it is easy to see that ignoring decryption leakages and assuming fresh nonces, every message block is encrypted with a fresh key. So at high-level, and using the simplified assumptions of [BBC+20], the security of our mode reduces to the SPA security of a single message block.

A bit more formally, to prove the CCAmL1 security of our mode, which stands for CCA security with misuse-resilience and leakage-resistance [GPPS19], we can rely on the hard-to-invert leakage assumption, which follows [YSPY10] and was used in [BGP+20] for TEDT. In the CCAmL1 game, the adversary is not granted access to a leaking decryption oracle but only to a black-box decryption. We briefly sketch the leakage function in encryption queries. For a fresh nonce in encryption, we can rely on Mennink's indifferentiability result for MenH, recalled in Appendix D in order to establish that the initial state (K_1, X_1) is independent and random up to the birthday bound (which is unavoidable in leakage confidentiality with the current state of cryptographic knowledge). We stress that a linear map $\mathcal{L}$ which satisfies the collision-resistance condition automatically satisfies the indifferentiability condition. Now, we also have to show that the initial state is kept secret and we will explain why any internal state will remain sufficiently hidden when nonces are not repeated (as required only for the computation of the challenge ciphertext).

For the first ephemeral key K_1, a first part of the leakage comes from the first call to MenH inside KDF as shown in Figure 8(b). The second part of the leakage comes from the next call to MenH and to $E_{K_1}(L_1 \oplus \theta)$ in the bulk of the computation. Assuming that the first part remain safe (since $K_1 = E_{K_0+L_0+2D_0}(N)$ is computed once and a single time with K_0), the leakage reduces to the second part, and more precisely to $E_{K_1}(L_1)$, $E_{K_1}(L_1 \oplus \theta)$ and the key-input $K_1+L_1+2D_1$. So K_1 is only used a very small constant number of times. Therefore, the protection of the internal ephemeral key material can be propagated and the $E_K(L \oplus \theta)$ value remains hidden enough and random. Now, each block of message, i.e., M-value, is only XORed with that previous value. In the hard-to-invert leakage model, we can iterate the argument until the final state. Finally, given a final state, TGF is also made independent of the message processing as argued in the integrity proof while the adversary here has much less information on the internal states. Therefore, a single execution on a final state $(A, B) := (K_{\ell+1}, X_{\ell+1})$ does not leak much about it and the intermediate computation in TGF also remains safe as already argued in the proof of CIML2.

We finally note that while the LR-BC-2 mode of operation ensures the strongest integrity in front of decryption leakages, formalized by the CIML2 notion, its confidentiality guarantees are limited to encryption leakages (i.e., CCAmL1). In order to obtain CCAmL2 security (where the 2 stands for decryption leakages), one option is to leverage a second pass, and to adapt the TEDT mode of operation to the block cipher case. We represent the resulting LR-BC-3 mode of operation in Figure 13: it works by replacing Hirose's hash function [Hir06] by Mennink's one [Men17] in the scheme of Berti et al. [BGP+20].

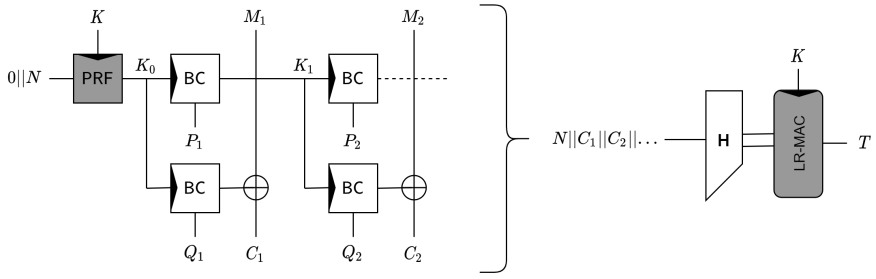


Figure 13: The LR-BC-3 mode of operation.

Alternatively, we observe that a simple trick to provide some heuristic confidentiality with decryption leakages is to limit the number of incorrect verifications a chip can perform at the protocol level (and to freeze it afterwards). In this case, the number of measurements a side-channel adversary can perform will be limited by this number of incorrect verifications for each target chip. Yet, in case multiple chips encrypt with the same key, it may still happen that a sufficient number of traces can theoretically be collected. In this respect, we observe that multi-user security could be of interest to limit such multi-chip attacks without requiring the distribution of more secret key material: by using a public key P in combination with the nonce $0||N$ in Figure 8(b), one could send firmware updates encrypted based on different public keys for different chips.

In the following section, we conclude the paper by analyzing the implementation cost of these different solutions and compare them with the benchmarking of [USS+20].

6 Performance evaluation

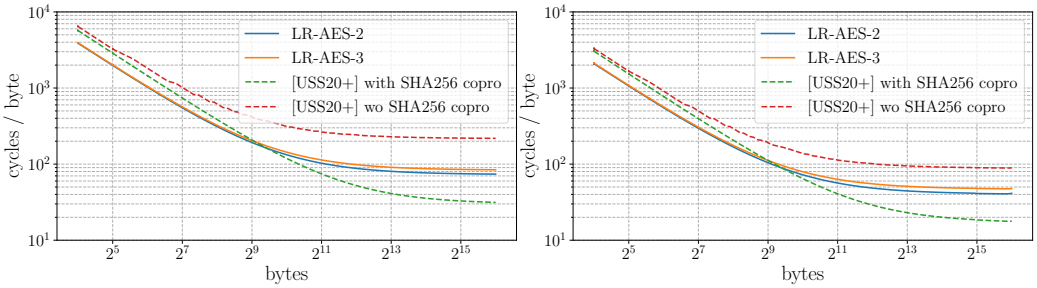
We next detail the performances obtained by our two main LR-BC modes of operation. We first compare them in terms of block cipher calls. Next, we detail the results obtained on the two previously mentioned targets and compare them with [USS+20].

6.1 BC calls count for LR-BC variants

In Table 1, we detail the number of AES calls required to encrypt l-block messages with LR-BC-2, LR-BC-3 and [USS+20]. For the two first modes, decryption requires 3 additional block cipher calls for the tag verification. We separate these figures for the three main steps of the modes, namely the KDF, bulk computation and TGF.

Table 1: Number of block cipher calls to encrypt l-block messages.

Scheme	Grade	KDF	Bulk	TGF	Total
LR-BC-2	2	$130+3$	$4 \cdot l$	$130+2$	$260+4 \cdot l+5$
LR-BC-3	3	130	$5 \cdot l+3$	$130+2$	$260+5 \cdot l+5$
[USS+20]	?	129	$2 \cdot l+$?	257	$386+2 \cdot l+$?



(a) Cortex-M33 (32-bit arch) **(b)** Cortex-M4 (128-bit arch)

Figure 14: Performances of LR-AES modes -O3 -flto optimizations and $n_b = 1$.

The main observations from this table are twofold. First, the constant part of the cost of these modes is dominated by the execution of the LR-PRF. In this respect, the LR-BC-2 and LR-BC-3 modes are similar since they only use the LR-PRF on 258 bits. By contrast, the CHES 2020 construction has a larger constant cost due to its more expensive finalization. Indeed, it uses the LR-PRF on 384 bits [USS+20]. Second, our one-pass modes asymptotically require $4 \cdot l$ calls to the block cipher and LR-BC-3 requires $5 \cdot l$ such calls, which directly reflects their security grade. Evaluating [USS+20] is more difficult for this part, since it requires $2 \cdot l$ calls to a block cipher and a call to a hash function on all the message (which can be implemented in software or hardware as discussed next).

6.2 LR-AES encryption throughput

We start by describing the two optimizations we used in order to implement the LR-BC mode variants based on AES coprocessors, that we next denote as LR-AES. Note that the CHES 2020 mode was instantiated with both the AES and SHA-256, as in [USS+20].[18] Since these schemes heavily rely on calls to the AES, the first guideline we followed is to use the ECB mode (available on most coprocessors) whenever multiple (e.g., 2 in our case) calls to the AES with the same key are needed. This trick allows saving at least the cost of writing the key to the coprocessor, as well as its initialization. Therefore, it significantly speeds up the bulk computation of all the variants of LR-BC as well as [USS+20]. In addition, if the coprocessor has a direct memory access (DMA), it will be able to run the two encryptions without relying on any software instruction. The second optimization is to work on explicitly aligned words (i.e., `uint32_t`): this ensures that the compiler will operate on machine words instead of bytes. Roughly, the combination of these optimizations lead to a $\approx 50\%$ improvement of the throughputs reported in [USS+20].

The curves in Figure 14 depict the encryption throughput ([cycles/byte]) for variants of LR-AES and [USS+20]. By comparing Figure 14a with Figure 14b, we notice that the Cortex-M4 systematically offers a better throughput than the Cortex-M33 thanks to its faster AES coprocessor (with a 128-bit architecture). For the rest, the comparative results are similar on both targets and follow the trends outlined in Table 1.

More into the details, for short messages (e.g., 16 bytes), the cost of the KDF and TGF dominates the cycle count. LR-AES variants reach the better throughput. Indeed,

[18] In the C implementation proposed by [USS+20] and accessible at https://github.com/Fraunhofer-AISEC/leakres-aead-microcontroller/blob/master/host/src/, the 256-bit digest provided by the hash function is truncated to 128 bits that are then fed to the LR-PRF. This leads to a possibility of forgery by looking for a collision on these 128 bits, which has a cost of 2^{64} offline queries to SHA-256. Hence, we report the results of a more secure version that does not truncate SHA256's output. In the same implementation, the tag comparison is done with `memcmp`, of which the execution time depends on the inputs to compare. This leads to a direct timing side-channel leakage that can be exploited by an adversary to forge tags. Listing 1 is constant time avoiding this source of leakage.

they require $\approx 2 \cdot 130$ calls to the AES for these steps, while [USS+20] require ≈ 128 additional calls. For long messages, the cost of the KDF and TGF are amortized and the bulk computation dominates the cycle count. Therefore, the Grade-2 variants of LR-AES give the best throughputs. The Grade-3 variant pays one additional call to the block cipher per message block. As for the comparison with [USS+20], it highly depends on the presence of a hardware accelerator for SHA-256. When available, its throughput is ≈ 2.3 better than our Grade-2 designs (but without strong integrity guarantees in the presence of leakage due to the attack in Subsection 3.2). If there is no hardware accelerator for the hash function, then LR-AES-2 is better with a factor ≈ 2.94 (resp., ≈ 2.17) on the ARM Cortex-M33 (resp., Cortex-M4). We note that because of LR-AES-3 and [USS+20] are two-pass, the entire ciphertext must be stored in memory when decrypting (it is not the case for LR-AES-2 which is online, but only offers CCAmL1 confidentiality).

Overall, these results suggest that the high cost of the LR-PRF will be quite well amortized for actual software updates, and that hardware coprocessors enable making the $4 \cdot l$ / $5 \cdot l$ block cipher calls per message block of our modes acceptable in practice. For example, the firmware of our ARM Cortex-M33 has a size of 21,184 bytes and the one of our ARM Cortex-M4 has a size of 19,064 bytes. Even with the most conservative security parameter for the LR-PRF ($n_b = 1$), encrypting this amount of data is sufficient to reach a good amortization (as shown in Figure 14). Also, considering the maximum clock frequencies of our two target devices, decrypting their ciphertext with LR-AES-2 will take 0.014 seconds (resp., 0.0043 seconds) on the ARM Cortex M33 (resp., M4).

Source code for LR-BC-2 is available here: https://github.com/cmomin/lr-bcx.

7 Conclusions

Securing low-cost embedded devices against side-channel attacks is a challenging task. It is for example well-known that implementing standard countermeasures like masking and shuffling hardly gives strong guarantees without a sufficient noise level, that may not be readily available in such devices. This fact naturally suggests the use of leakage-resilient modes of operation as an interesting alternative to these countermeasures, especially when hardware coprocessors with bounded leakage are available. A previous work of Unterstein et al. proposed one construction in this direction, taking secure firmware update as an exemplary motivation. In this work, we first showed that this proposal ignores one important attack vector against the integrity of their scheme (namely the leakage of the tag verification) and as a result, it does not offer the claimed guarantees. We therefore propose new modes of operation (LR-BC-2 and LR-BC-3) that fix this issue.[19] Next, we revisit the concrete side-channel security evaluation of the leakage-resilient PRF that is leveraged by our modes (and the one of Unterstein et al.) and show that it is quite sensitive to variations of target device and measurement setup. We conclude that such modes are interesting candidates for leveraging the existing ecosystem of embedded micro-controllers, but that their security parameters should be selected conservatively.

Acknowledgments

Thomas Peters and François-Xavier Standaert are respectively research associate and senior research associate of the Belgian Fund for Scientific Research (F.R.S.-FNRS). This work has been funded in parts by the EU through the ERC project SWORD (724725) and the Walloon Region through the Win2Wal project PIRATE.

[19] We note that the issue of leakage-resilient value comparison and its application to message authentication is also discussed in a recent and independent work of Dobraunig and Mennink [DM21].

References

[APSQ06] Cédric Archambeau, Eric Peeters, François-Xavier Standaert, and Jean-Jacques Quisquater. Template attacks in principal subspaces. In *CHES*, volume 4249 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2006.

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

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

[BHM+19] Olivier Bronchain, Julien M. Hendrickx, Clément Massart, Alex Olshevsky, and François-Xavier Standaert. Leakage certification revisited: Bounding model errors in side-channel security evaluations. In *CRYPTO (1)*, volume 11692 of *Lecture Notes in Computer Science*, pages 713–737. Springer, 2019.

[BK09] Alex Biryukov and Dmitry Khovratovich. Related-key cryptanalysis of the full AES-192 and AES-256. In *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2009.

[BKP+18] Francesco Berti, François Koeune, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Ciphertext integrity with misuse and leakage: Definition and efficient constructions with symmetric primitives. In *AsiaCCS*, pages 37–50. ACM, 2018.

[BMOS17] Guy Barwell, Daniel P. Martin, Elisabeth Oswald, and Martijn Stam. Authenticated encryption in the face of protocol and side channel leakage. In *ASIACRYPT (1)*, volume 10624 of *Lecture Notes in Computer Science*, pages 693–723. Springer, 2017.

[BÖS11] Joppe W. Bos, Onur Özen, and Martijn Stam. Efficient hashing using the AES instruction set. In *CHES*, volume 6917 of *Lecture Notes in Computer Science*, pages 507–522. Springer, 2011.

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

[BPS15] Guy Barwell, Daniel Page, and Martijn Stam. Rogue decryption failures: Reconciling AE robustness notions. In *IMACC*, volume 9496 of *Lecture Notes in Computer Science*, pages 94–111. Springer, 2015.

[BRSS10] John Black, Phillip Rogaway, Thomas Shrimpton, and Martijn Stam. An analysis of the blockcipher-based hash functions from PGV. *J. Cryptol.*, 23(4):519–545, 2010.

[BS20] Olivier Bronchain and François-Xavier Standaert. Side-channel countermeasures' dissection and the limits of closed source security evaluations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(2):1–25, 2020.

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.

[DEM+17] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, and Thomas Unterluggauer. ISAP - towards side-channel secure authenticated encryption. *IACR Trans. Symmetric Cryptol.*, 2017(1):80–105, 2017.

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

[DJS19] Jean Paul Degabriele, Christian Janson, and Patrick Struck. Sponges resist leakage: The case of authenticated encryption. In *ASIACRYPT (2)*, volume 11922 of *Lecture Notes in Computer Science*, pages 209–240. Springer, 2019.

[DM21] Christoph Dobraunig and Bart Mennink. Leakage resilient value comparison with application to message authentication. *IACR Cryptol. ePrint Arch.*, 2021:402, 2021.

[DP08] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *FOCS*, pages 293–302. IEEE Computer Society, 2008.

[DP10] Yevgeniy Dodis and Krzysztof Pietrzak. Leakage-resilient pseudorandom functions and side-channel attacks on feistel networks. In *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 21–40. Springer, 2010.

[FPS12] Sebastian Faust, Krzysztof Pietrzak, and Joachim Schipper. Practical leakage-resilient symmetric cryptography. In *CHES*, volume 7428 of *Lecture Notes in Computer Science*, pages 213–232. Springer, 2012.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

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

[GSWY19] Chun Guo, François-Xavier Standaert, Weijia Wang, and Yu Yu. Efficient side-channel secure message authentication with better bounds. *IACR Trans. Symmetric Cryptol.*, 2019(4):23–53, 2019.

[Hir06] Shoichi Hirose. Some plausible constructions of double-block-length hash functions. In *FSE*, volume 4047 of *Lecture Notes in Computer Science*, pages 210–225. Springer, 2006.

[Hot31] Harold Hotelling. The generalization of student's ratio. *The Annals of Mathematical Statistics*, 2(3):360–378, 1931.

[Jou04] Antoine Joux. Multicollisions in iterated hash functions. application to cascaded constructions. In *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 306–316. Springer, 2004.

[Koc05] Paul Kocher. Design and validation strategies for obtaining assurance in countermeasures to power analysis and related attacks. In *Proceedings of the NIST Physical Security Workshop*, volume 46, 2005.

[KS20] Juliane Krämer and Patrick Struck. Leakage-resilient authenticated encryption from leakage-resilient pseudorandom functions. In *COSADE*, volume 12244 of *Lecture Notes in Computer Science*, pages 315–337. Springer, 2020.

[Man04] Stefan Mangard. Hardware countermeasures against DPA ? A statistical analysis of their effectiveness. In *CT-RSA*, volume 2964 of *Lecture Notes in Computer Science*, pages 222–235. Springer, 2004.

[Men15] Bart Mennink. Optimally secure tweakable blockciphers. In *FSE*, volume 9054 of *Lecture Notes in Computer Science*, pages 428–448. Springer, 2015.

[Men17] Bart Mennink. Optimal collision security in double block length hashing with single length key. *Des. Codes Cryptogr.*, 83(2):357–406, 2017.

[MSJ12] Marcel Medwed, François-Xavier Standaert, and Antoine Joux. Towards super-exponential side-channel security with efficient leakage-resilient prfs. In *CHES*, volume 7428 of *Lecture Notes in Computer Science*, pages 193–212. Springer, 2012.

[MSNF16] Marcel Medwed, François-Xavier Standaert, Ventzislav Nikov, and Martin Feldhofer. Unknown-input attacks in the parallel setting: Improving the security of the CHES 2012 leakage-resilient PRF. In *ASIACRYPT (1)*, volume 10031 of *Lecture Notes in Computer Science*, pages 602–623, 2016.

[PGV93] Bart Preneel, René Govaerts, and Joos Vandewalle. Hash functions based on block ciphers: A synthetic approach. In *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 368–378. Springer, 1993.

[Pie09] Krzysztof Pietrzak. A leakage-resilient mode of operation. In *EUROCRYPT*, volume 5479 of *Lecture Notes in Computer Science*, pages 462–482. Springer, 2009.

[PSG16] Romain Poussier, François-Xavier Standaert, and Vincent Grosso. Simple key enumeration (and rank estimation) using histograms: An integrated approach. In *CHES*, volume 9813 of *Lecture Notes in Computer Science*, pages 61–81. Springer, 2016.

[PSV15] Olivier Pereira, François-Xavier Standaert, and Srinivas Vivek. Leakage-resilient authentication and encryption from symmetric cryptographic primitives. In *CCS*, pages 96–108. ACM, 2015.

[Rog02] Phillip Rogaway. Authenticated-encryption with associated-data. In *CCS*, pages 98–107. ACM, 2002.

[SA08] François-Xavier Standaert and Cédric Archambeau. Using subspace-based template attacks to compare and combine power and electromagnetic information leakages. In *CHES*, volume 5154 of *Lecture Notes in Computer Science*, pages 411–425. Springer, 2008.

[SLP05] Werner Schindler, Kerstin Lemke, and Christof Paar. A stochastic model for differential side channel cryptanalysis. In *CHES*, volume 3659 of *Lecture Notes in Computer Science*, pages 30–46. Springer, 2005.

[SM16] Tobias Schneider and Amir Moradi. Leakage assessment methodology - extended version. *J. Cryptogr. Eng.*, 6(2):85–99, 2016.

[SPY+10] François-Xavier Standaert, Olivier Pereira, Yu Yu, Jean-Jacques Quisquater, Moti Yung, and Elisabeth Oswald. Leakage resilient cryptography in practice. In *Towards Hardware-Intrinsic Security*, Information Security and Cryptography, pages 99–134. Springer, 2010.

[STKT08] Kazuhiro Suzuki, Dongvu Tonien, Kaoru Kurosawa, and Koji Toyota. Birthday paradox for multi-collisions. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 91-A(1):39–45, 2008.

[UHS+18] Florian Unterstein, Johann Heyszl, Fabrizio De Santis, Robert Specht, and Georg Sigl. High-resolution EM attacks against leakage-resilient prfs explained - and an improved construction. In *CT-RSA*, volume 10808 of *Lecture Notes in Computer Science*, pages 413–434. Springer, 2018.

[UHSS17] Florian Unterstein, Johann Heyszl, Fabrizio De Santis, and Robert Specht. Dissecting leakage resilient prfs with multivariate localized EM attacks - A practical security evaluation on FPGA. In *COSADE*, volume 10348 of *Lecture Notes in Computer Science*, pages 34–49. Springer, 2017.

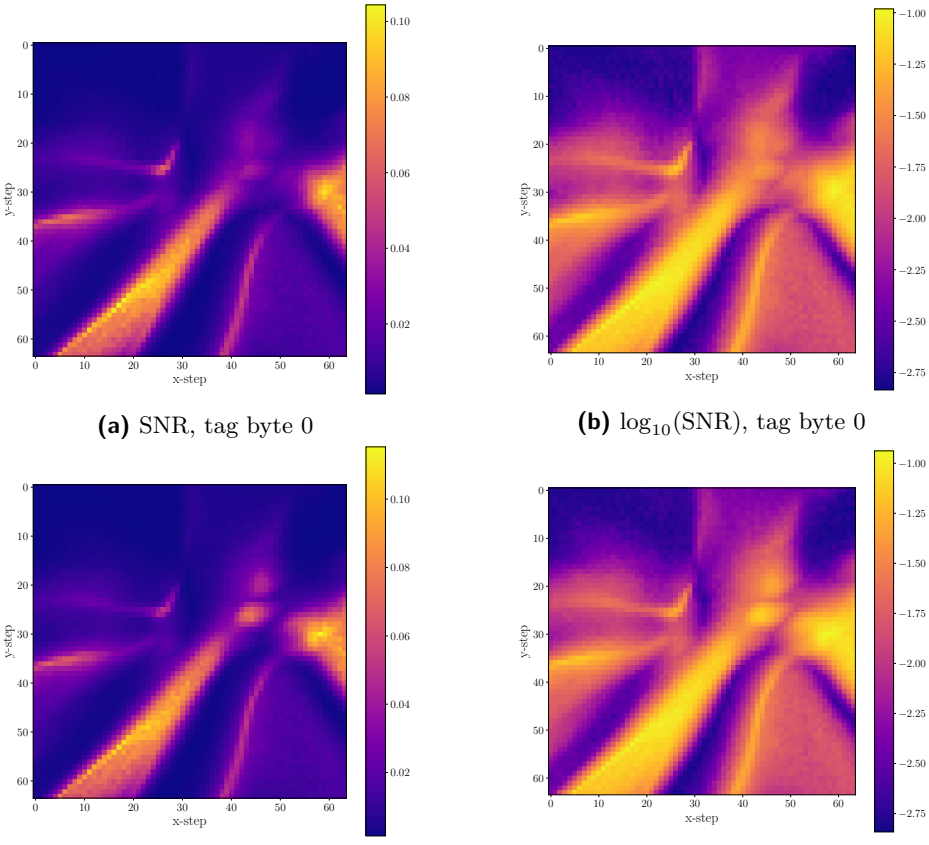
[USS+20] Florian Unterstein, Marc Schink, Thomas Schamberger, Lars Tebelmann, Manuel Ilg, and Johann Heyszl. Retrofitting leakage resilient authenticated encryption to microcontrollers. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(4):365–388, 2020.

[YSPY10] Yu Yu, François-Xavier Standaert, Olivier Pereira, and Moti Yung. Practical leakage-resilient pseudorandom generators. In *CCS*, pages 141–151. ACM, 2010.

A Additional figures

```
ldr r3, [r1]
ldr ip, [r0]
ldr r2, [r0, #4]
eor ip, ip, r3
ldr r3, [r1, #4]
eor r2, r2, r3
orr r2, r2, ip
ldr r3, [r0, #8]
ldr ip, [r1, #8]
ldr r0, [r0, #12]
ldr r1, [r1, #12]
eor r3, r3, ip
orr r3, r3, r2
eor r2, r0, r1
orrs r3, r2, r3
moveq r0, #1
movne r0, #0
bx lr
```

Listing 2: Tag verification ASM (from GCC)



(a) SNR, tag byte 0

(b) $\log_{10}$(SNR), tag byte 0

(c) SNR, tag byte 1

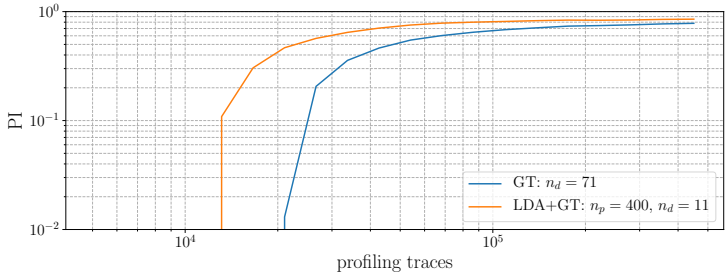
(d) $\log_{10}$(SNR), tag byte 1

Figure 15: Heatmaps on Cortex-M4 (128-bit arch) for various tag bytes.

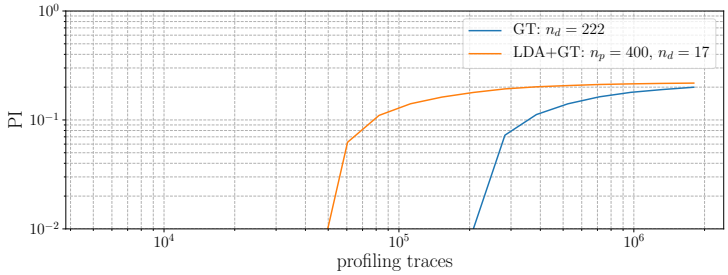
B Perceived Information Analysis

Next, we compare the Perceived Information (PI) for profiling methods using Gaussian templates in the original leakage space (GT) and in a linear subspace (LDA+GT).

The number of dimensions of the Gaussian templates are denoted as n_d. If applicable, the number of dimensions taken into account to project in the linear subspace is denoted as n_p. See Subsection 2.1 for additional details. Figure 16, that reports the PI according to the profiling set size, leads to two main observations: First, both types of templates converge to similar values, meaning that a similar amount of information can be extracted with GT and GT+LDA. Second, we observe that the LDA+GT approach converges faster, meaning that less profiling samples are needed in order to fit the model. The gap between LDA+GT and GT depends on the number of profiling samples.



(a) Cortex-M33 (32-bit arch), 256 averaging



(b) Cortex-M4 (128-bit arch), 2048 averaging

Figure 16: PI for the best models we found with and without using linear subspace.

C Compression functions and multi-collision

Ths Merkle-Damgard (MD) paradigm for hash functions is based on the iteration of a compression function. Here we give a multi-collision probability upper-bound for general compression functions that relies on a single call to a block cipher. Our result can be viewed as a generalization of the collision bound for PGV-like compression [BRSS10], for α-multi-collisions from $\alpha = 2$ to $\alpha \geq 2$.

Definition 3 (Good compression)**.** Let $E : \{0,1\}^n \times \{0,1\}^n \mapsto \{0,1\}^n$ be a block cipher. A *good compression* $f[E]$ is a compression function $f : \{0,1\}^n \times \{0,1\}^n \mapsto \{0,1\}^n$ based on E equipped with efficient functions g_{in}, g_{out}, g_{mid} such that $f(h,m) = z$, where (i) $(h,m) \mapsto (k,x) = g_{in}(h,m)$; (ii) $y \leftarrow E_k(x)$; (iii) $(h,m,y) \mapsto z = g_{out}(h,m,y)$; as well as (iv) $g_{mid}(k,x,y) \mapsto g_{out}(g_{in}^{-1}(k,x),y) = z$ with the following properties:

1. $g_{in} : \{0,1\}^n \times \{0,1\}^n \mapsto \{0,1\}^n \times \{0,1\}^n$ is invertible;
2. $g_{out}(h,m,\cdot) : \{0,1\}^n \mapsto \{0,1\}^n$ is invertible;
3. $g_{mid}(k,\cdot,y) : \{0,1\}^n \mapsto \{0,1\}^n$ is invertible.

Good compression functions are the compression functions satisfying the so-called *T1-property* in [BRSS10] of generalized PGV-like compression functions [PGV93]. The 12 secure PGV compression function of the so-called *group-1* satisfy T1.

For instance, the compression function $f_{\alpha,\beta,\gamma,\delta}(h,m) = \gamma E_{\alpha h}(\beta m) + \delta m$, for some non-zero constants over $\mathbb{F}_{2^n}$, satisfies all the above requirements. Hence, so are $E_h(m) \oplus 2m$ and $2E_h(m) \oplus m$ involved in Mennink's double-length compression function. Indeed, given $\alpha, \beta, \gamma, \delta \in \mathbb{F}_{2^n}^*$, it is easy to see that $g_{in}(h,m) = (\alpha h, \beta m)$, $g_{out}(h,m,y) = \gamma y + \delta m$ and $g'(k,x,y) = \gamma y + \delta/\beta x$ fulfills all the requirements.

In this paper, we use many compression functions. We will rely on Mennink's double length compression function which makes 3 calls to the underlying block cipher. This compression function has a double-length state and it is easy to see that it implicitly contains 3 different types of (single-length) good compression functions.

This is why a multi-collision bound on good compression functions can be usefull. While it is easy to compute exponentially many collisions on $\mathsf{MD}(f)$ from several 2-collisions when f is good compression function (due to a result of Joux [Jou04]), multi-collisions on f are still interesting to bound. We believe this to be of independent interest and especially for other modes based on Hirose's compression function or Tandem.

Although, the probability of multi-collisions have been studied in the case of random hash function [STKT08], we are not aware of a general lower-bound for multi-collisions on PGV-like compression functions. The following theorem fills this gap. We note that the definition of collision resistance of [PGV93, BRSS10] includes a target point as in the definition of preimage resistance. Here, there is no target point, which explains the difference between $(q+1)q$ in the original bounds and $q(q-1)$.

Theorem 2 (Multi-Collision)**.** *Let E be a n-bit block cipher modeled as an ideal cipher and let $f[E]$ be a good compression function. Then, by making at most q (forward and backward) queries to E, the probability of finding an s-multi-collisions is bounded by:*

$$\frac{2^n}{(2^n-q)^s}\binom{q}{s}.$$

Since $f[E]$ is a good compression, any forward query (k,x) defines a single f-query $g_{in}^{-1}(k,x)$. Likewise, any backward query (k,y) defines a single f-query $g_{in}^{-1}(k, E_k^{-1}(y))$. Therefore, we do not distinguish E-queries from f-queries.

Proof. Let $C_s(q)$ be the event that there is at least one s-multi-collision for the good compression $f[E]$ after q queries. We partition this event twice as follows by considering the events $C_s(q,z)$ and $C_s(q,z,i)$, for any $z \in \{0,1\}^n$ and any $s \leq i \leq q$. $C_s(q,z)$ is the

event that the first time an s-multi-collision occurs it is on the output z. $C_s(q,z,i)$ is the same as $C_s(q,z)$ except that the first time the s-multi-collision occurs is at the i-th query. We thus have:

$$\Pr[C_s(q)] = \sum_{z\in\{0,1\}^n} \sum_{i=s}^{q} \Pr[C_s(q,z,i)] \ .$$

Now, by considering all the possible colliding queries $1 \le i_1 < \ldots < i_{s-1} \le i-1$,

$$C_s(q,z,i) \subset \bigcup_{1\le i_1<\ldots<i_{s-1}\le i-1} C^*(z,i_1,\ldots,i_{s-1},i),$$

where $C^*(z,i_1,\ldots,i_s)$ is the event that all the i_j-th queries (h_{i_j},m_{i_j}) are exactly those which hit z, for $1\le j\le s$. That is, $f(h_{i_j},m_{i_j})=z$, while each of the other i'-th query for $i'\le i_s$ and $i'\neq i_j$, for $1\le j\le s$, does not collide on z (no matter whether other s-multi-collisions occur or not before the i_s-th query). We find:

$$\Pr[C^*(z,i_1,\ldots,i_{s-1},i)] = \prod_{j=1}^{s} \frac{1}{2^n - K_{i_j}(h_{i_j},m_{i_j})} \ ,$$

where $K_i(h_i,m_i)$ is the number of time the key k_i from $g_{in}(h_i,m_i)=(k_i,x_i)$ has already been used in some computation $E_{k_i}(\cdot)$ in a previous query *before* the i-th query. Assuming that $f(h_i,m_i)=z$, we argue why the factor $1/(2^n-K_i(h_i,m_i))$ is indeed the right piecewise probability: (i) in the case of a forward query, there is a single possible output y of E given z and the i-th query (h_i,m_i) (or equivalently (k_i,x_i)) since $g_{out}(h_i,m_i,\cdot)=z$ has exactly one solution by assumption on g_{out}; (ii) in the case of a backward query, there is a single possible input x of E given z and (k_i,y_i) since $g_{mid}(k_i,\cdot,y_i)=z$ has exactly one solution by assumption on g_{mid}. In both cases, the number of remaining possible outputs of forward or backward evaluations of the random permutation E_{k_i} is exactly 2^n minus all the previous calls to it, which is $K_i(h_i,m_i)$ by definition.

Therefore, since $\Pr[C^*(z,i_1,\ldots,i_{s-1},i)] \le 1/(2^n-i)^s \le 1/(2^n-q)^s$, we find:

$$\Pr[C_s(q)] \le \frac{1}{(2^n-q)^s} \sum_{z\in\{0,1\}^n} \sum_{i=s}^{q} \sum_{1\le i_1<\ldots<i_{s-1}\le i-1} 1 \ .$$

Since we have $\sum_{1\le i_1<\ldots<i_{s-1}\le i-1} 1 = \binom{i-1}{s-1}$ and $\sum_{i=s}^{q}\binom{s-1}{i-1} = 1+\sum_{i=s+1}^{q}\binom{s-1}{i-1} = 1+\sum_{i=s+1}^{q}\left[\binom{i}{s}-\binom{i-1}{s}\right] = 1+\binom{q}{s}-\binom{s}{s} = \binom{q}{s}$ because $\binom{i}{s}=\binom{i-1}{s-1}+\binom{i-1}{s}$, for $i\ge s+1$, we finally get:

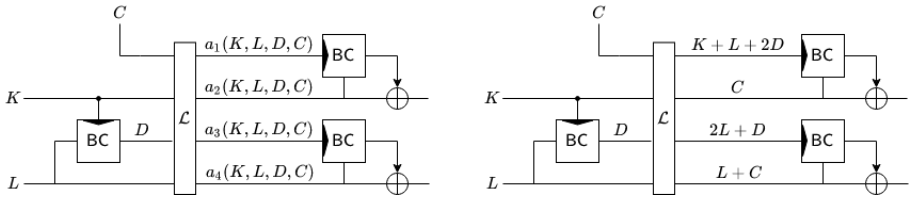
$$\Pr[C_s(q)] \le \frac{2^n}{(2^n-q)^s}\binom{q}{s} \ .$$

□

Let $q\le 2^{n-2}$. For $s=2$, we match the bound $\Pr[C_2(q)]\le q(q-1)/2^n$ of [BRSS10]. In that case, $\Pr[C_3(q)]\le q^3/2^{2n}$. More generally, for $s\ge 4$, and $q\le 2^{n-1}$, we have $\Pr[C_s(q)]\le q^s/2^{n(s-1)}$. To summarize, we have that good compression functions resist to s-multi-collision at least up to $2^{\frac{s-1}{s}n}$ queries, for $s\le n$.

D Mennink's Double-Length Compression Function

Mennink's double-length compression function provides optimal security based on 3 calls to the underlying BC modeled as an ideal cipher.



(a) General design **(b)** Our choice of the linear map

Figure 17: Mennink's optimally secure generic compression function.

Collision resistance. The linear map $\mathcal{L}$ over $\mathbb{F}_{2^n}$ of the compression function must be invertible. If the i-th row corresponds to the linear function a_i in the figure, we find the following matrices:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & 0 \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & 0 \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \qquad \begin{pmatrix} 1 & 1 & 2 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 2 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

To be collision resistant, we also must have: $a_{12}, a_{13}, a_{24}, a_{32}, a_{33}, a_{44} \neq 0$, and $a_{12} \neq a_{32}$, $a_{13} \neq a_{33}$. In that case, the collision advantage for $n = 128$ allows $q \approx 2^{118}$ ideal cipher queries to have a collision probability of $1/2$. The main Mennink's bound depends on two parameters t_1 and t_2 that control the maximum number of some bad events in the security proof. To give a concrete non-optimal simple bound, if we set $t_1 = q/t_2$, for $t_2 \geq 9$, we have:

$$\varepsilon_{\mathsf{MenH}} \leq \frac{3t_2^2 q}{(2^n - q)} + \frac{2^n}{3^{t_2-1}} \left(\frac{q}{2^n - q}\right)^{t_2},$$

where $\varepsilon_{\mathsf{MenH}}$ is the collision-resistance advantage. A closer look at Mennink's proof shows $\varepsilon_{\mathsf{MenH}} \leq \tilde{\varepsilon}_{\mathsf{MenH}} + \varepsilon_{\mathsf{mu\text{-}coll}}$, where $\varepsilon_{\mathsf{mu\text{-}coll}}$ is thrice the advantage of finding a $t_2 + 1$ multi-collision for any of the following 3 compression functions $a_1(K, L, D, C)$, $a_3(K, L, D, C)$ and $f_1(h, m) = E_h(m) + m$ involved in the last 2 BC calls of Figure 17. Since in our mode we separately deal with multi-collisions (which implies those of Mennink's proof), we are only interested in:

$$\tilde{\varepsilon}_{\mathsf{MenH}} \leq \frac{(2t_2^2 + 7t_2 + 18)q}{2^n - q},$$

for any $t_2 \geq 1$, as long as our multi-collision parameter equals t_2.

Preimage resistance. With more constraints on the linear map, the everywhere preimage resistance advantage of the compression function is roughly $\leq q^{2/3}/2^n$, but asymptotically in n [Men17]. For $n = 128$, we can actually have 180 bits of security (instead of 191 bits, for $q \approx 2^{3n/2}$). Yet, this is a strong (and tight) result.

However, we do not need such a bound in the security of our mode. The only place where we need to rely on the preimage resistance is when we argue about state collision resistance. Although most of the state collisions can be avoided by directly relying on the collision resistance of the compression function itself, the initial phase of our mode derives many IV's that serves to initialize MenH in the MD construction. This forces us to show that none of these IV's can appear in some internal (non initial) state as we

want to maintain a fresh final state to have independent tags. A natural strategy would be to generate random IV's from the nonces in the initialization and to directly apply Mennink's result. However, this does not lead to a strong security in our case: if we have q_{on} online queries (adding up encryption and decryption), we will have an asymptotic bound of $q_{on}q^{2/3}/2^n$ allowing "only" $q_{in} \approx 2^{n/2}$ and $q \approx 2^{3n/4}$ or $q_{on} \approx q \approx 2^{3n/5}$, for instance. And the bound would be even worse for $n = 128$.

Instead, our strategy is to generate the IV's as $(K_0, 1)$ for a random n-bit half state K_0 depending on the nonces to solve the above preimage issue. That way, we only need to rely on preimage reistance for a *single* half-state value. Collision between initial states and internal states thus imply computing a preimage of 1 for a PGV-like good compression. IV collisions are now more likely as it is enough to collide on the n-bit value K_0. Still, it is easy to avoid causing collisions in the next state by first absorbing the nonce in a compression function call, since the first evaluation input $(K_0, 1, N)$ is always fresh. This costs us one more compression function call but it improves the bound drastically.

Since we do not have to comply with the more restrictive choice for the linear map to be preimage resistant, we have more freedom to choose the coefficients. In our choice we avoid using K in plaintext-input to give more confidence in confidentiality since it lets open the possibility to make a proof of our mode based on a related-key secure block cipher in the standard model for a future work.

Indifferentiability. Mennink's compression function is also indistinguishable from a random function with domain $\{0,1\}^{3n}$ and range $\{0,1\}^{2n}$ if the linear map satisfy the requirement for collision resistance. The indifferentiability adantage is of birthday type in n.

This result is very convenient to argue about the confidentiality with leakage of our mode: the internal leakage does not help to distinguish our key-stream from a truly random stream up to the birthday bound. As long as the internal computation remains hidden enough, the security will follow easily by other standard techniques.