Strong Authenticity with Leakage under Weak and Falsifiable Physical Assumptions

Francesco Berti[1], Chun Guo[2], Olivier Pereira[1],
Thomas Peters[1], and François-Xavier Standaert[1]

[1] ICTEAM/ELEN/Crypto Group, UCL, Louvain-la-Neuve, Belgium
[2] School of Cyber Science and Technology and Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University

Abstract. Authenticity can be compromised by information leaked via side-channels (e.g., power consumption). Examples of attacks include direct key recoveries and attacks against the tag verification which may lead to forgeries. At FSE 2018, Berti et al. described two authenticated encryption schemes which provide authenticity assuming a *leak-free implementation* of a Tweakable Block Cipher (TBC). Precisely, security is guaranteed even if all the intermediate computations of the target implementation are leaked in full but the TBC long-term key. Yet, while a leak-free implementation reasonably models strongly protected implementations of a TBC, it remains an idealized physical assumption that may be too demanding in many cases, in particular if hardware engineers mitigate the leakage to a good extent but (due to performance constraints) do not reach leak-freeness. In this paper, we get rid of this important limitation by introducing the notion of *Strong Unpredictability with Leakage* for BC's and TBC's. It captures the hardness for an adversary to provide a fresh and valid input/output pair for a (T)BC, even having oracle access to the (T)BC, its inverse and their leakages. This definition is game-based and may be verified/falsified by laboratories. Based on it, we then provide two Message Authentication Codes (MAC) which are secure if the (T)BC on which they rely are implemented in a way that maintains a sufficient unpredictability. Thus, we improve the theoretical foundations of leakage-resilient MAC and extend them towards engineering constraints that are easier to achieve in practice.

1 Introduction

Message Authentication Codes (MAC) are widely used to authenticate data. Efficient MAC are usually constructed from conceptually simpler symmetric primitives such as (tweakable) block ciphers (e.g., CBC [5]) and hash functions (e.g., HMAC [5,4]), and enjoy reliable "provable security guarantees", i.e., security reductions to the underlying primitives.

Side-channel attacks, since introduced in the 1990s [21,22], have now been recognized as one of the main real-world security threats (e.g., see [2, chapter 1.2]). In response, various implementation-level countermeasures have been proposed and even formally proved effective. However, they typically induce significant overheads. As a complementary, the methodology of *leakage-resilience* was proposed [14] and followed by many (see [20] for a survey). Schemes proved leakage-resilient enjoy security even if a moderate amount of sensitive information is leaked via side-channels. Consequently, their implementations could leverage less protected circuits and thus reduce the overall overheads.

It is not a surprise that with leakages, classical MAC such as CBC and HMAC are not secure at all, even if leakages only contain the input/output values of the underlying functions (see, e.g., [12]). This means their implementations have to be heavily protected when used in sensitive settings such as the IoT, which may be hard to achieve given application cost constraints. Therefore, exploring the construction of leakage-resilient MAC is a natural direction,

which was initiated in [18,24,25,8] and later improved in [9,3,7] to achieve security in the presence of *both tag generation and verification leakages*.[1] We remark that the premises used in these works are significantly different. For example, [24,3] leveraged bilinear maps in the generic group model to ease secret-sharing/masking-based implementations of their MAC, while [25,8,9,7] model a heavily protected (tweakable) block cipher (e.g., using high-order masking [15,19]) as leak-free and focus on making the other mode-level leakages harmless.

From the efficiency viewpoint, sticking with simple symmetric primitives is naturally desirable.[2] Yet, a drawback of the aforementioned papers [25,8,9,7] is the use of leak-free cipher model. Despite it is theoretically possible to reach very high security levels with masking (approaching black box security [15,19]), it implies (very) high overheads that may not be acceptable in practice. Besides, the leak-free assumption (that is, nothing is leaked about the key used and the outputs remain pseudorandom) cannot be accompanied by any well-defined security game—somewhat resembling the random oracle model.

Our contribution. The goal of this paper is to bridge the above theory gap (i.e., seeking for some well-defined leakage assumptions on the block cipher that allows the leakage-resilient MAC security reductions) while also enabling more modular security guarantees that may degrade gracefully when the physical assumption is respected only to some extent. Our answer to this challenge is *Strong Unpredictability in the presence of Leakages* for a (T)BC, henceforth abbreviated as SUL2. In detail, it captures the hardness of providing a fresh input/output pair for the (T)BC even having access to its leaking oracle and leaking inverse oracle (following the notations of [17], the variant without leaking inverse oracle would be SUL1). It can be viewed as a natural extension of the unpredictable block cipher assumption introduced by Dodis and Steinberger [12,13].

With this new assumption, we revisit existing (tweakable) block cipher-based leakage-resilient MAC. We first consider the simplest Hash-then-BC scheme $\tau = \mathsf{F}_k(\mathsf{H}(m))$, the leakage security of which was analyzed by Berti et al. [8,9]. While the security reduction seems straightforward, Berti et al. [9] changed the verification process of $\mathsf{Vrfy}_k(m,\tau)$ from "If $\tau = \mathsf{F}^*_k(\mathsf{H}(m))$, then return 1" to "If $\mathsf{H}(m) = \mathsf{F}^{*,-1}_k(\tau)$, then return 1" (i.e., leveraging the inverse $\mathsf{F}^{*,-1}_k$ to avoid leaking sensitive information).[3] As a result, they achieve better mode-level leakage-resilience as the intermediate value $\mathsf{F}^{*,-1}_k(\tau)$ does not have to be protected: even given this additional value it is still hard to forge a tag. However, their proof relies on leak-free block ciphers. We show that the SUL2 assumption for F^* is actually sufficient to obtain similar guarantees, further assuming an ideal hash H.[4] We then revisit the recently proposed Hash-then-TBC scheme [7], which was also used in the NIST AE submission Spook [6]. In detail, its tag generation is $\mathsf{Tag}_k(m) = \tau = \mathsf{F}^*_k(h_1,h_2)$, where h_2 is the tweak of F^* and $h_1 \| h_2 = \mathsf{H}(m)$ (i.e., the $2n$-bit output of H is divided into two halves h_1 and h_2), while for verification $\mathsf{Vrfy}_k(m,\tau)$ we compute $\tilde{h}_1 = \mathsf{F}^{*,-1}_k(\tau,h_2)$ and compare it with h_1, with

[1] Note that some MACs were parts of authenticated encryption (AE) proposals.
[2] The MAC of [24,3] consumes $\approx$ 4 seconds to generate a tag on a 32-bit ARM.
[3] F^* means that the BC F is implemented in a leak-free way
[4] This idealized assumption is used for simplifying our analyzes, since our focus is on the leak-free blocks. We leave its relaxation as an interesting open problem.

$h_1 \| h_2 = \mathsf{H}(m)$. We show that using a $2n$-bit hash, we can improve the birthday bound of the previous Hash-then-BC construction.

Both schemes are natural, extremely simple and should be easy to implement for practical uses. We expect the block cipher based construction (next: HBC) to be slightly more efficient than the tweakable version (next: HTBC) as a secure TBC typically consumes more rounds than a secure BC. However, we also note that HBC admits forgery attacks with lower data complexity (simply utilizing a hash collision), while HTBC solves this problem by doubling the size of the hash and using a TBC (that has a larger input size) to absorb the digest.

In summary, our results improve the theory foundations for existing efficient (tweakable) block cipher-based leakage-resilient MAC. We believe the SUL2 assumption could find more applications in future leakage-resilient analyses. In practice, unpredictability is widely believed to be more relaxed than PRP [12].[5] Thus it potentially enables using *reduced-round* (tweakable) BCs for MAC. As the heavily protected ciphers are much more costly than the hash functions (that do not need protection), they are expected to be the performance bottleneck, and reducing the number of rounds may significantly improve the overall performance (e.g., in terms of latency and energy consumption).

Related work. The idea of basing MAC security on unpredictable ciphers is not new, dating back to [1] and witnessing recent achievements [12,28,13]. In fact, as argued in [12], it is natural to consider reducing the unpredictability of the "bigger" MAC to the unpredictability of the "smaller" ciphers.

2 Background

Notations. A $(q_1, ..., q_d, t)$-adversary A in an experiment against Π is an algorithm A having oracle access to $\mathsf{Algo}_1, ..., \mathsf{Algo}_d$, making at most q_i queries to oracle Algo_i, running in time bounded by t, and outputting a finite string of bits. The leakage function due to the implementation of an algorithm Algo is denoted $\mathsf{L}_{\mathsf{Algo}}$. This function might be non deterministic. A leaking query to Algo is denoted LAlgo and results in running both Algo and $\mathsf{L}_{\mathsf{Algo}}$ on the same input.

The set of binary strings of length n is denoted by $\{0,1\}^n$, while the set of all finite strings by $\{0,1\}^*$. Given two strings, x and y, we let $x||y$ denote the concatenation of these two strings.

The *view* of a game consists of all queries made by the adversary to his oracles, the oracles' answers and the final output of the adversary. In a transcript every oracle query is immediately followed by its answer. A value is *fresh* if it has never appeared in the view.

2.1 Multi-Collisions

Let $1 \leq s \leq q \leq N$. We consider the experiment where we uniformly throw q balls at random into N bins. $\mathsf{MultiColl}(N, q) \geq s$ denotes the event that at least one bin contains at least s balls. We recall a useful upper-bound on the probability of multi-collisions.

[5] Indeed, Unpr can be based on weaker complexity assumptions [11].

Theorem 1 ([26]).

$$\Pr[\mathsf{MultiColl}(N,q) \geq s] \leq \frac{1}{N^{s-1}}\binom{q}{s}.$$

We also need the following technical result.

Lemma 1. *If* $2q \leq N$,

$$\sum_{s=1}^{q}(s-1)\cdot\Pr[\mathsf{MultiColl}(N,q) \geq s] \leq \frac{1}{N}\binom{q}{2}\left(1+\frac{2q}{N}\right).$$

Proof. Looking at the generic term for $s \geq 3$ after applying the theorem leads to

$$\frac{s-1}{N^{s-1}}\binom{q}{s} \leq \frac{1}{N^{s-2}}\binom{q}{s-1}\cdot\frac{q}{N} \leq \frac{1}{N}\binom{q}{2}\cdot\left(\frac{q}{N}\right)^{s-2}.$$

Then, the whole sum is upper-bounded by

$$\frac{1}{N}\binom{q}{2}\cdot\sum_{s=2}^{q}\left(\frac{q}{N}\right)^{s-2} \leq \frac{1}{N}\binom{q}{2}\frac{N}{N-q} = \frac{1}{N}\binom{q}{2}\left(1+\frac{q}{N-q}\right).$$

Hence, the result since $q \leq N-q$. □

2.2 Cryptographic Primitives

Tweakable block ciphers A *tweakable block cipher* [23] (TBC) is a function $\mathsf{F}:\mathcal{K}\times\mathcal{TW}\times\mathcal{X}\longmapsto\mathcal{Z}$, where $\mathcal{K}$ and $\mathcal{TW}$ are respectively called the key space and the tweak space, and such that for any key $k\in\mathcal{K}$ and any tweak $tw\in\mathcal{TW}$, the function $\mathsf{F}_k^{tw}:\mathcal{X}\mapsto\mathcal{Z}; x\mapsto \mathsf{F}_k^{tw}(x) := \mathsf{F}_k(tw,x) := \mathsf{F}(k,tw,x)$ is a permutation. We denote the inverse of this function by $\mathsf{F}_{k,tw}^{-1}$ so that, if $\mathsf{F}_k^{tw}(x)=y$, then $\mathsf{F}_k^{-1}(tw,y) := \mathsf{F}_{k,tw}^{-1}(y) = x$. A Block Cipher (BC) is a TBC with an empty tweak space: the only tweak is the "empty string".

Message authentication codes with leakage A *message authentication code* (MAC) is a couple of algorithms $(\mathsf{Tag},\mathsf{Vrfy})$, $\mathsf{Tag}:\mathcal{K}\times\mathcal{M}\longmapsto\mathcal{T}$ and $\mathsf{Vrfy}:\mathcal{K}\times\mathcal{M}\times\mathcal{T}\longmapsto\{0,1\}$, where $\mathcal{K}$, $\mathcal{M}$ and $\mathcal{T}$ are respectively called the key space, the message space and the tag space, and such that for any key $k\in\mathcal{K}$ and any message $m\in\mathcal{M}$, $1\leftarrow\mathsf{Vrfy}_k(m,\mathsf{Tag}_k(m))$.

Definition 1. *A* $\mathsf{MAC}=(\mathsf{Tag},\mathsf{Vrfy})$ *is* (q_T,q_V,t,ϵ) strongly existentially unforgeable against chosen-message and verification attacks*, or simply* (q_T,q_V,t,ϵ)-$\mathsf{suf\text{-}vcma}$ *for short, if for all* (q_T,q_V,t)*-adversary* A*, we have*

$$\Pr[\mathsf{FORGE}_{\mathsf{A},\mathsf{MAC}}^{\mathsf{suf\text{-}vcma}}=1]\leq\epsilon$$

where the $\mathsf{FORGE}^{\mathsf{suf\text{-}vcma}}$ *experiment is defined in Tab. 1.*

$\mathsf{FORGE}^{\mathsf{suf\text{-}vcma}}_{\mathsf{MAC},\mathsf{A}}$ experiment.	
Initialization: $k \xleftarrow{\$} \mathcal{K}$ $\mathcal{S} \leftarrow \emptyset$ Finalization: $(m,\tau) \leftarrow \mathsf{A}^{\mathsf{Tag}(\cdot),\mathsf{Vrfy}(\cdot,\cdot)}$ If $(m,\tau) \in \mathcal{S}$ Return 0 Return $\mathsf{Vrfy}_k(m,\tau)$	Oracle $\mathsf{Tag}(m)$: $\tau \leftarrow \mathsf{Tag}_k(m)$ $\mathcal{S} \leftarrow \mathcal{S} \cup \{(m,\tau)\}$ Return τ Oracle $\mathsf{Vrfy}(m,\tau)$: Return $\mathsf{Vrfy}_k(m,\tau)$

Table 1. The $\mathsf{FORGE}^{\mathsf{suf\text{-}vcma}}$ experiment.

To model the ability of an adversary to get leakage on tag generation and verification, we extend the $\mathsf{FORGE}^{\mathsf{suf\text{-}vcma}}$ experiment by allowing the oracles to additionally return the evaluation of $\mathsf{L}_{\mathsf{Tag}}$ and $\mathsf{L}_{\mathsf{Vrfy}}$, where $\mathsf{L} = (\mathsf{L}_{\mathsf{Tag}}, \mathsf{L}_{\mathsf{Vrfy}})$ is the leakage function pair due to an implementation of the MAC. Given an adversary A, we write A^{L} to specify that the adversary knows the implementation and that it can learn the leakage for chosen keys, which models any leakage learning phase on other devices with the same implementation.

Definition 2. *A* $\mathsf{MAC} = (\mathsf{Tag}, \mathsf{Vrfy})$*, whose implementation has leakage function pair* $\mathsf{L} = (\mathsf{L}_{\mathsf{Tag}}, \mathsf{L}_{\mathsf{Vrfy}})$ *is* $(q_T, q_V, q_{\mathsf{L}}, t, \epsilon)$ strongly existentially unforgeable against chosen message and verification attacks with leakage in tag-generation and verification*, or* $(q_T, q_V, q_{\mathsf{L}}, t, \epsilon)$-$\mathsf{suf\text{-}L2}$*, if for any* $(q_T, q_V, q_{\mathsf{L}}, t)$*-adversary* A*, we have*

$$\Pr[\mathsf{FORGE\text{-}L2}^{\mathsf{suf\text{-}vcma}}_{\mathsf{A},\mathsf{MAC},\mathsf{L}} = 1] \leq \epsilon,$$

where the $\mathsf{FORGE\text{-}L2}^{\mathsf{suf\text{-}vcma}}$ *experiment is defined in Tab. 2, and where* A^{L} *makes at most* q_{L} *queries to* L*.*

$\mathsf{FORGE\text{-}L2}^{\mathsf{suf\text{-}vcma}}_{\mathsf{MAC},\mathsf{A}}$ experiment.	
Initialization: $k \xleftarrow{\$} \mathcal{K}$ $\mathcal{S} \leftarrow \emptyset$ Finalization: $(m,\tau) \leftarrow \mathsf{A}^{\mathsf{LTag}(\cdot),\mathsf{LVrfy}(\cdot,\cdot),\mathsf{L}}$ If $(m,\tau) \in \mathcal{S}$, return 0 Return $\mathsf{Vrfy}_k(m,\tau)$	Oracle $\mathsf{LTag}(m)$: $\tau \leftarrow \mathsf{Tag}_k(m)$, $\ell_{\mathsf{m}} \leftarrow \mathsf{L}_{\mathsf{Tag}}(k,m)$ $\mathcal{S} \leftarrow \mathcal{S} \cup \{(m,\tau)\}$ Return $(\tau, \ell_{\mathsf{m}})$ Oracle $\mathsf{LVrfy}(m,\tau)$: $\ell_{\mathsf{v}} \leftarrow \mathsf{L}_{\mathsf{Vrfy}}(k,m,\tau)$ Return $(\mathsf{Vrfy}_k(m,\tau), \ell_{\mathsf{v}})$

Table 2. The $\mathsf{FORGE\text{-}L2}^{\mathsf{suf\text{-}vcma}}$ experiment.

Note that the $\mathsf{L2}$ notation is for leakage during both tag generation and verification (the variant without tag verification leakage would use $\mathsf{L1}$, following [17]).

Unbounded leakage In the rest of this paper, we will consider that all the components of our system, except for the BC/TBC, have an unbounded leakage. That is, all the I/Os of these components (the compression function of hash functions, for instance) are offered by the leakage function. Our paper then focuses on the single component that is not expected to

leak its I/Os in full and is used once per MAC computation or tag verification (independently of the length of the message to be authenticated): the BC/TBC.

3 Unpredictability of leaking TBC

Dodis and Steinberger [12] introduced the definition of *unpredictability with leakage* for BC. At a high level, the definition says it is unfeasible to produce a valid input-output couple of the BC even if we got the leakage besides of the outcome of the computation of the BC on chosen inputs. We extend this notion by granting the adversary with the inverse oracle of the BC and its leakage. To save some place, we directly describe this notion for TBCs. We get the corresponding notion for BCs by removing all the tweaks in the definition below.

We denote by $\mathsf{L} = (\mathsf{L}_{\mathsf{Eval}}, \mathsf{L}_{\mathsf{Inv}})$ the leakage function pair associated to an implementation of the TBC, where $\mathsf{L}_{\mathsf{Eval}}(k, tw, x)$ (*resp.* $\mathsf{L}_{\mathsf{Inv}}(k, tw, z)$) is the leakage resulting from the computation of $\mathsf{F}_k(tw, x)$ (*resp.* $\mathsf{F}_k^{-1}(tw, z)$). We also allow the adversary A to profile the leakages and write A^{L} as before, like in [25].

Definition 3. *A tweakable block cipher* $\mathsf{F} : \mathcal{K} \times \mathcal{TW} \times \mathcal{X} \longmapsto \mathcal{Z}$ *with leakage function pair* $\mathsf{L} = (\mathsf{L}_{\mathsf{Eval}}, \mathsf{L}_{\mathsf{Inv}})$ *is* $(q_E, q_I, q_{\mathsf{L}}, t, \epsilon)$ strongly unpredictable with leakage in evaluation and inversion*, or* $(q_E, q_I, q_{\mathsf{L}}, t, \epsilon)$-SUL2*, if for any* $(q_E, q_V, q_{\mathsf{L}}, t)$*-adversary* A*, we have*

$$\Pr[\mathsf{SUL2}_{\mathsf{A},\mathsf{F},\mathsf{L}} \Rightarrow 1] \leq \epsilon,$$

where the SUL2 *experiment is defined in Tab. 3, and where* A^{L} *makes at most* q_{L} *(offline) queries to* L*.*

$\mathsf{SUL2}_{\mathsf{A},\mathsf{F},\mathsf{L}}$ experiment.	
Initialization:	Oracle $\mathsf{LEval}(tw, x)$:
$k \xleftarrow{\$} \mathcal{K}$	$z = \mathsf{F}_k(tw, x)$
$\mathcal{L} \leftarrow \emptyset$	$\ell_{\mathsf{e}} = \mathsf{L}_{\mathsf{Eval}}(k, tw, x)$
	$\mathcal{L} \leftarrow \mathcal{L} \cup \{(x, tw, z)\}$
Finalization:	Return (z, ℓ_{e})
$(x, tw, z) \leftarrow \mathsf{A}^{\mathsf{LEval}(\cdot,\cdot),\mathsf{LInv}(\cdot,\cdot),\mathsf{L}}$	
If $(x, tw, z) \in \mathcal{L}$	Oracle $\mathsf{LInv}(tw, z)$:
Return 0	$x = \mathsf{F}_k^{-1}(tw, z)$
If $z == \mathsf{F}_k(tw, x)$	$\ell_{\mathsf{i}} = \mathsf{L}_{\mathsf{Inv}}(k, tw, z)$
Return 1	$\mathcal{L} \leftarrow \mathcal{L} \cup \{(x, tw, z)\}$
Return 0	Return (x, ℓ_{i})

Table 3. Strong unpredictability with leakage in evaluation and inversion experiment.

4 First leakage-resilient MAC: HBC

We now revisit one of the most common designs to build a MAC from a hash function H and a block cipher F. This MAC is the well-known hash-then-BC scheme, named here HBC, except that we analyze it in a leakage setting and through the lens of the unpredictability of

F. As we want to show the security of HBC even when F leaks its inputs and outputs in full, we just have to tweak the usual verification algorithm by using the inversion of the BC to avoid leaking valid tags just by processing invalid pairs (m, τ). As mentioned in introduction, our analysis models H as a random oracle for simplicity and since our focus is on the leak-free blocks. Yet, it does not suggest any reason why an ideal object would be needed, and its relaxation is an interesting open problem.

4.1 HBC description

Let $\mathcal{M} = \{0,1\}^*$ and $\mathcal{X} = \mathcal{Z} = \{0,1\}^n$. Considering a hash function $\mathsf{H} : \mathcal{M} \mapsto \mathcal{X}$ and a block cipher $\mathsf{F} : \mathcal{K} \times \mathcal{X} \mapsto \mathcal{Z}$, we build $\mathsf{HBC} = (\mathsf{Tag}, \mathsf{Vrfy})$:

$\mathsf{Tag}_k(m)$: compute $h = \mathsf{H}(m)$, then compute and output $\tau = \mathsf{F}_k(h)$.
$\mathsf{Vrfy}_k(m, \tau)$: compute $h = \mathsf{H}(m)$ and $\tilde{h} = \mathsf{F}_k^{-1}(\tau)$, then
output 1 if $h = h'$, and 0 otherwise.

We highlight that Tag only evaluates F while Vrfy only computes its inverse. This feature is at the core of the argument showing that unbounded leakages do not decrease the unforgeability of this hash-then-BC design. This idea was already used for the authentication part of the AE modes $\mathsf{DTE2}$, EDT and FEMALE [9,17]. We illustrate HBC in Fig. 1.

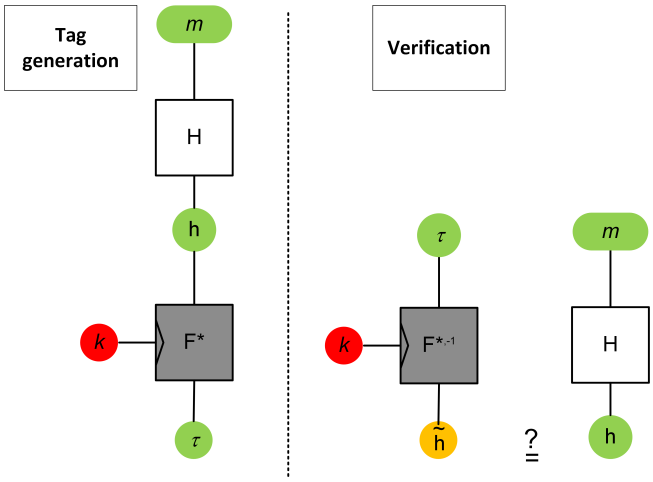

Fig. 1. The leakage resilient MAC HBC. Leakage reveals the orange value.

4.2 Security of HBC

In the unbounded leakage model, the adversary receives all the ephemeral values computed during the tag generation and the verification. Only the key of the BC, which is the key of the MAC, remains hidden as implicitly defined by the leakage function pair of its implementation $\mathsf{L} = (\mathsf{L}_{\mathsf{Eval}}, \mathsf{L}_{\mathsf{Inv}})$. More precisely, the unbounded leakage function pair $\mathsf{L}^* = (\mathsf{L}^*_{\mathsf{Tag}}, \mathsf{L}^*_{\mathsf{Vrfy}})$ of HBC is thus:

$\mathsf{L}^*_{\mathsf{Tag}}(k, m)$**:** return $h = \mathsf{H}(m)$ and $\mathsf{L}_{\mathsf{Eval}}(k, h)$;
$\mathsf{L}^*_{\mathsf{Vrfy}}(k, m, \tau)$**:** return $h = \mathsf{H}(m)$ and $\tilde{h} = \mathsf{F}_k^{-1}(\tau)$ as well as $\mathsf{L}_{\mathsf{Inv}}(k, \tau)$.

Despite H is a public function, we explicitly include its outputs in the leakage. It can be considered as redundant but, as we rely on a random oracle to prove the security of HBC, we prefer making them fully available to avoid any confusion.

Theorem 2. *Let* $\mathsf{F} : \mathcal{K} \times \{0,1\}^n \longmapsto \{0,1\}^n$ *be a* $(q_T, q_V, q_L, t, \epsilon_{\mathsf{SUL2}})$*-strongly unpredictable block cipher in the presence of leakage, and* $\mathsf{H} : \mathcal{S} \times \{0,1\}^* \longmapsto \{0,1\}^n$ *be a hash function modeled as a random oracle that is queried at most* q_H *times, then,* HBC *is a* $(q_T, q_V, q_{\mathsf{L}}, t, \epsilon)$*-strongly unforgeable MAC in the unbounded leakage setting, with* $\mathsf{L}^* = (\mathsf{L}^*_{\mathsf{Tag}}, \mathsf{L}^*_{\mathsf{Vrfy}})$ *defined above, where*

$$\epsilon \leq (q_H + q_V + 1)(q_V + 1)\epsilon_{\mathsf{SUL2}} + (q_H + q_T + q_V + 1)^2/2^n,$$

and $t_H(q_H + q_T + q_V + 1) + (q_T + q_L - q)t_{\mathsf{F}} + (q_V + q)t_{\mathsf{F}^{-1}} \leq t$ *for any* $q \leq q_L$*, and where we assume that all the* H*-query involved in the* q_L *queries are already among the* q_H *queries, and if* $q_V \leq q_H$ *(which can be artificially fulfilled at the end of the experiment).*

The advantage is bounded by $2q_Hq_V\epsilon_{\mathsf{SUL2}} + 4q_H^2 2^{-n}$ under the natural assumption $q_T + q_V + 1 \leq q_H$ (since q_T and q_V correspond to online queries while q_H corresponds to offline queries, it is expected to hold comfortably). The leading term is $2q_Hq_V\epsilon_{\mathsf{SUL2}}$: for $\epsilon_{\mathsf{SUL2}} = 2^{-128}$, it implies that security holds up to $q_H = 2^{64}$ and $q_V = 2^{63}$ (i.e., slightly below the birthday bound). For a more realistic $\epsilon_{\mathsf{SUL2}} = 2^{-96}$, it only holds with a stronger limit on the number of verification queries (e.g., $q_H \approx 2^{64}$ and $q_V = 2^{31}$). Note that the factor q_Hq_V may be due to the reduction proof technique, as it relates to a case where the adversary is likely to produce a forgery early in the experiment and therefore much of the computational power of the reduction seems useless to the adversary. Hence, it might be possible to obtain tighter bounds using a different reduction approach. It would also be interesting to explore the possibility of making a proof based on standard assumptions on the hash function. Such assumptions would require to exclude damaging and implausible interactions between the hash function and the PRF, and would be an interesting area for future research.

Idea of the proof. Assuming that an adversary A succeeds in the $\mathsf{FORGE\text{–}L2}^{\mathsf{suf\text{-}vcma}}_{\mathsf{A},\mathsf{HBC},\mathsf{L}^*}$ experiment by making a total of q_T leaking tag queries and q_V leaking verification queries, let (m, τ) be the forgery, i.e., the couple returned by A in the finalization phase. To bound this winning probability, we partition this event into sub events: (1) The tag τ appears in the answer to a leaking tag query (and thus, as an output of F_k); (2) The tag τ never appears in the answer to a leaking tag query (and thus, τ can only be involved as an input of F_k^{-1}) and: (a) m appears as an input of H before $\mathsf{F}_k^{-1}(\tau)$ was ever computed in the experiment; (b) τ appears as an input of F_k^{-1} before $\mathsf{H}(m)$ was ever computed in the experiment. We cover all the cases since when both m and τ are fresh in a verification query, we always compute (or ask the computation of) $\mathsf{H}(m)$ first so that we can say that m appears "before" (the computation of F_k^{-1} on input) τ, and since we view the last verification in the finalization as the $(q_V + 1)$-th verification query.

The goal of the proof is to show that the collision resistance of H ensures that winning in case 1 is negligible, the unpredictability of F ensures that winning in case 2a is negligible and that preimage resistance of H ensures that winning in case 2b is negligible as well.

In case 1, there is a tag query on m' which defines $\tau = \mathsf{F}_k(\mathsf{H}(m'))$. Since (m, τ) is a forgery we have $\mathsf{F}_k(\mathsf{H}(m)) = \tau$ with $m \neq m'$. Then, m and m' produce a collision as F_k is a permutation: $\mathsf{H}(m) = \mathsf{F}_k^{-1}(\tau) = \mathsf{H}(m')$.

In case 2a, we assume that F is $\mathsf{SUL2}$-secure. Since m appears before τ, as a challenger we have to use the value $h = \mathsf{H}(m)$ and we "wait" for the good tag in a verification query to win the $\mathsf{SUL2}$ game. We do not have to consider the message for which A makes a tag query. However, we cannot know in advance what will be the right tag and we cannot wait until the finalization of the unfogeability experiment because even if A's output (m, τ) is the right pair, τ may have been already used in a previous verification query. If so, the challenger should have already made a leaking inverse query of the block cipher with input τ to get $\tilde{h} = \mathsf{F}_k^{-1}(\tau)$ and $\ell_\mathsf{i} \leftarrow \mathsf{LInv}(k, m)$ to simulate $\mathsf{LVrfy}(m, \tau)$, and it could no more win the game against F with τ. Therefore, for all possible m involved in a H-query or a verification query, we have to guess what will be the right τ in verification. Then, we need to consider all the possible such pairs and we thus have to make at most $(q_\mathsf{H} + q_V + 1)(q_V + 1)$ reductions.

In case 2b, the reduction can generate the key k itself and then evaluate F and its inverse by itself. The first time τ appears (in a leaking verification query) we define the H-target $\tilde{h} = \mathsf{F}_k^{-1}(\tau)$ since the winning corresponding m is still not hashed by assumption. Note that we even do not care whether $\tilde{h}$ already appeared or not in a response to an H-query since fresh queries will result in independent hashes. Therefore, the validity of (m, τ) means that m is a preimage of $\tilde{h}$, as $\mathsf{H}(m) = \mathsf{F}_k^{-1}(\tau)$, while $\mathsf{H}(m)$ is random.

Proof. To prove the theorem, we use a sequence of games. Given an adversary A, we start with Game 0 which is the $\mathsf{FORGE\text{–}L2}^{\mathsf{suf\text{-}vcma}}_{\mathsf{A},\mathsf{HBC},\mathsf{L}^*}$ experiment and we end with a game where all the leaking verification queries deem the given input pair (m_i, τ_i) invalid, including the last verification at the finalization which is the $(q_V + 1)$-th verification query by convention.

Game 0. This game is depicted in Tab 2. Let E_0 be the event that the adversary $\mathsf{A}^{\mathsf{L}^*}$ wins this game, that is, the output of the experiment is 1.

Game 1. We introduce a failure event F_1 with respect to Game 0, where F_1 occurs if among the at most $(q_H + q_T + q_V + 1)$ hash computations there is at least one collision. In Game 1, if F_1 occurs we abort the game and return 0. We let E_1 be the event that the adversary A^{L} wins this game.

Bounding $|\Pr[E_0] - \Pr[E_1]|$. Since Game 0 and Game 1 are identical as long as F_1 does not occur, if $Q = q_H + q_T + q_V + 1$, we have

$$|\Pr[E_0] - \Pr[E_1]| \leq \Pr[F_1] \leq Q(Q+1)/2^{n+1}.$$

Note: from now on, A wins if τ never appears in a leaking tag query. Moreover, $\tilde{h} = \mathsf{F}_k^{-1}(\tau)$ is fresh when τ appears in a leaking verification query for the first time if m was never used as input of H at that time. (See above.)

Game 2. We modify the winning condition of the previous game. In the finalization, once A outputs (m, τ) we say that A does not win and return 0 if A fails as before or if m appears as an input of H before the first apparition of τ during a leaking verification query. If we call F_2 the event that makes the adversary winning in Game 1 but loosing in Game 2, we have $|\Pr[E_2] - \Pr[E_1]| \leq \Pr[F_2]$, where E_2 is the event that A wins in this game.

Bounding $\Pr[F_2]$. If we call V_i the event that (m, τ) appears for the first time in the i-th leaking verification query (m^i, τ^i), we just have to bound $\Pr[F_2 \cap V_i]$, for all $i = 1$ to q_V+1. By considering all the input-output pairs defined by H before the i-th leaking verification query, except those defined during a leaking tag query, we can build straightforwards reduction to the $\mathsf{SUL2}$-security of F. We thus have, $\Pr[F_2 \cap V_i] \leq (q_H + q_V + 1)\epsilon_{\mathsf{SUL2}}$ and finally

$$\Pr[F_2] = \sum_{i=1}^{q_V+1} \Pr[F_2 \cap V_i] \leq (q_H + q_V + 1)(q_V + 1)\epsilon_{\mathsf{SUL2}}.$$

Note: in Game 2, the adversary wins only if τ appears before m and τ first appears in a leaking verification. The random value of $\mathsf{H}(m)$ is still undefined at that time.

Game 3. In this game we follow the specification of $\mathsf{FORGE\text{–}L2}^{\mathsf{suf\text{-}vcma}}_{\mathsf{A},\mathsf{HBC},\mathsf{L}^*}$ except that we always output 0 at the end of the game.

Bounding $|\Pr[E_3] - \Pr[E_2]| = \Pr[E_2]$. From the last note, we know that $\tilde{h} = \mathsf{F}_k^{-1}(\tau)$ must be reached from a fresh computation of H. Since any fresh H evaluation results in a uniform output which is thus independent of the view of τ, $\Pr[H(m') = \tilde{h}] = 1/2^n$ for all hash evaluations on some m' appearing after τ in a H-query or in a LVrfy query. But the number of targets $\tilde{h}$ during the game is actually the number of different τ' in the LVrfy queries when considering the hash evaluations after each such τ'. Then $\Pr[E_2] \leq q_V(q_H + q_V)/2^n$.

To summarize, we have

$$\Pr[E_0] \leq (q_H + q_V + 1)(q_V + 1)\epsilon_{\mathsf{SUL2}} + \frac{q_V(q_H + q_V)}{2^n} + \frac{Q(Q+1)}{2 \cdot 2^n}$$

from which the result follows as $2q_V(q_H + q_V) \leq Q(Q-1)$. □

In the next section we show how to improve the security bound.

5 Second leakage-resilient MAC: HTBC

The design of our second construction is similar to that of HBC. The main difference in HTBC is that the hash function has a double output length. A TBC replaces the BC to use the tweak as a support for the additional part of the digest. The primary goal of this modification is to get a better bound, even in the unbounded leakage setting. However, we analyze this design under the perspective of the unpredictability of the TBC for the first time.

5.1 HTBC description

Let $\mathcal{M} = \{0,1\}^*$ and $\mathcal{TW} = \mathcal{X} = \mathcal{Z} = \{0,1\}^n$. Considering a hash function $\mathsf{H} : \mathcal{M} \mapsto \mathcal{X} \times \mathcal{TW} = \{0,1\}^{2n}$ and a tweakable block cipher $\mathsf{F} : \mathcal{K} \times \mathcal{TW} \times \mathcal{X} \mapsto \mathcal{Z}$, we build $\mathsf{HTBC} = (\mathsf{Tag}, \mathsf{Vrfy})$:

$\mathsf{Tag}_k(m)$: first compute $h_1 \| h_2 = \mathsf{H}(m)$ and $\tau = \mathsf{F}_k(h_2, h_1)$ and output τ.
$\mathsf{Vrfy}_k(m, \tau)$: first compute $h_1 \| h_2 = \mathsf{H}(m)$ and $\tilde{h}_1 = \mathsf{F}^{-1}_{k,h_2}(\tau)$, then
output 1 if $h_1 = \tilde{h}_1$, and 0 otherwise.

We stress again that Tag only evaluates F while Vrfy only computes its inverse. HTBC was proposed as the authenticator of TEDT [7] (also adopted in [16,6]), with the motivation to break the birthday security barrier in the Hash-then-Block-cipher HBC. As the hash digest has been increased to $2n$ bits, the standard hash collision-based attack turns unfeasible. We illustrate HTBC in Fig. 2.

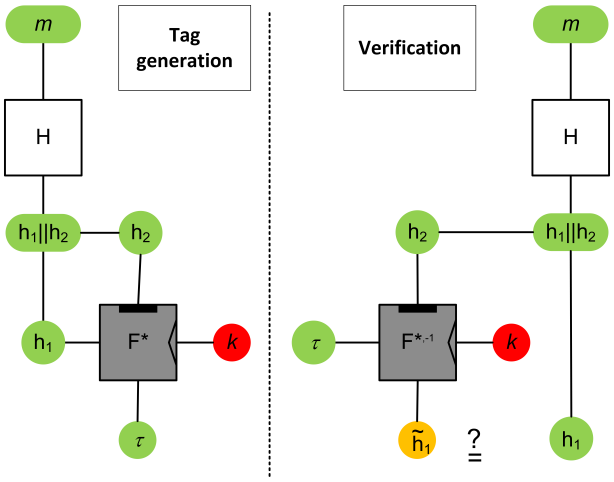

Fig. 2. The leakage resilient HBC-scheme.

5.2 Security of HTBC

The unbounded leakage function pair $\mathsf{L}^* = (\mathsf{L}^*_{\mathsf{Tag}}, \mathsf{L}^*_{\mathsf{Vrfy}})$ of HTBC is defined as

$\mathsf{L}^*_{\mathsf{Tag}}(k, m)$: return $h_1 \| h_2 = \mathsf{H}(m)$ and $\mathsf{L}_{\mathsf{Eval}}(k, h_2, h_1)$;
$\mathsf{L}^*_{\mathsf{Vrfy}}(k, m, \tau)$: return $h_1 \| h_2 = \mathsf{H}(m)$ and $\tilde{h}_1 = \mathsf{F}^{-1}_{k,h_2}(\tau)$ as well as $\mathsf{L}_{\mathsf{Inv}}(k, h_2, \tau)$.

As we rely on the random oracle model to prove the security of HTBC, we include the digests in the leakage to capture the fact that H is actually a public function.

Theorem 3. *Let $\mathsf{H} : \{0,1\}^* \longmapsto \{0,1\}^n \times \{0,1\}^n$ be a hash function modeled as a random oracle, and $\mathsf{F}^* : \mathcal{K} \times \{0,1\}^n \times \{0,1\}^n \longmapsto \{0,1\}^n$ be a $(q_T, q_V, q_\mathsf{L}, t + t_3, \epsilon_{\mathsf{SUL2}})$-strongly unpredictable tweakable block cipher with leakage $\mathsf{L} = (\mathsf{L}_{\mathsf{Eval}}, \mathsf{L}_{\mathsf{Inv}})$, then HTBC is a (q_T, q_V, t, ϵ)-$\mathsf{suf\text{-}L2}$ strongly unforgeable MAC with unbounded leakage function pair $\mathsf{L}^* = (\mathsf{L}^*_{\mathsf{Tag}}, \mathsf{L}^*_{\mathsf{Vrfy}})$ as*

defined above, where

$$\epsilon \leq \frac{(q_H + q_T + q_V)^2}{2^{2n}} + (q_V + 1) \cdot \epsilon_{\mathsf{SUL2}} + \frac{q_H^2 q_V}{2^n} \cdot \epsilon_{\mathsf{SUL2}} + \frac{q_V(q_H + q_V)}{2^{2n}},$$

and $t_H(q_H + q_T + q_V + 1) + (q_T + q_L - q)t_{\mathsf{F}} + (q_V + q)t_{\mathsf{F}^{-1}} \leq t$ *for any* $q \leq q_L$*, and where we assume that all the* H*-query involved in the* q_L *queries are already among the* q_H *queries, as long as* $4 \leq q_H + q_T + q_V$*,* $4q_V \leq q_H$ *and* $10q_H \leq 2^n$*.*

The leading term in the security bound is $\epsilon_{\mathsf{SUL2}} \cdot q_H^2 q_V 2^{-n}$. This time, for $n = 128$ and $\epsilon_{\mathsf{SUL2}} \approx 2^{-96}$, security holds up to $q_H \approx 2^{80}$ and $q_V = 2^{64}$. As for Theorem 2, we are not aware of a realistic matching attack (i.e., if a reasonable hash function and TBC are used in the construction). Investigating whether the additional $q_H 2^{-n}$ factor that we gain compared to the BC-based construction can get closer to 2^{-n} is an interesting open problem.

The structure of the proof for HTBC is different from that of HBC. The main reason is that the collision resistance of H does not cover all the winning cases when the adversary's τ of the forgery appears in an LTag query. Indeed, we might have $\mathsf{H}(m) = h_1 \| h_2 \neq h_1' \| h_2' = H(m')$ such that $\mathsf{F}_k(h_2, h_1) = \tau = \mathsf{F}_k(h_2', h_1')$ with m' in an LTag query. That is because F_{k,h_2} and $\mathsf{F}_{k,h_2'}$ can be seen as two different permutations given that $h_2 \neq h_2'$: an output τ defines many possible tweak-input pairs. As we will see the distribution and the freshness of (h_2, τ) will play an important role in the proof.

Idea of the proof. Let (m, τ) be a forgery and write $\mathsf{H}(m) = h_1 \| h_2$. If no triple of the form $(\star, h_2, \tau)$ appears during the computation of all the evaluations and inversions of F, (h_1, h_2, τ) is a valid fresh triple for F which breaks the unpredictability of the TBC. However, if it is not the case, the triple $(\star, h_2, \tau)$ appears either in the evaluation of F during an LTag query or only in the inversion of F in an LVrfy query. In the former case, as the answer to an LTag query is necessarily valid, the triple $(\star, h_2, \tau)$ must actually be $(\mathsf{F}_k^{-1}(h_2, \tau), h_2, \tau)$, i.e. (h_1, h_2, τ). Of course, if the adversary has made an LTag query on m, it cannot win. If the adversary is successful, it means that it managed to request the computation of a hash value which collides on $\mathsf{H}(m)$, which only occurs with a beyond-birthday probability. We can thus focus on the latter case where the triple $(\star, h_2, \tau)$ only appears when answering an LVrfy query, i.e. in an inversion of F.

We split the remaining winning conditions into: (1) m appears as an input of H before ever computing a TBC inversion on input (h_2, τ) when answering a leaking verification query; (2) m appears strictly after the first computation of a TBC inversion on input (h_2, τ) when answering a leaking verification query; no matter whether τ appears first in an LTag answer or in an LVrfy query. We note that we no more need to consider the H computation in the LTag queries as we already dealt with H-collisions. In a nutshell, the first case means the adversary chooses τ depending on the view of the hash value $h_1 \| h_2$ and hence it relates to the unpredictability of F. In the second case, the target $h_1 \| h_2$ is fixed in the leaking verification query while the output of $\mathsf{H}(m)$ remains uniformly random and independent of the view at that time. By convention, if m and τ first appear for the first time together in an LVrfy query,

we first compute $\mathsf{H}(m)$ so that we always consider that m appears "before" τ. In addition, we consider the forgery as the (q_V+1)-th LVrfy query.

In case 1, $\mathsf{H}(m) = h_1\|h_2$ appears before the computation of a TBC triple $(\star, h_2, \tau)$ which will first be run when answering a leaking verification query. We want to build an adversary B against F which ends by sending (h_1, h_2, τ). To make B successful, we have to prevent B from making an LInv query on input (h_2, τ) earlier, since otherwise (h_1, h_2, τ) is not a winning triple at the end. Such a query can happen only if A manages to make an LVrfy query on some (m', τ) such that $\mathsf{H}(m') = h'_1\|h'_2$ with $h'_2 = h_2$. Of course, this happens if $m = m'$ and, indeed, A can win if (m, τ) appears in an LVrfy query before the (q_V+1)-th one. However, it can also happen if $m' \neq m$, but then $h'_1 \neq h_1$. Fortunately, if the first time (h_2, τ) appears in an LVrfy query is with $m' \neq m$, we know that m appeared in a hash computation earlier for the first time (and it cannot be in an LTag query). To sum up, we cannot build a single B but by considering all the hash computations in the H queries and the LVrfy queries to combine with all the tags in the LVrfy queries, we have at most $(q_H + q_V + 1)(q_V + 1)$ reductions to build to cover all the possibilities. Fortunately, we only have to consider the messages m' with $\mathsf{H}(m') = h'_1\|h'_2$ such that h'_2 appears in a subsequent LVrfy query. Furthermore, we can see when this happens before having to invert the TBC with tweak h'_2. We will see in the proof that the probability of multi-collision on the h^i_2-value involved in the i-th LVrfy query will decrease the probability by a factor of roughly $q_H/2^n$, which gives us a beyond birthday term eventually.

In case 2, the adversary outputs a forgery (m, τ) while (h_2, τ) appears in a leaking verification query before the first computation of $\mathsf{H}(m)$. Here, we simply pick the key of the TBC to simulate the forgery experiment. If (h_2, τ) already appears in an LVrfy query the valid triple $(\tilde{h}_1, h_2, \tau)$ is already fixed in the answer to that query (necessarily invalid). Therefore $H(m)$ which is still uniformly random and independent of the view at that time will have to match the target $(\tilde{h}_1, h_2)$. This match thus happens with probability $1/2^{2n}$ for each future hash evaluation in a H-query or in a next LVrfy query. Of course we do not know what will be the right (h_2, τ) until the adversary output its forgery in the finalization phase. So, if (h^i_2, τ^i) denotes the input of the inversion of F in the i-th leaking verification query, we actually defines q_V targets $(\tilde{h}^i_1, h^i_2, \tau^i)$, since $i < q_V + 1$ here. Therefore, the probability that this case occurs is upper-bounded by $q_V(q_H + q_V)/2^{2n}$.

Proof. To prove the theorem, we use a sequence of games. Given an adversary A, we start with Game 0 which is the $\mathsf{FORGE\text{–}L2}^{\mathsf{suf\text{-}vcma}}_{\mathsf{A},\mathsf{HTBC},\mathsf{L}^*}$ experiment and we end with a game where all the leaking verification queries (m^i, τ^i) are deemed invalid, including the last and (q_V+1)-th verification which tests the validity of the potential forgery (m, τ). In the sequel, we note $\mathsf{H}(m) = h_1\|h_2$.

Game 0. This game is depicted in Tab 2. Let E_0 be the event that the adversary $\mathsf{A}^{\mathsf{L}^*}$ wins this game, that is, the output of the experiment is 1.

Game 1. We introduce a failure event F_1 with respect to Game 0, where F_1 occurs if among the at most $(q_H + q_T + q_V + 1)$ distinct hash computations there is at least one collision.

In Game 1, if F_1 occurs we abort the game and return 0. We let E_1 be the event that the adversary A^{L} wins this game.

Bounding $|\Pr[E_0] - \Pr[E_1]|$. Since Game 0 and Game 1 are identical as long as F_1 does not occur, and $4 \leq q_H + q_T + q_V$, we have

$$|\Pr[E_0] - \Pr[E_1]| \leq \Pr[F_1] \leq (q_H + q_T + q_V)^2/2^{2n}.$$

Note: from now on, in the case of a winning adversary, no TBC triple of the form $(\star, h_2, \tau)$ appears when answering to an LTag query.

Game 2. We modify the winning condition of the previous game. In the finalization, once A outputs (m, τ) we say that A does not win and returns 0 if A fails as before or if $\mathsf{H}(m) = h_1 \| h_2$ appears before the first apparition of (h_2, τ) as input to F_k^{-1} in a leaking verification query. If we call F_2 the event that makes the adversary winning in Game 1 but loosing in Game 2, we have $|\Pr[E_2] - \Pr[E_1]| \leq \Pr[F_2]$, where E_2 is the event that A wins in this game.

Bounding $\Pr[F_2]$. If we call V_i the event that the first time (h_2, τ) appears during the computation of the answer to a leaking verification query is in the i-th leaking verification query (m^i, τ^i), we just have to bound $\Pr[F_2 \cap V_i]$, for all $i = 1$ to $q_V + 1$. The event $F_2|V_i$ means m appears before m_i ($m = m_i$ included) and, if $\mathsf{H}(m^i) = h_1^i \| h_2^i$, we have $(h_2^i, \tau^i) = (h_2, \tau)$ while this never happens before in a previous LVrfy query. If $m = m_i$, $F_2|V_i$ reduces to the strong unpredictability of F with leakage. Indeed, it is easy to emulate LTag and LVrfy from LEval and LInv and to output the final triple (h_1^i, h_2^i, τ^i) against F at the time the i-th query is made in order to win with the same probability since (h_1^i, h_2^i) was never a LEval query (as m was never a LTag query in F_2 and there is no more collision since Game 1) and (h_2^i, τ^i) was never a LInv query before by definition of V_i. However, when $m \neq m^i$, we cannot follow such a simple strategy. Since $m \neq m^i$ we know that $h_1 \neq h_1^i$ as there is no collision by assumption and $(h_1^i, h_2^i = h_2, \tau^i = \tau)$ is not a winning triple against F. Of course, we could simply make an LInv query on (h_2^i, τ^i) to emulate the i-th LVrfy query but in that case we will actually "consume" our chance to win against F with (h_1, h_2, τ) because the input (h_2, τ) for inversion will be no more fresh after answering the i-th LVrfy query, and the reduction will fail. Fortunately, at the time we should emulate the i-th LVrfy query, we now that the history of the hash evaluations already contains the forged message m, that h_2 collides with h_2^i, and that τ^i is its valid tag. We thus have to make several hybrids on the collisions with h_2^i to anticipate the right m' to win against F with the triple $(h_1', h_2' = h_2^i, \tau^i)$. This number of hybrids is the number of collisions with h_2^i which remains small with high probability as it implies multi-collision in $\{0,1\}^n$. We note that only the hash evaluations of a H-query or of an LVrfy query matter. In the following, we write $\mathsf{H}_2(m') = h_2'$ when $\mathsf{H}(m') = h_1' \| h_2'$ for some h_1'.

Concretely, if q_i is the number of H evaluations made from all the H-queries and the LVrfy queries until the i-th LVrfy query, including $\mathsf{H}(m^i)$, and if S_i is the random variable counting

the number of H_2-collisions with h_2^i, for $i = 1$ to q_V, we have

$$\begin{aligned}\Pr[F_2 \cap V_i] &= \sum_{s=1}^{q_i} \Pr[F_2 \mid V_i \cap S_i = s] \cdot \Pr[V_i \cap S_i = s] \\ &\leq \Pr[F_2 \mid V_i \cap S_i = s \cap \textstyle\bigcup_{s=1}^{q_i} H_{i,s}] \\ &\quad + \sum_{s=2}^{q_i} \sum_{j=1}^{s-1} \Pr[F_2 \mid V_i \cap S_i = s \cap H_{i,j}] \cdot \Pr[\mathsf{H}_2\text{-}\mathsf{Coll}(q_i) \geq s]\end{aligned}$$

where $H_{i,j}$ is the event that among the s distinct messages that H_2-collide on h_2^i the j-th one is the forged message m. By convention, we always see m^i as the s-th and last such message even if the computation of $\mathsf{H}(m^i)$ appears earlier than in the i-th LVrfy query[6]. The case $\bigcup_{s=1}^{q_i} H_{i,s}$ thus corresponds to $m^i = m$ and the related probability in the expression is upper-bounded by ϵ_{SUL2} as explained above. Moreover, for each $j = 1$ to $s-1$, it is now easy to see that the event $F_2 \mid V_i \cap S_i = s \cap H_{i,j}$ reduces to $\mathsf{SUL2}$ by using the j-th message and τ^i as our guess against the TBC. Therefore,

$$\begin{aligned}\Pr[F_2 \cap V_i] &\leq \epsilon_{\mathsf{SUL2}} + \epsilon_{\mathsf{SUL2}} \sum_{s=2}^{q_i} (s-1) \cdot \Pr[\mathsf{H}_2\text{-}\mathsf{Coll}(q_i) \geq s] \\ &\leq \epsilon_{\mathsf{SUL2}} + \epsilon_{\mathsf{SUL2}} \cdot \frac{1}{2^n} \binom{q_i}{2} \left(1 + \frac{2q_i}{2^n}\right)\end{aligned}$$

by lemma 1, since $2q_i \leq 2(q_H + q_V) \leq 2^{n-2} \leq 2^n$ by assumption on the number of queries. In addition, $1 + 2q_i/2^n \leq 5/4$. Summing on all the i's, with $\Pr[F_2 \mid V_{q_V+1}] \leq \epsilon_{\mathsf{SUL2}}$, gives

$$\Pr[F_2] \leq (q_V + 1) \cdot \epsilon_{\mathsf{SUL2}} + \epsilon_{\mathsf{SUL2}} \cdot \frac{1}{2^n} \cdot \frac{5}{4} \cdot \sum_{i=2}^{q_V} \binom{q_i}{2}.$$

Some basic computation shows that $\sum_{i=1}^{q_V} \binom{q_i}{2} \leq \sum_{i=1}^{q_V} \binom{q_H+i}{2} \leq \frac{1}{2} q_H^2 q_V (1 + \frac{2q_V}{q_H})$, if $q_V \leq q_H$. But then, as $q_V \leq q_H/4$ by assumption, we have

$$\Pr[F_2] \leq (q_V + 1) \cdot \epsilon_{\mathsf{SUL2}} + \frac{q_H^2 q_V}{2^n} \cdot \epsilon_{\mathsf{SUL2}}.$$

Game 3. In this game we follow the specification of $\mathsf{FORGE\text{-}L2}^{\mathsf{suf\text{-}vcma}}_{\mathsf{A},\mathsf{HBC},\mathsf{L}^*}$ except that we always output 0 at the end of the game.

Bounding $|\Pr[E_3] - \Pr[E_2]| = \Pr[E_2]$. We end by showing that winning while the TBC input (h_2, τ) for inversion appears when answering a leaking verification query before the computation of $\mathsf{H}(m)$ is negligible. For each (h_2^i, τ^i) that appears when answering a leaking

[6] This is without loss of generality as the enumeration only matters in the reduction at the time we get the adversary's i-th LVrfy query (m^i, τ^i). The choice of (which is) the j-th message colliding on h_2^i can be made once the s messages are known. At that time, if $(h_2^i, \tau^i) = (h_2, \tau)$ as implied by V_i, $\mathsf{F}_k^{-1}(h_2, \tau)$ was never computed anyway.

verification query and *before* m' the only way for the adversary to win with the pair (m', τ') is to "hope" that $\mathsf{H}(m') = \tilde{h}_1^i \| h_2^i$. However, before the computation of $\mathsf{H}(m')$ its value remains independent of the adversary's view. Therefore, the probability that the random output of $\mathsf{H}(m')$ hits the full target $\tilde{h}_1^i \| h_2^i$ is $1/2^{2n}$. We now count the number of tries a winning adversary can make in it that case. Clearly, we do not have to count the tries related to any LTag. Considering the event V_i as in the previous analysis of Game 2, in $E_2 \cap V_i$ there at most q_H+q_V+1-i hash evaluations left after the i-th LVrfy query. Then, $\Pr[E_2 \mid V_i] \leq (q_H+q_V)/2^{2n}$ for $i = 1$ to q_V. Note that $\Pr[E_2 \mid V_{q_V+1}] = 0$ by definition. Finally, we get

$$\Pr[E_2] \leq \frac{q_V(q_H + q_V)}{2^{2n}}.$$

Hence, the bound of the theorem. □

6 Conclusion and open problems

We revisit the security proofs of MACs based on the "Hash-then-(T)BC" construction, an approach that has often been adopted in the context of leakage-resilient cryptography. While previous works have been modeling the (T)BC as a leak-free component, we only require that the (T)BC remains unpredictable in the presence of leakage, an assumption that has the major advantage of being easy to test on any implementation. We show that unpredictability with leakage is a suitable assumption for the analysis of the leakage-resilience of two standard MAC constructions, a result that has a direct impact on several recent constructions of AE modes of operation that are based on this approach. Apart from making security proofs more satisfactory, relying on unpredictability of the (T)BC rather than on its pseudorandomness prompts for investigating whether block cipher implementations that only seek to offer unpredictability with leakage could also require less rounds and, as a result, deliver better efficiency and cheaper protection against side-channel attacks.

As discussed in the paper, our bounds may not be tight due to some additional computations that are required in our reductions that actual adversaries may not need. Investigating whether these bounds can be improved is therefore an interesting challenge. Besides, for simplicity we model our hash function as an ideal random object (but do not require any form of progammability or other conveniences that come with the random oracle model). This fits well with many applications, for instance when the hash function is based on a sponge that is also traditionally modeled as an ideal permutation. Still, our analysis does not suggest any reason why an ideal object would be needed, and it would therefore be interesting to investigate whether and how this assumption could be relaxed.

As a final note, we conjecture that strong unpredictability with leakage may also be sufficient to prove confidentiality under ideal oracle (e.g., cipher, permutation) assumptions, assorted with oracle-free leakage functions, as introduced in [27] for block ciphers and recently used for the analysis of sponge-based designs [16,10]. Indeed, in such models the leakage about a key is useless as long as it does not lead to a full key-recovery. But as a

result, the interpretation of this unpredictability assumption in terms of quantitative security degradation (e.g., in the situation where an implementation is not leak-free but has high enough unpredictability with leakage) is also more delicate: the ideal objects transform an unpredictable value into something that is indistinguishable from random, hence possibly hiding the level of security degradation caused by the leakage.

Acknowledgments. Thomas Peters and François-Xavier Standaert are respectively postdoctoral researcher and senior research associate of the Belgian Fund for Scientific Research (F.R.S.-FNRS). This work has been funded in parts by the European Union through the ERC project SWORD (724725) and the Walloon Region FEDER USERMedia project 501907-379156. Chun Guo was partly supported by the Program of Qilu Young Scholars of Shandong University.

References

1. Jee Hea An and Mihir Bellare. Constructing vil-macsfrom fil-macs: Message authentication under weakened assumptions. In *CRYPTO*, volume 1666 of *LNCS*, pages 252–269. Springer, 1999.
2. Jean-Philippe Aumasson, Steve Babbage, Daniel J. Bernstein, Carlos Cid, Joan Daemen, Orr Dunkelman, Kris Gaj, Shay Gueron, Pascal Junod, Adam Langley, David McGrew, Kenny Paterson, Bart Preneel, Christian Rechberger, Vincent Rijmen, Matt Robshaw, Palash Sarkar, Patrick Schaumont, Adi Shamir, and Ingrid Verbauwhede. CHAE: Challenges in Authenticated Encryption. ECRYPT-CSA D1.1, Revision 1.05, 1 March 2017. https://chae.cr.yp.to/whitepaper.html.
3. 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 *LNCS*, pages 693–723. Springer, 2017.
4. Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In *CRYPTO*, volume 1109 of *LNCS*, pages 1–15. Springer, 1996.
5. Mihir Bellare, Joe Kilian, and Phillip Rogaway. The security of the cipher block chaining message authentication code. *J. Comput. Syst. Sci.*, 61(3):362–399, 2000.
6. Davide Bellizia, Francesco Berti, Olivier Bronchain, Gaëtan Cassiers, Sébastien Duval, Chun Guo, Gregor Leander, Gaëtan Leurent, Itamar Levi, Charles Momin, Olivier Pereira, Thomas Peters, François-Xavier Standaert, and Friedrich Wiemer. Spook: Sponge-based leakage-resilient authenticated encryption with a masked tweakable block cipher. *Submission to NIST Lightweight Cryptography, https://www.spook.dev/*, 2019.
7. Francesco Berti, Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Tedt, a leakage-resilient AEAD mode for high (physical) security applications. *IACR Cryptology ePrint Archive*, 2019:137, 2019.
8. 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.
9. 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.
10. Christoph Dobraunig and Bart Mennink. Leakage resilience of the duplex construction. *IACR Cryptology ePrint Archive*, 2019:225, 2019.
11. Yevgeniy Dodis, Eike Kiltz, Krzysztof Pietrzak, and Daniel Wichs. Message authentication, revisited. In *EUROCRYPT*, volume 7237 of *LNCS*, pages 355–374. Springer, 2012.
12. Yevgeniy Dodis and John P. Steinberger. Message authentication codes from unpredictable block ciphers. In *CRYPTO*, volume 5677 of *LNCS*, pages 267–285. Springer, 2009.
13. Yevgeniy Dodis and John P. Steinberger. Domain extension for macs beyond the birthday barrier. In *EUROCRYPT*, volume 6632 of *LNCS*, pages 323–342. Springer, 2011.
14. Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *FOCS*, pages 293–302. IEEE Computer Society, 2008.
15. Dahmun Goudarzi and Matthieu Rivain. How fast can higher-order masking be in software? In *EUROCRYPT (1)*, volume 10210 of *LNCS*, pages 567–597, 2017.
16. Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Towards low-energy leakage-resistant authenticated encryption from the duplex sponge construction. *IACR Cryptology ePrint Archive*, 2019:193.

17. 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). 11774:150–172, 2019.
18. Carmit Hazay, Adriana López-Alt, Hoeteck Wee, and Daniel Wichs. Leakage-resilient cryptography from minimal assumptions. In *EUROCRYPT*, volume 7881 of *LNCS*, pages 160–176. Springer, 2013.
19. Anthony Journault and François-Xavier Standaert. Very high order masking: Efficient implementation and security evaluation. In *CHES*, volume 10529 of *LNCS*, pages 623–643. Springer, 2017.
20. Yael Tauman Kalai and Leonid Reyzin. A survey of leakage-resilient cryptography. *IACR Cryptology ePrint Archive*, 2019:302, 2019.
21. Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *CRYPTO*, volume 1109 of *LNCS*, pages 104–113. Springer, 1996.
22. Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *CRYPTO*, volume 1666 of *LNCS*, pages 388–397. Springer, 1999.
23. Moses Liskov, Ronald L. Rivest, and David A. Wagner. Tweakable block ciphers. In *CRYPTO*, volume 2442 of *LNCS*, pages 31–46. Springer, 2002.
24. Daniel P. Martin, Elisabeth Oswald, Martijn Stam, and Marcin Wójcik. A leakage resilient MAC. In *IMA Int. Conf.*, volume 9496 of *LNCS*, pages 295–310. Springer, 2015.
25. Olivier Pereira, François-Xavier Standaert, and Srinivas Vivek. Leakage-resilient authentication and encryption from symmetric cryptographic primitives. In *ACM Conference on Computer and Communications Security*, pages 96–108. ACM, 2015.
26. Kazuhiro Suzuki, Dongvu Tonien, Kaoru Kurosawa, and Koji Toyota. Birthday paradox for multi-collisions. In *ICISC*, volume 4296 of *LNCS*, pages 29–40, 2006.
27. Yu Yu, François-Xavier Standaert, Olivier Pereira, and Moti Yung. Practical leakage-resilient pseudorandom generators. In *ACM Conference on Computer and Communications Security*, pages 141–151. ACM, 2010.
28. Liting Zhang, Wenling Wu, Peng Wang, Lei Zhang, Shuang Wu, and Bo Liang. Constructing rate-1 macs from related-key unpredictable block ciphers: PGV model revisited. In *FSE*, volume 6147 of *LNCS*, pages 250–269. Springer, 2010.