Leakage Resilient Value Comparison With Application to Message Authentication

Christoph Dobraunig[1,2] and Bart Mennink[3]

[1] Lamarr Security Research, Graz, Austria
[2] Graz University of Technology, Graz, Austria
[3] Radboud University, Nijmegen, The Netherlands
christoph.dobraunig@lamarr.at
b.mennink@cs.ru.nl

Abstract. Side-channel attacks are a threat to secrets stored on a device, especially if an adversary has physical access to the device. As an effect of this, countermeasures against such attacks for cryptographic algorithms are a well-researched topic. In this work, we deviate from the study of cryptographic algorithms and instead focus on the side-channel protection of a much more basic operation, the comparison of a known attacker-controlled value with a secret one. Comparisons sensitive to side-channel leakage occur in tag comparisons during the verification of message authentication codes (MACs) or authenticated encryption, but are typically omitted in security analyses. Besides, also comparisons performed as part of fault countermeasures might be sensitive to side-channel attacks. In this work, we present a formal analysis on comparing values in a leakage resilient manner by utilizing cryptographic building blocks that are typically part of an implementation anyway. Our results indicate that there is no need to invest additional resources into implementing a protected comparison operation itself if a sufficiently protected implementation of a public cryptographic permutation, or a (tweakable) block cipher, is already available. We complement our contribution by applying our findings to the SuKS message authentication code used by lightweight authenticated encryption scheme ISAP, and to the classical Hash-then-PRF construction.

Keywords: leakage resilience · value comparison · tag verification

1 Introduction

Side-channel attacks have been introduced to the public in the late 1990s [38, 39]. Especially differential power analysis (DPA) [39] turned out to be a very potent threat to implementations of cryptographic algorithms. A practical and sound countermeasure against differential power analysis is masking [12,28], and hence, a lot of research has been conducted in this direction bringing forward a myriad of different masking schemes [13,14,30,35,46,47,51,55]. Since the cost of masking is tied to the cryptographic primitive it protects, many newly designed

cryptographic primitives take protection against DPA into account and have been designed to reduce the cost of masking [3, 10, 15, 17, 21, 27, 29, 31, 52].

Later, a research direction called leakage resilient cryptography [7, 22–26, 32, 50, 54] emerged. In principle, leakage resilient cryptography leads to modes of operation that take side-channel attacks into account and, thus, ease the investment on side-channel countermeasures on the primitive level (e.g., the protection of the public cryptographic permutation or block cipher). For instance, research in this direction lead to modes of operation that protect against (higher-order) DPA without the need of applying (higher-order) masking [19, 20, 43, 44], or restrict the use of masking to a fraction of the building blocks [5, 8, 33, 49]. However, it is worth noting that leakage resilient modes can only solve part of the problem. Thus, the protection of primitives against potent and skillful attackers that can perform simple power analysis or template attacks is still crucial [37]. Nevertheless, leakage resilient schemes have been implemented on micro-controllers and practical evaluation shows that protection against side-channel attacks can be efficiently achieved in practice against a set of realistic adversaries [56].

An operation that is part of many cryptographic schemes, but also part of some fault countermeasures, is the comparison of two values for equality. In cases where this comparison is made between a value that an attacker should not know with a known and potentially chosen value, side-channel countermeasures for this comparison have to be in place.

Alternatively, one can perform the comparison using a cryptographic primitive. During the last years, many authenticated encryption schemes have been proposed that use a "perfectly" protected (tweakable) block cipher as the last step before the tag output [5, 7, 8, 33]. The advertised advantage of these schemes is that in the case of verification, the inverse of the "perfectly" protected (tweakable) block cipher can be applied to the candidate tag T. Then, the comparison of equality does not have to be done on the tag T directly but rather on an intermediate value. Therefore, the correct value of the tag $T^\star$ is never computed and cannot leak via side-channel attacks. Hence, the comparison operation itself does not need any protection against side-channel attacks. An alternative avenue is to use a public cryptographic permutation as, e.g., suggested in the ISAP v2.0 [19] specification. This approach was believed to have comparable advantages, with the added benefit that no key material needs to be protected.

Although various articles on this very topic appeared recently, culminating at a CRYPTO 2020 article [6], these works typically see the leakage resilient value comparison as integral part of a scheme or abstract the actual leakage resilience of the final value comparison and assume that it is "sufficiently leakage resilient secure." A formal qualitative and quantitative analysis of leakage resilient value comparison as a general method suitable for a wide range of applications is, despite its practical relevance, lacking.

1.1 Formal View on Leakage Resilient Value Comparison

In this paper, we present a formal leakage resilience study of comparing a secret value with a value chosen by an attacker, as would, e.g., typically happen during

verification of a tag. By considering the problem in isolation, it allows for a neater model and cleaner bound, that compose properly with broader schemes like authenticated encryption schemes or fault countermeasures.

In detail, the formal model considers a set of μ secretly computed target values $T_1, \ldots, T_\mu$, and an adversary that can guess values $T^\star$ for which value comparison succeeds. To resolve the fact that unprotected implementations of this comparison allow for DPA to recover any of the T_j's [6], we will incorporate a value processing function and consider comparison of *processed values*. This value processing function takes as additional input a salt S_j, tied to the target value T_j, and is based on a cryptographic primitive. The adversary wins the security game if it ever makes value comparison succeed, where it may be aided with side-channel leakage coming from the value processing and value comparison. This model captures the non-adaptive bounded leakage of the cryptographic functions and the leakage of the value comparison, a non-adaptive leakage model, and works in the standard model and ideal model dependent on the cryptographic function in use. It is described in Section 3.

1.2 Two Practical Solutions

In Sections 4 and 5, we present two concrete solutions that tackle the protection of value comparison. The first construction, PVP ("permutation-based value processing") of Section 4, processes the tag T and user input $T^\star$ along with a salt S using a cryptographic permutation to obtain an intermediate value U and $U^\star$, upon which comparison is evaluated. The cryptographic permutation can be a public permutation like Keccak-f [10] or a block cipher instantiated with a secret key like AES_K [18]. The construction with a public permutation is inspired by the informal proposal of the designers of ISAP v2.0 [19] to perform secure comparison. However, for PVP also the instantiation with a secret permutation is relevant, noting that this variant is naturally of use in implementations of schemes based on block ciphers that have an implementation of a heavily protected block cipher anyway, such as [1, 36, 40–42]. The scheme achieves very strong leakage resilience under the model defined in Section 3.

The second construction, TPVP ("tweakable permutation-based value processing") of Section 5, resembles much of PVP but is instantiated with a cryptographic tweakable permutation, which could in turn be a tweakable block cipher instantiated with a secret key like SKINNY_K [2]. The construction is particularly inspired by the idea to use a strongly protected tweakable block cipher for value comparison, as suggested by Berti et al. [8]. This construction, although different in nature from PVP, achieves comparable security.

These results are under the assumption that all target values $T_1, \ldots, T_\mu$ come with a unique and distinct salt $S_1, \ldots, S_\mu$. In Section 6 we discuss how the results on PVP and TPVP extend if one takes random salts or no salts at all.

1.3 Application to Message Authentication

A particularly interesting application is message authentication and authenticated encryption, after all the cradle of the problem that we tackle. In Section 7 we take a close look at how to apply the results from Sections 4 and 5 to message authentication.

The first construction that we present is StP ("SuKS then PVP"), a construction built as composition of the SuKS ("suffix keyed sponge") message authentication code [9, 20, 23] and the PVP value comparison function. In this construction, the SuKS function outputs a tag, and one also takes a salt from the internal computation of SuKS, and these values are fed to PVP for value comparison. We demonstrate that, in fact, leakage resilience of StP follows from the leakage resilient PRF security of SuKS and the leakage resilient value comparison security of PVP, provided that the two individual constructions are built on independent cryptographic primitives. In other words, the functions compose nicely and cheaply.

The second construction that we present is HaFuFu ("hash function function"), a hash-then-PRF message authentication code that uses the same PRF for value comparison. As the message authentication code and the value comparison function use the same cryptographic primitive, black-box composition is not an option. Instead, we prove direct security of HaFuFu, while still reusing many aspects of the security analysis of the schemes from Sections 4 and 5.

1.4 Comparison of Proposed Solutions

Our solutions fall into two categories depending on whether or not the used (tweakable) permutation is public or secret. In the case of public primitives, real-world instances would typically be based on public cryptographic permutations like Keccak-f [10], whereas for secret (tweakable) primitives, one would typically resort to (tweakable) block ciphers like AES [18] or SKINNY [2].

The most significant difference between using a public cryptographic permutation versus a (tweakable) block cipher is that the latter uses a secret key. Hence, this key has to be protected against a side-channel adversary that can freely choose inputs to this (tweakable) block cipher. As typical in this scenario, we assume that the block cipher is then perfectly protected [5, 7, 8, 33], meaning that the secret key cannot be extracted using a side-channel attack.

In contrast, basing the value comparison on public cryptographic permutations does not require to protect an additional secret value in addition to the candidate tag T. Hence, we *do not* require the assumption that the public cryptographic permutation is perfectly protected.

2 Preliminaries

Throughout the entire work, the parameters $k, m, n, c, r, s, t, u, p, q, \epsilon, \mu, \lambda, \lambda'$ are natural numbers. We denote by $\{0,1\}^n$ the set of n-bit strings. By $\mathsf{func}(m, n)$ we

define the set of all functions from $\{0,1\}^m$ to $\{0,1\}^n$, by $\mathsf{perm}(n) \subseteq \mathsf{func}(n,n)$ the set of permutations on $\{0,1\}^n$, and by $\mathsf{perm}(k,n)$ the set of families of 2^k permutations on $\{0,1\}^n$. We will write $\mathsf{func}(*,n)$ for the set of all functions from $\{0,1\}^*$ to $\{0,1\}^n$.

For a finite set $\mathcal{S}$, $S \xleftarrow{\$} \mathcal{S}$ denotes the uniformly random drawing of an element S from $\mathcal{S}$. We will sometimes abuse the notation a bit for infinite sets, as long as uniformly random sampling is possible. An example set is the family of functions $\mathsf{func}(*,n)$, for which uniformly random sampling can be simulated by lazy sampling (for each new input to the function, a random string of length n bits is generated). We denote by $S \xleftarrow{\epsilon} \mathcal{S}$ the drawing of an element S from $\mathcal{S}$ according to such a distribution that $\mathbf{Pr}\left(S = s\right) \leq 2^{\epsilon}/|S|$ for any $s \in \mathcal{S}$. Here, ϵ is some fixed constant which is typically required to be $\ll \log_2(|S|)$. Slightly abusing notation, we denote by $(S_1, \ldots, S_\mu) \xleftarrow{\epsilon} (\mathcal{S})^\mu$ the independent drawing of μ values $S_1, \ldots, S_\mu$ such that $\mathbf{Pr}\left(S_j = s\right) \leq 2^{\epsilon}/|S|$ for all $j = 1, \ldots, \mu$.

For a string $S \in \{0,1\}^n$, if $m \leq n$, we denote by $\mathrm{left}_m(S)$ (resp., $\mathrm{right}_m(S)$) the m leftmost (resp., rightmost) bits of S. For a predicate A, $[\![A]\!]$ equals 1 if A is true and 0 otherwise.

2.1 Multicollision Limit Function

We will use the notion of a multicollision limit function of Daemen et al. [16]. Consider the experiment of throwing q balls uniformly at random in 2^m bins, and let μ denote the maximum number of balls in a single bin. We define the multicollision limit function $\mathsf{mlf}^q_{m,n}$ as the smallest $x \in \mathbb{N}$ that satisfies

$$\mathbf{Pr}\left(\mu > x\right) \leq \frac{x}{2^n}\,.$$

Daemen et al. [16] demonstrated that this function is of the following order of magnitude:

$$\mathsf{mlf}^q_{m,n} \lesssim \begin{cases} (m+n)/\log_2\left(\dfrac{2^m}{q}\right)\,, & \text{for } q \lesssim 2^m\,, \\ (m+n)\cdot\dfrac{q}{2^m}\,, & \text{for } q \gtrsim 2^m\,. \end{cases}$$

In addition, if the balls are not thrown uniformly at random, but rather according to a distribution D that prescribes that the probability P that the i-th ball ends up in a certain bin satisfies

$$\frac{2^n - (i-1)}{2^{m+n} - (i-1)} \leq P \leq \frac{2^n}{2^{m+n} - (i-1)}\,, \tag{1}$$

the corresponding multicollision function, defined as $\mathsf{mlf}^{D,q}_{m,n}$, satisfies $\mathsf{mlf}^{D,q}_{m,n} \leq \mathsf{mlf}^{2q}_{m,n}$ [16, Lemma 6].

2.2 Block Ciphers and Tweakable Block Ciphers

A block cipher $\mathsf{E} : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ is a family of n-bit permutations indexed by a key $K \in \{0,1\}^k$. Its security is typically measured by the PRP-advantage. In detail, an adversary is given query access to either E_K for random and secret key $K \xleftarrow{\$} \{0,1\}^k$, or to a random permutation $\mathsf{P} \xleftarrow{\$} \mathsf{perm}(n)$, and its goal is to distinguish both worlds:

$$\mathbf{Adv}^{\mathrm{prp}}_{\mathsf{E}}(\mathcal{A}) = \left|\mathbf{Pr}\left(K \xleftarrow{\$} \{0,1\}^k \ : \ \mathcal{A}^{\mathsf{E}_K} = 1\right) - \mathbf{Pr}\left(\mathsf{P} \xleftarrow{\$} \mathsf{perm}(n) \ : \ \mathcal{A}^{\mathsf{P}} = 1\right)\right| .$$

Denoting by $\mathbf{Adv}^{\mathrm{prp}}_{\mathsf{E}}(q,\tau)$ the maximum advantage over any adversary making q construction queries and operating in time τ, the block cipher E is called PRP-secure if $\mathbf{Adv}^{\mathrm{prp}}_{\mathsf{E}}(q,\tau)$ is small.

A tweakable block cipher $\mathsf{TE} : \{0,1\}^k \times \{0,1\}^r \times \{0,1\}^n \to \{0,1\}^n$ is a family of n-bit permutations indexed by a key $K \in \{0,1\}^k$ and a tweak $R \in \{0,1\}^r$. Its security is typically measured by the TPRP-advantage. In detail, an adversary is given query access to either TE_K for random and secret key $K \xleftarrow{\$} \{0,1\}^k$, or to a family of random permutations $\mathsf{TP} \xleftarrow{\$} \mathsf{perm}(r,n)$, and its goal is to distinguish both worlds:

$$\mathbf{Adv}^{\mathrm{tprp}}_{\mathsf{TE}}(\mathcal{A}) = \left|\mathbf{Pr}\left(K \xleftarrow{\$} \{0,1\}^k \ : \ \mathcal{A}^{\mathsf{TE}_K} = 1\right) - \mathbf{Pr}\left(\mathsf{TP} \xleftarrow{\$} \mathsf{perm}(r,n) \ : \ \mathcal{A}^{\mathsf{TP}} = 1\right)\right| .$$

Denoting by $\mathbf{Adv}^{\mathrm{tprp}}_{\mathsf{TE}}(q,\tau)$ the maximum advantage over any adversary making q construction queries and operating in time τ, the block cipher TE is called TPRP-secure if $\mathbf{Adv}^{\mathrm{tprp}}_{\mathsf{TE}}(q,\tau)$ is small.

3 Security Model for Value Comparison

We will present a security model for leakage resilient value comparison. To do so, we first describe how, perhaps pedantically, value comparison in the black-box model can be modeled (Section 3.1). Then, we explain how value comparison in a leaky model can be described in Section 3.2. The model of leakage resilient value comparison is then given in Section 3.3.

3.1 Value Comparison in Black-Box Model

In a black-box setting, value comparison is trivial. If a tag $T^\star \in \{0,1\}^t$ must be tested against a target value $T \in \{0,1\}^t$, one simply performs a comparison, and outputs 1 if and only if the values are correct. We can capture this by the following, trivial, value comparison function $\mathsf{VC} : \{0,1\}^t \times \{0,1\}^t \to \{0,1\}$:

$$\mathsf{VC}(T, T^\star) = \left[\!\left[T \stackrel{?}{=} T^\star \right]\!\right] . \tag{2}$$

For the pure sake of understanding the model of leakage resilient value comparison in Section 3.3, it makes sense to formally define value comparison security in the black-box model. The model is entirely trivial, but we write it in a slightly more complex way to suit further discussion. This is done by considering an adversary $\mathcal{A}$ that engages in the following game. Prior to the game, a list of μ target values $\boldsymbol{T} = (T_1, \ldots, T_\mu) \xleftarrow{\$} (\{0,1\}^t)^\mu$ is randomly generated. The adversary has query access to a value comparison oracle

$$\mathcal{O}_{\boldsymbol{T}} : (j, T^\star) \mapsto \left[\!\left[T_j \stackrel{?}{=} T^\star \right]\!\right] .$$

It wins if $\mathcal{O}_{\boldsymbol{T}}$ ever outputs 1:

$$\mathbf{Adv}^{\mathrm{vc}[\mu]}_{\mathcal{O}}(\mathcal{A}) = \mathbf{Pr}\left(\boldsymbol{T} \xleftarrow{\$} (\{0,1\}^t)^\mu \ : \ \mathcal{A}^{\mathcal{O}_{\boldsymbol{T}}} \text{ wins} \right) . \tag{3}$$

For completeness, we can define by $\mathbf{Adv}^{\mathrm{vc}[\mu]}_{\mathcal{O}}(q)$ the maximum advantage over any adversary making q queries. To confirm that the model is entirely trivial: if $\mathcal{A}$ has q guessing attempts, its success probability is at most $q/2^t$. However, as mentioned, it makes sense to describe this model as starter for the model of leakage resilient value comparison in Section 3.3.

3.2 Value Comparison in Leaky Model

In a leaky setting, plain value comparison as in Section 3.1 is risky: performing the comparison may potentially leak data [6]. In detail, an adversary can repeatedly perform verification attempts against a single target value T_j, and each verification attempt might leak a certain number of bits of information about T_j. In addition, leakage obtained in a verification attempt against one target value T_j might be useful for a later verification against another target value $T_{j'}$. Besides securing (masking) the comparison itself, another method proposed to counter such side-channel attacks is to pre-process tags with a cryptographic value processing function, and *compare the processed tags*. This value processing function is, in turn, based on a cryptographic function.

Let $\Pi \in \mathsf{perm}(r, n)$ be a cryptographic primitive. A value processing function is a function $\mathsf{VP}^\Pi : \{0,1\}^s \times \{0,1\}^t \to \{0,1\}^u$ that gets as input a salt S, value T, and processes it using cryptographic primitive Π to obtain a value U. Now, the basic idea is to not perform value comparison on $(T, T^\star)$ directly (as in (2)), but rather on the subtags:

$$\mathsf{VC}(\mathsf{VP}^\Pi(S,T), \mathsf{VP}^\Pi(S,T^\star)) = \left[\!\left[\mathsf{VP}^\Pi(S,T) \stackrel{?}{=} \mathsf{VP}^\Pi(S,T^\star) \right]\!\right] . \tag{4}$$

Remark 1. Looking ahead, for $r = 0$, the cryptographic primitive Π might be a public permutation that can in practice then be instantiated with a strong permutation like Keccak-f [10], or it could be a secret permutation that could for instance be instantiated with AES_K [18] for a secret key. The difference is subtle. In the former case, an adversary knows the permutation and can make

queries to it. In the latter case, the adversary cannot make primitive evaluations, but this instantiation comes at the cost of the PRP-security of AES. In addition, the implementation of AES_K must then be strongly protected to prevent the key from leaking. We will elaborate on this in Sections 4.2 and 4.3.

Likewise, if $r > 0$, the cryptographic primitive $\mathsf{\Pi}$ might be a public tweakable permutation (like keyless SKINNY) or a secret tweakable permutation that could for instance be instantiated with SKINNY [2]. Also here, the same differences between the two cases surface. We will elaborate on these two cases in Sections 5.2 and 5.3.

Remark 2. Although our focus is on value processing functions instantiated with a (public or secret) family of permutations, the definition and later security models readily extend to instantiations with a different type of primitive, such as an arbitrary function $\mathsf{F} \in \mathsf{func}(r, n)$.

3.3 Security Model for Leakage Resilient Value Comparison

A straightforward generalization of the security model of Section 3.1 would be to consider a random $\mathsf{\Pi} \xleftarrow{\$} \mathsf{perm}(r, n)$, a list of μ distinct salts $\boldsymbol{S} = (S_1, \ldots, S_\mu) \subseteq \{0,1\}^s$ and a list of μ target values $\boldsymbol{T} = (T_1, \ldots, T_\mu) \xleftarrow{\in} (\{0,1\}^t)^\mu$, where we recall that each of the μ values T_j has min-entropy of at least $t - \epsilon$. This allows us to model the information an attacker might get via side-channels during the generation of the values T_j outside of our observation that just focuses on the value comparison and the leakage occurring there. Furthermore, we consider an adversary that has query access to a value comparison oracle

$$\mathcal{O}^{\mathsf{VP},\mathsf{\Pi}}_{\boldsymbol{S},\boldsymbol{T}} : (j, T^\star) \mapsto \left[\!\left[\mathsf{VP}^{\mathsf{\Pi}}(S_j, T_j) \stackrel{?}{=} \mathsf{VP}^{\mathsf{\Pi}}(S_j, T^\star) \right]\!\right] . \tag{5}$$

The adversary can learn the salts $\boldsymbol{S}$. It a priori has bi-directional access to $\mathsf{\Pi}$ (if $\mathsf{\Pi}$ is a secret permutation, the number of queries to $\mathsf{\Pi}$ is bounded to 0, below).

However, it is not as simple as that: we will consider value comparison security in case of leakage resilience. We will restrict our focus to non-adaptive $\mathcal{L}$-resilience of Dodis and Pietrzak [24], where the adversary receives leakage under any leakage $\mathsf{L} \in \mathcal{L}$ of the scheme under investigation. In our case, leakage of secret data can occur in two occasions: evaluation of $\mathsf{\Pi}$ within the two evaluations of $\mathsf{VP}^{\mathsf{\Pi}}$, and the value comparison. Therefore, $\mathcal{L}$ consists of a Cartesian product of two leakage sets.

Let $\mathcal{L}_{\mathsf{\Pi}} = \{\mathsf{L}_{\mathsf{\Pi}} \colon \{0,1\}^r \times \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^\lambda\}$ be a fixed set of leakage functions on the primitive $\mathsf{\Pi}$ within the value processing function VP, and let $\mathcal{L}_{\mathsf{C}} = \{\mathsf{L}_{\mathsf{C}} \colon \{0,1\}^u \times \{0,1\}^u \to \{0,1\}^{\lambda'}\}$ be a fixed set of leakage functions on the value comparison function VC. All functions are independent of $\mathsf{\Pi}$, i.e., they do not internally evaluate $\mathsf{\Pi}$ or $\mathsf{\Pi}^{-1}$. Write $\mathcal{L} = \mathcal{L}_{\mathsf{\Pi}} \times \mathcal{L}_{\mathsf{C}}$. For any leakage function $\mathsf{L} = (\mathsf{L}_{\mathsf{\Pi}}, \mathsf{L}_{\mathsf{C}}) \in \mathcal{L}$, define by $\left[\mathcal{O}^{\mathsf{VP},\mathsf{\Pi}}_{\boldsymbol{S},\boldsymbol{T}}\right]_{\mathsf{L}}$ an evaluation of $\mathcal{O}^{\mathsf{VP},\mathsf{\Pi}}_{\boldsymbol{S},\boldsymbol{T}}$ of (5) that not only

returns the response of this function, but also leaks the following values:

$$\mathsf{L}_{\Pi}(X,Y) \in \{0,1\}^{\lambda} \ \left(\forall\ \Pi\text{-evaluation } (X,Y) \text{ in } \mathsf{VP}^{\Pi}(S_j,T_j)\right),$$
$$\mathsf{L}_{\Pi}(X,Y) \in \{0,1\}^{\lambda} \ \left(\forall\ \Pi\text{-evaluation } (X,Y) \text{ in } \mathsf{VP}^{\Pi}(S_j,T^{\star})\right),$$
$$\mathsf{L}_{\mathsf{C}}\left(\mathsf{VP}^{\Pi}(S_j,T_j), \mathsf{VP}^{\Pi}(S_j,T^{\star})\right) \in \{0,1\}^{\lambda'}.$$

The security model of Section 3.2 now extends as suggested in the beginning of this section, but with $\mathcal{A}$ having access to the *leaky variant* of $\mathcal{O}^{\mathsf{VP},\Pi}_{\boldsymbol{S},\boldsymbol{T}}$. In detail, consider an adversary $\mathcal{A}$ that, for any given tuple of leakage functions $\mathsf{L} = (\mathsf{L}_{\Pi}, \mathsf{L}_{\mathsf{C}}) \in \mathcal{L}$ and any tuple of μ distinct salts $\boldsymbol{S} \subseteq \{0,1\}^s$, has query access to $\left[\mathcal{O}^{\mathsf{VP},\Pi}_{\boldsymbol{S},\boldsymbol{T}}\right]_{\mathsf{L}}$ and bi-directional access to Π (bounded to 0 queries if Π is a secret permutation). The adversary wins if $\left[\mathcal{O}^{\mathsf{VP},\Pi}_{\boldsymbol{S},\boldsymbol{T}}\right]_{\mathsf{L}}$ ever outputs 1:

$$\mathbf{Adv}^{\text{lr-vc}[\mu]}_{\mathcal{O}^{\mathsf{VP}}}(\mathcal{A}) = \max_{\mathsf{L}=(\mathsf{L}_{\Pi},\mathsf{L}_{\mathsf{C}})\in\mathcal{L}} \max_{\boldsymbol{S}\subseteq\{0,1\}^s} \mathbf{Pr}\left(\Pi \xleftarrow{\$} \mathsf{perm}(r,n)\,,\, \boldsymbol{T} \xleftarrow{\$} (\{0,1\}^t)^{\mu} \ :\ \mathcal{A}^{\left[\mathcal{O}^{\mathsf{VP},\Pi}_{\boldsymbol{S},\boldsymbol{T}}\right]_{\mathsf{L}},\Pi^{\pm}}(\boldsymbol{S}) \text{ wins}\right). \quad (6)$$

For completeness, we can define by $\mathbf{Adv}^{\text{lr-vc}[\mu]}_{\mathcal{O}^{\mathsf{VP}}}(q,p)$ the maximum advantage over any adversary making q queries to $\left[\mathcal{O}^{\mathsf{VP},\Pi}_{\boldsymbol{S},\boldsymbol{T}}\right]_{\mathsf{L}}$ and p bi-directional queries to $\Pi^{\pm}$. In the bigger picture, q refers to the number of verification queries an adversary can make. In case the primitive Π is a secretly keyed primitive, one restricts to $p = 0$.

4 Value Comparison Based on Permutation

Let $\mathsf{P} \in \mathsf{perm}(n)$ be a permutation (for now, we will not yet limit ourselves to secret or public permutation). Assume that $\log_2(\mu) \leq s$ and $s + t, u \leq n$. Define the following, arguably most straightforward, permutation-based value processing function $\mathsf{PVP}^{\mathsf{P}} : \{0,1\}^s \times \{0,1\}^t \to \{0,1\}^u$:

$$\mathsf{PVP}^{\mathsf{P}}(S,T) = \text{left}_u(\mathsf{P}(S \parallel T \parallel 0^{n-s-t})). \quad (7)$$

Value verification then follows as in (4), using above value processing function PVP (see also Figure 1):

$$\mathsf{PVC}(\mathsf{PVP}^{\mathsf{P}}(S,T), \mathsf{PVP}^{\mathsf{P}}(S,T^{\star})) = \left[\!\left[\mathsf{PVP}^{\mathsf{P}}(S,T) \stackrel{?}{=} \mathsf{PVP}^{\mathsf{P}}(S,T^{\star})\right]\!\right]. \quad (8)$$

A general security bound of value comparison using PVP is given in Section 4.1. Note that we did not put any stringent condition on s, t, u, and n yet: all we need is that $s + t, u \leq n$. Depending on whether P is a secret or public permutation, an additional condition is needed. Both cases are rather different in nature, in practical appearance, and in the security level that they achieve. We elaborate on the case of secret permutation in Section 4.2, and on the case of public permutation in Section 4.3.

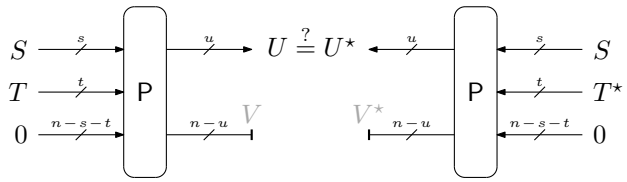

Fig. 1: Depiction of leakage resilient value comparison using permutation.

4.1 Leakage Resilience of Value Comparison With PVP

We derive a general bound on the leakage resilience of value comparison using PVP,

$$\mathcal{O}^{\mathsf{PVP},\mathsf{P}}_{\boldsymbol{S},\boldsymbol{T}} : (j, T^\star) \mapsto \left[\!\left[\mathsf{PVP}^{\mathsf{P}}(S_j, T_j) \stackrel{?}{=} \mathsf{PVP}^{\mathsf{P}}(S_j, T^\star) \right]\!\right] , \tag{9}$$

in the security definition of (6) against any adversary making q construction queries and p primitive queries. We note that the bound is meaningless for certain choices of n, s, t, u, q, p: in particular, if $p > 0$ (i.e., if we consider instantiation using a *public* permutation), one requires $t, u \ll n$. The bound is nevertheless derived in full generality, and will only be interpreted for the specific cases in Sections 4.2 and 4.3.

Theorem 1. *Assume that* $\log_2(\mu) \leq s$ *and* $s + t, u \leq n$. *For any adversary* $\mathcal{A}$ *with construction complexity* q *and primitive complexity* p,

$$\begin{aligned}\mathbf{Adv}^{\text{lr-vc}[\mu]}_{\mathcal{O}^{\mathsf{PVP}}}(\mathcal{A}) \leq\ & \frac{2(q+p)}{2^{\min\{t-\epsilon-\lambda,u\}} - (\mu+q+p)} \\ & + \frac{2\mathsf{mlf}^{2\mu}_{u,n-u}p}{2^{n-\max\{t,u+\lambda\}} - (\mu+q+p)} + \frac{\mathsf{mlf}^{2\mu}_{u,n-u}}{2^{n-u}} .\end{aligned}$$

Proof. Let $\mathsf{L} = (\mathsf{L}_\mathsf{P}, \mathsf{L}_\mathsf{C}) \in \mathcal{L}$ be any two leakage functions and let $\boldsymbol{S} \subseteq \{0,1\}^s$ be a list of q distinct salts. Let $\mathsf{P} \xleftarrow{\$} \mathsf{perm}(n)$ be a random permutation, and let $\boldsymbol{T} \xleftarrow{\epsilon} (\{0,1\}^t)^\mu$ be a list of μ target values T_j, where each T_j has at least a min-entropy of at least $t - \epsilon$. For any $j \in \{1, \ldots, \mu\}$, define $\mathsf{P}(S_j \| T_j \| 0^{n-s-t}) = U_j \| V_j$, where $U_j \in \{0,1\}^u$ and $V_j \in \{0,1\}^{n-u}$. By definition, we have $U_j = \mathsf{PVP}^{\mathsf{P}}(S_j, T_j)$. Consider any adversary $\mathcal{A}$ that can make q queries $(j, T^\star)$ to $\mathcal{O}^{\mathsf{PVP},\mathsf{P}}_{\boldsymbol{S},\boldsymbol{T}}$ of (9), and p direct queries to $\mathsf{P}^{\pm}$. For each of the q construction queries, $\mathcal{A}$ also learns the following values:

$$\begin{aligned}\mathsf{L}_\mathsf{P}\left(S_j \| T_j \| 0^{n-s-t}, U_j \| V_j\right) &\in \{0,1\}^\lambda , \\ \mathsf{L}_\mathsf{P}\left(S_j \| T^\star \| 0^{n-s-t}, \mathsf{P}(S_j \| T^\star \| 0^{n-s-t})\right) &\in \{0,1\}^\lambda , \\ \mathsf{L}_\mathsf{C}\left(U_j, \mathsf{PVP}^{\mathsf{P}}(S_j, T^\star)\right) &\in \{0,1\}^{\lambda'} .\end{aligned}$$

Note that, as L_P and L_C are fixed, predetermined, functions, the adversary learns *at most* λ bits of leakage on T_j, λ bits of leakage on V_j, and $\lambda + q\lambda'$ bits of leakage on U_j, for any $j \in \{1, \ldots, \mu\}$.

The adversary wins if any of its q construction queries returns 1. However, the probability for this to occur depends on "lucky" primitive queries. In detail, if the adversary happens to make a primitive query of the form

$$(S_j \parallel *^t \parallel 0^{n-s-t}, U_j \parallel *^{n-u}),$$

for any $j \in \{1, \ldots, \mu\}$, it can use this to make the construction oracle output 1 with probability 1. Therefore, we *also* say that the adversary wins if any of its p primitive queries is of above form. Finally, it turns out that the adversary might have a significantly increased success probability if there exists a multicollision in $\{U_1, \ldots, U_\mu\}$. We will also count that as a win for the adversary.

More detailed, write $\mathsf{m} = \mathsf{mlf}^{2\mu}_{u,n-u}$ for brevity. We denote by bad the event that there exist $\mathsf{m}+1$ distinct indices $j_1, \ldots, j_{\mathsf{m}+1} \in \{1, \ldots, \mu\}$ such that $U_{j_1} = \cdots = U_{j_{\mathsf{m}+1}}$. In addition, for $i \in \{1, \ldots, q+p\}$, we denote by win_i the event that the i-th query is

- a construction query $(j, T^\star)$ that satisfies $\mathsf{PVP}^{\mathsf{P}}(S_j, T^\star) = U_j$, or
- a primitive query (X, Y) that satisfies $\mathrm{left}_s(X) = S_j$, $\mathrm{right}_{n-s-t}(X) = 0^{n-s-t}$, and $\mathrm{left}_u(Y) = U_j$ for some $j \in \{1, \ldots, \mu\}$.

Write $\mathsf{win} = \bigvee_{i=1}^{q+p} \mathsf{win}_i$. Our goal is to bound

$$\begin{aligned}
\mathbf{Pr}(\mathsf{win}) &\leq \mathbf{Pr}(\mathsf{win} \wedge \neg\mathsf{bad}) + \mathbf{Pr}(\mathsf{bad}) \\
&= \mathbf{Pr}\left(\bigvee_{i=1}^{q+p} \mathsf{win}_i \wedge \neg\mathsf{bad}\right) + \mathbf{Pr}(\mathsf{bad}) \\
&\leq \sum_{i=1}^{q+p} \mathbf{Pr}(\mathsf{win}_i \wedge \neg\mathsf{win}_{1..i-1} \wedge \neg\mathsf{bad}) + \mathbf{Pr}(\mathsf{bad}), \qquad (10)
\end{aligned}$$

where $\mathsf{win}_{1..0} = \mathsf{false}$ by definition.

Bound on $\mathbf{Pr}(\mathsf{win}_i \wedge \neg\mathsf{win}_{1..i-1} \wedge \neg\mathsf{bad})$. Consider any $i \in \{1, \ldots, q+p\}$, and consider the i-th query. We will make a distinction between a construction query, forward primitive query, and inverse primitive query.

- Construction query. Consider any construction query $(j, T^\star)$ to $\mathcal{O}^{\mathsf{PVP},\mathsf{P}}_{\boldsymbol{S},\boldsymbol{T}}$. If there were an earlier primitive query of the form $S_j \| T^\star \| 0^{n-s-t}$, then by $\neg\mathsf{win}_{1..i-1}$ its outcome is not of the form $U_j \| *^{n-u}$, and the oracle will not output 1. Therefore, we can assume that this query has not been made directly to P yet.
  The oracle outputs 1 if:
  - $T^\star = T_j$. As the values T_j are randomly generated with a min-entropy of at least $t - \epsilon$, and as the adversary has so far learned at most λ bits of leakage on T_j, this condition is set with probability at most $1/2^{t-\epsilon-\lambda}$;
  - $T^\star \neq T_j$ but $\mathsf{PVP}^{\mathsf{P}}(S_j, T^\star) = U_j$. As there was no earlier evaluation of $\mathsf{P}(S_j \| T^\star \| 0^{n-s-t})$, the result will be randomly drawn from a set of size at least $2^n - (\mu + i - 1) \geq 2^n - (\mu + q + p)$ values, and at most 2^{n-u} of these satisfy $\mathsf{PVP}^{\mathsf{P}}(S_j, T^\star) = U_j$. Thus, the condition is set with probability at most $2^{n-u}/(2^n - (\mu + q + p))$.

Adding both cases, we get

$$\mathbf{Pr}\left(\mathsf{win}_i \wedge \neg\mathsf{win}_{1..i-1} \wedge i\text{-th query to construction}\right) \leq \frac{2}{2^{\min\{t-\epsilon-\lambda,u\}} - (\mu+q+p)}\,; \quad (11)$$

– Forward primitive query. Consider any forward primitive query (X, Y) to P. Without loss of generality, $X = S_j \| T^\star \| 0^{n-s-t}$ for some $j \in \{1, \ldots, \mu\}$ and $T^\star \in \{0,1\}^t$ (otherwise, the query cannot set win_i). Note that the value j is unique as $\boldsymbol{S}$ *is assumed to contain no collisions*. We can also assume that neither this query has been made to P yet, nor $(j, T^\star)$ has been queried to the construction oracle before.
Now, the forward primitive query sets win_i if $T^\star = T_j$ or if $Y = U_j \| *^{n-u}$, and the analysis is identical to that of construction queries. We thus obtain

$$\mathbf{Pr}\left(\mathsf{win}_i \wedge \neg\mathsf{win}_{1..i-1} \wedge i\text{-th query to forward primitive}\right) \leq \frac{2}{2^{\min\{t-\epsilon-\lambda,u\}} - (\mu+q+p)}\,; \quad (12)$$

– Inverse primitive query. Consider any inverse primitive query (X, Y) to P. We can assume that this query has not been made to P yet. At the point of making this primitive query, the adversary has learned at most $\lambda + q\lambda'$ bits of information about all U_j's. We will be more generous, and assume w.l.o.g. that any inverse query is of the form $U_j \| V^\star$ for some $j \in \{1, \ldots, \mu\}$ and $V^\star \in \{0,1\}^{n-u}$. Note that the value j might not be unique as there might be collisions in $\{U_1, \ldots, U_\mu\}$. However, due to $\neg\mathsf{bad}$, the largest size of a multicollision is at most $\mathsf{mlf}^{2\mu}_{u,n-u}$. Therefore, there are at most $\mathsf{mlf}^{2\mu}_{u,n-u}$ possible values j.
The inverse primitive query sets win_i if for any of these possible values j:
  - $V^\star = V_j$. As the adversary has so far learned at most λ bits of leakage on V_j, this condition is set with probability at most $1/2^{n-u-\lambda}$;
  - $V^\star \neq V_j$ but $X = S_j \| T^\star \| 0^{n-s-t}$ for some $T^\star$. As there was no earlier evaluation of $\mathsf{P}^{-1}(U_j \| V^\star)$, the result will be randomly drawn from a set of size at least $2^n - (\mu + i - 1) \geq 2^n - (\mu + q + p)$ values, and at most 2^t of these satisfy $\mathrm{left}_s(X) = S_j$ and $\mathrm{right}_{n-s-t}(X) = 0^{n-s-t}$. Thus, the condition is set with probability at most $2^t/(2^n - (\mu+q+p))$.

Adding both cases, and summing over all $\leq \mathsf{mlf}^{2\mu}_{u,n-u}$ possible value j, we get

$$\mathbf{Pr}\left(\mathsf{win}_i \wedge \neg\mathsf{win}_{1..i-1} \wedge i\text{-th query to inverse primitive}\right) \leq \frac{2\mathsf{mlf}^{2\mu}_{u,n-u}}{2^{n-\max\{t,u+\lambda\}} - (\mu+q+p)}\,. \quad (13)$$

Bound on $\mathbf{Pr}\left(\mathsf{bad}\right)$. The values U_j are all uniformly randomly drawn from a set of size $2^n - (j-1)$ values, and they are truncated to take any value from a set

of 2^u elements. The event is thus a balls-and-bins experiment in the notation of Section 2.1 with μ balls randomly thrown into 2^u bins, in such a way that any of the bins contains more than $\mathsf{mlf}^{2\mu}_{u,n-u}$ balls. The distribution satisfies the condition of (1). Therefore, we obtain that

$$\mathbf{Pr}\left(\mathsf{bad}\right) \leq \frac{\mathsf{mlf}^{2\mu}_{u,n-u}}{2^{n-u}}\,. \tag{14}$$

Conclusion. The adversary makes q construction queries, each of which succeeds with probability at most (11), and p primitive queries, each of which succeeds with probability the maximum of (12) and (13). For simplicity, we do not maximize, but rather take the sum. Finally, we have to add (14). We thus obtain from (10) that

$$\begin{aligned}\mathbf{Adv}^{\text{lr-vc}[\mu]}_{\mathcal{O}^{\mathsf{PVP}}}(\mathcal{A}) \leq\ & \frac{2(q+p)}{2^{\min\{t-\epsilon-\lambda,u\}}-(\mu+q+p)} \\ & + \frac{2\mathsf{mlf}^{2\mu}_{u,n-u}p}{2^{n-\max\{t,u+\lambda\}}-(\mu+q+p)} + \frac{\mathsf{mlf}^{2\mu}_{u,n-u}}{2^{n-u}}\,.\end{aligned}$$

The reasoning holds for any adversary making q construction queries and p primitive queries, and this completes the proof. □

4.2 PVP with Secret Permutation

Let $\mathsf{E} : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher. If E is PRP-secure (see Section 2.2), one can instantiate the *secret* permutation P in the value processing function $\mathsf{PVP}^{\mathsf{P}}$ using the block cipher with a secret key, and de facto consider

$$\mathsf{EVP}^{\mathsf{E}_K}(S,T) = \mathrm{left}_u(\mathsf{E}_K(S \parallel T \parallel 0^{n-s-t}))\,. \tag{15}$$

A value comparison via an inverse block cipher call is part of the constructions proposed in [8].

The security bound of Theorem 1 carries over to EVP, with the following four changes:

- The term $\mathbf{Adv}^{\mathrm{prp}}_{\mathsf{E}}(q,\tau)$ is added (where q is exactly the number of queries described in Theorem 1 and τ is an additional time complexity measure on $\mathcal{A}$);
- The function E_K must be strongly protected, so that the function leaks no information about its inputs and outputs;
- The number of primitive queries is bounded to $p = 0$;
- As the number of primitive queries is bounded to $p = 0$, the auxiliary bad event bad has become obsolete, and hence the term $\mathsf{mlf}^{2\mu}_{u,n-u}/2^{n-u}$ disappears.

More formally, we obtain the following corollary. Notably, the sole term with $2^{n-\max\{t,u\}}$ in the denominator disappeared, and we do not need to put any condition on $n - \max\{t,u\}$.

Corollary 1 (Value Comparison Using Block Cipher). *Assume that* $\log_2(\mu) \leq s$ *and* $s + t, u \leq n$*. Let* $\mathsf{E} : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ *be a block cipher that is perfectly protected. For any adversary* $\mathcal{A}$ *with construction complexity* q *and operating in time* τ*,*

$$\mathbf{Adv}^{\text{lr-vc}[\mu]}_{\mathcal{O}^{\mathsf{EVP}}}(\mathcal{A}) \leq \frac{2q}{2^{\min\{t-\epsilon-\lambda,u\}} - (\mu+q)} + \mathbf{Adv}^{\text{prp}}_{\mathsf{E}}(q,\tau)\,.$$

4.3 PVP with Public Permutation

Assuming that P is a *public* permutation, the permutation-based value processing function PVP of (7) is similar to the one proposed by the designers of NIST Lightweight Cryptography candidate ISAP [19]. In this case, the adversary can evaluate the public primitive, or in terms of Theorem 1: $p > 0$. This also means that, for the last term of this theorem to be small, we require $t, u \ll n$. We obtain the following corollary:

Corollary 2 (Value Comparison Using Permutation). *Assume that* $\log_2(\mu) \leq s \leq n - t$ *and* $t, u \ll n$*. Let* $\mathsf{P} \in \mathsf{perm}(n)$ *be a permutation that is assumed to be perfectly random. For any adversary* $\mathcal{A}$ *with construction complexity* q *and primitive complexity* p*,*

$$\begin{aligned}\mathbf{Adv}^{\text{lr-vc}[\mu]}_{\mathcal{O}^{\mathsf{PVP}}}(\mathcal{A}) \leq\ & \frac{2(q+p)}{2^{\min\{t-\epsilon-\lambda,u\}} - (\mu+q+p)} \\ & + \frac{2\mathsf{mlf}^{2\mu}_{u,n-u}p}{2^{n-\max\{t,u+\lambda\}} - (\mu+q+p)} + \frac{\mathsf{mlf}^{2\mu}_{u,n-u}}{2^{n-u}}\,.\end{aligned}$$

5 Value Comparison Based on Tweakable Permutation

Let $\mathsf{TP} \in \mathsf{perm}(r,n)$ be a cryptographic family of permutations (for now, we will not yet limit ourselves to families of secret or public permutations). Assume that $s \leq r$ and $t, u \leq n$. Define the following tweakable permutation-based value processing function $\mathsf{TPVP}^{\mathsf{TP}} : \{0,1\}^s \times \{0,1\}^t \to \{0,1\}^u$:

$$\mathsf{TPVP}^{\mathsf{TP}}(S,T) = \text{left}_u(\mathsf{TP}(S \parallel 0^{r-s}, T \parallel 0^{n-t}))\,. \tag{16}$$

Tag verification then follows as in (4), using above value processing function TPVP (see also Figure 2):

$$\mathsf{TPVC}(\mathsf{TPVP}^{\mathsf{TP}}(S,T), \mathsf{TPVP}^{\mathsf{TP}}(S,T^\star)) = \left[\!\left[\mathsf{TPVP}^{\mathsf{TP}}(S,T) \stackrel{?}{=} \mathsf{TPVP}^{\mathsf{TP}}(S,T^\star)\right]\!\right]\,. \tag{17}$$

As before, we did not put any stringent condition on r, s, t, u, and n yet: all we need is that $s \leq r$ and $t, u \leq n$. Depending on whether TP is a family of secret or public permutations, an additional condition is needed. A general security bound of value comparison using TPVP is given in Section 5.1. We elaborate on the case of families of secret permutations in Section 5.2, and on the case of families of public permutations in Section 5.3.

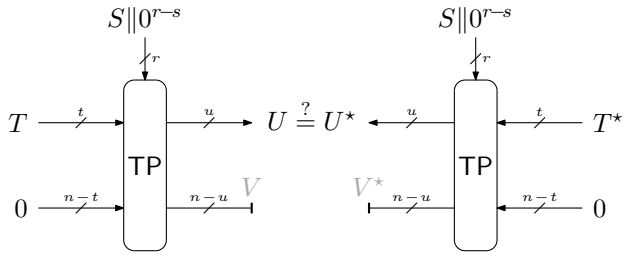

Fig. 2: Leakage resilient value comparison using a tweakable permutation.

5.1 Leakage Resilience of Value Comparison With TPVP

We derive a general bound on the leakage resilience of value comparison using TPVP,

$$\mathcal{O}_{\boldsymbol{S},\boldsymbol{T}}^{\mathsf{TPVP},\mathsf{TP}} : (j, T^\star) \mapsto \left[\!\left[\mathsf{TPVP}^{\mathsf{TP}}(S_j, T_j) \stackrel{?}{=} \mathsf{TPVP}^{\mathsf{TP}}(S_j, T^\star)\right]\!\right] , \tag{18}$$

in the security definition of (6) against any adversary making q construction queries and p primitive queries. We note that the bound is meaningless for certain choices of n, r, s, t, u, q, p: in particular, if $p > 0$ (i.e., if we consider instantiation using a *public* permutation), one requires $t, u \ll n$. The bound is nevertheless derived in full generality, and will only be interpreted for the specific cases in Sections 5.2 and 5.3.

Theorem 2. *Assume that* $\log_2(\mu) \leq s \leq r$ *and* $t, u \leq n$. *For any adversary* $\mathcal{A}$ *with construction complexity* q *and primitive complexity* p,

$$\mathbf{Adv}^{\text{lr-vc}[\mu]}_{\mathcal{O}^{\mathsf{TPVP}}}(\mathcal{A}) \leq \frac{2(q+p)}{2^{\min\{t-\epsilon-\lambda,u\}} - (\mu+q+p)} + \frac{2p}{2^{n-\max\{t,u+\lambda\}} - (\mu+q+p)} .$$

The proof is a direct simplification of the proof of Theorem 1. Most importantly, as the salt S_j is processed by TP as tweak input in both forward and inverse primitive queries, the adversary restricts itself to a unique choice of j (as salts are assumed not to collide) and hence there is no need to bother about multicollisions in $\{U_1, \ldots, U_\mu\}$. This means that event bad, as well as its analysis, drops out. A second change is in the analysis of the probability that an inverse primitive queries sets win_i: now we need that either "$V^\star = V_j$" or "$V^\star \neq V_j$ but $X = T^\star\|0^{n-t}$". The resulting bound is identical to the one before, with the term $\mathsf{mlf}^{2\mu}_{u,n-u}$ removed. A formal proof is included in the full version of the paper.

5.2 TPVP with Secret Tweakable Permutation

Let $\mathsf{TE} : \{0,1\}^k \times \{0,1\}^r \times \{0,1\}^n \to \{0,1\}^n$ be a tweakable block cipher. If TE is TPRP-secure (see Section 2.2), one can instantiate the *secret* tweakable

permutation TP in the value processing function $\mathsf{TPVP}^{\mathsf{TP}}$ using the block cipher with a secret key, and de facto consider

$$\mathsf{TEVP}^{\mathsf{TE}_K}(S,T) = \mathrm{left}_u(\mathsf{TE}_K(S \parallel 0^{r-s}, T \parallel 0^{n-t}))\,. \tag{19}$$

A variant of this using tweakable block ciphers is, in fact, proposed in NIST Lightweight Cryptography candidate Spook [5].

Identical to the analysis in Section 4.2, the security bound of Theorem 2 carries over to TEVP, with the following three changes:

- The term $\mathbf{Adv}^{\mathrm{tprp}}_{\mathsf{TE}}(q,\tau)$ is added (where q is exactly the number of queries described in Theorem 2 and τ is an additional time complexity measure on $\mathcal{A}$);
- The function TE_K must be strongly protected, so that the function leaks no information about its inputs and outputs;
- The number of primitive queries is bounded to $p = 0$.

More formally, we obtain the following corollary, in analogy with Corollary 1.

Corollary 3 (Value Comparison Using Tweakable Block Cipher). *Assume that $\log_2(\mu) \le s \le r$ and $t, u \le n$. Let $\mathsf{E} : \{0,1\}^k \times \{0,1\}^r \times \{0,1\}^n \to \{0,1\}^n$ be a tweakable block cipher that is perfectly protected. For any adversary $\mathcal{A}$ with construction complexity q and operating in time τ,*

$$\mathbf{Adv}^{\text{lr-vc}[\mu]}_{\mathcal{O}^{\mathsf{TEVP}}}(\mathcal{A}) \le \frac{2q}{2^{\min\{t-\epsilon-\lambda,u\}} - (\mu+q)} + \mathbf{Adv}^{\mathrm{tprp}}_{\mathsf{TE}}(q,\tau)\,.$$

5.3 TPVP with Public Tweakable Permutation

If one takes a block cipher $\mathsf{E} : \{0,1\}^r \times \{0,1\}^n \to \{0,1\}^n$ (see Section 4.2 for the definition) that does not only satisfy that its PRP-security is strong, but that does not even have any inherent weaknesses and that can be modeled as an ideal cipher, one can use this block cipher *as* tweakable permutation in the TPVP construction. Just like in Section 4.3, the adversary can evaluate the public primitive, or in terms of Theorem 2: $p > 0$. This also means that, for the last term of this theorem to be small, we require $t, u \ll n$. We obtain the following corollary:

Corollary 4 (Value Comparison Using Tweakable Permutation). *Assume that $\log_2(\mu) \le s \le r$ and $t, u \ll n$. Let $\mathsf{TP} \in \mathsf{perm}(r,n)$ be a family of permutations that is assumed to be perfectly random. For any adversary $\mathcal{A}$ with construction complexity q and primitive complexity p,*

$$\mathbf{Adv}^{\text{lr-vc}[\mu]}_{\mathcal{O}^{\mathsf{TPVP}}}(\mathcal{A}) \le \frac{2(q+p)}{2^{\min\{t-\epsilon-\lambda,u\}} - (\mu+q+p)} + \frac{2p}{2^{n-\max\{t,u+\lambda\}} - (\mu+q+p)}\,.$$

6 Freedom of Salts

In the security model of Section 3.3, the salts $\boldsymbol{S} \in (\{0,1\}^s)^\mu$ are unique and paired to the values in $\boldsymbol{T} \xleftarrow{\$} (\{0,1\}^t)^\mu$. This might require state and/or another technique to obtain these salts. Nevertheless, it appears that this condition can be released at almost no efficiency or security cost. In this section, we consider various cases and inspect how the bounds of Theorem 1 and 2 deteriorate. First, in Section 6.1 we consider the case of randomly generated salts. Then, in Section 6.2, we discuss how the bounds change if the salts are omitted. Finally, we briefly elaborate on the theoretical benefit of *not* disclosing salts to the adversary in Section 6.3.

6.1 Random Salts

One can simply take uniformly random $\boldsymbol{S} \xleftarrow{\$} (\{0,1\}^s)^\mu$. This will induce an additional term to the proof of Theorem 1. In detail, for the probability that the i-th query is a forward primitive query and sets win_i, we rely on the uniqueness of the values S_j. (In fact, closer inspection shows that it suffices to rely on uniqueness of the values $S_j \| T_j$, but the distribution of the T_j's might be a bit odd and might not fit the modeling of multicollisions as per Section 2.1.) This means that we need to expand the bad event bad to cover multicollisions in $\boldsymbol{S} \xleftarrow{\$} (\{0,1\}^s)^\mu$, leading to an additional term $\mathsf{mlf}^\mu_{s,t}/2^t$. Subsequently the multiplication of p in the numerator of the first term of the bound of Theorem 1 by $\mathsf{mlf}^\mu_{s,t}$. Here, when defining the multicollision event, we had some freedom to choose the value of the denominator, which we set to t to match the denominator in the first term of the bound. In total, the complete bound becomes:

$$\frac{2(q+\mathsf{mlf}^\mu_{s,t}p)}{2^{\min\{t-\epsilon-\lambda,u\}}-(\mu+q+p)} + \frac{2\mathsf{mlf}^{2\mu}_{u,n-u}p}{2^{n-\max\{t,u+\lambda\}}-(\mu+q+p)} + \frac{\mathsf{mlf}^{2\mu}_{u,n-u}}{2^{n-u}} + \frac{\mathsf{mlf}^\mu_{s,t}}{2^t}\,. \tag{20}$$

We remark that the changes are obsolete if we consider a secret primitive, in which case $p=0$, and also the two last terms of above equation disappear (see also the explanation before Corollary 1).

For the bound of TPVP, there was no bad event bad in the first place. However, we must consider multicollisions in $\{S_1\|T_1,\ldots,S_\mu\|T_\mu\}$ as well as in $\{S_1\|U_1,\ldots,S_\mu\|U_\mu\}$. As before, subtleties arise in the distribution of the T_j's as well as in the U_j's, and we will restrict our focus to multicollisions in $\boldsymbol{S} \xleftarrow{\$} (\{0,1\}^s)^\mu$. This can be bound by $\mathsf{mlf}^\mu_{s,t}/2^t$. The expanded bound becomes

$$\frac{2(q+\mathsf{mlf}^\mu_{s,t}p)}{2^{\min\{t-\epsilon-\lambda,u\}}-(\mu+q+p)} + \frac{2\mathsf{mlf}^\mu_{s,t}p}{2^{n-\max\{t,u+\lambda\}}-(\mu+q+p)} + \frac{\mathsf{mlf}^\mu_{s,t}}{2^t}\,.$$

6.2 Omission of Salt

In practice, it might not be that straightforward to pair salts with tags. However, an option that is always available is to just use the same salt for every tag.

Compared to the random selection of salts in Section 6.1, we do not have a strong bound on the largest multicollision on $\boldsymbol{S}$. Instead, in the worst case we have a single μ-collision. Hence, in contrast to Section 6.1, we do not need to introduce an additional term $\mathsf{mlf}^{\mu}_{s,t}/2^t$, since we cannot have more than a single μ-collision on μ-values. Akin to (20), the complete bound for the PVP scenario becomes:

$$\frac{2(q+\mu p)}{2^{\min\{t-\epsilon-\lambda,u\}}-(\mu+q+p)} + \frac{2\mathsf{mlf}^{2\mu}_{u,n-u}p}{2^{n-\max\{t,u+\lambda\}}-(\mu+q+p)} + \frac{\mathsf{mlf}^{2\mu}_{u,n-u}}{2^{n-u}} .$$

Since using a tweakable permutation with a single tweak/salt gives a single permutation scenario we omit to spell out the TPVP case.

Furthermore, we note that the term $\frac{2\mu p}{2^{\min\{t-\lambda,u\}}-(\mu+q+p)}$ that introduces a birthday-like trade-off in the bound between number of tags μ and primitive calls p stems from the ability of a side-channel adversary to recover all μ possible U_j's. In absence of a side-channel adversary, the bound in the black-box model omits this term. In particular, for fixed S and no leakage, we would allow the adversary access to a oracle similar to (9):

$$\mathcal{O}^{\mathsf{PVP},\mathsf{P}}_{\boldsymbol{T}} : (j, T^\star) \mapsto \left[\!\left[\mathsf{PVP}^{\mathsf{P}}(T_j) \stackrel{?}{=} \mathsf{PVP}^{\mathsf{P}}(T^\star)\right]\!\right] ,$$

and the adversary wins if $\mathcal{O}^{\mathsf{PVP},\mathsf{P}}_{\boldsymbol{T}}$ ever outputs 1:

$$\mathbf{Adv}^{\mathrm{pvc}[\mu]}_{\mathcal{O}^{\mathsf{PVP},\mathsf{P}}_{\boldsymbol{T}}}(\mathcal{A}) = \mathbf{Pr}\left(\mathsf{P} \xleftarrow{\$} \mathsf{perm}(n)\,,\, \boldsymbol{T} \xleftarrow{\$} (\{0,1\}^t)^\mu \;:\; \mathcal{A}^{\mathcal{O}^{\mathsf{PVP},\mathsf{P}}_{\boldsymbol{T}},\mathsf{P}^{\pm}} \text{ wins}\right) .$$

For completeness, we can define by $\mathbf{Adv}^{\mathrm{pvc}[\mu]}_{\mathcal{O}^{\mathsf{PVP},\mathsf{P}}_{\boldsymbol{T}}}(q,p)$ the maximum advantage over any adversary making q queries to $\mathcal{O}^{\mathsf{PVP},\mathsf{P}}_{\boldsymbol{T}}$ and p bi-directional queries to $\mathsf{P}^{\pm}$.

Proposition 1 (Saltless Value Comparison using Permutation in the Black-Box Model). *Assume that $t, u \ll n$. Let $\mathsf{P} \in \mathsf{perm}(n)$ be a permutation that is assumed to be perfectly random. For any adversary $\mathcal{A}$ with construction complexity q and primitive complexity p,*

$$\mathbf{Adv}^{\mathrm{pvc}[\mu]}_{\mathcal{O}^{\mathsf{PVP},\mathsf{P}}_{\boldsymbol{T}}}(\mathcal{A}) \leq \frac{2q}{2^{\min\{t,u\}}-q} .$$

Proof. Since we work in the black-box model, the only thing an adversary learns from a failed verification query is that $T_j \neq T$. What an adversary learns from a successful verification query does not matter, since the adversary has won anyway. As a consequence, an adversary cannot detect matches of forward primitive queries $(*^t \| 0^{n-t}, U^\star \| *^{n-u})$ with $U^\star = U_j$ only if it already won. The same counts for inverse primitive queries, hence the adversary does not profit from calls to P.

The possibilities for an adversary to win on a single query to the construction is to either guess the tag T_j correctly, or to be lucky that an incorrect guess still

leads to the same U. Summing over q construction queries and considering that all U_j's are computed via a perfectly random permutation, we hence get:

$$\mathbf{Adv}^{\mathrm{pvc}[\mu]}_{\mathcal{O}^{\mathsf{PVP},\mathsf{P}}_{T}}(\mathcal{A}) \leq \frac{2q}{2^{\min\{t,u\}} - q} \,.$$

The reasoning holds for any adversary making q construction queries and p primitive queries, and this completes the proof. □

6.3 Note on Disclosing Salts

We remark that the security model of Section 3.3 prescribes that $\mathcal{A}$ actually obtains the salts. In practice, it might often be more practical to not disclose them. This will, clearly, only *improve* security.

7 Application to Message Authentication

Our leakage resilient solutions have many applications. We already mentioned some in Section 1. In this section, we will consider the application of our solutions to message authentication. In Section 7.2, we consider a composition of SuKS with PVP, dubbed StP. The composition is very powerful against leakage resilience, even though it requires that the building blocks (SuKS and PVP) are built from independent cryptographic permutations. The result has immediate application to the ISAP authenticated encryption scheme [19, 20], that is currently in submission to the NIST Lightweight Cryptography competition. This function uses SuKS for message authentication.

In Section 7.3, we go one step further, and stretch the analysis to a MAC construction whose cryptographic primitive is related to that in value verification. In detail, we present HaFuFu, a hash-then-PRF message authentication code that uses the same PRF for value comparison, and prove that this construction is a leakage resilient MAC function. The result can be relevant for many other submissions to the NIST Lightweight Cryptography competition [48], given the prevalence of the hash-then-PRF construction.

Both results are derived in a model for leakage resilient message authentication plus value verification, that is described in Section 7.1. It is a slight extension of the model of Section 3.3.

7.1 Security Model for Leakage Resilient MAC Plus Value Comparison

We will describe the security model for leakage resilient message authentication with integrated value comparison in generality, so as it is applicable to both StP and HaFuFu.

Let $\Pi \in \mathsf{prims}$ be a cryptographic primitive or a set of cryptographic primitives, taken from a set of primitives prims from which uniform sampling is possible. A message authentication code $\mathsf{MAC}^{\Pi} : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^t$ takes as

input a key K and an arbitrarily-long message M, and uses the cryptographic primitive Π to generate a tag T. Associated to MAC^{Π} is a verification function $\mathsf{VFY}^{\Pi} : \{0,1\}^k \times \{0,1\}^* \times \{0,1\}^t \to \{0,1\}$ that gets as input a key K, a message M, and a tag $T^\star$, and it outputs 1 if the tag belongs to the message and 0 otherwise. Whereas typical verification function do plain value comparison of $\mathsf{MAC}^{\Pi}_K(M)$ with $T^\star$, in our case verification will include leakage resilient value comparison. Before proceeding, we remark that the key input to MAC^{Π} may be optional: sometimes, Π is a secretly keyed primitive (like a secret permutation) and the key would be implicit.

As before, we consider non-adaptive $\mathcal{L}$-resilience [24], where the adversary receives leakage under any leakage $\mathsf{L} \in \mathcal{L}$ of the scheme under investigation. Any cryptographic evaluation of secret material may leak information, and a proper definition of $\mathcal{L}$ depends on the scheme and primitive under consideration. For StP and HaFuFu, the set will thus be formalized as soon as we go on to prove leakage resilience (in Sections 7.2.3 and 7.3.2, respectively). For any leakage function $\mathsf{L} \in \mathcal{L}$, define by $\left[\mathsf{MAC}^{\Pi}_K\right]_{\mathsf{L}}$ an evaluation of MAC^{Π}_K of (25) that not only returns the response of this function, but also leaks secret material in consistency with the evaluation of L (details for the two specific schemes will follow in the corresponding sections). The function $\left[\mathsf{VFY}^{\Pi}_K\right]_{\mathsf{L}}$ is defined analogously.

Leakage resilience of the MAC function now extends from the conventional definition of unforgeability, but now with the adversary $\mathcal{A}$ having access to the leaky oracles. In detail, let $\mathsf{L} \in \mathcal{L}$ be any tuple of leakage functions. Consider an adversary $\mathcal{A}$ that has query access to $\left[\mathsf{MAC}^{\Pi}_K\right]_{\mathsf{L}}$ and $\left[\mathsf{VFY}^{\Pi}_K\right]_{\mathsf{L}}$. It wins if $\left[\mathsf{VFY}^{\Pi}_K\right]_{\mathsf{L}}$ ever outputs 1 on input of a message/tag tuple that was not the result of an earlier query to $\left[\mathsf{MAC}^{\Pi}_K\right]_{\mathsf{L}}$:

$$\mathbf{Adv}^{\text{lr-mac}}_{\mathsf{MAC}}(\mathcal{A}) = \max_{\mathsf{L}\in\mathcal{L}} \mathbf{Pr}\left(K \xleftarrow{\$} \{0,1\}^k, \Pi \xleftarrow{\$} \mathsf{prims} \ : \ \mathcal{A}^{\left[\mathsf{MAC}^{\Pi}_K\right]_{\mathsf{L}},\left[\mathsf{VFY}^{\Pi}_K\right]_{\mathsf{L}}} \text{ wins}\right). \tag{21}$$

For completeness, we can define by $\mathbf{Adv}^{\text{lr-mac}}_{\mathsf{MAC}}(q,v)$ the maximum advantage over any adversary making q authentication queries to $\left[\mathsf{MAC}^{\Pi}_K\right]_{\mathsf{L}}$ and v verification queries to $\left[\mathsf{VFY}^{\Pi}_K\right]_{\mathsf{L}}$.

7.2 StP: SuKS-then-PVP

7.2.1 Description of SuKS. Assume that $c + r = n$ and $k, s, t \leq n$. Let $\mathsf{P} \in \mathsf{perm}(n)$ be a cryptographic permutation and $\mathsf{G} : \{0,1\}^k \times \{0,1\}^s \to \{0,1\}^s$ be a keyed function. The suffix keyed sponge $\mathsf{SuKS} : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^t$, formalized by Dobraunig and Mennink [23], is depicted in Figure 3.

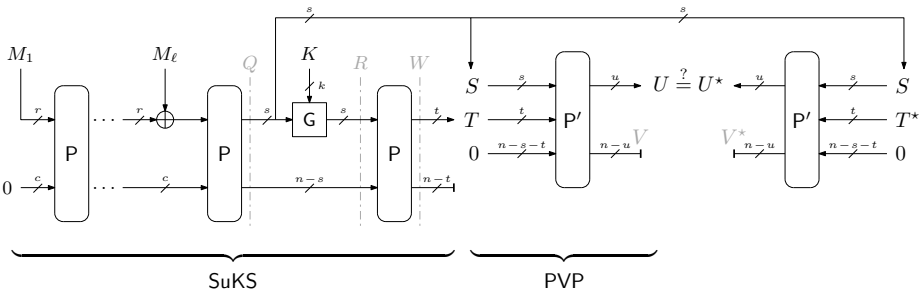

Fig. 3: The SuKS-then-PVP construction StP. The message M is first injectively padded into r-bit blocks $M_1 \dots M_\ell$.

Dobraunig and Mennink [23] proved that if P is a random permutation, G has good uniformity and universality,[4] then SuKS behaves like a random function. In addition, if G is strongly protected and any evaluation of P only leaks λ bits of data non-adaptively, SuKS still behaves like a random function.

The security model under consideration is PRF-security under non-adaptive leakage (as in Section 3.3). Let $\mathcal{L}_\mathsf{P} = \{\mathsf{L}_\mathsf{P} \colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^\lambda\}$ be a fixed set of leakage functions on the primitive P, and let $\mathcal{L}_\mathsf{G} = \{\mathsf{L}_\mathsf{G} \colon \{0,1\}^k \times \{0,1\}^s \times \{0,1\}^s \to \{0,1\}^{\lambda'}\}$ be a fixed set of leakage functions on the function G. All functions are independent of P, i.e., they do not internally evaluate P or P^{-1}. Write $\mathcal{L} = \mathcal{L}_\mathsf{P} \times \mathcal{L}_\mathsf{G}$. For any leakage function $\mathsf{L} = (\mathsf{L}_\mathsf{P}, \mathsf{L}_\mathsf{G}) \in \mathcal{L}$, define by $\left[\mathsf{SuKS}^\mathsf{P}_K\right]_\mathsf{L}$ an evaluation of $\mathsf{SuKS}^\mathsf{P}_K$ of Figure 3 that not only returns the response of this function, but also leaks the values $\mathsf{L}_\mathsf{G}(K, \mathrm{left}_s(Q), \mathrm{left}_s(R))$ and $\mathsf{L}_\mathsf{P}(R, W)$ (see Figure 3 for the values Q, R, and W). Then, non-adaptive leakage resilient pseudorandom function (LR-PRF) security is defined as the maximum advantage of any distinguisher to distinguish the following two worlds:

$$\mathbf{Adv}^{\text{lr-prf}}_{\mathsf{SuKS}}(\mathcal{A}) = \Big|\mathbf{Pr}\left(K \xleftarrow{\$} \{0,1\}^k, \mathsf{P} \xleftarrow{\$} \mathsf{perm}(n) \;:\; \mathcal{A}^{\left[\mathsf{SuKS}^\mathsf{P}_K\right]_\mathsf{L}, \mathsf{SuKS}^\mathsf{P}_K, \mathsf{P}} = 1\right) - \mathbf{Pr}\left(K \xleftarrow{\$} \{0,1\}^k, \mathsf{P} \xleftarrow{\$} \mathsf{perm}(n), \mathsf{F} \xleftarrow{\$} \mathsf{func}(*,t) \;:\; \mathcal{A}^{\left[\mathsf{SuKS}^\mathsf{P}_K\right]_\mathsf{L}, \mathsf{F}, \mathsf{P}} = 1\right)\Big| .$$

Under this model, Dobraunig and Mennink proved the following result.

Proposition 2 (Leakage Resilience of SuKS [23, Theorem 3]). *Assume that $c + r = n$ and $k, s, t \le n$. Consider the SuKS construction of Figure 3 based on random permutation $\mathsf{P} \xleftarrow{\$} \mathsf{perm}(n)$ and a function $\mathsf{G} : \{0,1\}^k \times \{0,1\}^s \to \{0,1\}^s$. Assume that G is strongly protected $2^{-\delta}$-uniform and $2^{-\varepsilon}$-universal. For any adversary $\mathcal{A}$ with construction complexity $q \ge 2$ and primitive complexity*

[4] Uniformity means that the probability (over the drawing of K) that any fixed input X maps to any fixed output Y is at most $2^{-\delta}$. Universality means that the probability (over the drawing of K) that any fixed distinct inputs X, X' map to the same value is at most $2^{-\varepsilon}$.

$p \leq 2^{n-1}$,

$$\mathbf{Adv}^{\text{lr-prf}}_{\mathsf{SuKS}}(\mathcal{A}) \leq \frac{2p^2}{2^c} + \frac{\mathsf{mlf}^{2(p-q)}_{s,n-s}}{2^{n-s}} + \frac{\mathsf{mlf}^{2(p-q)}_{n-s,s} \cdot p}{2^{\min\{\delta,\varepsilon\}-\mathsf{mlf}^{2(p-q)}_{s,n-s}\lambda}} + \frac{\mathsf{mlf}^{2q}_{t,n-t} \cdot p}{2^{n-t-\lambda}}\,.$$

One term that is important in this bound is $\frac{\mathsf{mlf}^{2(p-q)}_{s,n-s}}{2^{n-s}}$. In the proof of SuKS, the authors upper bound the maximum size of a multicollision on $\text{left}_s(Q)$ by $\mathsf{mlf}^{2(p-q)}_{s,n-s}$. The fact that this bounding is already performed in the proof of SuKS itself will become useful when we consider composition of SuKS with PVP.

7.2.2 Description of StP. Let $\mathsf{P}, \mathsf{P}' \in \mathsf{perm}(n)$, and let $\mathsf{MAC}^{\mathsf{P},\mathsf{P}'} : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^t$ be the SuKS message authentication code:

$$\mathsf{MAC}^{\mathsf{P},\mathsf{P}'}_K(M) = \mathsf{SuKS}^{\mathsf{P}}(K, M) = T\,. \tag{22}$$

Verification $\mathsf{VFY}^{\mathsf{P},\mathsf{P}'} : \{0,1\}^k \times \{0,1\}^* \times \{0,1\}^t \to \{0,1\}$ now incorporates $\mathsf{PVP}^{\mathsf{P}'}$. It takes $S = \text{left}_s(Q)$ from the computation of $\mathsf{SuKS}^{\mathsf{P}}(K, M)$ (see Figure 3) as salt, and is defined as follows:

$$\begin{aligned}&\mathsf{VFY}^{\mathsf{P},\mathsf{P}'}_K(M, T^\star) = \\ &\quad \left[\!\!\left[\text{left}_u(\mathsf{P}(S \parallel \mathsf{MAC}^{\mathsf{P},\mathsf{P}'}_K(M) \parallel 0^{n-s-t})) \stackrel{?}{=} \text{left}_u(\mathsf{P}(S \parallel T^\star \parallel 0^{n-s-t}))\right]\!\!\right]\,,\end{aligned} \tag{23}$$

where $S = \text{left}_s(Q)$ is a function of M as specified in Figure 3.

7.2.3 Leakage Resilience of StP. We will prove security of StP, provided that $\mathsf{P}, \mathsf{P}' \xleftarrow{\$} \mathsf{perm}(n)$ are two random permutations.

In StP, leakage occurs on evaluations of P, G, P′, and in the value comparison. Let $\mathcal{L}_{\mathsf{P}} = \{\mathsf{L}_{\mathsf{P}} \colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^\lambda\}$ be a fixed set of leakage functions on the primitive P, and let $\mathcal{L}_{\mathsf{G}} = \{\mathsf{L}_{\mathsf{G}} \colon \{0,1\}^k \times \{0,1\}^s \times \{0,1\}^s \to \{0,1\}^{\lambda'}\}$ be a fixed set of leakage functions on the function G. Let $\mathcal{L}_{\mathsf{P}'} = \{\mathsf{L}_{\mathsf{P}'} \colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^\lambda\}$ be a fixed set of leakage functions on the value processing function P′, and let $\mathcal{L}_{\mathsf{C}} = \{\mathsf{L}_{\mathsf{C}} \colon \{0,1\}^u \times \{0,1\}^u \to \{0,1\}^{\lambda'}\}$ be a fixed set of leakage functions on the value comparison within VFY. All functions are independent of P and P′. Write $\mathcal{L} = \mathcal{L}_{\mathsf{P}} \times \mathcal{L}_{\mathsf{G}} \times \mathcal{L}_{\mathsf{P}'} \times \mathcal{L}_{\mathsf{C}}$. For any leakage function $\mathsf{L} = (\mathsf{L}_{\mathsf{P}}, \mathsf{L}_{\mathsf{G}}, \mathsf{L}_{\mathsf{P}'}, \mathsf{L}_{\mathsf{C}}) \in \mathcal{L}$, define by $\left[\mathsf{MAC}^{\mathsf{P},\mathsf{P}'}_K\right]_{\mathsf{L}}$ an evaluation of $\mathsf{MAC}^{\mathsf{P},\mathsf{P}'}_K$ of (22) that not only returns the response of this function, but also leaks the following values:

$$\begin{aligned}\mathsf{L}_{\mathsf{G}}(K, \text{left}_s(Q), \text{left}_s(R)) &\in \{0,1\}^\lambda\,,\\ \mathsf{L}_{\mathsf{P}}(R, W) &\in \{0,1\}^\lambda\,,\end{aligned}$$

where K, Q, R, and W are values related to the computation of $\mathsf{MAC}^{\mathsf{P},\mathsf{P}'}_K(K, M)$, as outlined in Figure 3. Similarly, define by $\left[\mathsf{VFY}^{\mathsf{P},\mathsf{P}'}_K\right]_{\mathsf{L}}$ an evaluation of $\mathsf{VFY}^{\mathsf{P},\mathsf{P}'}_K$

of (23) that not only returns the response of this function, but also leaks the following values:

$$\begin{aligned}
\mathsf{L}_{\mathsf{G}}(K, \mathrm{left}_s(Q), \mathrm{left}_s(R)) &\in \{0,1\}^{\lambda}\,, \\
\mathsf{L}_{\mathsf{P}}(R, W) &\in \{0,1\}^{\lambda}\,, \\
\mathsf{L}_{\mathsf{P}'}\left(S\|T\|0^{n-s-t}, U\|V\right) &\in \{0,1\}^{\lambda}\,, \\
\mathsf{L}_{\mathsf{P}'}\left(S\|T^{\star}\|0^{n-s-t}, U^{\star}\|V^{\star}\right) &\in \{0,1\}^{\lambda}\,, \\
\mathsf{L}_{\mathsf{C}}\left(U, U^{\star}\right) &\in \{0,1\}^{\lambda'}\,,
\end{aligned}$$

where $K, Q, R, W, S, T, U, U^{\star}, V$, and $V^{\star}$ are values related to the computation of $\mathsf{VFY}_K^{\mathsf{P},\mathsf{P}}(M, T^{\star})$ as outlined in Figure 3.

We can now prove leakage resilience of StP in the security model of Section 7.1.

Theorem 3. *Assume that $k, s+t, u \leq n$. Consider the StP construction based on two random permutations $\mathsf{P}, \mathsf{P}' \xleftarrow{\$} \mathsf{perm}(n)$ and a function $\mathsf{G} : \{0,1\}^k \times \{0,1\}^s \to \{0,1\}^s$. Assume that G is strongly protected $2^{-\delta}$-uniform and $2^{-\varepsilon}$-universal. For any adversary $\mathcal{A}$ with construction query q and verification complexity v, with $q + v \geq 2$, and primitive complexity $p \leq 2^{n-1}$,*

$$\begin{aligned}
\mathbf{Adv}^{\text{lr-mac}}_{\mathsf{StP}}(\mathcal{A}) \leq\ & \frac{2p^2}{2^c} + \frac{\mathsf{mlf}^{2(p-q)}_{s,n-s}}{2^{n-s}} + \frac{\mathsf{mlf}^{2(p-q)}_{n-s,s} \cdot p}{2^{\min\{\delta,\varepsilon\} - \mathsf{mlf}^{2(p-q)}_{s,n-s}\lambda}} + \frac{\mathsf{mlf}^{2(q+v)}_{t,n-t} \cdot p}{2^{n-t-\lambda}} \\
& + \frac{2(v + \mathsf{mlf}^{2(p-q)}_{s,n-s} p)}{2^{\min\{t-2\lambda,u\}} - (2v+p)} + \frac{2\mathsf{mlf}^{2v}_{u,n-u} p}{2^{n-\max\{t,u+\lambda\}} - (2v+p)} + \frac{\mathsf{mlf}^{2v}_{u,n-u}}{2^{n-u}}\,.
\end{aligned}$$

Proof. Let $\mathsf{L} = (\mathsf{L}_{\mathsf{P}}, \mathsf{L}_{\mathsf{G}}, \mathsf{L}_{\mathsf{P}'}, \mathsf{L}_{\mathsf{C}}) \in \mathcal{L}$ be any four leakage functions, let $K \xleftarrow{\$} \{0,1\}^k$ and $\mathsf{P}, \mathsf{P}' \xleftarrow{\$} \mathsf{perm}(n)$. Consider any adversary $\mathcal{A}$ that aims to mount a forgery against $\mathsf{StP}_K^{\mathsf{P},\mathsf{P}'}$. It can make q construction queries, v verification queries, and p primitive queries to both P and P'.

It is important to note that the functions $\mathsf{SuKS}_K^{\mathsf{P}}$ and $\mathsf{PVP}^{\mathsf{P}'}$ are independent primitives. In addition, $\mathsf{SuKS}_K^{\mathsf{P}}$ is a pseudorandom function under leakage. Concretely, up to the bound of Proposition 2, each new evaluation of $\mathsf{SuKS}_K^{\mathsf{P}}$ outputs a T that has min-entropy at least $t - \lambda$ and is independent of earlier evaluations of the construction, and associated with this value T is a value S that is not secret but that has the property that if the construction is evaluated q times, the maximum size of a multicollision is $\mathsf{mlf}^{2(p-q)}_{s,n-s}$.

In fact, within StP, $\mathsf{SuKS}_K^{\mathsf{P}}$ gets evaluated up to q times for different inputs and *at most* v additional times in new verification queries. Say that the number of unique messages under which $\mathcal{A}$ queries $\mathsf{SuKS}_K^{\mathsf{P}}$ is q'. Then, we can replace $\mathsf{SuKS}_K^{\mathsf{P}}$ by generating a list of random elements $\boldsymbol{T} = (T_1, \ldots, T_{q'}) \xleftarrow{\epsilon} (\{0,1\}^t)^{q'}$ with $\epsilon = \lambda$, and an arbitrary randomly generated list $\boldsymbol{S} = (S_1, \ldots, S_{q'})$ of which each element occurs at most $\mathsf{mlf}^{2(p-q')}_{s,n-s} \leq \mathsf{mlf}^{2(p-q)}_{s,n-s}$. This replacement comes at

the cost of

$$\frac{2p^2}{2^c} + \frac{\mathsf{mlf}_{s,n-s}^{2(p-q')}}{2^{n-s}} + \frac{\mathsf{mlf}_{n-s,s}^{2(p-q')} \cdot p}{2^{\min\{\delta,\varepsilon\}-\mathsf{mlf}_{s,n-s}^{2(p-q')}\lambda}} + \frac{\mathsf{mlf}_{t,n-t}^{2q'} \cdot p}{2^{n-t-\lambda}}$$
$$\leq \frac{2p^2}{2^c} + \frac{\mathsf{mlf}_{s,n-s}^{2(p-q)}}{2^{n-s}} + \frac{\mathsf{mlf}_{n-s,s}^{2(p-q)} \cdot p}{2^{\min\{\delta,\varepsilon\}-\mathsf{mlf}_{s,n-s}^{2(p-q)}\lambda}} + \frac{\mathsf{mlf}_{t,n-t}^{2(q+v)} \cdot p}{2^{n-t-\lambda}} \,. \tag{24}$$

Having made this replacement, one can then see that, as evaluations of $\mathsf{SuKS}_K^{\mathsf{P}}$ are independent for different messages, only the elements in $\boldsymbol{T}$ and $\boldsymbol{S}$ that are considered in the evaluation of $\mathsf{PVP}^{\mathsf{P}}$ are useful. Therefore, the game of mounting a forgery against the resulting construction is equivalent to the game of mounting an attack against the value comparison function $\mathsf{PVP}^{\mathsf{P}'}$ in the model of Section 3.3, where $\mu = v$.

In summary, we have obtained that

$$\mathbf{Adv}_{\mathsf{StP}}^{\text{lr-mac}}(\mathcal{A}) \leq (24) + \mathbf{Adv}_{\mathcal{O}^{\mathsf{PVP}}}^{\text{lr-vc}[v]}(\mathcal{A}) \,,$$

for some adversary $\mathcal{A}'$ with construction complexity v and primitive complexity p, that operates in the game with salts that may repeat up to $\mathsf{mlf}_{s,n-s}^{2(p-q)}$ times. We can take the bound of Theorem 1 with the p in the numerator of the first term multiplied by $\mathsf{mlf}_{s,n-s}^{2(p-q)}$ (or, alternatively, take (20) with $\mathsf{mlf}_{s,t}^{v}$ replaced by $\mathsf{mlf}_{s,n-s}^{2(p-q)}$ and with the last term dropped as it is already accounted for in the bound of $\mathsf{SuKS}_K^{\mathsf{P}}$), for $\mu = v$, $\epsilon = \lambda$, and $q = v$. □

7.3 HaFuFu: MAC Plus Value Comparison With Same Primitive

7.3.1 Description of HaFuFu. We will describe the HaFuFu message authentication with dependent value comparison. Given the non-triviality of the problem, we consider a simpler scenario compared to the results of Section 4 and 5, namely one based on a random function (cf., Remark 2). In addition, for simplicity we assume that $s + t = n$ (the analysis easily extends to the case of $s + t \leq n$) and $t = u$. Let $\mathsf{H} \in \mathsf{func}(*, n)$ be a cryptographic hash function, and $\mathsf{F} \in \mathsf{func}(n, t)$ a (secret) cryptographic function. As F is a secret primitive, there is no key involved. Define the following message authentication code $\mathsf{MAC}^{\mathsf{H},\mathsf{F}} : \{0,1\}^* \to \{0,1\}^t$:

$$\mathsf{MAC}^{\mathsf{H},\mathsf{F}}(M) = \mathsf{F}(\mathsf{H}(M)) = T \,. \tag{25}$$

The corresponding verification function $\mathsf{VFY}^{\mathsf{H},\mathsf{F}} : \{0,1\}^* \times \{0,1\}^t \to \{0,1\}$ is defined as follows:

$$\mathsf{VFY}^{\mathsf{H},\mathsf{F}}(M, T^\star) = \left[\!\!\left[\mathsf{F}(\mathrm{left}_s(\mathsf{H}(M)) \| \mathsf{MAC}^{\mathsf{H},\mathsf{F}}(M)) \stackrel{?}{=} \mathsf{F}(\mathrm{left}_s(\mathsf{H}(M)) \| T^\star) \right]\!\!\right] \,, \tag{26}$$

The function is depicted in Figure 4. The picture also includes definitions of intermediate values R, S, T, U, and $U^\star$, that we will use when analyzing MAC and VFY. Note that the name HaFuFu is derived from the verification oracle, that operates in a Hash-then-Function-then-Function mode.

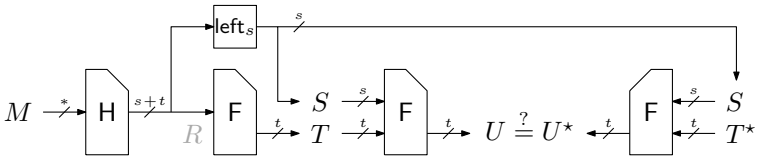

Fig. 4: HaFuFu algorithms MAC and VFY of (25) and (26), respectively. H is a cryptographic hash function and F a secret random permutation.

7.3.2 Leakage Resilience of HaFuFu. We will prove security of HaFuFu, provided that $\mathsf{H} \xleftarrow{\$} \mathsf{func}(*, n)$ is a random oracle, and $\mathsf{F} \xleftarrow{\$} \mathsf{func}(n, t)$ a secret random function. In practice, one might consider instantiating H with any good cryptographic hash function, and F by a strongly protected PRF, which can in turn be built from a (tweakable) block cipher with n-bit block size, followed by truncation [4, 11, 34, 45, 53].

In HaFuFu, leakage occurs on evaluations of F and in the value comparison. Let $\mathcal{L}_\mathsf{F} = \{\mathsf{L}_\mathsf{F} \colon \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^\lambda\}$ be a fixed set of leakage functions on the value processing function F, and let $\mathcal{L}_\mathsf{C} = \{\mathsf{L}_\mathsf{C} \colon \{0,1\}^t \times \{0,1\}^t \to \{0,1\}^{\lambda'}\}$ be a fixed set of leakage functions on the value comparison within VFY. All functions are independent of F itself, i.e., they do not internally evaluate F. Write $\mathcal{L} = \mathcal{L}_\mathsf{F} \times \mathcal{L}_\mathsf{C}$. For any leakage function $\mathsf{L} = (\mathsf{L}_\mathsf{F}, \mathsf{L}_\mathsf{C}) \in \mathcal{L}$, define by $\left[\mathsf{MAC}^{\mathsf{H},\mathsf{F}}\right]_\mathsf{L}$ an evaluation of $\mathsf{MAC}^{\mathsf{H},\mathsf{F}}$ of (25) that not only returns the response of this function, but also leaks the following value:

$$\mathsf{L}_\mathsf{F}\,(R, T) \in \{0,1\}^\lambda\,,$$

where R and T are values related to the computation of $\mathsf{MAC}^{\mathsf{H},\mathsf{F}}(M)$, as outlined in Figure 4. Similarly, define by $\left[\mathsf{VFY}^{\mathsf{H},\mathsf{F}}\right]_\mathsf{L}$ an evaluation of $\mathsf{VFY}^{\mathsf{H},\mathsf{F}}$ of (26) that not only returns the response of this function, but also leaks the following values:

$$\begin{aligned}
\mathsf{L}_\mathsf{F}\,(R, T) &\in \{0,1\}^\lambda\,,\\
\mathsf{L}_\mathsf{F}\,(S\|T, U) &\in \{0,1\}^\lambda\,,\\
\mathsf{L}_\mathsf{F}\,(S\|T^\star, U^\star) &\in \{0,1\}^\lambda\,,\\
\mathsf{L}_\mathsf{C}\,(U, U^\star) &\in \{0,1\}^{\lambda'}\,,
\end{aligned}$$

where R, S, T, U, and $U^\star$ are values related to the computation of $\mathsf{VFY}^{\mathsf{H},\mathsf{F}}(M, T^\star)$ as outlined in Figure 4.

We can now prove leakage resilience of HaFuFu in the security model of Section 7.1.

Theorem 4. *Assume that $s + t = n$. Consider the* HaFuFu *construction based on a random oracle $\mathsf{H} \xleftarrow{\$} \mathsf{func}(*, n)$ and a secret random function $\mathsf{F} \xleftarrow{\$} \mathsf{func}(n, t)$. For any adversary $\mathcal{A}$ with construction query q and verification complexity v,*

$$\mathbf{Adv}^{\text{lr-mac}}_{\mathsf{HaFuFu}}(\mathcal{A}) \le \frac{2q}{2^{t-2\lambda}} + \frac{2\binom{q+v}{2}}{2^n}\,.$$

Proof. Let $\mathsf{L} = (\mathsf{L}_\mathsf{F}, \mathsf{L}_\mathsf{C}) \in \mathcal{L}$ be any two leakage functions, let $\mathsf{H} \xleftarrow{\$} \mathsf{func}(*, n)$ be a random oracle and $\mathsf{F} \xleftarrow{\$} \mathsf{func}(n, t)$ a random function. Consider any adversary $\mathcal{A}$ that aims to mount a forgery against $\mathsf{HaFuFu}^{\mathsf{H},\mathsf{F}}$. It can make q construction queries and v verification queries. For each verification query $\mathsf{VFY}^{\mathsf{H},\mathsf{F}}(M, T^\star)$, $\mathcal{A}$ learns the following values:

$$\begin{aligned} \mathsf{L}_\mathsf{F}(R, T) &\in \{0,1\}^\lambda, \\ \mathsf{L}_\mathsf{F}(S \| T, U) &\in \{0,1\}^\lambda, \\ \mathsf{L}_\mathsf{F}(S \| T^\star, U^\star) &\in \{0,1\}^\lambda, \\ \mathsf{L}_\mathsf{C}(U, U^\star) &\in \{0,1\}^{\lambda'}. \end{aligned}$$

Here, R, S, T, U, and $U^\star$ are as described in Figure 4. *Under the assumption that outputs of* H *never collide*, we can observe that these are the only functions that leak information about R, T, and U for this message M. In other words, under this assumption, leakages for different messages are independent. As L_F and L_C are fixed, predetermined, functions, they adversary learns *at most* 2λ bits of leakage on T and at most $\lambda + v\lambda'$ bits of leakage on U, for any message M.

The adversary wins if any of its q construction queries returns 1. However, as suggested above, we have to argue based on the non-existence of collisions in the output of H, labeled R. In fact, it turns out that the adversary also has a gain if there are collisions in the values $S\|U$. Therefore, we will count both types of collisions as a win for the adversary.

More detailed, we denote by bad the event that there exist two queries to $\mathsf{MAC}^{\mathsf{H},\mathsf{F}}$ and $\mathsf{VFY}^{\mathsf{H},\mathsf{F}}$ that satisfy $R = R'$ or $S\|U = S'\|U'$. For $i \in \{1, \ldots, v\}$, we denote by win_i the event that the i-th verification query succeeds. Write $\mathsf{win} = \bigvee_{i=1}^{v} \mathsf{win}_i$. Our goal is to bound

$$\begin{aligned} \mathbf{Pr}(\mathsf{win}) &\leq \mathbf{Pr}(\mathsf{win} \wedge \neg\mathsf{bad}) + \mathbf{Pr}(\mathsf{bad}) \\ &= \mathbf{Pr}\left(\bigvee_{i=1}^{v} \mathsf{win}_i \wedge \neg\mathsf{bad}\right) + \mathbf{Pr}(\mathsf{bad}) \\ &\leq \sum_{i=1}^{v} \mathbf{Pr}(\mathsf{win}_i \wedge \neg\mathsf{win}_{1..i-1} \wedge \neg\mathsf{bad}) + \mathbf{Pr}(\mathsf{bad}), \end{aligned} \tag{27}$$

where $\mathsf{win}_{1..0} = \mathsf{false}$ by definition.

Bound on $\mathbf{Pr}(\mathsf{win}_i \wedge \neg\mathsf{win}_{1..i-1} \wedge \neg\mathsf{bad})$. Consider any $i \in \{1, \ldots, v\}$, and consider the i-th query $(M, T^\star)$. By $\neg\mathsf{bad}$, message M defines a unique R, so the construction query is independent of all other construction queries that were not made for the message M. The oracle outputs 1 if:

- $T^\star = T$. As the adversary has so far learned at most 2λ bits of leakage on T, this condition is set with probability at most $1/2^{t-2\lambda}$;
- $T^\star \neq T$ but $\mathsf{F}(S\|T^\star) = U$. Clearly, if there were an earlier message M' for which $S' = S$ and $T' = T^\star$, the equation $\mathsf{F}(S\|T^\star) = U$ would contradict

with the assumption that there is no collision $S\|U = S'\|U'$. Therefore, necessarily, there was no earlier evaluation of $\mathsf{F}(S\|T^\star)$, and the result will be randomly drawn from a set of size at least 2^t values. Thus, the condition is set with probability at most $1/2^t$.

Adding both cases, we get

$$\mathbf{Pr}\left(\mathsf{win}_i \wedge \neg\mathsf{win}_{1..i-1}\right) \leq \frac{2}{2^{t-2\lambda}}\,. \tag{28}$$

Bound on $\mathbf{Pr}\left(\mathsf{bad}\right)$. The hash function is invoked a total number of $q+v$ times, and any pair of invocations has colliding $R = R'$ with probability $1/2^n$ and colliding $S\|U = S'\|U'$ with probability $s+t$. As we assumed that $s+t=n$, we obtain that

$$\mathbf{Pr}\left(\mathsf{bad}\right) \leq \frac{2\binom{q+v}{2}}{2^n}\,. \tag{29}$$

Conclusion. The adversary makes q construction queries, each of which succeeds with probability at most (28). Next, we have to add (29). We thus obtain from (27) that

$$\mathbf{Adv}^{\text{lr-mac}}_{\mathsf{HaFuFu}}(\mathcal{A}) \leq \frac{2q}{2^{t-2\lambda}} + \frac{2\binom{q+v}{2}}{2^n}\,.$$

The reasoning holds for any adversary making q construction queries and v verification queries, and this completes the proof. □

8 Conclusion

In this paper, we examined leakage resilient value comparison via cryptographic building blocks. In short, we showed that is possible to perform value comparison via cryptographic building blocks in a sound and leakage resilient way without the need to protect the comparison operation at all. Hence, there is no strict need in putting resources into the additional protection of the comparison operation. Instead, implementers could choose an area/speed trade-off by just saving the area needed to implement a protected verification operation in exchange for two additional primitive executions during verification.

The probability that an adversary guesses the right value in q attempts for just a plain tag comparison in the black box setting is $q/2^t$. When comparing this with the security bounds we get for value comparison via cryptographic functions, we see that doing the comparison cryptographic functions give the adversary a slightly bigger advantage in succeeding. The main reason for this is that U and $U^\star$ can have the same value although T and $T^\star$ might differ. We consider this advantage to be negligible in most practical cases and value the benefits in resistance against side-channel attacks more. However, in case this additional advantage over a plain comparison is a concern, it is possible to lessen it by increasing the size of U and $U^\star$.

Acknowledgements. This work has been supported in part by the Austrian Science Fund (FWF): J 4277-N38, and the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 681402).

References

1. Andreeva, E., Bogdanov, A., Datta, N., Luykx, A., Mennink, B., Nandi, M., Tischhauser, E., Yasuda, K.: COLM v1. CAESAR, second choice for defense in depth (2016)
2. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY family of block ciphers and its low-latency variant MANTIS. In: CRYPTO. LNCS, vol. 9815, pp. 123–153. Springer (2016)
3. Beierle, C., Leander, G., Moradi, A., Rasoolzadeh, S.: CRAFT: lightweight tweakable block cipher with efficient protection against DFA attacks. IACR Trans. Symmetric Cryptol. 2019(1), 5–45 (2019)
4. Bellare, M., Impagliazzo, R.: A tool for obtaining tighter security analyses of pseudorandom function based constructions, with applications to prp to prf conversion. Cryptology ePrint Archive, Report 1999/024 (1999)
5. Bellizia, D., Berti, F., Bronchain, O., Cassiers, G., Duval, S., Guo, C., Leander, G., Leurent, G., Levi, I., Momin, C., Pereira, O., Peters, T., Standaert, F.X., Udvarhelyi, B., Wiemer, F.: Spook: Sponge-based leakage-resistant authenticated encryption with a masked tweakable block cipher. IACR Trans. Symmetric Cryptol. 2020, 295–349 (Jun 2020)
6. Bellizia, D., Bronchain, O., Cassiers, G., Grosso, V., Guo, C., Momin, C., Pereira, O., Peters, T., Standaert, F.X.: Mode-level vs. implementation-level physical security in symmetric cryptography - A practical guide through the leakage-resistance jungle. In: CRYPTO. LNCS, vol. 12170, pp. 369–400. Springer (2020)
7. Berti, F., Guo, C., Pereira, O., Peters, T., Standaert, F.X.: Tedt, a leakage-resist AEAD mode for high physical security applications. IACR Trans. Cryptogr. Hardw. Embed. Syst. 2020(1), 256–320 (2020)
8. Berti, F., Pereira, O., Peters, T., Standaert, F.X.: On leakage-resilient authenticated encryption with decryption leakages. IACR Trans. Symmetric Cryptol. 2017(3), 271–293 (2017)
9. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Cryptographic sponge functions (January 2011)
10. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: The Keccak reference. Submission to NIST (Round 3) (2011)
11. Bhattacharya, S., Nandi, M.: A note on the chi-square method: A tool for proving cryptographic security. Cryptogr. Commun. 10(5), 935–957 (2018)
12. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks. In: CRYPTO. LNCS, vol. 1666, pp. 398–412. Springer (1999)
13. Daemen, J.: Changing of the guards: A simple and efficient method for achieving uniformity in threshold sharing. In: CHES. LNCS, vol. 10529, pp. 137–153. Springer (2017)
14. Daemen, J., Dobraunig, C., Eichlseder, M., Groß, H., Mendel, F., Primas, R.: Protecting against statistical ineffective fault attacks. IACR Trans. Cryptogr. Hardw. Embed. Syst. 2020(3), 508–543 (2020)

15. Daemen, J., Hoffert, S., Van Assche, G., Van Keer, R.: The design of xoodoo and xoofff. IACR Trans. Symmetric Cryptol. 2018(4), 1–38 (2018)
16. Daemen, J., Mennink, B., Van Assche, G.: Full-state keyed duplex with built-in multi-user support. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT. LNCS, vol. 10625, pp. 606–637. Springer (2017)
17. Daemen, J., Peeters, M., Van Assche, G., Rijmen, V.: The NOEKEON block cipher (2000), nessie Proposal
18. Daemen, J., Rijmen, V.: The Design of Rijndael - The Advanced Encryption Standard (AES). Information Security and Cryptography, Springer (2002)
19. Dobraunig, C., Eichlseder, M., Mangard, S., Mendel, F., Mennink, B., Primas, R., Unterluggauer, T.: Isap v2.0. IACR Trans. Symmetric Cryptol. 2020(S1), 390–416 (2020)
20. Dobraunig, C., Eichlseder, M., Mangard, S., Mendel, F., Unterluggauer, T.: ISAP - towards side-channel secure authenticated encryption. IACR Trans. Symmetric Cryptol. 2017(1), 80–105 (2017)
21. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon v1.2. Submission to NIST Lightweight Cryptography (2019)
22. Dobraunig, C., Mennink, B.: Leakage resilience of the duplex construction. In: ASIACRYPT. LNCS, vol. 11923, pp. 225–255. Springer (2019)
23. Dobraunig, C., Mennink, B.: Security of the suffix keyed sponge. IACR Trans. Symmetric Cryptol. 2019(4), 223–248 (2019)
24. Dodis, Y., Pietrzak, K.: Leakage-resilient pseudorandom functions and side-channel attacks on feistel networks. In: CRYPTO. LNCS, vol. 6223, pp. 21–40. Springer (2010)
25. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: FOCS. pp. 293–302. IEEE Computer Society (2008)
26. Faust, S., Pietrzak, K., Schipper, J.: Practical leakage-resilient symmetric cryptography. In: CHES. LNCS, vol. 7428, pp. 213–232. Springer (2012)
27. Gérard, B., Grosso, V., Naya-Plasencia, M., Standaert, F.X.: Block ciphers that are easier to mask: How far can we go? In: CHES. LNCS, vol. 8086, pp. 383–399. Springer (2013)
28. Goubin, L., Patarin, J.: DES and differential power analysis (the "duplication" method). In: CHES. LNCS, vol. 1717, pp. 158–172. Springer (1999)
29. Goudarzi, D., Jean, J., Kölbl, S., Peyrin, T., Rivain, M., Sasaki, Y., Sim, S.M.: Pyjamask: Block cipher and authenticated encryption with highly efficient masked implementation. IACR Trans. Symmetric Cryptol. 2020(S1), 31–59 (Jun 2020)
30. Groß, H., Mangard, S.: Reconciling d+1 masking in hardware and software. In: CHES. LNCS, vol. 10529, pp. 115–136. Springer (2017)
31. Grosso, V., Leurent, G., Standaert, F.X., Varici, K.: Ls-designs: Bitslice encryption for efficient masked software implementations. In: FSE. LNCS, vol. 8540, pp. 18–37. Springer (2014)
32. Guo, C., Pereira, O., Peters, T., Standaert, F.X.: Towards low-energy leakage-resistant authenticated encryption from the duplex sponge construction. IACR Trans. Symmetric Cryptol. 2020(1), 6–42 (2020)
33. Guo, C., Standaert, F.X., Wang, W., Yu, Y.: Efficient side-channel secure message authentication with better bounds. IACR Trans. Symmetric Cryptol. 2019(4), 23–53 (2019)
34. Hall, C., Wagner, D.A., Kelsey, J., Schneier, B.: Building prfs from prps. In: CRYPTO. LNCS, vol. 1462, pp. 370–389. Springer (1998)
35. Ishai, Y., Sahai, A., Wagner, D.A.: Private circuits: Securing hardware against probing attacks. In: CRYPTO. LNCS, vol. 2729, pp. 463–481. Springer (2003)

36. Jean, J., Nikolić, I., Peyrin, T., Seurin, Y.: Deoxys v1.41. CAESAR, first choice for defense in depth (2016)
37. Kannwischer, M.J., Pessl, P., Primas, R.: Single-trace attacks on keccak. IACR Trans. Cryptogr. Hardw. Embed. Syst. 2020(3), 243–268 (2020)
38. Kocher, P.C.: Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In: CRYPTO. LNCS, vol. 1109, pp. 104–113. Springer (1996)
39. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: CRYPTO. LNCS, vol. 1666, pp. 388–397. Springer (1999)
40. Krovetz, T., Rogaway, P.: The software performance of authenticated-encryption modes. In: FSE. LNCS, vol. 6733, pp. 306–327. Springer (2011)
41. Krovetz, T., Rogaway, P.: The OCB authenticated-encryption algorithm. RFC 7253, 1–19 (2014)
42. McGrew, D.A., Viega, J.: The security and performance of the galois/counter mode (GCM) of operation. In: INDOCRYPT. LNCS, vol. 3348, pp. 343–355. Springer (2004)
43. Medwed, M., Standaert, F.X., Joux, A.: Towards super-exponential side-channel security with efficient leakage-resilient prfs. In: CHES. LNCS, vol. 7428, pp. 193–212. Springer (2012)
44. Medwed, M., Standaert, F.X., Nikov, V., Feldhofer, M.: Unknown-input attacks in the parallel setting: Improving the security of the CHES 2012 leakage-resilient PRF. In: ASIACRYPT. LNCS, vol. 10031, pp. 602–623 (2016)
45. Mennink, B.: Linking Stam's bounds with generalized truncation. In: CT-RSA. LNCS, vol. 11405, pp. 313–329. Springer (2019)
46. Nikova, S., Rechberger, C., Rijmen, V.: Threshold implementations against side-channel attacks and glitches. In: ICICS. LNCS, vol. 4307, pp. 529–545. Springer (2006)
47. Nikova, S., Rijmen, V., Schläffer, M.: Secure hardware implementation of nonlinear functions in the presence of glitches. J. Cryptology 24(2), 292–321 (2011)
48. NIST: Lightweight Cryptography (February 2019)
49. Pereira, O., Standaert, F.X., Vivek, S.: Leakage-resilient authentication and encryption from symmetric cryptographic primitives. In: ACM CCS. pp. 96–108. ACM (2015)
50. Pietrzak, K.: A leakage-resilient mode of operation. In: EUROCRYPT. LNCS, vol. 5479, pp. 462–482. Springer (2009)
51. Rivain, M., Prouff, E.: Provably secure higher-order masking of AES. In: CHES. LNCS, vol. 6225, pp. 413–427. Springer (2010)
52. Simon, T., Batina, L., Daemen, J., Grosso, V., Massolino, P.M.C., Papagiannopoulos, K., Regazzoni, F., Samwel, N.: Friet: An authenticated encryption scheme with built-in fault detection. In: EUROCRYPT. LNCS, vol. 12105, pp. 581–611. Springer (2020)
53. Stam, A.J.: Distance between sampling with and without replacement. Statistica Neerlandica 32(2), 81–91 (1978)
54. Standaert, F.X., Pereira, O., Yu, Y.: Leakage-resilient symmetric cryptography under empirically verifiable assumptions. In: CRYPTO. LNCS, vol. 8042, pp. 335–352. Springer (2013)
55. Trichina, E.: Combinational logic design for AES subbyte transformation on masked data. Cryptology ePrint Archive, Report 2003/236 (2003)
56. Unterstein, F., Schink, M., Schamberger, T., Tebelmann, L., Ilg, M., Heyszl, J.: Retrofitting leakage resilient authenticated encryption to microcontrollers. IACR Trans. Cryptogr. Hardw. Embed. Syst. 2020(4), 365–388 (2020)