IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2020, No. 4, pp. 195–212. DOI:10.46586/tosc.v2020.i4.195-212

Tightness of the Suffix Keyed Sponge Bound

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

[1] Graz University of Technology, Graz, Austria
christoph.dobraunig@iaik.tugraz.at
[2] Radboud University, Nijmegen, The Netherlands
b.mennink@cs.ru.nl

Abstract. Generic attacks are a vital ingredient in the evaluation of the tightness of security proofs. In this paper, we evaluate the tightness of the suffix keyed sponge (SuKS) bound. As its name suggests, SuKS is a sponge-based construction that absorbs the key after absorbing the data, but before producing an output. This absorption of the key can be done via an easy to invert operation, like an XOR, or a hard to invert operation, like a PRF. Using SuKS with a hard to invert absorption provides benefits with respect to its resistance against side-channel attacks, and such a construction is used as part of the authenticated encryption scheme Isap. We derive two key recovery attacks against SuKS with easy to invert key absorption, and a forgery in case of hard to invert key absorption. The attacks closely match the terms in the PRF security bound of SuKS by Dobraunig and Mennink, ToSC 2019(4), and therewith show that these terms are justified, even if the function used to absorb the key is a PRF, and regardless of whether SuKS is used as a PRF or a MAC.

Keywords: generic attacks · symmetric cryptography · permutation-based cryptography · SuKS

1 Introduction

The sponge construction [BDPV07] and its closely related duplex construction [BDPV11b] are very flexible and popular design strategies based on unkeyed cryptographic permutations. They allow us to provide many cryptographic functionalities like hashing, encryption, authentication, and authenticated encryption, in a wide range of use cases. Prominent examples following the sponge/duplex construction are the SHA-3 hash function family [Nat15, BDPV11d] and the first choice for lightweight application of CAESAR, called Ascon [DEMS19].

Given their practical importance, the sponge and duplex construction have faced extensive scientific research over time. Originally, Bertoni et al. [BDPV08] proved security of the unkeyed sponge construction in the indifferentiability framework [MRH04]. Such proof provides additional evidence for the structural soundness of the construction. This result appeared to be suitable to argue security of the keyed versions of the sponge [BDPV11c] and the duplex [BDPV11b], but it was quickly recognized that the bound was not strong enough and more dedicated security proofs arose that yielded stronger bounds for various refinements of the keyed sponge/duplex [CDH+12, ADMV15, NY16, JLM14, GPT15, MRV15, MRV15, DMV17, Men18]. However, it appears that with the rise of the more dedicated security proofs, the bounds became more complex, and it is not always clear if the bounds are tight, or can be improved further.

Received: 2020-09-01 Accepted: 2020-11-01 Published: 2020-12-10

1.1 Generic Attacks

A crucial counterpart of proofs of security are so-called generic attacks. The goal of generic attacks is to give the (complexity wise) best possible attack on a construction assuming ideality of the underlying primitive. Only by having generic attacks with a complexity matching the bounds of the proof, we can assure that the bounds given by the proof are tight. If this is not the case, we have an unclear situation not knowing if the gap between proof and best generic attack is an artifact of the proof, and hence, the proof can be improved, or if better generic attacks exist.

Finding and improving generic attacks has a very rich tradition in cryptography. Only in the context of MAC security, notable results are the generic attack of Preneel and van Oorschot [Pv95], generic attacks on various novel MAC constructions by Leurent et al. [LNS18], and generic attacks on hash-based MACs by Leurent et al. and Dinur et al. [LPW13, DL14]. Recently, the quest to understanding security bounds better has lead to work that re-evaluates constructions with matching attacks like counter mode encryption to provide stronger attacks like message recovery [LS18].

We focus on generic attacks for serial permutation-based constructions. For the unkeyed sponge, generic attacks were already discussed by Bertoni et al. [BDPV07]. As already mentioned, in the keyed case, for duplex-based and related constructions far more work exists giving improved bounds. In some cases, e.g., for the transform-then-permute construction [CJN20], matching generic attacks are given. Apart from that, generic attacks have been published, e.g., on the permutation-based constructions Oribatida [RS19] and Orange [DMM20], that point out flaws in the constructions.

1.2 Our Contribution

In this paper, we will focus on the suffix keyed sponge construction, SuKS. The idea of the suffix keyed sponge was proposed by Bertoni et al. [BDPV07, BDPV11a], and a security proof for a generalized construction was given by Dobraunig and Mennink [DM20b]. SuKS structurally differs from the "typical" keyed sponge/duplex in that the key is inserted after the message has been processed. Furthermore, the absorption of the key is done by mixing the outer part of the state with the key by using a function G. More specifically, this function G can be different from the XOR that is used to absorb the message.

For example, SuKS is an integral part of the authenticated encryption schemes Isap [DEM+17] and Isap v2.0 [DEM+19, DEM+20], which is participating in the second round of the NIST lightweight cryptography standardization process. In Isap, the function G itself is a sponge construction that is hard to invert.

The choice of G has huge implications for the meaning of generic attacks on the suffix keyed sponge. In detail: generic attacks on the regular sponge/duplex do not carry over in general, since G might be hard to invert. Interestingly, the bound on the security of SuKS given by Dobraunig and Mennink [DM20b] is invariant of whether G is easy or hard to invert. One reason for this is that the bound is derived for PRF security, but it does not clearly demonstrate the cost of recovering the key or making a forgery against SuKS as a MAC.

In this work, we perform a detailed investigation of the tightness of the security bound of Dobraunig and Mennink [DM20b] on SuKS working, as the proof, in the ideal permutation model. The focus is not on distinguishability in the PRF setting, but rather on mounting forgery attacks on SuKS as a MAC. In the analysis, we separate between easy to invert and hard to invert functions G, meaning that from its input and output the key K can or cannot be easily deduced. In both settings we derive generic attacks that come close to the complexities given by the dominating terms in the PRF security bound, but the attacks are rather different in nature depending on how easy it is to invert G.

In case of easy to invert key absorption, we present a key recovery attack based on

multi-collisions on the tag, as well as a key recovery attack based on multi-collisions on the inner part of the state at evaluation of G. For a typical instantiation of the construction for 128-bit security, the former attack needs around $2^{124.1}$ online queries to the construction and around $2^{125.8}$ queries to the underlying primitive, and the latter attack needs around 6 queries to the construction and around $2^{125.9}$ queries to the underlying primitive. The two attacks closely match two of the terms in the security bound of SuKS by Dobraunig and Mennink [DM20b].

In case of hard to invert key absorption, the situation is much more complicated, and a key recovery is hard to mount due to the fact that G is hard to invert. Nevertheless, we manage to mount a forgery attack on SuKS in this setting. The attack is inspired by the second attack of the case of easy to invert key absorption, although some technicalities occur and a more involved multi-collision structure must be targeted. For a typical instantiation of the construction for 128-bit security, the attack needs 5 online queries to the construction and around $2^{125.83}$ queries to the underlying primitive. As before, the attack closely matches the dominating term in the security bound of SuKS by Dobraunig and Mennink [DM20b].

1.3 Outline

We start the paper by giving some preliminaries in Section 2. A description of SuKS is given in detail in Section 3. Here, we also specify and discuss the bound of Dobraunig and Mennink. Attacks in the case of easy to invert absorption are derived in detail in Section 4; attacks in the case of hard to invert absorption are derived in Section 5. These sections also include applications of the attacks for concrete parameter choices. Finally, we conclude in Section 6.

2 Preliminaries

Let $m, n \in \mathbb{N}$. The set of n-bit strings is denoted $\{0,1\}^n$ and the set of arbitrarily long strings is denoted $\{0,1\}^*$. The set of n-bit permutations is denoted $\mathsf{perm}(n)$. For $X \in \{0,1\}^n$ and if $m \leq n$, we denote by $\mathrm{left}_m(X)$ (resp., $\mathrm{right}_m(X)$) the m leftmost (resp., rightmost) bits of X. For a finite set $\mathcal{X}$, $X \xleftarrow{\$} \mathcal{X}$ denotes the uniformly random drawing of an element X from $\mathcal{X}$.

By Stirling's approximation (see [Sti30, DM56]), $A! \geq (A/e)^A$, we know that

$$\left(\frac{A}{B}\right)^B \leq \binom{A}{B} \leq \left(\frac{Ae}{B}\right)^B . \tag{1}$$

2.1 PRF Security

Let $k, b, t \in \mathbb{N}$, and let $F^p : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^t$ be a function that internally uses a permutation $p \in \mathsf{perm}(b)$. The pseudorandom function (PRF) security of F against an adversary $\mathcal{A}$ is defined as

$$\mathbf{Adv}^{\mathrm{prf}}_F(\mathcal{A}) = \left| \mathbf{Pr}\left(1 \leftarrow \mathcal{A}^{F^p_K, p}\right) - \mathbf{Pr}\left(1 \leftarrow \mathcal{A}^{\$, p}\right) \right| , \tag{2}$$

where $p \xleftarrow{\$} \mathsf{perm}(b)$, $K \xleftarrow{\$} \{0,1\}^k$, and where $\$$ is a function that returns a uniformly random t-bit string for each new input. The adversary has bi-directional query access to p. Its complexity is usually measured in the number of queries q to the construction and in the number of primitive evaluations N. The latter accounts both for direct queries to p and its inverse, as well as the cost of construction queries as if $\mathcal{A}$ were communicating with F^p_K.

2.2 Uniformity and Universality

In contrast to the original definition of the (keyed/unkeyed) sponge [BDPV07, BDPV08] and duplex [BDPV11b] constructions, the suffix keyed sponge construction of Section 3 uses a function G to absorb the secret key. The security of the suffix keyed sponge is in part determined by the uniformity and universality of this function G.

Let $k, s \in \mathbb{N}$. Analogue to [DM20b], we consider $G : \{0,1\}^k \times \{0,1\}^s \to \{0,1\}^s$ to be $2^{-\delta}$-uniform (for $\delta \in [0, \infty)$) if any $X, Y \in \{0,1\}^s$ give

$$\mathbf{Pr}\left(G(K, X) = Y\right) \leq 2^{-\delta}$$

for a randomly drawn $K \xleftarrow{\$} \{0,1\}^k$. Similarly, we consider it to be $2^{-\varepsilon}$-universal (for $\varepsilon \in [0, \infty)$) if any distinct $X, X' \in \{0,1\}^s$ give

$$\mathbf{Pr}\left(G(K, X) = G(K, X')\right) \leq 2^{-\varepsilon}$$

for a randomly drawn $K \xleftarrow{\$} \{0,1\}^k$.

2.3 Multi-Collisions

In many proofs of sponge-like constructions, the bounding of the probability of multi-collisions appearing plays a central role. For instance, the works on the full-state keyed duplex [DMV17] and the suffix keyed sponge [DM20b] use a parameter called the multi-collision limit function. Let us consider $b, c, q \in \mathbb{N}$ with $c \leq b$. The multi-collision limit function $\mu^q_{b-c,b}$ serves to limit probabilistically the maximum number of identical $(b-c)$-bit elements μ one gets when drawing q $(b-c)$-bit elements uniformly from random. Clearly, the probability that the maximal number of elements remains smaller than $\mu^q_{b-c,b}$ depends on the concrete choice of $\mu^q_{b-c,b}$. Therefore, $\mu^q_{b-c,b}$ is defined to be the smallest natural number x that

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

Daemen et al. [DMV17] provide an in-depth analysis of $\mu^q_{b-c,b}$ with the conclusion that $\mu^q_{b-c,b}$ behaves roughly like

$$\mu^q_{b-c,b} \lesssim \begin{cases} b / \log_2\left(\dfrac{2^{b-c}}{q}\right), & \text{for } q \lesssim 2^{b-c}\,, \\ b \cdot \dfrac{q}{2^{b-c}}\,, & \text{for } q \gtrsim 2^{b-c}\,. \end{cases}$$

Comparable bounds have appeared in the context of cryptography in works of Jovanovic et al. [JLM+19] and Choi et al. [CLL19].

Since we consider multi-collisions from an attack point of view in this work, however, we are more interested in the maximal μ-collision we get *in the average case* after q queries. For values uniformly randomly drawn with replacement, we can resort to a result of Suzuki et al. [STKT06] that says that for

$$q \approx \left(\mu! \cdot 2^{(b-c)(\mu-1)}\right)^{1/\mu}$$

a μ-collision on a $(b-c)$-bit value is found with probability around $1/2$. In our case, however, we are concerned with μ-collisions on *truncated permutation outputs*. Therefore, we will derive a slightly different estimation.

First, consider any set of $\mu \leq q$ evaluations, and assume that we draw μ values $U_1, \ldots, U_\mu$ without replacement from a b-bit uniform distribution. These μ evaluations all

have colliding $\text{right}_{b-c}(U_i)$ with probability (here, assume $\mu \leq 2^c$):

$$\begin{aligned}\mathbf{Pr}\left(\mu\text{-collision after } \mu \text{ draws}\right) &= 1 \cdot \frac{2^c-1}{2^b-1} \cdot \frac{2^c-2}{2^b-2} \cdot \dots \cdot \frac{2^c-(\mu-1)}{2^b-(\mu-1)} \\ &= \frac{(2^c-1)! \cdot (2^b-\mu)!}{(2^c-\mu)! \cdot (2^b-1)!} = \frac{\binom{2^c-1}{\mu-1}}{\binom{2^b-1}{\mu-1}}.\end{aligned}$$

Hence, the expected number of μ-collisions on the inner $b-c$ bits of the state U_i, among all q values, equals

$$\mathbf{Ex}\left(\#\mu\text{-collision after } q \text{ draws}\right) = \binom{q}{\mu} \frac{\binom{2^c-1}{\mu-1}}{\binom{2^b-1}{\mu-1}}. \tag{3}$$

Typically, we want to find at least one μ-collision, so we require

$$\binom{q}{\mu} \frac{\binom{2^c-1}{\mu-1}}{\binom{2^b-1}{\mu-1}} \geq 1,$$

or equivalently,

$$\binom{q}{\mu} \geq \frac{\binom{2^b-1}{\mu-1}}{\binom{2^c-1}{\mu-1}}.$$

Our goal is to derive a lower bound for q as function of μ, using that $2^b > 2^c > q \gg \mu$. Using (1), we can further get

$$\left(\frac{q}{\mu}\right)^{\mu} \geq \left(\frac{(2^b-1)e}{2^c-1}\right)^{\mu-1},$$

or equivalently,

$$q \geq \sqrt[\mu]{(\mu)^{\mu} \cdot \left(\frac{(2^b-1)e}{2^c-1}\right)^{\mu-1}}, \tag{4}$$

which is quite close to Suzuki et al.'s [STKT06] approximation, and sufficient for our purposes.

3 The Suffix Keyed Sponge

3.1 The Construction

Let $k, b, c, r, s, t \in \mathbb{N}$. The suffix keyed sponge (SuKS) is a PRF construction that takes as input a k-bit key and an arbitrarily long plaintext, and outputs a t-bit tag. Internally, it operates on a b-bit state and is built on top of a b-bit permutation. The suffix keyed sponge construction is depicted in Figure 1. As the name suggests, the secret key K is the last element that is absorbed, while the plaintext blocks get absorbed before.

In more detail, during absorption of the plaintext, the b-bit state is split into an r-bit outer part and a c-bit inner part. The arbitrarily long plaintext is first injectively padded and split into r-bit blocks P^i. These blocks are absorbed by XORing them with the outer part of the state. After the absorption of each single plaintext block, the permutation p is applied to the state. After the last plaintext block is absorbed, the sizes of the inner and outer part are re-defined as $b-s$ and s, and the k-bit key is absorbed via a function G into the s-bit outer part. Finally, the permutation is applied a last time to the state, and t bits from the state are squeezed to produce the tag T. An algorithmic description can be found in Algorithm 1.

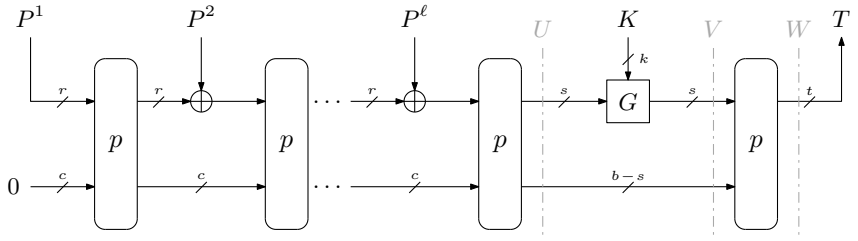

Figure 1: The suffix keyed sponge. The plaintext P is first injectively padded into r-bit blocks $P^1 \dots P^\ell$.

Algorithm 1 Algorithmic description of the suffix keyed sponge (SuKS)

Input: $(P, K) \in \{0,1\}^* \times \{0,1\}^k$
Output: $T \in \{0,1\}^t$
1: $P^1 \dots P^\ell \leftarrow P \| 1 \| 0^*$
2: $S \leftarrow 0^b$
3: **for** $i = 1, \dots, \ell$ **do**
4: $\quad S \leftarrow S \oplus P^i \| 0^c$
5: $\quad S \leftarrow p(S)$
6: $S \leftarrow G(K, \mathrm{left}_s(S)) \| \mathrm{right}_{b-s}(S)$
7: $S \leftarrow p(S)$
8: **return** $\mathrm{left}_t(S)$

3.2 The Bound

Dobraunig and Mennink [DM20b] give a security proof for SuKS in the leakage-resilient setting as well as in the black-box setting. Since we focus on generic attacks against this construction, we will stick to the bound in the black-box setting [DM20b, Theorem 2], which we restate in Theorem 1.

Theorem 1 (Dobraunig and Mennink [DM20b, Theorem 2])**.** *Let $k, b, c, r, s, t \in \mathbb{N}$ with $c + r = b$ and $k, s, t \leq b$. Consider the suffix keyed sponge shown in Figure 1. The suffix keyed sponge invokes a random permutation $p \xleftarrow{\$} \mathsf{perm}(b)$, and a function $G : \{0,1\}^k \times \{0,1\}^s \to \{0,1\}^s$ that is $2^{-\delta}$-uniform and $2^{-\varepsilon}$-universal for some $\delta, \varepsilon \in [0, \infty)$. For any adversary $\mathcal{A}$ making $q \geq 2$ queries to the construction oracle (SuKS based on p and instantiated with secret key, or a random function \$) and a total amount of $N \leq 2^{b-1}$ queries to the primitive p,*

$$\mathbf{Adv}_F^{\mathrm{prf}}(\mathcal{A}) \leq \frac{2N^2}{2^c} + \frac{\mu_{b-s,s}^{2(N-q)} \cdot N}{2^{\min\{\delta,\varepsilon\}}} + \frac{\mu_{t,b-t}^{q} \cdot N}{2^{b-t}} \,. \tag{5}$$

3.3 Interpretation of the Bound

The security bound of Theorem 1 consists of three terms.

The first term $\frac{2N^2}{2^c}$ of (5) corresponds to inner collisions on the plaintext absorption. This is a common term for keyless sponges (noting that, prior to the absorption of the key, SuKS is a keyless sponge), and is inevitable. The term implies that security of SuKS degenerates with the square of the number of calls N to the underlying primitive only in the first term, and that for a k-bit security level, the condition $c \gtrsim 2k$ must be fulfilled.

In the second and third term of (5), security degenerates with the product of the multi-collision limit function (either $\mu_{b-s,s}^{2(N-q)}$ or $\mu_{t,b-t}^{q}$) and N. As these multi-collision terms are typically small, one might get away with inner parts $b - s$ and $b - t$ that are

smaller than $2k$. As a matter of fact, it appears that for prominent choices of c, r, s, and t, the dominant term in (5) is either $\frac{2N^2}{2^c}$ or $\frac{\mu_{b-s,s}^{2(N-q)} \cdot N}{2^{\min\{\delta,\varepsilon\}}}$.

Consider for example the usage of SuKS in the Isap authenticated encryption scheme [DEM+17, DEM+19,DEM+20]. If it is instantiated with the 400-bit variant of the Keccak permutation, it is parameterized as $c = 256$, $r = 144$, $s = 128$, and $t = 128$, and we obtain [DM20b]:

$$\mathbf{Adv}_F^{\mathrm{prf}}(\mathcal{A}) \leq \frac{2N^2}{2^{256}} + \frac{3N}{2^{128}} + \frac{80N}{2^{272}} .$$

Alternatively, if it is instantiated with the 320-bit Ascon permutation, it is parameterized as $c = 256$, $r = 64$, $s = 128$, and $t = 128$, and we obtain [DM20b]:

$$\mathbf{Adv}_F^{\mathrm{prf}}(\mathcal{A}) \leq \frac{2N^2}{2^{256}} + \frac{5N}{2^{128}} + \frac{67N}{2^{192}} . \tag{6}$$

In both cases, we see that the second term of (5), $\frac{\mu_{b-s,s}^{2(N-q)} \cdot N}{2^{\min\{\delta,\varepsilon\}}}$, is the dominating one.

However, we stress that the result of Theorem 1 considers PRF security. In the next sections, we will explore if the terms of (5) including the multi-collision limit function (either $\mu_{b-s,s}^{2(N-q)}$ or $\mu_{t,b-t}^{q}$) are mere proof artifacts in the concrete use-case of a MAC, or if we can find generic attacks on SuKS as a MAC coming close to the bound. More detailed, in the remainder of this article, we will examine SuKS in the use case of a MAC with the goal to craft a forgery. We will distinguish between two cases: in Section 4 we will consider the case that G is relatively easy to invert, and in Section 5 we will consider the case that G is hard to invert.

4 Attacks with Easy to Invert Absorption

In this section, we discuss generic attacks on SuKS, where $G(K, X) = Y$ is a function that is easy to invert. This means that knowledge of a single pair (X, Y) allows for the efficient computation of K.

A typical example of such a function is $G(K, X) = K \oplus X$, which is 2^{-k}-uniform and 0-universal for k-bit values. For this function, a single pair (X, Y) allows one to deduce K as $K = G^{-1}(Y, X) = X \oplus Y$. For all attacks in this section, we set $s = k$. At first, in Section 4.1 we show how to exploit multi-collisions on the tag T. This attack corresponds to the third term of (5). Then, in Section 4.2 we discuss how to use multi-collision on the $(b - k)$-bit value during the absorption of the key. This attack corresponds to the second term of (5).

Note that there is some stretch in the meaning of "efficient computation" of K. Suppose a single pair (X, Y) allows for the efficient computation of K with probability π only. In this case, the attacks of this section would only succeed with a probability π, and one would have to do at least the computation of G on average $1/\pi$ times. If π becomes too small, this will have a similar cost as a brute force search for the key; in such a case, i.e., if the probability of computing the key from a single pair becomes too small, we talk about hard to invert absorption. This will be the topic in Section 5.

4.1 Multi-Collision on the Tag

The following attack is a key recovery attack that exploits multi-collisions on the tag T and exploits the fact that the secret key is absorbed at a single point in SuKS, while all used operations are easy to invert. The key recovery attack on SuKS is performed as follows:

(1) Make q construction queries for different plaintexts P_i to get tags T_i and corresponding U_i (see Figure 1).

(2) Find a μ-fold collision in the tags, i.e., such that μ values i satisfy $T_i = T^\star$ for some $T^\star$.

(3) Make N_I primitive queries $p^{-1}(T^\star \| Z_j)$ for varying Z_j. If the outcome is of the form $Y \| \text{right}_{b-k}(U_i)$ for some of the i's in the μ-fold collision, compute the key $K = G^{-1}(Y, \text{left}_k(U_i))$. Verify K by making one more encryption query.

4.1.1 Application

The complexity of above attack depends on the sizes chosen for b and t. For simplicity, assume that we set $t = k$, and $c = 2k$ as imposed by the first term of (5). Then, for $b = 256$ and $k = 128$, we can argue that above attack is expected to succeed for $(q, N) \approx (2^{124.1}, 2^{125.8})$. This is because, by (4), one requires

$$q \geq \sqrt[\mu]{(\mu)^\mu \cdot \left(\frac{(2^{256}-1)e}{2^{128}-1}\right)^{\mu-1}},$$

to obtain a μ-fold collision on the 128-bit tag with high probability. Concretely, for $\mu = 14$ and $q \approx 2^{124.1}$, a 14-fold collision can be obtained. From this, the key recovery in step (3) takes $2^{128}/14 \approx 2^{124.2}$ primitive queries. Since we can assume that each construction query needs at least two primitive queries, we get $N = 2^{124.2} + 2 \cdot 2^{124.1} \approx 2^{125.8}$ primitive queries in total.

Note that, acknowledgedly, the attack suffers from a rather high data complexity. Furthermore, in the case of SuKS, we would require $c/2 > 128$ to be able to achieve 128 bits of security. Hence, an instance of SuKS using a 128-bit key will likely use a b larger than 256 bits, such as $b = 272$ as used in Spongent-256 [BKL+11]. However, a rising b makes step (3), that requires to guess a $(b-t)$-bit value, quickly very expensive. This means that attacks corresponding to the other terms in (5) are typically much more efficient.

4.1.2 Remark

The attack mostly exploits the fact that the key is absorbed at a single position using an easy to invert mechanism. As such, the attack is not just for restricted use against SuKS, but variants of the attack also work on other permutation-based MACs and authenticated encryption schemes that absorb the key at a single position relying on easy to invert mechanisms of absorption of the data. For example, the attack is comparable to the multi-collision attack [DM20a] recently mounted against PHOTON-Beetle, a second round candidate of the NIST lightweight cryptography standardization process [BCD+19]. A comparable attack was mounted on sponge-based AE schemes by Chakraborty et al. [CJN20].

4.2 Multi-Collision on the Inner Part During Key Absorption

The attack from Section 4.1 aims to speed up the guess of a $(b-t)$-bit value with the help of multi-collisions. As explained, however, this becomes quickly no threat for practical values. In addition, this attack suffers from a rather high complexity in construction queries, often also referred to as online complexity. These problems are mitigated in an attack that focuses on multi-collisions targeting the inner part *during key absorption*. The attack works as follows:

(1) Use N_p primitive queries to compute SuKS on N_x different plaintexts P_i up to the state value U_i (see Figure 1). Since there is no key involved up to this point, these only count as primitive queries.

(2) Find a μ-fold collision in the $\text{right}_{b-k}(U_i)$, i.e., such that μ values i satisfy $\text{right}_{b-k}(U_i) = U^\star$ for some $U^\star$.

(3) For all μ plaintexts P_i with the same $\text{right}_{b-k}(U_i) = U^\star$, make a construction query to get the corresponding T_i.

(4) Make N_f primitive queries $p(Z_j \| U^\star)$ for varying Z_j. If the outcome matches T_i for some of the i's in the μ-fold collision, compute the key $K = G^{-1}(Z_j, \text{left}_k(U_i))$. Verify K by making one more encryption query.

4.2.1 Attack Complexity

We discuss the complexity of the attack. For simplicity, we assume that $t = k$. The number N_x of different plaintext we process determines the number of μ-collisions we expect to find on a $(b-k)$-bit value. From (4), we have

$$N_x \geq \sqrt[\mu]{(\mu)^\mu \cdot \left(\frac{(2^b-1)e}{2^k-1}\right)^{\mu-1}}. \tag{7}$$

Depending on the size of the rate r and the number N_x of different plaintext we process, we require more than one plaintext block to be processed, namely $\lceil \log_2(N_x)/r \rceil$ blocks. Hence, the number of primitive queries N_p made to the permutation p in this phase differs from N_x and is

$$N_p = \sum_{i=0}^{\lceil \log_2(N_x)/r \rceil} \frac{N_x}{2^{r \cdot i}}.$$

We have to make construction queries for all plaintexts belonging to our μ-collisions, which means that $q = \mu$. Finally, we have to make additionally N_f primitive queries for the permutation p, which sum to $N_f = 2^k/\mu$ on average and also count the $(\lceil \log_2(N_x)/r \rceil + 1)q$ primitive queries made during the q construction queries. To sum up, we expect to need

$$N = 2^k/\mu + (\lceil \log_2(N_x)/r \rceil + 1)q + \sum_{i=0}^{\lceil \log_2(N_x)/r \rceil} \frac{N_x}{2^{r \cdot i}}$$

queries to p, or p^{-1}, with N_x being the term of (7), and

$$q = \mu$$

queries to the construction, which also gives the data complexity.

4.2.2 Application

Assume that we take parameters $b = 272$, $k = 128$, and $t = c/2 = k$. Then, we can get a μ-collision on a 144-bit value with (cf., (7))

$$N_x \geq \sqrt[\mu]{(\mu)^\mu \cdot \left(\frac{(2^{272}-1)e}{2^{128}-1}\right)^{\mu-1}}.$$

Hence, we can find a $(\mu = 6)$-fold collision with a complexity of $2^{123.8}$. However, since we have a rate of 16 bits, we need to absorb 8 blocks to create $2^{123.8}$ values of U_i, which needs

$$N_p = 2^{123.8} + 2^{107.8} + 2^{91.8} + 2^{75.8} + 2^{59.8} + 2^{43.8} + 2^{27.8} + 2^{11.8} \approx 2^{123.9}$$

calls to p. Then, we need $q = 6$ calls to the construction with the plaintexts that give the μ-collision to get 6 tags T_i. This speeds up the search for a correct $\text{left}_k(V_i)$ by a factor of 6, thus needing another $N_f = 2^{128}/6 \approx 2^{125.415}$ calls to p. For this parameter set, we need a total of $q = 6$ online construction queries and $N = N_p + N_f + (\lceil \log_2(N_x)/r \rceil + 1)q \approx 2^{125.9}$.

If we look at a different parameter set with $b = 320$, $k = 128$, and $t = c/2 = k$, we see that already a 3-collision on 192 bits is quite unlikely. However, it is easily possible to get a 2-collision needing approximately 2^{98} permutation queries. Hence, a key recovery attack with these parameters requires $q = 2$ online construction queries and 2^{127} permutation queries.

5 Attack with Hard to Invert Absorption

The attacks of Section 4 aimed to recover $\text{left}_k(V_i)$ and then recover K from that. However, for some prominent use-cases of SuKS, like Isap [DEM+17, DEM+19, DEM+20], G is actually hard to invert so that the knowledge of an input-output tuple does not easily allow for a key recovery. The question, thus, remains if generic attacks on SuKS using a hard to invert G are possible that exploit the existence of multi-collisions.

In this section, we describe a forgery attack on SuKS as a MAC, even if a PRF is used for G. The attack, to a certain extent, corresponds the second term of (5). This makes sense: already for the case of easy to invert absorption the attack corresponding to the second term was more efficient than the attack corresponding to the third term of (5).

5.1 Attack Description

The attack is significantly more involved than those of Section 4, and therefore we will describe the steps in greater detail. As before, we use the notation of U, V, and W as shown in Figure 1 for the state before G, after G, and after the last application of p, respectively.

The attack consists of the following steps:

Finding multi-collisions. Use N_p primitive queries to compute SuKS on N_x different plaintexts P_i up to the state value U_i (see Figure 1). Since there is no key involved up to this point, these only count as primitive queries. The goal is to find a μ-fold collision in the $\text{right}_{b-s}(U_i)$, i.e., such that μ values i satisfy $\text{right}_{b-s}(U_i) = U^\star$ for some $U^\star$.

We will compute the expected number of μ-collisions in these N_x values U_i. Analogue to (3), the expected number of μ-collisions on the inner $(b-s)$ bits of the state U_i, among all N_x values, equals

$$\mathbf{Ex}\left(\#\mu\text{-collision after } N_x \text{ draws}\right) = \binom{N_x}{\mu} \frac{\binom{2^s-1}{\mu-1}}{\binom{2^b-1}{\mu-1}}.$$

Denote this value by Ξ.

Filter for suitable tuples. For each of the Ξ μ-collisions that have matching inner parts $\text{right}_{b-s}(U_i)$, but in general differing outer parts $\text{left}_s(U_i)$, we want to find for each of the μ different outer parts independently a collision with any of the $N_x - 1$ other queried values. For a single μ-collision, this happens with a probability of around

$$\left(\frac{N_x}{2^s}\right)^\mu.$$

We stress that these collisions need not be unique: it might for instance be that two queries in the μ-collision happened to collide on the entire state. The attack still

works: looking ahead, the reason that we need to link each value in the μ-collision with another query through $\mathrm{left}_s(U_i)$ is because this query will be used to eventually mount the forgery.

Stated differently, we wish to obtain a μ-collision consisting of a set $\mathcal{S}$ of μ different plaintexts P_i whose corresponding values U_i satisfy $\mathrm{right}_{b-s}(U_i) = U^\star$ for some $U^\star$. In addition, associated to this set is a second set $\mathcal{S}'$ of μ plaintexts P'_i such that $P_i \neq P'_i$ but $\mathrm{left}_s(U_i) = \mathrm{left}_s(U'_i)$ for all i. A visualization of the collision structure is given in Figure 2.

Recovering derived keys. For all plaintext values $P_1, \ldots, P_\mu$ in the set $\mathcal{S}$, we make a query to the SuKS oracle to get the corresponding tags T_i. These are $q = \mu$ queries in total.

Then, we guess the secret part Y corresponding to the inner part $\mathrm{right}_{b-s}(V_i) = \mathrm{right}_{b-s}(U_i) = U^\star$ shared among all μ queries (by design of $\mathcal{S}$). In detail, we guess the secret part Y and compute a guessed tag $T^\star$ by computing $p(Y\|U^\star)$. This tag matches one of the μ tags T_i with a probability $\mu/2^t$. Hence, after $N_f = 2^t/\mu$ guesses, we expect one match. Therefore, for one of the plaintexts in the set $\mathcal{S}$, say $P_{i^\star}$, we know the values

$$\begin{aligned} U_{i^\star} &= \mathrm{left}_s(U_{i^\star})\|\mathrm{right}_{b-s}(U_{i^\star}) \\ V_{i^\star} &= Y\|\mathrm{right}_{b-s}(U_{i^\star}) = Y\|U^\star\,, \\ T_{i^\star} &= T^\star\,. \end{aligned}$$

Forgery if G is a PRF. We will use these data, together with the colliding plaintext in the set $\mathcal{S}'$, to mount a forgery. In detail, recall that, by definition of $\mathcal{S}'$, the plaintext $P'_{i^\star} \in \mathcal{S}'$ satisfies $\mathrm{left}_s(U'_{i^\star}) = \mathrm{left}_s(U_{i^\star})$. By construction, this value will also have colliding outer parts *after evaluation of* G: $\mathrm{left}_s(V'_{i^\star}) = \mathrm{left}_s(V_{i^\star}) = Y$. Therefore, we can compute the tag for plaintext $P'_{i^\star}$ as

$$T'_{i^\star} = \mathrm{left}_t(p(Y\|\mathrm{right}_{b-s}(U'_{i^\star})))\,.$$

The pair $(P'_{i^\star}, T'_{i^\star})$ is a valid forgery.

5.2 Attack Complexity

We discuss the complexity of the attack assuming that $t = k = s$. In the first part of the attack, we search for values of μ and N_x, so that

$$1 \leq \left(\frac{N_x}{2^s}\right)^\mu \binom{N_x}{\mu} \frac{\binom{2^s-1}{\mu-1}}{\binom{2^b-1}{\mu-1}}\,.$$

We will derive a lower bound for N_x as function of μ. Here, we remark that $2^b > 2^s > N_x \gg \mu$. Clearly, above bound is satisfied if

$$\left(\frac{N_x}{2^s}\right)^\mu \binom{N_x}{\mu} \geq \frac{\binom{2^b-1}{\mu-1}}{\binom{2^s-1}{\mu-1}}\,.$$

Using (1), we obtain that the above holds provided that

$$(N_x)^{2\mu} \geq (\mu 2^s)^\mu \cdot \left(\frac{(2^b-1)e}{2^s-1}\right)^{\mu-1}\,,$$

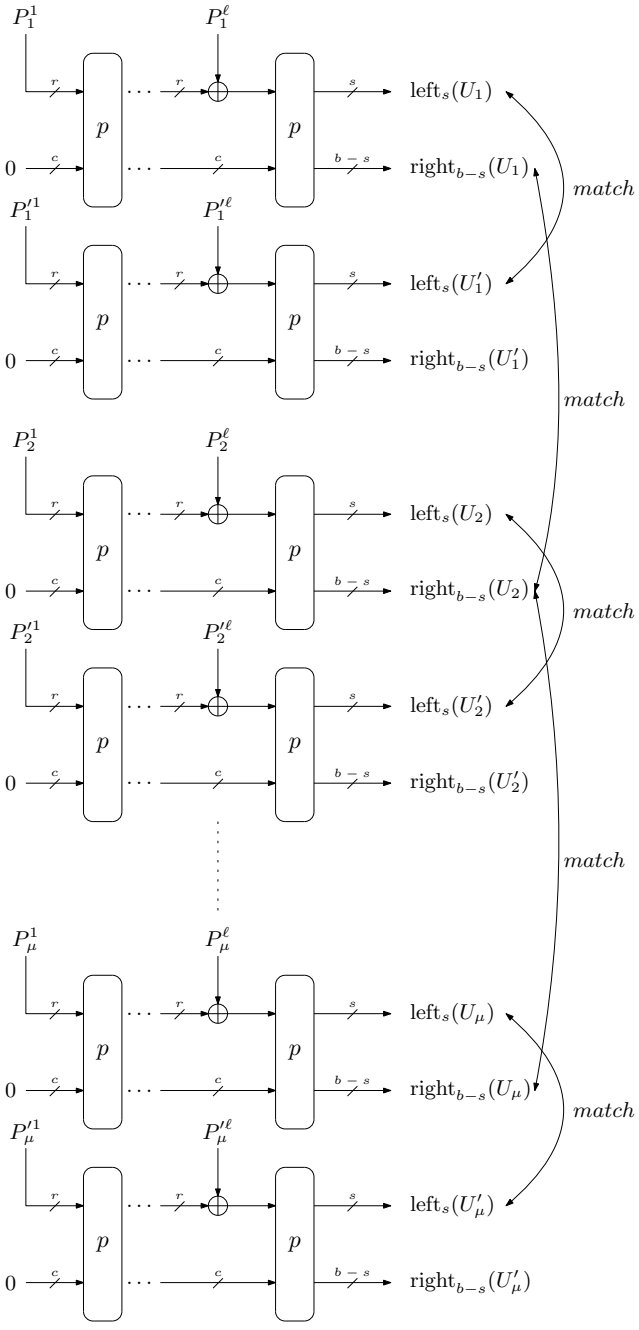

Figure 2: Visualization of the required collision structure. Here, plaintexts $P_1, \ldots, P_\mu$ are all distinct, and $P_i' \neq P_i$ for each i.

or equivalently,

$$N_x \geq \sqrt[2\mu]{(\mu 2^s)^\mu \cdot \left(\frac{(2^b-1)e}{2^s-1}\right)^{\mu-1}}. \tag{8}$$

This then accounts for

$$N_p = \sum_{i=0}^{\lceil \log_2(N_x)/r \rceil} \frac{N_x}{2^{r\cdot i}}$$

queries to the permutation p, noting that the absorption rate equals r. The online construction query complexity, which equals the data complexity, equals $q = \mu$. Finally, we have to do $N_f = 2^k/\mu$ queries on p before we can do a forgery on average. In total, we make $N = N_p + N_f + (\lceil \log_2(N_x)/r \rceil + 1)q$ queries to the permutation.

5.3 Application

5.3.1 Spongent-256-like Instance

To compute the complexity of the attack, it is most convenient to consider a concrete application. Let us assume the state geometry used in Spongent-256 [BKL+11], with $r = 16$ and $c = 256$, but with in addition a PRF G with $k = s = 128$ to insert the 128-bit key. We consider tag size $t = k$ as before. In this case, we have $b - s = 144$. Considering (8) and $\mu = 5$, we should choose

$$N_x \geq \sqrt[2\cdot 5]{(2^{128}\cdot 5)^5 \cdot \left(\frac{(2^{272}-1)e}{2^{128}-1}\right)^{5-1}} > 2^{123.4}.$$

Note, however, that this is a rather loose bound on the value of N_x. For this concrete instance, we observe that a slightly better choice of $N_x = 2^{122.5}$ should be sufficient for our attack with $\mu = 5$.

If we consider $\mu = 5$, we expect to find

$$\Xi = \binom{N_x}{\mu}\frac{\binom{2^s-1}{\mu-1}}{\binom{2^b-1}{\mu-1}} = \binom{2^{122.5}}{5}\frac{\binom{2^{128}-1}{5-1}}{\binom{2^{272}-1}{5-1}} \approx 2^{29.5}$$

5-collisions on the inner part of U by computing $2^{122.5}$ plaintexts P_i up to state value U_i. Such a single 5-collision also has independent outer part collisions in one of the other $2^{122.5}$ values with a probability of

$$\left(\frac{N_x}{2^s}\right)^\mu = \left(\frac{2^{122.5}}{2^{128}}\right)^5 \approx 2^{-27.5}.$$

Hence, we expect to have at least one valid 5-collision (in the sense of Figure 2) in our $2^{29.5}$ candidates. To compute $2^{122.5}$ values P_i with a 16-bit rate, we need

$$N_p = 2^{122.5} + 2^{106.5} + 2^{90.5} + 2^{74.5} + 2^{58.5} + 2^{42.5} + 2^{26.5} + 2^{10.5} \approx 2^{122.5}$$

evaluations of the permutation p.

Then, we query the construction oracle for those $q = 5$ different plaintexts of the 5-collision to get 5 tags. We expect to get one matching value for the value $\mathrm{left}_s(V_j)$ after $N_f = 2^{128}/5 = 2^{125.678}$ calls to p. This match also gives us the ability to compute one forgery. To sum up, the attack requires an online/data complexity of $q = 5$ chosen messages, needing a total evaluation of $N = N_p + N_f + (\lceil \log_2(N_x)/r \rceil + 1)q \approx 2^{125.83}$ permutation calls.

5.3.2 MAC of Isap's Ascon Instances

In the case of the MACs used in the instance of Isap based on the 320-bit Ascon permutation, we have $r = 64$, $c = 256$, $k = s = t = 128$, and $b - s = 192$. In contrast to the above, we are only able to find many 2-collisions, e.g., $2^{36.9}$ 2-collisions with after $N_p \approx 2^{115}$ calls to p. A single 2-collision is expected to have a collision with another of the 2^{115} values with a probability of 2^{-30}. Hence, we expect to have one valid 2-collision (in the sense of Figure 2). Since we only have a 2-collision, we can only speed up the exhaustive search for $\text{left}_s(V_j)$ by a factor 2. Hence, we have to make approximately $N_f \approx 2^{127}$ calls to p before we find a match. A match in $\text{left}_s(V_j)$ lets us compute one forgery. To sum up, this attack requires an online/data complexity of $q = 2$ chosen messages, needing a total evaluation of $N = N_p + N_f + (\lceil \log_2(N_x)/r \rceil + 1)q \approx 2^{127}$ permutation calls.

5.4 Remarks

The advantage of 2^{127} permutation calls in the case of the parameter set of the MACs of Isap's Ascon instances seems rather negligible if we compare it with a brute force key search. For example, a brute force key search already succeeds with a probability of 0.5 after 2^{127} calls to the permutation p and the function G. However, note that the computation of G can be equally expensive as a permutation call. A typical choice of G is for instance a sponge-based PRF with a rate of 1 bit, e.g, using permutation p_G. Hence, a brute force key search needs in addition to the offline computation of 2^{127} calls to p an additional $\sum_{i=0}^{127} \left(2^{127-i}\right) = 2^{128}$ calls to p_G, while our attack does not count any offline evaluations of G.

Furthermore, we remark that it is possible to turn the attack into an attack that exploits multi-collisions in the tag, akin to the first attack of Section 4. However, in this case the first phase of the attack, "finding multi-collisions", becomes an online phase as the adversary must make N_x construction queries. Therefore, we consider this variant of the attack inferior to the one described above.

6 Conclusion

In this paper, we have shown generic attacks on the suffix keyed sponge, which come close to the bounds given by Dobraunig and Mennink [DM20b]. While the bounds given by Dobraunig and Mennink consider the PRF security of SuKS, our attacks are performed when SuKS is considered as a MAC. Note that an attack against the MAC security can easily be turned into an attack against the PRF security. We were able to show generic attacks in this scenario for cases where the absorption of the key K by the function G is easy to invert, e.g., by an XOR, or hard to invert, e.g., by a PRF. For the easy to invert case, the shown attacks recover the key, while for the hard to invert case, we aim to craft a forgery. Although the attack in case where G is hard to invert is significantly more involved, the complexities for many parameter sets are close. For instance, for a Spongent-256-like instance of SuKS, the computational complexity is about $2^{125.9}$ in both cases. Also in the case of the MAC of Isap's Ascon instances the difference is negligible and in both cases approximately 2^{127}.

Nevertheless, if we compare the complexities we get from the attacks with the advantage we expect from the bound, the attack complexities are a bit higher. The provable security approach basically aims to use an upper bound on the probability that a μ-collision happens for all μ bigger than a certain threshold, where the probability is kept comparably low. In contrast, for attacks, we are interested in maximizing μ in μ-collisions in the average case. Hence, in attacks, we work with a μ that is typically smaller than what is considered in the provable security analysis.

Putting this into numbers, consider the message authentication in Isap's Ascon instances [DEM+17, DEM+19, DEM+20]. In the bound, the value $\mu_{b-s,s}^{2(N-q)}$ is bounded by $\mu_{192,128}^{2^{129}} \leq 5$, as in the second term of (6). In contrast, in the attack of Section 5.3.2, we have chosen the value for the μ-collision as $\mu = 2$, in order to get the best attack complexity on average. This is why the bound of (6) already indicates an advantage larger than 1 for $N = 2^{125.78}$, while we need $N = 2^{127}$ in the attacks. For the Spongent-256-like instance we discussed in Section 4.2.2 and Section 5.3.1, we chose $\mu = 6$ and $\mu = 5$ to improve the attack complexity on average, while a bound would have typically $\mu_{144,128}^{2^{129}} \leq 16$ following [DMV17, Equation (32)].

To sum up, the attacks show that the dominating terms appearing in the bounds are justified, even if G is a PRF, and regardless of whether SuKS is used as a PRF or a MAC.

Acknowledgments

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

References

[ADMV15] Elena Andreeva, Joan Daemen, Bart Mennink, and Gilles Van Assche. Security of Keyed Sponge Constructions Using a Modular Proof Approach. In Gregor Leander, editor, *FSE 2015*, volume 9054 of *LNCS*, pages 364–384. Springer, 2015.

[BCD+19] Zhenzhen Bao, Avik Chakraborti, Nilanjan Datta, Jian Guo, Mridul Nandi, Thomas Peyrin, and Kan Yasuda. PHOTON-Beetle Authenticated Encryption and Hash Family. Submission to NIST Lightweight Cryptography, 2019.

[BDPV07] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sponge functions. Ecrypt Hash Workshop 2007, May 2007.

[BDPV08] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the Indifferentiability of the Sponge Construction. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 181–197. Springer, 2008.

[BDPV11a] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Cryptographic sponge functions (Version 0.1). https://keccak.team/, January 2011.

[BDPV11b] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications. In Ali Miri and Serge Vaudenay, editors, *SAC 2011*, volume 7118 of *LNCS*, pages 320–337. Springer, 2011.

[BDPV11c] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the Security of the Keyed Sponge Construction. Symmetric Key Encryption Workshop, February 2011.

[BDPV11d] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. The Keccak SHA-3 submission (Version 3.0). http://keccak.noekeon.org/Keccak-submission-3.pdf, 2011.

[BKL+11] Andrey Bogdanov, Miroslav Knezevic, Gregor Leander, Deniz Toz, Kerem Varici, and Ingrid Verbauwhede. spongent: A Lightweight Hash Function. In Bart Preneel and Tsuyoshi Takagi, editors, *CHES 2011*, volume 6917 of *LNCS*, pages 312–325. Springer, 2011.

[CDH+12] Donghoon Chang, Morris Dworkin, Seokhie Hong, John Kelsey, and Mridul Nandi. A Keyed Sponge Construction with Pseudorandomness in the Standard Model. NIST SHA-3 Workshop, March 2012.

[CJN20] Bishwajit Chakraborty, Ashwin Jha, and Mridul Nandi. On the Security of Sponge-type Authenticated Encryption Modes. *IACR Transactions on Symmetric Cryptology*, 2020(2):93–119, Jul. 2020.

[CLL19] Wonseok Choi, ByeongHak Lee, and Jooyoung Lee. Indifferentiability of Truncated Random Permutations. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019*, volume 11921 of *LNCS*, pages 175–195. Springer, 2019.

[DEM+17] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, and Thomas Unterluggauer. ISAP – Towards Side-Channel Secure Authenticated Encryption. *IACR Transactions on Symmetric Cryptology*, 2017(1):80–105, Mar. 2017.

[DEM+19] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, Bart Mennink, Robert Primas, and Thomas Unterluggauer. ISAP v2. Submission to NIST Lightweight Cryptography, 2019.

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

[DEMS19] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2. Submission to NIST Lightweight Cryptography, 2019.

[DL14] Itai Dinur and Gaëtan Leurent. Improved Generic Attacks against Hash-Based MACs and HAIFA. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014*, volume 8616 of *LNCS*, pages 149–168. Springer, 2014.

[DM56] Abraham De Moivre. *The doctrine of chances*. 1756.

[DM20a] Christoph Dobraunig and Bart Mennink. Key Recovery Attack on PHOTON-Beetle. OFFICIAL COMMENT on NIST mailing list: https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/official-comments/photon-beetle-round2-official-comment.pdf, March 2020.

[DM20b] Christoph Dobraunig and Bart Mennink. Security of the Suffix Keyed Sponge. *IACR Transactions on Symmetric Cryptology*, 2019(4):223–248, Jan. 2020.

[DMM20] Christoph Dobraunig, Florian Mendel, and Bart Mennink. Practical forgeries for ORANGE. *Inf. Process. Lett.*, 159-160:105961, 2020.

[DMV17] Joan Daemen, Bart Mennink, and Gilles Van Assche. Full-State Keyed Duplex with Built-In Multi-user Support. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017*, volume 10625 of *LNCS*, pages 606–637. Springer, 2017.

[GPT15] Peter Gazi, Krzysztof Pietrzak, and Stefano Tessaro. The Exact PRF Security of Truncation: Tight Bounds for Keyed Sponges and Truncated CBC. In Rosario Gennaro and Matthew Robshaw, editors, *CRYPTO 2015*, volume 9215 of *LNCS*, pages 368–387. Springer, 2015.

[JLM14] Philipp Jovanovic, Atul Luykx, and Bart Mennink. Beyond $2^{c/2}$ Security in Sponge-Based Authenticated Encryption Modes. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014*, volume 8873 of *LNCS*, pages 85–104. Springer, 2014.

[JLM+19] Philipp Jovanovic, Atul Luykx, Bart Mennink, Yu Sasaki, and Kan Yasuda. Beyond Conventional Security in Sponge-Based Authenticated Encryption Modes. *J. Cryptology*, 32(3):895–940, 2019.

[LNS18] Gaëtan Leurent, Mridul Nandi, and Ferdinand Sibleyras. Generic Attacks Against Beyond-Birthday-Bound MACs. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018*, volume 10991 of *LNCS*, pages 306–336. Springer, 2018.

[LPW13] Gaëtan Leurent, Thomas Peyrin, and Lei Wang. New Generic Attacks against Hash-Based MACs. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013*, volume 8270 of *LNCS*, pages 1–20. Springer, 2013.

[LS18] Gaëtan Leurent and Ferdinand Sibleyras. The Missing Difference Problem, and Its Applications to Counter Mode Encryption. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018*, volume 10821 of *LNCS*, pages 745–770. Springer, 2018.

[Men18] Bart Mennink. Key Prediction Security of Keyed Sponges. *IACR Transactions on Symmetric Cryptology*, 2018(4):128–149, Dec. 2018.

[MRH04] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 21–39. Springer, 2004.

[MRV15] Bart Mennink, Reza Reyhanitabar, and Damian Vizár. Security of Full-State Keyed Sponge and Duplex: Applications to Authenticated Encryption. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015*, volume 9453 of *LNCS*, pages 465–489. Springer, 2015.

[Nat15] National Institute of Standards and Technology. FIPS PUB 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. Federal Information Processing Standards Publication 202, U.S. Department of Commerce, August 2015.

[NY16] Yusuke Naito and Kan Yasuda. New Bounds for Keyed Sponges with Extendable Output: Independence Between Capacity and Message Length. In Thomas Peyrin, editor, *FSE 2016*, volume 9783 of *LNCS*, pages 3–22. Springer, 2016.

[Pv95] Bart Preneel and Paul C. van Oorschot. MDx-MAC and Building Fast MACs from Hash Functions. In Don Coppersmith, editor, *CRYPTO '95*, volume 963 of *LNCS*, pages 1–14. Springer, 1995.

[RS19] Raghvendra Rohit and Sumanta Sarkar. Trivial Key Recovery Attack on Oribatida-192-96. OFFICIAL COMMENT on NIST mailing list: https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/official-comments/Oribatida-round2-official-comment.pdf, September 2019.

[Sti30] James Stirling. *Methodus Differentialis sive Tractatus de Summatione et Interpolatione Serierum Infinitarum*. 1730.

[STKT06] Kazuhiro Suzuki, Dongvu Tonien, Kaoru Kurosawa, and Koji Toyota. Birthday Paradox for Multi-collisions. In Min Surp Rhee and Byoungcheon Lee, editors, *ICISC 2006*, volume 4296 of *LNCS*, pages 29–40. Springer, 2006.