Tight Preimage Resistance of the Sponge Construction*

Charlotte Lefevre and Bart Mennink

Digital Security Group, Radboud University, Nijmegen, The Netherlands
charlotte.lefevre@ru.nl, b.mennink@cs.ru.nl

Abstract. The cryptographic sponge is a popular method for hash function design. The construction is in the ideal permutation model proven to be indifferentiable from a random oracle up to the birthday bound in the capacity of the sponge. This result in particular implies that, as long as the attack complexity does not exceed this bound, the sponge construction achieves a comparable level of collision, preimage, and second preimage resistance as a random oracle. We investigate these state-of-the-art bounds in detail, and observe that while the collision and second preimage security bounds are tight, the preimage bound *is not tight*. We derive an improved and tight preimage security bound for the cryptographic sponge construction.
The result has direct implications for various lightweight cryptographic hash functions. For example, the NIST Lightweight Cryptography finalist Ascon-Hash does not generically achieve 2^{128} preimage security as claimed, but even 2^{192} preimage security. Comparable improvements are obtained for the modes of Spongent, PHOTON, ACE, Subterranean 2.0, and QUARK, among others.

Keywords: sponge, hash function, preimage security, tightness

1 Introduction

The sponge construction of Bertoni et al. [9] is a popular approach for cryptographic hashing. At a high level, the sponge operates on a state of size b bits, which is split into an inner part of size c bits (the capacity) and an outer part of size r bits (the rate), where $b = c + r$. The sponge consists of an absorbing phase and a squeezing phase. In the absorbing phase, data is compressed into the state r bits at a time, interleaved with an evaluation of a b-bit permutation $\mathcal{P}$. In the squeezing phase, a digest is extracted from the state r bits at a time, again interleaved with an evaluation of $\mathcal{P}$. A slight relaxation of this approach, introduced by the developers of PHOTON [18], is to squeeze at a slightly larger rate $r' \geq r$. Throughout this work, we will in fact consider this generalized description of the sponge, as depicted in Fig. 1, but we will stick to calling it the "sponge".

The sponge found quick adoption right after its introduction, and its popularity is ever-increasing. Most notably, the eventual winner of the NIST SHA-3 competition [22], Keccak [11], relies on the sponge methodology. It was quickly acknowledged that the sponge was particularly well-suited for lightweight hashing, see, e.g., QUARK [3], Spongent [12], and PHOTON [18], and in the ongoing NIST Lightweight Cryptography competition [23], no less than 22 submissions (including 5 finalists) offer hashing via the sponge construction or a derivative thereof.

Two causes for this quick adoption were the conceptual simplicity of the sponge, and its ability to offer variable output length digests (later, functions that facilitate this were dubbed extendable output functions (XOFs) [22]). Another main cause was that the developers [10] proved security of the sponge construction in the indifferentiability framework [13, 20]. In a bit more detail, the authors proved that if $\mathcal{P}$ is assumed to be a random permutation, no adversary with an attack complexity less than $2^{c/2}$ can differentiate the sponge construction from a random oracle. (For the PHOTON construction with larger squeezing rate $r' \geq r$, a comparable bound was proven by Naito and Ohta [21].) The result, in words, implies that the sponge "behaves" like a random oracle and that it can be used in (most) applications that were proven secure in the random oracle model. This result also implies that, assuming that the query complexity is at most $2^{c/2}$, finding collisions, preimages, or second preimages for the sponge is not easier than for a random oracle. Andreeva et al. [2, Appendix A] made this implication explicit and demonstrated that for a sponge construction that outputs a digest of (fixed length) n bits, finding collisions requires at least

$$q \approx \min\{2^{c/2}, 2^{n/2}\} \tag{1}$$

work, and finding preimages or second preimages requires at least

$$q \approx \min\{2^{c/2}, 2^{n}\} \tag{2}$$

work (see also Section 3.1). These bounds have directly influenced the parameter choices of many sponge-based hash designs. Most notably, the SHA-3 hash function family consists of four functions: SHA3-n where $n \in \{224, 256, 384, 512\}$ defines the output size. Each of these four functions has its capacity c equal to *twice* the digest length n (see also Table 1).

It was clear from the start that the indifferentiability bound of Bertoni et al. [10] was tight. As a matter of fact, in around $2^{c/2}$ work, an adversary can find inner collisions, i.e., different sponge evaluations that collide on the c-bit inner part, and it can use these inner collisions to form a full collision for the sponge and this way distinguish it from random. Likewise, the collision security bound of (1) is tight, as a collision for a sponge with fixed n-bit output can be obtained either by finding a c-bit inner collision or an n-bit output collision. Finally, for second preimage resistance, tightness of the bound of (2) can be argued in a comparable way. Clearly, one approach the adversary can take to find a second preimage is an exhaustive search in 2^n work. Alternatively, given the first preimage, the attacker can recompute the sponge on input of this first

preimage to determine the final state value *before* squeezing. Then, it computes the sponge forward from the initial value 0^b and backward from the state value *before* squeezing in order to find a collision on the c-bit inner part.

For preimage security, the situation is different, and it appears that *for certain values c and n*, the bound of (2) is *not tight*. This is mainly caused by the fact that, unlike for second preimage security, the final state *before* squeezing cannot always be easily found. Already in the original introduction of the sponge construction in 2007, it was claimed that a preimage attack can only be found in $\max\{2^{n-r'}, 2^{c/2}\}$ work [9, Section 5.3], where we recall that c is the capacity during absorbing and r' the rate during squeezing (in the original proposal, $r' = r$). In 2011, both the developers of PHOTON and Spongent made a comparable claim regarding the preimage security of their construction [12,18]. We discuss this generic attack in detail in Section 3.2. Here, we also elaborate a bit more on the generic collision and second preimage attacks, noting that they are de facto simplifications of the preimage attack. Unfortunately, *proving* tight preimage security of this level has remained an open problem since.

1.1 Tight Preimage Security

We solve this open problem and prove tight preimage security of the sponge construction. In detail, assuming that the underlying permutation $\mathcal{P}$ is random, we prove that the sponge achieves preimage security up to around

$$q \approx \min\left\{\max\left\{2^{n-r'}, 2^{c/2}\right\}, 2^n\right\} \tag{3}$$

work, where we recall that n is the digest size, c the capacity of the sponge (during absorption), and r' the rate (during squeezing). A detailed bound is given in Section 4, and the bound tightly matches the generic attack of Section 3.2 (up to constant). The security relies on a careful investigation of what events are needed to happen in order for a preimage to be found, and subsequently a detailed computation of the probability of these events to occur.

At a very high level, suppose the attacker aims to obtain a preimage for a digest Z consisting of ℓ r'-bit blocks $Z_1 \| \cdots \| Z_\ell$, assuming $r' \mid n$ for the sake of simplicity. We assume, by definition, that the attacker is required to make all permutation queries that are required for the computation of its eventual preimage, and in particular, it must definitely obtain a cascaded evaluation of $\ell - 1$ permutation queries that correspond to outputs $Z_1, \ldots, Z_\ell$. In Fig. 1, these are the first permutation evaluation *after* outputting Z_1 up to and including the last permutation evaluation *before* outputting Z_ℓ. As we demonstrate in our proof, the attacker succeeds in finding such a path only after around $q \approx 2^{n-r'}$ queries.

However, the adversary is not done after just finding such cascade of permutation evaluations: the evaluations must also be *reached* from 0^b through the absorption of certain message blocks — these message blocks eventually constitute the preimage that the adversary would output. The adversary could succeed in this in two ways: either the last permutation query before squeezing is made

in forward direction, or it is made in inverse direction. If it is made in forward direction, we have to go one step back in our reasoning, namely to the discussion of the squeezing cascade, and observe that in this case the cascade of ℓ permutation evaluations can only be found in $q \approx 2^n$ queries. If it is in inverse direction, this particular permutation query can be made *for free* from the cascade of above $\ell-1$ evaluations, *but* in order to then connect the cascade to the initial value 0^b, the adversary must necessarily ever find a forward and an inverse permutation evaluation that collide on the inner part. This, in turn requires approximately $2^{c/2}$ work.

In summary, finding a preimage requires either around 2^n work, or the maximum of $2^{n-r'}$ and $2^{c/2}$ work, exactly as expressed in (3). Needless to say, the actual security analysis, and in particular the derivation of an upper bound on the probability of finding a matching cascaded permutation evaluation of length $\ell-1$ or ℓ, is much more involved, among others as any permutation query of the adversary may appear at any position in this cascade.

1.2 Application

For hash functions with a large capacity, e.g., Keccak and eventually the SHA-3 hash function family, the old bound of (2) accurately described the preimage security. However, with the advent of lightweight cryptography, many sponge constructions with small permutation size b, small capacity c, and small squeezing rate r' have appeared. In many of these cases, our bound has immediate implications as it confirms higher preimage security.

The ISO/IEC standardized Spongent hash function of Bogdanov et al. [12] and the PHOTON hash function of Guo et al. [18] are two such cases. Spongent consists of five hash functions, all of which are sponges instantiated with a permutation of size $b \in \{88, 136, 176, 240, 272\}$ bits, a rate of $r = r' = 8$ bits for the smallest two versions and $r = r' = 16$ for the larger three, and a capacity $c = b - r$. The smallest version outputs $n = b = 88$ bits whereas the other versions output $n = c$ bits. The old bound of (2) implied that a preimage attack required at least $2^{c/2}$ work, whereas our new bound (3) implies that a preimage attack requires at least 2^{n-r} work. For the smallest version of Spongent, this is an improvement from 2^{40} to 2^{80}, and for the largest version, this is an improvement from 2^{128} to 2^{240}. PHOTON, likewise, consists of five sponge hash functions (with larger squeezing rate than absorbing rate), instantiated with a permutation of size $b \in \{100, 144, 196, 256, 288\}$, corresponding capacities $c \in \{80, 128, 160, 224, 256\}$, and with output size $n = c$. The squeezing rate differs for the five versions, but also here, a significant gain in the security bound is achieved: 2^{40} to 2^{64} for the smallest variant and 2^{128} to 2^{224} for the largest variant.

More recently, a notable example is Ascon-Hash, the hash function in the Ascon [17] finalist in the NIST Lightweight Cryptography competition [23]. Ascon-Hash is a plain sponge construction on top of a $b = 320$-bit permutation, with a capacity $c = 256$ and a rate $r' = 64$. It outputs digests of size $n = 256$ bits, which are thus generated in four squeezes. In this case, the old bound of (2) implied

generic preimage security up to 2^{128} work, whereas our new bound (3) implies generic preimage security up to 2^{192} work. A similar effect is achieved for the modes of other second round and final candidates in the NIST Lightweight Cryptography competition, such as ACE [1], KNOT [25], SKINNY-HASH [6], Subterranean 2.0 [15], the hash proposal of Isap [16], and PHOTON-Beetle [4]. These sponge-based functions all have their parameters (c, r', n) satisfying $n - r' > c/2$.

In Table 1, we give a summary of these hash function constructions, and show how the new preimage security bound improves over the earlier bound. A more detailed evaluation of our new bound for SHA3-256, Spongent with $n = 256$, and Ascon-Hash with $n = 256$ is given in Section 5. We remark that in Table 1, we did not include hash functions that are sponge(-like) but squeeze digests in one round, such as Grindahl [19] and CubeHash [7], as our bound only improves over the state-of-the-art bound for sponge(-like) constructions that squeeze their digest in multiple rounds. Likewise, we did not include hash functions that squeeze digests over multiple rounds but that have a large enough c such that $n - r' \leq c/2$, such as Gimli [8], ESCH [5], and Xoodyak [14].

2 Preliminaries

2.1 Notation

We use $x := y$ to define x as being equal to y. For $b \in \mathbb{N}$, we denote by $\{0,1\}^b$ the set of binary strings of size b. Moreover, $\{0,1\}^*$ is defined to be $\bigcup_{b\in\mathbb{N}}\{0,1\}^b$. For a b-bit string s and $0 \leq x \leq y \leq b-1$, $s[x:y]$ denotes the substring containing the bits of s from position x to y. Moreover, $\mathrm{inner}_x(s) := s[b-x : b-1]$, $\mathrm{outer}_x(s) := s[0 : x-1]$. For a finite set $\mathcal{S}$, $x \xleftarrow{\$} \mathcal{S}$ means that x is sampled uniformly at random from $\mathcal{S}$. The set $Perm(b)$ denotes the set of permutations over $\{0,1\}^b$. For any $\mathcal{P} \in Perm(b)$ and $i \in \mathbb{N}^*$, $\mathcal{P}^0$ denotes the identity function and $\mathcal{P}^i$ is i iterations of $\mathcal{P}$. For $n, k \in \mathbb{N}$ such that $k \leq n$, we use $[n]_k$ to denote the falling factorial of n of depth k, i.e., the product $\prod_{i=0}^{k-1}(n-i)$. We remark that, provided $k^2 \leq n$, we have

$$\begin{aligned}
[n]_k &= n^k \prod_{i=0}^{k-1} \frac{n-i}{n} \\
&\geq n^k e^{\sum_{i=0}^{k-1} \frac{-i}{n-i}} \\
&\geq n^k e^{\sum_{i=0}^{k-1} \frac{-i}{n-k}} \\
&= n^k e^{\frac{-k(k-1)}{2(n-k)}} \\
&\geq n^k e^{-1/2} \\
&\geq \frac{n^k}{2}\,,
\end{aligned} \tag{4}$$

where the first inequality uses $1 + x \leq e^x$ applied with $x = \frac{i}{n-i}$.

Table 1: Preimage security of the modes of SHA-3 (added for reference only, as our bound does not improve the state-of-the-art bound) and selected lightweight hash functions. Security bounds only hold under the assumption that the underlying permutations are ideal.

Scheme	Parameters						Security bound		Note
	b	c	r	r'	n	ℓ	Old (2)	New (3)	
SHA3-n	1600	448	1152	1152	224	1	2^{224}	2^{224}	SHA-3 standard [22] (included for reference)
	1600	512	1088	1088	256	1	2^{256}	2^{256}	
	1600	768	832	832	384	1	2^{384}	2^{384}	
	1600	1024	576	576	512	1	2^{512}	2^{512}	
Spongent	88	80	8	8	88	11	2^{40}	2^{80}	ISO/IEC standard [12]
	136	128	8	8	128	16	2^{64}	2^{120}	
	176	160	16	16	160	10	2^{80}	2^{144}	
	240	224	16	16	224	14	2^{112}	2^{208}	
	272	256	16	16	256	16	2^{128}	2^{240}	
PHOTON	100	80	20	16	80	5	2^{40}	2^{64}	ISO/IEC standard [18]
	144	128	16	16	128	8	2^{64}	2^{112}	
	196	160	36	36	160	5	2^{80}	2^{124}	
	256	224	32	32	224	7	2^{112}	2^{192}	
	288	256	32	32	256	8	2^{128}	2^{224}	
U-QUARK	136	128	8	8	128	16	2^{64}	2^{120}	Family QUARK [3]
D-QUARK	176	160	16	16	160	10	2^{80}	2^{144}	
T-QUARK	256	224	32	32	224	7	2^{112}	2^{192}	
ACE-Hash	320	256	64	64	256	4	2^{128}	2^{192}	NIST LWC round 2 [1]
KNOT Hash	256	224	32	128	256	2	2^{112}	2^{128}	NIST LWC round 2 [25]
	384	256	128	128	256	2	2^{128}	2^{128}	
	384	336	48	192	384	2	2^{168}	2^{192}	
	512	448	64	256	512	2	2^{224}	2^{256}	
SKINNY-tk2-Hash	256	224	32	128	256	2	2^{112}	2^{128}	NIST LWC round 2 [6]
Subterranean 2.0	257	248	9	32	256	8	2^{124}	2^{224}	NIST LWC round 2 [15]
Ascon-Hash	320	256	64	64	256	4	2^{128}	2^{192}	NIST LWC finalist [17]
PHOTON-Beetle-Hash	256	224	32	128	256	2	2^{112}	2^{128}	NIST LWC finalist [4]

2.2 Generalized Sponge Construction

Let $b, c, r, c', r', n \in \mathbb{N}$ with $b = c+r = c'+r'$. Let $\mathcal{P} \in Perm(b)$ be a cryptographic permutation. Let *pad* be an injective padding function that transforms a message M of arbitrary length into k blocks of r bits such that the last block is non-zero. A minimal example is the 10^*-padding that appends M with a one and $(-|M| - 1) \bmod r$ zeros. We will restrict our focus to the sponge construction with a fixed-length output of size n, and we define $\ell = \lceil n/r' \rceil$.

Let $M \in \{0,1\}^*$ be an input message. The sponge construction instantiated with the permutation $\mathcal{P}$, denoted by $\mathcal{H}^{\mathcal{P}} : \{0,1\}^* \to \{0,1\}^n$, is now defined as follows.

- M is first padded into k message blocks using *pad*: $M_1 \| \cdots \| M_k \leftarrow pad(M)$;
- Absorbing phase: the state S is initialized as 0^b, and at the i^{th} iteration, for $i = 1, \ldots, k$, the state is updated as $S \leftarrow \mathcal{P}(S \oplus (M_i \| 0^c))$;
- Squeezing phase: at the i^{th} iteration, for $i = 1, \ldots, \ell$, the outer r' bits of S are extracted as $Z_i \leftarrow \text{outer}_{r'}(S)$ and the state is updated as $S \leftarrow \mathcal{P}(S)$;
- The digest is computed as $Z \leftarrow (Z_1 \| \cdots \| Z_\ell)[0 : n-1]$.

The sponge construction is illustrated in Fig. 1.

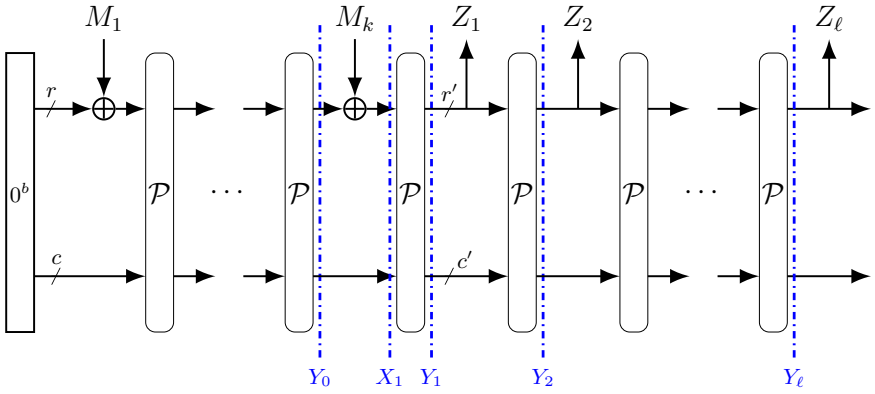


Fig. 1: Generalized sponge construction as described in Section 2.2. The values Y_i and X_i will be used in the proof in Section 4.

2.3 Security Model

An adversary $\mathcal{A}$ is a probabilistic algorithm. It has oracle access to a permutation $\mathcal{P}$ sampled uniformly at random. $\mathcal{A}$ is computationally restricted only by its number of evaluations of $\mathcal{P}$ and $\mathcal{P}^{-1}$, that we denote by q. We summarize all queries made by $\mathcal{A}$ in a query history $\mathcal{Q}$, an ordered list of tuples of the form $(X, Y, \mathrm{d}) \in \{0,1\}^b \times \{0,1\}^b \times \{\text{fwd}, \text{inv}\}$, where $\mathcal{P}(X) = Y$ and where d denotes

the query direction. We denote by $\mathcal{Q}_i$ the query history containing only the first i queries. Without loss of generality, we can assume that the adversary never makes a query that it already made before.

Preimage Resistance. We focus on everywhere preimage resistance [24]. In this model, we consider any image $Z \in \{0,1\}^n$ of length n and consider the adversary $\mathcal{A}$ that can query $\mathcal{P}$ and has as goal to eventually output a message M such that $\mathcal{H}^{\mathcal{P}}(M) = Z$. We require that the query history of $\mathcal{A}$ contains all evaluations of $\mathcal{P}$ required for the computation of $\mathcal{H}^{\mathcal{P}}(M)$.

Definition 1. *Let $b, n, q \in \mathbb{N}$, consider the sponge construction $\mathcal{H}$ of Section 2.2. For any adversary $\mathcal{A}$, we define its everywhere preimage advantage as*

$$\mathbf{Adv}^{\mathrm{epre}}_{\mathcal{H}}(\mathcal{A}) = \max_{Z \in \{0,1\}^n} \mathbf{Pr}\left(\mathcal{P} \xleftarrow{\$} \mathit{Perm}(b),\, M \leftarrow \mathcal{A}^{\mathcal{P}}(Z) : \mathcal{H}^{\mathcal{P}}(M) = Z\right).$$

We define by $\mathbf{Adv}^{\mathrm{epre}}_{\mathcal{H}}(q)$ the supremum advantage over all adversaries making at most q queries.

3 State-of-the-Art Generic Security Results

We will discuss the best known security lower bound in Section 3.1 and the best known generic attack in Section 3.2.

3.1 Security Lower Bound

Maurer et al. [20] introduced the indifferentiability framework as an extension of the notion of indistinguishability. The notion was tailored towards hash functions by Coron et al. [13]. One says that a hash function $\mathcal{H}$ based on an ideal permutation $\mathcal{P}$ is indifferentiable from a random oracle $\mathcal{R}$ if there exists a simulator $\mathcal{S}$ (based on the random oracle) such that $(\mathcal{H}^{\mathcal{P}}, \mathcal{P})$ is hard to distinguish from $(\mathcal{R}, \mathcal{S}^{\mathcal{R}})$. Denote by $\mathbf{Adv}^{\mathrm{indif}}_{\mathcal{H}}(q)$ the indifferentiability of $\mathcal{H}$ against any attacker with total complexity q (the number of primitive evaluations in $(\mathcal{H}^{\mathcal{P}}, \mathcal{P})$).

Bertoni et al. [10] proved that the sponge is indifferentiable from a random oracle up to bound $\mathbf{Adv}^{\mathrm{indif}}_{\mathcal{H}}(q) \leq \frac{q(q+1)}{2^{c+1}}$. Naito and Ohta [21] proved that for the PHOTON construction, indifferentiability holds with a bound of the form $\mathcal{O}\left(\frac{q}{2^{c/2}} + \frac{q}{2^r}\right)$ (refer to [21] for the details).

Indifferentiability of a hash function $\mathcal{H}$ from a random oracle $\mathcal{R}$ in words means that the hash function "behaves" like a random oracle. In the context of preimage resistance, this means that [2, Appendix A]

$$\mathbf{Adv}^{\mathrm{epre}}_{\mathcal{H}}(q) \leq \mathbf{Adv}^{\mathrm{indif}}_{\mathcal{H}}(q) + \mathbf{Adv}^{\mathrm{epre}}_{\mathcal{R}}(q)\,,$$

where we are slightly abusing notation for the latter term: to be precise, for $\mathbf{Adv}^{\mathrm{epre}}_{\mathcal{R}}(q)$ we consider the adversary to have query access to the random oracle

$\mathcal{R}$ and its goal is to output a message M such that $\mathcal{R}(M) = Z$ for the predetermined Z. Clearly, $\mathbf{Adv}^{\mathrm{epre}}_{\mathcal{R}}(q) = q/2^n$, and we thus obtain the state-of-the-art bound for preimage resistance of the sponge:

$$\mathbf{Adv}^{\mathrm{epre}}_{\mathcal{H}}(q) \leq \frac{q(q+1)}{2^{c+1}} + \frac{q}{2^n}\,. \tag{5}$$

Note that this is the bound that supports the complexity estimation given in (2): generically finding a preimage for $\mathcal{H}^{\mathcal{P}}$ requires at least $\min\{2^{c/2}, 2^n\}$ evaluations. A comparable reasoning applies to the second preimage and collision resistance bounds.

3.2 Security Upper Bound

The best known attack, however, does not meet the first term of (5). In this section, we describe the best known preimage attack against the sponge construction. The attack de facto resembles the generic exhaustive preimage search attack and the attack that the sponge developers described in [9].

Let $Z \in \{0,1\}^n$ be any given image. W.l.o.g., we assume the minimal padding of Section 2.2. We make a case distinction depending on the values c, n. As we will focus on tightness up to constant, we will sometimes ignore the fact that any sponge evaluation of a message M of length k costs $k + \ell$ permutation calls, and simply count any such evaluation as 1 query.

- Case $n \leq c/2$. The adversary fixes a message M and queries it to the construction. The query satisfies $\mathcal{H}^{\mathcal{P}}(M) = Z$ with probability around $1/2^n$. After $q \approx 2^n$ attempts, the adversary has with high probability found a preimage M.
- Case $c/2 < n$. The attack consists of two sequential parts.
  - First, the adversary fixes a state value Y_1 such that $Z_1 = \mathrm{outer}_{r'}(Y_1)$. It queries $Y_2 = \mathcal{P}(Y_1)$, $Y_3 = \mathcal{P}^2(Y_1)$, $\ldots$, $Y_\ell = \mathcal{P}^{\ell-1}(Y_1)$. The queries satisfy

$$Z_i = \begin{cases} \mathrm{outer}_{r'}(Y_i) & \text{for } i = 2, \ldots, \ell - 1\,, \\ \mathrm{outer}_{n-(\ell-1)r'}(Y_i) & \text{for } i = \ell\,, \end{cases}$$

    with probability approximately

$$\left(\frac{1}{2^{r'}}\right)^{\ell-2} \cdot \frac{1}{2^{n-(\ell-1)r'}} = \frac{1}{2^{n-r'}}\,.$$

    After $q \approx 2^{n-r'}$ attempts, the adversary has found a state value Y_1 such that ℓ squeezes result in Z.
  - Starting from 0^b, it computes $Y_0^{\rightarrow} := \mathcal{P}(M_1 \| 0^c)$ for q different values M_1. Starting from the value Y_1 found in the first part of the attack, it computes $Y_0^{\leftarrow} := \mathcal{P}^{-1}(\mathcal{P}^{-1}(Y_1) \oplus (M_3 \| 0^c))$ for q different non-zero values

M_3. If $q \approx 2^{c/2}$, there will with high probability be two values M_1, M_3 such that

$$\text{inner}_c(Y_0^{\rightarrow} \oplus Y_0^{\leftarrow}) = 0^c \,.$$

Let $M_2 := \text{outer}_r(Y_0^{\rightarrow} \oplus Y_0^{\leftarrow})$. Define the preimage as the unique message M such that $\text{pad}(M) = M_1 \| M_2 \| M_3$. We remark that if $r \leq c/2$ one will need multiple message blocks for both the forward and the inverse part in order to make $q \approx 2^{c/2}$ evaluations, but the attack works in a comparable way.

In total, in this case the attack requires $q \approx 2^{n-r'} + 2^{c/2}$ evaluations.

In general, the attack thus has a complexity of around

$$q \approx \min\{2^{n-r'} + 2^{c/2}, 2^n\}$$

evaluations. We remark that the generic second preimage attack, as sketched in Section 1, is basically a simplification of above preimage attack, where in the case of $c/2 < n$, the attacker does not need to perform the first part of the attack but can rather compute Y_1 in a constant number of permutation evaluations from the first preimage. The generic collision attack, also as sketched in Section 1, in turn differs from this second preimage attack in the sense that for $n/2 \leq c/2$ the attacker can perform exhaustive collision search in around $2^{n/2}$ evaluations (instead of 2^n).

4 Improved Preimage Resistance Lower Bound

In the following theorem, we state our main result.

Theorem 1. *Let $b, c, r, c', r', n, q \in \mathbb{N}$ with $b = c + r = c' + r'$, and let $\ell := \lceil \frac{n}{r'} \rceil$. If $q \leq 2^{c'-1}/3$ and $(\ell - 1)^2 \leq 2^b$ the sponge construction $\mathcal{H}$ of Section 2.2 satisfies the following bound:*

$$\mathbf{Adv}^{\text{epre}}_{\mathcal{H}}(q) \leq \frac{4q}{2^n} + \min\left\{\frac{4\ell q}{2^{n-r'}}, \frac{q(q+1)}{2^c}\right\} . \tag{6}$$

The proof is given in the remainder of this section. We note that the bound indeed matches the generic attack of Section 3.2 up to constant. In Section 5, we will evaluate the bound of Theorem 1 more closely, and compare it with the generic attack of Section 3.2 and the state-of-the-art bound of Section 3.1.

4.1 Setup

Let $Z \in \{0,1\}^n$ be any image, and write $Z = Z_1 \| Z_2 \| \cdots \| Z_\ell$, where $|Z_i| = r'$ for $i \in \{1, \ldots, \ell - 1\}$ and $|Z_\ell| = s \leq r'$. Consider any adversary $\mathcal{A}$ as defined in Section 2.3. To represent its knowledge from the query history, we use a graph representation as done for example in [10, 21]. Initially, the graph contains the

nodes $\{0,1\}^b$, which represent all possible internal states of the sponge. For each query $(X, Y, \mathrm{d}) \in \mathcal{Q}$ with $d \in \{\mathrm{fwd}, \mathrm{inv}\}$, and for any $M \in \{0,1\}^r$, the edge $Y' \xrightarrow{M} Y$ is added, where $Y' := X \oplus (M \| 0^c)$. Note that in the squeezing phase, such edge must appear for a zero-block message. In this case, the label is omitted. Let $\boldsymbol{Z}_i$ be defined as follows:

$$\boldsymbol{Z}_i := \begin{cases} \{Y_i \in \{0,1\}^b \mid \mathrm{outer}_{r'}(Y_i) = Z_i\}\,, & \text{for } i \in \{1, \ldots, \ell - 1\}\,, \\ \{Y_i \in \{0,1\}^b \mid \mathrm{outer}_{s}(Y_i) = Z_i\}\,, & \text{for } i = \ell\,. \end{cases}$$

Then, the goal of $\mathcal{A}$ is to find a preimage of Z, which implies the following event $\mathsf{PRE}(\mathcal{Q})$:

$$\mathsf{PRE}(\mathcal{Q}) : \mathcal{Q} \text{ defines a path } 0^b \xrightarrow{M_1} \cdots \xrightarrow{M_{k-1}} Y_0 \xrightarrow{M_k} Y_1 \longrightarrow \cdots \longrightarrow Y_\ell \\ \text{such that } Y_i \in \boldsymbol{Z}_i \text{ for } i = 1, \ldots, \ell\,.$$

We refer to Fig. 1 for a depiction of these parameters. In the case of the minimal injective padding presented in Section 2.2, finding a preimage corresponds to $\mathsf{PRE}(\mathcal{Q})$ with the restriction that the last message block is not zero. In such case, the preimage found by $\mathcal{A}$ is the unique message M such that $\mathrm{pad}(M) = M_1 \| \cdots \| M_k$.

4.2 Logic

We separate the event $\mathsf{PRE}(\mathcal{Q})$ as the disjoint union of the following two events:

$$\mathsf{PREFWD}(\mathcal{Q}) : \mathsf{PRE}(\mathcal{Q}) \text{ with the restriction that the query linking } Y_0 \text{ and } Y_1 \text{ must be made in forward direction}\,,$$
$$\mathsf{PREINV}(\mathcal{Q}) : \mathsf{PRE}(\mathcal{Q}) \text{ with the restriction that the query linking } Y_0 \text{ and } Y_1 \text{ must be made in inverse direction}\,.$$

Clearly,

$$\mathsf{PRE}(\mathcal{Q}) \iff \mathsf{PREFWD}(\mathcal{Q}) \vee \mathsf{PREINV}(\mathcal{Q})\,. \tag{7}$$

We will consider dedicated trigger points for $\mathsf{PREFWD}(\mathcal{Q})$ and $\mathsf{PREINV}(\mathcal{Q})$. Let $\mathcal{S} = \{Y_1 \mid \mathcal{P}^{i-1}(Y_1) \in \boldsymbol{Z}_i \text{ for all } i \in \{1, \ldots, \ell\}\} \subseteq \boldsymbol{Z}_1$. We define the set $\mathcal{S}_{\mathrm{fwd}}$ and the multiset $\mathcal{S}_{\mathrm{inv}}$ as follows:

$$\mathcal{S}_{\mathrm{fwd}} = \{\mathcal{P}^{-1}(Y_1) \mid Y_1 \in \mathcal{S}\}\,,$$
$$\mathcal{S}_{\mathrm{inv}} = \{Y_1, \mathcal{P}(Y_1), \ldots, \mathcal{P}^{\ell-1}(Y_1) \mid Y_1 \in \mathcal{S}\}\,.$$

Intuitively, $\mathcal{S}_{\mathrm{inv}}$ includes all the nodes $Y_1, \ldots, Y_\ell$ appearing in paths which set $\mathsf{PRE}(\mathcal{Q})$ if discovered, while $\mathcal{S}_{\mathrm{fwd}}$ captures all values X_1 from which such path $Y_1 \to \cdots \to Y_\ell$ starts. Looking ahead, $\mathcal{S}_{\mathrm{fwd}}$ includes the set of trigger points for $\mathsf{PREFWD}(\mathcal{Q})$, and $\mathcal{S}_{\mathrm{inv}}$, as a multiset, includes the set of trigger points for

$\mathsf{PREINV}(\mathcal{Q})$. These trigger points can be repeated in $\mathcal{S}_{\text{inv}}$ when some values Z_i are colliding. (Based on this, we will typically simply refer to $\mathcal{S}_{\text{inv}}$ as a set.) We now introduce the following three events:

$$\begin{aligned}
\mathsf{BADFWD}(\mathcal{Q}) &: \exists (X, Y, \text{fwd}) \in \mathcal{Q} \text{ such that } X \in \mathcal{S}_{\text{fwd}}\,, \\
\mathsf{BADINV}(\mathcal{Q}) &: \exists (X, Y, \text{fwd}) \in \mathcal{Q} \text{ such that } X \in \mathcal{S}_{\text{inv}} \text{ or} \\
&\quad\ \exists (X, Y, \text{inv}) \in \mathcal{Q} \text{ such that } Y \in \mathcal{S}_{\text{inv}}\,, \\
\mathsf{INNER}(\mathcal{Q}) &: \exists (X, Y, \text{fwd}), (X', Y', \text{inv}) \in \mathcal{Q} \cup \{(, 0^b, \text{fwd})\} \\
&\quad\ \text{such that } \text{inner}_c(Y) = \text{inner}_c(X')\,.
\end{aligned}$$

Note that for $\mathsf{INNER}(\mathcal{Q})$, the tuple $(, 0^b, \text{fwd})$ is explicitly added to cover the case where the adversary ever makes an inverse query that hits the initial state 0^b. Here, the first element of the tuple is irrelevant, and henceforth omitted.

Intuitively, for $\mathsf{PREFWD}(\mathcal{Q})$ to be set, the adversary must among others make a query $\mathcal{P}(X_1)$ with $X_1 \in \mathcal{S}_{\text{fwd}}$. Thus, $\mathsf{PREFWD}(\mathcal{Q})$ implies $\mathsf{BADFWD}(\mathcal{Q})$. Likewise, for $\mathsf{PREINV}(\mathcal{Q})$ to be set, the adversary must ever make a query that appears in a path $Y_1 \to \cdots \to Y_\ell$ in a query direction. In other words, it must ever query a value in $\mathcal{S}_{\text{inv}}$. Hence, $\mathsf{PREINV}(\mathcal{Q})$ implies $\mathsf{BADINV}(\mathcal{Q})$. Moreover, given that $\mathsf{PREINV}(\mathcal{Q})$ defines a path starting from 0^b that contains an edge between Y_0 and Y_1 corresponding to an inverse query, *somewhere* in this path from 0^b to Y_1 there must be a collision between an inverse query and a descendant of 0^b. This means that $\mathsf{PREINV}(\mathcal{Q})$ also implies $\mathsf{INNER}(\mathcal{Q})$. More formally:

$$\mathsf{PREFWD}(\mathcal{Q}) \implies \mathsf{BADFWD}(\mathcal{Q})\,, \tag{8}$$

$$\mathsf{PREINV}(\mathcal{Q}) \implies \mathsf{BADINV}(\mathcal{Q}) \wedge \mathsf{INNER}(\mathcal{Q})\,. \tag{9}$$

From (7) to (9), we logically obtain

$$\mathsf{PRE}(\mathcal{Q}) \implies \mathsf{BADFWD}(\mathcal{Q}) \vee (\mathsf{BADINV}(\mathcal{Q}) \wedge \mathsf{INNER}(\mathcal{Q}))\,, \tag{10}$$

and thus

$$\mathbf{Pr}\left(\mathsf{PRE}(\mathcal{Q})\right) \le \mathbf{Pr}\left(\mathsf{BADFWD}(\mathcal{Q})\right) + \min\left\{\mathbf{Pr}\left(\mathsf{BADINV}(\mathcal{Q})\right), \mathbf{Pr}\left(\mathsf{INNER}(\mathcal{Q})\right)\right\}. \tag{11}$$

4.3 Probability computation

We upper bound the three probabilities of (11), starting with $\mathbf{Pr}\left(\mathsf{INNER}(\mathcal{Q})\right)$ in Lemma 1, then $\mathbf{Pr}\left(\mathsf{BADFWD}(\mathcal{Q})\right)$ in Lemma 2, and finally $\mathbf{Pr}\left(\mathsf{BADINV}(\mathcal{Q})\right)$ in Lemma 3.

Lemma 1. *We have*

$$\mathbf{Pr}\left(\mathsf{INNER}(\mathcal{Q})\right) \le \frac{q(q+1)}{2^c}\,. \tag{12}$$

Proof (Proof of Lemma 1). We index the queries by the query number, i.e., the i^{th} query is denoted by (X^i, Y^i, d^i). $\mathsf{INNER}(\mathcal{Q})$ translates to the fact that either there is an inner collision between the set of forward and inverse queries, or that the output of an inverse query inner collides with 0^c. More formally, this implies that there exists $i \in \{1, \ldots, q\}$ such that one of the following two events happens:

$$\begin{aligned}
\mathsf{HIT}_i^{\text{fwd}}(\mathcal{Q}) &: (X^i, Y^i, \text{fwd}) \in \mathcal{Q} \text{ and} \\
&\quad \text{inner}_c(Y^i) \in \{\text{inner}_c(X^1), \ldots, \text{inner}_c(X^{i-1})\}\,, \\
\mathsf{HIT}_i^{\text{inv}}(\mathcal{Q}) &: (X^i, Y^i, \text{inv}) \in \mathcal{Q} \text{ and} \\
&\quad \text{inner}_c(X^i) \in \{\text{inner}_c(Y^1), \ldots, \text{inner}_c(Y^{i-1}), \text{inner}_c(0^c)\}\,.
\end{aligned}$$

By basic probability theory,

$$\begin{aligned}
\mathbf{Pr}\left(\mathsf{INNER}(\mathcal{Q})\right) &\leq \sum_{i=1}^{q} \mathbf{Pr}\left(\mathsf{INNER}(\mathcal{Q}_i) \wedge \neg\mathsf{INNER}(\mathcal{Q}_{i-1})\right) \\
&\leq \sum_{i=1}^{q} \mathbf{Pr}\left(\mathsf{HIT}_i^{\text{fwd}}(\mathcal{Q}_i) \vee \mathsf{HIT}_i^{\text{inv}}(\mathcal{Q}_i) \;\middle|\; \neg\mathsf{INNER}(\mathcal{Q}_{i-1})\right)\,.
\end{aligned}$$

For any i, the query is either in forward direction or in inverse direction, so it can only set one of the two events. The response at the i^{th} query is uniformly drawn from a set of size at least $2^b - q$, among which at most $i2^r$ elements set $\mathsf{HIT}_i^{\text{fwd}}(\mathcal{Q}_i)$ or $\mathsf{HIT}_i^{\text{inv}}(\mathcal{Q}_i)$. Thus:

$$\mathbf{Pr}\left(\mathsf{HIT}_i^{\text{fwd}}(\mathcal{Q}_i) \vee \mathsf{HIT}_i^{\text{inv}}(\mathcal{Q}_i) \;\middle|\; \neg\mathsf{INNER}(\mathcal{Q}_{i-1})\right) \leq \frac{i2^r}{2^b - q}\,.$$

Then, as $q \leq 2^{b-1}$,

$$\mathbf{Pr}\left(\mathsf{INNER}(\mathcal{Q})\right) \leq \sum_{i=1}^{q} \frac{2i}{2^c} \leq \frac{q(q+1)}{2^c}\,. \qquad \square$$

Remark 1. We remark that the PHOTON construction [18] in fact differs from the sponge construction [9] not only in the size of the squeezing blocks (as explained in Section 1), but also in the absorption of the *first* message block. To be precise, the PHOTON construction allows the first message block to be of size r'' bits. Extending our analysis, this would *only* affect the analysis of $\mathbf{Pr}\left(\mathsf{INNER}(\mathcal{Q})\right)$ in Lemma 1 above. In this case, the event $\mathsf{HIT}_i^{inv}(\mathcal{Q})$ would be triggered if $\text{inner}_c(X^i) \in \{\text{inner}_c(Y^1), \ldots, \text{inner}_c(Y^{i-1})\}$ or if $\text{inner}_{c''}(X^i) = 0^{c''}$, where $c'' := b - r''$. This change would eventually result in a bound of the form:

$$\mathbf{Pr}\left(\mathsf{INNER}(\mathcal{Q})\right) \leq \frac{q(q+1)}{2^c} + \frac{q}{2^{c''}}\,.$$

Lemma 2. *We have*

$$\mathbf{Pr}\left(\mathsf{BADFWD}(\mathcal{Q})\right) \leq \frac{4q}{2^n}\,. \tag{13}$$

Proof (Proof of Lemma 2). By basic probability theory,

$$\begin{aligned}&\mathbf{Pr}\left(\mathsf{BADFWD}(\mathcal{Q})\right)\\ &\qquad = \sum_{y=1}^{2^{c'}} \mathbf{Pr}\left(\mathsf{BADFWD}(\mathcal{Q}) \mid |\mathcal{S}_{\mathrm{fwd}}| = y\right) \cdot \mathbf{Pr}\left(|\mathcal{S}_{\mathrm{fwd}}| = y\right)\,. \end{aligned} \tag{14}$$

We start by upper bounding the probability of the conditioned $\mathsf{BADFWD}(\mathcal{Q})$ event for any $y = 1, \ldots, 2^{c'}$, which is similar to a guessing game: in order to win, the adversary must guess $X_1 \in \mathcal{S}_{\mathrm{fwd}}$ with a forward query. We start by remarking that $\mathcal{S}_{\mathrm{fwd}}$ is defined via inverse $\mathcal{P}$-calls, and that the adversary has no a priori knowledge about those. Thus, one single query $\mathcal{P}(X)$ from the adversary succeeds with probability at most $\frac{y}{2^b}$. Moreover, one query eliminates at most one candidate: if the query (X, Y, d) does not set $\mathsf{BADFWD}(\mathcal{Q})$, then X can be removed from the set of candidates values to be in $\mathcal{S}_{\mathrm{inv}}$. If additionally $\mathrm{d} = \mathrm{inv}$ and $X \in S_{\mathrm{fwd}}$, the adversary cannot guess this value anymore. Thus, defining $\mathsf{BADFWD}(\mathcal{Q}_0) := \bot$,

$$\begin{aligned}&\mathbf{Pr}\left(\mathsf{BADFWD}(\mathcal{Q}) \mid |\mathcal{S}_{\mathrm{fwd}}| = y\right)\\ &\qquad \leq \sum_{i=1}^{q} \mathbf{Pr}\left(\mathsf{BADFWD}(\mathcal{Q}_i) \mid |\mathcal{S}_{\mathrm{fwd}}| = y \wedge \neg\mathsf{BADFWD}(\mathcal{Q}_{i-1})\right)\\ &\qquad \leq \sum_{i=1}^{q} \frac{y}{2^b - i + 1} \leq \frac{yq}{2^b - q} \leq 2\frac{yq}{2^b}\,, \end{aligned} \tag{15}$$

where in the last inequality, we used $q \leq 2^{b-1}$.

Plugging this bound into (14) gives

$$\begin{aligned}\mathbf{Pr}\left(\mathsf{BADFWD}(\mathcal{Q})\right) &\leq \frac{2q}{2^b} \sum_{y=1}^{2^{c'}} y \cdot \mathbf{Pr}\left(|\mathcal{S}_{\mathrm{fwd}}| = y\right)\\ &\leq \frac{2q}{2^b} \mathsf{E}\left(|\mathcal{S}_{\mathrm{fwd}}|\right)\,. \end{aligned} \tag{16}$$

It remains to compute $\mathsf{E}\left(|\mathcal{S}_{\mathrm{fwd}}|\right) = \mathsf{E}\left(|\mathcal{S}|\right)$. For any $Y \in \{0,1\}^b$, define Bernoulli variable I_Y as

$$I_Y = 1 \iff Y \in \mathcal{S}\,.$$

Note that $I_Y = 0$ whenever $Y \notin \boldsymbol{Z}_1$. We have

$$\begin{aligned}
\mathsf{E}\left(|\mathcal{S}|\right) &= \mathsf{E}\left(\sum_{Y\in\{0,1\}^b} I_Y\right) \\
&= \sum_{Y\in\boldsymbol{Z}_1} \mathsf{E}\left(I_Y\right) \\
&= \sum_{Y\in\boldsymbol{Z}_1} \mathbf{Pr}\left(Y \in \mathcal{S}\right) \\
&\leq \sum_{Y\in\boldsymbol{Z}_1} \frac{2^{c'}}{2^b}\frac{2^{c'}}{2^b-1}\cdots\frac{2^{b-s}}{2^b-(\ell-2)} \\
&= \frac{(2^{c'})^{\ell-1}\cdot 2^{b-s}}{[2^b]_{\ell-1}} \\
&\leq 2\frac{(2^{c'})^{\ell-1}\cdot 2^{b-s}}{(2^b)^{\ell-1}}\,,
\end{aligned}$$

where the last inequality uses (4). Therefore,

$$\mathsf{E}\left(|\mathcal{S}|\right) \leq 2\frac{2^b}{2^n}\,. \tag{17}$$

Finally, from (16) and (17), we thus obtain

$$\mathbf{Pr}\left(\mathsf{BADFWD}(\mathcal{Q})\right) \leq \frac{4q}{2^n}\,, \tag{18}$$

which completes the proof. □

Lemma 3. *We have*

$$\mathbf{Pr}\left(\mathsf{BADINV}(\mathcal{Q})\right) \leq \frac{4\ell q}{2^{n-r'}}\,. \tag{19}$$

Proof (Proof of Lemma 3). We first note that if $\ell = 1$, $|\mathcal{S}_{\text{inv}}| = 2^{c'} \leq 2^{b-n}$, and the result will be meaningless as $\mathsf{BADINV}(\mathcal{Q})$ can be set with probability 1. We will henceforth focus on the case $\ell \geq 2$.

Similar to the proof of Lemma 2, by basic probability we obtain

$$\begin{aligned}
&\mathbf{Pr}\left(\mathsf{BADINV}(\mathcal{Q})\right) \\
&\qquad = \sum_{y=1}^{2^{c'}} \mathbf{Pr}\left(\mathsf{BADINV}(\mathcal{Q}) \mid |\mathcal{S}_{\text{inv}}| = \ell y\right) \cdot \mathbf{Pr}\left(|\mathcal{S}_{\text{inv}}| = \ell y\right)\,,
\end{aligned} \tag{20}$$

where we used that $\mathcal{S}_{\text{inv}}$ is a multiset with a size multiple of ℓ. We start by investigating the conditioned $\mathsf{BADINV}(\mathcal{Q})$ event for any $y = 1, \ldots, 2^{c'}$, which is

more involved than $\mathsf{BADFWD}(\mathcal{Q})$ studied in Lemma 2. Because of the condition $|\mathcal{S}_{\text{inv}}| = \ell y$, there are y paths $Y_1 \to \cdots \to Y_\ell$ with $Y_i \in \boldsymbol{Z}_i$ for $i = 1, \ldots, \ell$. The adversary wins if it ever queries a value that is on any of these paths. Note that this is different from the proof of Lemma 2, where the adversary had to guess any starting point of a path. In the current setting, the attacker learns additional information of failed attempts. For example, suppose that $Z_1 \neq Z_2$ and the adversary makes a forward query $\mathcal{P}(X) = Y$, where $X \in \boldsymbol{Z}_1$ and $Y \in \boldsymbol{Z}_2$ but which does *not* set $\mathsf{BADINV}(\mathcal{Q})$. As it does not set $\mathsf{BADINV}(\mathcal{Q})$, the adversary knows that querying $\mathcal{P}(Y)$ (i.e., guessing $Y \in \boldsymbol{Z}_2$ as candidate value for a chain) is fruitless.

To simplify our reasoning, we will be more generous to the adversary, and for each query input X that the adversary makes, it receives both the forward evaluation $\mathcal{P}(X)$ and the inverse evaluation $\mathcal{P}^{-1}(X)$. Stated differently, for the current game the query direction does not matter, and for each attempt X it learns $\mathcal{P}^{-1}(X) \to X \to \mathcal{P}(X)$. The adversary wins if this is a proper subpath of any of the y target paths $X_1 \to Y_1 \to \cdots \to Y_\ell \to Y_{\ell+1}$, where $X_1 = \mathcal{P}^{-1}(Y_1)$ and $Y_{\ell+1} = \mathcal{P}(Y_\ell)$.[1]

A visualization of this game is given in Fig. 2 for $\ell = 4$. For this example, recall that for $i = 1, 2, 3$, $\boldsymbol{Z}_i$ consists of all values $Y \in \{0,1\}^b$ such that $\text{outer}_{r'}(Y) = Z_i$, and that $\boldsymbol{Z}_4$ consists of all values $Y \in \{0,1\}^b$ such that $\text{outer}_s(Y) = Z_4$. By the conditioned event, there exist y paths through the sets $\boldsymbol{Z}_1, \ldots, \boldsymbol{Z}_4$. In the example, $y = 2$, hence there are two such paths. The adversary sets $\mathsf{BADINV}(\mathcal{Q})$ if and only if it ever queries one of the *at most* ℓy nodes on these lines (not including the ones on the outer shores $\{0,1\}^b$). It is noteworthy that these paths are disjoint: they never cross the same node in the same shore.

Now, suppose the adversary makes a query X, it thus results in a path $\mathcal{P}^{-1}(X) \to X \to \mathcal{P}(X)$. The query is considered a failed query if it is not a proper subpath of any of the y paths. In particular, a failed query either does not intersect with any of the y paths, *or* it intersects with one of the y paths at their very ends. In other words, a query is considered a failed one for path $X_1 \to Y_1 \to \cdots \to Y_\ell \to Y_{\ell+1}$ if and only if it intersects with either of $\varnothing, X_1, X_1 \to Y_1, Y_\ell \to Y_{\ell+1}, Y_\ell$, as illustrated in Fig. 2 (up to symmetry). Any other intersection of the failed query result with the path is impossible, as e.g., depicted in Fig. 2, due to the fact that $\mathcal{P}$ is a permutation.

We remark that for $0 \leq i < j \leq \ell + 1$, a query can be successful for one target path at position i, and failed for another target path at position j at the same time. In this case, the adversary is nevertheless successful. From this, we can conclude that any query attempt either is successful or it eliminates at most 3 possible values from further guessing.

In summary, to win, the adversary must make a query in $\mathcal{S}_{\text{inv}}$. This set is of size at most ℓy and is a subset of the set $\bigcup_{i=1}^{\ell} \boldsymbol{Z}_i$ of size at least $2^{c'}$.[2] Since one

[1] The usage of parameter X_1 in this path, as opposed to Y_0, appears illogical at first sight, but fits the parameter definitions as outlined in Fig. 1.

[2] Note that this correctly captures the case $i = \ell$, as $|\boldsymbol{Z}_\ell| = 2^{n-s} \geq 2^{c'}$.

query eliminates at most 3 candidates and this is the only information available for the adversary, after $i-1$ unsuccessful attempts, the i^{th} attempt succeeds with probability at most $\frac{\ell y}{2^{c'}-3(i-1)}$. Thus, defining $\mathsf{BADINV}(\mathcal{Q}_0) := \bot$,

$$\begin{aligned}\mathbf{Pr}\left(\mathsf{BADINV}(\mathcal{Q}) \mid |\mathcal{S}_{\text{inv}}| = \ell y\right) \\ &= \sum_{i=1}^{q} \mathbf{Pr}\left(\mathsf{BADINV}(\mathcal{Q}_i) \mid |\mathcal{S}_{\text{inv}}| = \ell y \wedge \neg\mathsf{BADINV}(\mathcal{Q}_{i-1})\right) \\ &\leq \sum_{i=1}^{q} \frac{\ell y}{2^{c'} - 3(i-1)} \leq \frac{\ell y q}{2^{c'} - 3q} \leq 2\frac{\ell y q}{2^{c'}}\,, \end{aligned} \tag{21}$$

where in the last inequality, we used $q \leq 2^{c'-1}/3$.

Now, it remains to plug the bound into (20). We can copy the analysis of Lemma 2 verbatim and obtain

$$\begin{aligned}\mathbf{Pr}\left(\mathsf{BADINV}(\mathcal{Q})\right) &\leq 2\frac{q}{2^{c'}} \sum_{y=1}^{2^{c'}} \ell y \cdot \mathbf{Pr}\left(|\mathcal{S}_{\text{inv}}| = \ell y\right) \\ &\leq 2\frac{q}{2^{c'}} \mathsf{E}\left(|\mathcal{S}_{\text{inv}}|\right) \\ &\leq \frac{4\ell q}{2^{n-r'}}\,, \end{aligned} \tag{22}$$

where the last inequality uses $|\mathcal{S}_{\text{inv}}| = \ell|\mathcal{S}|$ and (17). This completes the proof. □

Remark 2. In the proof of Lemma 3, we bounded the probability that the i^{th} query is successful by $\frac{y}{2^{c'}-3(i-1)}$. There is a small loss in this bound due to various simplifications we had to make. First note that as we provide the adversary both directions of the queries, we slightly increase its knowledge and thus success probability. In addition, each query attempt X has its leftmost r' bits $\text{outer}_{r'}(X)$ fixed, and the adversary thus commits itself to the value Z_i and thus to the position in Fig. 2 the query could occur. However, there is no way to make use of this property, as in the general case, the values $Z_1, \ldots, Z_\ell$ may be equal and the query can nevertheless occur at multiple positions. Finally, in the specific case where some values Z_i are mutually equal, this basically reduces the set of candidates to be in $\mathcal{S}$, thus also the set of possibly successful values, and possibly also the amount of information the adversary learns from a failed attempt.

4.4 Conclusion

From (11), Lemma 1, Lemma 2, and Lemma 3, we obtain

$$\mathbf{Pr}\left(\mathsf{PRE}(\mathcal{Q})\right) \leq \frac{4q}{2^n} + \min\left\{\frac{4\ell q}{2^{n-r'}}, \frac{q(q+1)}{2^c}\right\}, \tag{23}$$

and this completes the proof of Theorem 1.

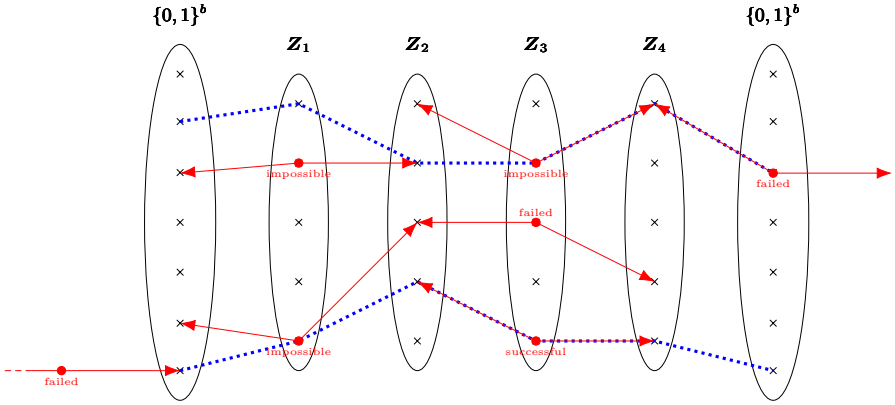


Fig. 2: Illustration of the conditioned $\mathsf{BADINV}(\mathcal{Q})$ event for $\ell = 4$ and $y = 2$. To win, the adversary must guess any node from $\bigcup_{i=1}^{4} \boldsymbol{Z}_i$ on any of the $y = 2$ dotted blue paths. A query attempt X is successful if and only if $\mathcal{P}^{-1}(X) \to X \to \mathcal{P}(X)$ (depicted solid red) is a proper subpath of any of the blue lines. As $\mathcal{P}$ is a permutation, a failed query attempt is either non-overlapping with any of the dotted blue paths, or it may partially overlap only at the ends of a blue line, the other cases are impossible and illustrated as such.

5 Conclusion

In this section, we compare our result with the state-of-the-art bound of (5) (Section 3.1) and the best existing attack. Recall that in Theorem 1, we obtained the following bound:

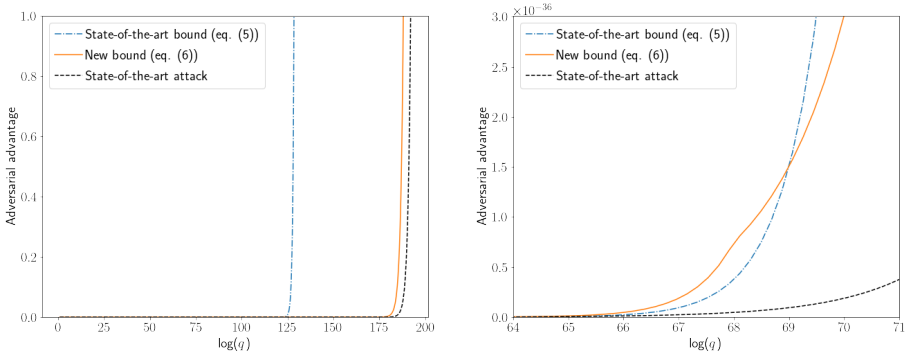
$$\mathbf{Adv}_{\mathcal{H}}^{\mathrm{epre}}(q) \leq \frac{4q}{2^n} + \min\left\{\frac{4\ell q}{2^{n-r'}}, \frac{q(q+1)}{2^c}\right\}.$$

If $\ell = 1$, our security bound matches the state-of-the-art bound up to a factor of 4, while if $\ell > 1$, our bound improves the existing state of the art significantly. In both cases, the bound matches the best known attack outlined in Section 3.2 (up to constant). In the following, we show the improvement with the parameters used in the modes Ascon-Hash [17] and Spongent [12].

First consider the Ascon-Hash mode with parameters $(b, c, r, r', n) = (320, 256, 64, 64, 256)$. In this case, $\ell = n/r' = 4$. In Fig. 3, we compare the state-of-the-art bound of Section 3.1, our new bound of (6), and the best known attack of Section 3.2. We observe that our new bound *significantly* improves the state-of-the-art bound starting at a very low value of q. In detail, the adversarial advantage is approximately 1.5×10^{-36} for $q \approx 2^{69}$ at the intersection point as shown in Fig. 3b, i.e., at the point where the old bound starts to degenerate but our new bound stays low.

It is also interesting to consider the largest mode of Spongent, i.e., with parameters $(b, c, r, r', n) = (272, 256, 16, 16, 256)$. It has a small rate, and consequently a high value $\ell = n/r = 16$, but the same capacity and output size as

the ones of Ascon-Hash. A comparison of the old bound, new bound, and best known attack is given in Fig. 4. Here, the intersection point occurs at $q \approx 2^{23}$, with an advantage of approximately 3×10^{-64}. Our bound thus improves even more the state-of-the-art bound to reach a preimage resistance close to 240 bits.

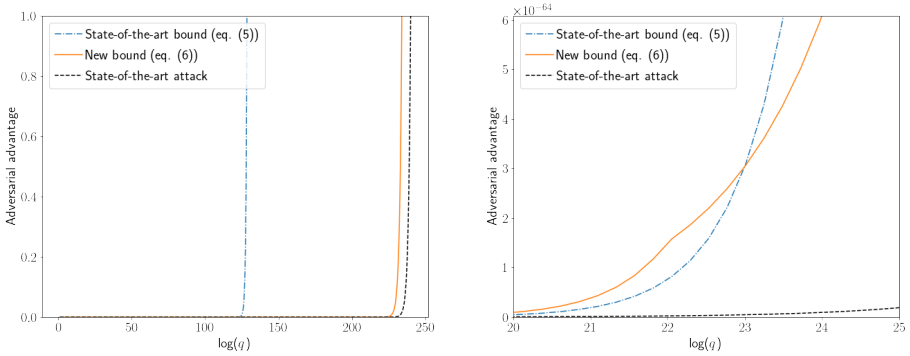


(a) Security bound (b) Close-up of intersection

Fig. 3: Comparison of the state-of-the-art security bound, new security bound, and best known attack for the Ascon-Hash mode with parameters $(b, c, r, r', n) = (320, 256, 64, 64, 256)$.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments, and in particular the reviewer that proposed a fix to the square root loss that was present in an earlier version of the proof. Charlotte Lefevre is supported by the Netherlands Organisation for Scientific Research (NWO) under grant OCENW.KLEIN.435. Bart Mennink is supported by the Netherlands Organisation for Scientific Research (NWO) under grant VI.Vidi.203.099.



(a) Security bound (b) Close-up of intersection

Fig. 4: Comparison of the state-of-the-art security bound, new security bound, and best known attack for the Spongent mode with parameters $(b, c, r, r', n) = (272, 256, 16, 16, 256)$.

References

1. Aagaard, M., AlTawy, R., Gong, G., Mandal, K., Rohit, R.: ACE: An Authenticated Encryption and Hash Algorithm. Second Round Submission to NIST Lightweight Cryptography (2019)
2. Andreeva, E., Mennink, B., Preneel, B.: Security Reductions of the Second Round SHA-3 Candidates. In: Burmester, M., Tsudik, G., Magliveras, S.S., Ilic, I. (eds.) Information Security - 13th International Conference, ISC 2010, Boca Raton, FL, USA, October 25-28, 2010, Revised Selected Papers. Lecture Notes in Computer Science, vol. 6531, pp. 39–53. Springer (2010), https://doi.org/10.1007/978-3-642-18178-8_5
3. Aumasson, J., Henzen, L., Meier, W., Naya-Plasencia, M.: Quark: A lightweight hash. In: Mangard, S., Standaert, F. (eds.) Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6225, pp. 1–15. Springer (2010), https://doi.org/10.1007/978-3-642-15031-9_1
4. Bao, Z., Chakraborti, A., Datta, N., Guo, J., Nandi, M., Peyrin, T., Yasuda, K.: PHOTON-Beetle. Final Round Submission to NIST Lightweight Cryptography (2019)
5. Beierle, C., Biryukov, A., Cardoso dos Santos, L., Großschädl, J., Perrin, L., Udovenko, A., Velichkov, V., Wang, Q., Moradi, A., Shahmirzadi, A.: Schwaemm and Esch: Lightweight Authenticated Encryption and Hashing using the Sparkle Permutation Family. Final Round Submission to NIST Lightweight Cryptography (2019)
6. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.: SKINNY-AEAD and SKINNY-Hash v1.1. Second Round Submission to NIST Lightweight Cryptography (2019)
7. Bernstein, D.J.: CubeHash specification (2.B.1). Second Round Submission to NIST SHA-3 Competition (2009)

8. Bernstein, D.J., Kölbl, S., Lucks, S., Massolino, P.M.C., Mendel, F., Nawaz, K., Schneider, T., Schwabe, P., Standaert, F.X., Todo, Y., Viguier, B.: Gimli. Second Round Submission to NIST Lightweight Cryptography (2019)
9. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Sponge functions. Ecrypt Hash Workshop 2007 (May 2007)
10. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the Indifferentiability of the Sponge Construction. In: Smart, N.P. (ed.) Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings. Lecture Notes in Computer Science, vol. 4965, pp. 181–197. Springer (2008), https://doi.org/10.1007/978-3-540-78967-3_11
11. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: The Keccak reference (January 2011)
12. Bogdanov, A., Knezevic, M., Leander, G., Toz, D., Varici, K., Verbauwhede, I.: spongent: A Lightweight Hash Function. In: Preneel, B., Takagi, T. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6917, pp. 312–325. Springer (2011), https://doi.org/10.1007/978-3-642-23951-9_21
13. Coron, J., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård Revisited: How to Construct a Hash Function. In: Shoup, V. (ed.) Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3621, pp. 430–448. Springer (2005), https://doi.org/10.1007/11535218_26
14. Daemen, J., Hoffert, S., Peeters, M., Van Assche, G., Van Keer, R.: Xoodyak, a lightweight cryptographic scheme. Final Round Submission to NIST Lightweight Cryptography (2019)
15. Daemen, J., Massolino, P., Rotella, Y.: The Subterranean 2.0 cipher suite. Second Round Submission to NIST Lightweight Cryptography (2019)
16. Dobraunig, C., Eichlseder, M., Mangard, S., Mendel, F., Mennink, B., Primas, R., Unterluggauer, T.: ISAP v2. Final Round Submission to NIST Lightweight Cryptography (2019)
17. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon v1.2. Final Round Submission to NIST Lightweight Cryptography (2019)
18. Guo, J., Peyrin, T., Poschmann, A.: The PHOTON Family of Lightweight Hash Functions. In: Rogaway, P. (ed.) Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6841, pp. 222–239. Springer (2011), https://doi.org/10.1007/978-3-642-22792-9_13
19. Knudsen, L.R., Rechberger, C., Thomsen, S.S.: The Grindahl Hash Functions. In: Biryukov, A. (ed.) Fast Software Encryption, 14th International Workshop, FSE 2007, Luxembourg, Luxembourg, March 26-28, 2007, Revised Selected Papers. Lecture Notes in Computer Science, vol. 4593, pp. 39–57. Springer (2007), https://doi.org/10.1007/978-3-540-74619-5_3
20. Maurer, U.M., Renner, R., Holenstein, C.: Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In: Naor, M. (ed.) Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings. Lecture Notes in Computer Science, vol. 2951, pp. 21–39. Springer (2004), https://doi.org/10.1007/978-3-540-24638-1_2

21. Naito, Y., Ohta, K.: Improved Indifferentiable Security Analysis of PHOTON. In: Abdalla, M., Prisco, R.D. (eds.) Security and Cryptography for Networks - 9th International Conference, SCN 2014, Amalfi, Italy, September 3-5, 2014. Proceedings. Lecture Notes in Computer Science, vol. 8642, pp. 340–357. Springer (2014), https://doi.org/10.1007/978-3-319-10879-7_20
22. National Institute of Standards and Technology: FIPS PUB 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions (August 2015), http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf
23. NIST: Lightweight Cryptography (February 2019), https://csrc.nist.gov/Projects/Lightweight-Cryptography
24. Rogaway, P., Shrimpton, T.: Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance. In: Roy, B.K., Meier, W. (eds.) Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers. Lecture Notes in Computer Science, vol. 3017, pp. 371–388. Springer (2004), https://doi.org/10.1007/978-3-540-25937-4_24
25. Zhang, W., Ding, T., Yang, B., Bao, Z., Xiang, Z., Ji, F., Zhao, X.: KNOT: Algorithm Specifications and Supporting Document. Second Round Submission to NIST Lightweight Cryptography (2019)