Post-quantum security of the sponge construction

Czajkowski, J.; Groot Bruinderink, L.; Hülsing, A.; Schaffner, C.; Unruh, D.

DOI
10.1007/978-3-319-79063-3_9

Publication date
2018

Document Version
Author accepted manuscript

Published in
Post-Quantum Cryptography

Post-quantum security of the sponge construction

Jan Czajkowski[1], Leon Groot Bruinderink[2], Andreas Hülsing[2], Christian Schaffner[1], Dominique Unruh[3]

[1]QuSoft, Univ of Amsterdam, {j.czajkowski,c.schaffner}@uva.nl

[2]TU Eindhoven, l.groot.bruinderink@tue.nl, andreas@huelsing.net

[3]University of Tartu, unruh@ut.ee

August 2, 2017

Abstract

We investigate the post-quantum security of hash functions based on the sponge construction. A crucial property for hash functions in the post-quantum setting is the collapsing property (a strengthening of collision-resistance). We show that the sponge construction is collapsing (and in consequence quantum collision-resistant) under suitable assumptions about the underlying block function. In particular, if the block function is a random function or a (non-invertible) random permutation, the sponge construction is collapsing.

Contents

1 Introduction

Cryptographic hash functions are one of the central primitives in cryptography. They are used virtually everywhere: As cryptographically secure checksums to verify integrity of software or

This work was supported in part by the Commission of the European Communities through the Horizon 2020 program under project number 645622 PQCRYPTO. CS and JC are supported by a NWO VIDI grant. DU was supported by institutional research funding IUT2-1 of the Estonian Ministry of Education and Research, and by the Estonian Centre of Exellence in IT (EXCITE) funded by the ERDF. Permanent ID of this document: a0bc1a357219db37fa343d9c09ca0613. Date: August 2, 2017

data packages, as building block in security protocols, including TLS, SSH, IPSEC, as part of any efficient variable-input-length signature scheme, to build full-fledged hash-based signature schemes, in transformations for CCA-secure encryption, and many more.

While all widely deployed public-key cryptography is threatened by the rise of quantum computers, hash functions are widely believed to only be mildly effected. The reason for this is twofold. On the one hand, generic quantum attacks achieve at most a square-root speed up compared to their pre-quantum counterparts and can be proven asymptotically optimal [BHT97; Zha15; HRS16]. On the other hand, there do not exist any dedicated quantum attacks on any specific hash function (excluding of course those based on number theory like, e.g., VSH [CLS06]) that perform better than the generic quantum attacks.

One of the most important properties of a hash function H is collision-resistance. That is, it is infeasible to find $x \neq x'$ with $H(x) = H(x')$. Intuitively, collision-resistance guarantees some kind of computational injectivity – given $H(x)$, the value x is effectively determined. Of course, information-theoretically, x is not determined, but in many situations, we can treat the preimage x as unique, because we will never see another value with the same hash. For example, collision-resistant hashes can be used to extend the message space of signature schemes (by signing the hash of the message), or to create commitment schemes (e.g., sending $H(x\|r)$ for random r commits us to x; the sender cannot change his mind about x because he cannot find another preimage).

In the post-quantum setting,[1] however, it was shown by Unruh [Unr16a] that collision-resistance is weaker than expected: For example, the commitment scheme sketched in the previous paragraph is not binding: it is possible for an attacker to send a hash h, then to be given a value x, and then to send a random value r such that $h = H(x\|r)$, thus opening the commitment to any desired value – even if H is collision-resistant against quantum adversaries.[2] This contradicts the intuitive requirement that $H(x)$ determines x.

Fortunately, Unruh [Unr16a] also presented a strengthened security definition for post-quantum secure hash functions: collapsing hash functions. Roughly speaking, a hash function is collapsing if, given a superposition of values m, measuring $H(m)$ has the same effect as measuring m (at least from the point of view of a computationally limited observer). Collapsing hash functions serve as a drop-in replacement for collision-resistant ones in the post-quantum setting: Unruh showed that several natural classical commitment schemes (namely the scheme sketched above, and the statistically-hiding schemes from [HM96]) become post-quantum secure when using a collapsing hash function instead of a collision-resistant one. The collapsing property also directly implies collision-resistance.

In light of these results, it is desirable to find hash functions that are collapsing. Unruh [Unr16a] showed that the random oracle is collapsing. (That is, a hash function $H(x) := \mathcal{O}(x)$ is collapsing when $\mathcal{O}$ is a random oracle.) However, this has little relevance for real-world hash functions: A practical hash function is typically constructed by iteratively applying some elementary building block (e.g., a "compression function") in order to hash large messages. So even if we are willing to model the elementary building block as a random oracle, the overall hash function construction should arguably not be modeled as a random oracle.[3]

For hash functions based on the Merkle-Damgård (MD) construction (such as SHA2 [Nat15]), Unruh [Unr16b] showed: If the compression function is collapsing, so is the hash function resulting

[1] We mean a situation in which the protocols and primitives that are studied are classical, but the attacker can perform quantum computations.

[2] More precisely, [Unr16a] shows that relative to certain oracles, a collision-resistant hash function exists that allows such attacks. In particular, this means that there cannot be a relativizing proof that the commitment scheme is binding assuming a collision-resistant hash function.

[3] For example, hash functions using the Merkle-Damgård construction are not well modeled as a random oracle. If we use $MAC(k, m) := H(k\|m)$ as a message authentication code (MAC) with key k, we have that MAC is secure (unforgeable) when H is a random oracle, but easily broken when H is a hash function built using the Merkle-Damgård construction.

from the MD construction. In particular, if we model the compression function as a random oracle (as is commonly done in the analysis of practical hash functions), we have that hash functions based on the MD construction are collapsing (and thus suitable for use in a post-quantum setting).

However, not all hash functions are constructed using MD. Another popular construction is the sponge construction [Ber+07], underlying for example the current international hash function standard SHA3 [NIS14], but also other hash functions such as Quark [Aum+10], Photon [GPP11], Spongent [Bog+13], and Gluon [Ber+12]. The sponge construction builds a hash function H from a block function[4] $\mathbf{f}$. In the classical setting, we know that the sponge construction is collision-resistant if the block function $\mathbf{f}$ is modeled as a random oracle, or a random permutation, or an invertible random permutation [Ber+08].[5] However, their proof does not carry over to the post-quantum setting: their proof relies on the fact that queries performed by the adversary to the block function are classical (i.e., not in superposition between different values). As first argued in [Bon+11], random oracles and related objects should be modeled as functions that can be queried in superposition of different inputs. (Namely, with a real hash function, an adversary can use a quantum circuit implementing SHA3 and can thereby query the function in superposition. The adversary could evaluate the sponge on the uniform superposition over all messages of a certain length, possibly helping him to, e.g., find a collision.) Thus, we do not know whether the sponge construction (and thus hash functions like SHA3) is collapsing (or at least collision-resistant in the post-quantum setting).

Our contributions. In the present paper we tackle the question whether the sponge construction is collision-resistant and collapsing in the post-quantum setting. We show:

- If the block function $\mathbf{f}$ is collision-resistant when restricted to the left and right half of its output and it is hard to find a zero-preimage of $\mathbf{f}$ (restricted to the right half of its output), then the sponge construction is collision resistant.
- We give a quantum algorithm for finding collisions in any function (given access to a random oracle), in particular in the sponge construction. The number of quantum queries to $\mathbf{f}$ asymptotically matches our bounds for collision resistance.
- If the block function $\mathbf{f}$ is collapsing when restricted to the left and right half of its output, respectively, and if it is hard to find a zero-preimage of $\mathbf{f}$ (restricted to the right half of its output), then the sponge construction is collapsing.
- If the block function $\mathbf{f}$ is a random oracle or a random permutation, then the sponge construction is collapsing.

It should be stressed that we *do not* show that the sponge construction is collapsing (or even collision-resistant) if the block function $\mathbf{f}$ is an *efficiently invertible* random permutation. In this case, it is trivial to find zero-preimages by applying the inverse permutation to 0. This means that the present result cannot be directly used to show the security of, say, SHA3, because SHA3 uses an efficiently invertible permutation as block function. Our results apply to hash functions where the block function is not (efficiently) invertible, e.g., Gluon [Ber+12]. But we believe that our results are also a first step towards understanding the sponge construction for invertible block functions, and towards showing the post-quantum security of SHA3.

Open questions / future work. Beyond the results of this paper, there are a few natural avenues for future research:

- *Efficiently invertible permutations.* Many hash functions (e.g., SHA3) use the sponge construction with an invertible permutation. Our results do not apply to those. How can

[4] It is not called a compression function, since the domain and range of $\mathbf{f}$ are identical.

[5] [Ber+08] shows that the sponge construction is indifferentiable from a random oracle *in the classical setting.* Together with the fact that the random oracle is collision-resistant, collision-resistance of the sponge construction follows.

we prove the collapsing property (or at least quantum collision-resistance) in that setting?

- *Other hash functions.* [Unr16b] has covered hash functions based on the Merkle-Damgård construction, we have covered hash functions based on the sponge construction. Which other common constructions of hash functions are collapsing?

- *Tightness of the results.* Are our concrete security bounds tight? If not, can they be improved? In particular, it would be interesting to improve our analysis of the squeezing phase (i.e., the second half of the sponge construction) since our analysis only takes into account the first output block and thus probably looses a lot in terms of concrete security in the case of hash functions with long output.

- *Other properties of the sponge construction.* Classically, we get many properties of the sponge construction for free from the fact that the sponge construction is indifferentiable from a random oracle [Ber+08]. For example, it immediately follows that the sponge construction is a pseudo-random function (PRF). Can we show those properties in the quantum setting, too? E.g., is the sponge construction a PRF secure against quantum adversaries? Is it secure even against quantum adversaries that have superposition access to the PRF (as in [Zha12])?

Organization. In Section 2 ("Preliminaries") we recall some standard notation and definitions. In Section 3 ("Collapsing hash functions"), we recall the definition of collapsing hash functions and some important properties of that definition. In Section 4 ("The sponge construction") we recall the sponge construction. In Section 5 ("Collision-resistance of the sponge construction") we first show the collision resistance of the sponge construction. Then in Section 7 ("Quantum Attack") we provide a quantum attack that finds collisions in the sponge construction. In Section 6 ("Sponges are collapsing"), we present our main result – that the sponge construction is collapsing. In Section 6.1 ("Using random oracles or random permutations") we specialize this result to the case where the block function is a random function or a random permutation. In Section 6.2 ("Concrete security results"), we state and prove the results from Section 6 with concrete security bounds.

2 Preliminaries

In this section we briefly introduce basic notations and quantum computing as needed for this work.

Basic notations. $x \oplus y$ denotes the bitwise XOR of bitstrings x and y. Let $\operatorname{range} f$ denote the range of the function f, and $\operatorname{im} f$ its image (i.e., $\operatorname{im} f$ contains all values that the function f actually attains, while $\operatorname{range} f$ refers to the set into which f maps according to the declaration of f).

By $\{0,1\}^n$ we denote the set of all bitstrings of length n. By $(\{0,1\}^n)^+$ we denote the set of non-empty bitstrings that consists of blocks of length n. (I.e., bitstrings of length kn for some $k \geq 1$.) $|m|$ denotes the length of a bitstring, but in abuse of notation, for elements $m \in (\{0,1\}^n)^+$, $|m|$ denotes the number of n-bit blocks in m. (But for sets M, $|M|$ is the cardinality of M.)

We call a real-valued function $f \geq 0$ *negligible* iff for any polynomial $p > 0$, we have $f < 1/p$ eventually. We call $f \leq 1$ *overwhelming* if $1 - f$ is negligible.

Quantum computing. We assume the reader is familiar with the usual notations in quantum computation, but we give a very short introduction here. A quantum system A is a complex Hilbert space $\mathcal{H}$, together with an inner product $\langle\cdot|\cdot\rangle$.The state of a quantum system is given by a vector $|\Psi\rangle$ of unit norm ($\langle\Psi|\Psi\rangle = 1$). A joint system of $\mathcal{H}_1$ and $\mathcal{H}_2$ is denoted by $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2$, with elements $|\Psi\rangle = |\Psi_1\rangle|\Psi_2\rangle$ for $|\Psi_1\rangle \in \mathcal{H}_1, |\Psi_2\rangle \in \mathcal{H}_2$. Operations on quantum states are represented by unitary operations $\mathbf{U}$ on the quantum states, or by projective measurements. A unitary transformation $\mathbf{U}$ over a d-dimensional Hilbert space $\mathcal{H}$ is a $d \times d$ matrix $\mathbf{U}$ such that $\mathbf{U}\mathbf{U}^\dagger = \mathbf{I}_d$, where $\mathbf{U}^\dagger$ represents the conjugate transpose. A projective measurement is specified by a family of projectors $\{P_i\}$ that are mutually orthogonal and sum up to 1, one projector P_i for each possibly measurement outcome i. For any quantum state $|\Psi\rangle$, let $\langle\Psi|$ denote the adjoint of $|\Psi\rangle$. In particular, $|\Psi\rangle\langle\Psi|$ denotes the orthogonal projector onto $|\Psi\rangle$. We assume familiarity with these concepts throughout the technical parts of this paper. For an introduction, we refer the reader to a textbook on quantum computation or quantum information such as [NC10].

For algorithms A (classical or quantum), we use the notation $(a, b, c) \leftarrow A(d, e, f)$ to denote that A is executed with inputs d, e, f, and the output triple is assigned to the variables a, b, c. We use capital letters to denote quantum registers, i.e., subsystems of the quantum state of the whole system. E.g., $(y, M') \leftarrow A(x, M)$ means the algorithm A has classical input x, and gets the quantum register M, and it outputs classical output y and a new quantum register M' (the register M is consumed by A). By $x \overset{\$}{\leftarrow} M$ we mean that x is chosen uniformly at random from the set M. If $\mathcal{M}$ denotes a projective measurement, we write $x \leftarrow \mathcal{M}(M)$ to denote that x is assigned the outcome of measuring the register M with $\mathcal{M}$. Note that in this case, the register M is not discarded, instead it contains the post-measurement state. We write $A^{\mathcal{O}}$ when an algorithm A has access to the function $\mathcal{O}$ as an oracle. That is, $A^{\mathcal{O}}$ can evaluate the unitary $\mathbf{U}_{\mathcal{O}} : |x, y\rangle \to |x, y \oplus \mathcal{O}(x)\rangle$ in a single step. Any quantum algorithm making q queries can then be written as a final transformation $\mathbf{U}_q\mathbf{U}_{\mathcal{O}} \dots \mathbf{U}_1\mathbf{U}_{\mathcal{O}}\mathbf{U}_0$ for unitaries $\mathbf{U}_i$ applied between queries and oracle queries $\mathbf{U}_{\mathcal{O}}$.In results with asymptotic statements (referring, e.g., to quantum-polynomial-time adversaries), we assume a security parameter that is implicitly provided to all adversaries, and that all parameters and functions may implicitly depend on.

A game is a sequence of steps (such as $(y, M') \leftarrow A(x, M)$ or $x \overset{\$}{\leftarrow} M$ or $x \leftarrow \mathcal{M}(M)$). We write $\Pr[C : G]$ to denote the probability that the condition C holds after executing the steps in game G. (E.g., $\Pr[b = 1 : b \leftarrow A()]$ denote the probability that the bit b returned by $A()$ is 1.)

Definition 1 (Zero-preimage-resistance) *We call a function $f^{\mathcal{O}} : \{0,1\}^d \to \{0,1\}^e$* zero-preimage-resistant *iff for any quantum-polynomial-time adversary $A^{\mathcal{O}}$, we have that* $\Pr[f^{\mathcal{O}}(x) = 0^e : x \leftarrow A^{\mathcal{O}}()]$ *is negligible.*

We will also make use of the following technical lemma:

Lemma 2 *Let ρ be a quantum state (density operator of trace 1). Let $\mathcal{M}$ be a projective measurement consisting of projectors $P_1, \dots, P_n$. Assume that applying $\mathcal{M}$ to ρ gives outcome 1 with probability $\geq 1 - \varepsilon$.*

Let ρ' be the result of applying $\mathcal{M}$ to ρ (and discarding the result).

Then the trace distance between ρ and ρ' is $\leq \sqrt{\varepsilon}$. (I.e., no algorithm can distinguish those states better than with probability $\sqrt{\varepsilon}$.)

Proof. Without loss of generality, we can assume that ρ is a pure state $\rho = |\Psi\rangle\langle\Psi|$. (The general case of the lemma is then obtained by considering a purification $|\Psi\rangle\langle\Psi|$ of the mixed state ρ.)

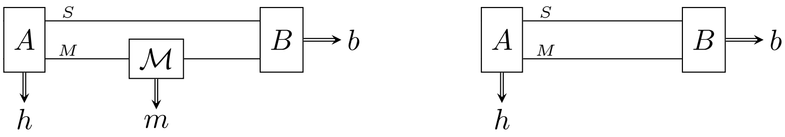
Then $\rho' = \sum_i P_i|\Psi\rangle\langle\Psi|P_i$. Let F denote the fidelity and let TD denote the trace distance. By [NC10, (9.60)], we have $F(|\Psi\rangle\langle\Psi|, \rho') = \sqrt{\langle\Psi|\rho'|\Psi\rangle}$. Thus,

$$F(\rho, \rho')^2 = \langle\Psi|\rho'|\Psi\rangle = \sum_i \langle\Psi|P_i|\Psi\rangle\langle\Psi|P_i|\Psi\rangle = \sum_i \bigl|\langle\Psi|P_i|\Psi\rangle\bigr|^2 \geq \bigl|\langle\Psi|P_1|\Psi\rangle\bigr|^2 \geq 1 - \varepsilon.$$

Thus

$$\mathrm{TD}(\rho, \rho') \overset{(*)}{\leq} \sqrt{1 - F(\rho, \rho')^2} \leq \sqrt{1 - (1 - \varepsilon)} = \sqrt{\varepsilon}.$$

Here $(*)$ uses [NC10, (9.101)]. □



(a) – Game_1 (b) – Game_2

Figure 1: Games from the definition of collapsing hash functions. $\mathcal{M}$ represents a measurement in the computational basis. (A, B) is assumed to satisfy the property that $\mathcal{M}$ always returns m with $H(m) = h$. A function is collapsing if the probability of $b = 1$ is negligibly close in both games.

3 Collapsing hash functions

In this section, we recall the notion of collapsing hash functions H from [Unr16a]. We describe both the underlying intuition, as well as the formal definitions.

A *hash function* is a function $H^{\mathcal{O}} : X \to Y$ for some range X and domain Y. (Typically, Y consists of fixed length bitstrings, and X consists of fixed length bitstrings or $\{0,1\}^*$.) H can depend on an oracle $\mathcal{O}$. (Typically, $\mathcal{O}$ will be a random function, a random permutation, or simply be missing if we are in the standard model. Unless specified otherwise, we make no assumptions about the distribution of $\mathcal{O}$.)

As mentioned in the introduction, intuitively, we wish that $H(m)$ uniquely identifies m in some sense. In the classical setting, this naturally leads to the requirement that it is hard to find $m \neq m'$ with $H(m) = H(m')$. Then we can treat $H(m)$ as if it had only a single preimage (even though, of course, a compressing H will have many preimages, we just cannot find them). In the quantum setting, there is another interpretation of the requirement that $H(m)$ identifies m. Namely, if we are given a register M that contains a superposition of many values m, then measuring $H(m)$ on that register should – intuitively – fully determine m. That is, the effect on the register M should be the same, no matter whether we measure just the hash $H(m)$ or the whole message m. One can see that for any compressing function H, it is impossible that measuring $H(m)$ and m has information-theoretically the same effect on the state.[6] However, what we can hope for is that for a computationally limited adversary, the two situations are indistinguishable. In other words, we require that no quantum-polynomial-time adversary can distinguish whether we measure $H(m)$ or m. This property is then useful in proofs, because we can replace $H(m)$-measurements by m-measurements and vice versa.

We can slightly simplify this condition if we require that the register M already contains a superposition of values m that all have the same hash $H(m)$. In this case, measuring $H(m)$ has no effect on the state, so we can state the requirement as: If M contains a superposition of messages m with the same $H(m) = h$, then no quantum-polynomial-time adversary can distinguish whether we measure M in the computational basis, or whether we do not measure it at all.

Or slightly more formally: We let the adversary A produce a register M and a hash value h (subject to the promise that measuring M would lead to an m with $H(m) = h$). The adversary additionally keeps an internal state in register S. Then we either measure M in the computational basis (Game_1, depicted in Figure 1 (a)), or we do not perform any such measurement (Game_2, depicted in Figure 1 (b)). Finally, we give registers S (the internal state) and M (the potentially measured message register) to the adversary's second part B. We call H *collapsing* if no quantum-polynomial-time (A, B) can distinguish Game_1 and Game_2.

This is formalized by the following definition:

[6] E.g., M could contain $\sum_m 2^{-|m|/2}|m\rangle$. Then measuring $H(m)$ will lead to the state $\sum_{m \text{ s.t. } H(m)=h} \frac{1}{\sqrt{|H^{-1}(h)|}}|m\rangle$ which is almost orthogonal for large $|H^{-1}(h)|$ to the state $|m\rangle$ we get when measuring m.

Definition 3 (Collapsing [Unr16a]) *For algorithms A, B, consider the following games:*

$$\begin{array}{ll} \mathsf{Game}_1: & (S,M,h) \leftarrow A^{\mathcal{O}}(),\ m \leftarrow \mathcal{M}(M),\ b \leftarrow B^{\mathcal{O}}(S,M) \\ \mathsf{Game}_2: & (S,M,h) \leftarrow A^{\mathcal{O}}(), \qquad\qquad\qquad b \leftarrow B^{\mathcal{O}}(S,M) \end{array}$$

Here S, M are quantum registers. $\mathcal{M}(M)$ is a measurement of M in the computational basis.

For a set $\boldsymbol{m}$, we call an adversary (A,B) valid on $\mathbf{m}$ for $H^{\mathcal{O}}$ *iff* $\Pr[H^{\mathcal{O}}(m) = h \wedge m \in \boldsymbol{m}] = 1$ *when we run $(S,M,h) \leftarrow A^{\mathcal{O}}()$ and measure M in the computational basis as m. If we omit "on $\boldsymbol{m}$", we assume $\boldsymbol{m}$ to be the domain of $H^{\mathcal{O}}$.*

A function H is collapsing *(on $\boldsymbol{m}$) iff for any quantum-polynomial-time adversary (A,B) that is valid for $H^{\mathcal{O}}$ (on $\boldsymbol{m}$), $\bigl|\Pr[b=1 : \mathsf{Game}_1] - \Pr[b=1 : \mathsf{Game}_2]\bigr|$ is negligible.*

The definition follows [Unr16a], except that we made the oracle $\mathcal{O}$ explicit (which was implicit in [Unr16a]).

Miscellaneous facts. The following properties of collapsing hash functions will be useful throughout this paper. They are immediate consequences of their concrete-security variants in Section 3.1.

Lemma 4 *If $H^{\mathcal{O}}$ is injective, then $H^{\mathcal{O}}$ is collapsing.*

Theorem 5 *If $\mathcal{O} : \{0,1\}^e \to \{0,1\}^d$ is a random function with superlogarithmic d (in the security parameter), then $H^{\mathcal{O}} := \mathcal{O}$ is collapsing.*

Lemma 6 *If $G^{\mathcal{O}} \circ H^{\mathcal{O}}$ is collapsing, and $G^{\mathcal{O}}$ is quantum-polynomial-time computable, then $H^{\mathcal{O}}$ is collapsing.*

Lemma 7 *If $G^{\mathcal{O}}$ and $H^{\mathcal{O}}$ are collapsing, and $H^{\mathcal{O}}$ is quantum-polynomial-time computable, then $G^{\mathcal{O}} \circ H^{\mathcal{O}}$ is collapsing.*

3.1 Definitions for concrete security

Definition 3 allows us to state the results of this paper in asymptotic terms (namely, that the sponge construction is collapsing). However, when stating concrete security results, one can achieve tighter results by directly analyzing the security of t parallel evaluations of the hash function (see [Unr16b]). This leads to the following definition:

Definition 8 (Collapsing – concrete security) *For algorithms A, B, and an integer t, consider the following games:*

$$\begin{array}{ll} \mathsf{Game}_1: & (S, M_1, \dots, M_t, h_1, \dots, h_t) \leftarrow A^{\mathcal{O}}(), \\ & m_1 \leftarrow \mathcal{M}(M_1),\ \dots,\ m_t \leftarrow \mathcal{M}(M_t), \\ & b \leftarrow B^{\mathcal{O}}(S, M_1, \dots, M_t) \\ \mathsf{Game}_2: & (S, M_1, \dots, M_t, h_1, \dots, h_t) \leftarrow A^{\mathcal{O}}(), \\ & b \leftarrow B^{\mathcal{O}}(S, M_1, \dots, M_t) \end{array}$$

Here $S, M_1, \dots, M_t$ are quantum registers. $\mathcal{M}(M_i)$ is a measurement of M_i in the computational basis.

For a set $\boldsymbol{m}$, we call an adversary (A,B) t-valid on $\mathbf{m}$ for $H^{\mathcal{O}}$ *iff* $\Pr[\forall i.\ H^{\mathcal{O}}(m_i) = h_i \wedge m_i \in \boldsymbol{m}] = 1$ *when we run $(S, M_1, \dots, M_t, h_1, \dots, h_t) \leftarrow A^{\mathcal{O}}()$ and measure all M_i in the computational basis as m_i. If we omit "on $\boldsymbol{m}$", we assume $\boldsymbol{m}$ to be the domain of $H^{\mathcal{O}}$.*

We call $\bigl|\Pr[b=1 : \mathsf{Game}_1] - \Pr[b=1 : \mathsf{Game}_2]\bigr|$ the collapsing-advantage *of (A,B) against H.*

This definition is from [Unr16b], with the only difference that now adversaries and hash functions may depend on an oracle $\mathcal{O}$, instead of depending on a public parameter.

The two definitions of collapsing hash functions are equivalent in the following sense:

Lemma 9 ([Unr16b]) *$H^{\mathcal{O}}$ is collapsing (according to Definition 3) on $\boldsymbol{m}$ iff for any quantum-polynomial-time $(A^{\mathcal{O}}, B^{\mathcal{O}})$ that is t-valid on $\boldsymbol{m}$, the collapsing-advantage (according to Definition 8) is negligible.*

Miscellaneous facts. We restate the facts from Section 3, with concrete security bounds. Unless specified otherwise, these facts were proven in [Unr16b] (in a setting without oracle $\mathcal{O}$, but all proofs from [Unr16b] relativize).

All results in this paper hold both if runtime is measured in computation steps, and when time is measured in the number oracle queries.

Lemma 10 *If $H^{\mathcal{O}}$ is injective, and $(A^{\mathcal{O}}, B^{\mathcal{O}})$ is a t-valid adversary with collapsing-advantage ε against $H^{\mathcal{O}}$, then $\varepsilon = 0$.*

Theorem 11 *If $\mathcal{O} : \{0,1\}^e \to \{0,1\}^d$ is a random function, and $(A^{\mathcal{O}}, B^{\mathcal{O}})$ is t-valid for $\mathcal{O}$ and has collapsing-advantage ε, then $\varepsilon \in O(tq^{3/2}2^{-d/2})$.*

This was shown for the case $t = 1$ in [Unr16a]. For general t, this theorem then follows by using the fact that the collapsing property parallel composes (fully analogous to the parallel composition of collapse-binding commitments shown in [Unr16a]).

Lemma 12 *Fix oracle functions $G^{\mathcal{O}}$ and $H^{\mathcal{O}}$. Let (A, B) be a t-valid adversary against some function $H^{\mathcal{O}}$ with runtime τ and collapsing-advantage ε against $H^{\mathcal{O}}$.*

Then there is an adversary (A', B') that is t-valid for $G^{\mathcal{O}} \circ H^{\mathcal{O}}$, has runtime $\tau + t\tau_G$, and collapsing-advantage ε against $G^{\mathcal{O}} \circ H^{\mathcal{O}}$.

Here τ_G is the time required for computing $G^{\mathcal{O}}$.

Proof. Let A' perform the following steps: Run $(S, M_1, \dots, M_t, h_1, \dots, h_t) \leftarrow A^{\mathcal{O}}()$, and return $\big(S, M_1, \dots, M_t, G^{\mathcal{O}}(h_1), \dots, G^{\mathcal{O}}(h_t)\big)$.

Let B' be identical to B.

Since (A, B) is is t-valid for $H^{\mathcal{O}}$, the t-validity of (A', B') for $G^{\mathcal{O}} \circ H^{\mathcal{O}}$ follows directly from the construction of (A', B') and the definition of t-validity.

The runtime of (A', B') is $\tau + t\tau_G$, since there are t additional calls to $G^{\mathcal{O}}$ compared with (A, B).

Since (A', B') differs from (A, B) only in the classical outputs $h_1, \dots, h_t$, and since the games Game_1 and Game_2 from Definition 8 do not use these classical outputs, we have that $\Pr[b = 1 : \mathsf{Game}_1 \text{ using } A, B] = \Pr[b = 1 : \mathsf{Game}_1 \text{ using } A', B']$ and $\Pr[b = 1 : \mathsf{Game}_2 \text{ using } A, B] = \Pr[b = 1 : \mathsf{Game}_2 \text{ using } A', B']$. Thus (A', B') has collapsing-advantage

$$\begin{aligned}&\Big|\Pr[b = 1 : \mathsf{Game}_1 \text{ using } A', B'] - \Pr[b = 1 : \mathsf{Game}_2 \text{ using } A', B']\Big| \\ &\quad = \Big|\Pr[b = 1 : \mathsf{Game}_1 \text{ using } A, B] - \Pr[b = 1 : \mathsf{Game}_2 \text{ using } A, B]\Big| = \varepsilon. \qquad \square\end{aligned}$$

Lemma 13 *Fix oracle functions $G^{\mathcal{O}}$ and $H^{\mathcal{O}}$. If there is a τ-time adversary (A, B), t-valid for $G^{\mathcal{O}} \circ H^{\mathcal{O}}$, with collapsing-advantage ε against $G^{\mathcal{O}} \circ H^{\mathcal{O}}$, then there are:*

- *a $(\tau + O(t\tau_H))$-time adversary (A', B'), t-valid for $G^{\mathcal{O}}$ on $\operatorname{im} H^{\mathcal{O}}$, with some collapsing-advantage ε' against $G^{\mathcal{O}}$,*

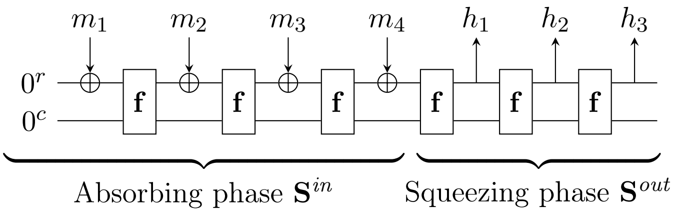

m_1
m_2
m_3
m_4
h_1
h_2
h_3
0^r
0^c
f
f
f
f
f
f
Absorbing phase $\mathbf{S}^{in}$
Squeezing phase $\mathbf{S}^{out}$

Figure 2: The sponge construction $\mathbf{S}$ with a four block input $m_1\|m_2\|m_3\|m_4$ and a three block output $h_1\|h_2\|h_3$. The application of the padding function is not depicted (we assume $m_1\|m_2\|m_3\|m_4 = pad(m)$).

- *a $(\tau + O(t\tau_H))$-time adversary (A'', B''), t-valid for $H^{\mathcal{O}}$, with some collapsing-advantage ε'' against $H^{\mathcal{O}}$.*

such that $\varepsilon \leq \varepsilon' + \varepsilon''$.

Here τ_H is an upper bound on the time for evaluating $H^{\mathcal{O}}$ (on the messages that A outputs on the registers M_j).

In [Unr16b], this lemma had an additional $O(t\ell_{mid})$ in the runtime of (A'', B'') where ℓ_{mid} denotes the length of the output of $H^{\mathcal{O}}$. Since this is always dominated by $O(t\tau_H)$, we omit this term here.

4 The sponge construction

In this section, we review the sponge construction introduced by [Ber+07]. The sponge construction has two internal parameters r and c called the *rate* and the *capacity*, respectively. The internal state has $r + c$ bits. We refer to the first part of the state as the left state, and to the second part of the state as the right state. Underlying the sponge construction is a block function $\mathbf{f}$ that inputs and outputs $r + c$ bits. To hash a message m, the message is first padded to a non-zero multiple of the rate r. That is, we use some injective *padding function pad* to get $k \geq 1$ message blocks $m_1\|\dots\|m_k = pad(m)$.[7] Then we XOR m_1 to the left state, apply $\mathbf{f}$ to the (whole) state, XOR m_2 to the left state, apply $\mathbf{f}$ to the state, …, apply $\mathbf{f}$ to the state, XOR m_k to the left state. The steps performed so far are referred to as the *absorbing phase* (denoted in this paper with $\mathbf{S}^{in}$). Now we start with the *squeezing phase* $\mathbf{S}^{out}$: We apply $\mathbf{f}$ to the state, read the left state as h_1, apply $\mathbf{f}$ to the state, read the left state as h_2, …. We continue to do so until $h_1\|h_2\|\dots$ contains $\geq n$ bits (where n is a parameter specifying the desired output length), and return the first n bits of $h_1\|h_2\|\dots$. The whole process described here (padding, absorbing phase, squeezing phase) is the sponge construction, referred to as $\mathbf{S}$ in this paper. Note that the use of the terms absorbing and squeezing phase in this paper slightly differ from the description in [Ber+07]: In this paper, we end the absorbing phase just before the last application of $\mathbf{f}$, whilst the original sponge paper includes that application of $\mathbf{f}$ in the absorbing phase. The separation we use helps to simplify the proofs in later sections. The resulting sponge construction is the same as in [Ber+07], though. The sponge construction is illustrated in Figure 2 for the special case of $k = 4$ and $n = 3r$ (four input blocks and three output blocks). The following definition makes the above explanation precise:

Definition 14 (Sponge construction) *Fix integers $c > 0$ (the* capacity*) and $r > 0$ (the* rate*), and $n > 0$ (the* output length*). Fix $\mathbf{f} : \{0,1\}^{r+c} \to \{0,1\}^{r+c}$ (the* block function*) and $pad : \{0,1\}^* \to (\{0,1\}^r)^+$.*

[7] The original construction requires that the last block of $pad(m)$ is non-zero, this is important for other properties than collision-resistance/collapsing. In this work, we do not put any such requirement on pad. We do, however, assume that pad outputs at least one block.

For $m_1, \dots, m_k \in \{0,1\}^r$, let

$$\mathbf{S}^{in}_{c,r,\mathbf{f}}(m_1\|\dots\|m_k) := \mathbf{f}\big(\mathbf{S}^{in}_{c,r,\mathbf{f}}(m_1\|\dots\|m_{k-1})\big) \oplus (m_k\|0^c)$$
$$\mathbf{S}^{in}_{c,r,\mathbf{f}}(m_1) := m_1\|0^c$$

(We call $\mathbf{S}^{in}$ the absorbing phase.*)*

For $s \in \{0,1\}^{r+c}$, let

$$\mathbf{S}^{out}_{c,r,\mathbf{f},n}(s) = \begin{cases} s'\|\mathbf{S}^{out}_{c,r,\mathbf{f},n-|s'|}(\mathbf{f}(s)) & (n>0) \\ \textit{empty word} & (n=0) \end{cases}$$

where s' consists of the first $\min\{n,r\}$ bits of $\mathbf{f}(s)$. (We call $\mathbf{S}^{out}$ the squeezing phase.*)*

Let $\mathbf{S}_{c,r,\mathbf{f},pad,n} := \mathbf{S}^{out}_{c,r,\mathbf{f},n} \circ \mathbf{S}^{in}_{c,r,\mathbf{f}} \circ pad$. We call $\mathbf{S}_{c,r,\mathbf{f},pad,n}$ the sponge construction*.*

Usually, $c, r, \mathbf{f}, pad, n$ will be clear from the context. Then we omit them and simply write $\mathbf{S}^{in}, \mathbf{S}^{out}$, and $\mathbf{S}$.

Notation: The sponge construction operates on a state of size $r+c$, and we will often need to refer to the two halves of that state separately: For any $s \in \{0,1\}^{r+c}$, let s^{left} denote the first r bits of s, and let s^{right} refer to the last c bits of s. If f is a function with $r+c$ bit output, then we write f^{left} for the function defined by $f^{\mathsf{left}}(x) := f(x)^{\mathsf{left}}$. And f^{right} analogously.

As the output of the sponge function can be smaller than the rate, i.e. $n \leq r$, we also define the function $\mathbf{f}^{\mathsf{left}/n} : \{0,1\}^{r+c} \to \{0,1\}^{\min(n,r)}$, which is the function that outputs the first $\min(n,r)$ bits of $\mathbf{f}$. In particular, $\mathbf{f}^{\mathsf{left}/n} := \mathbf{f}^{\mathsf{left}}$ for $n \geq r$.

In [Ber+08], it was shown that the sponge construction is indifferentiable from a random oracle, assuming that $\mathbf{f}$ is a random oracle or an (invertible) random permutation. From this collision-resistance follows. However, the proof from [Ber+08] works only in the classical case. If the adversary has superposition access to $\mathbf{f}$, their proof breaks down because it needs to track on which inputs $\mathbf{f}$ has been queried. As far as we know, no results concerning the post-quantum security of the sponge construction are known (besides what we show in this paper).

5 Collision-resistance of the sponge construction

In this section we state our result concerning collision-resistance of the sponge construction. We motivate our statement with Lemma 16 connecting attacks on some features of the block function with collision-resistance of the overall construction. Those features are collision-resistance of $\mathbf{f}^{\mathsf{right}}$, collision resistance of $\mathbf{f}^{\mathsf{left}/n}$, and zero-preimage-resistance of $\mathbf{f}^{\mathsf{right}}$ (for details of zero-preimage-resistance please refer to Definition 1). Let us state the main result of this section.

Theorem 15 *Assume that $\mathbf{f}^{\mathsf{right}}$ and $\mathbf{f}^{\mathsf{left}/n}$ are collision resistant and $\mathbf{f}^{\mathsf{right}}$ is zero-preimage resistant. Then $\mathbf{S}_{c,r,\mathbf{f},pad,n}$ is collision-resistant.*

Proof sketch. We prove this theorem by a reduction to adversaries attacking the block function. Namely finding collisions in $\mathbf{f}^{\mathsf{right}}$ or $\mathbf{f}^{\mathsf{left}/n}$, or a zero-preimage under $\mathbf{f}^{\mathsf{right}}$. This reduction is presented in Lemma 16. Knowing that every collision in $\mathbf{S}$ results in breach in the security of $\mathbf{f}^{\mathsf{right}}$ or $\mathbf{f}^{\mathsf{left}/n}$, allows us to state the claim of the theorem. □

Let us present the lemma relating the output of a sponge-collision-finder with collisions and pre-images under $\mathbf{f}$.

Lemma 16 *Assume that pad is injective. There is a deterministic polynomial-time oracle algorithm A such that for any $m \neq \hat{m}$ with $\mathbf{S}(m) = \mathbf{S}(\hat{m})$, $A^{\mathbf{f}}(m, \hat{m})$, outputs one of the following:*

- $(\mathsf{right}, (s, \hat{s}))$ *where* $(s, \hat{s})$ *is a collision of* $\mathbf{f}^{\mathsf{right}}$*,*
- (zero, s) *where* s *is a zero-preimage of* $\mathbf{f}^{\mathsf{right}}$*,*
- *or* $(\mathsf{left}, (s, \hat{s}))$ *where* $(s, \hat{s})$ *is a collision of* $\mathbf{f}^{\mathsf{left}/n}$*.*

The runtime of the algorithm is at most $2\tau_{pad} + O(T_m \tau_{\mathbf{f}})$*, where* T_m *denotes the bound on the number of blocks in the padded input messages,* $\tau_{\mathbf{f}}$ *is the time required for a single classical invocation of* $\mathbf{f}$ *and* τ_{pad} *is the time of computing pad.*

Proof. A starts by computing the first right-state of the squeezing phase on input of the two colliding messages, i.e., it evaluates $\mathbf{f} \circ \mathbf{S}^{in} \circ pad$. We will denote the states traversed during this calculation by s_i and $\hat{s}_i$ for m and $\hat{m}$, respectively. As our analysis starts with the final state of this computation and revisits the intermediate states in backwards direction, we denote by s_0 the final state, whose left part is output (for $n < r$ only the first n bits), by s_{-1} the state just before the last application of $\mathbf{f}$ and so on. A figure including this notation is presented later in Figure 3. Using $p := pad(m)$ and $\hat{p} := pad(\hat{m})$, the intermediate states s_{-i} for $1 \leq i \leq |p| - 1$ are defined by $s_{-i} := \mathbf{f}(s_{-i-1}) \oplus p_{|p|+1-i} \| 0^c$, $s_0 := \mathbf{f}(s_{-1})$ and $s_{-|p|} := p_1 \| 0^c$. As m and $\hat{m}$ collide per assumption, we have $s_0^{\mathsf{left}/n} = \hat{s}_0^{\mathsf{left}/n}$.

1. Algorithm A first checks if s_{-1} or $\hat{s}_{-1}$ are a preimage of 0^c, or form a collision under $\mathbf{f}^{\mathsf{left}/n}$. If the right part of s_0 (or $\hat{s}_0$) is 0^c, s_{-1} ($\hat{s}_{-1}$) is a pre-image of 0^c under f^{right} and A outputs (zero, s_{-1}) ($(\mathsf{zero}, \hat{s}_{-1})$, respectively). If $s_{-1} \neq \hat{s}_{-1}$, A outputs $(\mathsf{left}, s_{-1}, \hat{s}_{-1})$. These two states form a collision under $\mathbf{f}^{\mathsf{left}/n}$ because they are the inputs to the last $\mathbf{f}$ in $\mathbf{S}$ and $s_0^{\mathsf{left}/n} = \hat{s}_0^{\mathsf{left}/n}$. Otherwise, $s_{-1} = \hat{s}_{-1}$ and there are no preimages of zero.

2. If not done yet, $s_{-1} = \hat{s}_{-1}$ and A checks for a preimage of 0^c or a collision in $\mathbf{f}^{\mathsf{right}}$. If $s_{-1}^{\mathsf{right}} = 0^c$, A found a preimage of 0^c. This is true as if both messages ended here then $s_{-1} = \hat{s}_{-1}$ would imply that $p = \hat{p}$ (and so $m = \hat{m}$) which contradicts the assumptions of the lemma. Hence, at least one message must be longer. Assuming the longer message is m, A outputs (zero, s_{-2}) (or $(\mathsf{zero}, \hat{s}_{-2})$ if it was $\hat{m}$).

   Next the algorithm checks if $p_{-1} = \hat{p}_{-1}$, where we follow a similar notation for message blocks as for the states. The last block of the input is denoted by p_{-1}. If $p_{-1} \neq \hat{p}_{-1}$, A outputs $(\mathsf{right}, s_{-2}, \hat{s}_{-2})$. This is a collision of $\mathbf{f}^{\mathsf{right}}$ because $p_{-1} \neq \hat{p}_{-1}$ but $s_{-1} = \hat{s}_{-1}$. Thus $\mathbf{f}(s_{-2}) \neq \mathbf{f}(\hat{s}_{-2})$ which in turn implies $s_{-2} \neq \hat{s}_{-2}$ while $\mathbf{f}^{\mathsf{right}}(s_{-2}) = \mathbf{f}^{\mathsf{right}}(\hat{s}_{-2})$. We can be certain that there are at least two applications of $\mathbf{f}$ both in $\mathbf{S}(m)$ and $\mathbf{S}(\hat{m})$ because the right half of $s_{-1} = \hat{s}_{-1}$ is not 0^c.

3. If $p_{-1} = \hat{p}_{-1}$ we end up in the same situation as before but now for $i = 2$. Namely we have that $s_{-2} = \hat{s}_{-2}$ and the algorithm performs the same checks as before but for a bigger i. Repeat Step 2 for all $2 \leq i \leq \min\{|p|, |\hat{p}|\}$.

If the iteration ends without success, this especially means that no collision was found but at least one message was fully processed. In this case A outputs a preimage of 0^c under $\mathbf{f}^{\mathsf{right}}$. That is because no collisions means that all compared message blocks are the same but the two messages are different per assumption. Hence, they must have different lengths. With different length messages that traverse the same state values at the point of $i = \min\{|p|, |\hat{p}|\}$ the right part of both states is 0^c, so the algorithm will output $(\mathsf{zero}, \hat{s}_{-|p|-1})$ (assuming $|p| < |\hat{p}|$).

□

Here we present the concrete upper bound on the probability of success of any quantum adversary finding a collision in $\mathbf{S}$.

Theorem 17 *Assume a quantum* τ*-time adversary* B *that finds a collision in* $\mathbf{S}_{c,r,\mathbf{f},pad,n}$ *with probability* ε*. Then there exist:*

- *a* $(\tau + 2\tau_{pad} + O((4T_m + 2\lceil \frac{n}{r} \rceil)\tau_{\mathbf{f}}))$*-time adversary* A_1 *that finds a collision in* $\mathbf{f}^{\mathsf{right}}$ *with probability* ε_c*,*
- *a* $(\tau + 2\tau_{pad} + O((4T_m + 2\lceil \frac{n}{r} \rceil)\tau_{\mathbf{f}}))$*-time adversary* A_2 *that finds a pre-image of* 0^c *under* $\mathbf{f}^{\mathsf{right}}$ *with probability* ε_0*,*
- *and a* $(\tau + 2\tau_{pad} + O((4T_m + 2\lceil \frac{n}{r} \rceil)\tau_{\mathbf{f}}))$*-time adversary* A_3 *that finds a collision in* $\mathbf{f}^{\mathsf{left}/n}$ *with probability* ε_n*,*

such that $\varepsilon \leq \varepsilon_c + \varepsilon_0 + \varepsilon_n$*.* T_m *denotes the bound on the number of blocks in the padding of the input messages,* $\tau_{\mathbf{f}}$ *is the time required for a single classical invocation of* $\mathbf{f}$ *and* τ_{pad} *is the time required for one invocation of pad.*

Proof. We reduce the problem of finding collisions in $\mathbf{S}$ to attacks on the block function. Adversaries A_1, A_2, and A_3 run B and then classically compute $\mathbf{S}$ on the outputs of the collision finder. If that in fact is a collision they run algorithm A from Lemma 16 and output the last register of its output. Otherwise the algorithms output $\perp$. Note that the runtime of the described adversaries agrees with the claim. That allows us to write

$$\begin{aligned}
&\Pr[m \neq \hat{m} \wedge \mathbf{S}(m) = \mathbf{S}(\hat{m}) : (m, \hat{m}) \leftarrow B] = \\
&\quad \Pr\Big[\Big(s \neq \hat{s} \wedge \mathbf{f}^{\mathsf{right}}(s) = \mathbf{f}^{\mathsf{right}}(\hat{s}) : (s, \hat{s}) \leftarrow A_1\Big) \\
&\quad \vee \Big(s \in \mathbf{f}^{\mathsf{right}-1}(0^c) : s \leftarrow A_2\Big) \vee \Big(s \neq \hat{s} \wedge \mathbf{f}^{\mathsf{left}/n}(s) = \mathbf{f}^{\mathsf{left}/n}(\hat{s}) : (s, \hat{s}) \leftarrow A_3\Big)\Big] \\
&\quad \leq \varepsilon_c + \varepsilon_0 + \varepsilon_n, \qquad (1)
\end{aligned}$$

where the inequality comes from the union bound. □

Note that the same bounds hold when measuring the time in number of oracle queries. It is true that for $n > r$ our bound seems to be not optimal but our reductionist approach is not well suited to deal with consecutive applications of $\mathbf{f}$. For that reason we leave this issue as an open problem to be tackled later.

5.1 Random sponges

Let us now analyse the case of a random sponge. The success probability of a generic collision-finding algorithm in $\mathbf{h} : \{0,1\}^{r+c} \to \{0,1\}^c$ is $O\left(\frac{q^3}{2^c}\right)$, as proved by Zhandry [Zha15]. To use Zhandry's results we need to make sure the distribution of $\mathbf{f}^{\mathsf{right}}$ is in fact uniform.

Lemma 18 *If* $\mathbf{f} : \{0,1\}^{r+c} \to \{0,1\}^{r+c}$ *is a random function and* A *that looks for collisions in* $\mathbf{f}^{\mathsf{right}}$ *makes at most* q *queries to* $\mathbf{f}$ *and has success probability* ε_c*, then* $\varepsilon_c \in O(\frac{q^3}{2^c})$*.*

Proof. Let $\mathbf{h} : \{0,1\}^{r+c} \to \{0,1\}^c$ be a random function. Let A' perform the following steps: It picks a random function $\mathbf{g} : \{0,1\}^{r+c} \to \{0,1\}^r$ (we do not care about the runtime of A', so it is not a problem that picking $\mathbf{g}$ takes exponential time). Then it simulates A where $\mathbf{f}'$ is the function defined by $\mathbf{f}'(x) := \mathbf{g}(x) \| \mathbf{h}(x)$. Note that an oracle query to $\mathbf{f}'$ can be implemented using a single oracle query to $\mathbf{h}$. So A' still performs q oracle queries.

The joint distribution of $\mathbf{f}'$ and $\mathbf{h}$ is the same as that of $\mathbf{f}$ and $\mathbf{f}^{\mathsf{right}}$. Thus the advantage of A' against $\mathbf{h}$ is the same as the advantage ε_c of A against $\mathbf{f}^{\mathsf{right}}$.

Thus we can use results proven in [Zha15], $\varepsilon_c \in O(\frac{q^3}{2^c})$. □

Similarly for $\mathbf{f}^{\mathsf{left}/n}$ and $\mathbf{f}^{\mathsf{left}}$ it holds that the advantage of a q-query collision finder is $\varepsilon_n \in O\left(\frac{q^3}{2^n}\right)$ and $\varepsilon_r \in O\left(\frac{q^3}{2^r}\right)$, respectively. Zero-pre-image finding in a random function has probability of success at most $\varepsilon_0 \in O\left(\frac{q^2}{2^c}\right)$ as shown next.

Lemma 19 *Let* $\mathbf{f} : \{0,1\}^{r+c} \to \{0,1\}^{r+c}$ *be a random function. Then a q-query adversary finds a zero-preimage of* $\mathbf{f}^{\mathsf{right}}$ *with probability* $\leq (q+1)^2 2^{-c+5}$.

Proof. Assume an algorithm $A^{\mathbf{f}}$ that finds a zero-preimage of $\mathbf{f}^{\mathsf{right}}$ with some probability ε. Let $F : \{0,1\}^{r+c} \to \{0,1\}$ be a random function where each $F(z)$ is independently chosen, with $\Pr[F(z) = 1] = \gamma := 2^{-c}$. Let B^F be the following algorithm: It picks functions $\mathbf{f}_1, \mathbf{f}_0 : \{0,1\}^{r+c} \to \{0,1\}^{r+c}$, where each $\mathbf{f}_0(x)$ is independently and uniformly chosen from $\{0,1\}^r \times (\{0,1\}^c \setminus \{0^c\})$ and each $\mathbf{f}_1(x)$ is independently and uniformly chosen from $\{0,1\}^r \times \{0^c\}$. Define $\hat{\mathbf{f}}(x) := \mathbf{f}_{F(x)}(x)$. Then B^F runs $x \leftarrow A^{\hat{\mathbf{f}}}$ and returns x. A query to $\hat{\mathbf{f}}$ can be implemented using 2 queries to F. Thus B^F performs $\leq 2q$ queries to F. (We do not care about the runtime of B^F. Thus the exponential time required for implementing $\mathbf{f}_0, \mathbf{f}_1$ is not a problem.)

Note that $\hat{\mathbf{f}} : \{0,1\}^{r+c} \to \{0,1\}^{r+c}$ is a uniformly random function (for F distributed as described above). Thus $A^{\hat{\mathbf{f}}}$ finds a zero-preimage of $\mathbf{f}^{\mathsf{right}}$ with probability ε, i.e., an x with $\mathbf{f}(x)^{\mathsf{right}} = 0^c$. Such an x then satisfies $F(x) = 1$. Thus B^F finds a 1-preimage of F with probability ε. [HRS16, Theorem 1] shows that a $2q$-query adversary can find a 1-preimage in F with probability at most $8\gamma(2q+1)^2$. Thus $\varepsilon \leq 8\gamma(2q+1)^2 \leq (q+1)^2 2^{-c+5}$. □

We are now ready to bound the probability to find collisions in a random $\mathbf{S}$.

Corollary 20 *For any quantum adversary B finding collisions in a random* $\mathbf{S}_{c,r,\mathbf{f},pad,n}$, *we have that*

$$\Pr[m \neq \hat{m} \wedge \mathbf{S}(m) = \mathbf{S}(\hat{m}) : (m,\hat{m}) \leftarrow B^{\mathbf{f}}] \leq O\left(q^3 \max\{2^{-c}, 2^{-r}, 2^{-n}\}\right). \tag{2}$$

The fact that we have a 2^{-r} term in the above bound is a result of restricting $\mathbf{f}^{\mathsf{left}/n}$ to $\mathbf{f}^{\mathsf{left}}$ in the case of $n > r$.

6 Sponges are collapsing

In this section, we show that the sponge constrution is collapsing, under certain assumptions about the block function $\mathbf{f}$. We only state the qualitative results here, more precise statements with concrete security bounds will be given in Section 6.2.

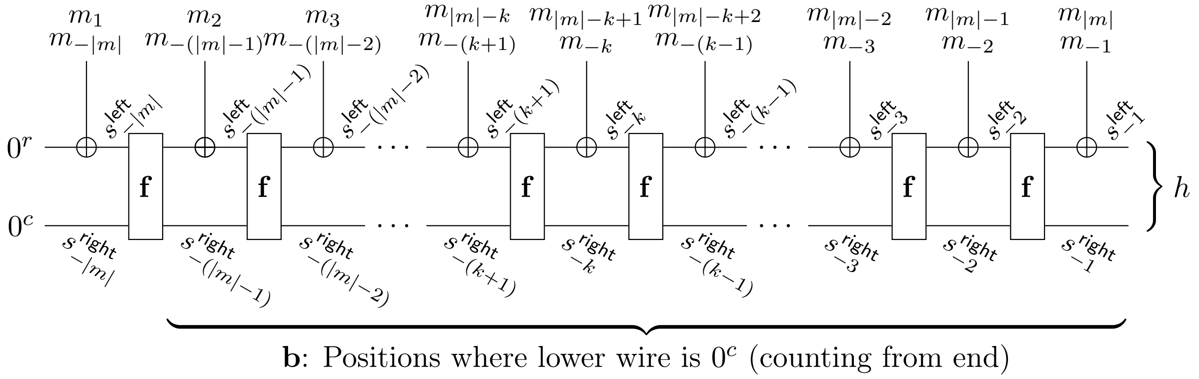
The results in this section hold for all distributions of the oracle $\mathcal{O}$ (including the case that there is no oracle $\mathcal{O}$). The specific cases of random functions and random permutations are covered in Section 6.1. Since all adversaries $(A, B, A', B', \dots)$ and the block function $\mathbf{f}$ have oracle access to $\mathcal{O}$ throughout the section, we omit the oracle $\mathcal{O}$ from our notation for increased readability (i.e., we write $A, \mathbf{f}$ instead of $A^{\mathcal{O}}, \mathbf{f}^{\mathcal{O}}$). Throughout this section, we assume that $\mathbf{f}^{\mathcal{O}}$ can be computed in quantum-polynomial-time (given oracle access to $\mathcal{O}$).

We will analyze the sponge construction in three parts. First, we analyze the security of the absorbing phase $\mathbf{S}^{in}$, then we analyze the security of the squeezing phase $\mathbf{S}^{out}$, and finally we conclude security of the whole sponge $\mathbf{S}$, consisting of padding, absorbing, and squeezing.

First, we analyze the absorbing phase. For the absorbing phase (without padding or squeezing) to be collapsing, we will need two properties of $\mathbf{f}^{\mathsf{right}}$:

- $\mathbf{f}^{\mathsf{right}}$ is collapsing. This is the main property required from the block function $\mathbf{f}$. If we would restrict $\mathbf{S}^{in}$ to fixed length messages, then we could show the collapsing property of $\mathbf{S}^{in}$ based on that property alone.
- $\mathbf{f}^{\mathsf{right}}$ is zero-preimage-resistant (see Definition 1). To see why we need this property, consider a block function $\mathbf{f}$ where the adversary can find, e.g., $x, y \in \{0,1\}^r$ with $f(x\|0^c) = y\|0^c$. Then we can see that $\mathbf{S}^{in}(x\|y) = 0^{c+r}$, and thus $\mathbf{S}^{in}(x\|y\|z) = z\|0^c = \mathbf{S}^{in}(z)$ for any $z \in \{0,1\}^r$. Thus $\mathbf{S}^{in}$ would not be collision-resistant, and in particular not collapsing.

We state the result formally:



m_1
$m_{-|m|}$
m_2
$m_{-(|m|-1)}$
m_3
$m_{-(|m|-2)}$
$m_{|m|-k}$
$m_{-(k+1)}$
$m_{|m|-k+1}$
m_{-k}
$m_{|m|-k+2}$
$m_{-(k-1)}$
$m_{|m|-2}$
m_{-3}
$m_{|m|-1}$
m_{-2}
$m_{|m|}$
m_{-1}
$s^{\mathsf{left}}_{-|m|}$
$s^{\mathsf{left}}_{-(|m|-1)}$
$s^{\mathsf{left}}_{-(|m|-2)}$
$s^{\mathsf{left}}_{-(k+1)}$
s^{left}_{-k}
$s^{\mathsf{left}}_{-(k-1)}$
s^{left}_{-3}
s^{left}_{-2}
s^{left}_{-1}
0^r
0^c
$\mathbf{f}$
h
$s^{\mathsf{right}}_{-|m|}$
$s^{\mathsf{right}}_{-(|m|-1)}$
$s^{\mathsf{right}}_{-(|m|-2)}$
$s^{\mathsf{right}}_{-(k+1)}$
s^{right}_{-k}
$s^{\mathsf{right}}_{-(k-1)}$
s^{right}_{-3}
s^{right}_{-2}
s^{right}_{-1}
$\mathbf{b}$: Positions where lower wire is 0^c (counting from end)

Figure 3: Values occurring in the computation of $h = \mathbf{S}^{in}(m)$, using the notation from the proof sketch of Lemma 21.

Lemma 21 (Absorbing phase is collapsing) *Assume that* $\mathbf{f}^{\mathsf{right}}$ *is collapsing, and that* $\mathbf{f}^{\mathsf{right}}$ *is zero-preimage-resistant. Then* $\mathbf{S}^{in}$ *is collapsing.*

Note that this lemma does not explicitly state anything about the size of r and c. But of course, $\mathbf{f}^{\mathsf{right}}$ can only be collapsing and zero-preimage-resistant is the capacity c is superlogarithmic.

We only give a detailed proof sketch for Lemma 21. Since Lemma 21 is an immediate corollary of its concrete security variant Lemma 29, we give the full proof directly for Lemma 29 in Section 6.2 below.

Proof sketch. Consider a quantum-polynomial-time adversary (A, B) where A outputs a hash h, and a superposition of messages m on the register M, and B expects M back and outputs a guess b. We need to show that the two games in Definition 3 (see also Figure 1) are indistinguishable, i.e., the probability of $b = 1$ is approximately the same in both games. Since the domain of $\mathbf{S}^{in}$ is $(\{0,1\}^r)^+$, we can assume that (A, B) is valid on $(\{0,1\}^r)^+$, i.e., M contains a superposition of messages $m \in (\{0,1\}^r)^+$ with $\mathbf{S}^{in}(m) = h$.

To show that the two games are indistinguishable, we start with Game_2 from Definition 3 (Figure 1 (b)) and transform it step by step into Game_1.

Game 1 $(M, h) \leftarrow A()$. $b \leftarrow B(M)$. *(Same as* Game_2 *from Definition 3.)*

Note that we keep the register S (the state of (A, B)) implicit in this proof sketch, to improve readability.

Now, in each successive game, we measure more and more information about the message m contained in M, until in the final game, we measure m completely (like in Game_1 from Definition 3). In order to refer to the different values derived from m that we measure, we will need to use a lot of notation to refer to various intermediate values occurring in the computation of $\mathbf{S}^{in}(m)$. To make it easier to follow the proof, all relevant notation has been depicted in Figure 3, which shows an evaluation of $\mathbf{S}^{in}(m)$.

Since (A, B) is valid, we know that $\mathbf{S}^{in}(m) = h$ where h is the classical output of A. That is, $h = \mathbf{S}^{in}(m)$ has already been measured.[8] In the computation of $\mathbf{S}^{in}(m)$, let s_{-2} refer to the state that goes into the last application of $\mathbf{f}$ (see Figure 3), and let m_{-1} refer to the last block of the input message m (i.e., $m_{-1} = m_{|m|}$). Then $h = \mathbf{f}(s_{-2}) \oplus (m_{-1} \| 0^c)$ by definition of $\mathbf{S}^{in}$, and thus $h^{\mathsf{right}} = \mathbf{f}^{\mathsf{right}}(s_{-2})$. Now since by assumption, $\mathbf{f}^{\mathsf{right}}$ is collapsing, we can reason: Since

[8] In this proof sketch, when we use the expression "measure a" where a is some expression depending on the message m (e.g., a could be $\mathbf{S}^{in}(m)$), then we mean that we measure the register M, but not with a complete measurement, but with a measurement that gives outcome a (e.g., $\mathbf{S}^{in}(m)$) when M contains $|m\rangle$. Formally, that measurement would consist of the projectors P_i defined by $P_i := \sum_{m \text{ s.t. } a=i} |m\rangle\langle m|$. E.g., if we "measure $\mathbf{S}^{in}(m)$", the projectors are $P_i := \sum_{m \text{ s.t. } \mathbf{S}^{in}(m)=i} |m\rangle\langle m|$.

$\mathbf{f}^{\mathsf{right}}$ is collapsing, and we have measured the output h^{right} of $\mathbf{f}^{\mathsf{right}}(s_{-2})$, it follows that we can additionally measure s_{-2}, and the adversary B will not be able to notice the difference. That is, we get the following game with only negligibly different $\Pr[b = 1]$:

Game 2_{attempt} $(M, h) \leftarrow A()$. *Measure s_{-2}.* $b \leftarrow B(M)$.

This would indeed work, if we knew that $|m| \geq 2$. However, it could be that $|m| = 1$. In this case, we have $s_{-2} = \bot$ (i.e., s_{-2} does not occur in the computation). Worse, if M contains a superposition of messages m, some of length $|m| = 1$, others of length $|m| \geq 2$, then measuring s_{-2} will reveal whether $|m| \geq 2$ or $|m| = 1$. We cannot guarantee that this measurement will not change the quantum state of M in a noticeable way. Then $\Pr[b = 1 : \text{Game } 1] \not\approx \Pr[b = 1 : \text{Game } 2_{\text{attempt}}]$. The collapsing property of $\mathbf{f}^{\mathsf{right}}$ does not help here, because to apply that property, we need to know that h^{right} is indeed the output of $\mathbf{f}^{\mathsf{right}}$ (which is not the case when $|m| = 1$).

A similar problem also occurs in later games: Let s_{-k} denote the k-th state from the end, with h being s_{-1}, the input to the last $\mathbf{f}$ being s_{-2}, the input to the previous $\mathbf{f}$ being s_{-3}, etc., see Figure 3. (We count backwards because this will make notation easier, since our games will start measuring states from the end.) When we have measured some right state s_{-k}^{right}, we want to argue that we can measure the previous state $s_{-(k+1)}$ because $s_{-k}^{\mathsf{right}} = \mathbf{f}^{\mathsf{right}}(s_{-(k+1)})$. Again, this will not be possible because we do not know whether s_{-k} is not already the first state of the computation of $\mathbf{S}^{in}(m)$. (That is, we do not know whether $|m| = k$.)

To get around this problem, we need a mechanism to decide whether a state s_{-k} is the initial state, i.e., whether s_{-k} is $s_{-|m|}$. How do we do that? By construction of $\mathbf{S}^{in}$, the initial state $s_{-|m|}$ satisfies $s_{-|m|}^{\mathsf{right}} = 0^c$. Thus, we might try to decide whether s_{-k} is the initial state by checking whether $s_{-k}^{\mathsf{right}} = 0^c$. For example, if we want to measure s_{-2}, we do so only when $h^{\mathsf{right}} = s_{-1}^{\mathsf{right}} \neq 0^c$. This approach is basically sound, but what happens when a state in the middle has $s_{-k}^{\mathsf{right}} = 0^c$? We would be mislead, and the proof would break down.

To avoid this problem, we will first measure at which positions this bad case happens. Let $\mathbf{b}$ be the set of all indices $k < |m|$ such that $s_{-k}^{\mathsf{right}} = 0^c$. (That is, the indices of all states in which we observe a 0^c in the right part, but which are not the initial state.) Once we know the set $\mathbf{b}$, then we can decide whether s_{-k} is the initial state or not. Namely, s_{-k} is the initial state (i.e., $k = |m|$) iff $s_{-k}^{\mathsf{right}} = 0^c$ and $k \notin \mathbf{b}$.

So the first step in our sequence of games is to measure the set $\mathbf{b}$:

Game 2 $(M, h) \leftarrow A()$. *Measure* $\mathbf{b}$. $b \leftarrow B(M)$.

We assumed that $\mathbf{f}^{\mathsf{right}}$ is zero-preimage-resistant. This implies that with overwhelming probability, $s_{-k}^{\mathsf{right}} = \mathbf{f}^{\mathsf{right}}(s_{-(k+1)}) \neq 0^c$ for all $k < |m|$. Thus $\mathbf{b} = \varnothing$ with overwhelming probability. Therefore measuring $\mathbf{b}$ has only negligible effect on the quantum state. Thus

$$\Pr[\text{Game } 1] \approx \Pr[\text{Game } 2].$$

(We use the shorthand $\Pr[\text{Game } 1]$ for $\Pr[b = 1 : \text{Game } 1]$. And $\approx$ denotes a negligible difference.)

Now we can proceed with measuring more and more states from the computation of $\mathbf{S}^{in}(m)$. First, we measure s_{-1}:

Game 4_1 $(M, h) \leftarrow A()$. *Measure* $\mathbf{b}$ *and s_{-1}.* $b \leftarrow B(M)$.

(Note: The numbering of games in this proof sketch has gaps so that the game numbers here match the game numbers from the full proof of Lemma 29.) Since $s_{-1} = h$ by definition, and since h is already measured by A, the additional measurement does not change the quantum state, and we have:

$$\Pr[\text{Game } 2] = \Pr[\text{Game } 4_1].$$

Now we add a measurement whether s_{-2} is defined:

Game 5_1 $(M, h) \leftarrow A()$. *Measure* $\mathbf{b}$ *and* s_{-1} *and whether* $s_{-2} = \bot$.[9] $b \leftarrow B(M)$.

Given $\mathbf{b}$ and s_{-1} we can already tell whether $s_{-2} = \bot$. Namely, $s_{-2} = \bot$ iff $s_{-1}^{\mathsf{right}} = 0^c$ and $1 \notin \mathbf{b}$. Thus measuring whether $s_{-2} = \bot$ has no effect on the quantum state, and we get

$$\Pr[\text{Game } 4_1] = \Pr[\text{Game } 5_1].$$

Now, finally, we can do what we already intended to do in Game 2_{attempt}: We measure s_{-2}, and use the collapsing property of $\mathbf{f}^{\mathsf{right}}$ to show that this measurement does not noticeably disturb the quantum state:

Game 4_2 $(M, h) \leftarrow A()$. *Measure* $\mathbf{b}$, *and* s_{-1}, *and whether* $s_{-2} = \bot$, *and* s_{-2}. $b \leftarrow B(M)$.

In case that we measured that $s_{-2} = \bot$, measuring s_{-2} in Game 4_2 has no effect on the quantum state (since we know that the outcome will be $\bot$). And in case that we measured that $s_{-2} \neq \bot$, we know that $s_{-1}^{\mathsf{right}} = \mathbf{f}^{\mathsf{right}}(s_{-2})$ (as already discussed above), and thus measuring s_{-2} can be noticed with at most noticeable probability by a quantum-polynomial-time adversary. Thus

$$\Pr[\text{Game } 5_1] \approx \Pr[\text{Game } 4_2].$$

And then we continue by adding a measurement whether $s_{-3} \neq \bot$:

Game 5_2 $(M, h) \leftarrow A()$. *Measure* $\mathbf{b}$, *and* s_{-1}, *and whether* $s_{-2} = \bot$, *and* s_{-2}, *and whether* $s_{-3} = \bot$. $b \leftarrow B(M)$.

Since $s_{-3} = \bot$ iff $s_{-2} = \bot$ or $s_{-2}^{\mathsf{right}} = 0^c$ and $2 \notin \mathbf{b}$, measuring whether $s_{-3} = \bot$ holds has no effect on the quantum state. Thus we get

$$\Pr[\text{Game } 4_2] = \Pr[\text{Game } 5_2].$$

And then we measure s_{-3}:

Game 4_3 $(M, h) \leftarrow A()$. *Measure* $\mathbf{b}$, *and* s_{-1}, *and whether* $s_{-2} = \bot$, *and* s_{-2}, *and whether* $s_{-3} = \bot$, *and* s_{-3}. $b \leftarrow B(M)$.

Using that $\mathbf{f}^{\mathsf{right}}$ is collapsing, we get

$$\Pr[\text{Game } 5_2] \approx \Pr[\text{Game } 4_3].$$

We continue in this way, alternatively adding a measurement whether the next state $s_{-k} = \bot$, and then adding a measurement of s_{-k}, each time using the collapsing property of $\mathbf{f}^{\mathsf{right}}$. After ℓ such steps, where ℓ is a polynomial upper bound on the length of m, we get the following game:

Game 4_ℓ $(M, h) \leftarrow A()$. *Measure* $\mathbf{b}$, *measure whether* $s_{-1}, \dots, s_{-\ell} = \bot$, *measure* $s_{-1}, \dots, s_{-\ell}$. $b \leftarrow B(M)$.

Since in each of the steps, we accrue only a negligible distinguishing probability between consecutive games, we get:

$$\Pr[\text{Game } 4_1] \approx \Pr[\text{Game } 4_\ell].$$

(Formally, this argument is not correct, because the sum of polynomially many possibly different negligible functions is not necessarily negligible. However, this is easily fixed by bounding all differences $\Pr[\text{Game } 5_k] - \Pr[\text{Game } 4_{k+1}]$ simultaneously using a reduction that randomly chooses k (a standard technique). Details are given in the full proof of Lemma 29.)

[9] Measuring "whether $s_{-2} = \bot$" means a measurement on M defined by projectors P and $1 - P$ where $P := \sum_{m \text{ s.t. } s_{-2}=\bot} |m\rangle\langle m|$.

In Game 4_ℓ, we measure $s_{-1},\dots,s_{-\ell}$. From these, we can compute $|m|$ (since $s_{-k}=\bot$ for $k>|m|$). Furthermore, each message block m_{-k} (the k-th message block from the end) can be computed as follows: We have $s_{-k}=\mathbf{f}(s_{-(k+1)})\oplus(m_{-k}\|0^c)$, and thus $m_{-k}=s_{-k}^{\mathsf{left}}\oplus\mathbf{f}(s_{-(k+1)})^{\mathsf{left}}$. Except for $m_{-|m|}$, which can be computed as $m_{-|m|}=s_{-|m|}^{\mathsf{left}}$. (Cf. Figure 3.) Finally, we can compute m as $m=m_{-|m|}\|\dots\|m_{-1}$.

Since we can compute m from the measurements performed in Game 4_ℓ, it follows that those measurements are equivalent (in their effect on the quantum state) to a measurement of m. Thus

$$\Pr[\text{Game } 4_\ell]=\Pr[\text{Game } 6]$$

for the following final game:

Game 6 $(M,h)\leftarrow A()$. *Measure m.* $b\leftarrow B(M)$.

Altogether, we have shown

$$\Pr[\text{Game } 1]\approx\Pr[\text{Game } 6].$$

And Game 1 and Game 6 are identical to the games Game_2 and Game_1 from Definition 3, respectively. Since (A,B) was an arbitrary quantum-polynomial-time adversary that is valid for $\mathbf{S}^{in}$, it follows by Definition 3 that $\mathbf{S}^{in}$ is collapsing. □

Next, we show that the squeezing phase is collapsing. Let $\mathbf{f}^{\mathsf{left}/n}$ be defined for $n>0$, as the first $\min(n,r)$ bits of the output of $\mathbf{f}$ (in particular, $\mathbf{f}^{\mathsf{left}/n}=\mathbf{f}^{\mathsf{left}}$ for $n\geq r$). Then the collapsing property of the squeezing phase is a relatively trivial consequence of the fact that $\mathbf{f}^{\mathsf{left}/n}$ is collapsing.

Lemma 22 (Squeezing phase is collapsing) *Let $n>0$ be the output length and assume that $\mathbf{f}^{\mathsf{left}/n}$ is collapsing. Then $\mathbf{S}^{out}$ is collapsing.*

A concrete security variant of this lemma is given in Lemma 30.

Proof. Let $G_\eta(x)$ return the first $\eta=\min(r,n)$ bits of x. Then $G_\eta(\mathbf{S}^{out}(s))=\mathbf{f}^{\mathsf{left}/n}(s)$. Thus the lemma follows directly from Lemma 6. □

And finally we get that the sponge construction as a whole is collapsing. This is a simple corollary from the fact that both the absorbing and the squeezing phase are collapsing.

Theorem 23 (Sponge construction is collapsing) *Let $n>0$ be the output length and assume that $\mathbf{f}^{\mathsf{left}/n}$ and $\mathbf{f}^{\mathsf{right}}$ are collapsing, and that $\mathbf{f}^{\mathsf{right}}$ is zero-preimage-resistant. Assume that pad is injective. Then $\mathbf{S}$ is collapsing.*

A concrete security variant of this theorem is given in Theorem 31.

Proof. By Lemma 21, $\mathbf{S}^{in}$ is collapsing, and by Lemma 22, $\mathbf{S}^{out}$ is collapsing. Then by Lemma 7, $\mathbf{S}^{out}\circ\mathbf{S}^{in}$ is collapsing. Since *pad* is injective, by Lemma 4, *pad* is collapsing. Thus by Lemma 7, $\mathbf{S}=(\mathbf{S}^{out}\circ\mathbf{S}^{in})\circ \mathit{pad}$ is collapsing. □

6.1 Using random oracles or random permutations

In preceding section, we have reduced the security of the sponge construction $\mathbf{S}$ to certain properties of the block function $\mathbf{f}$. In many cases, however, a hash function is designed by constructing a block function heuristically, and we cannot say what specific security properties the block function has. Instead, we model the block function as, e.g., a random function or a random permutation. The present section specifies the security of the sponge construction under those circumstances. (Concrete security statements are deferred to Section 6.3.)

We start by deriving the required properties of a random block function:

Lemma 24 *If $\mathcal{O} : \{0,1\}^{r+c} \to \{0,1\}^{r+c}$ is a random function, and $\mathbf{f}^{\mathcal{O}}(x) := \mathcal{O}(x)$, and r, c and n are superlogarithmic, then $\mathbf{f}^{\mathsf{left}/n}$ (as defined above Lemma 22) and $\mathbf{f}^{\mathsf{right}}$ are collapsing.*

This is an immediate corollary of its concrete security variant Lemma 32, for which we give a full proof in Section 6.3.

Proof sketch. By Theorem 11, a random function (with superlogarithmic output length) is collapsing. Since $\mathbf{f}^{\mathsf{right}} = \mathcal{O}^{\mathsf{right}}$ has superlogarithmic output length r, and $\mathbf{f}^{\mathsf{right}} = \mathcal{O}^{\mathsf{right}}$ is a uniformly random function if $\mathcal{O}$ is, it follows that $\mathbf{f}^{\mathsf{right}}$ is collapsing. (A slight technicality is that the adversary does not only get access to $\mathbf{f}^{\mathsf{right}} = \mathcal{O}^{\mathsf{right}}$, but additionally to $\mathcal{O}^{\mathsf{left}}$, but it is easy to see that this does not help him since $\mathcal{O}^{\mathsf{right}}$ is distributed independently from $\mathcal{O}^{\mathsf{left}}$.)

Analogously we get that $\mathbf{f}^{\mathsf{left}/n}$ is collapsing since $\min(n, r)$ is superlogarithmic. □

Lemma 25 *If $\mathcal{O} : \{0,1\}^{r+c} \to \{0,1\}^{r+c}$ is a random function, and $\mathbf{f}^{\mathcal{O}}(x) := \mathcal{O}(x)$, and c is superlogarithmic, then $\mathbf{f}^{\mathsf{right}}$ is zero-preimage-resistant.*

This is an immediate corollary of its concrete security variant Lemma 25, for which we give a full proof in Section 6.3. It is a simple consequence of the hardness of quantum searching in unstructured data [Boy+98].

With these two lemmas, security of the sponge using a random block function follows.

In this section, we show that $\mathbf{S}$ is collapsing, when $\mathcal{O}$ is a random function or random permutation and $\mathbf{f}^{\mathcal{O}}(x) := \mathcal{O}(x)$. The collapsing of $\mathbf{f}^{\mathsf{right}}, \mathbf{f}^{\mathsf{left}/n}$ follows from [Unr16a], and the zero-preimage-resistance of $\mathbf{f}^{\mathsf{right}}$ follows from the optimality of Grover's algorithm. The computation of the precise advantage is given in Section 6.2.

Theorem 26 *If $\mathcal{O} : \{0,1\}^{r+c} \to \{0,1\}^{r+c}$ is a random function, and $\mathbf{f}^{\mathcal{O}}(x) := \mathcal{O}(x)$, and r, c and output length n are superlogarithmic, then $\mathbf{S}$ is collapsing.*

A concrete security variant (with security bounds in terms of the number of oracle-queries) is given in Theorem 33.

Proof. Immediate from Theorem 23, using Lemma 24 and Lemma 25 to show that its preconditions are satisfied. □

And since random functions and random permutations are known to be indistinguishable, we readily derive the security of the sponge construction also for block functions that are random permutations.

Theorem 27 *If $\mathcal{O} : \{0,1\}^{r+c} \to \{0,1\}^{r+c}$ is a random permutation, and $\mathbf{f}^{\mathcal{O}}(x) := \mathcal{O}(x)$, and r, c and output length n are superlogarithmic, then $\mathbf{S}$ is collapsing.*

A concrete security variant (with security bounds in terms of the number of oracle-queries) is given in Theorem 34.

Proof. Zhandry [Zha15] shows that no adversary making a polynomial number of queries can distinguish a random permutation from a random function with more than negligible probability (assuming that the output length is superlogarithmic). Thus the advantage of the adversary attacking the collapsing property of $\mathbf{S}$ when $\mathcal{O}$ is a random permutation can only be negligibly higher than the advantage of the same adversary attacking the collapsing property of $\mathbf{S}$ when $\mathcal{O}$ is a random function. The latter advantage is negligible by Theorem 26, thus the former advantage is negligible, too. Hence $\mathbf{S}$ is collapsing when $\mathcal{O}$ is a random permutation. □

On invertible permutations. Theorem 34 tells us that the sponge construction is collapsing if the block function $\mathbf{f}$ is a random permutation. There is a caveat, though: If $\mathbf{f}$ is a permutation that we can efficiently invert, the theorem does not tell us anything. More specifically, if $\mathbf{f}$ is a random permutation, and the adversary has access to $\mathcal{O} = (\mathbf{f}, \mathbf{f}^{-1})$,[10] then Theorem 34 does not apply.

In fact, if $\mathbf{f}$ is efficiently invertible, we cannot apply our main result Theorem 23 at all: In that case $\mathbf{f}^{\mathsf{right}}$ is not zero-preimage-resistant (to find a zero-preimage, we simply invoke $\mathbf{f}^{-1}(y\|0^c)$ for an arbitrary $y \in \{0,1\}^r$).

And $\mathbf{f}^{\mathsf{right}}$ is also not collapsing: We get a collision of $\mathbf{f}^{\mathsf{right}}$ by computing $x := \mathbf{f}^{-1}(y\|z)$ and $x' := \mathbf{f}^{-1}(y'\|z)$ for arbitrary y, y', z with $y \neq y'$. So $\mathbf{f}^{\mathsf{right}}$ is not collision-resistant. And since collapsing functions are collision-resistant [Unr16a], $\mathbf{f}^{\mathsf{right}}$ is not collapsing.

Similarly, $\mathbf{f}^{\mathsf{left}/n}$ is not collapsing, either.

So, none of the preconditions of Theorem 23 are satisfied, and we cannot derive that the sponge construction is collapsing (for efficiently invertible $\mathbf{f}$).

Does that mean that the sponge construction is not collapsing in that setting? No. At least it is not obvious how one would use $\mathbf{f}^{-1}$ to break the collapsing property. For example, if we try to find a two-block collision $(m_1\|m_2, m_1'\|m_2')$ for $\mathbf{S}$, then we need that $\mathbf{f}(m_1\|0^c) \oplus m_2\|0^c = \mathbf{f}(m_1'\|0^c) \oplus m_2'\|0^c$. How can we solve this equation by using $\mathbf{f}^{-1}$? Unclear. For other kinds of collisions that we could think of, we fail similarly.

In fact, we conjecture that the sponge construction is still collapsing in this setting:

Conjecture 28 *If $\mathcal{O} : \{0,1\} \times \{0,1\}^{r+c} \to \{0,1\}^{r+c}$ is an invertible random permutation (see footnote 10), and $\mathbf{f}^{\mathcal{O}}(x) := \mathcal{O}(0,x)$, and r, c and output length n are superlogarithmic, then $\mathbf{S}$ is collapsing.*

Showing this conjecture is quite important because it will give evidence for the post-quantum security of SHA3. (Only in the idealized invertible random permutation model. But then, classically, we also have evidence for the collision-resistance of the sponge construction only in that idealized model [Ber+08].)

6.2 Concrete security results

In this section, we prove the concrete security variants of the results from Section 6. The proofs follow the lines sketched in Section 6, except that we need to be more explicit about the adversaries constructed during the reductions.

The reader interested only in the qualitative fact that the sponge construction is secure, but not in the precise bounds, can safely skip this section.

As in Section 6, we omit $\mathcal{O}$ from $A^{\mathcal{O}}, \mathbf{f}^{\mathcal{O}}$, etc.

Lemma 29 *Let (A, B) be a τ-time adversary, t-valid against $\mathbf{S}$ on $(\{0,1\}^r)^+$, with collapsing-advantage ε. Assume that A outputs messages of length at most ℓr bits.*

Then there are:

- *a $(\tau + t\ell\tau_{\mathbf{f}} + O(t\ell c))$-time adversary A' that finds a zero-preimage[11] of $\mathbf{f}^{\mathsf{right}}$ with probability ε', and*
- *a $\big(\tau + O(\ell\tau_{\mathbf{f}})\big)$-time adversary (A'', B''), t-valid against $\mathbf{f}^{\mathsf{right}}$, with collapsing-advantage ε'' such that $\varepsilon \leq \sqrt{\varepsilon'} + (\ell - 1)\varepsilon''$.*

Here $\tau_{\mathbf{f}}$ is the time required a single classical invocation of $\mathbf{f}$. If time is measured in $\mathcal{O}$-queries (instead of computation steps), the term $O(t\ell c)$ in the runtime of A' can be omitted.

[10] Formally, we mean by $\mathcal{O} = (\mathbf{f}, \mathbf{f}^{-1})$: $\mathcal{O}(0, \cdot)$ is a random permutation, $\mathcal{O}(1, \cdot)$ is its inverse, and $\mathbf{f}(x) := \mathcal{O}(0, x)$.

[11] By *zero-preimage* we mean any value x with $\mathbf{f}^{\mathsf{right}}(x) = 0^c$.

Proof. We prove this lemma using a sequence of games. The first game is identical to Game_2 from Definition 8.

Game 1 $(S, M_1, \dots, M_t, h_1, \dots, h_t) \leftarrow A()$. $b \leftarrow B(S, M_1, \dots, M_t)$.

To state the second game, we need to introduce some notation. (We recommend the reader to compare with Figure 3. The notation there is slightly different, e.g., we write there s_{-1} and here more precisely $s_{-1}(m)$, but the figure can be helpful to get an overview which parts of the message the different values defined below refer to.)

For some $m \in (\{0,1\}^r)^+$, we denote by $|m|$ the number of r-bit blocks in m. (That is, if m has d bits, then $|m| = d/r$.) By m_i we denote the i-th block of m. We denote by m_{-i} the i-th block of m from the end, i.e., $m_{-i} = m_{|m|-i+1}$ and $m = m_{-|m|} \| \dots \| m_{-1}$.

For $i = 1, \dots, |m|$, let $s_{-i}(m) := \mathbf{S}^{in}(m_1, \dots, m_{|m|-i+1})$. That is, $s_{-i}(m)$ is the i-th state during the evaluation of $\mathbf{S}^{in}(m)$, counted from the end.

In particular, $s_{-1}(m) = \mathbf{S}^{in}(m)$, and $s_{-|m|}(m) = m_1 \| 0^c$. (Since we consider only $|m| \in (\{0,1\}^r)^+$, the border case $|m| = 0$ does not occur.) We define $s_{-i}(m) := \bot$ for $i > |m|$.

Let $\mathbf{b}(m) := \{i : s_{-i}(m)^{\mathsf{right}} = 0^c \wedge i < |m|\}$. Note that we always have $s_{-|m|}^{\mathsf{right}} = 0^c$ by definition, but due to the condition $i < |m|$, we never have $|m| \in \mathbf{b}(m)$.

Notice that $i \in \mathbf{b}(m)$ implies that $s_{-(i+1)}(m)$ is a zero-preimage of $\mathbf{f}^{\mathsf{right}}$.

The intuitive meaning of $\mathbf{b}(m)$ is that it tells us whether (and where) at some point during the computation of $\mathbf{S}^{in}(m)$, there is a state s with $s^{\mathsf{right}} = 0^c$. This is important to know, because later we will interpret such a state as an indication that we have reached the beginning of the computation of $\mathbf{S}^{in}$. $\mathbf{b}(m)$ tells us where there are exceptions to this rule.

Finally, in the following, for a function f, let $\mathcal{M}_f(M)$ denote a measurement of the register M that measures $f(m)$ when $|m\rangle$ is the state of M. More formally, $\mathcal{M}_f(M)$ consists of the projectors $\{P_i\}_{i \in \mathrm{range}\, f}$ with $P_i := \sum_{m \text{ s.t. } f(m)=i} |m\rangle\langle m|$. For example, if f is the identity, then $\mathcal{M}_f(M)$ is a complete measurement in the computational basis, and if f is a constant function, then $\mathcal{M}_f(M)$ does not do anything. We write $x \leftarrow \mathcal{M}_f(M)$ to denote that the outcome of $\mathcal{M}_f(M)$ is stored in x.

Note that all measurements $\mathcal{M}_f$ commute, even for different functions f. We will use this fact implicitly by treating sequences of measurements as equal when they differ only in their ordering.

Game 2 $(S, M_1, \dots, M_t, h_1, \dots, h_t) \leftarrow A()$. $\mathbf{b}_j \leftarrow \mathcal{M}_{\mathbf{b}}(M_j)$ *for* $j = 1, \dots, t$. $b \leftarrow B(S, M_1, \dots, M_t)$.

That is, compared to Game 1, we additionally measure $\mathbf{b}(m)$ for each of the messages m that are in the registers $M_1, \dots, M_t$. We will need this below in order to decide whether a given message block is the first message block (when working our way backwards through the message). Namely, m_{-i} is the beginning of the message m, iff $s_{-i}(m)^{\mathsf{right}} = 0^c$ and $i \notin \mathbf{b}(m)$.

For any game Game X, let Pr[Game X] denote the probability that $b = 1$ in Game X. (I.e., Pr[Game X] := Pr[$b = 1$: Game X].)

We will now bound $\big|\Pr[\text{Game 1}] - \Pr[\text{Game 2}]\big|$. Let ρ denote the state of $S, M_1, \dots, M_t$ after executing $(S, M_1, \dots, M_t, h_1, \dots, h_t) \leftarrow A()$. Let ρ' denote the state after additionally performing the measurement "$\mathbf{b}_j \leftarrow \mathcal{M}_{\mathbf{b}}(M_j)$ for $j = 1, \dots, t$".[12] By Lemma 2, the trace distance between ρ and ρ' is bounded by $\sqrt{\varepsilon_0'}$, where $1 - \varepsilon_0' := \Pr[(\mathbf{b}_1, \dots, \mathbf{b}_t) = (\varnothing, \dots, \varnothing) : \text{Game 2}]$. And ρ and ρ' are the state before executing $b \leftarrow B(S, M_1, \dots, M_t)$ in Game 1 and Game 2, respectively.

[12] Strictly speaking, these are t measurements $\mathcal{M}_{\mathbf{b}}(M_1), \dots, \mathcal{M}_{\mathbf{b}}(M_t)$. However, since the measurements $\mathcal{M}_{\mathbf{b}}(M_j)$ commute, one can see them as a single measurement on $M_1, \dots, M_t$ with outcome $\mathbf{b}_1, \dots, \mathbf{b}_t$.

Thus the probability of $b = 1$ in these two games can differ at most by the trace distance between ρ and ρ', hence

$$\Big|\Pr[\text{Game 1}] - \Pr[\text{Game 2}]\Big| \leq \sqrt{\varepsilon'_0}. \tag{3}$$

Since $\Pr[(\mathbf{b}_1, \dots, \mathbf{b}_t) = (\varnothing, \dots, \varnothing) : \text{Game 2}] = 1 - \varepsilon'_0$, we have $\Pr[\exists j : \ \mathbf{b}_j \neq \varnothing] = \varepsilon'_0$ in the following game:

Game 3 *$(S, M_1, \dots, M_t, h_1, \dots, h_t) \leftarrow A()$. $m_j \leftarrow \mathcal{M}(M_j)$ for $j = 1, \dots, t$. $\mathbf{b}_j := \mathbf{b}(m_j)$ for $j = 1, \dots, t$. $h'_j := \mathbf{S}^{in}(m_j)$ for $j = 1, \dots, t$.*

By definition of $\mathbf{b}(m)$, we have that $\mathbf{b}(m) \neq \varnothing$ implies that $s_{-(i+1)}(m)$ is a zero-preimage of $\mathbf{f}^{\mathsf{right}}$ for some i. Thus, if $\exists j : \ \mathbf{b}_j \neq \varnothing$ in Game 3, then one of the $s_{-i}(m_j)$ is a zero-preimage of $\mathbf{f}^{\mathsf{right}}$. Thus with probability $\geq \varepsilon'_0$, one of the computations $\mathbf{S}^{in}(m_j)$ in Game 3 performs a call $\hat{s} := \mathbf{f}(s)$ such that $\hat{s}^{\mathsf{right}} = 0^c$.

Consider the following algorithm A': It computes $(S, M_1, \dots, M_t, h_1, \dots, h_t) \leftarrow A()$, measures $m_j \leftarrow \mathcal{M}(M_j)$ for $j = 1, \dots, t$, and then computes $\mathbf{S}^{in}(m_j)$ for $j = 1, \dots, t$. If in one of the invocations $\hat{s} := \mathbf{f}(s)$ performed by $\mathbf{S}^{in}$ we have $\hat{s}^{\mathsf{right}} = 0$, A' returns s.

Then A' outputs a zero-preimage of $\mathbf{f}^{\mathsf{right}}$ with some probability $\varepsilon' \geq \varepsilon'_0$. A' has runtime $\leq \tau + t\ell\tau_{\mathbf{f}} + O(t\ell c)$. And from (3), we get

$$\Big|\Pr[\text{Game 1}] - \Pr[\text{Game 2}]\Big| \leq \sqrt{\varepsilon'}. \tag{4}$$

For the next game, we introduce one more function, d_i, defined by: $d_{-i}(m) := 1$ if $s_{-i}(m) \neq \bot$, and $d_{-i}(m) := 0$ otherwise. Equivalently, $d_{-i}(m) = 1$ iff $|m| \geq i$.

The next game is a variation of Game 2 and is parametrized by an integer $k = 1, \dots, \ell$.

Game 4$_k$ *$(S, M_1, \dots, M_t, h_1, \dots, h_t) \leftarrow A()$. $\mathbf{b}_j \leftarrow \mathcal{M}_{\mathbf{b}}(M_j)$ for $j = 1, \dots, t$. $s_{ij} \leftarrow \mathcal{M}_{s_{-i}}(M_j)$ for $i = 1, \dots, k$ and $j = 1, \dots, t$. $d_{ij} \leftarrow \mathcal{M}_{d_{-i}}(M_j)$ for $i = 1, \dots, k$ and $j = 1, \dots, t$. $b \leftarrow B(S, M_1, \dots, M_t)$.*

In this game, we measure additionally $s_{-1}(m), \dots, s_{-k}(m)$ for each of the messages m on registers $M_1, \dots, M_t$. That is, we measure a *suffix* of length k of the list of states occurring during the calculation of $\mathbf{S}^{in}(m)$. Thus, in each successive game Game 4_k (for increasing k), we measure one more state from the calculation (starting from the end). We also measure $d_{-1}(m), \dots, d_{-k}(m)$, but this measurement has no effect since $d_{-i}(m)$ is determined by $s_{-i}(m)$ by definition.

Notice also that $s_{-k}(m) = \bot$ if $k > |m|$ (by definition of s_{-k}). Thus, measuring $s_{-1}(m), \dots, s_{-k}(m)$ implicitly also measures whether $|m| \leq k$, and if so, what the value of $|m|$ is.

We first consider the case $k = 1$: The only difference between Game 4_1 and Game 2 are the measurements $s_{1j} \leftarrow \mathcal{M}_{s_{-1}}(M_j)$ and $d_{1j} \leftarrow \mathcal{M}_{d_{-1}}(M_j)$. But as mentioned above, $s_{-1}(m) = \mathbf{S}^{in}(m)$. Thus $s_{1j} \leftarrow \mathcal{M}_{s_{-1}}(M_j)$ is the same as $s_{1j} \leftarrow \mathcal{M}_{\mathbf{S}^{in}}(M_j)$. Since (A, B) is a t-valid adversary, we have by definition of t-validity that measuring $\mathcal{M}_{\mathbf{S}^{in}}$ on the registers M_j returns h_j with probability 1 (i.e., $s_{1j} = h_j$). Thus the measurement $s_{1j} \leftarrow \mathcal{M}_{s_{-1}}(M_j)$ has no effect on the state that is measured. Hence we can omit it without changing the distribution of b. Furthermore, $s_{1j}(m) \neq \bot$ for all $m \in (\{0,1\}^r)^+$. Thus we always have $d_{1j}(m) = 1$, thus the measurement $d_{1j} \leftarrow \mathcal{M}_{d_{-1}}(M_j)$ has no effect on b either. Thus

$$\Pr[\text{Game } 4_1] = \Pr[\text{Game 2}]. \tag{5}$$

Game 5$_k$ *$(S, M_1, \dots, M_t, h_1, \dots, h_t) \leftarrow A()$. $\mathbf{b}_j \leftarrow \mathcal{M}_{\mathbf{b}}(M_j)$ for $j = 1, \dots, t$. $s_{ij} \leftarrow \mathcal{M}_{s_{-i}}(M_j)$ for $i = 1, \dots, k$ and $j = 1, \dots, t$. $d_{ij} \leftarrow \mathcal{M}_{d_{-i}}(M_j)$ for $i = 1, \dots, k+1$ and $j = 1, \dots, t$. $b \leftarrow B(S, M_1, \dots, M_t)$.*

Thus, in comparison with Game 4_k, we additionally measure $d_{-(k+1)}(m)$ for the messages m on the registers M_j. In other words, we measure whether $s_{-k}(m) \neq \bot$, which in turn is the same as measuring whether $|m| \geq k+1$.

Notice further that for any $m \in (\{0,1\}^r)^+$, the following holds: If $d_{-(k+1)}(m) = 0$, we have $|m| \leq k$. In case $|m| < k$, we have $s_{-k}(m) = \bot$. In case $|m| = k$, we have $s_{-k}(m)^{\mathsf{right}} = 0^c$ (because $s_{-|m|}(m) = m_1 \| 0^c$ as mentioned above) and $k \notin \mathbf{b}(m)$ (by definition of $\mathbf{b}$). Thus

$$\forall m \in (\{0,1\}^r)^+. \qquad d_{-(k+1)}(m) = 0 \quad \Longrightarrow \quad s_{-k}(m) = \bot \vee \big(s_{-k}(m)^{\mathsf{right}} = 0^c \wedge k \notin \mathbf{b}(m)\big).$$

Furthermore, the converse holds, too: If $s_{-k}(m) = \bot$, then $|m| < k$, hence $d_{-(k+1)}(m) = 0$. And if $s_{-k}(m)^{\mathsf{right}} = 0^c \wedge k \notin \mathbf{b}(m)$, by definition of $\mathbf{b}(m)$, we have $s_{-k}(m)^{\mathsf{right}} = 0^c \wedge \neg\big(s_{-k}(m)^{\mathsf{right}} = 0^c \wedge k < |m|\big)$, thus $k \geq |m|$, thus $s_{-(k+1)}(m) = \bot$ thus $d_{-(k+1)}(m) = 0$. So, altogether we have

$$\forall m \in (\{0,1\}^r)^+. \qquad d_{-(k+1)}(m) = 0 \quad \Longleftrightarrow \quad s_{-k}(m) = \bot \vee \big(s_{-k}(m)^{\mathsf{right}} = 0^c \wedge k \notin \mathbf{b}(m)\big).$$

Thus, in the context of Game 5_k (and using that $m \in (\{0,1\}^r)^+$ since (A,B) is t-valid on $(\{0,1\}^r)^+$), we have for all $j = 1,\dots,t$:

$$d_{k+1,j} = 0 \quad \Longleftrightarrow \quad s_{kj} = \bot \vee \big(s{-}kj^{\mathsf{right}} = 0^c \wedge k \notin \mathbf{b}_j\big).$$

Thus, the outcome $d_{k+1,j} \in \{0,1\}$ of the measurement $d_{k+1,j} \leftarrow \mathcal{M}_{d_{-(k+1)}}(M_j)$ is determined by the prior measurement outcomes $s_{kj}, \mathbf{b}_j$. Hence $d_{k+1,j} \leftarrow \mathcal{M}_{d_{-(k+1)}}(M_j)$ has no effect on the state and thus omitting it does not change the distribution of b in Game 5_k. Since this measurement is the only difference between Game 4_k and Game 5_k, we have

$$\Pr[\text{Game } 4_k] = \Pr[\text{Game } 5_k]. \tag{6}$$

Let Game $5_\$$ denote the game in which we pick $k \xleftarrow{\$} \{1,\dots,\ell-1\}$, and then execute Game 5_k.

Let Game $4_{\$+1}$ denote the game in which we pick $k \xleftarrow{\$} \{1,\dots,\ell-1\}$, and then execute Game 4_{k+1}.

We will now bound $\big|\Pr[\text{Game } 4_{\$+1}] - \Pr[\text{Game } 5_\$]\big|$. For this, consider the following algorithms A'', B''. Let $\mathbf{U}_{s_k}$ be the unitary with $\mathbf{U}_{s_k}|m\rangle|y\rangle = |m\rangle|y \oplus s_{-(k+1)}(m)\rangle$. $A''()$ performs the steps described in Algorithm 1.

Algorithm 1 Algorithm A''

```
Output: Quantum registers $(S'', M''_1, \dots, M''_t, h''_1, \dots, h''_t)$
Pick $k \xleftarrow{\$} \{1,\dots,\ell-1\}$.
Run $(S, M_1, \dots, M_t, h_1, \dots, h_t) \leftarrow A()$.
Measure $\mathbf{b}_j \leftarrow \mathcal{M}_{\mathbf{b}}(M_j)$ for $j = 1,\dots,t$.
Measure $s_{ij} \leftarrow \mathcal{M}_{s_{-i}}(M_j)$ for $i = 1,\dots,k$ and $j = 1,\dots,t$.
Measure $d_{ij} \leftarrow \mathcal{M}_{d_{-i}}(M_j)$ for $i = 1,\dots,k+1$ and $j = 1,\dots,t$.
For $j = 1,\dots,t$:
    Initialize $M''_j$ with $|0^{r+c}\rangle$.
    If $d_{k+1,j} = 1$:
        apply $\mathbf{U}_{s_k}$ to the registers $M_j, M''_j$.
Combine the registers $S, M_1, \dots, M_t$ and classical values $k$, $(d_{k+1,j})_j$ into a single register $S''$.
Let $h''_j := s_{kj}^{\mathsf{right}}$ if $s_{kj} \neq \bot$, and $h''_i := \mathbf{f}(0^{r+c})^{\mathsf{right}}$ otherwise.
Return $(S'', M''_1, \dots, M''_t, h''_1, \dots, h''_t)$.
```

And $B''(S'', M''_1, \dots, M''_t)$ performs the steps in Algorithm 2.

Algorithm 2 Algorithm B''

```
Input: Quantum registers $(S'', M''_1, \dots, M''_t, h''_1, \dots, h''_t)$
Output: Bit $b$
Split the register $S''$ into registers $S, M_1, \dots, M_t$ and classical values $k$, $(d_{kj})_j$.
For $j = 1, \dots, t$:
    If $d_{k+1,j} = 1$:
        Apply $\mathbf{U}_{s_k}$ to the registers $M_j, M''_j$.
$b \leftarrow B(S, M_1, \dots, M_t)$.
Return $b$.
```

First, note that (A'', B'') is t-valid for $\mathbf{f}^{\mathsf{right}}$: If $d_{k+1,j} = 0$, then M''_j contains $|0^{r+c}\rangle$ and we have $h''_j = \mathbf{f}^{\mathsf{right}}(0^{r+c})$. If $d_{k+1,j} = 1$, then M_j contains a superposition of values m with $s_{kj} = s_{-k}(m) \neq \bot$. For these m we have $s_{-k}(m) = \mathbf{f}(s_{-(k+1)}(m)) \oplus (m_{-k} \| 0^c)$ by definition of s_{-k} and thus

$$\mathbf{f}^{\mathsf{right}}(s_{-(k+1)}(m)) = \big(s_{-k}(m) \oplus (m_{-k} \| 0^c)\big)^{\mathsf{right}} = s_{-k}(m)^{\mathsf{right}} = s_{kj}^{\mathsf{right}} = h''_j.$$

Furthermore, since we applied $\mathbf{U}_{s_k}$ to M_j, M''_j, we have that M''_j contains $s_{-(k+1)}(m)$ when M_j contains m. Thus M''_j contains a superposition of values x with $\mathbf{f}^{\mathsf{right}}(x) = h''_j$. So in both cases $d_{k+1,j} = 0$ and $d_{k+1,j} = 1$, M''_j contains a superposition of values x with $\mathbf{f}^{\mathsf{right}}(x) = h''_j$. Thus (A'', B'') is t-valid for $\mathbf{f}^{\mathsf{right}}$.

(A'', B'') has runtime $\tau + O(t\ell\tau_{\mathbf{f}})$. Namely, τ is the time for executing A and B. Each application $\mathbf{U}_{s_k}$ takes time $O(\ell\tau_{\mathbf{f}})$, and there are t of them. Each measurement $\mathcal{M}_{\mathbf{b}}(M_j)$, takes time $O(\ell\tau_{\mathbf{f}})$, and there are t of them. Each measurement $\mathcal{M}_{s_{-i}}(M_j)$, takes time $O(\ell\tau_{\mathbf{f}})$, and there are kt of them, measuring them individually would take time $O(kt\ell\tau_{\mathbf{f}})$. However, $\mathcal{M}_{s_{-1}}(M_j), \dots, \mathcal{M}_{s_{-k}}(M_j)$ can be performed in time $O(\ell\tau_{\mathbf{f}})$ when executed together: one performs a single evaluation of $\mathbf{S}^{in}$ on M_j (in superposition) while keeping all intermediate states in ancilla qubits. Then one measures those intermediate states and uncomputes $\mathbf{S}^{in}$. The effect is to measure $\mathcal{M}_{s_{-1}}(M_j), \dots, \mathcal{M}_{s_{-k}}(M_j)$ in time $O(\ell\tau_{\mathbf{f}})$. Since there are t registers M_j, the total time is $O(t\ell\tau_{\mathbf{f}})$. Similarly, the measurements $\mathcal{M}_{d_{-i}}(M_j)$ can be performed in time $O(t\ell\tau_{\mathbf{f}})$. All other computations can be performed in time $O(t\ell c)$ which is dominated by $O(t\ell\tau_{\mathbf{f}})$. Thus the total time spent by (A'', B'') is $\tau + O(t\ell\tau_{\mathbf{f}})$.

Let Game''_1 and Game''_2 be the games from Definition 8 for adversary (A'', B''). We define ε'' as the collapsing-advantage of (A'', B'') against $\mathbf{f}^{\mathsf{right}}$. In other words, $\varepsilon'' := \big|\Pr[\mathsf{Game}''_1] - \Pr[\mathsf{Game}''_2]\big|$.

Then (A'', B'') has the properties claimed in the statement of this lemma.

Consider Game''_2. We claim that $\Pr[\mathsf{Game}''_2] = \Pr[\text{Game } 5_\$]$. This is because A'', B'' in Game''_2 perform exactly the steps executed in Game $5_\$$, except for some packaging and subsequent unpackaging of registers inside S'', and for the applications of $\mathbf{U}_{s_k}$ to M_j, M''_j. But the latter has no effect because the two applications of $\mathbf{U}_{s_k}$ cancel each other out. So $\Pr[\mathsf{Game}''_2] = \Pr[\text{Game } 5_\$]$.

Consider Game''_1. Here, in comparison with Game''_2, we have an additional measurement of M''_j in the computational basis. In the case $d_{k+1,j} = 1$, this measurement occurs between two applications of $\mathbf{U}_{s_k}$ (for a $|0\rangle$-initialized M''_j). The sequence of steps $\mathbf{U}_{s_k}$, $\mathcal{M}(M''_j)$, $\mathbf{U}_{s_k}$ is equivalent to the measurement $\mathcal{M}_{s_{-(k+1)}}(M_j)$. And in the case $d_{k+1,j} = 0$, this measurement measures the $|0\rangle$-initialized M''_j and has thus no effect on the state, thus it is equivalent to measuring $\mathcal{M}_{s_{-(k+1)}}(M_j)$ which also has no effect on the state (because it always returns outcome $\bot$ when $d_{k+1,j} = 0$). Thus, the additional measurement in Game''_1 is equivalent to a an additional measurement $\mathcal{M}_{s_{-(k+1)}}(M_j)$. Thus $\Pr[\mathsf{Game}''_1] = \Pr[\text{Game } 4_{\$+1}]$ (note that Game $4_{\$+1}$ differs from $\Pr[\text{Game } 5_\$]$ only by this one additional measurement).

It follows that

$$\big|\Pr[\text{Game } 4_{\$+1}] - \Pr[\text{Game } 5_\$]\big| = \big|\Pr[\mathsf{Game}''_1] - \Pr[\mathsf{Game}''_2]\big| = \varepsilon''. \tag{7}$$

By construction of Game $5_\$$, we have $\Pr[\text{Game } 5_\$] = \frac{1}{\ell-1}\sum_{k=1}^{\ell-1}\Pr[\text{Game } 5_k]$. And by construction of Game $4_{\$+1}$, we have $\Pr[\text{Game } 4_{\$+1}] = \frac{1}{\ell-1}\sum_{k=1}^{\ell-1}\Pr[\text{Game } 4_{k+1}]$. Thus

$$\varepsilon'' \overset{(7)}{=} \bigl|\Pr[\text{Game } 4_{\$+1}] - \Pr[\text{Game } 5_\$]\bigr| = \frac{1}{\ell-1}\Bigl|\sum_{k=1}^{\ell-1}\Pr[\text{Game } 4_{k+1}] - \sum_{k=1}^{\ell-1}\Pr[\text{Game } 5_k]\Bigr|$$

$$\overset{(6)}{=} \frac{1}{\ell-1}\Bigl|\sum_{k=1}^{\ell-1}\Pr[\text{Game } 4_{k+1}] - \sum_{k=1}^{\ell-1}\Pr[\text{Game } 4_k]\Bigr| = \frac{1}{\ell-1}\bigl|\Pr[\text{Game } 4_\ell] - \Pr[\text{Game } 4_1]\bigr|.$$

Hence

$$\bigl|\Pr[\text{Game } 4_\ell] - \Pr[\text{Game } 4_1]\bigr| \le (\ell-1)\varepsilon''. \tag{8}$$

We now investigate Game 4_ℓ. For any $m \in (\{0,1\}^r)^+$ with $|m| \le \ell$, we have: $|m|$ is the largest i such that $s_{-i}(m) \ne \bot$. And $s_{-i}(m) = \mathbf{f}(s_{-(i+1)}(m)) \oplus (m_{-i}\|0^c)$, hence $m_{-i} = \mathbf{f}(s_{-(i+1)}(m))^{\mathsf{left}} \oplus s_{-i}(m)^{\mathsf{left}}$. Thus there is a function γ (depending on $\mathbf{f}$) such that $\gamma\bigl(\mathbf{b}(m), s_{-1}(m), \dots, s_{-\ell}(m), d_{-1}(m), \dots, d_{-\ell}(m)\bigr) = m$. And trivially, there is a function γ' (depending on $\mathbf{f}$) such that $\gamma'(m) = \bigl(\mathbf{b}(m), s_{-1}(m), \dots, s_{-\ell}(m), d_{-1}(m), \dots, d_{-\ell}(m)\bigr)$. Thus measuring $\mathbf{b}(m), s_{-1}(m), \dots, s_{-\ell}(m), d_{-1}(m), \dots, d_{-\ell}(m)$ is equivalent to measuring m (in its effects on the measured state), assuming $|m| \le \ell$. Thus, the measurements $\mathbf{b}_j \leftarrow \mathcal{M}_{\mathbf{b}}(M_j)$ and $s_{ij} \leftarrow \mathcal{M}_{s_{-i}}(M_j)$ and $d_{ij} \leftarrow \mathcal{M}_{d_{-i}}(M_j)$ for $i = 1, \dots, \ell$ and $j = 1, \dots, t$ performed in Game 4_ℓ have the same effect on the state as just performing the measurements $m_j \leftarrow \mathcal{M}(M_j)$ for $j = 1, \dots, t$. Thus

$$\Pr[\text{Game } 4_\ell] = \Pr[\text{Game } 6] \tag{9}$$

with the following game:

Game 6 $(S, M_1, \dots, M_t, h_1, \dots, h_t) \leftarrow A()$. *$m_j \leftarrow \mathcal{M}(M_j)$ for $j = 1, \dots, t$.* $b \leftarrow B(S, M_1, \dots, M_t)$

We can now relate the initial and the final game:

$$\bigl|\Pr[\text{Game } 1] - \Pr[\text{Game } 6]\bigr| \overset{(4),(5),(8),(9)}{\le} \sqrt{\varepsilon'} + (\ell-1)\varepsilon''.$$

Since Game 6 is identical to Game_1 from Definition 8, and Game 1 is identical to Game_2 from the same definition, it follows that the collapsing-advantage ε of (A,B) is $\bigl|\Pr[\text{Game } 1] - \Pr[\text{Game } 6]\bigr|$. Thus $\varepsilon \le \sqrt{\varepsilon'} + (\ell-1)\varepsilon''$ and the lemma follows. □

Lemma 30 *Let $n > 0$ be the output length of $\mathbf{S}$.*

Let (A,B) be a t-valid adversary against $\mathbf{S}^{out}$ with runtime τ and collapsing-advantage ε.

Then there is an adversary (A',B') that is t-valid for $\mathbf{f}^{\mathsf{left}/n}$, has runtime $\tau + O(t\min(n,r))$, and collapsing-advantage ε. If time is measured in $\mathcal{O}$-queries (instead of computation steps), the runtime of (A',B') is τ.

Proof. Let $G_\eta(x)$ return the first $\eta = \min(n,r)$ bits of x. Then $G_\eta(\mathbf{S}^{out}(s)) = \mathbf{f}^{\mathsf{left}/n}$. Thus the lemma follows directly from Lemma 12. □

Theorem 31 *Let $n > 0$ be the output length of $\mathbf{S}$. Assume that pad is injective. Assume a τ-time adversary (A,B), t-valid for $\mathbf{S}$, with collapsing-advantage ε against $\mathbf{S}$. Then there are:*

- *a $(\tau + O(t\tau_{pad}) + O(t\tau_{\mathbf{S}^{in}}))$-time adversary A_1 that finds a zero-preimage of $\mathbf{f}^{\mathsf{right}}$ with probability ε_1, and*
- *a $\bigl(\tau + O(t\tau_{pad}) + O(t\tau_{\mathbf{S}^{in}})\bigr)$-time adversary (A_2,B_2), t-valid against $\mathbf{f}^{\mathsf{right}}$, with collapsing-advantage ε_2, and*

- *a $(\tau+O(t\tau_{pad})+O(t\tau_{\mathbf{S}^{in}}))$-time adversary (A_3, B_3) that is t-valid for $\mathbf{f}^{\mathsf{left}/n}$, with collapsing-advantage ε_3,*

such that $\varepsilon \le \sqrt{\varepsilon_1} + (\ell-1)\varepsilon_2 + \varepsilon_3$.

Here τ_{pad} refers to the maximum runtime of pad. (For values m that A may output on the M_j.) And $\tau_{\mathbf{S}^{in}}$ refers to the maximum runtime of $\mathbf{S}^{in}$ (on outputs of pad).

Proof. Since $\mathbf{S} = (\mathbf{S}^{out} \circ \mathbf{S}^{in}) \circ pad$, we can apply Lemma 13 to get:

- a $(\tau + O(t\tau_{pad}))$-time adversary (A_4, B_4), that is t-valid on $\operatorname{im} pad \subseteq (\{0,1\}^r)^+$ and has collapsing-advantage ε_4 against $\mathbf{S}^{out} \circ \mathbf{S}^{in}$,
- a $(\tau + O(t\tau_{pad}))$-time adversary (A_5, B_5), that is t-valid and has collapsing-advantage ε_5 against pad,

such that $\varepsilon \le \varepsilon_4 + \varepsilon_5$. Since pad is injective, we have $\varepsilon_5 = 0$ (Lemma 10). Thus $\varepsilon \le \varepsilon_4$. We can now apply Lemma 13 to $\mathbf{S}^{out} \circ \mathbf{S}^{in}$ and (A_4, B_4). This gives us:

- a $(\tau + O(t\tau_{pad}) + O(t\tau_{\mathbf{S}^{in}}))$-time adversary (A_6, B_6), t-valid with collapsing-advantage ε_6 against $\mathbf{S}^{out}$,
- a $(\tau + O(t\tau_{pad}) + O(t\tau_{\mathbf{S}^{in}}))$-time adversary (A_7, B_7), t-valid on $(\{0,1\}^r)^+$ with collapsing-advantage ε_7 against $\mathbf{S}^{in}$,

where $\varepsilon_4 \le \varepsilon_6 + \varepsilon_7$. Since $\varepsilon \le \varepsilon_4$, we have $\varepsilon \le \varepsilon_6 + \varepsilon_7$. We now apply Lemma 29 to (A_7, B_7). We get:

- a $(\tau + O(t\tau_{pad}) + O(t\tau_{\mathbf{S}^{in}}) + t\ell\tau_{\mathbf{f}} + O(t\ell c)) = (\tau + O(t\tau_{pad}) + O(t\tau_{\mathbf{S}^{in}}))$-time adversary A_1 that finds a zero-preimage of $\mathbf{f}^{\mathsf{right}}$ with probability ε_1, and
- a $\big(\tau + O(t\tau_{pad}) + O(t\tau_{\mathbf{S}^{in}}) + O(t\ell\tau_{\mathbf{f}})\big) = \big(\tau + O(t\tau_{pad}) + O(t\tau_{\mathbf{S}^{in}})\big)$-time adversary (A_2, B_2), t-valid against $\mathbf{f}^{\mathsf{right}}$, with collapsing-advantage ε_2

such that $\varepsilon_7 \le \sqrt{\varepsilon_1} + (\ell-1)\varepsilon_2$. And we apply Lemma 30 to (A_6, B_6). We get:

- a $(\tau + O(t\tau_{pad}) + O(t\tau_{\mathbf{S}^{in}}) + O(t\min(n,r))) = (\tau + O(t\tau_{pad}) + O(t\tau_{\mathbf{S}^{in}}))$-time adversary (A_3, B_3) that is t-valid for $\mathbf{f}^{\mathsf{left}/n}$, with collapsing-advantage $\varepsilon_3 = \varepsilon_6$.

We have

$$\varepsilon \le \varepsilon_6 + \varepsilon_7 \le \varepsilon_3 + \sqrt{\varepsilon_1} + (\ell-1)\varepsilon_2. \qquad \square$$

6.3 Using random oracles or random permutations

Lemma 32 *If $\mathbf{f} : \{0,1\}^{r+c} \to \{0,1\}^{r+c}$ is a random function, and $(A^{\mathbf{f}}, B^{\mathbf{f}})$ is t-valid for $\mathbf{f}^{\mathsf{left}/n}$ (or for $\mathbf{f}^{\mathsf{right}}$), makes $\le q$ queries to $\mathbf{f}$, and has collapsing-advantage ε, then $\varepsilon \in O(tq^{3/2}2^{-\min(n,r)/2})$ (or $\varepsilon \in O(tq^{3/2}2^{-c/2})$).*

Proof. Assume first that $A^{\mathbf{f}}$ is t-valid for $\mathbf{f}^{\mathsf{left}/n}$.

Let $\mathbf{h} : \{0,1\}^{r+c} \to \{0,1\}^{\min(n,r)}$ be a random function. Let $(\hat{A}^{\mathbf{h}}, \hat{B}^{\mathbf{h}})$ perform the following steps: It picks a random function $\mathbf{g} : \{0,1\}^{r+c} \to \{0,1\}^{r+c-\min(n,r)}$ (we do not care about the runtime of $\hat{A}, \hat{B}$, so it is not a problem that picking $\mathbf{g}$ takes exponential time). Then it simulates $(A^{\hat{\mathbf{f}}}, B^{\hat{\mathbf{f}}})$ where $\hat{\mathbf{f}}$ is the function defined by $\hat{\mathbf{f}}(x) := \mathbf{h}(x)\|\mathbf{g}(x)$. Note that an oracle query to $\hat{\mathbf{f}}$ can be implemented using a single oracle query to $\mathbf{h}$. So $\hat{A}$ still performs q oracle queries.

The joint distribution of $\hat{\mathbf{f}}$ and $\mathbf{h}$ is the same as that of $\mathbf{f}$ and $\mathbf{f}^{\mathsf{left}/n}$. Thus the advantage of $(\hat{A}^{\mathbf{h}}, \hat{B}^{\mathbf{h}})$ against $\mathbf{h}$ is the same as the advantage ε of $(A^{\mathbf{f}}, B^{\mathbf{f}})$ against $\mathbf{f}^{\mathsf{left}}$. And $(\hat{A}^{\mathbf{h}}, \hat{B}^{\mathbf{h}})$ is t-valid for $\mathbf{h}$.

Thus by Theorem 11, $\varepsilon \in O(tq^{3/2}2^{-\min(n,r)/2})$.

When $A^{\mathbf{f}}$ is t-valid for $\mathbf{f}^{\mathsf{right}}$, an analogous proof gives us $\varepsilon \in O(tq^{3/2}2^{-c/2})$. $\square$

Theorem 33 *Let $\mathbf{f} : \{0,1\}^{r+c} \to \{0,1\}^{r+c}$ be a random function. Let (A, B) be an adversary that is t-valid against $\mathbf{S}$ with output length n and makes $\leq q$ queries to $\mathbf{f}$. Then (A, B) has collapsing-advantage*

$$O\Big(\ell t(q + t\ell)^{3/2} 2^{-c/2} + t(q + t\ell)^{3/2} 2^{-\min(n,r)/2}\Big).$$

Proof. Let ε denote the collapsing-advantage of (A, B). By Theorem 31, there are:

- a $(q + O(t\ell))$-query adversary A_1 that finds a zero-preimage of $\mathbf{f}^{\mathsf{right}}$ with probability ε_1, and
- a $(q + O(t\ell))$-query adversary (A_2, B_2), t-valid against $\mathbf{f}^{\mathsf{right}}$, with collapsing-advantage ε_2, and
- a $(q + O(t\ell))$-query adversary (A_3, B_3) that is t-valid for $\mathbf{f}^{\mathsf{left}/n}$, with collapsing-advantage ε_3,

with

$$\varepsilon \leq \sqrt{\varepsilon_1} + (\ell - 1)\varepsilon_2 + \varepsilon_3. \tag{10}$$

By Lemma 19, $\varepsilon_1 \in O((q + t\ell)2^{-c/2})$. By Lemma 32, $\varepsilon_2 \in O(t(q + t\ell)^{3/2} 2^{-c/2})$ and $\varepsilon_3 \in O(t(q + t\ell)^{3/2} 2^{-\min(n,r)/2})$. Thus

$$\begin{aligned} \varepsilon &\in O\Big((q + t\ell)2^{-c/2} + \ell t(q + t\ell)^{3/2} 2^{-c/2} + t(q + t\ell)^{3/2} 2^{-\min(n,r)/2}\Big) \\ &= O\Big(\ell t(q + t\ell)^{3/2} 2^{-c/2} + t(q + t\ell)^{3/2} 2^{-\min(n,r)/2}\Big). \end{aligned}$$

□

Theorem 34 *Let $\mathbf{f} : \{0,1\}^{r+c} \to \{0,1\}^{r+c}$ be a random permutation. Let (A, B) be an adversary that is t-valid against $\mathbf{S}$ with output length n and makes $\leq q$ queries to $\mathbf{f}$. Then (A, B) has collapsing-advantage*

$$O\Big(\ell t(q + t\ell)^{3/2} 2^{-c/2} + t(q + t\ell)^{3/2} 2^{-\min(n,r)/2} + q^3 2^{-(r+c)}\Big).$$

Proof. It was shown by [Zha15] that a q-query adversary cannot distinguish a random function on $r + c$ bits from a random permutation on $r + c$ bits with probability greater than $O(q^3 2^{-(r+c)})$.

Thus the collapsing-advantage of $(A^{\mathbf{f}}, B^{\mathbf{f}})$ changes at most by $O(q^3 2^{-(r+c)})$ when we replace the random permutation $\mathbf{f}$ by a random function $\mathbf{f}$. Thus the advantage of $(A^{\mathbf{f}}, B^{\mathbf{f}})$ is at most $O(q^3 2^{-(r+c)})$ larger than the one given in Theorem 33. □

7 Quantum Attack

In the following we present a quantum collision-finding attack against the Sponge construction. The attack is based on a quantum collision-finding algorithm for any function (Theorem 36 below) that assumes access to a random oracle (RO).

The general working of our attack is to select a *suitable* function g, run a collision finding algorithm by Ambainis [Amb07] to obtain a collision for g, and finally turn this collision into a collision for the target Sponge. The *suitable* function in this context refers to the function giving the optimal result. First, we make a case distinction whether the length of the required collision n is smaller or bigger than the capacity c of the sponge. In case $n < c$, we simply search for an output collision in $\mathbf{S}$. In the other case $n \geq c$, it is more efficient to search for a right-collision, as these are collisions in a function with c bits of output and can be extended to arbitrary-length output collisions. Second, the function has to be selected or rather constructed in a way that allows for efficient iteration, in case a first run of the core algorithm does not succeed.

Our attack makes heavy use of the following quantum algorithm by Ambainis [Amb07].

Theorem 35 *([Amb07] Theorem 3) Let $\mathbf{g} : X \to Y$ be a function that has at least one collision. The size of the set X is M. Then there exists a constant k_{Amb} and a quantum algorithm making $k_{\text{Amb}} \cdot M^{2/3}$ quantum queries to $\mathbf{g}$ that finds a collision with probability at least 15/16.*

We note that [Amb07] also gives guarantees on the actual quantum running time and memory requirements of the quantum collision-finding algorithm. Concretely, there exist (small) constants $k'_{\text{Amb}}, k''_{\text{Amb}}$ such that the running time and quantum memory is at most $k'_{\text{Amb}} \cdot M^{2/3} \cdot \log^{k''_{\text{Amb}}}(M + |Y|)$. Therefore, all our results of this section which are stated in terms of query complexity also yield guarantees on the running time and memory, incurring the same blowup by a poly-logarithmic factor in the number of queries.

7.1 Quantum Collision Finding With Random Oracle

We start by showing how to use Ambainis' algorithm to generically find a collision in any function as long as we have access to a random oracle.

Theorem 36 *For finite sets A, B with $|B| \geq 3$ and $|A| \geq 40|B|$, and any function $h : A \to B$, there exists a quantum algorithm which requires access to a random oracle $\mathcal{H} : T \to A$ and outputs a collision of h with probability at least $1/8$ after at most $k_{\text{Amb}} \cdot |B|^{1/3}$ queries to h and at most $2k_{\text{Amb}} \cdot |B|^{1/3} + 2$ queries to $\mathcal{H}$ where k_{Amb} is the constant from Theorem 35.*

As noted after Theorem 35, there exist constants $k'_{\text{Amb}}, k''_{\text{Amb}}$ such that the running time and quantum memory of the collision-finding algorithm is at most $k'_{\text{Amb}} \cdot |B|^{1/3} \cdot \log^{k''_{\text{Amb}}}(|B|)$.

Proof. Let $T := \{1, 2, \ldots, \left\lceil \sqrt{|B|} \right\rceil + 1\}$ be a finite set of $\left\lceil \sqrt{|B|} \right\rceil + 1$ elements. In the description of our generic collision-finding algorithm COLL-RO below, we use the random oracle (RO) $\mathcal{H} : T \to A$. When repeating the algorithm in order to improve the success probability, we assume that a "fresh" random oracle is used in every run, which can be achieved using standard techniques such as prepending an iteration-counter to the inputs.

Algorithm 3 Algorithm COLL-RO

```
Input: $h : A \to B$ and access to random oracle $\mathcal{H} : T \to A$
Output: $m \neq \hat{m}$ such that $h(m) = h(\hat{m})$ or "fail"
Set $\mathbf{g} := h \circ \mathcal{H}$, $X := T$ with size $M = \left\lceil \sqrt{|B|} \right\rceil + 1$, $Y := B$
Run Ambainis' algorithm from Theorem 35 on $\mathbf{g}$, making $k_{\text{Amb}} \cdot M^{2/3}$ queries to $\mathbf{g}$.
If it outputs $(t, \hat{t})$
        Set $(m, \hat{m}) := (\mathcal{H}(t), \mathcal{H}(\hat{t}))$
        If $m \neq \hat{m}$, output $(m, \hat{m})$
Output "fail"
```

If Ambainis' algorithm succeeds in outputting a collision and $\mathcal{H}$ does not have any collisions, then we obtain a collision of h. Hence,

$$\begin{aligned} &\Pr[\text{COLL-RO outputs } m \neq \hat{m} \text{ such that } h(m) = h(\hat{m})] \\ &\geq \Pr[\text{Ambainis outputs a collision} \wedge \mathcal{H} \text{ does not have collisions}] \\ &\geq \Pr[\text{Ambainis outputs a collision}] - \Pr[\mathcal{H} \text{ has a collision}]. \end{aligned} \tag{11}$$

We can lower bound the first probability as follows:

$$\begin{aligned} &\Pr[\text{Ambainis outputs a collision}] = \Pr[\mathbf{g} \text{ has a collision} \wedge \text{Ambainis outputs a collision}] \\ &= \Pr[\mathbf{g} \text{ has a collision}] - \Pr[\mathbf{g} \text{ has a collision} \wedge \text{Ambainis does not output a collision}] \\ &\geq \Pr[\mathbf{g} \text{ has a collision}] - \frac{1}{16}\,. \end{aligned}$$

Note that $\mathbf{g}$ maps M messages to B. If these outputs were distributed independently and uniformly, we could lower bound the collision probability with a birthday bound. In our case, these outputs are not necessarily uniformly (due to h) but still independently (due to $\mathcal{H}$) distributed. It is proven in [KB73] that in this case, the same lower bound on the collision probability remains true. Therefore, it follows (e.g. from [KL14, Lemma A.16]) that

$$\Pr[\mathbf{g} \text{ has a collision}] \geq \frac{M(M-1)}{4 \cdot |B|} \geq \frac{(M-1)^2}{4 \cdot |B|} \geq \frac{\left(\left\lceil \sqrt{|B|} \right\rceil\right)^2}{4 \cdot |B|} \geq \frac{1}{4} . \tag{12}$$

In order to upper bound the second term of (11), observe that $\mathcal{H}$ maps M messages to independent and uniform elements in A. From a union bound (see, e.g. [KL14, Lemma A.15], noting that $M \leq \sqrt{2|A|}$), we get that

$$\Pr[\mathcal{H} \text{ has a collision}] \leq \frac{M^2}{2|A|} \leq \frac{(\sqrt{|B|}+2)^2}{2|A|} \leq \frac{|B| + 4\sqrt{|B|} + 4}{2|A|} \leq \frac{5|B|}{2|A|} \leq \frac{1}{16} . \tag{13}$$

The second-to-last inequality holds because $4\sqrt{|B|} + 4 \leq 4|B|$ for $|B| \geq 3$, and the last inequality is due to our assumption $40|B| \leq |A|$.

Combining (12) and (13), COLL-RO outputs a collision with probability at least

$$\frac{1}{4} - \frac{1}{16} - \frac{1}{16} = \frac{1}{8} .$$

The quantum circuit for $\mathbf{g} := h \circ \mathcal{H}$ makes one query to h and two queries to $\mathcal{H}$. Therefore, the total number of queries to h is at most $k_{\text{Amb}} \cdot M^{2/3} = k_{\text{Amb}} \cdot |B|^{1/3}$ and the number of queries to $\mathcal{H}$ is at most $2k_{\text{Amb}} \cdot |B|^{1/3} + 2$. □

We use this generic collision-finder to find sponge collisions.

Theorem 37 *Let $\mathbf{S}_{c,r,\mathbf{f},pad,n}(m)$ be a sponge construction with arbitrary block function $\mathbf{f}$. There exists a quantum algorithm* COLL-RO *making at most $q_{\mathbf{f}}$ quantum queries to $\mathbf{f}$ and $q_{\mathcal{H}}$ quantum queries to a random oracle $\mathcal{H}$.* COLL-RO *outputs colliding messages $m \neq \hat{m}$ such that $\mathbf{S}_{c,r,\mathbf{f},pad,n}(m) = \mathbf{S}_{c,r,\mathbf{f},pad,n}(\hat{m})$ with probability at least $1/8$, where $q_{\mathbf{f}} := 2k_{\text{Amb}} \cdot \min\{\frac{c+6+2r}{r}2^{c/3}, \frac{2n+6+3r}{r}2^{n/3}\}$, and $q_{\mathcal{H}} := 2k_{\text{Amb}} \cdot \min\{2^{c/3}, 2^{n/3}\} + 2$, where k_{Amb} is the constant from Theorem 35 and pad is any padding function which appends at most $2r$ bits.*

Typical padding functions do not append more than $r+1$ bits to the message, and are therefore covered by the theorem. Otherwise, the proof below can be easily modified to take longer paddings into account, resulting in increased factors in the expression of $q_{\mathbf{f}}$ above.

Proof. We make a case distinction whether the length n of the required collision n is smaller or bigger than the capacity c of the sponge. In case $n < c$, it is more efficient to directly search for an output collision in $\mathbf{S}$. In the other case $n \geq c$, it is more efficient to search for collisions in the right internal state, as these are collisions in a function with c bits of output and can be extended to arbitrary-length output collisions.

```
Algorithm 4 Algorithm SPONGE-COLL-RO
Input: Sponge parameters $n, c, r$ and access to RO $\mathcal{H}$
Output: $m \neq \hat{m}$ such that $\mathbf{S}_{c,r,\mathbf{f},pad,n}(m) = \mathbf{S}_{c,r,\mathbf{f},pad,n}(\hat{m})$ or "fail"
 1: If $n < c$
 2:     Set $h := \mathbf{S}$, domain $A := \{0,1\}^{n+6}$ and range $B := \{0,1\}^n$.
 3: If $n \geq c$
 4:     Set $h := \mathbf{f}^{\mathsf{right}} \circ \mathbf{S}^{in} \circ pad$, domain $A := \{0,1\}^{c+6}$ and range $B := \{0,1\}^c$
 5: Run COLL-RO from Theorem 36 on $h$, making $k_{\mathrm{Amb}} \cdot |B|^{1/3}$ queries to $h$
 6: If it outputs $(m, \hat{m})$
 7:     If $n < c$, output $(m, \hat{m})$
 8:     If $n \geq c$
 9:         Set $a := (\mathbf{f}^{\mathsf{left}} \circ \mathbf{S}^{in} \circ pad)(m) \oplus (\mathbf{f}^{\mathsf{left}} \circ \mathbf{S}^{in} \circ pad)(\hat{m})$
10:         Output $(pad(m) \| a, pad(\hat{m}) \| 0^r)$
11: Output "fail"
```

Let us analyze the case $n < c$. According to Theorem 36, COLL-RO outputs a collision with probability at least $\frac{1}{8}$ using at most $k_{\mathrm{Amb}} \cdot |B|^{1/3} = k_{\mathrm{Amb}} \cdot 2^{n/3}$ quantum queries to h. A single evaluation of $\mathbf{S}$ requires at most $2 \cdot \lceil \max\{|pad(m)| : m \in \{0,1\}^{n+6}\}/r \rceil \leq 2 \cdot (n + 6 + 2r)/r$ queries to the block function $\mathbf{f}$ in the absorbing phase and $2 \cdot \lceil n/r \rceil \leq 2(n + r)/r$ queries to $\mathbf{f}$ in the squeezing phase. Hence, one query to h requires at most $2 \cdot (2n + 6 + 3r)/r$ queries to the block function $\mathbf{f}$. Therefore, a collision in $\mathbf{S}$ can be found with at most $2k_{\mathrm{Amb}} \cdot \frac{2n+6+3r}{r} 2^{n/3}$ queries to $\mathbf{f}$.

In the other case $n \geq c$, the algorithm COLL-RO finds two messages $m \neq \hat{m}$ such that $(\mathbf{f}^{\mathsf{right}} \circ \mathbf{S}^{in} \circ pad)(m) = (\mathbf{f}^{\mathsf{right}} \circ \mathbf{S}^{in} \circ pad)(\hat{m})$ with probability at least $\frac{1}{8}$. Such a right-collision can then be extended to a full-state collision by appending to the padded colliding messages $pad(m)$ and $pad(\hat{m})$ one more suitably chosen message block resulting in $y := pad(m) \| a$ and $\hat{y} := pad(\hat{m}) \| 0^r$. As both $|y|$ and $|\hat{y}|$ are (possibly different) multiples of r, the same bits will be appended by pad according to our assumptions on the padding function. By the choice of a in Step 9, we have that $(\mathbf{f} \circ \mathbf{S}^{in} \circ pad)(y) = (\mathbf{f} \circ \mathbf{S}^{in} \circ pad)(\hat{y})$, i.e. the full states collide and therefore, all n output bits produced from this state will coincide. The algorithm makes $k_{\mathrm{Amb}} \cdot |B|^{1/3} = k_{\mathrm{Amb}} \cdot 2^{c/3}$ queries to $h := \mathbf{f}^{\mathsf{right}} \circ \mathbf{S}^{in} \circ pad$. In this case, one query to h requires at most $2 \cdot (c + 6 + 2r)/r$ queries to the block function $\mathbf{f}$. Therefore, a collision in $(\mathbf{f} \circ \mathbf{S}^{in} \circ pad)$ can be found with at most $2k_{\mathrm{Amb}} \cdot \frac{c+6+2r}{r} 2^{c/3}$ queries to $\mathbf{f}$, resulting in the claimed bound. □

Index

Symbol index

References

[Amb07] Andris Ambainis. “Quantum walk algorithm for element distinctness”. In: *SIAM Journal on Computing* 37.1 (2007), pp. 210–239.

[Aum+10] Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and María Naya-Plasencia. “Quark: A Lightweight Hash”. In: *CHES 2010*. Vol. 6225. LNCS. Springer, 2010, pp. 1–15. ISBN: 978-3-642-15030-2. DOI: `10.1007/978-3-642-15031-9_1`.

[Ber+12] Thierry P. Berger, Joffrey D’Hayer, Kevin Marquet, Marine Minier, and Gaël Thomas. “The GLUON Family: A Lightweight Hash Function Family Based on FCSRs”. In: *Africacrypt 2012*. Berlin, Heidelberg: Springer, 2012, pp. 306–323. ISBN: 978-3-642-31410-0. DOI: `10.1007/978-3-642-31410-0_19`.

[Ber+07] Guido Bertoni, J. Daemen, Michaël Peeters, and Gilles van Assche. *Sponge functions*. Ecrypt Hash Workshop, `http://sponge.noekeon.org/SpongeFunctions.pdf`. May 2007.

[Ber+08] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles van Assche. "On the Indifferentiability of the Sponge Construction". In: *Eurocrypt 2008*. Vol. 4965. LNCS. Berlin, Heidelberg: Springer, 2008, pp. 181–197. ISBN: 978-3-540-78966-6. DOI: 10.1007/978-3-540-78967-3_11.

[Bog+13] Andrey Bogdanov, Miroslav Knezevic, Gregor Leander, Deniz Toz, Kerem Varici, and Ingrid Verbauwhede. "SPONGENT: The Design Space of Lightweight Cryptographic Hashing". In: *IEEE Transactions on Computers* 62.10 (2013), pp. 2041–2053. ISSN: 0018-9340. DOI: 10.1109/TC.2012.196.

[Bon+11] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. "Random oracles in a quantum world". In: *Asiacrypt 2011*. Seoul, South Korea: Springer, 2011, pp. 41–69. ISBN: 978-3-642-25384-3. DOI: 10.1007/978-3-642-25385-0_3.

[Boy+98] Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. "Tight Bounds on Quantum Searching". In: *Fortschritte der Physik* 46.4-5 (1998). Eprint is arXiv:quant-ph/9605034, pp. 493–505. ISSN: 1521-3978. DOI: 10.1002/(SICI)1521-3978(199806)46:4/5<493::AID-PROP493>3.0.CO;2-P.

[BHT97] Gilles Brassard, Peter Hoyer, and Alain Tapp. "Quantum algorithm for the collision problem". In: *arXiv preprint quant-ph/9705002* (1997).

[CLS06] Scott Contini, Arjen K. Lenstra, and Ron Steinfeld. "VSH, an Efficient and Provable Collision-Resistant Hash Function". In: *Eurocrypt 2006*. Springer, 2006, pp. 165–182. ISBN: 978-3-540-34547-3. DOI: 10.1007/11761679_11.

[GPP11] Jian Guo, Thomas Peyrin, and Axel Poschmann. "The PHOTON Family of Lightweight Hash Functions". In: *Crypto 2011*. Springer, 2011, pp. 222–239. ISBN: 978-3-642-22792-9. DOI: 10.1007/978-3-642-22792-9_13.

[HM96] Shai Halevi and Silvio Micali. "Practical and Provably-Secure Commitment Schemes from Collision-Free Hashing". English. In: *Crypto '96*. Vol. 1109. LNCS. Springer, 1996, pp. 201–215. ISBN: 978-3-540-61512-5. DOI: 10.1007/3-540-68697-5_16.

[HRS16] Andreas Hülsing, Joost Rijneveld, and Fang Song. "Mitigating Multi-target Attacks in Hash-Based Signatures". In: *PKC 2016*. Springer, 2016, pp. 387–416. ISBN: 978-3-662-49384-7. DOI: 10.1007/978-3-662-49384-7_15.

[KL14] J. Katz and Y. Lindell. *Introduction to Modern Cryptography, Second Edition*. Chapman & Hall/CRC Cryptography and Network Security Series. Taylor & Francis, 2014. ISBN: 9781466570269.

[KB73] William Knight and D. M. Bloom. "E2386". In: *The American Mathematical Monthly* 80.10 (1973), pp. 1141–1142. ISSN: 00029890, 19300972. DOI: 10.2307/2318556. URL: http://www.jstor.org/stable/2318556.

[Nat15] National Institute of Standards and Technology (NIST). *Secure Hash Standard (SHS)*. FIPS PUBS 180-4. 2015. DOI: 10.6028/NIST.FIPS.180-4.

[NC10] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. 10th anniversary. Cambridge: Cambridge University Press, 2010. ISBN: 978-1107002173.

[NIS14] NIST. *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. Draft FIPS 202. Available at http://csrc.nist.gov/publications/drafts/fips-202/fips_202_draft.pdf. 2014.

[Unr16a] Dominique Unruh. "Computationally binding quantum commitments". In: *Eurocrypt 2016*. LNCS. Springer, 2016, pp. 497–527. ISBN: 978-3-662-49896-5. DOI: 10.1007/978-3-662-49896-5_18.

[Unr16b] Dominique Unruh. “Collapse-binding quantum commitments without random oracles”. In: *AsiaCrypt 2016*. Vol. 10032. LNCS. Springer, 2016, pp. 166–195. DOI: 10.1007/978-3-662-53890-6_6.

[Zha12] Mark Zhandry. “How to Construct Quantum Random Functions”. In: *FOCS 2013*. Online version is IACR ePrint 2012/182. Los Alamitos, CA, USA: IEEE Computer Society, 2012, pp. 679–687. DOI: 10.1109/FOCS.2012.37.

[Zha15] Mark Zhandry. “A note on the quantum collision and set equality problems”. In: *Quantum Information & Computation* 15.7&8 (2015), pp. 557–567. URL: http://www.rintonpress.com/xxqic15/qic-15-78/0557-0567.pdf.