Quantum Indifferentiability of SHA-3

Jan Czajkowski*

QuSoft, University of Amsterdam

May 12, 2021

Abstract

In this paper we prove quantum indifferentiability of the sponge construction instantiated with random (invertible) permutations. With this result we bring the post-quantum security of the standardized SHA-3 hash function to the level matching its security against classical adversaries. To achieve our result, we generalize the compressed-oracle technique of Zhandry (Crypto'19) by defining and proving correctness of a compressed permutation oracle. We believe our technique will find applications in many more cryptographic constructions.

Contents

*j.czajkowski@uva.nl

1 Introduction

With efforts to build a quantum computer on the rise, cryptography resilient to quantum attackers (i.e. attackers equipped with a large-scale quantum computer) becomes a necessity. *Post-quantum* cryptography [BBD09], in other words quantum-safe (classical) cryptography, is considered to be important enough that the NIST organized the process to standardize post-quantum cryptographic schemes [NIS17].

A prominent primitive, used in many of the submissions to the NIST standardization process, is the SHA-3 hash function [NIS14]. The cryptographic construction used in the design of this hash function, is the sponge construction [Ber+07]. One of the reasons the sponge construction was chosen as the backbone of this standardized hash function, was the strong security guarantees offered by the proof of its *indifferentiability* from a random oracle [Ber+08]. Indifferentiability [MRH04] is a notion similar to the Random Oracle Model (ROM) [BR93]—where we claim a hash function is a random oracle—but much better tailored to cryptographic constructions. To prove indifferentiability, we assume the internal function called by the construction to be ideal (a uniformly random function or permutation) but also give the adversary access to it.

For five years now, the research community has worked to rigorously prove post-quantum security of the sponge construction [Cza+18; CHS19; Cza+19]. One of the goals of this line of research was to match the security level guaranteed by proofs valid for classical adversaries. In [Cza+19] the authors managed to prove quantum indifferentiability, but with one shortcoming, they only treated uniformly random internal functions. Now is it important to note, that SHA-3 is based on an invertible *permutation*, this design choice is dictated, among other reasons, by the cryptographic practice showing that it is much easier to design secure permutations than one-way functions. In our work, we finally manage to close the gap between classical and post-quantum security of SHA-3: We prove quantum indifferentiability of the sponge construction instantiated with random permutations.

The natural proof technique suitable for indifferentiability proofs is that of lazy-sampling. In a recent work, Zhandry developed the compressed-oracle technique [Zha19], which is the quantum version of lazy-sampling. Until now, however, the quantum techniques did not cover lazy-sampling of random (invertible) permutations. Changing that, is the main technical contribution of this paper. Basing on the quantum game-playing framework of [Cza+19], we define a quantum compressed permutation oracle. Our result introduces errors, but they are smaller than the leading terms in the distinguishing advantage of the indifferentiability adversary. Arguably, the total bound on the distinguishing advantage that we prove is tight. Although we do not show an algorithm with a matching query complexity, our bound on the distinguishing advantage[1] of $O(\sqrt{q^3/2^c})$ gives rise to security of up to $\Omega\left(2^{c/3}\right)$ queries. A generic collision finding algorithm for the sponge construction, e.g. from [Cza+18], is expected to run in $O\left(2^{c/3}\right)$ queries to the internal function. In [Ros21], Rosmanis (independently from this paper) also proposed a technique for dealing with invertible random permutations in a quantum world.

In section 2 we give more details on the methods we use to achieve our results. The compressed permutation oracle is defined and proven indistinguishable from a random permutation in section 3. The main result of this paper, quantum indifferentiability of the sponge construction instantiated with random permutations, is located in section 4. A reader mostly interested in the application of our new technique can go directly to section 4, as we provide a high level introduction to the main ingredients of the proof.

[1] By distinguishing advantage, we mean the absolute value of the difference of the probabilities that the adversary outputs 1 when interacting with the construction or a random oracle.

2 Preliminaries

In this section we start by going over notation used in this paper. Later we present some key concepts that we use to achieve our results. A summary of symbols used throughout the paper can be found in the Symbol Index.

In this paper we assume the reader has basic knowledge of quantum computing. If any concept is not familiar though, we refer to [NC10; Wol11]. A quantum state is a unit vector in a Hilbert space $|\psi\rangle \in \mathbb{C}^d$. We often refer to vectors of norm smaller than 1 as states as well. In what follows we denote the Euclidean norm of a vector by $\||\psi\rangle\|$. The trace norm of an operator A on a Hilbert space is defined as the trace of the absolute value of the operator: $\|\mathsf{A}\|_1 := \mathrm{Tr}\sqrt{\mathsf{A}^\dagger\mathsf{A}}$. Subspaces of tensor products of several Hilbert spaces are referred to by superscripts, so if $\mathcal{H}_D = \bigotimes_{i=1}^q \mathcal{H}_{X_i} \otimes \mathcal{H}_{Y_i}$, then D^X refers to $\bigotimes_{i=1}^q \mathcal{H}_{X_i}$, and same for Y. We write A^D to denote that A acts on register D.

We write $[N] := \{0, 1, \ldots, N-1\}$ for the set of size N. Whenever we consider sets of bits $\{0,1\}^n$, we denote the bitwise XOR by $\oplus$.

When discussing queries to oracles that have interfaces for forward and backward direction, we denote the vector of s queries by

$$\vec{x} \in \mathcal{Q}_s := (\{+,-\} \times [N])^s \,. \tag{1}$$

We also use set operations[2] on these vectors, by identifying them with sets with elements from $\vec{x}$. We consider $\vec{x}$ to be always sorted according to values from $[N]$ in a raising fashion and contain no repetitions, so set operations are unambiguous. The i-th tuple in $\vec{x}$ is denoted by x_i, by x_i^I we denote the first part of the pair x_i that is the *interface* marked by $\{+,-\}$ and by x_i^V we denote the second part, i.e., the *value* from $[N]$.

The set of databases of size s is defined as

$$\mathcal{D}_s := \{((x_1,y_1),(x_2,y_2),\ldots,(x_s,y_s)) \in ([N]\times[N])^s : \forall i,j \neq i, x_i \neq x_j\} \,. \tag{2}$$

For $D \in \mathcal{D}_s$ we denote the i-th pair by D_i, the first element of the pair is D_i^X and the second D_i^Y. For $D \in \mathcal{D}_s$ and $\vec{\eta} \in \mathcal{Q}_s$ we define $\vec{\eta} \circ D := \sum_{i=1}^s \eta_i^V \cdot D_i^{\eta_i^I}$, where $D_i^{\eta_i^I} = \begin{cases} D_i^Y & \text{if } \eta_i^I = +, \\ D_i^X & \text{if } \eta_i^I = - \end{cases}$. When referring to tuples containing elements of $\vec{x} \in \mathcal{Q}_s$ we also write $D(\vec{x})$ and $D(x)$ for a particular $x \in \vec{x}$. Note that $x \in \{+,-\} \times [N]$, so contains the information about the interface.

By D^X we denote the set of D_i^X for all $1 \leq i \leq s$. The set of *injective* partial functions and its size are denoted by

$$\mathcal{I}(s \mid D) := \{D' \in \mathcal{D}_s : D^X \cap D'^X = \emptyset, \forall i,j \neq i, y_i' \neq y_j' \wedge y_i' \notin D^Y\}, \tag{3}$$

$$|\mathcal{I}(s)| = (N)_s := N(N-1)\cdots(N-s+1), \tag{4}$$

where in the definition of $(N)_s$ we use the notation $\mathcal{I}(s) := \mathcal{I}(s \mid \emptyset)$.

By $x \leftarrow \mathsf{A}$ we denote sampling x from a distribution or getting the output of a randomized algorithm. By square brackets we denote (classical or quantum) oracle access to some algorithm, we also use A^{H} if the oracle is denoted by a more confined symbol.

An object that we make heavy use of in this paper is the random oracle:

$$\mathsf{R} : \{0,1\}^* \times \mathbb{N} \to \{0,1\}^*. \tag{5}$$

A random oracle grants access to a function sampled from distribution $\mathfrak{R}$ on functions $\{0,1\}^* \times \mathbb{N} \to \{0,1\}^*$, that is defined as follows: To sample a function $h \leftarrow \mathfrak{R}$ we

- choose g uniformly at random from $\{g : \{0,1\}^* \to \{0,1\}^\infty\}$, where by $\{0,1\}^\infty$ we denote the set of infinitely long bitstrings,

[2] Such as the union $\cup$, intersection $\cap$, or subtraction $\setminus$.

- for each $(x, \ell) \in \{0,1\}^* \times \mathbb{N}$ set $h(x, \ell) := \lfloor g(x) \rfloor_\ell$, that is output the first ℓ bits of the output of g.

2.1 Indifferentiability

Whenever, the hash function is constructed out of a public internal function (like in the case of, e.g., SHA-2 [NIS15] and SHA-3 [NIS14]), a security notion that includes the adversary's access to the internal (and publicly specified) function and aims at showing that the hash function behaves as a uniformly random function, is *indifferentiability* [MRH04].

In the setting of indifferentiability the adversary is given access to two *interfaces*, the *public* interface, denoted by "pub" and the *private* interface, denoted by "priv". Indifferentiability of a system C is phrased in terms of a distinguishing game, where the adversary distinguishes between the real and the ideal world. In the real world, the private interface is the construction calling the internal function, denoted as $\mathsf{C}^{\mathrm{priv}}[\mathsf{C}^{\mathrm{pub}}]$, and the public interface is the internal function $\mathsf{C}^{\mathrm{pub}}$. In the ideal world, where the construction is often replaced by a random oracle, the private interface is just $\mathsf{R}_k^{\mathrm{priv}}$, but the public interface is provided by a *simulator* that replaces the internal function, she is also given access to the ideal system and the interface is $\mathsf{S}[\mathsf{R}_k^{\mathrm{pub}}]$. In case of R being a random oracle both the public and the private interfaces of R are equal and just return outputs of a random function.

Definition 1 (Indifferentiability [MRH04])**.** *A cryptographic system* C *is* (q, ε)-indifferentiable *from* R*, if there is an efficient* (*classical or quantum*) simulator S *and a negligible function* ε *such that for any efficient* (*classical or quantum*) distinguisher D *with binary output* (*0 or 1*) *the advantage*

$$\left| \mathbb{P}\left[1 \leftarrow \mathsf{D}[\mathsf{C}_k^{\mathrm{priv}}[\mathsf{C}_k^{\mathrm{pub}}], \mathsf{C}_k^{\mathrm{pub}}]\right] - \mathbb{P}\left[1 \leftarrow \mathsf{D}[\mathsf{R}_k^{\mathrm{priv}}, \mathsf{S}[\mathsf{R}_k^{\mathrm{pub}}]]\right] \right| \leq \varepsilon(k)\,, \tag{6}$$

where k *is the security parameter. The distinguisher makes at most* q (*classical or quantum*) *queries.*

2.2 Compressed-Oracles Technique

The compressed oracle technique was developed by Zhandry [Zha19]. The key idea of this technique is to purify a quantum-accessible random oracle so all the randomness is kept in a quantum register. The standard oracle for a uniformly random function is defined as:

$$\mathsf{StO}|x, y\rangle_{A^{XY}} \sum_{f \in \mathcal{F}} \frac{1}{\sqrt{|\mathcal{F}|}} |f\rangle_F = \sum_{f \in \mathcal{F}} \frac{1}{\sqrt{|\mathcal{F}|}} |x, y + f(x)\rangle_{A^{XY}} |f\rangle_F, \tag{7}$$

where $x, y \in [N]$, addition is done modulo N, and f is the full truth table of the function from $\mathcal{F} := \{f : [N] \to [N]\}$[3]. The update procedure StO just adds the correct row of f to adversary's A^Y register. The crucial fact is that an adversary A interacting with StO acting on the oracle register from Eq. (7) has zero probability of distinguishing it from $f \xleftarrow{\$} \mathcal{F}$. Whenever we consider $\{0,1\}^n$ instead of $[N]$, addition is the bitwise XOR.

A compressed oracle CStO is an oracle with the oracle register F compressed to register D that just holds the outputs to queries asked by the adversary A. The update procedure in this case is more complicated but allows to keep a superposition of small database of the form $D = ((x_1, y_1), \ldots, (x_q, y_q))$, where q is the bound on the maximal number of quantum queries made by A. In the case A made $s < q$ queries, the last $q - s$ entries hold $\perp$ in the X-type part of the entries. More details on this technique can be found in [Zha19; LZ19; Cza+19; HI19; Chu+20; Unr21]. Similarly to StO, the distinguishing advantage between CStO and $f \xleftarrow{\$} \mathcal{F}$ is 0.

[3] In general, the standard and compressed oracle can be defined for any set of functions.

A change of the adversary's basis, that can be done by applying the quantum Fourier transform $\mathsf{QFT}_N|y\rangle := \frac{1}{\sqrt{N}}\sum_{\eta\in[N]}\omega_N^{\eta\cdot y}|\eta\rangle$, where $\omega_N := e^{\frac{2\pi i}{N}}$ is the N-th root of unity, gives rise to the phase oracle:

$$\mathsf{PhO}|x,\eta\rangle_{AXY}\sum_{f\in\mathcal{F}}\frac{1}{\sqrt{|\mathcal{F}|}}|f\rangle_F = \sum_{f\in\mathcal{F}}\frac{1}{\sqrt{|\mathcal{F}|}}\omega_N^{\eta\cdot f(x)}|x,\eta\rangle_{AXY}|f\rangle_F, \tag{8}$$

when we consider $\{0,1\}^n$ instead of $[N]$, the product in the phase factor is the inner product of bitstrings. Similarly, by changing the adversary's interface to the Fourier basis, we get the compressed phase oracle CPhO. For readers less familiar with quantum accessible oracles, we note that a quantum query does not always increase the number of input-output pairs A has knowledge of. Say A queries (x,η_x) at some point. If A later queries the same input with a different η, then she updates her knowledge of $f(x)$. If $\eta = -\eta_x$ then she might even "forget" the query x, or—if she interacts with a compressed oracle—remove x from the database.

Whenever we talk about CStO and CPhO, we consider them to be oracles for the uniform distribution. Moreover, when providing the subscript $[N]$, by $\mathsf{CPhO}_{[N]}$ we mean that functions are distributed uniformly from $\{f : [N] \to [N]\}$. We often omit the subscript when it is clear from the context. More on compressed oracles for different distributions can be found in [Cza+19].

Building upon the compressed oracle technique the authors of [Cza+19] developed a quantum game-playing framework. The key object of this framework, that we make heavy use of, is the *punctured* compressed oracle. Puncturing is defined as performing a binary measurement of the oracle register D of a compressed oracle after every query done by A. As long as the probability of one of the outputs (say 0) is almost 1, the puncturing does not give the adversary a major distinguishing advantage. The measurement, however, allows us to argue about the adversary's behavior conditioned on the database being of some specific form.

The measurements in punctured oracles are binary projective measurement. We project to databases that fulfill a given *relation*. A relation is a subset of all possible databases of size bounded by q. An important relation, that we use in the reminder of this paper, is the *collision* relation:

$$R_{\text{coll}} := \left\{ D \in \bigcup_{s=2}^{q} ([N]\times[N])^s : \exists i,j\neq i, x_i \neq x_j, y_i = y_j \right\}, \tag{9}$$

where $D = ((x_1,y_1),\ldots,(x_s,y_s))$. Measuring a relation on the quantum compressed oracle includes checking the actual size of the database, so it omits the entries starting with $\perp$.

A formal definition of a compressed oracle H punctured on relation R, denoted by $\mathsf{H}\setminus R$, is stated below. More details on the above concepts can be found in [Cza+19].

Definition 2 (Punctured compressed oracle $\mathsf{H}\setminus R$, Definition 9 in [Cza+19])**.** *Let* H *be a compressed oracle and* R *a relation on its database. The punctured compressed oracle* $\mathsf{H}\setminus R$ *is equal to* H*, except that* R *is measured after every query. By* Find *we denote the event that* R *outputs* 1 *at least once among all queries.*

2.3 Game-Playing Proofs

Game-playing proofs are a common framework used in cryptography. Originally developed by Bellare and Rogaway [BR06] the main concept is that if two games—in other words, interactive protocols—are identical until a "bad" event happens, then they are indistinguishable except for the probability that this event occurs.

Using recently developed techniques, this reasoning can be applied to *quantum* interactive protocols as well. Below, we present the classical and the quantum game-playing framework. We use the statements from below in our indifferentiability proofs in section 4.

2.3.1 Classical Game-Playing Proofs

Two games are considered to be *identical-until-bad* if they are syntactically identical except for code that follows after setting Bad to true. Bad is a special flag, we also write Bad to denote the event "Bad set to true". The main ingredient of the classical game-playing framework is the fundamental game-playing lemma:

Lemma 3 (Fundamental lemma of game-playing, Lemma 2 in [BR06]). *Let* G *and* H *be identical-until-bad games and let* A *be an adversary that outputs a bit* b*. Then*

$$\left|\mathbb{P}[1 \leftarrow \mathrm{A}^{\mathsf{H}}] - \mathbb{P}[1 \leftarrow \mathrm{A}^{\mathsf{G}}]\right| \leq \mathbb{P}[\mathrm{Bad} = 1 : \mathrm{A}^{\mathsf{G}}]. \tag{10}$$

2.3.2 Quantum Game-Playing Proofs

The quantum game-playing framework, developed in [Cza+19], makes use of two powerful proof techniques. One being the compressed oracle technique by Zhandry [Zha19], the other being the One-way To Hiding (O2H) lemma by Unruh [Unr14; AHU19]. The former (which we described in section 2.2), together with the idea of puncturing oracles, is used to define quantum games. The latter provides the quantum counterpart of the fundamental game playing lemma.

Here we present the version of the O2H lemma from [AHU19] that was later repurposed to treat punctured compressed oracles in [Cza+19].

Lemma 4 (Compressed oracle O2H, Theorem 10 in [Cza+19]). *Let* R_1 *and* R_2 *be relations on the database of a quantum oracle* H*. Let* A *be an oracle algorithm making* q *quantum queries, then*

$$\left|\mathbb{P}[1 \leftarrow \mathrm{A}^{\mathsf{H}\setminus R_1}] - \mathbb{P}[1 \leftarrow \mathrm{A}^{\mathsf{H}\setminus R_1 \cup R_2}]\right| \leq \sqrt{(q+1)\mathbb{P}[\mathrm{Find} : \mathrm{A}^{\mathsf{H}\setminus R_1 \cup R_2}]}, \tag{11}$$

where Find *is the event that measuring* $R_1 \cup R_2$ *succeeds.*

An important comment that relates to the main subject of this work is that the above lemma works just as well for *invertible* oracles, such as random permutations. Following the proof, in [Cza+19], we notice that it relies only on measuring the database and the general interface of compressed oracles. These aspects do not change in our definition of the compressed permutation oracle. More discussion on which, can be found in section 3.

The notion of identical-until-bad games can also be brought to the world of quantum compressed oracles.

Definition 5 (Almost identical oracles, Definition 11 in [Cza+19]). *Let* H *and* G *be compressed oracles and* R_i*,* $i = 1, 2$ *relations on their databases. We call the oracles* $\mathsf{H} \setminus R_1$ *and* $\mathsf{G} \setminus R_2$ almost identical *if they are equal conditioned on the events* $\neg\mathrm{Find}_1$ *and* $\neg\mathrm{Find}_2$ *respectively, i.e. for any quantum algorithm* A

$$\mathbb{P}[1 \leftarrow A^{\mathsf{H}\setminus R_1} \mid \neg\mathrm{Find}_1] = \mathbb{P}[1 \leftarrow A^{\mathsf{G}\setminus R_2} \mid \neg\mathrm{Find}_2]. \tag{12}$$

Distinguishing advantage for almost-identical oracles can be bounded using the following lemma.

Lemma 6 (Distinguishing almost identical punctured oracles, Lemma 12 in [Cza+19]). *If* $\mathsf{H}\setminus R_1$ *and* $\mathsf{G} \setminus R_2$ *are almost identical according to Def.5 then*

$$\left|\mathbb{P}[1 \leftarrow A^{\mathsf{H}\setminus R_1}] - \mathbb{P}[1 \leftarrow A^{\mathsf{G}\setminus R_2}]\right| \leq 2\mathbb{P}[\mathrm{Find}_1 : A^{\mathsf{H}\setminus R_1}] + 2\mathbb{P}[\mathrm{Find}_2 : A^{\mathsf{G}\setminus R_2}]. \tag{13}$$

2.4 Sponge Construction

The sponge construction SPONGE [Ber+07] is prominently used to design the standardized hash function SHA-3 [NIS14]. It can be instantiated with a function or a permutation $f : \{0,1\}^{r+c} \to \{0,1\}^{r+c}$. The construction is divided in two phases: In the absorbing phase the padded input message is XOR-ed block by block (blocks are r-bits long) into the first r bits of the internal state (consisting of $r + c$ bits) of SPONGE_f and after each block f is applied to the whole state. In the following squeezing phase the first r bits are output and f is applied, this repeats until ℓ bits of output is generated. This distinction into the first r bits and the last c bits calls for special notation. For a state $s \in \{0,1\}^{r+c}$, the first r bits are called the *outer* part and are denoted by s^{out}, the last c bits are called the *inner* part and are denoted by s^{inn}. By $\text{PAD} : \{0,1\}^* \to (\{0,1\}^r)^*$ (where $\{0,1\}^* := \bigcup_{n=0}^{\infty}\{0,1\}^n$ and by $(\{0,1\}^r)^*$ we denote the strings consisting of an arbitrary number of r-bit blocks) we denote a padding function. The requirements on PAD in the sponge construction are that it is an efficiently computable injection such that $|\text{PAD}(m)| \geq r$ and that the last bit of $\text{PAD}(m)$ is never 0. The *rate* of the sponge is r and the *capacity* is c, these key parameters, together with the internal function and the padding function give rise to the hash function $\text{SPONGE}_f[\text{PAD}, r, c] : \{0,1\}^* \times \mathbb{N} \to \{0,1\}^*$. In what follows, by $|p|_r$ we denote the number of r-bit blocks in p and by $\lfloor p \rfloor_i$ we denote the i-th r-bit block of p. The formal definition of the sponge construction is presented in Algorithm 1.

Algorithm 1: $\text{SPONGE}_f[\text{PAD}, r, c]$

Input : $m \in \{0,1\}^*, \ell \geq 0$
Output: $z \in \{0,1\}^\ell$

$p := \text{PAD}(m)$
$s := (0,0) \in \{0,1\}^r \times \{0,1\}^c$
for $i = 1$ to $|p|_r$ **do** `// Absorbing phase`
  $s := (s^{\text{out}} \oplus \lfloor p \rfloor_i, s^{\text{inn}})$
  $s := f(s)$
$z := s^{\text{out}}$ `// Squeezing phase`
while $|z| < \ell$ **do**
  $s := f(s)$
  $z := z \| s^{\text{out}}$
Output $\lfloor z \rfloor_\ell$ `// Output the first` ℓ `bits of` z

It was already noticed in [Ber+07] that one can associate a graph $G = (\mathcal{V}, \mathcal{E})$ with the internal function of a sponge. This graph is called the *sponge graph* and plays an important role in indifferentiability proofs of the sponge construction [Ber+08; Cza+19]. The set of nodes of this graph is $\mathcal{V} := \{0,1\}^{r+c}$, it corresponds to all possible states of the sponge. A directed edge connects any two nodes (s,t) whenever $f(s) = t$, meaning there are 2^{r+c} edges in $\mathcal{E}$. From each node starts exactly one edge. A *supernode* groups the nodes with the same inner-part, thus we have 2^c supernodes and each such supernode consists of 2^r nodes. Edges between nodes are also edges between supernodes.

The initial state of any call of SPONGE is the $(0^r, 0^c)$ node. This node is called the *root*. The first r-bit block $\lfloor p \rfloor_1$ of the padded message $p = \text{PAD}(m)$ is then added to the outer-part of the state and fed to the internal function $f(\lfloor p \rfloor_1 \| 0^c) = s_2$. This evaluation of f defines the edge $(\lfloor p \rfloor_1 \| 0^c, s_2) \in \mathcal{E}$. In the absorbing phase this operation is repeated until we run out of blocks of p. In the squeezing phase we no longer modify the state, in other words just add 0^r to the outer part. Note that knowing just p and G we can get to the last node traversed by $\text{SPONGE}_f(m)$. This leads to the definition of a *sponge path*.

Definition 7 (Sponge path, Definition 3 in [Ber+08])**.** *First, the empty string is a sponge path to*

the node $0^r\|0^c$. Then, if p is a sponge path to node $s = s^{\text{out}}\|s^{\text{inn}}$ and there is an edge $(s^{\text{out}} \oplus a\|s^{\text{inn}}, t)$ in the sponge graph G, $p' = p\|a$ is a sponge path to node t.

Given the above definition, let us say that p forms a sponge path to s, then we define a function

$$\mathsf{SpPath}(s, G) := p. \tag{14}$$

Output of the above function is the input to the construction $\textsc{Sponge}_f(., \ell = r)$ that yields the output s^{out}.

In indifferentiability proofs, the simulator has access to the *simulator graph*. The graph kept by the simulator differs from the sponge graph discussed above by the number of edges in it. As the simulator lazy samples the internal function f the set of edges $\mathcal{E}$ grows by at most one edge per a single adversary's query. Other than that, everything that has been said above holds. We refer to the simulator graph G as just the (sponge) graph whenever it is clear from context.

A supernode is called *rooted* if there is a path (a regular path that is just a set of edges connected by the end-start nodes) leading to it that starts in the root (the 0^c-supernode). The set $\mathcal{R}$ is the set of all rooted supernodes in G. By $\mathcal{U}$ we denote the set of supernodes with a node with an outgoing edge.

A simulator graph is called *saturated* if $\mathcal{R} \cup \mathcal{U} = \{0,1\}^c$. It means that for every inner state in $\{0,1\}^c$ there is an edge in G that leads to it from 0^c (the root) or leads from it to another node. Saturation will be important in the proof of indifferentiability as the simulator wants to pick outputs of f without colliding inner parts (so not in $\mathcal{R}$) and making the path leading from 0^c to the output longer by just one edge (so not in $\mathcal{U}$).

3 Compressed Oracle for Random Permutations

In this section we define the compressed oracle for *invertible* random permutation. Our approach is rather simple, we use a compressed oracle for the uniform distribution and puncture it on collisions. The collision-free databases can be easily inverted and the same update procedure as for the forward direction provides the backward-direction oracle.

The hard part in this section is proving that our compressed permutation oracle is in fact indistinguishable from a full permutation oracle. The full oracle keeps a superposition of all bijective functions and responds to both forward and backward queries.

Our approach to proving indistinguishability relies on a relatively simple auxiliary *good* state. The good state is generated by the adversary interacting with the compressed oracle for the uniform distribution, after the interaction we project the database to contain only collision-free databases. Such state is easy to handle and we can prove its closeness of the state coming from the full permutation oracle. We then reuse the proof from [Cza+19] to show closeness of the good state and the compressed oracle state.

In the end of this section we also provide bounds on the probability of Find for punctured compressed permutation oracles. These results are important for our indifferentiability proof.

3.1 Definition of the Compressed Permutation Oracle

In this section we define the compressed permutation oracle. This oracle accepts forward and backward queries to the lazily-sampled permutation. Before we state the formal definition, though, we discuss inverse queries in general.

Inverse queries Before we state the definition of the compressed permutation oracle, let us define the interface of two-directional functions. We assume the query register, prepared by the adversary, consists of three parts: the *interface* register A^I, the *query-input* register A^X, and the *query-output* register A^Y. The former part holds values from $\{+,-\}$, corresponding to forward

and backward queries respectively. The latter two registers hold the actual query. The important thing to note is that an adversary can make queries in two directions *in superposition*. In the rest of the paper (unless explicitly stated) we allow the adversary to make both forward and backward queries.

To treat forward and backward queries consistently, we change the full truth tables to include the inputs: $|y_1, \dots, y_N\rangle$ changes to $|(x_1, y_1) \dots, (x_N, y_N)\rangle$. In this picture it is easier to write down queries in both directions. Let us consider an example, the adversary queries x_1 in the forward direction and y_2 in the backward direction:

$$\sum_{D\in\mathcal{I}(N)} \frac{1}{\sqrt{N!}} \omega_N^{\eta_1 f(x_1)} \omega_N^{\eta_2 f^{-1}(y_2)} |D\rangle_F = \sum_{D\in\mathcal{I}(2)} \frac{1}{\sqrt{(N)_2}} \omega_N^{\eta_1 \cdot D(x_1)^Y} \omega_N^{\eta_2 \cdot D(y_2)^X}$$
$$\sum_{D'\in\mathcal{I}(N-2|D)} \frac{1}{\sqrt{(N-2)!}} |D \cup D'\rangle_F. \tag{15}$$

Using the $\circ$ operation defined in section 2 we can also write the phase factor as $\omega_N^{\vec{\eta}\circ D}$. To get the query outputs we do the following: First we fix the position (because the database is sorted according to the first entry) and answer with the second entry, second we fix the value of the second entry and take the position (or the value of the first entry). Note that there are $N-1$ databases with (x, y_1) fixed, and for every one of these there are $N-2$ databases with also (x_2, y) fixed. If we fix a pair (x_2, y), then we have a superposition of databases with y in different positions.

Definition of CPerO The core of the compressed permutation oracle is a compressed oracle for the uniform distribution punctured on the collision relation $\mathsf{CPhO}_{[N]} \setminus R_{\text{coll}}$. To allow for inverse access we define a unitary that flips the database, so the outputs become inputs and vice versa. Then queries are handled with the same $\mathsf{CPhO}_{[N]} \setminus R_{\text{coll}}$ procedure.

In more detail, we provide access to the inverse of the permutation by flipping the database: treat the content of D^Y as inputs and D^X as outputs and sort by D^Y, we call the appropriate unitary Flip. Note that Flip works as anticipated only if both in D^Y and D^X there are no collisions; The definition of Flip can be easily extended to a full unitary by keeping the order unchanged within a tuple with colliding outputs. We thereby obtain a (superposition) database where the entries (y_i, x_i) are sorted in a rising order according to the first value. A more detailed description of Flip is:

1. Controlled on D^Y copy D^X to a fresh D'^Y and arrange in an increasing order according to values of y_i in D^Y.
2. Controlled on D^X and D'^Y copy D^Y to a fresh D'^X in the new order.
3. Controlled on D' erase the old D: Take the smallest x_i in D'^Y and subtract it from the first register of D^X, also subtract the corresponding y_i, and so on.

A formal definition of the compressed permutation oracle $\mathsf{CPerO}_{[N]}$ is:

Definition 8. *The compressed permutation oracle for permutations $f : [N] \to [N]$ is a unitary that applies one of the two following operations controlled on the adversary's A^I register holding $a \in \{+, -\}$. Depending on whether $a = +$ or $a = -$ the procedure is :*

$$\mathsf{CPerO}_{[N]} := \mathsf{CPhO}_{[N]} \setminus R_{\text{coll}} \text{ or} \tag{16}$$
$$\mathsf{CPerO}^{-1}_{[N]} := \mathsf{Flip} \circ \left(\mathsf{CPhO}_{[N]} \setminus R_{\text{coll}}\right) \circ \mathsf{Flip} \tag{17}$$

respectively. Flip *is defined above and* $\mathsf{CPhO}_{[N]}$ *is the compressed phase oracle for the uniform distribution over* $\{f : [N] \to [N]\}$.

3.2 Indistinguishability of CPerO and PerO

By PerO we denote the full oracle for a uniform distribution over permutations, technically it is defined in the same way as PhO from Eq. (8) but we always use it with the oracle register holding a superposition of permutations:

$$\mathsf{PerO}_{[N]}|a,x,\eta\rangle_{A^{IXY}} \sum_{f\in\mathcal{P}} \frac{1}{\sqrt{|\mathcal{P}|}}|f\rangle_F = \sum_{f\in\mathcal{P}} \frac{1}{\sqrt{|\mathcal{P}|}}\omega_N^{\eta\cdot f^a(x)}|x,\eta\rangle_{A^{IXY}}|f\rangle_F, \tag{18}$$

where $a \in \{+,-\}$, $f^a(x) = \begin{cases} f(x) & \text{if } a = + \\ f^{-1}(x) & \text{if } a = - \end{cases}$, and $\mathcal{P} := \{f : [N] \to [N] : f \text{ is a bijection}\}$.
The main statement of this section is the correctness of the compressed permutation oracle.

Theorem 9 (Correctness of $\mathsf{CPerO}_{[N]}$)**.** *No quantum algorithm* A *making at most* q *(where* $q \in O(\sqrt[3]{N})$*) quantum queries, in the forward or backward direction, can distinguish* $\mathsf{CPerO}_{[N]}$ *from the full permutation oracle* $\mathsf{PerO}_{[N]}$ *except for a small error:*

$$\left|\mathbb{P}\left[1 \leftarrow \mathrm{A}^{\mathsf{PerO}_{[N]}}\right] - \mathbb{P}\left[1 \leftarrow \mathrm{A}^{\mathsf{CPerO}_{[N]}}\right]\right| \leq 17\frac{q^2}{N-8q}. \tag{19}$$

Before getting into the proof of the main statement we suggest the reader to go over the observation presented in Appendix A. The observation is a theorem stating the distinguishability advantage of any adversary trying to discern a uniformly random function from a random permutation using a single quantum query. The approach to the proof of Theorem 15 is quite related to that from of proof of Theorem 9.

In the following three paragraphs we define and provide some useful notation on the two important classes of quantum states that we will make heavy use of.

The permutation states First we go over the full permutation oracle. The state that results from A interacting with the full permutation oracle PerO is

$$\begin{aligned}|\Psi^{\mathsf{PerO}}\rangle_{AF} &:= \sum_{\vec{x}}|\Psi^{\mathsf{PerO}}(\vec{x})\rangle_{AF} := \sum_{\vec{x}}\sum_{\vec{\eta}}\alpha_{\vec{x},\vec{\eta}}|\psi(\vec{x},\vec{\eta})\rangle_A \\ &\underbrace{\sum_{D\in\mathcal{I}(s)}\frac{1}{\sqrt{(N)_s}}\omega_N^{\vec{\eta}\circ D}}_{(*)\text{: effective queries}}\sum_{D'\in\mathcal{I}(N-s|D)}\frac{1}{\sqrt{(N-s)!}}|D\cup D'\rangle_F,\end{aligned} \tag{20}$$

where $|\Psi^{\mathsf{PerO}}(\vec{x})\rangle$ are sub-normalized and $|\psi(\vec{x},\vec{\eta})\rangle$ are normalized. Register A holds all of the adversary's state, this includes her work register, query register, and any other auxiliary registers she decides to use, we denote it by $\psi(\vec{x},\vec{\eta})$. In the above state $\vec{x} = (x_1,\ldots,x_s) \in \mathcal{Q}_s$ is the set of *effective* queries. As querying a quantum oracle can result in "un-querying" a value, by $\vec{x}$ we denote only the queried values that have a non-trivial phase factor dependent on the queried function after all q queries of A. For these queries we also have $\eta^V_{x_i} \neq 0$ for all $x_i \in \vec{x}$. The size of $\vec{x}$ is s and $s \leq q$. The crucial concept of this paragraph are the effective queries, we define $|\Psi^{\mathsf{PerO}}(\vec{x})\rangle$ as the branches of superposition that queried only $\vec{x}$.

Another way of defining effective queries is by considering histories of queries. We track the (forward and backward) queries made by the adversary A, write down the whole state of A, and separate all branches of superposition. In every such branch there is a set of queries made by the adversary, this is what we call a history. Effective queries are derived by summing all η_x corresponding to a single x in the query history. By combining the values we get the set of queries with $\eta \neq 0$. Given effective queries $\vec{x}$, the combined adversary's registers and their amplitudes are the normalized $|\psi(\vec{x},\vec{\eta})\rangle_A$ and $\alpha_{\vec{x},\vec{\eta}}$ respectively.

The permutation state is the partial trace of the above state

$$\rho_{\mathsf{PerO}} := \mathrm{Tr}_F |\Psi^{\mathsf{PerO}}\rangle_{AF}\langle\Psi^{\mathsf{PerO}}| = \sum_{\vec{x},\vec{x}',\vec{x}''} \rho_{\mathsf{PerO}}(\vec{x}\cup\vec{x}', \vec{x}\cup\vec{x}''), \tag{21}$$

where in the above sums we have $\vec{x}' \cap \vec{x} = \emptyset$, $\vec{x}'' \cap \vec{x} = \emptyset$, and $\vec{x}'' \cap \vec{x}' = \emptyset$. Moreover $\vec{x}\cup\vec{x}'$ and $\vec{x}\cup\vec{x}''$ range over $\mathcal{Q}_{\leq q}$, where $\mathcal{Q}_{\leq q} := \bigcup_{s\leq q} \mathcal{Q}_s$. The state is defined as

$$\rho_{\mathsf{PerO}}(\vec{x}\cup\vec{x}', \vec{x}\cup\vec{x}'') := \mathrm{Tr}_F |\Psi^{\mathsf{PerO}}(\vec{x}\cup\vec{x}')\rangle_{AF}\langle\Psi^{\mathsf{PerO}}(\vec{x}\cup\vec{x}'')|, \tag{22}$$

where $|\Psi^{\mathsf{PerO}}(\vec{x}\cup\vec{x}')\rangle_{AF}$ is the branch of the superposition in $|\Psi^{\mathsf{PerO}}\rangle$ corresponding to *effective* queries in $\vec{x}\cup\vec{x}'$.

Below we define *well-formed* matrices. The intuition behind well-formed matrices is that they are parts of quantum states resulting from the interaction of an adversary with the full permutation oracle. They are natural parts of density matrices corresponding to A interacting with PerO.

Definition 10. *A vector is* well-formed *if it is of the form from Eq.* (20)*. Coefficients $\alpha_{\vec{x},\vec{\eta}}$ can be such that the overall state is sub-normalized.*

A matrix is well-formed *if it is of the form from Eq.* (22) *with the vectors being well formed.*

The good states In what follows we write $\vec{x}$ to denote the effective queries. The state $|\Psi^{\mathrm{Good}}\rangle_{AD}$ corresponds to the adversary's state after q queries. The size of the database is $s = |\vec{x}|$. After q queries s can range from 0 to q and the joint state of A and the oracle can be a superposition over different database sizes. We define the good state as:

$$\begin{aligned}|\Psi^{\mathrm{Good}}\rangle_{AD} := \sum_{\vec{x},\vec{\eta}} \alpha_{\vec{x},\vec{\eta}} |\psi(\vec{x},\vec{\eta})\rangle_A \sum_{D\in\mathcal{I}(s)} \frac{1}{\sqrt{(N)_s}} \omega_N^{\vec{\eta}\circ D} \sum_{y_{s+1},\ldots,y_q\in[N]} \frac{1}{\sqrt{N^{q-s}}} \\ |(x_1,y_1),\ldots,(x_s,y_s),(\perp,y_{s+1}),\ldots,(\perp,y_q)\rangle_D.\end{aligned} \tag{23}$$

The good state is also defined as running the adversary and giving her access to the compressed standard oracle $\mathsf{CPhO}_{[N]}$. After q queries we project register D to collision free databases. Good state is in fact a class of states generated by interaction with CPhO and projecting to collision-free databases.

Proof of Theorem 9. We first simplify the distinguishing advantage to include interaction with the punctured oracle conditioned on ¬Find:

$$\begin{aligned}&\left|\mathbb{P}\left[1 \leftarrow \mathrm{A}^{\mathsf{PerO}}\right] - \mathbb{P}\left[1 \leftarrow \mathrm{A}^{\mathsf{CPerO}}\right]\right| \\ &\leq \left|\mathbb{P}\left[1 \leftarrow \mathrm{A}^{\mathsf{PerO}}\right] - \mathbb{P}\left[1 \leftarrow \mathrm{A}^{\mathsf{CPerO}} \mid \neg\mathrm{Find}\right]\mathbb{P}\left[\neg\mathrm{Find}\right]\right| + \mathbb{P}\left[\mathrm{Find}\right] &(24)\\ &\leq \left|\mathbb{P}\left[1 \leftarrow \mathrm{A}^{\mathsf{PerO}}\right] - \mathbb{P}\left[1 \leftarrow \mathrm{A}^{\mathsf{CPerO}} \mid \neg\mathrm{Find}\right]\right| + \mathbb{P}\left[\mathrm{Find}\right]. &(25)\end{aligned}$$

In the first step above we write down the right term as $\mathbb{P}[1 \leftarrow \mathrm{A}^{\mathsf{CPerO}}] = \mathbb{P}[1 \leftarrow \mathrm{A}^{\mathsf{CPerO}} \mid \neg\mathrm{Find}]\mathbb{P}\left[\neg\mathrm{Find}\right] + \mathbb{P}[1 \leftarrow \mathrm{A}^{\mathsf{CPerO}} \mid \mathrm{Find}]\mathbb{P}\left[\mathrm{Find}\right]$. Next we use the triangle inequality and bound $\mathbb{P}[1 \leftarrow \mathrm{A}^{\mathsf{CPerO}} \mid \mathrm{Find}] \leq 1$. Without loss of generality we can choose A such that outputs 1 with greater probability when she interacts with CPerO. If an adversary outputs 1 with greater probability when she interacts with PerO we instead consider A' that operates exactly like A with the only difference that she flips the output. The second inequality is a result of this reasoning and bounding $\mathbb{P}\left[\neg\mathrm{Find}\right] \leq 1$.

By $\Phi^{AD}_{\mathsf{CPerO}}$ we denote the joint state of A and the database D conditioned on ¬Find ($\Phi^{A}_{\mathsf{CPerO}}$ is the same state after a partial trace over D). To generate the final answer, any quantum algorithm

has to perform a quantum measurement, let us denote by Q_1 is the measurement corresponding to A outputting 1. Then we have

$$\begin{aligned}&\left|\mathbb{P}\left[1 \leftarrow \mathrm{A}^{\mathsf{PerO}}\right] - \mathbb{P}\left[1 \leftarrow \mathrm{A}^{\mathsf{CPerO}} \mid \neg\mathrm{Find}\right]\right| \\ &= \left|\mathrm{Tr}\left(\mathsf{Q}_1 \rho^A_{\mathsf{PerO}}\right) - \mathrm{Tr}\left(\mathsf{Q}_1 \Phi^A_{\mathsf{CPerO}}\right)\right| \leq \frac{1}{2}\left\|\rho^A_{\mathsf{PerO}} - \Phi^A_{\mathsf{CPerO}}\right\|_1, \end{aligned} \tag{26}$$

where the inequality above comes from the fact that trace distance gives the optimal measurement for distinguishing two states [NC10].

In the following we introduce $\rho^{AD}_{\mathrm{Good}} := \mathrm{Tr}_D\left(|\Psi^{\mathrm{Good}}\rangle_{AD}\langle\Psi^{\mathrm{Good}}|\right)$, the good state generated by A. Using the bound from Eq. (26) we bound Eq. (25) as follows:

$$\text{Eq. (25)} \leq \frac{1}{2}\left\|\rho^A_{\mathsf{PerO}} - \Phi^A_{\mathsf{CPerO}}\right\|_1 + \mathbb{P}\left[\mathrm{Find}\right] \tag{27}$$

$$\leq \frac{1}{2}\left\|\rho^A_{\mathsf{PerO}} - \rho^A_{\mathrm{Good}}\right\|_1 + \frac{1}{2}\left\|\rho^{AD}_{\mathrm{Good}} - \Phi^{AD}_{\mathsf{CPerO}}\right\|_1 + \mathbb{P}\left[\mathrm{Find}\right] \leq 17\frac{q^2}{N-8q}, \tag{28}$$

where in the second inequality we use the triangle inequality and the fact that trace distance is decreasing under partial trace. The final bound is a simplification (using the assumption that $q \in O(\sqrt[3]{N})$) of bounds from Lemma 11 and Lemma 13 that are proven in the reminder of this section. □

In the following lemma we prove a bound on the trace distance of the full permutation state and the good state.

Lemma 11. *The permutation and good states are close in trace distance:*

$$\frac{1}{2}\left\|\rho^A_{\mathsf{PerO}} - \rho^A_{\mathrm{Good}}\right\|_1 \leq \frac{1}{2}\frac{9q^2}{N-8q}. \tag{29}$$

Proof. Our initial claim is that the $\alpha_{\vec{x},\vec{\eta}}$ coefficients and $|\psi(\vec{x},\vec{\eta})\rangle$ states in Eqs. (20) and (23) are the same. Given an adversary A, after q queries we fix the database that she interacted with in a single branch of the superposition. The amplitudes are the same because they only depend on A and D.

In this lemma we compare reduced density matrices. If we take two matrices with the same effective queries, then they also necessarily interacted with the same random function. This fact is true for the states and distributions over databases we defined earlier in this section. Next we observe that ρ_{Good} is diagonal in the effective queries. Moreover the diagonal terms of ρ_{PerO} are equal to the good state. The only difference in the two states are the off-diagonal parts of ρ_{PerO}. The main idea of our proof is that one can "reduce" the off-diagonal part to the diagonal part of a state generated by a slightly different adversary. This reduction introduces the factor resulting in the final bound on the distance of the states.

Diagonal terms Let us consider two worlds an adversary A can be in. In the ideal world she interacts with the full permutation oracle, resulting in the joint state from Eq. (20). In the real world she interacts with a compressed oracle for the uniform distribution. In the real world, after A's interaction we project the database register to collision-free databases, resulting in the good state from Eq. (23).

One can see that the outputs in the part of the oracle register corresponding to the effective queries are the same in both cases. However, by saving the inputs in register D^X, the density matrix reduced to the adversarial register will differ. The good density matrix ρ_{Good} will consist solely of terms diagonal in effective queries $\vec{x}$. Note however that these—diagonal in queries—terms in ρ_{PerO} are exactly the same. One can see it by taking the state from Eq. (20), fixing the branch of superposition corresponding to a single $\vec{x}$, and calculating $\rho_{\mathsf{PerO}}(\vec{x},\vec{x})$. Tracing over

the oracle register F results in a sum over $D(\vec{x}) \in \mathcal{I}(s)$. For $|\Psi^{\text{Good}}\rangle$ from Eq. (23) with a fixed $\vec{x}$ the same operations will lead to the same matrix (the coefficients are the same because we take the same A and the sum over $\vec{y}$ is the same for diagonal terms). The sum over $\vec{x}$ is the whole diagonal. As the good state has no off-diagonal terms but gives the same diagonal terms we have

$$\sum_{\vec{x}} \rho_{\mathsf{PerO}}(\vec{x},\vec{x}) = \rho_{\text{Good}}. \tag{30}$$

Reducing the off-diagonals Below we state and prove the claim that a well-formed off-diagonal matrix can be reduced to a well-formed matrix that is "closer" to the diagonal.

Claim 12. *Any* well-formed *matrix* (*i.e. following Definition 10*) $\rho_{\mathsf{PerO}}(\vec{x}\cup\vec{x}'\cup\{x\},\vec{x}\cup\vec{x}'')$*, for any* $\vec{x},\vec{x}',\vec{x}'',\{x\}$ *with empty intersections, can be reduced to a sum of well-formed matrices that do not include the effective query* x*:*

$$\begin{aligned}\rho_{\mathsf{PerO}}(\vec{x}\cup\vec{x}'\cup\{x\},\vec{x}\cup\vec{x}'') = -\frac{s}{N-s}\Big(&\widetilde{\rho}^{\text{UPD}}_{\mathsf{PerO}}(\vec{x}\cup\vec{x}',\vec{x}\cup\vec{x}'')\\ &+\widetilde{\rho}^{\text{ADD}}_{\mathsf{PerO}}(\widehat{\vec{x}}\cup\vec{x}',\widehat{\vec{x}}\cup\overline{\vec{x}}'')+\widetilde{\rho}^{\text{REM}}_{\mathsf{PerO}}(\overline{\vec{x}}\cup\vec{x}',\overline{\vec{x}}\cup\widehat{\vec{x}}'')\\ &+\widetilde{\rho}^{\text{REM}}_{\mathsf{PerO}}(\vec{x}\cup\overline{\vec{x}}',\vec{x}\cup\vec{x}'')+\widetilde{\rho}^{\text{REM}}_{\mathsf{PerO}}(\vec{x}\cup\vec{x}',\vec{x}\cup\overline{\vec{x}}'')\Big),\end{aligned} \tag{31}$$

where $s := |\vec{x}\cup\vec{x}'\cup\vec{x}''|$*,* $\widehat{\vec{x}}$ *is the set* $\vec{x}$ *with a single element added, and* $\overline{\vec{x}}$ *is* $\vec{x}$ *with a single element removed. The same element is added and removed when* $\widehat{\vec{x}}$ *and* $\overline{\vec{x}}$ *appears in both inputs to* $\widetilde{\rho}_{\mathsf{PerO}}$*. Overline and hat notation works the same for other sets. Detailed definition of the new matrices can be found in the proof.*

Proof. Let us inspect $\rho_{\mathsf{PerO}}(\vec{x}\cup\vec{x}'\cup\{x\},\vec{x}\cup\vec{x}'')$ in more detail. Below we have $t := |\vec{x}|$, $t' := |\vec{x}'|$, and $t'' := |\vec{x}''|$ (and $s = t+t'+t''$):

$$\begin{aligned}&\rho_{\mathsf{PerO}}(\vec{x}\cup\vec{x}'\cup\{x\},\vec{x}\cup\vec{x}'') = \sum_{\vec{\eta},\vec{\eta}'}\alpha_{\vec{x}\cup\vec{x}'\cup\{x\},\vec{\eta}}\bar{\alpha}_{\vec{x}\cup\vec{x}'',\vec{\eta}'}\\ &|\psi(\vec{x}\cup\vec{x}'\cup\{x\},\vec{\eta})\rangle_A\langle\psi(\vec{x}\cup\vec{x}'',\vec{\eta}')|\frac{1}{(N)_{t+t'+t''}}\sum_{D(\vec{x})\in\mathcal{I}(t)}\omega_N^{\vec{\eta}_{\vec{x}}\circ D(\vec{x})}\bar{\omega}_N^{\vec{\eta}'_{\vec{x}}\circ D(\vec{x})}\\ &\sum_{D(\vec{x}')\in\mathcal{I}(t'|D(\vec{x}))}\omega_N^{\vec{\eta}_{\vec{x}'}\circ D(\vec{x}')}\sum_{D(\vec{x}'')\in\mathcal{I}(t''|D(\vec{x})\cup D(\vec{x}'))}\bar{\omega}_N^{\vec{\eta}'_{\vec{x}''}\circ D(\vec{x}'')}\\ &\frac{1}{N-t-t'-t''}\sum_{D(x)\in\mathcal{I}(1|D(\vec{x})\cup D(\vec{x}')\cup D(\vec{x}''))}\omega_N^{\eta_x\circ D(x)}.\end{aligned} \tag{32}$$

To reduce the effective query of x we evaluate the last sum in the above formula

$$\begin{aligned}&\frac{1}{N-t-t'-t''}\sum_{D(x)\in\mathcal{I}(1|D(\vec{x})\cup D(\vec{x}')\cup D(\vec{x}''))}\omega_N^{\eta_x\circ D(x)} = \frac{1}{N-t-t'-t''}\\ &\cdot\left(\underbrace{N\delta_{\eta_x,0}}_{=0}-\sum_{i=1}^{t}\omega_N^{\eta_x\circ D(x_i)}-\sum_{i'=1}^{t'}\omega_N^{\eta_x\circ D(x'_{i'})}-\sum_{i''=1}^{t''}\omega_N^{\eta_x\circ D(x''_{i''})}\right).\end{aligned} \tag{33}$$

We note that the result of evaluating the sum are changes to other effective queries held by the state.

Given the above observation, we define the matrices from the statement of the lemma. We can write the initial matrix as

$$\begin{aligned}&\rho_{\mathsf{PerO}}(\vec{x}\cup\vec{x}'\cup\{x\},\vec{x}\cup\vec{x}'')\\ &= -\frac{s}{N-s}\text{Tr}_{FA^{\widetilde{W}}}|\widetilde{\Psi}^{\mathsf{PerO}}(x;\vec{x}\cup\vec{x}')\rangle\langle\widetilde{\Psi}^{\mathsf{PerO}}(\vec{x}\cup\vec{x}'')|,\end{aligned} \tag{34}$$

where $|\widetilde{\Psi}^{\mathsf{PerO}}(x;\vec{x}\cup\vec{x}')\rangle$ and $|\widetilde{\Psi}^{\mathsf{PerO}}(\vec{x}\cup\vec{x}'')\rangle$ are well-formed vectors. The former vector is defined as the branch of the superposition from Eq. (20) corresponding to $\vec{x}\cup\vec{x}'$ with the effective queries (the part marked with $(*)$) changed to

$$\sum_{\tilde{x}\in\vec{x}\cup\vec{x}'\cup\vec{x}''}\frac{1}{\sqrt{s}}|\tilde{x}\rangle_{A^{\widetilde{W}}}\sum_{D(\vec{x}\cup\vec{x}'\cup\vec{x}'')\in\mathcal{I}(s)}\frac{1}{\sqrt{(N)_s}}\omega_N^{\vec{\eta}\circ D(\vec{x}\cup\vec{x}'\cup\vec{x}'')}\omega_N^{\eta_x\circ D(\tilde{x})}. \tag{35}$$

The latter vector is the original $|\Psi^{\mathsf{PerO}}(\vec{x}\cup\vec{x}'')\rangle$ with the additional register $\sum_{\tilde{x}\in\vec{x}\cup\vec{x}'\cup\vec{x}''}\frac{1}{\sqrt{s}}|\tilde{x}\rangle_{A^{\widetilde{W}}}$. Note that without that the trace would be ill-defined. The numerator s in Eq. (31) comes from the normalization of the new register.

To prove that $|\widetilde{\Psi}^{\mathsf{PerO}}(x;\vec{x}\cup\vec{x}')\rangle$ is well-formed we define a new adversary $\widetilde{\mathrm{A}}$ based on the initial A. Our vector is the result of subtracting the states generated by the two adversaries. If a vector is a branch of the superposition generated by a valid adversary, then naturally it is well-formed.

The new adversary $\widetilde{\mathrm{A}}$ first prepares a fresh register $\frac{1}{\sqrt{s}}\sum_{\tilde{x}\in\vec{x}\cup\vec{x}'\cup\vec{x}''}|\tilde{x}\rangle_{A^{\widetilde{W}}}$. Next, after every query performed by A, the new $\widetilde{\mathrm{A}}$ immediately applies $\mathsf{PerO}^\dagger$ controlled on A^X holding x, so uncomputes the query x. Next (immediately after the uncomputation of x), $\widetilde{\mathrm{A}}$ applies $\mathsf{PerO}^{A^{\widetilde{W}Y}}$ controlled on A^X being x. To sum up, whenever the old adversary would query x, the new adversary queries a superposition of $\tilde{x}$ instead. It is crucial to note that if there was no effective query to x then $\widetilde{\mathrm{A}}$ generates the same state as A.

Say that $\widetilde{\mathrm{A}}$ produces the vector $|\widetilde{\Psi}^{\mathsf{PerO}}(\vec{x}\cup\vec{x}')\rangle$ and A the vector $|\Psi^{\mathsf{PerO}}(\vec{x}\cup\vec{x}')\rangle$, then

$$|\widetilde{\Psi}^{\mathsf{PerO}}(x;\vec{x}\cup\vec{x}')\rangle := |\widetilde{\Psi}^{\mathsf{PerO}}(\vec{x}\cup\vec{x}')\rangle - |\Psi^{\mathsf{PerO}}(\vec{x}\cup\vec{x}')\rangle. \tag{36}$$

Now that we showed that $|\widetilde{\Psi}^{\mathsf{PerO}}(x;\vec{x}\cup\vec{x}')\rangle$ is a well-formed state we define the different matrices in the right hand side of Eq. (31). The new vector is a sum of five well-formed vectors, they are well-formed because they are just branches of the superposition constituting a well-formed vector:

$$\begin{aligned}|\widetilde{\Psi}^{\mathsf{PerO}}(x;\vec{x}\cup\vec{x}')\rangle &= |\widetilde{\Psi}^{\mathsf{PerO}}_{\mathrm{UPD}}(x;\vec{x}\cup\vec{x}')\rangle + |\widetilde{\Psi}^{\mathsf{PerO}}_{\mathrm{ADD}}(\widehat{\vec{x}}\cup\vec{x}')\rangle\\ &+ |\widetilde{\Psi}^{\mathsf{PerO}}_{\mathrm{REM}}(x;\overline{\vec{x}}\cup\vec{x}')\rangle + |\widetilde{\Psi}^{\mathsf{PerO}}_{\mathrm{REM}}(x;\vec{x}\cup\overline{\vec{x}}')\rangle + |\widetilde{\Psi}^{\mathsf{PerO}}_{\mathrm{REM}}(\vec{x}\cup\vec{x}')\rangle.\end{aligned} \tag{37}$$

The vectors in the right-hand side of Eq. (37) come from updating one of the other queries, adding a new query to a query from $\vec{x}''$, or removing a query from $\vec{x}$ or $\vec{x}'$. The branch corresponding to $\eta_x \neq -\eta_{\tilde{x}}$ (for any $\tilde{x}\in\vec{x}\cup\vec{x}'$) is denoted by $|\widetilde{\Psi}^{\mathsf{PerO}}_{\mathrm{UPD}}(x;\vec{x}\cup\vec{x}')\rangle$. The branch where $\tilde{x}\in\vec{x}''$ and $\eta_x\neq\eta_{\tilde{x}}$ is marked with the subscript ADD. State $|\widetilde{\Psi}^{\mathsf{PerO}}_{\mathrm{REM}}(\vec{x}\cup\vec{x}')\rangle$ denote the branch where $\tilde{x}\in\vec{x}''$ and $\eta_x=\eta_{\tilde{x}}$. The branches where for some $\tilde{x}\in\vec{x}\cup\vec{x}'$ we have $\eta_x=-\eta_{\tilde{x}}$ are denoted by the states in Eq. (37) with subscript REM. The first two REM states correspond to removing a query to $\vec{x}$ or $\vec{x}'$.

The matrices in the right hand side of Eq. (31) are defined by taking the partial trace over registers $FA^{\widetilde{W}}$ of the states from Eq. (37) multiplied by $\langle\widetilde{\Psi}^{\mathsf{PerO}}(\vec{x}\cup\vec{x}'')|$. □

By using Claim 12 recursively on the new well-formed matrices from Eq. (31), we reduce every matrix $\rho_{\mathsf{PerO}}(\vec{x}\cup\vec{x}'\cup\{x\},\vec{x}\cup\vec{x}'')$ to a well-formed matrix diagonal in effective queries. After the multiple application of Claim 12 we end up with a sum of matrices (diagonal in $\vec{x}$) multiplied by different coefficients coming from Eq. (31). We combine all matrices corresponding to the same "path" (so multiplied by the same coefficient) into one matrix denoted by $\widetilde{\rho}^{\vec{p}}_{\mathsf{PerO}}(\vec{x},\vec{x})$ (more details on this matrix are provided below). The diagonal matrices that we get are well-formed. This is because they are a sum of well-formed vectors coming from Claim 12, that are reduced off-diagonal terms of matrices with distinct effective queries. Intuitively speaking, we sum the branches we picked apart in the proof of Claim 12.

More concretely we have

$$\rho_{\mathsf{PerO}} - \rho_{\mathrm{Good}} = \sum_{\vec{x},\vec{x}',\vec{x}''} \rho_{\mathsf{PerO}}(\vec{x} \cup \vec{x}', \vec{x} \cup \vec{x}'') \tag{38}$$

$$= \sum_{d=1}^{2q-2} \sum_{t=1}^{t_{\max}(d)} \sum_{\vec{p}} C_{t,\vec{p}} \sum_{\vec{x}:|\vec{x}|=t} \widetilde{\rho}^{\vec{p}}_{\mathsf{PerO}}(\vec{x},\vec{x}), \tag{39}$$

where in the first sum $\vec{x} \cap \vec{x}' = \emptyset$, $\vec{x} \cap \vec{x}'' = \emptyset$, $\vec{x}' \cap \vec{x}'' = \emptyset$, and $\vec{x}' \cup \vec{x}'' \neq \emptyset$. The sum over d goes over the depth of recursion. The bound in the sum over t is just the maximal number of effective queries on the diagonal, this number depends on d but is bounded by q so we do not calculate it strictly. By $\vec{p} \in [2q]^d$ we denote the *path* of the number of effective queries traversed in the recursive applications of Claim 12. By $C_{t,\vec{p}}$ we denote the coefficient coming from Eq. (31):

$$C_{t,\vec{p}} := \left(\prod_{s=t}^{t+d-1} \frac{\vec{p}_{s+1-t}}{N - \vec{p}_{s+1-t}} \right). \tag{40}$$

To give some intuition for the values stored in $\vec{p}$, let us consider a path that involves only steps of length one, so that in Eq. (31) we never get the last two matrices. Then the coefficient is $\left(\prod_{s=t}^{t+d-1} \frac{s}{N-s}\right)$. By step length we mean the number of effective queries that is reduced in a single application of Claim 12. If on the other hand, we only look at branches where the queries are removed (so just the last two matrices), the coefficient is $\frac{t+d-1}{N-(t+d-1)} \cdot \frac{t+d-3}{N-(t+d-3)} \cdot \frac{t+d-5}{N-(t+d-5)} \cdots$. An important note is that as any step can be of length either one or two. So for a fixed number of recursive steps d, there are at most 2^d distinct vectors $\vec{p}$.

The matrix

$$\widetilde{\rho}^{\vec{p}}_{\mathsf{PerO}}(\vec{x},\vec{x}) = \mathrm{Tr}_{FA\widetilde{W}} \left(|\widetilde{\Psi}^{\mathsf{PerO}}_1(\vec{p};\vec{x})\rangle\langle\widetilde{\Psi}^{\mathsf{PerO}}_2(\vec{p};\vec{x})| + |\widetilde{\Psi}^{\mathsf{PerO}}_2(\vec{p};\vec{x})\rangle\langle\widetilde{\Psi}^{\mathsf{PerO}}_1(\vec{p};\vec{x})| \right) \tag{41}$$

for some $|\widetilde{\Psi}^{\mathsf{PerO}}_1(\vec{p};\vec{x})\rangle$ and $|\widetilde{\Psi}^{\mathsf{PerO}}_2(\vec{p};\vec{x})\rangle$ that are well-formed vectors holding a sum of vectors from Claim 12. The parameter $\vec{p}$ signifies the vectors contributing to Eq. (41), it is not crucial to keep track of it but we supply it for completeness.

Final bound Now we can define

$$|\widetilde{\Psi}^{\mathrm{Good}}(\vec{p};\vec{x})\rangle := |\widetilde{\Psi}^{\mathrm{Good}}_1(\vec{p};\vec{x})\rangle + |\widetilde{\Psi}^{\mathrm{Good}}_2(\vec{p};\vec{x})\rangle. \tag{42}$$

First of all, $|\widetilde{\Psi}^{\mathrm{Good}}_i(\vec{p};\vec{x})\rangle$ for $i \in \{1,2\}$ are defined as the same vectors as $|\widetilde{\Psi}^{\mathsf{PerO}}_i(\vec{p};\vec{x})\rangle$ but generated by A interacting with a compressed oracle for the uniform distribution and projected to injective databases at the end.

Finally we can calculate the norm of the off-diagonal terms of the permutation state. By $\pm$ we denote adding and removing the state:

$$\begin{aligned}
&\left\| \sum_{d=1}^{2q-2} \sum_{t=1}^{t_{\max}(d)} \sum_{\vec{p}} C_{t,\vec{p}} \sum_{\vec{x}:|\vec{x}|=t} \left(\widetilde{\rho}^{d}_{\mathsf{PerO}}(\vec{x},\vec{x}) \pm |\widetilde{\Psi}^{\mathrm{Good}}(\vec{p};\vec{x})\rangle\langle\widetilde{\Psi}^{\mathrm{Good}}(\vec{p};\vec{x})| \right) \right\| \\
&= \Bigg\| \sum_{d=1}^{2q-2} \sum_{t=1}^{t_{\max}(d)} \sum_{\vec{p}} C_{t,\vec{p}} \sum_{\vec{x}:|\vec{x}|=t} \Big(\mathrm{Tr}_{FA\widetilde{W}} |\widetilde{\Psi}^{\mathsf{PerO}}_1(\vec{p};\vec{x})\rangle\langle\widetilde{\Psi}^{\mathsf{PerO}}_1(\vec{p};\vec{x})| \\
&+ \mathrm{Tr}_{FA\widetilde{W}} |\widetilde{\Psi}^{\mathsf{PerO}}_2(\vec{p};\vec{x})\rangle\langle\widetilde{\Psi}^{\mathsf{PerO}}_2(\vec{p};\vec{x})| + |\widetilde{\Psi}^{\mathrm{Good}}(\vec{p};\vec{x})\rangle\langle\widetilde{\Psi}^{\mathrm{Good}}(\vec{p};\vec{x})| \Big) \Bigg\| \\
&\leq \sum_{d=1}^{2q-2} \sum_{t=1}^{t_{\max}(d)} \sum_{\vec{p}} C_{t,\vec{p}} \Bigg\| \sum_{\vec{x}:|\vec{x}|=t} \Big(\mathrm{Tr}_{FA\widetilde{W}} |\widetilde{\Psi}^{\mathsf{PerO}}_1(\vec{p};\vec{x})\rangle\langle\widetilde{\Psi}^{\mathsf{PerO}}_1(\vec{p};\vec{x})|
\end{aligned} \tag{43}$$

$$+ \operatorname{Tr}_{FA^{\widetilde{W}}} |\widetilde{\Psi}_2^{\mathsf{PerO}}(\vec{p};\vec{x})\rangle\langle\widetilde{\Psi}_2^{\mathsf{PerO}}(\vec{p};\vec{x})| + |\widetilde{\Psi}^{\mathrm{Good}}(\vec{p};\vec{x})\rangle\langle\widetilde{\Psi}^{\mathrm{Good}}(\vec{p};\vec{x})|\Big)\Big\| \tag{44}$$

$$\leq 2 \sum_{d=1}^{2q-2} \sum_{t=1}^{t_{\max}(d)} \sum_{\vec{p}} C_{t,\vec{p}}, \tag{45}$$

where the first equality comes from the fact that the cross terms of $|\widetilde{\Psi}^{\mathrm{Good}}(\vec{x})\rangle\langle\widetilde{\Psi}^{\mathrm{Good}}(\vec{x})|$ equal $\widetilde{\rho}^s_{\mathsf{PerO}}(\vec{x},\vec{x})$, and the first inequality follows from the triangle inequality. The bound on the norm comes from the observation that $\operatorname{Tr}_{FA^{\widetilde{W}}} |\widetilde{\Psi}_1^{\mathsf{PerO}}(\vec{p};\vec{x})\rangle\langle\widetilde{\Psi}_1^{\mathsf{PerO}}(\vec{p};\vec{x})| + |\widetilde{\Psi}_2^{\mathsf{PerO}}(\vec{p};\vec{x})\rangle\langle\widetilde{\Psi}_2^{\mathsf{PerO}}(\vec{p};\vec{x})|$ can be interpreted as a partial trace over registers $FA^{\widetilde{W}}B$ of the density matrix of the state $|\widetilde{\Psi}_1^{\mathsf{PerO}}(\vec{p};\vec{x})\rangle|1\rangle_B + \operatorname{Tr}_{FA^{\widetilde{W}}} |\widetilde{\Psi}_2^{\mathsf{PerO}}(\vec{p};\vec{x})\rangle|2\rangle_B$. Both matrices are well-formed so their norm is bounded by 1. Considering the discussion following Eq. (40) we can bound the coefficient $C_{t,\vec{p}}$ by

$$\prod_{s=t}^{t+d-1} \frac{\vec{p}_{s+1-t}}{N-\vec{p}_{s+1-t}} \leq \prod_{s=t}^{t+d-1} \frac{s}{N-s}. \tag{46}$$

Moreover $\sum_{\vec{p}} \leq 2^d$.

To bound $\sum_{d=1}^{2q-2} \sum_{t=1}^{t_{\max}(d)} 2^d \left(\prod_{s=t}^{t+d-1} \frac{s}{N-s}\right)$ we first bound $\frac{s}{N-s} \leq \frac{t+d-1}{N-2q}$. Next we integrate the product by t:

$$\sum_{t=1}^{t_{\max}(d)} \left(\prod_{s=t}^{t+d-1} \frac{s}{N-s}\right) \leq \sum_{t=1}^{q} \left(\frac{t+d-1}{N-2q}\right)^d \leq \int_1^q \mathrm{d}t \left(\frac{t+d-1}{N-2q}\right)^d \tag{47}$$

$$= \frac{\left((d+q-1)^{d+1} - d^{d+1}\right)\left(\frac{1}{N-2q}\right)^d}{d+1} \leq \frac{3q}{2}\left(\frac{3q}{N-2q}\right)^d. \tag{48}$$

Next we sum over d to get the final bound:

$$\sum_{d=1}^{2q-2} \frac{3q}{2}\left(\frac{2\cdot 3q}{N-2q}\right)^d = \frac{36q^2 - 36^q q^{2q}\left(\frac{1}{N-2q}\right)^{2q-2}}{4(N-8q)} \leq \frac{9q^2}{N-8q}. \tag{49}$$

This concludes the proof. □

The second important lemma provides a bound on the distance of the good state and the punctured oracle state conditioned on ¬Find and $\mathbb{P}\left[\mathrm{Find}\right]$.

Lemma 13. *The good state and the compressed permutation states are close in trace distance:*

$$\frac{1}{2}\left\|\rho^{AD}_{\mathrm{Good}} - \Phi^{AD}_{\mathsf{CPerO}}\right\|_1 \leq \frac{1}{2}\cdot\frac{3q^2}{(N-q)}, \tag{50}$$

moreover the probability of Find *in this case is*

$$\mathbb{P}\left[\mathrm{Find}\right] \leq \frac{2q^2}{N} + \frac{4q^{7/2}}{N\sqrt{N-q}} + \frac{5q^5}{2N(N-q)}, \tag{51}$$

that for $q \in O(\sqrt[3]{N})$ is just $\frac{11q^2}{N}$.

Proof. We observe that the effective queries and the database register are essentially the same in ρ_{CPerO} and $\rho_{\mathsf{CPhO}\setminus R_{\mathrm{coll}}}$ and then use Claim 21 and Lemma 13 from [Cza+19]. The proof of Lemma 13 from [Cza+19] does not depend on the particular form of queries or the adversary, hence all results translate to our case. □

3.3 Punctured Compressed Permutation Oracle

In the proof of indifferentiability we puncture the compressed permutation oracle. So not only we puncture on regular collisions, but also on collisions in a part of the output (we call them *inner*-collisions). The inner-collisions relation is defined as:

$$R_{\text{inner}} := \Bigg\{ D \in \bigcup_{s=1}^{q} \left(\{0,1\}^{r+c}\right)^{2s} : \\ \exists i, j \neq i, y_i, y_j \in D^Y, y_i^{\text{inn}} = y_j^{\text{inn}} \vee y_i^{\text{inn}} = 0^c \Bigg\}, \tag{52}$$

where $D = ((x_1, y_1), \ldots, (x_s, y_s))$, and by y^{inn} we denote the last c bits of $y \in \{0,1\}^{r+c}$. To use the O2H lemma we just need to bound probability of Find for the combined relation (regular and inner collisions). In the proof of Lemma 13 in [Cza+19], the authors state the following bound on $\mathbb{P}\left[\text{Find}\right]$, that can be applied to the adversary interacting with $\mathsf{CPerO}_{\{0,1\}^{r+c}}$ punctured on R_{inner}:

$$\mathbb{P}\left[\text{Find} : \text{A}[\mathsf{CPerO}_{\{0,1\}^{r+c}} \setminus R_{\text{inner}}]\right] \\ \leq \sum_{i=1}^{q} \left(\sum_{j=1}^{i-1} 5\frac{b(j)}{\sqrt{N(N-b(q))}} + 2\sqrt{\frac{b(i)}{N}} + \frac{b(i)^{3/2}}{N\sqrt{N-b(q)}} \right)^2, \tag{53}$$

where $N = 2^{r+c}$.

In our case, of regular collisions combined with inner-collisions we have

$$b(s) = s - 1 + 2^r s. \tag{54}$$

So the final bound is

$$\mathbb{P}\left[\text{Find} : \text{A}[\mathsf{CPerO}_{\{0,1\}^{r+c}} \setminus R_{\text{inner}}]\right] \leq 12\frac{q^2}{2^c} + 66\frac{q^{7/2}}{2^c\sqrt{2^c - 2q}} + 332\frac{q^5}{2^c(2^c - 2q)}, \tag{55}$$

which for $q \in O(\sqrt[3]{2^c})$ simplifies to $410\frac{q^2}{2^c}$.

4 Indifferentiability of Sponges with Random Permutations

In this section we use the new technique to prove quantum indifferentiability of the sponge construction instantiated with a random permutation. We present proofs both of classical and quantum indifferentiability. The classical proof is located in Appendix B. Our proof of quantum security mimics the structure of the classical one. This approach is the same as in [Cza+19] and we hope will similarly lead to easier understanding of the second proof.

In our proofs we follow the (quantum) game-playing framework. Classical games make have use of lazy-sampled functions and "Bad" events. The key statement in classical game-playing proofs is the fundamental game-playing lemma (Lemma 3). It states that two games are identical until the bad event occurs. Moreover the fundamental lemma provides a bound depending on the probability of the bad event. The quantum game-playing framework gives us similar tools that are based on the compressed oracle technique and the O2H lemma.

In the quantum case, the role of bad events is played by punctured oracles, defined in Def. 2. Thanks to measurement that occur after every query we can condition on the content of quantum databases. That is a feature that allows to follow the classical proof so closely. The quantum version of the fundamental game-playing lemma is the O2H lemma. The second distinguishability bound that we use is shown in Lemma 6.

The general idea behind both indifferentiability proofs is that if there are no inner-collisions, then the outputs of the sponge construction are uniformly distributed. We condition on random permutations that do not have inner-collisions (in the q queries made by the adversary) using the (quantum) game-playing framework. All the details on sponge-specific notation used in this section are located in section 2.4.

To save space we present multiple algorithms in one. To do that we follow a convention where only the boxed algorithms perform the boxed operations. In our case there are actually more than two algorithms, so that the color of the box also matters.

In both proofs we denote the random oracle, defined in Eq. (5), by R. Moreover, we omit the second input and fix the output length to a single r-bit block.

4.1 Quantum Indifferentiability of Sponges with Random Permutations

In the proof of quantum security we will be using $\mathsf{CPerO}_{\{0,1\}^{r+c}}$ and puncture it on parts of the database, concretely on the inner parts of outputs. To maintain the structure of the proof similar to the classical case we still write the simulators to include separate procedures for sampling the inner and the outer part of the output. In fact though we sample both parts at once and puncture only on a part of the database. This change is only a semantic one, and the core of reasoning is the same. We use the fact that we can use the O2H lemma between two punctured oracles.

We denote the part of $\mathsf{CPerO}_{\{0,1\}^{r+c}}$ responsible for sampling the inner part by $\mathsf{CPerO}^{\mathsf{inn}}_{\{0,1\}^{r+c}}$ and the part of $\mathsf{CPerO}_{\{0,1\}^{r+c}}$ responsible for sampling the outer part by $\mathsf{CPerO}^{\mathsf{out}}_{\{0,1\}^{r+c}}$. The correct understanding of this notation is that in fact there is just the single $\mathsf{CPerO}_{\{0,1\}^{r+c}}$ applied and is punctured on all relations at once. In case of any statements dividing the calls to the inner and outer parts, we just move all the operations between the calls in the pseudocode to before of the first oracle.

Another important detail concerns the type of oracle we consider. For the sake of presentation we chose the basis used for the adversary's queries to be the standard basis. This is not how we defined CPerO, but the change does not influence the applicability of the compressed permutation oracle. We just need to keep in mind that the oracle used in this section is actually $\mathsf{HT}^{A^Y}_{r+c}\mathsf{CPerO}_{\{0,1\}^{r+c}}\mathsf{HT}^{A^Y}_{r+c}$, where $\mathsf{CPerO}_{\{0,1\}^{r+c}}$ is the oracle defined in Definition 8 and $\mathsf{HT}_n|y\rangle := \sum_{\eta\in\{0,1\}^n}(-1)^{y\cdot\eta}|\eta\rangle$ is the Hadamard transform.

In **Game 4** of the classical proof, we replace the sampled outer part with the output of the random oracle. Note that before the classical simulator terminates she saves the full sate in the sponge graph. There are two issues with doing the same in the quantum simulator. First of all, we cannot save information that we have not provided the adversary with. This would cause to increase the distinguishing advantage and not necessarily by a negligible amount. Second of all, we sample a random permutation, so without saving all the outputs we could cause collisions we could not track.

To solve both these problems we do not save the oracle outputs in D. However, at the beginning of each run of the simulator we reprepare the sponge graph and the missing values in D. To prepare the sponge graph we query R on all necessary inputs to f^{inn} (where f^{inn} is the function f with its output reduced to the last c bits), i.e. on the inputs that are consistent with a path from the root to a rooted node. This is done gradually by iterating over the length of the paths. We begin with the length-0 paths, i.e. with all inputs in the database D^{inn} (database D with the Y parts of all tuples reduced to the last c bits) where the inner part is the all zero string. If the outer part of such an input (which is not changed by the application of SpPath) is equal to a padding of an input, that input is queried to determine the outer part of the output of f, creating an edge in the sponge graph. We can continue with length-1 paths. For each entry of the database D^{inn}, check whether the input register is equal to a node in the current partial sponge graph. If so, the entry corresponds to a rooted node. Using the entry and the edge

connecting its input to the root, a possible padded input to Sponge is created using SpPath. If it is a valid padding, R is queried to determine the outer part of the output of f, etc.

We denote by U_G the unitary that acting on $|0\rangle$ constructs G including edges consistent with queries held by the quantum compressed database from register D. Similarly we define $\mathsf{U}_{\mathcal{R}\cup\mathcal{U}}$ to temporarily create a description of the set of supernodes that are rooted or have an outgoing edge.

It is important to note that the "IF" statements in the quantum simulator are in fact quantum controlled operations. To correctly perform the controlled operation we postpone the measurements occurring in punctured oracles to after uncomputing of G and $\mathcal{R}\cup\mathcal{U}$.

We define the function corresponding to the probability of Find, it follows from Eq. (55):

$$b^Q_{\text{coll}}(q) := 410\frac{q^2}{2^c}. \tag{56}$$

Below we state and prove the main result of this paper, quantum indifferentiability of sponges with random permutations.

The leading term in the bound below is $O\left(\sqrt{q^3/2^c}\right)$, which for $q \in O(\sqrt[3]{2^c})$ approaches a constant. Considering the complexity of the inner-collision-finding algorithm from [Cza+18], which is of the order $\Omega\left(2^{c/3}\right)$, we state that our bound is tight, at least in the presence of an external random oracle (which is the assumption made in [Cza+18]).

Theorem 14 (Sponge with permutations, quantum indifferentiability)**.** $\textsc{Sponge}_f[\textsc{pad}, r, c]$ *calling a random permutation f is (q,ε)-indifferentiable from a random oracle (defined in Eq. (5)) for* quantum *adversaries for any $q \in O(\sqrt[3]{2^c})$ (which is trivially $q < |2^c|$) and $\varepsilon = \frac{17q^2}{2^{r+c}-8q} + 3280\frac{q^2}{2^c} + \sqrt{410(q+1)\frac{q^2}{2^c}}$.*

Proof. In Algorithm 2 we describe the simulators we use in this proof.

Game 1 We start with the real world where the distinguisher A has quantum access to a random permutation $f : \{0,1\}^{r+c} \to \{0,1\}^{r+c}$ and the construction $\textsc{Sponge}_f$ using this function. The formal definition of the first game is

$$\textbf{Game 1} := \left(1 \leftarrow \mathsf{A}[\textsc{Sponge}_f, (f, f^{-1})]\right). \tag{57}$$

Game 2 In the second game we introduce the simulator S_2—defined in Algorithm 2—that lazy-samples the random permutation f. The definition of the second game is

$$\textbf{Game 2} := \left(1 \leftarrow \mathsf{A}[\textsc{Sponge}_{\mathsf{S}_2}, (\mathsf{S}_2, \mathsf{S}_2^{-1})]\right), \tag{58}$$

where by S^{-1} we denote the backward interface of S. The simulator S_2 models the quantum random permutation with errors given in Theorem 9:

$$|\mathbb{P}[\textbf{Game 2}] - \mathbb{P}[\textbf{Game 1}]| \leq \frac{17q^2}{2^{r+c}-8q}. \tag{59}$$

Game 3 In the next step we modify S_2 to S_3. The game is then

$$\textbf{Game 3} := \left(1 \leftarrow \mathsf{A}[\textsc{Sponge}_{\mathsf{S}_3}, (\mathsf{S}_3, \mathsf{S}_3^{-1})]\right). \tag{60}$$

We made a single change in S_3 compared to S_2, we introduced puncturing on the relation $\mathcal{R}\cup\mathcal{U}$ in the forward direction and $\mathcal{R}$ in the backward direction. With such a change of the simulators we can use Lemma 4 to bound the difference of probabilities:

$$|\mathbb{P}[\textbf{Game 3}] - \mathbb{P}[\textbf{Game 2}]| \leq \sqrt{(q+1)\mathbb{P}[\text{Find} : \mathsf{A}[\textsc{Sponge}_{\mathsf{S}_3}, \mathsf{S}_3]]}, \tag{61}$$

Algorithm 2: Quantum $\boxed{\mathsf{S}_2}$, $\boxed{\mathsf{S}_3}$, $\boxed{\mathsf{S}_4}$, permutations

State : Quantum compressed database register D

Interface: f, forward queries
Input : $|s, v\rangle_{XY} \in \mathcal{H}^{\otimes 2}_{\{0,1\}^{r+c}}$
Output : $|s, v \oplus f(s)\rangle_{XY}$

Apply $\mathsf{U}_{\mathcal{R}\cup\mathcal{U}}\mathsf{U}_G$ to register D and two fresh registers
if $s \in \mathcal{R} \wedge \mathcal{R} \cup \mathcal{U} \neq \{0,1\}^c$ **then** `// s-rooted, no saturation`
  Apply $\boxed{\mathsf{CPerO}^{\mathrm{inn}\,XYDG}_{\{0,1\}^{r+c}}}$, $\boxed{(\mathsf{CPerO}_{\{0,1\}^{r+c}} \setminus (\mathcal{R} \cup \mathcal{U}))^{\mathrm{inn}\,XYDG}}$, result: t^{inn}
  Construct a path to s: $p := \mathsf{SpPath}(s, G)$
  if $\exists x : p = \textsc{pad}(x)$ **then**
    $\boxed{\text{Apply } \mathsf{CPerO}^{\mathrm{out}\,XYDG}_{\{0,1\}^{r+c}}}$, result: t^{out}
    Write x in a fresh register X_R, $\boxed{\text{apply } \mathsf{R}^{XX_RYDG}}$, uncompute x from X_R, result: t^{out} `// Random oracle`
  else
    Apply $\mathsf{CPerO}^{\mathrm{out}\,XYDG}_{\{0,1\}^{r+c}}$, result: t^{out}
  $t := (t^{\mathrm{out}}, t^{\mathrm{inn}})$, the value of the output stored in D
else
  Apply $\mathsf{CPerO}^{XYDG}_{\{0,1\}^{r+c}}$, result t
Uncompute G and $\mathcal{R} \cup \mathcal{U}$
Output $|s, v + t\rangle_{XY}$

Interface: f^{-1}, backward queries
Input : $|s, v\rangle \in \mathcal{H}^{\otimes 2}_{\{0,1\}^{r+c}}$
Output : $|s, v + f^{-1}(s)\tau\rangle$

Apply $\mathsf{U}_{\mathcal{R}\cup\mathcal{U}} \circ \mathsf{U}_G$ to registers D and two fresh registers
Apply $\boxed{\mathsf{CPerO}^{\mathrm{inn}\,-1\,XYDG}_{\{0,1\}^{r+c}}}$, $\boxed{(\mathsf{CPerO}_{\{0,1\}^{r+c}} \setminus \mathcal{R})^{\mathrm{inn}\,-1\,XYDG}}$, result t^{inn}
Apply $\mathsf{CPerO}^{\mathrm{out}\,-1\,XYDG}_{\{0,1\}^{r+c}}$, result: t^{out}
$t := (t^{\mathrm{out}}, t^{\mathrm{inn}})$, the value of the output saved to D
Uncompute G and $\mathcal{R} \cup \mathcal{U}$
Output $|s, v + t\rangle_{XY}$

where Find denotes the success of the measurement in the punctured oracle. We can bound $\mathbb{P}[\text{Find} : \mathrm{A}[\textsc{Sponge}_{\mathsf{S}_3}, \mathsf{S}_3]]$ using Eq. (55):

$$\mathbb{P}[\text{Find} : \mathrm{A}[\textsc{Sponge}_{\mathsf{S}_3}, \mathsf{S}_3]] \leq b^Q_{\text{coll}}(q). \tag{62}$$

Game 4 In this step we introduce the random oracle R but only to generate the outer part of the output of f. The game is defined as

$$\textbf{Game 4} := \left(1 \leftarrow \mathrm{A}[\textsc{Sponge}_{\mathsf{S}_4}, (\mathsf{S}_4^{\mathsf{R}}, \mathsf{S}_4^{-1})]\right). \tag{63}$$

Thanks to the classical argument, Claim 17 we have that S_4 and S_3 are identical until bad, as in Definition 5. Then we can use Lemma 6 to bound the advantage of the adversary

$$|\mathbb{P}[\textbf{Game 4}] - \mathbb{P}[\textbf{Game 3}]| \leq 4\mathbb{P}[\text{Find} : \mathrm{A}[\textsc{Sponge}_{\mathsf{S}_3}, (\mathsf{S}_3, \mathsf{S}_3^{-1})]] \leq 4b^Q_{\text{coll}}(q) \tag{64}$$

Game 5 In this stage of the proof we change the private interface to contain the actual random oracle. The simulator is the same and the game is

$$\textbf{Game 5} := \left(1 \leftarrow \mathrm{A}[\mathsf{R}, (\mathsf{S}_4^{\mathsf{R}}, \mathsf{S}_4^{-1})]\right). \tag{65}$$

The advantage is

$$|\mathbb{P}[\textbf{Game 5}] - \mathbb{P}[\textbf{Game 4}]| \leq 4\mathbb{P}[\text{Find} : \mathrm{A}[\textsc{Sponge}_{\mathsf{S}_4}, (\mathsf{S}_4^{\mathsf{R}}, \mathsf{S}_4^{-1})]] \leq 4b^Q_{\text{coll}}(q), \tag{66}$$

conditioned on $\neg$Find, outputs of the private interface are the same, then the games are identical until bad and we can use Lemma 6 to bound the advantage of the adversary.

Saturation certainly does not occur for $q < 2^c$ as the database in every branch of the superposition increases by at most one in every query. Collecting the differences between games yields the claimed $\varepsilon = \frac{17q^2}{2^{r+c}-8q} + 8b^Q_{\text{coll}}(q) + \sqrt{(q+1)b^Q_{\text{coll}}(q)}$. □

Acknowledgments

The author would like to thank Christian Majenz for enlightening discussions and Christian Schaffner for his comments that improved parts of this manuscript. He also thanks Minki Hhan for a number of valuable comments, in particular for noticing a missing point (now added) in the proof of the main proof of the paper. He would also like to acknowledge the support of a NWO VIDI grant (Project No. 639.022.519).

References

[AHU19] Andris Ambainis, Mike Hamburg, and Dominique Unruh. "Quantum Security Proofs Using Semi-classical Oracles". In: *Advances in Cryptology - CRYPTO 2019*. 2019, pp. 269–295. DOI: 10.1007/978-3-030-26951-7_10.

[BR93] Mihir Bellare and Phillip Rogaway. "Random oracles are practical: A paradigm for designing efficient protocols". In: *Proceedings of the 1st ACM conference on Computer and communications security*. ACM. 1993, pp. 62–73. DOI: 10.1145/168588.168596.

[BR06] Mihir Bellare and Phillip Rogaway. "The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs". In: *Advances in Cryptology - EUROCRYPT 2006*. Springer Berlin Heidelberg, 2006, pp. 409–426. DOI: 10.1007/11761679_25.

[BBD09] D.J. Bernstein, J. Buchmann, and E. Dahmen. *Post-Quantum Cryptography*. Springer Berlin Heidelberg, 2009.

[Ber+07] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "Sponge functions". In: *ECRYPT hash workshop*. Vol. 2007. 9. https://keccak.team/files/SpongeFunctions.pdf. Citeseer. 2007.

[Ber+08] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "On the Indifferentiability of the Sponge Construction". In: *Advances in Cryptology – EUROCRYPT 2008*. Springer Berlin Heidelberg, 2008, pp. 181–197. DOI: 10.1007/978-3-540-78967-3_11.

[Chu+20] Kai-Min Chung, Serge Fehr, Yu-Hsuan Huang, and Tai-Ning Liao. "On the Compressed-Oracle Technique, and Post-Quantum Security of Proofs of Sequential Work". In: (2020). arXiv: 2010.11658.

[Cor+05] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. "Merkle-Damgård Revisited: How to Construct a Hash Function". In: *Advances in Cryptology – CRYPTO 2005*. Springer Berlin Heidelberg, 2005, pp. 430–448. DOI: 10.1007/11535218_26.

[Cza+18] Jan Czajkowski, Leon Groot Bruinderink, Andreas Hülsing, Christian Schaffner, and Dominique Unruh. "Post-quantum Security of the Sponge Construction". In: *Post-Quantum Cryptography*. Springer International Publishing, 2018, pp. 185–204. DOI: 10.1007/978-3-319-79063-3_9.

[CHS19] Jan Czajkowski, Andreas Hülsing, and Christian Schaffner. "Quantum Indistinguishability of Random Sponges". In: *Advances in Cryptology - CRYPTO 2019*. 2019, pp. 296–325. DOI: 10.1007/978-3-030-26951-7_11.

[Cza+19] Jan Czajkowski, Christian Majenz, Christian Schaffner, and Sebastian Zur. "Quantum Lazy Sampling and Game-Playing Proofs for Quantum Indifferentiability". In: (2019). arXiv: 1904.11477.

[HI19] Akinori Hosoyamada and Tetsu Iwata. "4-Round Luby-Rackoff Construction is a qPRP". In: *Advances in Cryptology - ASIACRYPT 2019*. 2019, pp. 145–174. DOI: 10.1007/978-3-030-34578-5_6.

[LZ19] Qipeng Liu and Mark Zhandry. "On Finding Quantum Multi-collisions". In: *Advances in Cryptology - EUROCRYPT 2019*. 2019, pp. 189–218. DOI: 10.1007/978-3-030-17659-4_7. URL: https://doi.org/10.1007/978-3-030-17659-4_7.

[MRH04] Ueli Maurer, Renato Renner, and Clemens Holenstein. "Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology". In: *Theory of Cryptography*. Springer Berlin Heidelberg, 2004, pp. 21–39. DOI: 10.1007/978-3-540-24638-1_2.

[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. 2014. URL: http://csrc.nist.gov/publications/drafts/fips-202/fips_202_draft.pdf.

[NIS15] NIST. *Secure Hash Standard (SHS)*. Draft FIPS 180-4. 2015. DOI: 10.6028/NIST.FIPS.180-4.

[NIS17] NIST. *Post-Quantum Cryptography Standardization process*. 2017. URL: https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization.

[Ros21] Ansis Rosmanis. "Tight Bounds for Inverting Permutations via Compressed Oracle Arguments". In: *arXiv preprint arXiv:2103.08975* (2021). URL: https://arxiv.org/abs/2103.08975.

[Unr14] Dominique Unruh. "Revocable Quantum Timed-Release Encryption". In: *Advances in Cryptology – EUROCRYPT 2014*. Springer Berlin Heidelberg, 2014, pp. 129–146. DOI: 10.1007/978-3-642-55220-5_8.

[Unr21] Dominique Unruh. "Compressed Permutation Oracles (And the Collision-Resistance of Sponge/SHA3)". Cryptology ePrint Archive, Report 2021/062. 2021. URL: https://eprint.iacr.org/2021/062.

[Wol11] Ronald de Wolf. "Quantum computing: Lecture notes". In: *University of Amsterdam* (2011). URL: https://arxiv.org/abs/1907.09415.

[Zha19] Mark Zhandry. "How to Record Quantum Queries, and Applications to Quantum Indifferentiability". In: *Advances in Cryptology – CRYPTO 2019*. Springer International Publishing, 2019, pp. 239–268. DOI: 10.1007/978-3-030-26951-7_9.

A Single-query Distinguisher

One interesting, in our opinion, observation concerning the permutation oracle is that a single query conveys a different amount of information depending on the number of previous queries. Note that the first query not only informs us that $f(x) = y$ but also that the output on all other $x' \neq x$ in not y. The more queries we make, the more "negative" information we have. The importance of the above observation is confirmed with the indistinguishability bound for a single-query distinguisher.

Theorem 15. *Given quantum access to a function sampled either according to the uniform distribution $\mathfrak{U}$ over functions from $\{f : [N] \to [N]\}$ or the uniform distribution over permutations $\mathfrak{P}$, we have that for all quantum distinguishers making a single quantum query to the oracle, the distinguishability advantage is at most $\frac{1}{N-1}$:*

$$\left| \mathop{\mathbb{P}}_{f \leftarrow \mathfrak{U}} \left[1 \leftarrow \mathrm{A}^f \right] - \mathop{\mathbb{P}}_{f \leftarrow \mathfrak{P}} \left[1 \leftarrow \mathrm{A}^f \right] \right| \leq \frac{1}{N-1}. \tag{67}$$

Moreover there is a distinguisher B *that has advantage $\frac{1}{2(N-1)}$.*

Proof. Lets us inspect the indistinguishability bound for the uniform distribution over functions $\mathfrak{U}$ and the uniform distribution over permutations $\mathfrak{P}$:

$$\left| \mathop{\mathbb{P}}_{f \leftarrow \mathfrak{U}} \left[1 \leftarrow \mathrm{A}^f \right] - \mathop{\mathbb{P}}_{f \leftarrow \mathfrak{P}} \left[1 \leftarrow \mathrm{A}^f \right] \right| = \left| \mathrm{Tr}\left(\mathsf{Q}_1 \rho^A_{\mathfrak{U}} \right) - \mathrm{Tr}\left(\mathsf{Q}_1 \rho^A_{\mathfrak{P}} \right) \right| \tag{68}$$

$$\leq \frac{1}{2} \left\| \rho^A_{\mathfrak{U}} - \rho^A_{\mathfrak{P}} \right\|_1 \tag{69}$$

where Q_1 is the measurement corresponding to A outputting 1. The inequality above comes from the fact that trace distance gives the optimal measurement for distinguishing two states [NC10]. Register A holds all of the adversary's state, this includes her work register, query register, and any other auxiliary registers she decides to use.

We model the oracle access with the full phase oracle. The adversary's initial state is:

$$|\psi_0\rangle_A := \sum_x \sum_{\eta_x, w} \alpha_{x,\eta_x,w} |x, \eta_x, w\rangle_{A^{XYW}}, \tag{70}$$

the states $\rho_{\mathfrak{U}}$ and $\rho_{\mathfrak{U}}$ are defined as the partial trace of the joint adversary-oracle state after a single query to $\mathsf{PhO}_{\mathfrak{U}}$ or $\mathsf{PhO}_{\mathfrak{P}}$, respectively. The full definitions of the reduced states are:

$$\rho_{\mathfrak{U}} = \sum_{x,x'} \sum_{\eta_x, \eta'_{x'}, w, w'} \alpha \bar{\alpha}' \sum_{y, y_{x'}} \frac{1}{N^2} \omega_N^{\eta_x y_x} \bar{\omega}_N^{\eta'_{x'} y_{x'}} |x, \eta_x, w\rangle\langle x', \eta'_{x'}, w'| \tag{71}$$

$$\rho_{\mathfrak{P}} = \sum_{x,x'} \sum_{\eta_x, \eta'_{x'}, w, w'} \alpha \bar{\alpha}' \sum_{y, y_{x'} \neq y_x} \frac{1}{N(N-1)} \omega_N^{\eta_x y_x} \bar{\omega}_N^{\eta'_{x'} y_{x'}} |x, \eta_x, w\rangle\langle x', \eta'_{x'}, w'|, \tag{72}$$

where we write α and α' for $\alpha_{x,\eta_x,w}$ and $\alpha_{x',\eta'_{x'},w'}$ respectively.

First of all we see that terms with $x' = x$ are equal, in this case in both states the sum over y_x and $y_{x'}$ simplify to just a single sum over $y_x \in [N]$. The off-diagonal (so for $x' \neq x$) terms are a bot more complicated to calculate. Let us notice that $\sum_{y, y_{x'} \neq y_x} = \sum_{y, y_{x'}} - \sum_{y, y_{x'} = y_x}$, this

splits the difference $\rho_{\mathfrak{U}} - \rho_{\mathfrak{P}}$ into two parts:

$$\rho_{\mathfrak{U}} - \rho_{\mathfrak{P}} = \left(\frac{1}{N^2} - \frac{1}{N(N-1)}\right)$$
$$\sum_{x,x' \neq x} \sum_{\eta_x, \eta'_{x'}, w, w'} \alpha\bar{\alpha}' \sum_{y, y_{x'}} \omega_N^{\eta_x y_x} \bar{\omega}_N^{\eta'_{x'} y_{x'}} |x, \eta_x, w\rangle\langle x', \eta'_{x'}, w'|$$
$$+ \sum_{x,x' \neq x} \sum_{\eta_x, \eta'_{x'}, w, w'} \alpha\bar{\alpha}' \underbrace{\sum_{y, y_{x'} = y_x} \frac{1}{N(N-1)} \omega_N^{\eta_x y_x} \bar{\omega}_N^{\eta'_{x'} y_{x'}}}_{= \frac{1}{N-1}\delta_{\eta_x, \eta'_{x'}}} |x, \eta_x, w\rangle\langle x', \eta'_{x'}, w'| \tag{73}$$
$$= -\frac{1}{N-1}\rho_{\mathfrak{U}} + \frac{1}{N-1}\mathsf{D}_{A^Y}(|\psi_0\rangle\langle\psi_0|), \tag{74}$$

where $\mathsf{D}_A(\rho) := \sum_a |a\rangle_A\langle a|\rho|a\rangle_A\langle a|$ denotes the map that outputs the diagonal entries in register A.

Finally we have

$$\frac{1}{2}\left\|\rho_{\mathfrak{U}}^A - \rho_{\mathfrak{P}}^A\right\|_1 = \frac{1}{2}\left\|-\frac{1}{N-1}\rho_{\mathfrak{U}} + \frac{1}{N-1}\mathsf{D}_{A^Y}(|\psi_0\rangle\langle\psi_0|)\right\|_1 \tag{75}$$
$$\leq \frac{1}{2}\frac{1}{N-1}\left\|\rho_{\mathfrak{U}}\right\|_1 + \frac{1}{2}\frac{1}{N-1}\left\||\psi_0\rangle\langle\psi_0|\right\|_1 = \frac{1}{N-1}, \tag{76}$$

where we use the triangle inequality and the fact that any CPTP (Completely Positive Trace-Preserving) map, like D, can only decrease the norm of a state.

Now we present the distinguisher B. We define B such that she performs the optimal measurement, and hence has advantage $\|\rho_{\mathfrak{U}} - \rho_{\mathfrak{P}}\|_1$, as stated in Eq. (69). Now let us consider the initial state of B, it does not have a work register and is the following state:

$$|\psi_0^{\mathsf{B}}\rangle_A := \sum_{x \in \{x_1, x_2\}} \frac{1}{\sqrt{2}}|x, \eta\rangle_{A^{XY}}, \tag{77}$$

where $\eta \neq 0$ and x_1 and x_2 are any distinct inputs. We know that the diagonal (in x) terms are equal, no matter if B interacts with $\mathfrak{U}$ or $\mathfrak{P}$, and the only part of the state that influences the trace distance is the off-diagonal. We know that in the state after interaction with a random function, Eq. (71), in the branch of superposition where $x \neq x'$ the sum $\sum_{y, y_{x'}} \frac{1}{N^2}\omega_N^{\eta_x y_x}\bar{\omega}_N^{\eta'_{x'} y_{x'}} = \delta_{\eta_x, 0}\delta_{\eta'_{x'}, 0}$. We observe that, as we have set $\eta \neq 0$, the off-diagonal part of $\rho_{\mathfrak{U}}$ (the case with $x \neq x'$) is 0.

The above discussion leads to the following equality:

$$\left|\mathop{\mathbb{P}}_{f \leftarrow \mathfrak{U}}\left[1 \leftarrow \mathsf{B}^f\right] - \mathop{\mathbb{P}}_{f \leftarrow \mathfrak{P}}\left[1 \leftarrow \mathsf{B}^f\right]\right| = \frac{1}{2}\left\|\rho_{\mathfrak{U}}^A - \rho_{\mathfrak{P}}^A\right\|_1 \tag{78}$$
$$= \frac{1}{2}\frac{1}{N-1}\left\|\sum_{x \in \{x_1, x_2\}} \sum_{x' \in \{x_1, x_2\}, x' \neq x} \frac{1}{2}|x, \eta\rangle\langle x', \eta|\right\|_1 = \frac{1}{2(N-1)}, \tag{79}$$

where the last equality comes from the definition of the trace norm: $\|\mathsf{A}\|_1 = \sum_\lambda |\lambda|$, where λ are the eigenvalues of the matrix A. □

B Classical Indifferentiability of Sponges with Random Permutations

Here we present a slightly modified proof of indifferentiability from [Ber+08]. We modify the proof, so that our classical and quantum proofs are more similar. We also introduce more steps

in the proof, as we want to keep a slow paste of statements so that each transition between games is clear, in that we follow the style of proof of [Cor+05]. Before we state the theorem, let us briefly go over lazy sampling of a random permutation. Specifically for permutations $f : \{0,1\}^{r+c} \to \{0,1\}^{r+c}$, sampling done in two stages, first sampling the inner part of the output and then the outer part.

By $\mathcal{O}$ we denote the set of outputs of previous queries. In the language of the sponge graph $\mathcal{O}^{-1}$ is the set of nodes with outgoing edges and $\mathcal{O}$ is the set of nodes with incoming edges. By $\mathcal{O}^{\text{inn}}$ we denote the set of supernodes with all nodes having an incoming edge. By $\mathcal{O}^{\text{out}}(t^{\text{inn}})$ we denote the set of nodes in the supernode t^{inn} with an incoming edge.

We need to define a procedure to lazy-sample outputs of a random permutation. The obvious solution of sampling uniformly from $\{0,1\}^{r+c} \setminus \mathcal{O}$ is not good enough as we want to sample the inner part before the outer part and retain the step-by-step structure of our proof. Instead we are going to first sample uniformly from $\{0,1\}^{r+c} \setminus \mathcal{O}$ but then discard the outer state. The value of the inner state t^{inn} is then effectively sampled from $\{0,1\}^c$ with weights $\frac{\left|\{0,1\}^r \setminus \mathcal{O}^{\text{out}}(t^{\text{inn}})\right|}{|\{0,1\}^{r+c} \setminus \mathcal{O}|}$. We call this distribution $\mathfrak{C}$. At this point we will be introducing bad events concerning the inner part of the sampled state. To sample the outer state we just sample uniformly from $\{0,1\}^r \setminus \mathcal{O}^{\text{out}}(t^{\text{inn}})$. We denote this distribution by $\mathfrak{A}(t^{\text{inn}})$.

In the case of the inverse of the random permutation we use a similar distribution but in the above definitions we take $\mathcal{O}^{-1}$—i.e. the set of nodes with outgoing edges—in both $\mathfrak{C}$ and $\mathfrak{A}$. We denote those distributions by $\mathfrak{C}^{-1}$ and $\mathfrak{A}^{-1}(t^{\text{inn}})$ respectively.

For the proof of indifferentiability we also need an upper bound on the probability of finding a collision in the inner part of outputs of a uniformly random function $f : \{0,1\}^{r+c} \to \{0,1\}^{r+c}$. Considering how Sponge is defined we want a bound on finding collisions and zero-preimages. We define the bound as a function of the number of queries q to f:

$$b_{\text{coll}}(q) := \frac{q(q+1)}{2^{c+1}}. \tag{80}$$

The bound can be found with standard techniques, a detailed derivation is presented in section 5.1 of [Cza+19].

The bound on $\mathbb{P}[\text{Bad}]$ in our proof for random permutations is the same as in the case of random functions [Cza+19]. We achieve a slightly weaker bound than in [Ber+08] because in our gradual argument we first sample from the set of inner states excluding only "full" supernodes $\mathcal{O}^{\text{inn}}$.

With these additional comments in mind we can proceed with the classical indifferentiability result.

Theorem 16 (Sponge with permutations, classical indifferentiability)**.** $\text{Sponge}_f[\text{pad}, r, c]$ *calling a random permutation f is (q, ε)-indifferentiable from a random oracle (defined in Eq. (5)) for* classical *adversaries for any $q < |2^c|$ and $\varepsilon = 7\frac{q(q+1)}{2^{c+1}}$.*

Proof. In Algorithm 3 we present the indifferentiability simulators.

Game 1 We start with the real world where the distinguisher A has access to a random permutation $f : \{0,1\}^{r+c} \to \{0,1\}^{r+c}$ and the construction Sponge_f using this function. The formal definition of the first game is

$$\textbf{Game 1} := \left(1 \leftarrow \text{A}[\text{Sponge}_f, (f, f^{-1})]\right). \tag{81}$$

Game 2 In the second game we introduce the simulator S_2—defined in Algorithm 3—that lazy-samples the random permutation f. The definition of the second game is

$$\textbf{Game 2} := \left(1 \leftarrow \text{A}[\text{Sponge}_{\text{S}_2}, (\text{S}_2, \text{S}_2^{-1})]\right), \tag{82}$$

Algorithm 3: Classical S_2, $\boxed{\mathsf{S}_3}$, $\boxed{\mathsf{S}_4}$, permutations

State : Current sponge graph G

Interface: f, forward queries
Input : $s \in \{0,1\}^{r+c}$
Output : $f(s)$

if s has no outgoing edge **then** `// Fresh query`
  if $s^{\text{inn}} \in \mathcal{R} \wedge \mathcal{R} \cup \mathcal{U} \neq \{0,1\}^c$ **then** `// $s^{\text{inn}}$-rooted, no saturation`
    $t^{\text{inn}} \leftarrow \mathfrak{C}$, $\boxed{\textbf{if } t^{\text{inn}} \in \mathcal{R} \cup \mathcal{U}, \text{ set } \mathsf{Bad}_1 = 1}$
    Construct a path to s: $p := \mathsf{SpPath}(s, G)$
    if $\exists x : p = \textsc{pad}(x)$ **then**
      $t^{\text{out}} \leftarrow \mathfrak{A}(t^{\text{inn}})$ `// Resampled in` S_4
      $\boxed{t^{\text{out}} := \mathsf{R}(x)}$
    else
      $t^{\text{out}} \leftarrow \mathfrak{A}(t^{\text{inn}})$
    $t := (t^{\text{out}}, t^{\text{inn}})$
  else
    $t \xleftarrow{\$} (\{0,1\}^{r+c}) \setminus \mathcal{O}$
  Add an edge (s, t) to $\mathcal{E}$.
Set t to the vertex at the end of the edge starting at s
Output t

Interface: f^{-1}, backward queries
Input : $s \in \{0,1\}^{r+c}$
Output : $f^{-1}(s)$

Construct the sponge graph G
if s has no incoming edge **then** `// Fresh query`
  $t^{\text{inn}} \leftarrow \mathfrak{C}^{-1}$, $\boxed{\textbf{if } t^{\text{inn}} \in \mathcal{R}, \text{ set } \mathsf{Bad}_2 = 1}$
  $t^{\text{out}} \leftarrow \mathfrak{A}^{-1}(t^{\text{inn}})$
  $t := (t^{\text{out}}, t^{\text{inn}})$
  Add an edge (t, s) to $\mathcal{E}$.
Set t to the vertex at the beginning of the edge ending at s
Output t

where by S^{-1} we denote the backward interface of S. Because the simulator S_2 perfectly models a random permutation and we use the same function for the private interface we have

$$|\mathbb{P}[\textbf{Game 2}] - \mathbb{P}[\textbf{Game 1}]| = 0. \tag{83}$$

Game 3 In the next step we modify S_2 to S_3. The game is then

$$\textbf{Game 3} := \left(1 \leftarrow \mathrm{A}[\textsc{Sponge}_{\mathsf{S}_3}, (\mathsf{S}_3, \mathsf{S}_3^{-1})]\right). \tag{84}$$

We made a single change in S_3 compared to S_2, we introduce the "bad" event $\mathrm{Bad} := \mathrm{Bad}_1 \vee \mathrm{Bad}_2$ that marks the difference between algorithms. We use this event as the bad event in Lemma 3. With such a change of the simulators we can use Lemma 3 to bound the difference of probabilities:

$$|\mathbb{P}[\textbf{Game 3}] - \mathbb{P}[\textbf{Game 2}]| \leq \mathbb{P}[\mathrm{Bad} = 1]. \tag{85}$$

It is now quite easy to bound $\mathbb{P}[\mathrm{Bad} = 1]$ as this is the probability of finding a collision or a preimage of the root in the set $\{0,1\}^c$ having made q random samples. Then we have that

$$\mathbb{P}[\mathrm{Bad} = 1] \leq b_{\mathrm{coll}}(q), \tag{86}$$

where the inequality comes from the fact that not all queries are made to rooted nodes.

Game 4 In this step we introduce the random oracle R but only to generate the outer part of the output of f. The game is defined as

$$\textbf{Game 4} := \left(1 \leftarrow \mathrm{A}[\textsc{Sponge}_{\mathsf{S}_4}, (\mathsf{S}_4^{\mathsf{R}}, \mathsf{S}_4^{-1})]\right). \tag{87}$$

Now we need to observe that if $\mathrm{Bad} = 0$ the outputs are identically distributed.

Claim 17. *Given that* $\mathrm{Bad} = 0$, **Game 4** *and* **Game 3** *are identical:*

$$|\mathbb{P}[\textbf{Game 4} \mid \mathrm{Bad} = 0] - \mathbb{P}[\textbf{Game 3} \mid \mathrm{Bad} = 0]| = 0. \tag{88}$$

Proof. If $\mathrm{Bad} = 0$, the inner-part is distributed in the same way in both games, so the only difference in distributions can come from the outer part. Given our discussion in section 2.4, though, if there are no inner-collisions, then the outer part is distributed uniformly at random. This reasoning is also presented in Lemma 1 and Lemma 2 of [Ber+07]. □

The two games are identical-until-bad, this implies that the probability of setting Bad to one in both games is the same $\mathbb{P}[\mathrm{Bad} = 1 : \textbf{Game 3}] = \mathbb{P}[\mathrm{Bad} = 1 : \textbf{Game 4}]$. Together with the above claim we can derive the advantage:

$$\begin{aligned}
&|\mathbb{P}[\textbf{Game 4}] - \mathbb{P}[\textbf{Game 3}]| \overset{\text{Claim 17}}{=} \Big| \mathbb{P}[\textbf{Game 4} \mid \mathrm{Bad} = 0] \\
&\cdot \underbrace{(\mathbb{P}[\mathrm{Bad} = 1 : \textbf{Game 3}] - \mathbb{P}[\mathrm{Bad} = 1 : \textbf{Game 4}])}_{=0} \\
&+ \underbrace{\mathbb{P}[\textbf{Game 3} \mid \mathrm{Bad} = 1]}_{\leq 1}\mathbb{P}[\mathrm{Bad} = 1] + \underbrace{\mathbb{P}[\textbf{Game 4} \mid \mathrm{Bad} = 1]}_{\leq 1}\mathbb{P}[\mathrm{Bad} = 1] \Big| \qquad (89) \\
&\leq 2\mathbb{P}[\mathrm{Bad} = 1] \leq 2b_{\mathrm{coll}}(q). \qquad (90)
\end{aligned}$$

Game 5 In this stage of the proof we change the private interface to contain the actual random oracle. The simulator is the same and the game is

$$\textbf{Game 5} := \left(1 \leftarrow \mathrm{A}[\mathsf{R}, (\mathsf{S}_4^{\mathsf{R}}, \mathsf{S}_4^{-1})]\right). \tag{91}$$

Conditioned on Bad $= 0$, the outputs of the simulator in Games 4 and 5 are the same and consistent with R. To calculate the adversary's advantage in distinguishing between the two games we can follow the proof of Lemma 6. We change $\mathsf{R} \setminus R_1$ to **Game 5**, $\mathsf{G} \setminus R_2$ to **Game 4**, and event Find to Bad $= 1$. As the derivation of Lemma 6 uses no quantum mechanical arguments and the assumption holds—the games are identical conditioned on Bad $= 0$—the bound holds:

$$|\mathbb{P}[\textbf{Game 5}] - \mathbb{P}[\textbf{Game 4}]| \leq 4\mathbb{P}[\mathrm{Bad}_1 = 1] \leq 4b_{\mathrm{coll}}(q). \tag{92}$$

Collecting and adding all the differences yields the claimed $\varepsilon = 7b_{\mathrm{coll}}(q)$. □

Symbol Index

$\lvert x\rvert$	Cardinality of a set x / length of a string x/ absolute value	
$\mathfrak{A}$	Distribution of outer part of outputs of a random permutation	26
Bad	A "bad" event in a game.	6
$\mathfrak{C}$	Distribution of inner part of outputs of a random permutation	26
R_{coll}	The collision relation.	5
$\mathsf{CPerO}_{[N]}$	Compressed Permutation Oracle	9
$\mathsf{CPhO}_{[N]}$	Compressed Phase Oracle	5
$\mathsf{CStO}_{[N]}$	Compressed Standard Oracle, for the unifrom distribution over functions from $\{f : [N] \to [N]\}$.	4
$\mathcal{D}_s$	The set of databases: $\mathcal{D}_s := \{((x_1, y_1), \ldots, (x_s, y_s)) \in [N]^{2s} : \forall i, j \neq i, x_i \neq x_j\}$.	3
$\mathcal{E}$	The set of edges of a sponge graph	7
Find	Event of measurement of the relation R returning 1	5
Flip	Algorithm flipping the database in the standard basis, inputs are outputs and vice versa.	9
$\mathcal{F}$	The set of functions $[N] \to [N]$.	4
$\lvert\Psi^{\text{Good}}\rangle$	The of A interacting with $\mathsf{CPhO}_{[N]}$ projected to injective databases.	11
HT_n	The Hadamard transform	18
$\mathcal{I}(s)$	The set of s-element tuples with distinct entries.	3
s^{inn}	The inner part of a state, last c bits of $s \in \{0,1\}^{r+c}$	
R_{inner}	The inner-collision relation.	17
$\lvert\psi\rangle$	A quantum state, a normalized or subnormalized vector in a Hilbert space.	
$O(n)$	Complexity class "big O"	
$\mathcal{O}$	The set of outputs of queries	26
s^{out}	The outer part of a state, first r bits of $s \in \{0,1\}^{r+c}$	
PAD	Padding function	7
$\mathsf{SpPath}(s, G)$	Function constructing an input to SPONGE leading to a given node	8
$\mathcal{P}$	The set of permutations $[N] \to [N]$.	10
PerO	Full permutation oracle	10
PhO	Phase Oracle, $\mathsf{QFT}_N^Y \circ \mathsf{StO} \circ \mathsf{QFT}_N^{\dagger Y}$	5
QFT_N	The Quantum Fourier Transform	5
$\mathcal{Q}_s$	The set of queries: $\mathcal{Q}_s := (\{+,-\} \times [N])^s$.	3
R	A random oracle.	3
$\mathcal{R}$	The set of rooted supernodes	8
$\text{SPONGE}_f[\text{PAD}, r, c]$	Sponge construction with the internal function f, capacity c, and rate r	7
StO	Standard Oracle	4
$\mathcal{U}$	The set of supernodes with outgoing edges	8

$\mathcal{V}$	The set of vertices of a sponge graph	7
$\oplus$	Bitwise XOR	3