Quantum Indistinguishability of Random Sponges

Jan Czajkowski*[1], Andreas Hülsing†[2], and Christian Schaffner‡[1]

[1]QuSoft, University of Amsterdam
[2]Technische Universiteit Eindhoven

January 21, 2019

Abstract

In this work we show that the sponge construction can be used to construct quantum-secure pseudorandom functions. As our main result we prove that random sponges are quantum indistinguishable from random functions. In this setting the adversary is given superposition access to the input-output behavior of the construction but not to the internal function. Our proofs hold under the assumption that the internal function is a random function or permutation. We then use this result to obtain a quantum-security version of a result by Andreeva, Daemen, Mennink, and Van Assche (FSE'15) which shows that a sponge that uses a secure PRP or PRF as internal function is a secure PRF. This result also proves that the recent attacks against CBC-MAC in the quantum-access model by Kaplan, Leurent, Leverrier, and Naya-Plasencia (Crypto'16) and Santoli, and Schaffner (QIC'16) can be prevented by introducing a state with a non-trivial inner part.

The proof of our main result is derived by analyzing the joint distribution of any q input-output pairs. Our method analyzes the statistical behavior of the considered construction in great detail. The used techniques might prove useful in future analysis of different cryptographic primitives considering quantum adversaries. Using Zhandry's PRF/PRP switching lemma we then obtain that quantum indistinguishability also holds if the internal block function is a random permutation.

Contents

*j.czajkowski@uva.nl
†andreas@huelsing.net
‡c.schaffner@uva.nl

1 Introduction

Originally introduced in the context of cryptographic hash functions, the sponge construction [2] became one of the most widely used constructions in symmetric cryptography. Consequently, sponges get used in keyed constructions, including message authentication codes (MAC), stream ciphers, and authenticated encryption (AE), see e.g. [5, 4, 7, 15, 18, 1, 11]. For all these applications it is either necessary or at least sufficient for security if a secretly keyed sponge is indistinguishable from a random function. That this is indeed the case was already shown in the original security proof for the sponge construction [3] where cryptographic sponges were shown to be indifferentiable from random functions. This result is widely applicable and consequently was followed up with several improved bounds for specific applications. Recent works [15, 1, 11] improved the bound for the setting of indistinguishability of secretly keyed sponges.

While these results show the applicability of the sponge construction in today's computing environment, they leave open the question of its applicability in a future post-quantum setting where adversaries have access to quantum computers. Such an attacker can for example run Shor's algorithm [20] to break the security of constructions based on the RSA or discrete-logarithm problem. While such constructions are hardly ever considered for practical symmetric cryptography due to their slow operations, the impact of quantum adversaries goes beyond Shor's algorithm. Conventional security proofs, especially in idealized models, might break down in the light of quantum attackers who are allowed to ask queries in superposition [8]. Going even further, allowing adversaries superposition access to secretly keyed primitives, it was shown that several well known MACs and encryption schemes, including CBC-MAC and the Even-Mansour block cipher become insecure [14, 12, 19]. While these latter attacks are not applicable in the post-quantum setting, they are indications that secret-key cryptography does not trivially withstand quantum adversaries and that it is necessary to study the security of symmetric cryptography in the post-quantum setting.

In this work we do exactly this: We study the security of secretly keyed sponges against quantum adversaries.

Sponges. The sponge construction [2] is an eXtendable Output Function (XOF) that maps arbitrary-length inputs to outputs of a length specified by an additional input. The construction operates on an $(r+c)$-bit state. The parameter r is called the rate and the parameter c is called the capacity. The first r bits of the state are called the outer part or outer state, the remaining c bits are called the inner part or inner state. The sponge uses an internal function $\mathbf{f}$ mapping $(r+c)$-bit strings to $(r+c)$-bit strings. To process a message consisting of several r-bit blocks, the sponge alternates between mixing a new message block into the outer state and applying $\mathbf{f}$, as shown in Figure 1. When all message blocks are processed (i.e. absorbed into the internal state) the sponge can be squeezed to produce outputs by alternating between applying $\mathbf{f}$ and outputting the outer state. We write $\textsc{Sponge}_{\mathbf{f}}$ for the sponge using $\mathbf{f}$ as internal function.

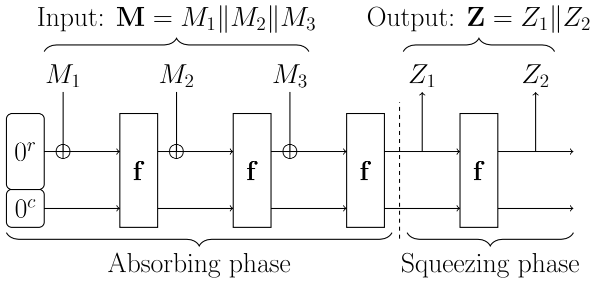


Figure 1: A scheme illustrating the sponge construction.

Sponges can be keyed in several ways. For example, the state can be initialized with the key, referred to as root-keyed sponge in [1]. Another option is to just apply the sponge on the concatenation of key and message. This was called the keyed sponge in [4] and the outer-keyed sponge in [1]. The last and for us most relevant concept is keying the sponge by replacing $\mathbf{f}$ with a keyed function $\mathbf{f}_K$. For the special case of $\mathbf{f}_K$ being a single-key Even-Mansour construction this was called E-M keyed sponge construction in [10] and later the inner-keyed sponge in [1]. We refer to the general case for any keyed function $\mathbf{f}_K$ as keyed-internal-function sponge.

Our results. As main result, we prove that the sponge construction using a random function or permutation is quantumly indistinguishable from a random function (see Theorems 8 and 16). This result can be used to obtain a quantum version of Theorem 1 from [1] (see Theorem 12) which states that the indistinguishability of keyed-internal-function sponges can be derived from the quantum-PRF-security (or quantum-PRP-security in case of a block-cipher) of the keyed internal function. Thereby we not only provide a proof for the security of keyed-internal-function sponges in the post-quantum setting, but even in the stronger quantum settings where the adversary gets full quantum-access to the keyed-internal-function sponge, i.e we prove that keyed-internal-function sponges are quantum PRFs.

Another implication of our result is that the quantum attacks against CBC-MAC mentioned above can be prevented using a state with a non-trivial inner part. The authors of the attack already noted[1] that their attack does not work in this case. More specifically, CBC-MAC can be viewed as full-width sponge (where the state has no inner part, i.e., the capacity is 0). On the other hand, a CBC-MAC where all message blocks are padded with 0^c and the output is truncated to the first r bits can be viewed as an keyed-internal-function sponge. Hence, our result applies and shows that the quantum attacks by Kaplan, Leurent, Leverrier, and Naya-Plasencia [12] and Santoli, and Schaffner [19] using Simon's algorithm are not applicable any longer. Even more, our result proves that this little tweak of CBC-MAC indeed results in a quantum secure MAC.

In Appendix A we show a direct proof of indistinguishability for $\mathbf{f}$ being a random permutation. In this proof we state and prove Lemma 19 that generalizes the average case polynomial

[1]See slide 16 (page 26) of their Crypto 2016 presentation available at `https://who.rocq.inria.fr/Gaetan.Leurent/files/Simon_CR16_slides.pdf`.

method to allow for functions that are not necessarily polynomials but are close to one; this result is not necessary to achieve the main goal of the paper but might be useful in other works using similar techniques.

A limitation. The authors of [1] use their Theorem 1 to show security of inner-keyed sponges using the PRP-security of single-key Even-Mansour. Their result does not carry over to the quantum setting as Even-Mansour is vulnerable in the quantum setting [14]. This does not lead an actual attack on inner-keyed sponges in the quantum setting. The attack needs access to the full input to the Even-Mansour cipher, which is never the case for inner-keyed sponges as long as a non-trivial inner state is used. However, the attack on Even-Mansour does render the modular proof strategy not applicable for inner-keyed sponges.

Our approach. The main technical contribution of our work is a proof that the probability for any given input-output behavior of $\textsc{Sponge}_{\mathbf{f}}$ is a polynomial in the capacity of the sponge. This observation allows us then to apply the average-case polynomial method of [22] (see Theorem 4 below).

In more detail, recall that the capacity of a $\textsc{Sponge}_{\mathbf{f}}$ is the size of the inner state (there are 2^c possible inner states for a sponge as in Figure 1). If the capacity of a sponge increases, it becomes less and less likely that there are collisions in the inner state. Hence for infinite capacity, the inner states are unique and so the internal functions are called on unique inputs and therefore, the sponge behaves like a random function. Our proof formalizes this intuition by carefully analyzing the probabilities for q given input-output values of the sponge in terms of the capacity. We show that these probabilities are in fact polynomials in the inverse of the capacity of degree at most q times the length of the input-output values. We refer to Lemma 9 for the formal statement.

By establishing the capacity as this crucial parameter, we fit directly into the proof technique from [22] that uses approximating polynomials of low degree to show closeness of distributions and in turn small quantum distinguishing advantage. By the PRF/PRP switching lemma from [23], quantum indistinguishability also holds for the case of $\mathbf{f}$ being a random permutation. In the appendix, we provide an alternative proof for this case by generalizing the proof technique of [22] to the case of permutations.

Organization. Section 2 introduces the definition of quantum indistinguishability and other notions used throughout this work. In Section 3 we extend the above informal discussion of the sponge construction with a more formal description. At the end of the section we show that $\textsc{Sponge}_{\mathbf{f}}$ is indistinguishable from a random oracle in the conventional-access setting (in contrast to the quantum-access model). In Section 4 we state the main result of our paper as well as several derived results. Section 5 contains an example proof valid for limited distinguishers but giving sufficient details to understand our approach and verify correctness without all the particulars of the full proof. Section 6 contains the proof of Lemma 9, the main technical result of this work. The case of random permutations is covered in Section 7. We conclude the paper with Section 8 discussing some open problems related to the problem we analyze and related work.

2 Preliminaries and Tools

In the Symbol Index 9 we list the most important notation used in our paper. We use $\textsc{small caps}$ for algorithms, CAPITAL letters for strings, **CAPITAL** and boldface for arrays of strings. *lower* case, italic letters denote parameters and counters. Functions are denoted by **lower** case boldface letters. Sets are denoted with $\mathcal{CAPITAL}$ calligraphic letters. Finally distributions are denoted with $\mathfrak{CAPITAL}$ letters using fraktur font. The general guideline for denoting elements of different sets is that we use a different letter together with indices. So if $\mathcal{A}$ is some set then A_i is the i-th element of that set. If we write $\mathcal{A}_i$ that means that the set $\mathcal{A}_i$ is a member of some family.

2.1 Quantum threat model

The quantum threat model we consider allows the adversary to query oracles in superposition. Oracles are modeled as unitary operators $\mathbf{U_h}$ acting on computational basis states as follows

$$\mathbf{U_h}|X, Y\rangle \mapsto |X, Y \oplus \mathbf{h}(X)\rangle. \tag{1}$$

The adversary is considered to have access to a fault-tolerant (perfect) quantum computer. We do not provide more details on quantum computing as we do not directly require it here, but we refer to [17] instead.

2.2 Distributions

A distribution $\mathfrak{D}$ on a set $\mathcal{X}$ is a function $\mathfrak{D} : \mathcal{X} \to [0,1]$ such that $\sum_{X\in\mathcal{X}} \mathfrak{D}(X) = 1$. We denote sampling X from $\mathcal{X}$ according to $\mathfrak{D}$ by $X \leftarrow \mathfrak{D}$. $\mathcal{Y}^{\mathcal{X}}$ denotes the set of functions $\{\mathbf{f} : \mathcal{X} \to \mathcal{Y}\}$. If $\mathfrak{D}$ is a distribution on $\mathcal{Y}$ then $\mathfrak{D}^{\mathcal{X}}$ denotes a distribution on $\mathcal{Y}^{\mathcal{X}}$ where the output for each input is chosen independently according to $\mathfrak{D}$. By $\overset{\$}{\leftarrow} \mathcal{X}$ we denote sampling uniformly at random from the set $\mathcal{X}$.

2.3 Classical and Quantum Indistinguishability

By classical indistinguishability we mean a feature of two distributions that are hard to distinguish if only polynomially many classical queries are allowed. The mentioned polynomial is evaluated on the security parameter. Note however that we have not yet specified it. For now though we leave it implicit, the security parameter will be specified for the particular construction we are going to analyze. In the following we are going to use functions $\mathbb{N} \to \mathbb{R}$ that for big enough argument are smaller than any inverse polynomial, they are called *negligible* functions.

Definition 1 (Classical Indistinguishability)**.** *Two distributions $\mathfrak{D}_1$ and $\mathfrak{D}_2$ over a set $\mathcal{Y}^{\mathcal{X}}$ are computationally classically indistinguishable if no quantum algorithm* A *can distinguish $\mathfrak{D}_1$ from $\mathfrak{D}_2$ using a polynomial number of classical queries. That is, for all* A, *there is a negligible function $\boldsymbol{\epsilon}$ such that*

$$\left| \underset{\mathbf{g}\leftarrow\mathfrak{D}_1}{\mathbb{P}} [\mathrm{A}^{\mathbf{g}}(.) = 1] - \underset{\mathbf{g}\leftarrow\mathfrak{D}_2}{\mathbb{P}} [\mathrm{A}^{\mathbf{g}}(.) = 1] \right| \leq \boldsymbol{\epsilon}. \tag{2}$$

We write $\mathrm{A}^{\mathbf{g}}$ to denote that adversary A has classical oracle access to $\mathbf{g}$. We will use the following generalization of the above definition to specify our goal.

Definition 2 (Quantum Indistinguishability [22])**.** *Two distributions $\mathfrak{D}_1$ and $\mathfrak{D}_2$ over a set $\mathcal{Y}^{\mathcal{X}}$ are computationally quantumly indistinguishable if no quantum algorithm* A *can distinguish $\mathfrak{D}_1$ from $\mathfrak{D}_2$ using a polynomial number of quantum queries. That is, for all* A, *there is a negligible function $\boldsymbol{\epsilon}$ such that*

$$\left| \underset{\mathbf{g}\leftarrow\mathfrak{D}_1}{\mathbb{P}} \left[\mathrm{A}^{|\mathbf{g}\rangle}(.) = 1\right] - \underset{\mathbf{g}\leftarrow\mathfrak{D}_2}{\mathbb{P}} \left[\mathrm{A}^{|\mathbf{g}\rangle}(.) = 1\right] \right| \leq \boldsymbol{\epsilon}. \tag{3}$$

We write $\mathrm{A}^{|\mathbf{g}\rangle}$ to denote that adversary A has quantum oracle access to $\mathbf{g}$, i.e. she can query $\mathbf{g}$ on a superposition of inputs.

In what follows the setting that we focus on is indistinguishability from a random oracle. The first distribution is the one analyzed and the other is the uniform distribution over the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$, i.e. $\mathcal{Y}^{\mathcal{X}}$. Sampling a uniformly random function is denoted by $\overset{\$}{\leftarrow} \mathcal{Y}^{\mathcal{X}}$.

2.4 Main tools

In this section we describe the proof technique—based on approximating polynomials—that proves useful when dealing with notions like quantum indistinguishability. In the following $[q] := \{1, 2, \ldots, q\}$.

Theorem 3 (Theorem 3.1 in [24])**.** *Let* A *be a quantum algorithm making q quantum queries to an oracle $\mathbf{h} : \mathcal{X} \to \mathcal{Y}$. If we draw $\mathbf{h}$ from some distribution $\mathfrak{D}$, then the quantity $\mathbb{P}_{\mathbf{h}\leftarrow\mathfrak{D}}[\mathrm{A}^{|\mathbf{h}\rangle}() = 1]$ is a linear combination of the quantities $\mathbb{P}_{\mathbf{h}\leftarrow\mathfrak{D}}[\forall i \in [2q] : \mathbf{h}(X^i) = Y^i]$, where $\forall i \in [2q] : (X^i, Y^i) \in \mathcal{X} \times \mathcal{Y}$.*

The intuition behind the above theorem is that with q queries the amplitudes of the quantum state of the algorithm depend on at most q input-output pairs. The probability of any outcome is a linear combination of squares of amplitudes, that is why we have $2q$ input-output pairs in the probability function. Finally as the probability of any measurement depends on just $2q$ input-output pairs the same holds for the algorithm's output probability. All the information about $\mathbf{h}$ comes from the queries A made.

We use the above theorem together with statements about approximating polynomials to connect the probability of some input-output behavior of a function from a given distribution with the probability of the adversary distinguishing two distributions.

Theorem 4 (Theorem 7.3 in [22])**.** *Fix q, and let $\mathfrak{F}_t$ be a family of distributions on $\mathcal{Y}^{\mathcal{X}}$ indexed by $t \in \mathbb{Z}^+ \cup \{\infty\}$. Suppose there is an integer d such that for every $2q$ pairs $\forall i \in [2q] : (X^i, Y^i) \in \mathcal{X} \times \mathcal{Y}$, the function $\mathbf{p}(1/t) = \mathbb{P}_{\mathbf{h}\leftarrow\mathfrak{F}_t}\left[\forall i \in [2q] : \mathbf{h}(X^i) = Y^i\right]$ is a polynomial of degree at most d in $1/t$. Then for any quantum algorithm* A *making at most q quantum queries, the output distribution under $\mathfrak{F}_t$ and $\mathfrak{F}_\infty$ are $\pi^2 d^3/3t$-close*

$$\left| \underset{\mathbf{h}\leftarrow\mathfrak{F}_t}{\mathbb{P}}\left[\mathrm{A}^{|\mathbf{h}\rangle}() = 1\right] - \underset{\mathbf{h}\leftarrow\mathfrak{F}_\infty}{\mathbb{P}}\left[\mathrm{A}^{|\mathbf{h}\rangle}() = 1\right] \right| < \frac{\pi^2 d^3}{6t}. \tag{4}$$

This theorem is an average case version of the polynomial method often used in complexity theory. If the polynomial approximating the ideal behavior of $\mathbf{h} \leftarrow \mathfrak{F}_\infty$ is of low degree the distance between polynomials must be small.

3 The Sponge Construction

3.1 Definition of Sponges

While an informal explanation of sponges was given in the introduction, we now give a more formal definition.

We define a *sponge-compliant* padding as:

Definition 5 (Definition 1 in [6])**.** *A padding rule is* sponge-compliant *if it never results in the empty string and if it satisfies the following criterion:*

$$\forall \nu \geq 0\ \forall \mathbf{M}, \mathbf{M}' \in \{0,1\}^* : \mathbf{M} \neq \mathbf{M}' \Rightarrow \mathbf{M} \| \textsc{pad}(|\mathbf{M}|) \neq \mathbf{M}' \| \textsc{pad}(|\mathbf{M}'|) \| 0^{\nu r}, \tag{5}$$

where $\|$ denotes concatenation of bit strings.

A formal definition of the construction is provided as Algorithm 1. Note that $\oplus$ denotes the bitwise XOR, $|\mathbf{P}|_r$ denotes the number of blocks of length r in $\mathbf{P}$, $\mathbf{P}_i$ is the i-th block of $\mathbf{P}$ and $\lfloor \mathbf{Z} \rfloor_\ell$ are the first ℓ bits of $\mathbf{Z}$.

Algorithm 1: $\textsc{Sponge}_{\mathbf{f}}[\textsc{pad}, r]$

```
Input  : $\mathbf{M} \in \{0,1\}^*$, $\ell \geq 0$.
Output: $\mathbf{Z} \in \{0,1\}^{\ell}$
1 $\mathbf{P} := \mathbf{M} \| \textsc{pad}[r](|\mathbf{M}|)$, and $S := 0^{r+c}$.
2 for $i = 0$ to $|\mathbf{P}|_r - 1$ do                      // Absorbing phase
3     $S = S \oplus (\mathbf{P}_i \| 0^c)$
4     $S = \mathbf{f}(S)$
5 $\mathbf{Z} := \lfloor S \rfloor_r$                          // Squeezing phase
6 while $|\mathbf{Z}| < \ell$ do
7     $S = \mathbf{f}(S)$
8     $\mathbf{Z} = \mathbf{Z} \| \lfloor S \rfloor_r$
9 Output $\lfloor \mathbf{Z} \rfloor_{\ell}$
```

3.2 Classical indistinguishability of random Sponges

In the following we state the indistinguishability result in the classical domain. We use the following notation for a set of arbitrary finite-length bit strings:

$$\{0,1\}^* := \bigcup_{l \geq 0} \{0,1\}^l, \tag{6}$$

we usually denote this set by $\mathcal{M}$. Before we proceed let us define what we mean by a random oracle.

Definition 6 (Random Oracle)**.** *A random oracle is sampled from a distribution $\mathfrak{R}$ on functions from $\mathcal{M} \times \mathbb{N}$ to $\mathcal{M}$, where $\mathcal{M} := \{0,1\}^*$. We define $\mathbf{h} \leftarrow \mathfrak{R}$ as follows:*

- *Choose $\mathbf{g}$ uniformly at random from $\{\mathbf{g} : \mathcal{M} \to \{0,1\}^{\infty}\}$, where by $\{0,1\}^{\infty}$ we denote the set of infinitely long bit-strings.*
- *For each $(X, \ell) \in \mathcal{M} \times \mathbb{N}$ set $\mathbf{h}(X, \ell) := \lfloor \mathbf{g}(X) \rfloor_{\ell}$, that is output the first ℓ bits of the output of $\mathbf{g}$.*

Theorem 7 (Classical indistinguishability of $\textsc{Sponge}$)**.** *If $\mathbf{f}$ is a random transformation or a random permutation then $\textsc{Sponge}_{\mathbf{f}}$ defined in Alg. 1 is classically indistinguishable from a random oracle. Namely for all quantum algorithms* A *making polynomially many classical queries there is a negligible function $\boldsymbol{\epsilon}$ such that*

$$\left| \underset{\mathbf{f} \xleftarrow{\$} \mathcal{S}^{\mathcal{S}}}{\mathbb{P}} \left[\mathrm{A}^{\textsc{Sponge}_{\mathbf{f}}}(.) = 1 \right] - \underset{\mathbf{h} \leftarrow \mathfrak{R}}{\mathbb{P}} \left[\mathrm{A}^{\mathbf{g}}(.) = 1 \right] \right| \leq \boldsymbol{\epsilon}, \tag{7}$$

where $\mathcal{S} = \{0,1\}^{r+c}$, and $\mathfrak{R}$ is defined according to Definition 6.

Proof. The proof follows closely the proof of Theorem 2 of [2]. Even though we give more power to the adversary giving her access to a quantum computer, the queries are considered to be classical. All arguments in the proof of Bertoni and others depend only on the queries made by the adversary and not her computing power. For that reason we can use the result of [16], which states that a query-based classical result easily translates to the quantum case if we do not change the query model. □

4 Random Sponges are quantumly indistinguishable from random oracles

We want to show that the distribution corresponding to random sponges is quantumly indistinguishable from a random oracle. We can define a family of distributions indexed by the security parameter that intuitively gets closer to a random oracle with increasing parameter. For that reason Theorem 4 is a perfect theoretical tool to be used. The relevant tasks that remain are to identify the family of distributions that correspond to our figure of merit, to show that in fact the most secure member of the family with $t = \infty$ is a random oracle, and to prove that the assumptions of Theorem 4 are fulfilled.

The security parameter in SPONGE is the capacity; we parametrize the family of random sponges by the size of the inner state space $t = 2^c$. Intuitively speaking, for $c \to \infty$ each evaluation of the internal function is done with a different inner state. In this case irrespective of the input, the output is a completely random string, which is the definition of a random oracle (RO). Hence we conclude that we identified a family of distributions that is well suited to be used with Theorem 4. If we show that indeed for $t = \infty$ the member of the family is the random oracle we have that:

$$\begin{aligned} &\mathfrak{F}_{2^c} \text{ is quantumly indistinguishable from } \mathfrak{F}_\infty \\ \Rightarrow\ &\text{random sponge is quantumly indisitinguishable from RO.} \end{aligned} \tag{8}$$

We are left with the task to prove the left-hand side of the above statement. The assumption of Theorem 4 is that the probability of witnessing any input-output behavior on q queries is a polynomial in $1/2^c$. At this point we stumble upon a problem with the set of indices. If we want to use the statement about closeness of polynomials we have to show that $\mathbf{p}$ is a polynomial for any inverse integer and not only for 2^{-c}. This difficulty brings us to the definition of the *generalized sponge construction* SPGEN. The only difference between SPGEN and SPONGE is the space of inner states, we change it from $\{0,1\}^c$ to any finite-size set $\mathcal{C}$. This modification solves the problem of defining distributions for any integer, not only powers of 2. It remains to prove that $\mathbf{p}(|\mathcal{C}|^{-1})$ is in fact a polynomial in $|\mathcal{C}|^{-1}$, where by $|\mathcal{C}|$ we denote cardinality of the set. With that statement proven we fulfill the assumptions of Theorem 4 and show quantum indistinguishability of SPGEN, which implies the same for SPONGE.

In Algorithm 2 we present a generalization of SPONGE. The set of inner states is denoted by $\mathcal{C}$ and can be any finite set, to be specified by the user. The internal function is generalized to any map $\boldsymbol{\varphi}_\mathbf{f} : \{0,1\}^r \times \mathcal{C} \to \{0,1\}^r \times \mathcal{C}$. In the following we denote the part of the entire state S in $\{0,1\}^r$ by $\bar{S}$ and call it the *outer* part and the part in $\mathcal{C}$ by $\hat{S}$, we will refer to it as the *inner* part of a state.

Let us now formally state the main claim of this paper. We are going to focus on the internal function being modeled as a random function, in Section 7 though, we are going to cover the case of random permutations.

Theorem 8. SPGEN$_{\boldsymbol{\varphi}_\mathbf{f}}$ *for random* $\boldsymbol{\varphi}_\mathbf{f}$ *is quantumly indistinguishable from a random oracle. More concretely, for all quantum algorithms* A *making at most* q *quantum queries to* SPGEN*, such that the input length is at most* $m \cdot r$ *bits long and the output length is at most* $z \cdot r$ *bits long,*

$$\left| \underset{\boldsymbol{\varphi}_\mathbf{f} \xleftarrow{\$} \mathcal{S}^\mathcal{S}}{\mathbb{P}} \left[\mathrm{A}^{|\mathrm{SpGen}_{\boldsymbol{\varphi}_\mathbf{f}}\rangle}(.) = 1 \right] - \underset{\mathbf{h} \leftarrow \mathfrak{R}}{\mathbb{P}} \left[\mathrm{A}^{|\mathbf{h}\rangle}(.) = 1 \right] \right| < \frac{\pi^2}{6} \eta^3 |\mathcal{C}|^{-1}, \tag{9}$$

where $\eta := 2q(m + z - 2)$ *and* $\mathfrak{R}$ *is defined according to Definition 6. The domain is defined as* $\mathcal{S} = \{0,1\}^r \times \mathcal{C}$ *for some non-empty finite set* $\mathcal{C}$.

Before we prove the above theorem we state the main technical lemma.

Algorithm 2: $\text{SpGen}_{\boldsymbol{\varphi}_{\mathbf{f}}}[\text{pad}, r, \mathcal{C}]$

```
Input   : $\mathbf{M} \in \{0,1\}^*$, $\ell \geq 0$.
Output: $\mathbf{Z} \in \{0,1\}^\ell$
1  $\mathbf{P} := \text{pad}(\mathbf{M})$
2  $S := (0^r, I_{\mathcal{C}}) \in \{0,1\}^r \times \mathcal{C}$.                 // $I_{\mathcal{C}}$-initial value
3  for $i = 1$ to $|\mathbf{P}|_r$ do                                  // Absorbing phase
4      $S := (\bar{S} \oplus \mathbf{P}_i, \hat{S})$
5      $S := \boldsymbol{\varphi}_{\mathbf{f}}(S)$
6  $\mathbf{Z} := \bar{S}$                                                 // Squeezing phase
7  while $|\mathbf{Z}| < \ell$ do
8      $S := \boldsymbol{\varphi}_{\mathbf{f}}(S)$
9      $\mathbf{Z} := \mathbf{Z} \| \bar{S}$
10 Output $\lfloor \mathbf{Z} \rfloor_\ell$
```

Lemma 9. *For a fixed q and for every $(\mathbf{M}, \mathbf{Z}) := \big((\mathbf{M}^i, \mathbf{Z}^i)\big)_{i\in[2q]}$, where $\forall i \in [2q] : (\mathbf{M}^i, \mathbf{Z}^i) \in \{0,1\}^* \times \{0,1\}^*$, such that $\forall i \in [2q] : |\mathbf{M}^i|_r \leq m, |\mathbf{Z}^i|_r \leq z$, it holds that*

(i) the probability function is a polynomial in $|\mathcal{C}|^{-1}$ of degree η

$$\mathbb{P}\left[\forall i \in [2q] : \text{SpGen}_{\boldsymbol{\varphi}_{\mathbf{f}}}(\mathbf{M}^i, \ell_i) = \mathbf{Z}^i\right] = \sum_{j=0}^{\eta} a_j |\mathcal{C}|^{-j} =: \mathbf{p}(|\mathcal{C}|^{-1}) \tag{10}$$

(ii) and the coefficient

$$a_0 = \prod_{i=1}^{2q} \boldsymbol{\delta}(\mathbf{M}, \mathbf{Z}, i) 2^{-|\mathbf{Z}^i|}. \tag{11}$$

All coefficients a_j are real, and the degree of the polynomial equals $\eta := 2q(m + z - 2)$. In the equation describing a_0 we use $\boldsymbol{\delta}(\mathbf{M}, \mathbf{Z}, i)$ to denote a Boolean function that is 0 if $\mathbf{M}^i$ is input more than once and $\mathbf{Z}^i$ is not the longest output of SpGen *on $\mathbf{M}^i$ or is inconsistent with other outputs (inputting the same message for the second time should yield the same output) and is* 1 *otherwise.*

The full proof is presented in Section 6.

Proof idea. Our goal is to explicitly evaluate $\mathbb{P}\left[\forall i \in [2q] : \text{SpGen}_{\boldsymbol{\varphi}_{\mathbf{f}}}(\mathbf{M}^i, \ell_i) = \mathbf{Z}^i\right]$. We base all of our discussion on two facts: SpGen has a structure that we know and it involves multiple evaluations of the internal function $\boldsymbol{\varphi}_{\mathbf{f}}$. $\boldsymbol{\varphi}_{\mathbf{f}}$ is a random function with well specified probability of yielding some output on a given input. The main idea of our approach is to extract terms like $\mathbb{P}[\boldsymbol{\varphi}_{\mathbf{f}}(S_1) = S_2]$ for some states S_1, S_2 from the overall probability expression and evaluate them.

Let us go through a more detailed plan of the proof. Fix $(\mathbf{M}, \mathbf{Z})$ and set $\ell_i := |\mathbf{Z}^i|$. In the first step we include all intermediate states in the probabilistic event $\big(\forall i \in [2q] : \text{SpGen}_{\boldsymbol{\varphi}_{\mathbf{f}}}(\mathbf{M}^i, \ell_i) = \mathbf{Z}^i\big)$. We write explicitly all inner states and outer states not specified by the input-output pairs $(\mathbf{M}, \mathbf{Z})$. Next we rewrite the full probability expression in the form $\sum\prod \mathbb{P}[\boldsymbol{\varphi}_{\mathbf{f}}(S_1) = S_2 \mid \ldots]$. The sum comes from the fact that there are many possible intermediate states that yield the given input-output behavior. The product is the result of using Bayes' rule to isolate a single evaluation of $\boldsymbol{\varphi}_{\mathbf{f}}$ in the probability. To correctly evaluate the summands we need to analyze all states in $\mathbb{P}[\boldsymbol{\varphi}_{\mathbf{f}}(S_1) = S_2 \mid \ldots]$ from the perspective of *uniqueness*—we say a state is unique if it is input to $\boldsymbol{\varphi}_{\mathbf{f}}$ just a single time. Given a specific setup of unique states in all $2q$ evaluations of SpGen we can easily evaluate the probabilities, as the only thing we need to know is that $\boldsymbol{\varphi}_{\mathbf{f}}$ is random. The final step of the proof is to calculate the number of states in the sum. We sum over all values of states that fulfill the constraints of $\left(\forall i \in [2q] : \text{SpGen}_{\boldsymbol{\varphi}_{\mathbf{f}}}(\mathbf{M}^i, \ell_i) = \mathbf{Z}^i\right)$ and $\boldsymbol{\varphi}_{\mathbf{f}}$ being a function. The previous analysis of uniqueness of states makes it easier to include the latter constraint; non-unique states have predetermined outputs under $\boldsymbol{\varphi}_{\mathbf{f}}$ decreasing the number of possible states. After those steps we end up with an explicit expression for $\mathbb{P}\left[\forall i \in [2q] : \text{SpGen}_{\boldsymbol{\varphi}_{\mathbf{f}}}(\mathbf{M}^i, \ell_i) = \mathbf{Z}^i\right]$, which allows us to show that $\mathbf{p}$ is a polynomial of the claimed degree and its limit in $t \to \infty$, i.e. the coefficient a_0 is the probability of uniformly random outputs. $\square$

Proof of Theorem 8. Let us define a family $\mathfrak{F}_t$ indexed by $t \in \mathbb{N} \cup \{\infty\}, t > 0$. $\mathfrak{F}_t$ is a distribution on functions from $\mathcal{M} \times \mathbb{N}$ to $\mathcal{M}$, where $\mathcal{M} := \{0,1\}^*$. The family is additionally parametrized by the choice of $r \in \mathbb{N}$ and a sponge-compliant padding function pad. We define $\mathbf{h} \leftarrow \mathfrak{F}_t$ as follows:

- Choose $\boldsymbol{\varphi}_{\mathbf{f}}$ uniformly at random from $\mathcal{S}^{\mathcal{S}}$, where $\mathcal{S} := \{0,1\}^r \times \mathcal{C}$ and $\mathcal{C}$ is any finite set of size $t > 0$.
- Use $\boldsymbol{\varphi}_{\mathbf{f}}$, $\mathcal{C}$, the fixed r, and pad to construct $\text{SpGen}_{\boldsymbol{\varphi}_{\mathbf{f}}}[\text{pad}, r, \mathcal{C}]$.
- For each $(X, \ell) \in \mathcal{M} \times \mathbb{N}$ set $\mathbf{h}(X, \ell) := \text{SpGen}_{\boldsymbol{\varphi}_{\mathbf{f}}}[\text{pad}, r, \mathcal{C}](X, \ell)$.

To show that we defined $\mathfrak{F}_t$ in the right way, let us analyze Eq. (8) from the point of view of the newly defined distribution. On the one hand from our definition it follows that

$$\underset{\mathbf{h}\leftarrow\mathfrak{F}_t}{\mathbb{P}}\left[\mathrm{A}^{|\mathbf{h}\rangle}() = 1\right] = \underset{\mathbf{h}\leftarrow\mathfrak{F}_t}{\mathbb{P}}\left[\mathrm{A}^{|\text{SpGen}_{\boldsymbol{\varphi}_{\mathbf{f}}}\rangle}() = 1\right] = \underset{\boldsymbol{\varphi}_{\mathbf{f}}\overset{\$}{\leftarrow}\mathcal{S}^{\mathcal{S}}}{\mathbb{P}}\left[\mathrm{A}^{|\text{SpGen}_{\boldsymbol{\varphi}_{\mathbf{f}}}\rangle}() = 1\right], \tag{12}$$

where the first equality follows from our definition of $\mathbf{h}$ and the second from the fact that all randomness in $\mathfrak{F}_t$ comes from choosing a random function $\boldsymbol{\varphi}_{\mathbf{f}}$. On the other hand if we take $t \to \infty$ the internal function is going to be injective on its inner part. Namely $\hat{\boldsymbol{\varphi}}_{\mathbf{f}}$—the internal function with its output restricted to the inner part—is injective. That implies a different inner state in every evaluation of $\boldsymbol{\varphi}_{\mathbf{f}}$ in SpGen what in turn implies a random and independent outer part in every step of generating the output, formally

$$\underset{\mathbf{h}\leftarrow\mathfrak{F}_\infty}{\mathbb{P}}\left[\mathrm{A}^{|\mathbf{h}\rangle}() = 1\right] = \underset{\mathbf{h}\leftarrow\mathfrak{R}}{\mathbb{P}}\left[\mathrm{A}^{|\mathbf{h}\rangle}() = 1\right]. \tag{13}$$

This intuition is formally captured by Statement (*ii*) of Lemma 9, where we state that in the limit of $|\mathcal{C}| \to \infty$ the probability of getting particular outputs of SpGen is the same as for a random oracle.

From the above discussion we get that

$$\left|\underset{\mathbf{h}\leftarrow\mathfrak{F}_t}{\mathbb{P}}\left[\mathrm{A}^{|\mathbf{h}\rangle}() = 1\right] - \underset{\mathbf{h}\leftarrow\mathfrak{F}_\infty}{\mathbb{P}}\left[\mathrm{A}^{|\mathbf{h}\rangle}() = 1\right]\right| =$$
$$\left|\underset{\boldsymbol{\varphi}_{\mathbf{f}}\overset{\$}{\leftarrow}\mathcal{S}^{\mathcal{S}}}{\mathbb{P}}\left[\mathrm{A}^{|\text{SpGen}_{\boldsymbol{\varphi}_{\mathbf{f}}}\rangle}() = 1\right] - \underset{\mathbf{h}\leftarrow\mathfrak{R}}{\mathbb{P}}\left[\mathrm{A}^{|\mathbf{h}\rangle}() = 1\right]\right|, \tag{14}$$

which is the crucial equality for using Theorem 4 to prove our statement. The last element of the proof is the assumption about $\mathbf{p}$ being a polynomial and that is exactly the statement of Lemma 9. $\square$

Quantum indistinguishability of commonly used sponges with binary state follows directly from the general result.

Corollary 10. *If* $\mathbf{f}$ *is a random function or a random permutation, then* $\text{Sponge}_{\mathbf{f}}$ *is quantumly indistinguishable from a random oracle.*

Proof. For a random function we use Theorem 8 and for a random permutation Theorem 16 and set $\mathcal{C} = \{0,1\}^c$. $\square$

4.1 Application to keyed-internal-function sponges

We show that Theorem 8 implies that keyed-internal-function sponges are indistinguishable from a random oracle under quantum access if the used internal function is a quantum-secure PRF (or if the internal function is a permutation, a quantum-secure PRP). This means that in the case $\mathbf{f}$ is a quantum-secure pseudorandom function or permutation the sponge construction is a quantum-secure pseudorandom function. For keyed primitives, indistinguishability from a random oracle/permutation is exactly what we call pseudorandomness.

We first formally define *quantum-secure* pseudorandom functions (PRF) and pseudorandom permutations (PRP).

Definition 11 (Quantum-secure PRF/PRP). *Say $\mathbf{f} : \mathcal{K} \times \mathcal{S} \to \mathcal{S}$ is a keyed function (permutation), then we say that $\mathbf{f}$ is a quantum-secure pseudorandom function (permutation) if for every quantum algorithm running in polynomial time, there is a negligible function $\boldsymbol{\epsilon}^{\mathrm{PR}}$ such that*

$$\left| \underset{K \overset{\$}{\leftarrow} \mathcal{K}}{\mathbb{P}} \left[\mathrm{A}^{|\mathbf{f}_K\rangle}(.) = 1 \right] - \underset{\mathbf{g} \overset{\$}{\leftarrow} \mathcal{S}^{\mathcal{S}}}{\mathbb{P}} \left[\mathrm{A}^{|\mathbf{g}\rangle}(.) = 1 \right] \right| \leq \boldsymbol{\epsilon}^{\mathrm{PR}}(n), \tag{15}$$

where $n := \lfloor \log |\mathcal{K}| \rfloor$ and $\mathbf{g}$ is sampled uniformly from the set of functions (permutations) from $\mathcal{S}$ to $\mathcal{S}$. Below, we refer to $\boldsymbol{\epsilon}^{\mathrm{PR}}$ as advantage.

Now we state and prove a quantum version of Theorem 1 of [1] which formalizes the above statement about quantum security of keyed-internal-function sponges. Note that we state the theorem for the general sponge construction but thanks to Corollary 10 it holds for the regular construction as well.

Theorem 12. *If the internal function $\mathbf{f}$ used in $\mathrm{SpGen}_{\mathbf{f}}$ is a quantum-secure PRF/PRP with advantage $\boldsymbol{\epsilon}^{\mathrm{PR}}$, then the resulting keyed-internal-function sponge is a quantum-secure PRF with advantage*

$$\left| \underset{K \overset{\$}{\leftarrow} \mathcal{K}}{\mathbb{P}} \left[\mathrm{A}^{|\mathrm{SpGen}_{\mathbf{f}_K}\rangle}(.) = 1 \right] - \underset{\mathbf{g} \leftarrow \mathfrak{R}}{\mathbb{P}} \left[\mathrm{A}^{|\mathbf{g}\rangle}(.) = 1 \right] \right| \leq \boldsymbol{\epsilon}^{\mathrm{PR}} + \frac{\pi^2}{6} \eta^3 |\mathcal{C}|^{-1}, \tag{16}$$

where $\eta := 2q(m + z - 2)$, q is the number of queries A makes to its oracle, m and z are as defined in the statement of Thm. 8, and $\mathfrak{R}$ is defined according to Definition 6.

Proof. We give the proof for $\mathbf{f}$ being a keyed function. The proof when $\mathbf{f}$ is a keyed permutation is obtained by using Theorem 16 in place of Theorem 8 and restricting the sets from which $\mathbf{g}$ and $\boldsymbol{\varphi}_{\mathbf{f}}$ are drawn below to permutations.

We show that the advantage of any quantum adversary in distinguishing the keyed-internal-function sponge from a random oracle is bound by its ability to distinguish $\mathbf{f}$ from a random oracle (permutation, respectively) plus its ability to distinguish a random sponge from a random oracle. In the following calculation we use the triangle inequality and the result of Theorem 8.

$$\begin{aligned} &\left| \underset{K \overset{\$}{\leftarrow} \mathcal{K}}{\mathbb{P}} \left[\mathrm{A}^{|\mathrm{SpGen}_{\mathbf{f}_K}\rangle}(.) = 1 \right] - \underset{\mathbf{g} \leftarrow \mathfrak{R}}{\mathbb{P}} \left[\mathrm{A}^{|\mathbf{g}\rangle}(.) = 1 \right] \right| \\ = &\left| \underset{K \overset{\$}{\leftarrow} \mathcal{K}}{\mathbb{P}} \left[\mathrm{A}^{|\mathrm{SpGen}_{\mathbf{f}_K}\rangle}(.) = 1 \right] - \underset{\boldsymbol{\varphi}_{\mathbf{f}} \overset{\$}{\leftarrow} \mathcal{S}^{\mathcal{S}}}{\mathbb{P}} \left[\mathrm{A}^{|\mathrm{SpGen}_{\boldsymbol{\varphi}_{\mathbf{f}}}\rangle}(.) = 1 \right] + \right. \\ &\left. \underset{\boldsymbol{\varphi}_{\mathbf{f}} \overset{\$}{\leftarrow} \mathcal{S}^{\mathcal{S}}}{\mathbb{P}} \left[\mathrm{A}^{|\mathrm{SpGen}_{\boldsymbol{\varphi}_{\mathbf{f}}}\rangle}(.) = 1 \right] - \underset{\mathbf{g} \leftarrow \mathfrak{R}}{\mathbb{P}} \left[\mathrm{A}^{|\mathbf{g}\rangle}(.) = 1 \right] \right| \end{aligned} \tag{17}$$

$$\leq \underbrace{\left| \underset{K \xleftarrow{\$} \mathcal{K}}{\mathbb{P}} \left[\mathrm{A}^{|\textsc{SpGen}_{\mathbf{f}_K}\rangle}(.) = 1 \right] - \underset{\boldsymbol{\varphi}_{\mathbf{f}} \xleftarrow{\$} \mathcal{S}^{\mathcal{S}}}{\mathbb{P}} \left[\mathrm{A}^{|\textsc{SpGen}_{\boldsymbol{\varphi}_{\mathbf{f}}}\rangle}(.) = 1 \right] \right|}_{\leq \left| \underset{K \xleftarrow{\$} \mathcal{K}}{\mathbb{P}} \left[\mathrm{B}^{|\mathbf{f}_K\rangle}(.)=1 \right] - \underset{\boldsymbol{\varphi}_{\mathbf{f}} \xleftarrow{\$} \mathcal{S}^{\mathcal{S}}}{\mathbb{P}} \left[\mathrm{B}^{|\boldsymbol{\varphi}_{\mathbf{f}}\rangle}(.)=1 \right] \right|} +$$

$$\underbrace{\left| \underset{\boldsymbol{\varphi}_{\mathbf{f}} \xleftarrow{\$} \mathcal{S}^{\mathcal{S}}}{\mathbb{P}} \left[\mathrm{A}^{|\textsc{SpGen}_{\boldsymbol{\varphi}_{\mathbf{f}}}\rangle}(.) = 1 \right] - \underset{\mathbf{g} \leftarrow \mathfrak{R}}{\mathbb{P}} \left[\mathrm{A}^{|\mathbf{g}\rangle}(.) = 1 \right] \right|}_{\text{Quantum Indistinguishability, Thm. 8 or 16}} \leq \boldsymbol{\epsilon}^{\mathrm{PR}} + \frac{\pi^2}{3} \eta^3 |\mathcal{C}|^{-1}, \tag{18}$$

where B is an adversary that uses A as a subroutine, simulating A's oracle using its own oracle and the sponge construction. B outputs the same output as A. □

5 Example proof of Lemma 9

In this section we prove Lemma 9 in a setting limited enough that every step can be done in all details. The main difficulty of our technique is of combinatorial nature, namely counting the possible values of intermediate states in multiple evaluations of SpGen. In the full proof we provide an algorithmic explanation of some steps but here we can execute these algorithms and explicitly write down their outputs.

We want to show that the probability function describing the input-output behavior of SpGen is a polynomial of bounded degree in $|\mathcal{C}|^{-1}$. By that we mean that the expression for $\mathbf{p}(|\mathcal{C}|^{-1})$ can be written as $\sum_i a_i |\mathcal{C}|^{-i}$. The proof goes as follows: Firstly we expand the event that on some inputs SpGen gives some outputs, this allows us to pinpoint the individual evaluations of $\boldsymbol{\varphi}_{\mathbf{f}}$. Secondly we impose an order on the evaluations of the internal function; which in turn allows us to exclude state values that would require $\boldsymbol{\varphi}_{\mathbf{f}}$ to output different values on the same input, calculate the probability of $\boldsymbol{\varphi}_{\mathbf{f}}$ having particular input-output behavior, and divide the set of state values in a way allowing to calculate its size. Finally we obtain a closed expression for $\mathbf{p}(|\mathcal{C}|^{-1})$.

The limitation we make in the example proof is to consider only single-query algorithms ($q = 1$). We also restrict ourselves to a *limited* SpGen that allows only 2-block inputs and always outputs a single r-bit block. As $q = 1$ the number of input-output pairs we need to consider is 2. The array of inputs and outputs is $(\mathbf{M}, \mathbf{Z}) = ((\mathbf{M}^1, \mathbf{Z}^1), (\mathbf{M}^2, \mathbf{Z}^2))$ and for $i \in \{1, 2\} : \mathbf{M}^i = M_1^i \| M_2^i$, $\mathbf{Z}^i = Z_1^i$. In the following example $\boldsymbol{\varphi}_{\mathbf{f}} : \mathcal{S} \to \mathcal{S}$, where $\mathcal{S} := \{0,1\}^r \times \mathcal{C}$.

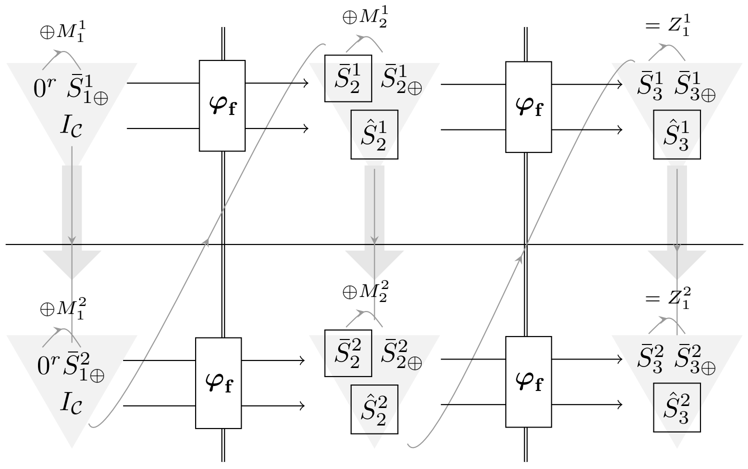
The probabilistic event we analyze throughout this section is

$$\forall i \in [2] : \textsc{SpGen}(\mathbf{M}^i) = \mathbf{Z}^i \Leftrightarrow \forall i \in [2] : \bar{\boldsymbol{\varphi}}_{\mathbf{f}} \left(\bar{\boldsymbol{\varphi}}_{\mathbf{f}}(M_1^i, I_{\mathcal{C}}) \oplus M_2^i, \hat{\boldsymbol{\varphi}}_{\mathbf{f}}(M_1^i, I_{\mathcal{C}}) \right) = Z_1^i, \tag{19}$$

where by $\bar{\boldsymbol{\varphi}}_{\mathbf{f}} : \mathcal{S} \to \{0,1\}^r$ and $\hat{\boldsymbol{\varphi}}_{\mathbf{f}} : \mathcal{S} \to \mathcal{C}$ we denote the first part and the second part of the output of $\boldsymbol{\varphi}_{\mathbf{f}}$ respectively. In the following paragraph we are going to make explicit all inputs to $\boldsymbol{\varphi}_{\mathbf{f}}$. Throughout this section we will discuss two evaluations of SpGen which are depicted in Fig. 2. In Figure 2 we show the two evaluations of SpGen we analyze. The values of not-boxed-states are fixed by the requirement of inputs being $\mathbf{M}$ and outputs $\mathbf{Z}$.

By M_j^i we denote the j-th block of $\mathbf{M}^i$, and similarly by Z_j^i the j-th block of $\mathbf{Z}^i$. Note that by including intermediate states we can further expand the above event. By *intermediate states* we mean the value of the state of SpGen during calculation of SpGen($\mathbf{M}$). Namely

$$\begin{aligned} \forall i : &\bar{\boldsymbol{\varphi}}_{\mathbf{f}} \left(\bar{\boldsymbol{\varphi}}_{\mathbf{f}}(M_1^i, I_{\mathcal{C}}) \oplus M_2, \hat{\boldsymbol{\varphi}}_{\mathbf{f}}(M_1^i, I_{\mathcal{C}}) \right) = Z_1^i \Leftrightarrow \\ \forall i : &\bigvee_{\hat{S}_3^i \in \mathcal{C}} \boldsymbol{\varphi}_{\mathbf{f}} \left(\bar{\boldsymbol{\varphi}}_{\mathbf{f}}(M_1^i, I_{\mathcal{C}}) \oplus M_2, \hat{\boldsymbol{\varphi}}_{\mathbf{f}}(M_1^i, I_{\mathcal{C}}) \right) = (Z_1^i, \hat{S}_3^i). \end{aligned} \tag{20}$$



$\oplus M_1^1$
$0^r\ \bar{S}_{1\oplus}^1$
$I_{\mathcal{C}}$
$\boldsymbol{\varphi}_{\mathbf{f}}$
$\oplus M_2^1$
$\boxed{\bar{S}_2^1}\ \bar{S}_{2\oplus}^1$
$\boxed{\hat{S}_2^1}$
$\boldsymbol{\varphi}_{\mathbf{f}}$
$= Z_1^1$
$\bar{S}_3^1\ \bar{S}_{3\oplus}^1$
$\boxed{\hat{S}_3^1}$
$\oplus M_1^2$
$0^r\bar{S}_{1\oplus}^2$
$I_{\mathcal{C}}$
$\boldsymbol{\varphi}_{\mathbf{f}}$
$\oplus M_2^2$
$\boxed{\bar{S}_2^2}\ \bar{S}_{2\oplus}^2$
$\boxed{\hat{S}_2^2}$
$\boldsymbol{\varphi}_{\mathbf{f}}$
$= Z_1^2$
$\bar{S}_3^2\ \bar{S}_{3\oplus}^2$
$\boxed{\hat{S}_3^2}$

Figure 2: Table showing the intermediate states of the limited SpGen. Boxed states are the elements of $\nabla\text{-}\mathsf{c}$ that do not have a fixed value across different $\nabla\text{-}\mathsf{c} \in \nabla\text{-}\mathsf{C}(\mathbf{M}, \mathbf{Z})$, the arrows indicate the order in which Flag-Assign assigns flags.

We are using the upper index to count the number of the evaluation of SpGen. Note that there is one inner state that we have not made explicit, the one being output by the first $\boldsymbol{\varphi}_{\mathbf{f}}$. Following the above reasoning we get

$$\forall i : \bigvee_{S_2^i \in \{0,1\}^r \times \mathcal{C}} \bigvee_{\hat{S}_3^i \in \mathcal{C}} \boldsymbol{\varphi}_{\mathbf{f}}(M_1^i, I_{\mathcal{C}}) = S_2^i \ \wedge\ \boldsymbol{\varphi}_{\mathbf{f}}(\bar{S}_2^i \oplus M_2^i, \hat{S}_2^i) = (Z_1^i, \hat{S}_3^i), \tag{21}$$

where $S_2^i = (\bar{S}_2^i, \hat{S}_2^i)$, we denote the above as

$$\forall i : \bigvee_{S_2^i \in \{0,1\}^r \times \mathcal{C}} \bigvee_{\hat{S}_3^i \in \mathcal{C}} \boldsymbol{\varphi}_{\mathbf{f}}(S_{1\oplus}^i) = S_2^i \ \wedge\ \boldsymbol{\varphi}_{\mathbf{f}}(S_{2\oplus}^i) = (\bar{S}_3^i, \hat{S}_3^i),$$
$$\text{where } \forall i : S_{1\oplus}^i := (M_1^i, I_{\mathcal{C}}) \ \wedge\ S_{2\oplus}^i := (\bar{S}_2^i \oplus M_2^i, \hat{S}_2^i) \ \wedge\ \bar{S}_3^i := Z_1^i. \tag{22}$$

By adding the subscript "$\oplus$" we highlight that the state output by $\boldsymbol{\varphi}_{\mathbf{f}}$ has been updated by XORing the appropriate block of $\mathbf{M}$. Up to this point we have expanded the initial event from Eq. (19) to a form with all inputs and outputs of $\boldsymbol{\varphi}_{\mathbf{f}}$ being explicit, namely $\boldsymbol{\varphi}_{\mathbf{f}}(S_1) = S_2$. From now on we are going to denote the set of the states by $\nabla\text{-}\mathsf{c}$ (read as "*nabla configuration*", where the ∇ is suggested by the three values that can be seen as vertices of a triangle), which we define as a matrix

$$\nabla\text{-}\mathsf{c} := \begin{bmatrix} \begin{pmatrix} \bar{S}_1^1\ \bar{S}_{1\oplus}^1 \\ \hat{S}_1^1 \end{pmatrix} & \begin{pmatrix} \bar{S}_2^1\ \bar{S}_{2\oplus}^1 \\ \hat{S}_2^1 \end{pmatrix} & \begin{pmatrix} \bar{S}_3^1\ \bar{S}_{3\oplus}^1 \\ \hat{S}_3^1 \end{pmatrix} \\ \begin{pmatrix} \bar{S}_1^2\ \bar{S}_{1\oplus}^2 \\ \hat{S}_1^2 \end{pmatrix} & \begin{pmatrix} \bar{S}_2^2\ \bar{S}_{2\oplus}^2 \\ \hat{S}_2^2 \end{pmatrix} & \begin{pmatrix} \bar{S}_3^2\ \bar{S}_{3\oplus}^2 \\ \hat{S}_3^2 \end{pmatrix} \end{bmatrix}, \tag{23}$$

where $\forall i : S_{1\oplus}^i = (M_1^i, I_{\mathcal{C}}) \wedge S_{2\oplus}^i = (\bar{S}_2^i \oplus M_2^i, \hat{S}_2^i) \wedge \bar{S}_3^i = Z_1^i = \bar{S}_{3\oplus}^i$; the constraints we impose fix the input-output behavior of the two evaluations of SpGen. Nabla-configurations $\nabla\text{-}\mathsf{c}$ are matrices of triples but when we want to refer to a part of the triple in row i and column j of $\nabla\text{-}\mathsf{c}$ we are going to write $S_j^i \in \nabla\text{-}\mathsf{c}$. More formally $S_j^i \in \nabla\text{-}\mathsf{c} \Leftrightarrow \nabla\text{-}\mathsf{c}_j^i = \begin{pmatrix} \bar{S}\ \bar{S}_\oplus \\ \hat{S} \end{pmatrix} \wedge S_j^i = (\bar{S}, \hat{S})$, where by $\nabla\text{-}\mathsf{c}_j^i$ we denote the element of $\nabla\text{-}\mathsf{c}$ in row i and column j, similarly for the second part of the state $S^i_{j\oplus} = (\bar{S}^i_{j\oplus}, \hat{S}^i_j)$, of the corresponding triple. We introduce this notation of ∇-c to capture possible values of the states in $\textsc{SpGen}(\mathbf{M}^i)$ that are consistent with $(\mathbf{M}, \mathbf{Z})$.

The set of all possible values of states in ∇-c is denoted by $\nabla\text{-C}(\mathbf{M}, \mathbf{Z})$ (*the set of nabla configurations*). The size of $\nabla\text{-C}(\mathbf{M}, \mathbf{Z})$ is the number of different ∇-c for a particular $(\mathbf{M}, \mathbf{Z})$,

$$|\,\nabla\text{-C}(\mathbf{M}, \mathbf{Z})\,| = |(\{0,1\}^r)^2 \times \mathcal{C}^4| = 2^{2r} \cdot |\mathcal{C}|^4. \tag{24}$$

In Figure 2 values of not-boxed-states are fixed by the requirement of inputs being $\mathbf{M}$ and outputs $\mathbf{Z}$. In what follows we analyze this set to find out how many possible values of states correspond to each value of probability of seeing $(\forall i \in [2] : \textsc{SpGen}(\mathbf{M}^i) = \mathbf{Z}^i)$. To better understand our approach we should clarify the implicit equivalence between ∇-c—so values of the internal states of SpGen—and $\varphi_\mathbf{f}$—the function taken at random from $\mathcal{S}^\mathcal{S}$. Note that for every $\varphi_\mathbf{f} \in \mathcal{S}^\mathcal{S}$ we have at most a single ∇-c, we say at most because some $\varphi_\mathbf{f}$ are not consistent with the input-output pairs $(\mathbf{M}, \mathbf{Z})$. On the other hand for a single ∇-c we have plenty of functions: all those that have input-output pairs consistent with values of states in ∇-c and any outputs on all inputs not present in ∇-c. Also note that there are many ∇-c that will result in $(\forall i \in [2] : \textsc{SpGen}(\mathbf{M}^i) = \mathbf{Z}^i)$. What we do is basically counting the number of functions $\varphi_\mathbf{f}$ that will result in SpGen evaluating to $\mathbf{Z}$ on $\mathbf{M}$ and dividing it by the number of all functions. The only difference is that we immediately simplify the result by not counting the functions with behavior outside of our scope—limited to few (in this section four) evaluations. This simplification is made easier by focusing on relevant values of inputs and outputs; on a few rows of the evaluation tables of $\varphi_\mathbf{f}$.

The events we take the OR of are disjoint, so in terms of probabilities we get

$$\begin{aligned}&\mathop{\mathbb{P}}_{\varphi_\mathbf{f} \xleftarrow{\$} \mathcal{S}^\mathcal{S}}[\forall i \in [2] : \textsc{SpGen}(\mathbf{M}^i) = \mathbf{Z}^i] = \\ &\sum_{S^1_2, S^2_2, \hat{S}^1_3, \hat{S}^2_3} \mathop{\mathbb{P}}_{\varphi_\mathbf{f} \xleftarrow{\$} \mathcal{S}^\mathcal{S}}[\forall i \in [2] : \varphi_\mathbf{f}(S^i_{1\oplus}) = S^i_2 \wedge \varphi_\mathbf{f}(S^i_{2\oplus}) = (\bar{S}^i_3, \hat{S}^i_j)],\end{aligned} \tag{25}$$

where the sum is taken over $S^1_2, S^2_2 \in \mathcal{S}$ and $\hat{S}^1_3, \hat{S}^2_3 \in \mathcal{C}$ and $\bar{S}^i_{1\oplus}, \bar{S}^i_{2\oplus}, \bar{S}^i_{3\oplus}$ are constrained by $(\mathbf{M}, \mathbf{Z})$. Now that we have exposed the individual evaluations of $\varphi_\mathbf{f}$ we can use the chain rule to specify the order in which we analyze the evaluations of the internal function. This order is only a tool for the analysis of the probability, not the actual time evolution. Note that the probability on the right hand side of Eq. (25) is taken over a conjunction of events depending only on a single evaluation of $\varphi_\mathbf{f}$. The next step is to extract events with a single evaluation of $\varphi_\mathbf{f}$. We can do it simply by using Bayes' formula and the chain rule,

$$\mathop{\mathbb{P}}_{\varphi_\mathbf{f} \xleftarrow{\$} \mathcal{S}^\mathcal{S}}[\textsc{SpGen}(\mathbf{M}) = \mathbf{Z}] = \sum_{S^1_2, S^2_2, \hat{S}^1_3, \hat{S}^2_3} \mathbb{P}[\forall i : \varphi_\mathbf{f}(S^i_{1\oplus}) = S^i_2 \ \wedge\ \varphi_\mathbf{f}(S^i_{2\oplus}) = (\bar{S}^i_3, \hat{S}^i_3)] \tag{26}$$

$$\begin{aligned}&= \sum_{\nabla\text{-c}} \mathbb{P}[\forall i : \varphi_\mathbf{f}(S^i_{1\oplus}) = S^i_2 \ \wedge\ \varphi_\mathbf{f}(S^i_{2\oplus}) = S^i_3; \forall i, j : S^i_j, S^i_{j\oplus} \in \nabla\text{-c}] \\ &= \sum_{\nabla\text{-c}} \mathbb{P}[\varphi_\mathbf{f}(S^1_{1\oplus}) = S^1_2] \\ &\qquad \cdot \mathbb{P}[\varphi_\mathbf{f}(S^2_{1\oplus}) = S^2_2 \mid \varphi_\mathbf{f}(S^1_{1\oplus}) = S^1_2] \\ &\qquad \cdot \mathbb{P}[\varphi_\mathbf{f}(S^1_{2\oplus}) = S^1_3 \mid \varphi_\mathbf{f}(S^2_{1\oplus}) = S^2_2 \ \wedge\ \varphi_\mathbf{f}(S^1_{1\oplus}) = S^1_2] \\ &\qquad \cdot \mathbb{P}[\varphi_\mathbf{f}(S^2_{2\oplus}) = S^2_3 \mid \varphi_\mathbf{f}(S^1_{2\oplus}) = S^1_3 \ \wedge\ \varphi_\mathbf{f}(S^2_{1\oplus}) = S^2_2 \ \wedge\ \varphi_\mathbf{f}(S^1_{1\oplus}) = S^1_2],\end{aligned} \tag{27}$$

where in the last equation we have omitted $\forall i, j : S^i_j, S^i_{j\oplus} \in \nabla\text{-c}$ in each probability function. We denote the order specified above by "$\prec$".

We still cannot evaluate the above expression because we do not know if $\varphi_\mathbf{f}$ is queried on a "fresh" input or not. First of all, note that thanks to conditioning on one event, we can treat

$(\varphi_{\mathbf{f}}(S^1_{1\oplus}) = S^1_2)$ from the second factor in Eq. (27) as being prior to $(\varphi_{\mathbf{f}}(S^1_{2\oplus}) = S^1_3)$. Prior in that case means that $\varphi_{\mathbf{f}}$ is sampled on $S^1_{1\oplus}$ before it is sampled on $S^2_{1\oplus}$. That implies, e.g., that if $S^2_{1\oplus} = S^1_{1\oplus}$ then the outputs have to be the same, otherwise the probability is 0. This is what we mean when saying that an input is fresh or not. To separate a particular ∇-c with different numbers of fresh states, we perform a procedure on each ∇-c that assigns flags to the states. Flags mark whether the value of the state was previously input to $\varphi_{\mathbf{f}}$ or not. By performing this procedure we want to divide ∇-C$(\mathbf{M}, \mathbf{Z})$ into subsets with the same probability—i.e. having the same probability of sampling $\varphi_{\mathbf{f}}$ that yields a particular input-output behavior. Let us call this procedure Flag-Assign. Running it also identifies impossible values of internal states, calculates the probabilities of each transition, and divides ∇-C$(\mathbf{M}, \mathbf{Z})$ into sets of cardinalities we can compute.

Algorithm Flag-Assign, see Alg. 4 in the next section, takes as input ∇-c and goes through each state starting from the first column going down, then down from the top of the second column and so on. The order in which Flag-Assign operates is depicted by arrows in Fig. 4. If the value of the "$\oplus$" part of the state which is input to $\varphi_{\mathbf{f}}$ appears in ∇-c just once, the algorithm assigns the flag "$\boxed{\text{u}}$" to it, we call such states *unique*. If the value is not unique, i.e. it appears in ∇-c more than once, the state that is encountered first is assigned the flag "$\boxed{\text{f}}$" and the rest of the states get the flag "$\boxed{\text{n}}$". We call states with the flag "$\boxed{\text{n}}$" *non-unique*. Flag-Assign also appends to each non-unique "$\oplus$"-state the output it should yield, i.e. the output of the corresponding "$\boxed{\text{f}}$" state. If the state in ∇-c that follows the considered state is different than the claimed output we discard the whole configuration. We denote the set ∇-C$(\mathbf{M}, \mathbf{Z})$ without states that conflict with $\varphi_{\mathbf{f}}$ being a proper map by p-∇-CF$(\mathbf{M}, \mathbf{Z})$ (*set of p-nabla configurations with flags*, p emphasizes the fact that we have restricted $\varphi_{\mathbf{f}}$ to proper transformations). By a proper map we mean that it does not output different states on the same input. Elements of p-∇-CF$(\mathbf{M}, \mathbf{Z})$ are denoted by p-∇-cf (*p-nabla configurations*).

After running Flag-Assign on every $\nabla\text{-c} \in \nabla\text{-C}(\mathbf{M}, \mathbf{Z})$ and discarding the configurations with bad output states we still need to add more details to our picture. The procedure we describe below is depicted in Figure 3. Firstly we discriminate between p-∇-cf with different numbers of unique states. The total number of flags is 4, the final states are not inputs to $\varphi_{\mathbf{f}}$ and are not assigned a flag. We denote the number of unique states by u, the number of states that are non-unique but appear for the first time is f, and the number of non-unique states is n. Note that $u+f+n = 4$. In general there are 5 possible sets of those numbers in the case of $q = 1$ and lengths of the input and output strings we specified. These are as follows: $(u = 4, f = 0)$, $(u = 2, f = 1)$, $(u = 1, f = 1)$, $(u = 0, f = 2)$, and $(u = 0, f = 1)$. Secondly we discriminate between different placements of flags. For each setup there are several possible placements of flags. For $(u = 4, f = 0)$ and $(u = 0, f = 1)$, flags can be set in only one configuration. If we have 2 unique states and the setup is $(u = 2, f = 1)$ then there are 6 possible configurations of flags. For $(u = 1, f = 1)$ there are 4 and for $(u = 0, f = 2)$ only 2. While calculating the number of configurations it is important to remember that the flag of the first state $S^1_{1\oplus}$ is either $\boxed{\text{u}}$ or $\boxed{\text{f}}$. There are some details of how to find the placements but they are made explicit only in the full proof of the lemma. All possible placements are depicted in Figure 3.

In most steps we perform using Flag-Assign the distinction between $\boxed{\text{u}}$ and $\boxed{\text{f}}$ seems unimportant. We will need it to properly identify different placements of flags in p-∇-CF$(\mathbf{M}, \mathbf{Z})$, but indeed in all other tasks one can treat them as a single "unique" flag.

The next step is calculating the number of values that can be assigned to states in a given setup and for a given placement of flags. We calculated those numbers assuming it is possible to have such placement. This assumption is not always true as particular messages and outputs exclude some options. For example, if both messages start with the same block then all positions where the two first states are unique are impossible. By Calc we denote the algorithm calculating the cardinality of subsets of p-∇-CF$(\mathbf{M}, \mathbf{Z})$, the details of Calc are specified below in Alg. 5. It goes through a single placement of flags. The basic rules of its operation are: for

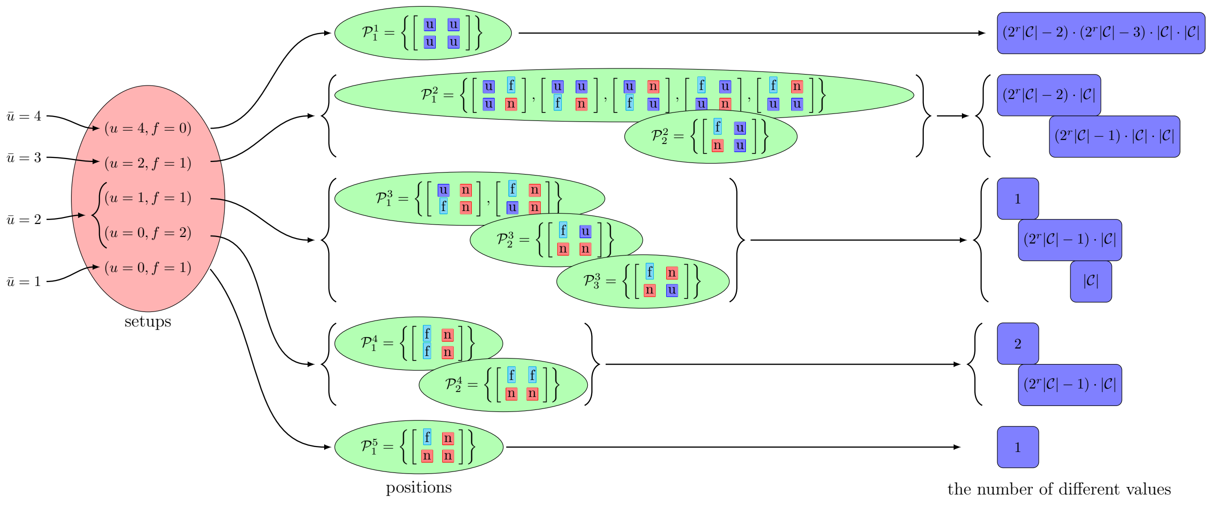

$\bar{u}=4$
$\bar{u}=3$
$\bar{u}=2$
$\bar{u}=1$
$(u=4, f=0)$
$(u=2, f=1)$
$(u=1, f=1)$
$(u=0, f=2)$
$(u=0, f=1)$
setups
positions
the number of different values
$(2^r|\mathcal{C}|-2)\cdot(2^r|\mathcal{C}|-3)\cdot|\mathcal{C}|\cdot|\mathcal{C}|$
$(2^r|\mathcal{C}|-2)\cdot|\mathcal{C}|$
$(2^r|\mathcal{C}|-1)\cdot|\mathcal{C}|\cdot|\mathcal{C}|$
1
$(2^r|\mathcal{C}|-1)\cdot|\mathcal{C}|$
$|\mathcal{C}|$
2
$(2^r|\mathcal{C}|-1)\cdot|\mathcal{C}|$
1

Figure 3: Possible placements for the limited SpGen. $\begin{bmatrix} \cdot & \cdot \\ \cdot & \cdot \end{bmatrix}$ denotes the flags assigned to the four states. $\bar{u}$ denotes the total number of unique states: $\bar{u} := u + f$.

every unique flag that maps to a unique flag multiply the result by $2^r|\mathcal{C}|$, for every unique flag that maps to a non-unique flag multiply the result by 1, for every non-unique flag that maps to any state multiply the result by 1, and for every unique flag in the last column of flagged states multiply the result by $|\mathcal{C}|$. The first two rules are adjusted a bit to keep track of non-repeating unique states and allow for multiple values of non-unique states respectively. The last column in Fig. 3 lists the results of Calc for placements in the respective rows. If the squeezing phases were longer we would have to account for the fact that outer part of the state can be either unique or not which slightly changes the final outcome.

Now we want to show that the probability function $\mathbf{p}$ is a polynomial in $|\mathcal{C}|^{-1}$. Up to this point we have shown that

$$\sum_{\nabla\text{-c}} \prod_{(i,j)=(1,1)}^{(2,2)} \mathbb{P}\Bigg[\varphi_{\mathbf{f}}(S^i_{j\oplus}) = S^i_{j+1} \mid \bigwedge_{(i',j')\prec(i,j)} \varphi_{\mathbf{f}}(S^{i'}_{j'\oplus}) = S^{i'}_{j'+1}; \forall i,j,i',j' : S^i_j, S^i_{j\oplus} \in \nabla\text{-c}\Bigg]$$
$$= \sum_{\text{p-}\nabla\text{-cf}} \prod_{(i,j)=(1,1)}^{(2,2)} \mathbb{P}\Bigg[\varphi_{\mathbf{f}}(S^i_{j\oplus}) = S^i_{j+1} \mid \bigwedge_{(i',j')\prec(i,j)} \varphi_{\mathbf{f}}(S^{i'}_{j'\oplus}) = S^{i'}_{j'+1}; \forall i,j,i',j' : S^i_j, S^i_{j\oplus} \in \text{p-}\nabla\text{-cf}\Bigg], \tag{28}$$

where the order "$\prec$" is the same as in Eq. (27). In the above equation we have discarded those ∇-c that require $\varphi_{\mathbf{f}}$ to output different states on the same input, because the probability is then 0. The sum in Eq. (28) can be expanded to

$$\sum_{\text{p-}\nabla\text{-cf}} \cdots = \sum_{u=4} \cdots + \sum_{u=0}^{2} \sum_{f=1}^{\lfloor(4-u)/2\rfloor} \sum_{P\in\mathcal{P}(u,f)} \sum_{\text{p-}\nabla\text{-cf}(u,f,P)} \cdots, \tag{29}$$

where by $\text{p-}\nabla\text{-cf}(u,f,P)$ we denote the configuration with the number of unique and non-unique states and placement fixed. $\mathcal{P}(u,f)$ is the set of all placements in which the flags can be arranged given the number of unique and non-unique states. We omitted the input $(\mathbf{M}, \mathbf{Z})$ to $\mathcal{P}$ for brevity. Making use of information from Fig. 3 we can now evaluate expression (28). Note that setting u and f to some particular values allows us to evaluate the probabilities. Denoting the total number of unique states by $\bar{u}$ we get that

$$\prod_{(i,j)=(1,1)}^{(2,2)} \mathbb{P}\left[\boldsymbol{\varphi}_{\mathbf{f}}(S^i_{j\oplus}) = S^i_{j+1} \mid \bigwedge_{(i',j')\prec(i,j)} \boldsymbol{\varphi}_{\mathbf{f}}(S^{i'}_{j'\oplus}) = S^{i'}_{j'+1}\right] = (2^r|\mathcal{C}|)^{-\bar{u}} \tag{30}$$

for $\mathsf{p}\text{-}\nabla\text{-}\mathsf{cf}$ with $u + f = \bar{u}$. Finally we arrive at

$$\begin{aligned}\mathbf{p}(|\mathcal{C}|^{-1}) = &\sum_{P\in\mathcal{P}^1_1} (2^r|\mathcal{C}|)^{-4} \cdot (2^r|\mathcal{C}|-2)(2^r|\mathcal{C}|-3)|\mathcal{C}|^2\boldsymbol{\delta}(P) + \\ &\sum_{P\in\mathcal{P}^2_1} (2^r|\mathcal{C}|)^{-3} \cdot (2^r|\mathcal{C}|-2)|\mathcal{C}|\boldsymbol{\delta}(P) + \sum_{P\in\mathcal{P}^2_2} (2^r|\mathcal{C}|)^{-3} \cdot (2^r|\mathcal{C}|-1)|\mathcal{C}|^2\boldsymbol{\delta}(P) + \\ &\sum_{P\in\mathcal{P}^3_1} (2^r|\mathcal{C}|)^{-2} \cdot \boldsymbol{\delta}(P) + \sum_{P\in\mathcal{P}^3_2} (2^r|\mathcal{C}|)^{-2} \cdot (2^r|\mathcal{C}|-1)|\mathcal{C}|\boldsymbol{\delta}(P) + \\ &\sum_{P\in\mathcal{P}^3_3} (2^r|\mathcal{C}|)^{-2} \cdot |\mathcal{C}|\boldsymbol{\delta}(P) + \sum_{P\in\mathcal{P}^4_1} (2^r|\mathcal{C}|)^{-2} \cdot 2\boldsymbol{\delta}(P) + \\ &\sum_{P\in\mathcal{P}^4_2} (2^r|\mathcal{C}|)^{-2} \cdot (2^r|\mathcal{C}|-1)|\mathcal{C}|\boldsymbol{\delta}(P) + \sum_{P\in\mathcal{P}^5_1} (2^r|\mathcal{C}|)^{-1} \cdot \boldsymbol{\delta}(P),\end{aligned} \tag{31}$$

where the sets are denoted as in Fig. 3. The function appearing in the above equation is defined as

$$\boldsymbol{\delta}(\mathbf{M}, \mathbf{Z}, P) := \begin{cases} 0 & \text{if } \mathsf{p}\text{-}\nabla\text{-}\mathsf{CF}(\mathbf{M}, \mathbf{Z})(u, f, P) = \emptyset \\ 1 & \text{otherwise} \end{cases}, \tag{32}$$

where in Eq. (31) we omitted the input of $(\mathbf{M}, \mathbf{Z})$ for readability.

The degree of $\mathbf{p}(|\mathcal{C}|^{-1})$ is at most 2, as claimed for messages of length $2r$-bits. The coefficient for $|\mathcal{C}|^0$ is

$$a_0 = \sum_{P\in\mathcal{P}^1_1} 2^{-2r}\boldsymbol{\delta}(P) + \sum_{P\in\mathcal{P}^2_2} 2^{-2r}\boldsymbol{\delta}(P) + \sum_{P\in\mathcal{P}^3_2} 2^{-r}\boldsymbol{\delta}(P) + \sum_{P\in\mathcal{P}t_2} 2^{-r}\boldsymbol{\delta}(P). \tag{33}$$

Let us recapitulate the results of this section. First we characterized the possible internal functions by the outputs of their consecutive evaluations, Eq. (26). Secondly we captured the features of the intermediate states that determine the probability of seeing a particular input-output behavior, Eq. (30). Finally we calculated an explicit formula for the probability function, Eq. (31), (33).

6 Proof of Lemma 9

In this section we give the complete proof of Lemma 9 for the general case of $q \geq 1$ queries the adversary makes and message lengths bounded by some m, not fixed to 2 like in the previous section. In Subsection 6.1 we expand the probability expression to encompass all intermediate states of $\left(\forall i \in [2q] : \textsc{SpGen}_{\boldsymbol{\varphi}_{\mathbf{f}}}(\mathbf{M}^i, \ell_i) = \mathbf{Z}^i\right)$ and individual evaluations of $\boldsymbol{\varphi}_{\mathbf{f}}$. In Subsection 6.2 we introduce the concept of unique states to evaluate the probabilities of $\mathbb{P}[\boldsymbol{\varphi}_{\mathbf{f}}(S_1) = S_2]$. In Subsection 6.3 we define the algorithm that calculates the cardinality of the set of intermediate states—and equivalently inner functions—consistent with given characteristics. In Subsection 6.4 we conclude the proof and provide the final expression for the probability of an input-output pair under a random $\textsc{SpGen}_{\boldsymbol{\varphi}_{\mathbf{f}}}$.

We omit the padding function of the sponge construction and assume that the length of all $\mathbf{M}^i$ is a multiple of r. This is done without loss of generality since we can just say that all the considered messages are in fact messages after padding and we do not use any properties of the padding in the proof. Also we focus on q evaluations of SpGen instead of $2q$ to improve readability.

6.1 Expansion of the probability function

In this section we expand the probability function to the point that all intermediate states are accounted for. We follow exactly the same reasoning as at the beginning of Section 5. We consider the event $\left(\forall i \in [q] : \textsc{SpGen}_{\boldsymbol{\varphi}_\mathbf{f}}(\mathbf{M}^i, \ell_i) = \mathbf{Z}^i\right)$ and then include the states that appear between consecutive evaluations of $\boldsymbol{\varphi}_\mathbf{f}$, similarly to the steps in Equations (19)–(21).

To keep track of the states we introduce the following notation. By the upper-index we denote the number of evaluations of SpGen, going from 1 to q. The lower index corresponds to the number of evaluations of $\boldsymbol{\varphi}_\mathbf{f}$ in the i-th calculation of SpGen. A state occurring during the calculation on $\mathbf{M}^i$ that is the input to the j-th evaluation of $\boldsymbol{\varphi}_\mathbf{f}$ is denoted by $S^i_{j\oplus}$. The output of that evaluation is S^i_{j+1}. All states traversed in q evaluations of SpGen can be depicted in a similar way as in Figure 2 but in an array with q rows with $|\mathbf{M}^i|_r + |\mathbf{Z}^i|_r$ columns each.

We call an array like that presented in Figure 2 with values assigned to every state a *nabla configuration* $\nabla\text{-c}$. ∇ symbolizes the triangle shape in which we put states between evaluations of $\boldsymbol{\varphi}_\mathbf{f}$, each corner being an outer or inner part of the state. Note that in the figure we assume the initial state to be equal to $(0^r, I_\mathcal{C})$ which is not included in our definition of $\nabla\text{-c}$. By *array* we mean a 2-dimensional matrix with unequal length of rows. Now we define $\nabla\text{-c}$ relative to input-output pairs $(\mathbf{M}, \mathbf{Z})$. The size of the array is determined by the number of blocks in $\mathbf{M}^i$ and $\mathbf{Z}^i$.

Definition 13 ($\nabla\text{-c}$). *The nabla configuration $\nabla\text{-c}$ for $(\mathbf{M}, \mathbf{Z})$ is an array of triples $\begin{pmatrix} \bar{S}\ \bar{S}_\oplus \\ \hat{S} \end{pmatrix} \in \{0,1\}^{2r} \times \mathcal{C}$, where $\mathcal{C}$ is an arbitrary non-empty finite set. The array $\nabla\text{-c}$ consists of q rows, for every i row i has k_i columns and $k_i := |\mathbf{M}^i|_r + |\mathbf{Z}^i|_r$ ($|\mathbf{M}^i|_r$ denotes the number of r-bit blocks in $\mathbf{M}^i$). Formally we have*

$$\nabla\text{-c} := \left[\begin{pmatrix} \bar{S}^i_j\ \bar{S}^i_{j\oplus} \\ \hat{S}^i_j \end{pmatrix}\right]_{\substack{i\in[q] \\ j\in[k_i]}} . \tag{34}$$

To refer to the element of $\nabla\text{-c}$ that lies in row i and column j we write $\nabla\text{-c}^i_j$. To refer to parts of the triple that lies in row i and column j we write

$$\begin{aligned} S^i_j \in \nabla\text{-c} &\Leftrightarrow \nabla\text{-c}^i_j = \begin{pmatrix} \bar{S}\ \bar{S}_\oplus \\ \hat{S} \end{pmatrix} \wedge S^i_j = (\bar{S}, \hat{S}) \\ S^i_{j\oplus} \in \nabla\text{-c} &\Leftrightarrow \nabla\text{-c}^i_j = \begin{pmatrix} \bar{S}\ \bar{S}_\oplus \\ \hat{S} \end{pmatrix} \wedge S^i_{j\oplus} = (\bar{S}_\oplus, \hat{S}) \end{aligned} \tag{35}$$

Let us define the number of evaluations of $\boldsymbol{\varphi}_\mathbf{f}$ in $\nabla\text{-c}$ for $(\mathbf{M}, \mathbf{Z})$ as

$$\kappa := \sum_{i=1}^{q} (k_i - 1), \tag{36}$$

note that $|\nabla\text{-c}| = \kappa + q$.

To make good use of the newly introduced concept of nabla configurations $\nabla\text{-c}$ we want to restrict the set of arrays we discuss. Similarly to Equation (22) we want to put constraints on the set of $\nabla\text{-c}$ to make explicit the requirement that states correspond to a correct input-output behavior of SpGen. The *set of* $\nabla\text{-c}$ *for* $(\mathbf{M}, \mathbf{Z})$ is defined as follows.

Definition 14 ($\nabla\text{-C}(\mathbf{M}, \mathbf{Z})$)**.** *The set of nabla configurations* $\nabla\text{-c}$ *for* $(\mathbf{M}, \mathbf{Z})$ *is a set of arrays of size specified by* $(\mathbf{M}, \mathbf{Z})$*,* $\nabla\text{-C}(\mathbf{M}, \mathbf{Z}) \subset \left(\{0,1\}^{2r} \times \mathcal{C}\right)^{\kappa+q}$*. We define* $\nabla\text{-C}(\mathbf{M}, \mathbf{Z})$ *by the following constraints*

$$\begin{aligned} \forall i \in [q] :\ & \hat{S}_1^i = I_{\mathcal{C}}, \\ \forall i \in [q] :\ & \bar{S}_1^i = 0^r, \\ \forall i \in [q], 1 \leq j \leq |\mathbf{M}^i|_r :\ & \bar{S}_{j\oplus}^i = \bar{S}_j^i \oplus M_j^i, \\ \forall i \in [q], |\mathbf{M}^i|_r < j \leq k_i :\ & \bar{S}_{j\oplus}^i = \bar{S}_j^i = Z_{j-|\mathbf{M}^i|_r}^i. \end{aligned} \tag{37}$$

The formal definition reads

$$\nabla\text{-C}(\mathbf{M}, \mathbf{Z}) := \{\nabla\text{-c} \text{ for } (\mathbf{M}, \mathbf{Z}) : \nabla\text{-c} \text{ fulfills constraints (37)}\}. \tag{38}$$

In the following we assume that rows of all $\nabla\text{-c} \in \nabla\text{-C}(\mathbf{M}, \mathbf{Z})$ are initially sorted according to the following relation. We arrange $(\mathbf{M}^i, \mathbf{Z}^i)$ in non-decreasing order in terms of length, so $\forall i < j : k_i \leq k_j$, this also means that rows of $\nabla\text{-c}$ are ordered in this way.

Having established the notation we move on to realizing the goal of this section: rewriting the probability function in a suitable way for further analysis. In the following when we consider $\left(\boldsymbol{\varphi}_{\mathbf{f}}(S_{j\oplus}^i) = S_{j+1}^i\right)$ for some $\nabla\text{-c}$ we leave implicit that $S_{j\oplus}^i, S_{j+1}^i \in \nabla\text{-c}$. We have that

$$\begin{aligned} \forall i \in [q] :& \textsc{SpGen}(\mathbf{M}^i) = \mathbf{Z}^i \\ &\Leftrightarrow \forall i \in [q] : \bigvee_{\nabla\text{-c} \in \nabla\text{-C}(\mathbf{M}, \mathbf{Z})} \left(\boldsymbol{\varphi}_{\mathbf{f}}(S_{1\oplus}^i) = S_2^i\right) \wedge \left(\boldsymbol{\varphi}_{\mathbf{f}}(S_{2\oplus}^i) = S_3^i\right) \wedge \cdots \wedge \left(\boldsymbol{\varphi}_{\mathbf{f}}(S_{(k_i-1)\oplus}^i) = S_{k_i}^i\right) \end{aligned} \tag{39}$$

$$\Leftrightarrow \bigvee_{\nabla\text{-c} \in \nabla\text{-C}(\mathbf{M}, \mathbf{Z})} \bigwedge_{i=1}^{q} \bigwedge_{j=1}^{k_i-1} \left(\boldsymbol{\varphi}_{\mathbf{f}}(S_{j\oplus}^i) = S_{j+1}^i\right). \tag{40}$$

In the above equations we first include the intermediate states and then combine all evaluations of $\boldsymbol{\varphi}_{\mathbf{f}}$. In the following we make use of the fact that the events we take the disjunction of are disjoint and the logical disjunction turns into a sum of the probability.

$$\begin{aligned} \underset{\mathbf{f} \xleftarrow{\$} \mathcal{S}^{\mathcal{S}}}{\mathbb{P}} \left[\forall i \in [q] : \textsc{SpGen}(\mathbf{M}^i) = \mathbf{Z}^i\right] &= \mathbb{P}\left[\bigvee_{\nabla\text{-c} \in \nabla\text{-C}(\mathbf{M}, \mathbf{Z})} \bigwedge_{i=1}^{q} \bigwedge_{j=1}^{k_i-1} \left(\boldsymbol{\varphi}_{\mathbf{f}}(S_{j\oplus}^i) = S_{j+1}^i\right)\right] \\ &= \sum_{\nabla\text{-c} \in \nabla\text{-C}(\mathbf{M}, \mathbf{Z})} \mathbb{P}\left[\bigwedge_{i=1}^{q} \bigwedge_{j=1}^{k_i-1} \left(\boldsymbol{\varphi}_{\mathbf{f}}(S_{j\oplus}^i) = S_{j+1}^i\right)\right]. \end{aligned} \tag{41}$$

To further extract an expression involving the probability of a single $\left(\boldsymbol{\varphi}_{\mathbf{f}}(S_{j\oplus}^i) = S_{j+1}^i\right)$ we use Bayes' rule. By a chain of conditions we want to arrive at a function we can evaluate in the end. At this point we want to choose a particular order of $\left(\boldsymbol{\varphi}_{\mathbf{f}}(S_{j\oplus}^i) = S_{j+1}^i\right)$ events. Let us define the order $\prec$ as

$$(i,j) \prec (i',j') \Leftrightarrow \left(j < j'\right) \vee \left(j = j' \wedge i < i'\right). \tag{42}$$

The above rule imposes an order that begins with the top-left corner of Figure 2 and proceeds downwards to the end of the column to continue from the second column from the left.

$$\begin{aligned}
\mathbf{p}(|\mathcal{C}|^{-1}) &= \sum_{\nabla\text{-c}\in\nabla\text{-C}(\mathbf{M},\mathbf{Z})} \mathbb{P}\left[\bigwedge_{i=1}^{q}\bigwedge_{j=1}^{k_i-1}\left(\boldsymbol{\varphi}_{\mathbf{f}}(S^i_{j\oplus}) = S^i_{j+1}\right)\right] \\
&= \sum_{\nabla\text{-c}} \mathbb{P}\left[\left(\boldsymbol{\varphi}_{\mathbf{f}}(S^q_{(k_q-1)\oplus}) = S^q_{k_q}\right) \mid \bigwedge_{(i,j)\prec(q,k_q-1)} \left(\boldsymbol{\varphi}_{\mathbf{f}}(S^i_{j\oplus}) = S^i_{j+1}\right)\right] \mathbb{P}\left[\bigwedge_{(i,j)\prec(q,k_q-1)} \left(\boldsymbol{\varphi}_{\mathbf{f}}(S^i_{j\oplus}) = S^i_{j+1}\right)\right] \\
&= \sum_{\nabla\text{-c}} \mathbb{P}\left[\left(\boldsymbol{\varphi}_{\mathbf{f}}(S^q_{(k_q-1)\oplus}) = S^q_{k_q}\right) \mid \bigwedge_{(i,j)\prec(q,k_q-1)} \left(\boldsymbol{\varphi}_{\mathbf{f}}(S^i_{j\oplus}) = S^i_{j+1}\right)\right] \\
&\cdot \mathbb{P}\left[\left(\boldsymbol{\varphi}_{\mathbf{f}}(S^q_{(k_q-1)\oplus}) = S^q_{k_q}\right) \mid \bigwedge_{(i,j)\prec(q-1,k_q-1)} \left(\boldsymbol{\varphi}_{\mathbf{f}}(S^i_{j\oplus}) = S^i_{j+1}\right)\right] \\
&\cdot \mathbb{P}\left[\bigwedge_{(i,j)\prec(q-1,k_q-1)} \left(\boldsymbol{\varphi}_{\mathbf{f}}(S^i_{j\oplus}) = S^i_{j+1}\right)\right] = \cdots = \\
&= \sum_{\nabla\text{-c}\in\nabla\text{-C}(\mathbf{M},\mathbf{Z})} \prod_{(i,j)=(1,1)}^{(q,k_q-1)} \mathbb{P}\left[\left(\boldsymbol{\varphi}_{\mathbf{f}}(S^i_{j\oplus}) = S^i_{j+1}\right) \mid \bigwedge_{(i',j')\prec(i,j)} \left(\boldsymbol{\varphi}_{\mathbf{f}}(S^{i'}_{j'\oplus}) = S^{i'}_{j'+1}\right)\right]. \qquad (43)
\end{aligned}$$

In the case there is no state $(q-1, k_q-1)$ we just take the next state preceding (q, k_q-1) in the order given by Equation (42).

Up to this point we have performed some transformations of the event $\left(\forall i \in [q] : \textsc{SpGen}_{\boldsymbol{\varphi}_{\mathbf{f}}}(\mathbf{M}^i, \ell_i) = \mathbf{Z}^i\right)$, but we did not address the issue of correctness. Is it correct to consider state values in evaluations of SpGen instead of different $\boldsymbol{\varphi}_{\mathbf{f}}$—are we in fact discussing the probability over the random choice of the internal function? The answer to this question is "yes", that is because of the equivalence of every ∇-c with some set of $\boldsymbol{\varphi}_{\mathbf{f}}$. We can treat the input-output pairs for $\boldsymbol{\varphi}_{\mathbf{f}}$ assigned in ∇-c as values in the function table of $\boldsymbol{\varphi}_{\mathbf{f}}$. By picking a single ∇-c we fix at most κ rows of this table. As we sample $\boldsymbol{\varphi}_{\mathbf{f}}$ uniformly at random we are interested in the fraction of functions that are consistent with the input-output pairs $(\mathbf{M}, \mathbf{Z})$ among all functions. Note however, that we only care about κ evaluations of $\boldsymbol{\varphi}_{\mathbf{f}}$ and all the details of those future evaluations are implicitly simplified in the fraction. This allows us to focus only on the part of the function table corresponding to those few evaluations and that is exactly ∇-c. The summing over nabla configurations ∇-c corresponds to different values of the function table that are still consistent with $(\mathbf{M}, \mathbf{Z})$.

The probability $\mathbb{P}\left[\left(\boldsymbol{\varphi}_{\mathbf{f}}(S^i_{j\oplus}) = S^i_{j+1}\right) \mid \bigwedge_{(i',j')\prec(i,j)} \left(\boldsymbol{\varphi}_{\mathbf{f}}(S^{i'}_{j'\oplus}) = S^{i'}_{j'+1}\right)\right]$ equals either $\frac{1}{2^r\cdot|\mathcal{C}|}$ or 1 or 0. If the internal function is queried on a "fresh" input, it outputs any value with uniform probability. If on the other hand it is queried on the same input for the second time, it outputs the value it has output before with probability 1. One might think that the proof is finished, $\mathbf{p}(\lambda) = \sum_i \mathbf{w}_i(\lambda)$, where $\mathbf{w}_i$ are monomials in λ of degree up to $\kappa + q$. There is one problem with that reasoning, namely that the sum limits depend on the variable λ. Up until now we have shown that $\mathbf{p}(\lambda) = \sum_{i=1}^{\mathbf{v}(1/\lambda)} \mathbf{w}_i(\lambda)$, where $\mathbf{v}$ is another polynomial. Even for $\mathbf{v} = \mathbf{id}$ (the identity function) the degree of $\mathbf{p}$ is different than the maximal degree of $\mathbf{w}_i$. This means that we have to analyze the expression derived in Equation (43) in more detail. To this end, we add more structure to ∇-C$(\mathbf{M}, \mathbf{Z})$ which will make it easier to count the number of values that the intermediate states can assume, i.e. the number of nabla configurations ∇-c in ∇-C$(\mathbf{M}, \mathbf{Z})$.

6.2 Unique and non-unique states

The goal of this section is to evaluate $\mathbb{P}\left[\left(\varphi_{\mathbf{f}}(S^i_{j\oplus}) = S^i_{j+1}\right) \mid \bigwedge_{(i',j')\prec(i,j)} \left(\varphi_{\mathbf{f}}(S^{i'}_{j'\oplus}) = S^{i'}_{j'+1}\right)\right]$ for any ∇-c and any $(\mathbf{M}, \mathbf{Z})$. We approach this problem by recognizing which states in a particular ∇-c are fed to $\varphi_{\mathbf{f}}$ once and which are repeated. We define an algorithm that includes the information about *uniqueness* of the intermediate states in ∇-c. The notion of uniqueness is derived relative to the events we condition on in Eq. (43), that is why we took special care of the order in which we use the chain rule.

In this section we introduce two algorithms PREP and FLAG-ASSIGN. The former is an auxiliary algorithm that prepares the array ∇-c for further analysis. The latter algorithm assigns flags to states in ∇-c. Flags signify if a state appears once or more in the array. We use an algorithmic definition to explicitly show every step of the procedure.

Algorithm 3 takes as input an array ∇-c and groups its elements according to the value input to $\varphi_{\mathbf{f}}$. An important detail is the sorting rule among states with the same "$\oplus$"-state value; we use the order defined in Eq. 42. The output of Algorithm 3 PREP(∇-c) is a vector (1-dimensional matrix), to access its l-th element we write ∇-c$_l$.

```
Algorithm 3: PREP
  Input  : $\nabla$-c for $(\mathbf{M}, \mathbf{Z})$
  Output: $\widetilde{\nabla\text{-c}}$
1 $\widetilde{\nabla\text{-c}} := \nabla$-c, append three work spaces to each element of $\widetilde{\nabla\text{-c}}$
2 foreach $1 \le i \le q, 1 \le j \le k_i - 1$ do
3     $\widetilde{\nabla\text{-c}}^i_j = (\nabla\text{-c}^i_j, \texttt{index}, \oplus\texttt{-state}, \texttt{image}) := \left(\nabla\text{-c}^i_j, (i,j), S^i_{j\oplus}, S^i_{j+1}\right)$
4 Sort $\widetilde{\nabla\text{-c}}$ primarily according to the third entry and secondarily according to the second
    entry (using the order defined in Equation (42)).
5 Output $\widetilde{\nabla\text{-c}}$
```

The main contribution of this subsection is Algorithm 4 which adds to each ∇-c information about the repetitions of the internal states. Running PREP groups the state values. The next step is to assign specific flags to states that are first (according to a specified rule) in each group. To each $S^i_{j\oplus}$ we will assign a flag, $\boxed{\text{u}}$ for unique states, $\boxed{\text{n}}$ for non-unique states, and $\boxed{\text{f}}$ for states that appear twice or more in total but from our perspective it is their first appearance. The output of Algorithm 4 is FLAG-ASSIGN(∇-c) $=$ ∇-cf ("nabla configuration with flags") and $\forall i,j : \nabla\text{-cf}^i_j = (\boxed{\text{F}}\,\nabla\text{-c}^i_j, S)$, where the first register is the whole state between evaluations together with the assigned flag of $\varphi_{\mathbf{f}}$ and S is the corresponding image. To refer to the l-th register of ∇-c^i_j we write ∇-c$^i_j(l)$. Flag $\boxed{\text{f}}$ is important when discussing the relative position of unique flags ($\boxed{\text{u}}$ or $\boxed{\text{f}}$) in the array of ∇-cf. In the end of this section and in the beginning of the next section we are not going to need this distinction but it will become important when analyzing the final probability expression.

Let us define a simple function acting on elements of arrays ∇-cf output by FLAG-ASSIGN. FLAG : $\{\boxed{\text{u}}, \boxed{\text{f}}, \boxed{\text{n}}\} \times \{0,1\}^{2r} \times \mathcal{C} \to \{\boxed{\text{u}}, \boxed{\text{f}}, \boxed{\text{n}}\}$,

$$\text{FLAG}(\nabla\text{-cf}^i_j) = \text{FLAG}(\begin{pmatrix} \bar{S}^i_j \, \boxed{\text{F}} \bar{S}^i_{j\oplus} \\ \boxed{\text{F}} \hat{S}^i_j \end{pmatrix}, S) := \boxed{\text{F}}. \tag{44}$$

Transition probabilities in Equation (43) depend on the flags we assigned to states in ∇-c. We

Algorithm 4: FLAG-ASSIGN

```
   Input  : $\nabla$-c for $(\mathbf{M}, \mathbf{Z})$
   Output: $\nabla$-cf
1  $\nabla$-cf $= \emptyset$
2  $\widetilde{\nabla\text{-c}} :=$ PREP($\nabla$-c)
3  Set counter $l := 1$
4  while $l \leq |\widetilde{\nabla\text{-c}}| = \kappa + q$ do
5      Set counter $i := 1$                 // the number of states with the same value
6      while $\widetilde{\nabla\text{-c}}_{l+i}(3) = \widetilde{\nabla\text{-c}}_l(3)$ do
7          $i := i + 1$
8      if $i = 1$ then
9          $\begin{pmatrix} \bar{S}\ \bar{S}_{\oplus} \\ \hat{S} \end{pmatrix} := \widetilde{\nabla\text{-c}}_l(1)$, append $\left(\begin{pmatrix} \bar{S}\ \boxed{\text{u}}\bar{S}_{\oplus} \\ \boxed{\text{u}}\hat{S} \end{pmatrix}, \widetilde{\nabla\text{-c}}_l(2), \widetilde{\nabla\text{-c}}_l(4)\right)$ to $\nabla$-cf          //
                               // (state with the same value and a flag, indices, image)
10         $(i', j') := \widetilde{\nabla\text{-c}}_l(2)$
11     if $i > 1$ then
12         $\begin{pmatrix} \bar{S}\ \bar{S}_{\oplus} \\ \hat{S} \end{pmatrix} := \widetilde{\nabla\text{-c}}_l(1)$, append $\left(\begin{pmatrix} \bar{S}\ \boxed{\text{f}}\bar{S}_{\oplus} \\ \boxed{\text{f}}\hat{S} \end{pmatrix}, \widetilde{\nabla\text{-c}}_l(2), \widetilde{\nabla\text{-c}}_l(4)\right)$ to $\nabla$-cf
13         for $j = 1, 2, \ldots, i-1$ do
14             $\begin{pmatrix} \bar{S}\ \bar{S}_{\oplus} \\ \hat{S} \end{pmatrix} := \widetilde{\nabla\text{-c}}_l(1)$, append $\left(\begin{pmatrix} \bar{S}\ \boxed{\text{n}}\bar{S}_{\oplus} \\ \boxed{\text{n}}\hat{S} \end{pmatrix}, \widetilde{\nabla\text{-c}}_l(2), \widetilde{\nabla\text{-c}}_l(4)\right)$ to $\nabla$-cf
15     $l := l + i$
16 Make a 2-dimensional array out of $\nabla$-cf according to the second entry in a standard
     left-to-right order $((i,j) \prec_{\text{l-r}} (i',j') \Leftrightarrow (i < i') \vee (i = i' \wedge j < j'))$, delete the second entry
     of $\nabla$-cf                              // $\nabla$-cf$^{i}_{j}$ =(state with a flag, image)
17 Output $\nabla$-cf
```

have that

$$\begin{aligned}\textsc{Flag}(\nabla\text{-cf}^i_j) \in \{\boxed{\mathrm{u}},\boxed{\mathrm{f}}\} \Rightarrow \mathbb{P}\left[\varphi_{\mathbf{f}}({}^{(\boxed{\mathrm{u}}\vee\boxed{\mathrm{f}})}S^i_{j\oplus}) = S \mid \bigwedge_{(i',j')\prec(i,j)} \left(\varphi_{\mathbf{f}}(S^{i'}_{j'\oplus}) = S^{i'}_{j'+1}\right)\right] &= \frac{1}{2^r\cdot|\mathcal{C}|},\\ \textsc{Flag}(\nabla\text{-cf}^i_j) = \boxed{\mathrm{n}} \Rightarrow \mathbb{P}\left[\varphi_{\mathbf{f}}({}^{\boxed{\mathrm{n}}}S^i_{j\oplus}) = S \mid \bigwedge_{(i',j')\prec(i,j)} \left(\varphi_{\mathbf{f}}(S^{i'}_{j'\oplus}) = S^{i'}_{j'+1}\right)\right] &= \begin{cases}1 & \text{if } S = \nabla\text{-cf}^i_j(2)\\ 0 & \text{otherwise}\end{cases}.\end{aligned} \tag{45}$$

6.3 Cardinality of ∇-C$(\mathbf{M},\mathbf{Z})$

In this section we evaluate the number of intermediate states that give $\left(\forall i\in[q] : \textsc{SpGen}_{\varphi_{\mathbf{f}}}(\mathbf{M}^i,\ell_i) = \mathbf{Z}^i\right)$. First we impose the constraint of $\varphi_{\mathbf{f}}$ being a function. Then we want to calculate the product of probabilities in Eq. (43). It depends on the number of unique states in ∇-c so we divide the set of possible states into subsets with the same number of states with the flag $\boxed{\mathrm{u}}$ or $\boxed{\mathrm{f}}$. The next steps involve further divisions of ∇-C$(\mathbf{M},\mathbf{Z})$.

In the process of calculating the conditional probabilities in Eq. (43) we included in each state in ∇-c the image it should have under $\varphi_{\mathbf{f}}$. The set ∇-C$(\mathbf{M},\mathbf{Z})$ does however contain states that would violate the constraint of $\varphi_{\mathbf{f}}$ being a function. The first step to calculate the cardinality of ∇-C$(\mathbf{M},\mathbf{Z})$ is to exclude ∇-c that do not fulfill this requirement. The set of states that should be taken into consideration is defined below, we denote this set by p-∇-CF$(\mathbf{M},\mathbf{Z})$ (p emphasizes the fact that $\varphi_{\mathbf{f}}$ is a proper function).

Definition 15 (p-∇-CF$(\mathbf{M},\mathbf{Z})$)**.** *The set of nabla configurations* ∇-c *for* $(\mathbf{M},\mathbf{Z})$ *with flags and a proper function* $\varphi_{\mathbf{f}}$ *is a set of arrays of size specified by* $(\mathbf{M},\mathbf{Z})$. p-∇-CF$(\mathbf{M},\mathbf{Z}) \subset \left(\left(\{\boxed{\mathrm{u}},\boxed{\mathrm{f}},\boxed{\mathrm{n}}\}\times\{0,1\}^{2r}\times\mathcal{C}\right)\times\left(\{0,1\}^r\times\mathcal{C}\right)\right)^{\kappa+q}$*, the set is defined in two steps, first we define the set of* ∇-cf *that are output by* $\textsc{Flag-Assign}$*,*

$$\nabla\text{-CF}(\mathbf{M},\mathbf{Z}) := \{\nabla\text{-c} : \exists\,\nabla\text{-c}_0 \in \nabla\text{-C}(\mathbf{M},\mathbf{Z}), \nabla\text{-c} = \textsc{Flag-Assign}(\nabla\text{-c}_0)\}\,. \tag{46}$$

We define p-∇-CF$(\mathbf{M},\mathbf{Z})$ *by the following constraints on* ∇-CF$(\mathbf{M},\mathbf{Z})$*:*

$$\forall S^i_j \in \nabla\text{-cf}\;\forall j>1 : S^i_j = \nabla\text{-cf}^i_{j-1}(2). \tag{47}$$

The formal definition reads

$$\text{p-}\nabla\text{-CF}(\mathbf{M},\mathbf{Z}) := \{\nabla\text{-cf} \in \nabla\text{-CF}(\mathbf{M},\mathbf{Z}) : \nabla\text{-cf fulfills constraints (47)}\}\,. \tag{48}$$

One may think about p-∇-CF$(\mathbf{M},\mathbf{Z})$ as follows, first we consider ∇-c: an array of states. The collection of all those arrays—with the exception of those that do not fulfill constraints (37)—is denoted by ∇-C$(\mathbf{M},\mathbf{Z})$. On each ∇-c $\in\nabla$-C$(\mathbf{M},\mathbf{Z})$ we run the algorithm $\textsc{Flag-Assign}$, getting a collection of ∇-cf—denoted by ∇-CF$(\mathbf{M},\mathbf{Z})$. Now we discard all those ∇-cf that do no fulfill constraints (47). The collection we are left with is denoted by p-∇-CF$(\mathbf{M},\mathbf{Z})$. We have the following relations between sets:

$$\nabla\text{-CF}(\mathbf{M},\mathbf{Z})(1) \overset{\text{omitting the flags}}{\simeq} \nabla\text{-C}(\mathbf{M},\mathbf{Z}) \tag{49}$$
$$\text{p-}\nabla\text{-CF}(\mathbf{M},\mathbf{Z}) \subset \nabla\text{-CF}(\mathbf{M},\mathbf{Z})\,. \tag{50}$$

Each p-∇-cf $\in$ p-∇-CF$(\mathbf{M},\mathbf{Z})$ has some number of unique states: with flag $\boxed{\mathrm{u}}$ or $\boxed{\mathrm{f}}$. Let us denote this number by $\bar{u}$. Eq. (45) implies that no matter in what configurations the unique states are, the product of probabilities in Eq. (43) is the same. Hence the first division of

$\mathsf{p}\text{-}\nabla\text{-}\mathsf{CF}(\mathbf{M},\mathbf{Z})$ is in terms of the total number of unique states. We denote the state with a fixed number $\bar{u}$ by $\mathsf{p}\text{-}\nabla\text{-}\mathsf{CF}(\mathbf{M},\mathbf{Z},\bar{u})$, we have that

$$\mathsf{p}\text{-}\nabla\text{-}\mathsf{CF}(\mathbf{M},\mathbf{Z}) = \bigcup_{\bar{u}=1}^{\kappa} \mathsf{p}\text{-}\nabla\text{-}\mathsf{CF}(\mathbf{M},\mathbf{Z},\bar{u}). \tag{51}$$

The product in Eq. (43) for $\mathsf{p}\text{-}\nabla\text{-}\mathsf{cf} \in \mathsf{p}\text{-}\nabla\text{-}\mathsf{CF}(\mathbf{M},\mathbf{Z},\bar{u})$ evaluates to

$$\prod_{(i,j)=(1,1)}^{(q,k_q-1)} \mathbb{P}\left[\left(\varphi_{\mathbf{f}}(S^i_{j\oplus}) = S^i_{j+1}\right) \mid \bigwedge_{(i',j')\prec(i,j)} \left(\varphi_{\mathbf{f}}(S^{i'}_{j'\oplus}) = S^{i'}_{j'+1}\right)\right] = \left(\frac{1}{2^r\cdot|\mathcal{C}|}\right)^{\bar{u}}, \tag{52}$$

where all states $\mathsf{p}\text{-}\nabla\text{-}\mathsf{cf}$ are in $\mathsf{p}\text{-}\nabla\text{-}\mathsf{CF}(\mathbf{M},\mathbf{Z},\bar{u})$.

We have to work a bit more to calculate the total number of states. The number of possibilities in which a single transition event can be realized depends both on the input and the output. For that reason we need to specify the configuration of flags in more detail, not just by the total number of unique states. Let us denote a transition event from a unique state to a unique state by $\left(\varphi_{\mathbf{f}}({}^{(\boxed{\mathrm{u}}\vee\boxed{\mathrm{f}})}S_\oplus) = {}^{(\boxed{\mathrm{u}}\vee\boxed{\mathrm{f}})}S\right)$ and similarly for other flags. The flag of the output is defined by the XORed message block or the output block. Before we go into details of the analysis of the structure of $\mathsf{p}\text{-}\nabla\text{-}\mathsf{CF}(\mathbf{M},\mathbf{Z})$, we list the intuitive principles of counting the output states depending on the input and output states:

1. $\left(\varphi_{\mathbf{f}}({}^{(\boxed{\mathrm{u}}\vee\boxed{\mathrm{f}})}S_\oplus) = {}^{(\boxed{\mathrm{u}}\vee\boxed{\mathrm{f}})}S\right)$—the only constraint is that the output cannot be the same as any on the previous unique states, the number of possible output values is at most $2^r\cdot|\mathcal{C}|$ or $|\mathcal{C}|$ and can be smaller by at most κ (the bound is $2^r\cdot|\mathcal{C}|$ if the transition is in the absorbing phase and $|\mathcal{C}|$ if it is in the squeezing phase),

2. $\left(\varphi_{\mathbf{f}}({}^{(\boxed{\mathrm{u}}\vee\boxed{\mathrm{f}})}S_\oplus) = {}^{\boxed{\mathrm{n}}}S\right)$—the output has to be in the set of outputs of states with the flag $\boxed{\mathrm{f}}$, the number of possible output values is at most κ,

3. $\left(\varphi_{\mathbf{f}}({}^{\boxed{\mathrm{n}}}S_\oplus) = {}^{(\boxed{\mathrm{u}}\vee\boxed{\mathrm{f}}\vee\boxed{\mathrm{n}})}S\right)$—the output is defined by the image memorized in the second entry of the state, the number of possible output values $= 1$.

The actual numbers in the above guidelines can be calculated precisely but they depend on the actual case we deal with.

To properly treat the transition events we need to keep track of not only the total number of unique states but also the number of truly unique $\boxed{\mathrm{u}}$ states. We denote the latter by u and the set with those numbers fixed by $\mathsf{p}\text{-}\nabla\text{-}\mathsf{CF}(\mathbf{M},\mathbf{Z},\bar{u},u)$. In the above paragraph we also noticed that we should include in our considerations the number of unique states in different phases of SpGen. The number of states with the flag $\boxed{\mathrm{u}}$ in the absorbing phase is denoted by u_{abs}. Note that we are addressing all q absorbing phases so we take into account flags of all states with indices $(i,j) \in \{(i',j')\}_{i'\in\{1,\dots,q\},j'\in\{1,\dots,|\mathbf{M}^{i'}|_r\}}$. The number of states with the flag $\boxed{\mathrm{u}}$ in the squeezing phase is denoted by u_{squ} and we take into account states with indices $(i,j) \in \{(i',j')\}_{i'\in\{1,\dots,q\},j'\in\{|\mathbf{M}^{i'}|_r+1,\dots,k_{i'}-1\}}$. Similarly the total number of unique states is denoted by $\bar{u}_{\mathrm{abs}}$ and $\bar{u}_{\mathrm{squ}}$.

Next we fix particular placements of flags in the arrays $\mathsf{p}\text{-}\nabla\text{-}\mathsf{cf} \in \mathsf{p}\text{-}\nabla\text{-}\mathsf{CF}(\mathbf{M},\mathbf{Z},\bar{u}_{\mathrm{abs}},u_{\mathrm{abs}},\bar{u}_{\mathrm{squ}},u_{\mathrm{squ}})$. We no longer need to keep u and $\bar{u}$ explicit as $u = u_{\mathrm{abs}}+u_{\mathrm{squ}}$ and $\bar{u} = \bar{u}_{\mathrm{abs}} + \bar{u}_{\mathrm{squ}}$. Let us define a *placement* P for $(\mathbf{M},\mathbf{Z})$ as an array of flags $\boxed{\mathrm{F}} \in \{\boxed{\mathrm{u}},\boxed{\mathrm{f}},\boxed{\mathrm{n}}\}$ with its dimensions determined by $(\mathbf{M},\mathbf{Z})$ in the same way as for nabla configurations $\nabla\text{-}\mathsf{c}$. The set of placements $\mathcal{P}(\mathbf{M},\mathbf{Z},\bar{u}_{\mathrm{abs}},u_{\mathrm{abs}},\bar{u}_{\mathrm{squ}},u_{\mathrm{squ}})$ is defined as the set of all placements P encountered in elements of $\mathsf{p}\text{-}\nabla\text{-}\mathsf{CF}(\mathbf{M},\mathbf{Z},\bar{u}_{\mathrm{abs}},u_{\mathrm{abs}},\bar{u}_{\mathrm{squ}},u_{\mathrm{squ}})$. We are going to write $\textsc{Flag}(P^i_j)$ to determine the flag in the position (i,j) in placement P. For each P we are able to calculate the size of $\mathsf{p}\text{-}\nabla\text{-}\mathsf{CF}(\mathbf{M},\mathbf{Z},P)$, we no longer add $\bar{u}_{\mathrm{abs}}$ and other parameters as they are already included in P. Before we define the algorithm performing this calculation we need to bound the number of different placements.

Let us assume for a moment that $(\mathbf{M}, \mathbf{Z})$ restrains only the size of $\mathsf{p}\text{-}\nabla\text{-}\mathsf{cf}$ and not the values of the states. If there were no constraints coming from the workings of FLAG-ASSIGN then unique states would be distributed in all combinations of picking $\bar{u}_{\text{abs}}$ elements among states in absorbing phases. Additionally, we also want to take into account combinations of u_{abs} elements among the $\bar{u}_{\text{abs}}$ flags. Let us recapitulate: first we distribute $\bar{u}_{\text{abs}}$ flags (without specifying whether they are u or f) and then assign them concrete values (u or f). The total number of state-triples in the absorbing phases of $\mathsf{p}\text{-}\nabla\text{-}\mathsf{cf}$ is $\mu := \sum_{i=1}^{q} |\mathbf{M}^i|_r$. The number of possibilities for the first step is $\binom{\mu}{\bar{u}_{\text{abs}}}$ and the second step is $\binom{\bar{u}_{\text{abs}}}{u_{\text{abs}}}$. The total number of possibilities of placing the unique flags in absorbing phases is $\binom{\mu}{\bar{u}_{\text{abs}}} \cdot \binom{\bar{u}_{\text{abs}}}{u_{\text{abs}}}$.

The problem of distributing unique states in squeezing phases is the same as in absorbing phases. The total number of state-triples with flags in the squeezing phases of $\mathsf{p}\text{-}\nabla\text{-}\mathsf{cf}$ is $\zeta := \sum_{i=1}^{q} (|\mathbf{Z}^i|_r - 1)$. The number of placements is $\binom{\zeta}{\bar{u}_{\text{squ}}}$. We also need to multiply this result by the number of placements of states with flag u among all unique states.

The two calculations above bring us to the conclusion that our analysis is sufficiently detailed; we have identified and taken into account all parts of $\left(\forall i \in [q] : \textsc{SpGen}_{\boldsymbol{\varphi}_{\mathbf{f}}}(\mathbf{M}^i, \ell_i) = \mathbf{Z}^i\right)$ that depend on $|\mathcal{C}|$. In summary we divided $\mathsf{p}\text{-}\nabla\text{-}\mathsf{CF}(\mathbf{M}, \mathbf{Z})$ into a small (relatively to $|\mathcal{C}|$) number of subsets whose size we can actually calculate. The last result assures that even though we do not formally describe the structure of the last level of division of $\mathsf{p}\text{-}\nabla\text{-}\mathsf{CF}(\mathbf{M}, \mathbf{Z})$, the number of possibilities of next divisions does not depend on $|\mathcal{C}|$. So we have that

$$|\mathcal{P}(\mathbf{M}, \mathbf{Z}, \bar{u}_{\text{abs}}, u_{\text{abs}}, \bar{u}_{\text{squ}}, u_{\text{squ}})| \le \binom{\mu}{\bar{u}_{\text{abs}}} \binom{\bar{u}_{\text{abs}}}{u_{\text{abs}}} \cdot \binom{\zeta}{\bar{u}_{\text{squ}}} \binom{\bar{u}_{\text{squ}}}{u_{\text{squ}}} \tag{53}$$

$$\le \binom{\mu}{\mu/2}^2 \binom{\zeta}{\zeta/2}^2 \le \binom{\kappa}{\kappa/2}^4 \le \kappa^{4\kappa}. \tag{54}$$

Our assumption is that κ is fixed so the number of placements is independent of $|\mathcal{C}|$. Note that we can compute $|\mathcal{P}(\mathbf{M}, \mathbf{Z}, \bar{u}_{\text{abs}}, u_{\text{abs}}, \bar{u}_{\text{squ}}, u_{\text{squ}})|$ for fixed parameters and the above inequality just shows that irrespective of the exact value of the calculation the number of placements does not depend on $|\mathcal{C}|$ and is relatively small.

Let us define a function that helps us accommodate for the fact that some subsets of $\mathsf{p}\text{-}\nabla\text{-}\mathsf{CF}(\mathbf{M}, \mathbf{Z})$ are empty for some specific $(\mathbf{M}, \mathbf{Z})$:

$$\boldsymbol{\delta}(\mathbf{M}, \mathbf{Z}, P) := \begin{cases} 1 \text{ if } \mathsf{p}\text{-}\nabla\text{-}\mathsf{CF}(\mathbf{M}, \mathbf{Z}, P) \neq \emptyset \\ 0 \text{ otherwise} \end{cases}. \tag{55}$$

In what follows we leave out the input to $\boldsymbol{\delta}$, as it can be inferred from context. For example $\boldsymbol{\delta}$ evaluates to 0 if the input includes $\bar{u}_{\text{abs}} = \mu$ and the first block of the input messages is not always different.

The last division we make is done be characterizing uniqueness of outer and inner parts of states. This step is done to get the precise and correct result, but the high level explanation and an approximation of the output of CALC is already captured in 1. We have not captured this situation in detail in our example proof because it becomes important only if longer outputs are present. Here we explain the procedure of including the necessary details.

Main detail we add is assigning flags to outer and inner parts of states individually. We introduce those flags only now to keep the proof as clear as possible; technically to include the additional flags we modify the algorithm FLAG-ASSIGN in such a way that it runs over a configuration $\nabla\text{-}\mathsf{c}$ two additional times but acting solely on outer states and inner states. Those two additional runs assign the same flags as the original one but corresponding to just one of the parts of $S_\oplus$ states. Rest of the discussion after applying Flag-Assign is unchanged and depends only on flags of the full states.

When discussing placements note that a unique state ($\boxed{\mathrm{u}}$ or $\boxed{\mathrm{f}}$) can consist of a unique outer state and a unique inner state but also out of a non-unique outer state and a unique inner state or vice versa. After we assign a particular placement $P \in \mathcal{P}(\mathbf{M}, \mathbf{Z}, \bar{u}_{\mathrm{abs}}, u_{\mathrm{abs}}, \bar{u}_{\mathrm{squ}}, u_{\mathrm{squ}})$ there are still many possibilities of arranging outer and inner states flags. There are exactly three possibilities every unique state can be arranged in: $\begin{pmatrix} \boxed{\mathrm{u}} \vee \boxed{\mathrm{f}} \\ \boxed{\mathrm{u}} \vee \boxed{\mathrm{f}} \end{pmatrix}$, $\begin{pmatrix} \boxed{\mathrm{u}} \vee \boxed{\mathrm{f}} \\ \boxed{\mathrm{n}} \end{pmatrix}$, and $\begin{pmatrix} \boxed{\mathrm{n}} \\ \boxed{\mathrm{u}} \vee \boxed{\mathrm{f}} \end{pmatrix}$, where we symbolize a state $S_\oplus$ by a column vector with flags assigned to its outer state in the first row and inner state in the second row. Hence, for every placement P we have $3^{\bar{u}_{\mathrm{abs}}+\bar{u}_{\mathrm{squ}}}$ placements of the outer and inner states flags. We are going to mark the fact that we have included those additional details into placements by adding a star to the set of placements $P \in \mathcal{P}^*(\mathbf{M}, \mathbf{Z}, \bar{u}_{\mathrm{abs}}, u_{\mathrm{abs}}, \bar{u}_{\mathrm{squ}}, u_{\mathrm{squ}})$. We have that

$$|\mathcal{P}^*(\mathbf{M}, \mathbf{Z}, \bar{u}_{\mathrm{abs}}, u_{\mathrm{abs}}, \bar{u}_{\mathrm{squ}}, u_{\mathrm{squ}})| \le \kappa^{4\kappa} \cdot 3^{\bar{u}_{\mathrm{abs}}+\bar{u}_{\mathrm{squ}}}. \tag{56}$$

We also write $\textsc{Flag}(\bar{P}^i_j)$ and $\textsc{Flag}(\hat{P}^i_j)$ to access the flag of the outer and inner part of P^i_j respectively.

Alg. 5 below shows the algorithm Calc that outputs the number of different $\mathsf{p}\text{-}\nabla\text{-}\mathsf{cf} \in \mathsf{p}\text{-}\nabla\text{-}\mathsf{CF}(\mathbf{M}, \mathbf{Z}, \bar{u}_{\mathrm{abs}}, u_{\mathrm{abs}}, \bar{u}_{\mathrm{squ}}, u_{\mathrm{squ}})$ for some given placement $P \in \mathcal{P}^*(\mathbf{M}, \mathbf{Z}, \bar{u}_{\mathrm{abs}}, u_{\mathrm{abs}}, \bar{u}_{\mathrm{squ}}, u_{\mathrm{squ}})$. To capture the fact that the number of possible values a unique state can have depends on the number of unique states with already assigned values we define the following sets. For unique outer states we have

$$\bar{\mathrm{U}}_{\mathrm{prev}}(P,i,j) := \left| \left\{ P^{i'}_{j'} : (i',j') \prec (i,j) \wedge \textsc{Flag}(\bar{P}^{i'}_{j'}) \in \{\boxed{\mathrm{u}}, \boxed{\mathrm{f}}\} \right\} \right|, \tag{57}$$

$$\bar{\mathrm{U}}^{\boxed{\mathrm{f}}}_{\mathrm{prev}}(P,i,j) := \left| \left\{ P^{i'}_{j'} : (i',j') \prec (i,j) \wedge \textsc{Flag}(\bar{P}^{i'}_{j'}) = \boxed{\mathrm{f}} \right\} \right|. \tag{58}$$

For unique inner states we have

$$\hat{\mathrm{U}}_{\mathrm{prev}}(P,i,j) := \left| \left\{ P^{i'}_{j'} : (i',j') \prec (i,j) \wedge \textsc{Flag}(\hat{P}^{i'}_{j'}) \in \{\boxed{\mathrm{u}}, \boxed{\mathrm{f}}\} \right\} \right|, \tag{59}$$

$$\hat{\mathrm{U}}^{\boxed{\mathrm{f}}}_{\mathrm{prev}}(P,i,j) := \left| \left\{ P^{i'}_{j'} : (i',j') \prec (i,j) \wedge \textsc{Flag}(\hat{P}^{i'}_{j'}) = \boxed{\mathrm{f}} \right\} \right|. \tag{60}$$

Note that all of the above quantities (57, 58, 59, 60) are bounded by

$$1 \le \bar{\mathrm{U}}_{\mathrm{prev}}(P,i,j), \hat{\mathrm{U}}_{\mathrm{prev}}(P,i,j), \bar{\mathrm{U}}^{\boxed{\mathrm{f}}}_{\mathrm{prev}}(P,i,j), \hat{\mathrm{U}}^{\boxed{\mathrm{f}}}_{\mathrm{prev}}(P,i,j) \le \bar{u}_{\mathrm{abs}} + \bar{u}_{\mathrm{squ}} \le \kappa. \tag{61}$$

In the algorithm we also use N-Possibilities, defined in Eq. (87), is the number of possibilities in which one can assign values to non-unique states in a nabla configuration. N-Possibilities is bounded by κ^κ, which is an upper bound for Eq. (88).

Thanks to the additional details we get the precise form of the expression $\mathbf{p}$ but note that when we sum over placements of outer and inner states flags we have that same $\boldsymbol{\delta}(\mathbf{M}, \mathbf{Z}, P)$ for all cases in the absorbing phases, so we sum the expressions listed in Calc and get the same result as in Section 5. In the squeezing phases the outer states are fixed by the outputs $\mathbf{Z}$ and we can do the sum over placements with the same flag of the outer state in Calc.

6.4 Final expression

In the previous subsections we formalized algorithms that help us analyze the expression in Eq. (43). First we introduced Flag-Assign that analyzes $\nabla\text{-}\mathsf{c}$ from the perspective of having the same input to $\boldsymbol{\varphi}_{\mathbf{f}}$ multiple times. Then we defined Calc that counts the arrays of states that fulfill a given set of constraints, the number and arrangement of unique states. The final

Algorithm 5: CALC

```
Input  : $P \in \mathcal{P}^*(\mathbf{M}, \mathbf{Z}, \bar{u}_{\text{abs}}, u_{\text{abs}}, \bar{u}_{\text{squ}}, u_{\text{squ}})$
Output: $\alpha \in \mathbb{N}$, cardinality of the set $\mathsf{p}\text{-}\nabla\text{-}\mathsf{CF}(\mathbf{M}, \mathbf{Z}, P)$
$\alpha := 1$
for $j = 1, \dots, k_i - 2,\ i = 1, \dots, q$ do
    if $j < |\mathbf{M}^i|_r$ and FLAG$(P^i_j) \in \{\boxed{u}, \boxed{f}\}$ then                // Absorbing phases
        if FLAG$(P^i_{j+1}) \in \{\boxed{u}, \boxed{f}\}$ then                // $\left(\varphi_{\mathrm{f}}({}^{(\boxed{u}\vee\boxed{f})}S_\oplus) = {}^{(\boxed{u}\vee\boxed{f})}S\right)$
            if FLAG$(\bar{P}^i_{j+1}) \in \{\boxed{u}, \boxed{f}\}$ and FLAG$(\hat{P}^i_{j+1}) \in \{\boxed{u}, \boxed{f}\}$ then  // $P^i_{j+1} = \begin{pmatrix} \boxed{u}\vee\boxed{f} \\ \boxed{u}\vee\boxed{f} \end{pmatrix}$
                $\alpha = \alpha \cdot \left(2^r - \bar{\mathrm{U}}_{\text{prev}}(P, i, j+1)\right) \cdot \left(|\mathcal{C}| - \hat{\mathrm{U}}_{\text{prev}}(P, i, j+1)\right)$
            if FLAG$(\bar{P}^i_{j+1}) \in \{\boxed{u}, \boxed{f}\}$ and FLAG$(\hat{P}^i_{j+1}) = \boxed{n}$ then       // $P^i_{j+1} = \begin{pmatrix} \boxed{u}\vee\boxed{f} \\ \boxed{n} \end{pmatrix}$
                $\alpha = \alpha \cdot \left(2^r - \bar{\mathrm{U}}_{\text{prev}}(P, i, j+1)\right) \cdot \hat{\mathrm{U}}^{\boxed{f}}_{\text{prev}}(P, i, j+1)$
            if FLAG$(\bar{P}^i_{j+1}) = \boxed{n}$ and FLAG$(\hat{P}^i_{j+1}) \in \{\boxed{u}, \boxed{f}\}$ then       // $P^i_{j+1} = \begin{pmatrix} \boxed{n} \\ \boxed{u}\vee\boxed{f} \end{pmatrix}$
                $\alpha = \alpha \cdot \bar{\mathrm{U}}^{\boxed{f}}_{\text{prev}}(P, i, j+1) \cdot \left(|\mathcal{C}| - \hat{\mathrm{U}}_{\text{prev}}(P, i, j+1)\right)$
    if $j \geq |\mathbf{M}^i|_r$ and FLAG$(P^i_j) \in \{\boxed{u}, \boxed{f}\}$ then                // Squeezing phases
        if FLAG$(P^i_{j+1}) \in \{\boxed{u}, \boxed{f}\}$ then                // $\left(\varphi_{\mathrm{f}}({}^{(\boxed{u}\vee\boxed{f})}S_\oplus) = {}^{(\boxed{u}\vee\boxed{f})}S\right)$
            if FLAG$(\hat{P}^i_{j+1}) \in \{\boxed{u}, \boxed{f}\}$ then          // $P^i_{j+1} \in \left\{\begin{pmatrix} \boxed{u}\vee\boxed{f} \\ \boxed{u}\vee\boxed{f} \end{pmatrix}, \begin{pmatrix} \boxed{n} \\ \boxed{u}\vee\boxed{f} \end{pmatrix}\right\}$
                $\alpha = \alpha \cdot \left(|\mathcal{C}| - \hat{\mathrm{U}}_{\text{prev}}(P, i, j+1)\right)$
            if FLAG$(\hat{P}^i_{j+1}) = \boxed{n}$ then                       // $P^i_{j+1} = \begin{pmatrix} \boxed{u}\vee\boxed{f} \\ \boxed{n} \end{pmatrix}$
                $\alpha = \alpha \cdot \hat{\mathrm{U}}^{\boxed{f}}_{\text{prev}}(P, i, j+1)$
for $i = 1, \dots, q,\ j = k_i - 1$ do
    if FLAG$(P^i_j) \in \{\boxed{u}, \boxed{f}\}$ then
        $\alpha = \alpha \cdot |\mathcal{C}| \cdot 2^{r|\mathbf{Z}^i|_r - \ell_i}$
$\alpha = \alpha \cdot$ N-POSSIBILITIES$(\kappa - \bar{u}_{\text{abs}} - \bar{u}_{\text{squ}}, \bar{u}_{\text{abs}} + \bar{u}_{\text{squ}} - u_{\text{abs}} - u_{\text{squ}}, P)$
Output $\alpha \cdot \boldsymbol{\delta}(\mathbf{M}, \mathbf{Z}, P)$
```

part of the proof of Lemma 9 is to use those algorithms to show that $\mathbf{p}(|\mathcal{C}|^{-1})$ is of the claimed form. We start by formally writing down the expression in terms of divisions of $\textsf{p-}\nabla\textsf{-CF}(\mathbf{M},\mathbf{Z})$ we introduced and the outputs of Calc. Next we identify crucial elements of the sum that lead to the claim of the lemma, showing the maximal degree of $|\mathcal{C}|^{-1}$ in the expression $\mathbf{p}(\lambda)$.

In the previous sections we showed that

$$\mathbf{p}(|\mathcal{C}|^{-1}) = \sum_{\nabla\textsf{-c}\in\nabla\textsf{-C}(\mathbf{M},\mathbf{Z})} \prod_{(i,j)=(1,1)}^{(q,k_q-1)} \mathbb{P}\left[\left(\boldsymbol{\varphi}_{\mathbf{f}}(S^i_{j\oplus}) = S^i_{j+1}\right) \middle| \bigwedge_{(i',j')\prec(i,j)} \left(\boldsymbol{\varphi}_{\mathbf{f}}(S^{i'}_{j'\oplus}) = S^{i'}_{j'+1}\right)\right] \tag{62}$$

$$= \underbrace{\sum_{\textsf{p-}\nabla\textsf{-cf}\in\textsf{p-}\nabla\textsf{-CF}(\mathbf{M},\mathbf{Z})}}_{\text{Eq. (64),(65)}} \underbrace{\prod_{(i,j)=(1,1)}^{(q,k_q-1)} \mathbb{P}\left[\left(\boldsymbol{\varphi}_{\mathbf{f}}(S^i_{j\oplus}) = S^i_{j+1}\right) \middle| \bigwedge_{(i',j')\prec(i,j)} \left(\boldsymbol{\varphi}_{\mathbf{f}}(S^{i'}_{j'\oplus}) = S^{i'}_{j'+1}\right)\right]}_{\text{Eq. (52)}}, \tag{63}$$

where the second equality comes from the fact that constraints (47) exclude those $\nabla\textsf{-c}$ that have probability 0. Let us also make the division of $\textsf{p-}\nabla\textsf{-CF}(\mathbf{M},\mathbf{Z})$ explicit

$$\textsf{p-}\nabla\textsf{-CF}(\mathbf{M},\mathbf{Z}) = \bigcup_{\bar{u}_{\text{abs}}=1}^{\mu} \bigcup_{u_{\text{abs}}=0}^{\mu} \bigcup_{\bar{u}_{\text{squ}}=0}^{\zeta} \bigcup_{u_{\text{squ}}=0}^{\zeta} \bigcup_{P\in\mathcal{P}^*(\mathbf{M},\mathbf{Z},\bar{u}_{\text{abs}},u_{\text{abs}},\bar{u}_{\text{squ}},u_{\text{squ}})} \textsf{p-}\nabla\textsf{-CF}(\mathbf{M},\mathbf{Z},P). \tag{64}$$

Next we use Eq. (52) and the fact that for $P \in \mathcal{P}(\mathbf{M},\mathbf{Z},\bar{u}_{\text{abs}},u_{\text{abs}},\bar{u}_{\text{squ}},u_{\text{squ}})$ we have

$$|\textsf{p-}\nabla\textsf{-CF}(\mathbf{M},\mathbf{Z},P)| = \textsc{Calc}(P) \tag{65}$$

to expand $\mathbf{p}(|\mathcal{C}|^{-1})$ to

$$\mathbf{p}(|\mathcal{C}|^{-1}) = \sum_{\bar{u}_{\text{abs}},u_{\text{abs}},\bar{u}_{\text{squ}},u_{\text{squ}},P} \textsc{Calc}(P) \left(\frac{1}{2^r\cdot|\mathcal{C}|}\right)^{\bar{u}_{\text{abs}}+\bar{u}_{\text{squ}}} \tag{66}$$

To calculate a_0 and the maximal degree of $\mathbf{p}$ let us focus on $\mathbf{p}(|\mathcal{C}|^{-1})$ for all unique (with the flag u in both outer and inner part) sates:

$$\prod_{i=1}^{q}\prod_{j=1}^{|\mathbf{M}^i|_r-1} (2^r-jq-i)\,(|\mathcal{C}|-jq-i) \prod_{i=1}^{q}\prod_{j=|\mathbf{M}^i|_r}^{k_i-2} (|\mathcal{C}|-jq-i) \prod_{i=1}^{q}\left(2^{r|\mathbf{Z}^i|_r-\ell_i}|\mathcal{C}|\right)(2^r|\mathcal{C}|)^{-\kappa}. \tag{67}$$

In the above expression if we take all messages of maximal length m and outputs of maximal length z we get a polynomial of degree $\kappa - q = q(m+z-2)$. This is necessarily the maximal degree as every evaluation of $\boldsymbol{\varphi}_{\mathbf{f}}$ increases the degree by one, except for the last but this cannot be changed, the last column does not matter at all for the overall probability. Hence the maximal degree of $\mathbf{p}$ is as claimed

$$\eta := q(m+z-2). \tag{68}$$

In the case all states are unique, i.e. $|\mathcal{C}|\to\infty$, $\mathbf{p}(|\mathcal{C}|^{-1})$ evaluates to $\sim 2^{-\sum_i \ell_i}$. This expression corresponds to the output probability of a random oracle, exactly how expected of a sponge with all different inner states. If we only take the terms $2^r|\mathcal{C}|$ and $|\mathcal{C}|$ and the probability we arrive at $2^{-\sum_i \ell_i}$. This result is only one of the terms in a_0 but note that all other terms will correspond to different placements and will include $\boldsymbol{\delta}(\mathbf{M},\mathbf{Z},P)$ with different inputs, being non-zero for different $(\mathbf{M},\mathbf{Z})$. Hence for any given input-output pairs $(\mathbf{M},\mathbf{Z})$ for $|\mathcal{C}|\to\infty$ the probability function approaches the probability of a random oracle outputting $\mathbf{Z}$ on $\mathbf{M}$. To get the power of $|\mathcal{C}|$ equal to zero we need to have the same number of unique states (probability terms decreasing the degree by one) as pairs of unique states (increasing the degree by one). Configurations that satisfy those conditions come from inputs and outputs that are either fully unique or exactly the same as at least one other input or output, respectively. One special case occurs if the output is just a single block long then messages can differ by just the last block and still have different outputs (like in our example proof in Section 5).

In our proof we have focused on the case of $\varphi_{\mathbf{f}}$ being a random transformation. In Appendix 7 we provide the details that should be considered to show that Theorem 8 holds also for random permutations.

7 Internal Permutations

In this section we prove the main result but for the internal function $\varphi_{\mathbf{f}}$ being a random permutation. We use Zhandry's PRF/PRP switching lemma from [23]. In Appendix A, we also give a direct proof, resulting in a slightly worse bound.

Theorem 16. $\mathrm{SpGen}_{\varphi_{\mathbf{f}}}$ *for a random permutation* $\varphi_{\mathbf{f}}$ *is quantumly indistinguishable from a random oracle. More concretely, for all quantum algorithms* A *making at most* q *quantum queries to* SpGen*, such that the input length is at most* $m \cdot r$ *bits long and the output length is at most* $z \cdot r$ *bits long,*

$$\left| \underset{\varphi_{\mathbf{f}} \xleftarrow{\$} \mathcal{T}(\mathcal{S})}{\mathbb{P}} \left[\mathrm{A}^{|\mathrm{SpGen}_{\varphi_{\mathbf{f}}}\rangle}(.) = 1 \right] - \underset{\mathbf{h} \leftarrow \mathfrak{R}}{\mathbb{P}} \left[\mathrm{A}^{|\mathbf{h}\rangle}(.) = 1 \right] \right| < \frac{\pi^2}{3} \eta^3 |\mathcal{C}|^{-1}, \tag{69}$$

where the set of permutations is denoted by $\mathcal{T}(\mathcal{S}) := \{\varphi_{\mathbf{f}} : \mathcal{S} \to \mathcal{S} \mid \varphi_{\mathbf{f}} \text{ is a bijection}\}$*. The domain is defined as* $\mathcal{S} = \{0,1\}^r \times \mathcal{C}$ *for some non-empty finite set* $\mathcal{C}$*.*

Proof. It was proven in [23] that a random permutation can be distinguished from a random function with probability at most $\pi^2 q^2 / 6|\mathcal{C}|$ for any adversary making at most q quantum queries. We can use this result in a reduction from distinguishing SpGen using a random permutation from SpGen using a random function to distinguishing of a random permutation from a random function. Using this result together with Theorem 8 gives us the resulting bound as follows

$$\begin{aligned}
&\left| \underset{\varphi_{\mathbf{f}} \xleftarrow{\$} \mathcal{T}(\mathcal{S})}{\mathbb{P}} \left[\mathrm{A}^{|\mathrm{SpGen}_{\varphi_{\mathbf{f}}}\rangle}(.) = 1 \right] - \underset{\mathbf{h} \leftarrow \mathfrak{R}}{\mathbb{P}} \left[\mathrm{A}^{|\mathbf{h}\rangle}(.) = 1 \right] \right| \\
&\quad = \left| \underset{\varphi_{\mathbf{f}} \xleftarrow{\$} \mathcal{T}(\mathcal{S})}{\mathbb{P}} \left[\mathrm{A}^{|\mathrm{SpGen}_{\varphi_{\mathbf{f}}}\rangle}(.) = 1 \right] - \underset{\varphi_{\mathbf{f}} \xleftarrow{\$} \mathcal{S}^{\mathcal{S}}}{\mathbb{P}} \left[\mathrm{A}^{|\mathrm{SpGen}_{\varphi_{\mathbf{f}}}\rangle}(.) = 1 \right] \right. \\
&\qquad \left. + \underset{\varphi_{\mathbf{f}} \xleftarrow{\$} \mathcal{S}^{\mathcal{S}}}{\mathbb{P}} \left[\mathrm{A}^{|\mathrm{SpGen}_{\varphi_{\mathbf{f}}}\rangle}(.) = 1 \right] - \underset{\mathbf{h} \leftarrow \mathfrak{R}}{\mathbb{P}} \left[\mathrm{A}^{|\mathbf{h}\rangle}(.) = 1 \right] \right| \qquad (70) \\
&\quad \leq \left| \underset{\varphi_{\mathbf{f}} \xleftarrow{\$} \mathcal{T}(\mathcal{S})}{\mathbb{P}} \left[\mathrm{A}^{|\mathrm{SpGen}_{\varphi_{\mathbf{f}}}\rangle}(.) = 1 \right] - \underset{\phi \xleftarrow{\$} \mathcal{S}^{\mathcal{S}}}{\mathbb{P}} \left[\mathrm{A}^{|\mathrm{SpGen}_{\varphi_{\mathbf{f}}}\rangle}(.) = 1 \right] \right| \\
&\qquad + \left| \underset{\varphi_{\mathbf{f}} \xleftarrow{\$} \mathcal{S}^{\mathcal{S}}}{\mathbb{P}} \left[\mathrm{A}^{|\mathrm{SpGen}_{\varphi_{\mathbf{f}}}\rangle}(.) = 1 \right] - \underset{\mathbf{h} \leftarrow \mathfrak{R}}{\mathbb{P}} \left[\mathrm{A}^{|\mathbf{h}\rangle}(.) = 1 \right] \right| \qquad (71) \\
&\quad \leq \left| \underset{\varphi_{\mathbf{f}} \xleftarrow{\$} \mathcal{T}(\mathcal{S})}{\mathbb{P}} \left[\mathrm{B}^{|\varphi_{\mathbf{f}}\rangle}(.) = 1 \right] - \underset{\phi \xleftarrow{\$} \mathcal{S}^{\mathcal{S}}}{\mathbb{P}} \left[\mathrm{B}^{|\varphi_{\mathbf{f}}\rangle}(.) = 1 \right] \right| + \frac{\pi^2}{6} \eta^3 |\mathcal{C}|^{-1} \qquad (72) \\
&\quad \leq \frac{\pi^2}{3} \eta^3 |\mathcal{C}|^{-1}. \qquad (73)
\end{aligned}$$

□

8 Open Question

One of the most desirable security notions for hash functions is indifferentiability from a random oracle which is defined with respect to a possible simulator that fools a distinguisher into believing that it interacts with the internal function instead of a simulation of it. Proving indifferentiability is more challenging than indistinguishability. It is not clear whether the natural translation of the classical notion of indidfferentiability to the quantum setting is achievable. Only recently, two articles [9, 25] opened the discussion, but so far, the results remain inconclusive.

In our work, we provide a quantum security guarantee more suitable for keyed primitives where an attacker does not have access to the internal building block. On the one hand, we increase the trust that hash functions based on the sponge construction are quantum safe and on the other hand, we formally prove that it is a quantum secure pseudorandom function when used with a keyed internal function—like it is used in the hash-based signatures scheme SPHINCS+ [21] in the instantiation using the Haraka hash function [13].

Acknowledgments

The authors would like to thank Dominique Unruh and Leon Groot Bruiderink for helpful discussions. CS and JC are supported by a NWO VIDI grant (Project No. 639.022.519).

9 Symbol index

$\mathcal{Y}^{\mathcal{X}}$	The set of functions $\{\mathbf{f}:\mathcal{X}\to\mathcal{Y}\}$	5
$X \xleftarrow{\$} \mathcal{X}$	X chosen uniformly from set $\mathcal{X}$	5
$[q]$	The q-element set $[q] := \{1, 2, \ldots, q\}$	6
$\oplus$	Bitwise XOR	6
$X\|Y$	Concatenation of strings X and Y.	6
$\|X\|, \|\mathcal{C}\|$	length of a string X (cardinality of a set $\mathcal{C}$)	8
$\textsc{Sponge}_{c,r,\mathbf{f},\text{PAD},\ell}$	Sponge construction (capacity c, bit rate r, block function $\mathbf{f}$, output length ℓ)	7
$\textsc{SpGen}_{\mathcal{C},r,\varphi_{\mathbf{f}},\text{PAD},\ell}$	The generalized Sponge construction (capacity set $\mathcal{C}$, bit rate r, internal map $\varphi_{\mathbf{f}}$, output length ℓ)	9
$\varphi_{\mathbf{f}}$	The general map between states	8
$\bar{S}, \bar{\varphi}_{\mathbf{f}}$	Outer part of a state $S \in \{0,1\}^r \times \mathcal{C}$, $\bar{S} \in \{0,1\}^r$, function $\varphi_{\mathbf{f}} : \{0,1\}^r \times \mathcal{C} \to \{0,1\}^r \times \mathcal{C}$ with its output limited to the part in $\{0,1\}^r$.	8
$\hat{S}, \hat{\varphi}_{\mathbf{f}}$	Inner part of a state $S \in \{0,1\}^r \times \mathcal{C}$, $\hat{S} \in \mathcal{C}$, function $\varphi_{\mathbf{f}} : \{0,1\}^r \times \mathcal{C} \to \{0,1\}^r \times \mathcal{C}$ with its output limited to the part in $\mathcal{C}$.	8
$\mathbf{M}$	Array of messages, $\mathbf{M} = (\mathbf{M}^1, \mathbf{M}^2, \ldots, \mathbf{M}^q)$, where $\forall i, \mathbf{M}^i = M_1^i \| M_2^i \| \ldots \| M^i_{\|\mathbf{M}^i\|_r}$	12
$\|\mathbf{M}\|_r$	The number of r-bit blocks in $\mathbf{M}$, $\|\mathbf{M}\|_r := \left\lceil \frac{\|\mathbf{M}\|}{r} \right\rceil$.	6
$\mathbf{Z}$	Array of output strings	12
$\kappa := \sum_{i=1}^{q}(k_i - 1)$	The total number of evaluations of $\varphi_{\mathbf{f}}$ in q evaluations of $\textsc{SpGen}$ on $\mathbf{M}^i$ with outputs $\mathbf{Z}^i$.	18
$k_i := \|\mathbf{M}^i\|_r + \|\mathbf{Z}^i\|_r$	The number of internal states in evaluation of $\textsc{SpGen}(\mathbf{M}^i)$ outputting $\mathbf{Z}^i$.	18
$\nabla\text{-}\mathsf{C}(\mathbf{M}, \mathbf{Z})$	The set of all ∇-c constrained by $(\mathbf{M}, \mathbf{Z})$.	19

Flag-Assign	The algorithm for assigning $\boxed{\mathrm{u}}$, $\boxed{\mathrm{f}}$, and $\boxed{\mathrm{n}}$ flags in ∇-c.	22
$\textsc{Flag}(\nabla\text{-c}_j^i)$	The flag of ∇-c $\in$ p-∇-CF$(\mathbf{M},\mathbf{Z})$, $\textsc{Flag}(\nabla\text{-c}_j^i) \in \{\boxed{\mathrm{u}}, \boxed{\mathrm{f}}, \boxed{\mathrm{n}}\}$	21
∇-CF$(\mathbf{M},\mathbf{Z})$	The set of all ∇-c constrained by $(\mathbf{M},\mathbf{Z})$ with Flag-Assign run on it.	23
p-∇-cf	A particular assignment of states in Sponge evaluations of $(\mathbf{M},\mathbf{Z})$ with additional constraints for $\varphi_{\mathbf{f}}$ being a valid function.	23
p-∇-CF$(\mathbf{M},\mathbf{Z})$	The set of all p-∇-cf constrained by $(\mathbf{M},\mathbf{Z})$ and $\varphi_{\mathbf{f}}$.	23
$\mathcal{P}$	The set representing a particular placement of unique and non-unique flags in p-∇-cf	24
p-∇-CF$(\mathbf{M},\mathbf{Z},*)$	p-∇-CF$(\mathbf{M},\mathbf{Z})$ with specified parameters $*$.	24
Calc	The algorithm that outputs the cardinality of p-∇-CF$(\mathbf{M},\mathbf{Z},P)$.	27
$\mu := \sum_{i=1}^{q} \lvert\mathbf{M}^i\rvert_r$	The number of states with the next step being in the squeezing phase of SpGen.	25
$\zeta := \sum_{i=1}^{q} \left(\lvert\mathbf{Z}^i\rvert_r - 1\right)$	The number of states with the next state being in the absorbing phase, $\kappa = \mu + \zeta$.	25
$\mathcal{T}(\mathcal{S})$	The set of permutations from $\mathcal{S}$ to $\mathcal{S}$, $\{\varphi_{\mathbf{f}} : \mathcal{S} \to \mathcal{S} \mid \varphi_{\mathbf{f}}$ is a bijection$\}$	29

References

[1] Elena Andreeva, Joan Daemen, Bart Mennink, and Gilles Van Assche. "Security of keyed sponge constructions using a modular proof approach". In: *International Workshop on Fast Software Encryption*. Springer. 2015, pp. 364–384.

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

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

[4] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. "On the security of the keyed sponge construction". In: *Symmetric Key Encryption Workshop*. Vol. 2011. 2011.

[5] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "Sponge-based pseudo-random number generators". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2010, pp. 33–47.

[6] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications." In: *Selected Areas in Cryptography*. Vol. 7118. Springer. 2011, pp. 320–337.

[7] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "Permutation-based encryption, authentication and authenticated encryption". In: *Directions in Authenticated Ciphers* (2012).

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

[9] Tore Vincent Carstens, Ehsan Ebrahimi, Gelo Noel Tabia, and Dominique Unruh. *On Quantum Indifferentiability.* Tech. rep. Cryptology ePrint Archive, Report 2018/257, 2018. https://eprint.iacr.org/2018/257, 2018.

[10] Dong H. Chang, Morris J. Dworkin, Seokhie Hong, John M. Kelsey, and Mridul Nandi. *A Keyed Sponge Construction with Pseudorandomness in the Standard Model.* NIST. The Third SHA-3 Candidate Conference. 2012.

[11] Peter Gaži, Krzysztof Pietrzak, and Stefano Tessaro. "The exact PRF security of truncation: Tight bounds for keyed sponges and truncated CBC". In: *Annual Cryptology Conference.* Springer. 2015, pp. 368–387.

[12] Marc Kaplan, Gaëtan Leurent, Anthony Leverrier, and María Naya-Plasencia. "Breaking symmetric cryptosystems using quantum period finding". In: *Annual Cryptology Conference.* Springer. 2016, pp. 207–237.

[13] Stefan Kölbl, Martin M Lauridsen, Florian Mendel, and Christian Rechberger. "Haraka v2–efficient short-input hashing for post-quantum applications". In: *IACR Transactions on Symmetric Cryptology* (2016), pp. 1–29.

[14] Hidenori Kuwakado and Masakatu Morii. "Security on the quantum-type Even-Mansour cipher". In: *Information Theory and its Applications (ISITA), 2012 International Symposium on.* IEEE. 2012, pp. 312–316.

[15] Bart Mennink, Reza Reyhanitabar, and Damian Vizár. "Security of full-state keyed sponge and duplex: Applications to authenticated encryption". In: *International Conference on the Theory and Application of Cryptology and Information Security.* Springer. 2014, pp. 465–489.

[16] Bart Mennink and Alan Szepieniec. "XOR of PRPs in a Quantum World". In: *International Workshop on Post-Quantum Cryptography.* Springer. 2017, pp. 367–383.

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

[18] RL Rivest and JCN Schuldt. "Spritz–a spongy RC4-like stream cipher and hash function (2014)". In: *Charles River Crypto Day (2014-10-24)* ().

[19] Thomas Santoli and Christian Schaffner. "Using Simon's algorithm to attack symmetric-key cryptographic primitives". In: *arXiv preprint arXiv:1603.07856* (2016).

[20] Peter W Shor. "Algorithms for quantum computation: Discrete logarithms and factoring". In: *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on.* Ieee. 1994, pp. 124–134.

[21] Sphincs+ Team. *SPHINCS+.* 2017. URL: `https://sphincs.org/`.

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

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

[24] Mark Zhandry. "Secure identity-based encryption in the quantum random oracle model". In: *International Journal of Quantum Information* 13.04 (2015), p. 1550014.

[25] Mark Zhandry. *How to Record Quantum Queries, and Applications to Quantum Indifferentiability.* Tech. rep. Cryptology ePrint Archive, Report 2018/276, 2018. https://eprint.iacr.org/2018/276, 2018.

A Direct proof of indistinguishability with permutations

Here we prove Theorem 16 by direct application of Theorem 4 instead of relying on the PRF/PRP switching lemma. For this proof we need to generalize the average-case polynomial method. We show how to use it if the probability of a certain input-output behavior is not a polynomial but is close to a polynomial. This small generalization might prove useful in other applications of the polynomial method. The following is a restatement of Theorem 16, with a slightly worse bound.

Theorem 17. $\textsc{SpGen}_{\varphi_{\mathbf{f}}}$ *for a random permutation* $\varphi_{\mathbf{f}}$ *is quantumly indistinguishable from a random oracle. More concretely, for all quantum algorithms* A *making at most* q *quantum queries to* $\textsc{SpGen}$*, such that the input length is at most* $m \cdot r$ *bits long and the output length is at most* $z \cdot r$ *bits long,*

$$\left| \underset{\varphi_{\mathbf{f}} \xleftarrow{\$} \mathcal{T}(\mathcal{S})}{\mathbb{P}} \left[\mathrm{A}^{|\textsc{SpGen}_{\varphi_{\mathbf{f}}}\rangle}(.) = 1 \right] - \underset{\mathbf{h} \leftarrow \mathfrak{R}}{\mathbb{P}} \left[\mathrm{A}^{|\mathbf{h}\rangle}(.) = 1 \right] \right| < \frac{\pi^2}{6} (2\eta)^3 (|\mathcal{C}| - 1)^{-1}, \tag{74}$$

where $\eta := 2q(m + z - 2)$, $\mathfrak{R}$ *is defined according to Definition 6, and the set of permutations is denoted by* $\mathcal{T}(\mathcal{S}) := \{\varphi_{\mathbf{f}} : \mathcal{S} \to \mathcal{S} \mid \varphi_{\mathbf{f}} \text{ is a bijection}\}$*. The domain is defined as* $\mathcal{S} = \{0,1\}^r \times \mathcal{C}$ *for some non-empty finite set* $\mathcal{C}$*.*

Proof sketch. The proof follows the same reasoning as the proof of Theorem 8 with small differences explained in the following. We define the family of distributions $\mathfrak{F}_t$ with random permutations $\varphi_{\mathbf{f}}$ from $\mathcal{T}(\mathcal{S})$. When we get to Eq (14) though, we need an argument different from Lemma 9 because it does not hold for permutations in $\textsc{SpGen}$. We perform the same analysis of the probability function as in the proof of Lemma 9. Only the final argument is missing as we cannot use Theorem 4: $\mathbb{P}\left[\forall i \in [2q] : \textsc{SpGen}_{\varphi_{\mathbf{f}}}(\mathbf{M}^i, \ell_i) = \mathbf{Z}^i\right]$ is not a polynomial in $|\mathcal{C}|^{-1}$ if $\varphi_{\mathbf{f}}$ is a permutation. Instead we formulate a generalization of Theorem 4 in Lemma 19 below that leads to the claimed bound. □

Let us now highlight the differences we encounter when analyzing the case of permutations when following the reasoning of the proof of Lemma 9. The first and main difference is that the expression for the probability of a single evaluation of $\varphi_{\mathbf{f}}$ (Equations (45)) changes to:

$$\begin{aligned} \textsc{Flag}(\nabla\text{-}\mathsf{cf}^i_j) \in \{\mathrm{u}, \mathrm{f}\} &\Rightarrow \mathbb{P}\left[\varphi_{\mathbf{f}}({}^{(\mathrm{u} \vee \mathrm{f})} S^i_{j\oplus}) = S \mid \bigwedge_{(i',j') \prec (i,j)} \left(\varphi_{\mathbf{f}}(S^{i'}_{j'\oplus}) = S^{i'}_{j'+1}\right)\right] = \frac{1}{2^r \cdot |\mathcal{C}| - \mathrm{U}_{\mathrm{prev}}(i,j)}, \\ \textsc{Flag}(\nabla\text{-}\mathsf{cf}^i_j) = \mathrm{n} &\Rightarrow \mathbb{P}\left[\varphi_{\mathbf{f}}({}^{\mathrm{n}} S^i_{j\oplus}) = S \mid \bigwedge_{(i',j') \prec (i,j)} \left(\varphi_{\mathbf{f}}(S^{i'}_{j'\oplus}) = S^{i'}_{j'+1}\right)\right] = \begin{cases} 1 & \text{if } S = \nabla\text{-}\mathsf{cf}^i_j(2) \\ 0 & \text{otherwise} \end{cases}. \end{aligned} \tag{75}$$

where

$$\mathrm{U}_{\mathrm{prev}}(i,j) := \left| \left\{ P^{i'}_{j'} : (i',j') \prec (i,j) \wedge \textsc{Flag}(P^{i'}_{j'}) \in \{\mathrm{u}, \mathrm{f}\} \right\} \right| \tag{76}$$

is the number of unique states preceding the position (i,j). Note that we assume we have done all steps leading to Eq.(45).

The product in Eq. (43) for $\mathsf{p}\text{-}\nabla\text{-}\mathsf{cf} \in \mathsf{p}\text{-}\nabla\text{-}\mathsf{CF}(\mathbf{M}, \mathbf{Z}, \bar{u})$ and for $\varphi_{\mathbf{f}}$ being a random permutation evaluates not to Eq. (52), but instead to

$$\prod_{(i,j)=(1,1)}^{(q,k_q-1)} \mathbb{P}\left[\left(\varphi_{\mathbf{f}}(S^i_{j\oplus}) = S^i_{j+1}\right) \mid \bigwedge_{(i',j') \prec (i,j)} \left(\varphi_{\mathbf{f}}(S^{i'}_{j'\oplus}) = S^{i'}_{j'+1}\right)\right] = \prod_{i=0}^{\bar{u}-1} \frac{1}{2^r \cdot |\mathcal{C}| - i}. \tag{77}$$

Algorithm 5 also changes when we consider random permutations. We need to shrink the set of possible states in the last column to $(|\mathcal{C}| - \mathrm{U}_{\mathrm{prev}}(P, i, k_i))$ (line **19** in Calc) and modify the number of possible assignments of values of non-unique states (line **20** in Calc). The rest is exactly the same, we will refer to the modified algorithm as CalcPer. The definition of the modified N-Possibilities can be found in Appendix B.

Up to this point we have shown how to deal with combinatorial problems emerging from changing the internal function to a permutation. The main problem is different though; for random permutations the expression we derive in the last step of the proof in Section 6.4 is not an expression in $1/|\mathcal{C}|$. The expression we end up with is similar to Eq. (66), but the probability comes from Eq. (77):

$$\mathbf{p}(|\mathcal{C}|^{-1}) = \sum_{\bar{u}_{\mathrm{abs}}, u_{\mathrm{abs}}, \bar{u}_{\mathrm{squ}}, u_{\mathrm{squ}}, P} \textsc{CalcPer}(P) \prod_{i=0}^{\bar{u}-1} \frac{1}{2^r \cdot |\mathcal{C}| - i} \,. \tag{78}$$

Unfortunately the above expression does not fit into the assumptions of Theorem 18 below which is the basis of the polynomial method allowing us to bound the adversary's advantage.

Theorem 18 (Theorem B.1. in [22] for $\Delta = 1$)**.** *Let $\mathbf{p}(\lambda)$ be a polynomial in λ of degree d such that $0 \le \mathbf{p}(0) \le 1$, and $0 \le \mathbf{p}(1/t) \le 1$ for all $t \in \mathbb{Z}_+$ Then $|\mathbf{p}(1/t) - \mathbf{p}(0)| < \frac{\pi^2 d^3}{6t}$ for all $t \in \mathbb{Z}_+$.*

To deal with this problem we state a lemma relaxing a bit the requirements of Theorem 3.

Lemma 19. *For every $2q$ pairs $\forall i \in [2q] : (X^i, Y^i) \in \mathcal{X} \times \mathcal{Y}$ we define a function $\mathbf{g}_j(1/t) := \underset{\mathbf{h} \leftarrow \mathfrak{F}_t}{\mathbb{P}}[\forall i \in [2q] : \mathbf{h}(X^i) = Y^i]$, where j is the index enumerating different pairs. Assume that there exists $a \in \mathbb{N}$ and for every j there exist polynomials $\mathbf{p}'_j, \mathbf{p}''_j$, such that for every j and every $t \in \mathbb{N}$ with $t > a$, $\mathbf{g}_j(1/t)$ is bounded from below and above by polynomials $\mathbf{p}'_j, \mathbf{p}''_j$ respectively:*

$$\mathbf{p}'_j\left(\frac{1}{t-a}\right) \le \mathbf{g}_j\left(\frac{1}{t}\right) \le \mathbf{p}''_j\left(\frac{1}{t-a}\right), \tag{79}$$

such that $\mathbf{p}'_j(0) = \mathbf{p}''_j(0) = \mathbf{g}_j(0)$ and the degrees of the polynomials are d' and d'' respectively. Moreover $0 \le \mathbf{p}'_j\left(\frac{1}{t-a}\right)$ and $\mathbf{p}''_j\left(\frac{1}{t-a}\right) \le 1$ for all $t \in \mathbb{N}$ with $t > a$. Then

$$\left| \underset{\mathbf{h} \leftarrow \mathfrak{F}_t}{\mathbb{P}}\left[\mathrm{A}^{|\mathbf{h}\rangle}() = 1\right] - \underset{\mathbf{h} \leftarrow \mathfrak{F}_\infty}{\mathbb{P}}\left[\mathrm{A}^{|\mathbf{h}\rangle}() = 1\right] \right| < \frac{\pi^2 (\max\{d', d''\})^3}{6(t-a)}. \tag{80}$$

Proof. Here we follow the proof of Theorem 7.3 in [22] to make sure all assumptions are fulfilled to use the lemma stating that closeness of polynomials implies small adversarial advantage. Let us say that $\underset{\mathbf{h} \leftarrow \mathfrak{F}_t}{\mathbb{P}}\left[\mathrm{A}^{|\mathbf{h}\rangle}() = 1\right] - \underset{\mathbf{h} \leftarrow \mathfrak{F}_\infty}{\mathbb{P}}\left[\mathrm{A}^{|\mathbf{h}\rangle}() = 1\right] = \boldsymbol{\epsilon}$, w.l.o.g. we can assume that $\boldsymbol{\epsilon} > 0$. Also without loss of generality, using the fact that $\underset{\mathbf{h} \leftarrow \mathfrak{F}_t}{\mathbb{P}}\left[\mathrm{A}^{|\mathbf{h}\rangle}() = 1\right]$ is a linear combination of $\underset{\mathbf{h} \leftarrow \mathfrak{F}_t}{\mathbb{P}}[\forall i \in [2q] : \mathbf{h}(X^i) = Y^i] = \mathbf{g}_j(1/t)$ we can assume that all the coefficients in this combination are real. Therefore, we have

$$\boldsymbol{\epsilon} = \sum_j \alpha_j (\mathbf{g}_j(1/t) - \mathbf{g}_j(0)) \le \sum_j \alpha_j (\tilde{\mathbf{p}}_j\left(\frac{1}{t-a}\right) - \mathbf{g}_j(0)), \tag{81}$$

where $\tilde{\mathbf{p}}_j = \begin{cases} \mathbf{p}'_j & \text{if } \alpha_j < 0 \\ \mathbf{p}''_j & \text{if } \alpha_j \ge 0 \end{cases}$. Note that $\tilde{\mathbf{p}}(\frac{1}{t-a}) := \sum_j \alpha_j \tilde{\mathbf{p}}_j\left(\frac{1}{t-a}\right)$ is a polynomial of degree at most $\max\{d', d''\}$. If we set $t' := t - a$, it is straightforward to verify that $\tilde{\mathbf{p}}$ fulfills the assumptions of Theorem 18 above for all $t' \in \mathbb{Z}_+$. As $\mathbf{g}_j, \mathbf{p}'_j, \mathbf{p}''_j, \tilde{\mathbf{p}}$ all take on the same value for $t \to \infty$, we obtain that $\tilde{\mathbf{p}}(0) = \mathbf{g}_j(0)$, and hence the claim follows. $\square$

Now we just need to show that $\mathbb{P}[\forall i \in [2q] : \textsc{SpGen}_{\boldsymbol{\varphi}_\mathbf{f}}(\mathbf{M}^i, \ell_i) = \mathbf{Z}^i]$ is bounded by polynomials and find their degree. We are going to show that there are $\mathbf{p}', \mathbf{p}''$ (indexed with j in the statement of the lemma) that are polynomials and that fulfill the assumptions of Lemma 19 for $a = 1$. For each set of pairs of inputs and outputs we consider $\mathbf{g}$ to be a sum like in Eq.(78). To do that we need to distinguish between placements P that involve at least two consecutive unique states and those that do not. Let us deal with the former case first. We can bound the probability part of Eq. (78) as follows:

$$\prod_{i=1}^{\bar{u}-1} \frac{1}{2^r \cdot |\mathcal{C}| - i} \leq \prod_{i=1}^{\bar{u}-1} \frac{1}{2^r \cdot |\mathcal{C}| - (\kappa - 1)} = \left(\frac{1}{2^r} \cdot \frac{1}{|\mathcal{C}| - \frac{\kappa-1}{2^r}}\right)^{\bar{u}-1} \leq \frac{1}{2^{r(\bar{u}-1)}} \left(\frac{1}{|\mathcal{C}| - 1}\right)^{\bar{u}-1}, \tag{82}$$

$$\prod_{i=1}^{\bar{u}-1} \frac{1}{2^r \cdot |\mathcal{C}| - i} \geq \prod_{i=1}^{\bar{u}-1} \frac{1}{2^r \cdot |\mathcal{C}|} \geq \frac{1}{2^{r(\bar{u}-1)}} \left(1 - \frac{1}{|\mathcal{C}| - 1}\right)^{\bar{u}-1} \left(\frac{1}{|\mathcal{C}| - 1}\right)^{\bar{u}-1}. \tag{83}$$

Note that we have skipped the first term in the product that is supposed to range from 0 to $\bar{u} - 1$. We have done it because in the case we discuss, the first term that is output by $\textsc{CalcPer}$ necessarily involves $|\mathcal{C}|$ (not $|\mathcal{C}| - 1$). Thanks to how we divide the product the final expression is a polynomial in $\frac{1}{|\mathcal{C}|-1}$. In the latter case, no consecutive pairs of unique states, we bound every element of the product like in the above inequalities.

As for $\textsc{CalcPer}$ we just treat $|\mathcal{C}| - 1$ as the new variable. Note that now $\mathbf{p}'_j$ and $\mathbf{p}''_j$ are polynomials in $(|\mathcal{C}| - 1)^{-1}$. Polynomial $\mathbf{p}''$ has the same degree as the the polynomial corresponding to $\mathbf{g}$ in the proof for functions, i.e. following the derivation of Eq. (68) it equals $d'' = \eta = 2q(m + z - 2)$. From the above lower bound however, we get that $d' = 2d''$.

The last assumption we need to check is for $\mathbf{p}', \mathbf{p}''$ to be bounded by 0 and 1 for $|\mathcal{C}| > 1$. Note that it is enough to show that $\mathbf{p}' \geq 0$ and $\mathbf{p}'' \leq 1$. We already know that $\mathbf{p}'$ and $\mathbf{p}''$ bound $\mathbf{g}$. For the lower bound $\mathbf{p}' \geq 0$ comes from the fact that all coefficients are positive (they equal $\textsc{CalcPer}(P)$) and so is $\left(1 - \frac{1}{|\mathcal{C}|-1}\right) \frac{1}{|\mathcal{C}|-1}$ for $|\mathcal{C}| > 1$. For the upper bound of $\mathbf{g}$ we need to check that $\mathbf{p}'' \leq 1$. Following the algorithm Alg. 5 we can see that $\textsc{CalcPer}(P)$ is bounded by $2^{r(\bar{u}-q)}|\mathcal{C}|(|\mathcal{C}| - 1)^{\bar{u}-1}$, so as long as the number of terms in Eq. (78) is smaller than 2^{qr}—which is our implicit assumption—then $\mathbf{p}'' \leq 1$.

The above discussion, together with Lemma 19 proves Theorem 17.

B Additional details

B.1 Auxiliary algorithms, functions

Let us consider the problem of assigning values to the non-unique states given some placement P. Let us denote the number of non-unique flags n by n and the number of "first" non-unique states with flags f by f. We are going to analyze the combinatorial problem of assigning n objects to f classes in a way that each class has assigned at least one object. Objects in a single class are indistinguishable but are distinguishable between different classes. For example three objects that we divide among two classes putting one object in the first class and two in the second can be assigned in three ways: we put into the first class the first object or the second object or the third. We do not count the fact that the two objects that are in class two can be in one order or another.The number of permutations in such a problem in the general case of n objects and f classes is given by

$$\mathbf{P}(n; n_1, n_2, \ldots, n_f) = \left(\frac{n!}{n_1! n_2! \cdots n_f!}\right). \tag{84}$$

Note that the above formula requires that we specify the occupation of classes. These occupation numbers are not fixed by the placement P so we also need to analyze these occupation numbers. The occupations of the classes are defined by the possible distribution of objects among different classes. Let us define the set of possible distributions:

$$\mathcal{D}(n, f) := \Big\{(n_1, n_2, \ldots, n_f) \in \mathbb{N}^f : \forall i \in [f], n_i \geq 1, n_1 + n_2 + \cdots + n_f = n\Big\}. \tag{85}$$

There is one more detail we need to add to properly count all possible assignments of non-unique states; repeated values of each different f appear in P only after the initial unique state. Let us denote by $\Pi_{\text{class-ind}}(n; n_1, n_2, \ldots, n_f)$ the set of permutations with classes of indistinguishable objects, enumerated by $\mathbf{P}\,(n; n_1, n_2, \ldots, n_f)$. We need to implement the requirement coming from the nature of working of Flag-Assign. The set of permutations after including this constraint is

$$\begin{aligned}\Pi\,(P, n; n_1, n_2, \ldots, n_f) := \{\pi \in \Pi_{\text{class-ind}}(n; n_1, n_2, \ldots, n_f) \mid \\ \text{there are no objects in class } i \text{ prior to the } i\text{-th state according to } P\}\,.\end{aligned} \tag{86}$$

Eventually we count the number of possible assignments of values of non-unique states:

$$\text{N-Possibilities}(n, f, P) := \sum_{(n_1, n_2, \ldots, n_f) \in \mathcal{D}(n,f)} |\Pi\,(P, n; n_1, n_2, \ldots, n_f)|\,. \tag{87}$$

The most crucial observation of this subsection is that the number of possible assignments does not depend on $|\mathcal{C}|$ and

$$\text{N-Possibilities}(n, f, P) \leq f^n. \tag{88}$$

The above is a trivial bound found by ignoring all structure and only counting the total number of possibilities to put one of f values in every of the n places.

B.2 Auxiliary algorithms, permutations

In the case of permutations we need to add one constraint to N-Possibilities

$$\begin{aligned}\Pi_{\text{per}}\,(P, n; n_1, n_2, \ldots, n_f) := \{\pi \in \Pi_{\text{class-ind}}(n; n_1, n_2, \ldots, n_f) \mid \\ \text{there are no objects in class } i \text{ prior to the } i\text{-th state according to } P \wedge \\ \text{non-unique } \boxed{\text{n}} \text{ outputs of unique states } (\boxed{\text{u}} \vee \boxed{\text{f}}) \text{ have different values}\}.\end{aligned} \tag{89}$$

Eventually we count the number of possible assignments of values of non-unique states:

$$\text{N-PossibilitiesPer}(n, f, P) := \sum_{(n_1, n_2, \ldots, n_f) \in \mathcal{D}(n,f)} |\Pi_{\text{per}}\,(P, n; n_1, n_2, \ldots, n_f)|\,. \tag{90}$$