Quantum Indistinguishability of Random Sponges

Jan Czajkowski$^{1\star[0000-0002-5441-7894]}$, Andreas Hülsing$^{2\star\star[0000-0003-2215-4134]}$, and Christian Schaffner$^{1\star\star\star[0000-0002-1754-1415]}$

[1] QuSoft, University of Amsterdam
[2] TU Eindhoven

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.

Keywords: Symmetric cryptography, keyed sponges, indistinguishability, quantum security, message-authentication codes

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, 17, 20, 1, 13]. For all these applications it is

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

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 [17, 1, 13] 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 [22] 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 [16, 14, 21]. 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. Quantum security of sponges was also analyzed in [11], although the authors there focused on different properties then we.

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.

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 $\mathbf{f}_K$.

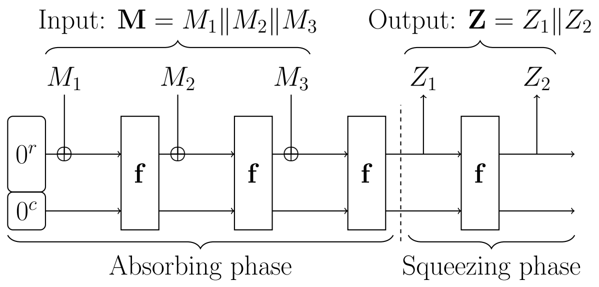
Input: $\mathbf{M} = M_1 \| M_2 \| M_3$
Output: $\mathbf{Z} = Z_1 \| Z_2$
M_1
M_2
M_3
Z_1
Z_2
0^r
0^c
f
f
f
f
Absorbing phase
Squeezing phase

Fig. 1. A scheme illustrating the sponge construction.

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[3] 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 [14] and Santoli, and Schaffner [21] 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 the full version of the paper [12] we show a direct proof of indistinguishability for $\mathbf{f}$ being a random permutation. In this proof we state and prove a lemma that generalizes the average case polynomial method to allow for func-

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

tions 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 [16]. 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. We also need to stress that our result so far does not cover the commonly used approaches to secretly key SHA3 for this very reason.

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 [24] (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 [24] 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 [25], 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 [24] 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. In the full version [12] we also provide 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 5 contains the proof of Lemma 9, the main technical result of this work. The case of random permutations is covered in Section 6. We conclude the paper with Section 7 discussing some open problems related to the problem we analyze and related work.

2 Preliminaries and Tools

In this section we introduce the definition of quantum indistinguishability and other notions used throughout this work.

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 [19] 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 $\xleftarrow{\$} \mathcal{X}$ 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 [24])**.** *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 $\xleftarrow{\$} \mathcal{Y}^{\mathcal{X}}$.

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 $[q] := \{1, 2, \dots, q\}$.

Theorem 3 (Theorem 3.1 in [26])**.** *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 [24])**.** *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

In this section we give a formal definition of sponges and recall a known result about their indisitinguishability.

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 Sponge)**.** *If* $\mathbf{f}$ *is a random transformation or a random permutation then* $\text{Sponge}_{\mathbf{f}}$ *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*

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

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 [18], 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:

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

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 6 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,*

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

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* $\mathcal{C}$.

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

```
   Input  : $\mathbf{M} \in \{0,1\}^*$, $\ell \geq 0$.
   Output: $\mathbf{Z} \in \{0,1\}^\ell$
1  $\mathbf{P} := \textsc{pad}(\mathbf{M})$
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 $\lfloor \mathbf{Z} \rfloor_\ell$
```

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

Lemma 9. *For a fixed q and for every $(\mathbf{M}, \mathbf{Z}) := \left((\mathbf{M}^i, \mathbf{Z}^i)\right)_{i \in [2q]}$, 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

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

(ii) and the coefficient

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

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 5.

Proof idea. Our goal is to explicitly evaluate $\mathbb{P}\left[\forall i \in [2q] : \textsc{SpGen}_{\boldsymbol{\varphi}_{\mathbf{f}}}(\mathbf{M}^i, \ell_i) = \mathbf{Z}^i\right]$. 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 $\left(\forall i \in [2q] : \textsc{SpGen}_{\boldsymbol{\varphi}_{\mathbf{f}}}(\mathbf{M}^i, \ell_i) = \mathbf{Z}^i\right)$. 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 $\textsc{SpGen}$ 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 $\left(\forall i \in [2q] : \textsc{SpGen}_{\boldsymbol{\varphi}_{\mathbf{f}}}(\mathbf{M}^i, \ell_i) = \mathbf{Z}^i\right)$ 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 $\mathbb{P}\left[\forall i \in [2q] : \textsc{SpGen}_{\boldsymbol{\varphi}_{\mathbf{f}}}(\mathbf{M}^i, \ell_i) = \mathbf{Z}^i\right]$, 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 $\textsc{pad}$. 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 $\textsc{pad}$ to construct $\textsc{SpGen}_{\boldsymbol{\varphi}_{\mathbf{f}}}[\textsc{pad}, r, \mathcal{C}]$.
- For each $(X, \ell) \in \mathcal{M} \times \mathbb{N}$ set $\mathbf{h}(X, \ell) := \textsc{SpGen}_{\boldsymbol{\varphi}_{\mathbf{f}}}[\textsc{pad}, r, \mathcal{C}](X, \ell)$.

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

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

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 $\textsc{SpGen}$ what in turn implies a random and independent outer part in every step of generating the output, formally

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

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

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

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* 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*

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

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* $\textsc{SpGen}_{\mathbf{f}}$ *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*

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

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.

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

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

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 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 5.1 we expand the probability expression to encompass all intermediate states of $\big(\forall i \in [2q] : \textsc{SpGen}_{\boldsymbol{\varphi}_{\mathbf{f}}}(\mathbf{M}^i, \ell_i) = \mathbf{Z}^i\big)$ and individual evaluations of $\boldsymbol{\varphi}_{\mathbf{f}}$. In Subsection 5.2 we introduce the concept of unique states to evaluate the probabilities of $\mathbb{P}[\boldsymbol{\varphi}_{\mathbf{f}}(S_1) = S_2]$. In Subsection 5.3 we define the algorithm that calculates the cardinality of the set of intermediate states—and equivalently inner functions—consistent with given characteristics. In Subsection 5.4 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 $\textsc{SpGen}$ instead of $2q$ to improve readability.

5.1 Expansion of the probability function

In this section we expand the probability function to the point that all intermediate states are accounted for. We consider the event $\big(\forall i \in [q] : \textsc{SpGen}_{\boldsymbol{\varphi}_{\mathbf{f}}}(\mathbf{M}^i, \ell_i) = \mathbf{Z}^i\big)$ and then include the states that appear between consecutive evaluations of $\boldsymbol{\varphi}_{\mathbf{f}}$.

To keep track of the states we introduce the following notation. By the upper-index we denote the number of evaluations of $\textsc{SpGen}$, 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 $\textsc{SpGen}$. 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}. States traversed in q evaluations of $\textsc{SpGen}$ can be represented by an array with q rows with $|\mathbf{M}^i|_r + |\mathbf{Z}^i|_r$ columns each. By *array* we mean a 2-dimensional matrix with unequal length of rows.

We call an array like that with values assigned to every state a *nabla configuration* ∇-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. Now we define ∇-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 (∇-c)**.** *The nabla configuration* ∇-c *for* $(\mathbf{M}, \mathbf{Z})$ *is an array of triples* $\begin{pmatrix} \bar{S} & \bar{S}_{\oplus} \\ \multicolumn{2}{c}{\hat{S}} \end{pmatrix} \in \{0,1\}^{2r} \times \mathcal{C}$*, where* $\mathcal{C}$ *is an arbitrary non-empty finite set. The array* ∇-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*

$$\nabla\text{-}\mathsf{c} := \left[\begin{pmatrix} \bar{S}^i_j \; \bar{S}^i_{j\oplus} \\ \hat{S}^i_j \end{pmatrix} \right]_{\substack{i\in[q] \\ j\in[k_i]}} . \tag{19}$$

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

$$\begin{aligned} S^i_j \in \nabla\text{-}\mathsf{c} \Leftrightarrow \nabla\text{-}\mathsf{c}^i_j = \begin{pmatrix} \bar{S} \; \bar{S}_\oplus \\ \hat{S} \end{pmatrix} \wedge \; S^i_j = (\bar{S}, \hat{S}) \\ S^i_{j\oplus} \in \nabla\text{-}\mathsf{c} \Leftrightarrow \nabla\text{-}\mathsf{c}^i_j = \begin{pmatrix} \bar{S} \; \bar{S}_\oplus \\ \hat{S} \end{pmatrix} \wedge \; S^i_{j\oplus} = (\bar{S}_\oplus, \hat{S}) \end{aligned} \tag{20}$$

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

$$\kappa := \sum_{i=1}^{q} (k_i - 1), \tag{21}$$

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

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

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

$$\begin{aligned} \forall i \in [q] :\; & \hat{S}^i_1 = I_{\mathcal{C}}, \\ \forall i \in [q] :\; & \bar{S}^i_1 = 0^r, \\ \forall i \in [q], 1 \le j \le |\mathbf{M}^i|_r :\; & \bar{S}^i_{j\oplus} = \bar{S}^i_j \oplus M^i_j, \\ \forall i \in [q], |\mathbf{M}^i|_r < j \le k_i :\; & \bar{S}^i_{j\oplus} = \bar{S}^i_j = Z^i_{j-|\mathbf{M}^i|_r}. \end{aligned} \tag{22}$$

The formal definition reads

$$\nabla\text{-}\mathsf{C}(\mathbf{M}, \mathbf{Z}) := \{\nabla\text{-}\mathsf{c} \text{ for } (\mathbf{M}, \mathbf{Z}) : \nabla\text{-}\mathsf{c} \text{ fulfills constraints (22)}\}. \tag{23}$$

In the following we assume that rows of all $\nabla\text{-}\mathsf{c} \in \nabla\text{-}\mathsf{C}(\mathbf{M}, \mathbf{Z})$ 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 $\forall i < j : k_i \le k_j$, this also means that rows of $\nabla\text{-}\mathsf{c}$ 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 $\left(\boldsymbol{\varphi}_{\mathbf{f}}(S^i_{j\oplus}) = S^i_{j+1}\right)$ for some $\nabla\text{-}\mathsf{c}$ we leave implicit that $S^i_{j\oplus}, S^i_{j+1} \in \nabla\text{-}\mathsf{c}$. We have that

$$\begin{aligned}\forall i \in [q] : \textsc{SpGen}(\mathbf{M}^i) = \mathbf{Z}^i \Leftrightarrow & \forall i \in [q] : \bigvee_{\nabla\text{-}\mathsf{c} \in \nabla\text{-}\mathsf{C}(\mathbf{M},\mathbf{Z})} \left(\boldsymbol{\varphi}_{\mathbf{f}}(S^i_{1\oplus}) = S^i_2\right) \\ & \wedge \left(\boldsymbol{\varphi}_{\mathbf{f}}(S^i_{2\oplus}) = S^i_3\right) \wedge \cdots \wedge \left(\boldsymbol{\varphi}_{\mathbf{f}}(S^i_{(k_i-1)\oplus}) = S^i_{k_i}\right) \quad (24) \\ \Leftrightarrow & \bigvee_{\nabla\text{-}\mathsf{c} \in \nabla\text{-}\mathsf{C}(\mathbf{M},\mathbf{Z})} \bigwedge_{i=1}^{q} \bigwedge_{j=1}^{k_i-1} \left(\boldsymbol{\varphi}_{\mathbf{f}}(S^i_{j\oplus}) = S^i_{j+1}\right). \quad (25)\end{aligned}$$

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.

$$\begin{aligned}&\mathop{\mathbb{P}}_{\boldsymbol{\varphi}_{\mathbf{f}} \xleftarrow{\$} \mathcal{S}^{\mathcal{S}}} \left[\forall i \in [q] : \textsc{SpGen}(\mathbf{M}^i) = \mathbf{Z}^i\right] = \mathbb{P}\left[\bigvee_{\nabla\text{-}\mathsf{c}} \bigwedge_{i=1}^{q} \bigwedge_{j=1}^{k_i-1} \left(\boldsymbol{\varphi}_{\mathbf{f}}(S^i_{j\oplus}) = S^i_{j+1}\right)\right] \\ &= \sum_{\nabla\text{-}\mathsf{c} \in \nabla\text{-}\mathsf{C}(\mathbf{M},\mathbf{Z})} \mathbb{P}\left[\bigwedge_{i=1}^{q} \bigwedge_{j=1}^{k_i-1} \left(\boldsymbol{\varphi}_{\mathbf{f}}(S^i_{j\oplus}) = S^i_{j+1}\right)\right]. \quad (26)\end{aligned}$$

To further extract an expression involving the probability of a single $\left(\boldsymbol{\varphi}_{\mathbf{f}}(S^i_{j\oplus}) = S^i_{j+1}\right)$ 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 $\left(\boldsymbol{\varphi}_{\mathbf{f}}(S^i_{j\oplus}) = S^i_{j+1}\right)$ events. Let us define the order $\prec$ as

$$(i, j) \prec (i', j') \Leftrightarrow (j < j') \vee (j = j' \wedge i < i'). \quad (27)$$

The above rule imposes an order that begins with the top-left corner of a $\nabla\text{-}\mathsf{c}$ and proceeds downwards to the end of the column to continue from the second column from the left.

$$\begin{aligned}\mathbf{p}(|\mathcal{C}|^{-1}) &= \sum_{\nabla\text{-}\mathsf{c} \in \nabla\text{-}\mathsf{C}(\mathbf{M},\mathbf{Z})} \mathbb{P}\left[\bigwedge_{i=1}^{q} \bigwedge_{j=1}^{k_i-1} \left(\boldsymbol{\varphi}_{\mathbf{f}}(S^i_{j\oplus}) = S^i_{j+1}\right)\right] \\ &= \sum_{\nabla\text{-}\mathsf{c}} \mathbb{P}\left[\left(\boldsymbol{\varphi}_{\mathbf{f}}(S^q_{(k_q-1)\oplus}) = S^q_{k_q}\right) \mid \bigwedge_{(i,j) \prec (q,k_q-1)} \left(\boldsymbol{\varphi}_{\mathbf{f}}(S^i_{j\oplus}) = S^i_{j+1}\right)\right] \\ &\cdot \mathbb{P}\left[\bigwedge_{(i,j) \prec (q,k_q-1)} \left(\boldsymbol{\varphi}_{\mathbf{f}}(S^i_{j\oplus}) = S^i_{j+1}\right)\right] \\ &= \sum_{\nabla\text{-}\mathsf{c} \in \nabla\text{-}\mathsf{C}(\mathbf{M},\mathbf{Z})} \prod_{(i,j)=(1,1)}^{(q,k_q-1)} \mathbb{P}\left[\left(\boldsymbol{\varphi}_{\mathbf{f}}(S^i_{j\oplus}) = S^i_{j+1}\right) \mid \bigwedge_{(i',j') \prec (i,j)} \left(\boldsymbol{\varphi}_{\mathbf{f}}(S^{i'}_{j'\oplus}) = S^{i'}_{j'+1}\right)\right]. \quad (28)\end{aligned}$$

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 (27).

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 (28) in more detail. To this end, we add more structure to ∇-$\mathsf{C}(\mathbf{M}, \mathbf{Z})$ 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 ∇-$\mathsf{C}(\mathbf{M}, \mathbf{Z})$.

5.2 Unique and non-unique states

The goal of this section is to evaluate
$\mathbb{P}\left[\left(\boldsymbol{\varphi}_{\mathbf{f}}(S^i_{j\oplus}) = S^i_{j+1}\right) \mid \bigwedge_{(i',j')\prec(i,j)} \left(\boldsymbol{\varphi}_{\mathbf{f}}(S^{i'}_{j'\oplus}) = S^{i'}_{j'+1}\right)\right]$ for any ∇-c and any $(\mathbf{M}, \mathbf{Z})$. We approach this problem by recognizing which states in a particular ∇-c are fed to $\boldsymbol{\varphi}_{\mathbf{f}}$ 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. (28), 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 $\nabla\text{-}\mathsf{c}$. 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 $\nabla\text{-}\mathsf{c}$ 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. 27. The output of Algorithm 3 $\textsc{Prep}(\nabla\text{-}\mathsf{c})$ is a vector (1-dimensional matrix), to access its l-th element we write $\nabla\text{-}\mathsf{c}_l$.

```
Algorithm 3: PREP
    Input  : $\nabla\text{-}\mathsf{c}$ for $(\mathbf{M}, \mathbf{Z})$
    Output: $\widetilde{\nabla\text{-}\mathsf{c}}$
1 $\widetilde{\nabla\text{-}\mathsf{c}} := \nabla\text{-}\mathsf{c}$, append three work spaces to each element of $\widetilde{\nabla\text{-}\mathsf{c}}$
2 foreach $1 \le i \le q, 1 \le j \le k_i - 1$ do
3     $\widetilde{\nabla\text{-}\mathsf{c}}^i_j = \big(\nabla\text{-}\mathsf{c}^i_j, \mathtt{index}, \oplus\text{-}\mathtt{state}, \mathtt{image}\big) := \big(\nabla\text{-}\mathsf{c}^i_j, (i,j), S^i_{j\oplus}, S^i_{j+1}\big)$
4 Sort $\widetilde{\nabla\text{-}\mathsf{c}}$ primarily according to the third entry and secondarily according to the
    second entry (using the order defined in Equation (27)).
5 Output $\widetilde{\nabla\text{-}\mathsf{c}}$
```

The main contribution of this subsection is Algorithm 4 which adds to each $\nabla\text{-}\mathsf{c}$ information about the repetitions of the internal states. Running $\textsc{Prep}$ 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, $\boxed{\mathrm{u}}$ for unique states, $\boxed{\mathrm{n}}$ for non-unique states, and $\boxed{\mathrm{f}}$ for states that appear twice or more in total but from our perspective it is their first appearance. The output of Algorithm 4 is $\textsc{Flag-Assign}(\nabla\text{-}\mathsf{c}) = \nabla\text{-}\mathsf{cf}$ ("nabla configuration with flags") and $\forall i,j : \nabla\text{-}\mathsf{cf}^i_j = (\boxed{\mathrm{F}}\,\nabla\text{-}\mathsf{c}^i_j, S)$, 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 $\nabla\text{-}\mathsf{c}^i_j$ we write $\nabla\text{-}\mathsf{c}^i_j(l)$. Flag $\boxed{\mathrm{f}}$ is important when discussing the relative position of unique flags ($\boxed{\mathrm{u}}$ or $\boxed{\mathrm{f}}$) in the array of $\nabla\text{-}\mathsf{cf}$. 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 $\nabla\text{-}\mathsf{cf}$ output by $\textsc{Flag-Assign}$. $\textsc{Flag} : \{\boxed{\mathrm{u}}, \boxed{\mathrm{f}}, \boxed{\mathrm{n}}\} \times \{0,1\}^{2r} \times \mathcal{C} \to \{\boxed{\mathrm{u}}, \boxed{\mathrm{f}}, \boxed{\mathrm{n}}\}$,

$$\textsc{Flag}(\nabla\text{-}\mathsf{cf}^i_j) = \textsc{Flag}(\begin{pmatrix} \bar{S}^i_j \; \boxed{\mathrm{F}}\bar{S}^i_{j\oplus} \\ \boxed{\mathrm{F}}\hat{S}^i_j \end{pmatrix}, S) := \boxed{\mathrm{F}}. \tag{29}$$

Algorithm 4: Flag-Assign

```
    Input  : $\nabla$-c for $(\mathbf{M}, \mathbf{Z})$
    Output: $\nabla$-cf
 1  $\nabla\text{-cf} = \emptyset$
 2  $\widetilde{\nabla\text{-c}} := \textsc{Prep}(\nabla\text{-c})$
 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 $\left(\begin{pmatrix}\bar{S}\,\boxed{u}\bar{S}_\oplus \\ \boxed{u}\hat{S}\end{pmatrix}, \widetilde{\nabla\text{-c}}_l(2), \widetilde{\nabla\text{-c}}_l(4)\right)$ 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 $\left(\begin{pmatrix}\bar{S}\,\boxed{f}\bar{S}_\oplus \\ \boxed{f}\hat{S}\end{pmatrix}, \widetilde{\nabla\text{-c}}_l(2), \widetilde{\nabla\text{-c}}_l(4)\right)$ 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 $\left(\begin{pmatrix}\bar{S}\,\boxed{n}\bar{S}_\oplus \\ \boxed{n}\hat{S}\end{pmatrix}, \widetilde{\nabla\text{-c}}_l(2), \widetilde{\nabla\text{-c}}_l(4)\right)$ 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       // $\nabla\text{-cf}^{\,i}_{j}$ =(state with a flag, image)
17  Output $\nabla$-cf
```

Transition probabilities in Equation (28) depend on the flags we assigned to states in ∇-c. We have that

$$\begin{aligned}
\textsc{Flag}(\nabla\text{-cf}^i_j) \in \{\boxed{u}, \boxed{f}\} \Rightarrow \mathbb{P}&\left[\varphi_{\mathbf{f}}(^{\boxed{u}\vee\boxed{f}}S^i_{j\oplus}) = S \mid \bigwedge_{(i',j')\prec(i,j)} \left(\varphi_{\mathbf{f}}(S^{i'}_{j'\oplus}) = S^{i'}_{j'+1}\right)\right] \\
&= \frac{1}{2^r \cdot |\mathcal{C}|}, \\
\textsc{Flag}(\nabla\text{-cf}^i_j) = \boxed{n} \Rightarrow \mathbb{P}&\left[\varphi_{\mathbf{f}}(^{\boxed{n}}S^i_{j\oplus}) = S \mid \bigwedge_{(i',j')\prec(i,j)} \left(\varphi_{\mathbf{f}}(S^{i'}_{j'\oplus}) = S^{i'}_{j'+1}\right)\right] \\
&= \begin{cases} 1 & \text{if } S = \nabla\text{-cf}^i_j(2) \\ 0 & \text{otherwise} \end{cases}.
\end{aligned} \tag{30}$$

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

In this section we evaluate the number of intermediate states that give $\big(\forall i \in [q] : \textsc{SpGen}_{\boldsymbol{\varphi}_{\mathbf{f}}}(\mathbf{M}^i, \ell_i) = \mathbf{Z}^i\big)$. First we impose the constraint of $\boldsymbol{\varphi}_{\mathbf{f}}$ being a function. Then we want to calculate the product of probabilities in Eq. (28). 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 ∇-$\mathsf{C}(\mathbf{M},\mathbf{Z})$.

In the process of calculating the conditional probabilities in Eq. (28) we included in each state in ∇-c the image it should have under $\boldsymbol{\varphi}_{\mathbf{f}}$. The set ∇-$\mathsf{C}(\mathbf{M},\mathbf{Z})$ does however contain states that would violate the constraint of $\boldsymbol{\varphi}_{\mathbf{f}}$ being a function. The first step to calculate the cardinality of ∇-$\mathsf{C}(\mathbf{M},\mathbf{Z})$ 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-∇-$\mathsf{CF}(\mathbf{M},\mathbf{Z})$ (p emphasizes the fact that $\boldsymbol{\varphi}_{\mathbf{f}}$ is a proper function).

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

$$\nabla\text{-}\mathsf{CF}(\mathbf{M},\mathbf{Z}) := \{\nabla\text{-}\mathsf{c} : \exists\, \nabla\text{-}\mathsf{c}_0 \in \nabla\text{-}\mathsf{C}(\mathbf{M},\mathbf{Z}), \nabla\text{-}\mathsf{c} = \textsc{Flag-Assign}(\nabla\text{-}\mathsf{c}_0)\}\,. \tag{31}$$

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

$$\forall S^i_j \in \nabla\text{-}\mathsf{cf}\; \forall j > 1 : S^i_j = \nabla\text{-}\mathsf{cf}^i_{j-1}(2). \tag{32}$$

The formal definition reads

$$\mathsf{p}\text{-}\nabla\text{-}\mathsf{CF}(\mathbf{M},\mathbf{Z}) := \{\nabla\text{-}\mathsf{cf} \in \nabla\text{-}\mathsf{CF}(\mathbf{M},\mathbf{Z}) : \nabla\text{-}\mathsf{cf} \text{ fulfills constraints (32)}\}\,. \tag{33}$$

One may think about p-∇-$\mathsf{CF}(\mathbf{M},\mathbf{Z})$ 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 (22)—is denoted by ∇-$\mathsf{C}(\mathbf{M},\mathbf{Z})$. On each ∇-$\mathsf{c} \in \nabla$-$\mathsf{C}(\mathbf{M},\mathbf{Z})$ we run the algorithm $\textsc{Flag-Assign}$, getting a collection of ∇-cf—denoted by ∇-$\mathsf{CF}(\mathbf{M},\mathbf{Z})$. Now we discard all those ∇-cf that do no fulfill constraints (32). The collection we are left with is denoted by p-∇-$\mathsf{CF}(\mathbf{M},\mathbf{Z})$. We have the following relations between sets:

$$\nabla\text{-}\mathsf{CF}(\mathbf{M},\mathbf{Z})(1) \overset{\text{omitting the flags}}{\simeq} \nabla\text{-}\mathsf{C}(\mathbf{M},\mathbf{Z}) \tag{34}$$

$$\mathsf{p}\text{-}\nabla\text{-}\mathsf{CF}(\mathbf{M},\mathbf{Z}) \subset \nabla\text{-}\mathsf{CF}(\mathbf{M},\mathbf{Z})\,. \tag{35}$$

Each p-∇-$\mathsf{cf} \in \mathsf{p}$-$\nabla$-$\mathsf{CF}(\mathbf{M},\mathbf{Z})$ has some number of unique states: with flag $\boxed{\mathrm{u}}$ or $\boxed{\mathrm{f}}$. Let us denote this number by $\bar{u}$. Eq. (30) implies that no matter in what configurations the unique states are, the product of probabilities in Eq. (28) is the same. Hence the first division of p-∇-$\mathsf{CF}(\mathbf{M},\mathbf{Z})$ 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

$$\mathsf{p}\text{-}\nabla\text{-}\mathsf{CF}(\mathbf{M}, \mathbf{Z}) = \bigcup_{\bar{u}=1}^{\kappa} \mathsf{p}\text{-}\nabla\text{-}\mathsf{CF}(\mathbf{M}, \mathbf{Z}, \bar{u}). \tag{36}$$

The product in Eq. (28) for $\mathsf{p}\text{-}\nabla\text{-}\mathsf{cf} \in \mathsf{p}\text{-}\nabla\text{-}\mathsf{CF}(\mathbf{M}, \mathbf{Z}, \bar{u})$ evaluates to

$$\prod_{(i,j)=(1,1)}^{(q,k_q-1)} \mathbb{P}\left[\left(\boldsymbol{\varphi}_{\mathbf{f}}(S^{i}_{j\oplus}) = S^{i}_{j+1}\right) \mid \bigwedge_{(i',j') \prec (i,j)} \left(\boldsymbol{\varphi}_{\mathbf{f}}(S^{i'}_{j'\oplus}) = S^{i'}_{j'+1}\right) \right] = \left(\frac{1}{2^r \cdot |\mathcal{C}|} \right)^{\bar{u}}, \tag{37}$$

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 $\left(\boldsymbol{\varphi}_{\mathbf{f}}({}^{(\boxed{u}\vee\boxed{f})}S_{\oplus}) = {}^{(\boxed{u}\vee\boxed{f})}S\right)$ 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:

(a) $\left(\boldsymbol{\varphi}_{\mathbf{f}}({}^{(\boxed{u}\vee\boxed{f})}S_{\oplus}) = {}^{(\boxed{u}\vee\boxed{f})}S\right)$—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),

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

(c) $\left(\boldsymbol{\varphi}_{\mathbf{f}}({}^{\boxed{n}}S_{\oplus}) = {}^{(\boxed{u}\vee\boxed{f}\vee\boxed{n})}S\right)$—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 $\boxed{u}$ 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 $\boxed{u}$ 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 $(i, j) \in \{(i', j')\}_{i' \in \{1,\ldots,q\}, j' \in \{1,\ldots,|\mathbf{M}^{i'}|_r\}}$. The number of states with the flag $\boxed{u}$ in the squeezing phase is denoted by u_{squ} and we take into account states with indices $(i, j) \in \{(i', j')\}_{i' \in \{1,\ldots,q\}, j' \in \{|\mathbf{M}^{i'}|_r+1,\ldots,k_{i'}-1\}}$. 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 $\mathsf{p}\text{-}\nabla\text{-}\mathsf{cf} \in \mathsf{p}\text{-}\nabla\text{-}\mathsf{CF}(\mathbf{M}, \mathbf{Z}, \bar{u}_{\text{abs}}, u_{\text{abs}}, \bar{u}_{\text{squ}}, u_{\text{squ}})$. We no longer need to keep u and $\bar{u}$ explicit as $u = u_{\text{abs}} + u_{\text{squ}}$ and $\bar{u} = \bar{u}_{\text{abs}} + \bar{u}_{\text{squ}}$. 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 $\mathcal{P}(\mathbf{M}, \mathbf{Z}, \bar{u}_{\text{abs}}, u_{\text{abs}}, \bar{u}_{\text{squ}}, u_{\text{squ}})$ is defined as the set of all placements P encountered in elements of $\mathsf{p}\text{-}\nabla\text{-}\mathsf{CF}(\mathbf{M}, \mathbf{Z}, \bar{u}_{\text{abs}}, u_{\text{abs}}, \bar{u}_{\text{squ}}, u_{\text{squ}})$. 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 $\bar{u}_{\text{abs}}$ 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 $\textsc{Flag-Assign}$ 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 $\boxed{\mathrm{u}}$ or $\boxed{\mathrm{f}}$) and then assign them concrete values ($\boxed{\mathrm{u}}$ or $\boxed{\mathrm{f}}$). 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 $\boxed{\mathrm{u}}$ 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

$$|\mathcal{P}(\mathbf{M}, \mathbf{Z}, \bar{u}_{\text{abs}}, u_{\text{abs}}, \bar{u}_{\text{squ}}, u_{\text{squ}})| \leq \binom{\mu}{\bar{u}_{\text{abs}}} \binom{\bar{u}_{\text{abs}}}{u_{\text{abs}}} \cdot \binom{\zeta}{\bar{u}_{\text{squ}}} \binom{\bar{u}_{\text{squ}}}{u_{\text{squ}}} \qquad (38)$$

$$\leq \binom{\mu}{\mu/2}^2 \binom{\zeta}{\zeta/2}^2 \leq \binom{\kappa}{\kappa/2}^4 \leq \kappa^{4\kappa}. \qquad (39)$$

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 $(\mathbf{M}, \mathbf{Z})$:

$$\boldsymbol{\delta}(\mathbf{M}, \mathbf{Z}, P) := \begin{cases} 1 \text{ if } \mathsf{p}\text{-}\nabla\text{-}\mathsf{CF}(\mathbf{M}, \mathbf{Z}, P) \neq \emptyset \\ 0 \text{ otherwise} \end{cases} . \tag{40}$$

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 by principle (a). 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 ($\boxed{\text{u}}$ or $\boxed{\text{f}}$) 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 $P \in \mathcal{P}(\mathbf{M}, \mathbf{Z}, \bar{u}_{\text{abs}}, u_{\text{abs}}, \bar{u}_{\text{squ}}, u_{\text{squ}})$ there are still many possibilities of arranging outer and inner states flags. There are exactly three possibilities every unique state can be arranged in: $\begin{pmatrix}\boxed{\text{u}}\vee\boxed{\text{f}}\\ \boxed{\text{u}}\vee\boxed{\text{f}}\end{pmatrix}$, $\begin{pmatrix}\boxed{\text{u}}\vee\boxed{\text{f}}\\ \boxed{\text{n}}\end{pmatrix}$, and $\begin{pmatrix}\boxed{\text{n}}\\ \boxed{\text{u}}\vee\boxed{\text{f}}\end{pmatrix}$, 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 $3^{\bar{u}_{\text{abs}}+\bar{u}_{\text{squ}}}$ 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 $P \in \mathcal{P}^*(\mathbf{M}, \mathbf{Z}, \bar{u}_{\text{abs}}, u_{\text{abs}}, \bar{u}_{\text{squ}}, u_{\text{squ}})$. We have that

$$|\mathcal{P}^*(\mathbf{M}, \mathbf{Z}, \bar{u}_{\text{abs}}, u_{\text{abs}}, \bar{u}_{\text{squ}}, u_{\text{squ}})| \leq \kappa^{4\kappa} \cdot 3^{\bar{u}_{\text{abs}}+\bar{u}_{\text{squ}}} . \tag{41}$$

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 $\mathsf{p}\text{-}\nabla\text{-}\mathsf{cf} \in \mathsf{p}\text{-}\nabla\text{-}\mathsf{CF}(\mathbf{M}, \mathbf{Z}, \bar{u}_{\text{abs}}, u_{\text{abs}}, \bar{u}_{\text{squ}}, u_{\text{squ}})$ for some given placement $P \in \mathcal{P}^*(\mathbf{M}, \mathbf{Z}, \bar{u}_{\text{abs}}, u_{\text{abs}}, \bar{u}_{\text{squ}}, u_{\text{squ}})$. To capture the fact that the number of pos-

Algorithm 5: Calc

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

sible 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

$$\bar{\mathrm{U}}_{\mathrm{prev}}(P,i,j) := \left|\left\{P^{i'}_{j'} : (i',j') \prec (i,j) \wedge \textsc{Flag}(\bar{P}^{i'}_{j'}) \in \{\boxed{\mathrm{u}}, \boxed{\mathrm{f}}\}\right\}\right|, \tag{42}$$

$$\bar{\mathrm{U}}^{\boxed{\mathrm{f}}}_{\mathrm{prev}}(P,i,j) := \left|\left\{P^{i'}_{j'} : (i',j') \prec (i,j) \wedge \textsc{Flag}(\bar{P}^{i'}_{j'}) = \boxed{\mathrm{f}}\right\}\right|. \tag{43}$$

For unique inner states we have

$$\hat{\mathrm{U}}_{\mathrm{prev}}(P,i,j) := \left|\left\{P^{i'}_{j'} : (i',j') \prec (i,j) \wedge \textsc{Flag}(\hat{P}^{i'}_{j'}) \in \{\boxed{\mathrm{u}}, \boxed{\mathrm{f}}\}\right\}\right|, \tag{44}$$

$$\hat{\mathrm{U}}^{\boxed{\mathrm{f}}}_{\mathrm{prev}}(P,i,j) := \left|\left\{P^{i'}_{j'} : (i',j') \prec (i,j) \wedge \textsc{Flag}(\hat{P}^{i'}_{j'}) = \boxed{\mathrm{f}}\right\}\right|. \tag{45}$$

Note that all of the above quantities (42, 43, 44, 45) are bounded by

$$1 \le \bar{\mathrm{U}}_{\mathrm{prev}}(P,i,j), \hat{\mathrm{U}}_{\mathrm{prev}}(P,i,j), \bar{\mathrm{U}}^{\boxed{\mathrm{f}}}_{\mathrm{prev}}(P,i,j), \hat{\mathrm{U}}^{\boxed{\mathrm{f}}}_{\mathrm{prev}}(P,i,j) \le \bar{u}_{\mathrm{abs}} + \bar{u}_{\mathrm{squ}} \le \kappa. \tag{46}$$

In the algorithm we also use N-Possibilities 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}. More details on that are provided in the full version [12].

Thanks to the additional details we get the precise form of the expression $\mathbf{p}$.

5.4 Final expression

In the previous subsections we formalized algorithms that help us analyze the expression in Eq. (28). First we introduced Flag-Assign that analyzes ∇-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 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 p-∇-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

$$\begin{aligned}\mathbf{p}(|\mathcal{C}|^{-1}) = & \sum_{\nabla\text{-c} \in \nabla\text{-C}(\mathbf{M},\mathbf{Z})} \\ & \prod_{(i,j)=(1,1)}^{(q,k_q-1)} \mathbb{P}\left[\left(\boldsymbol{\varphi}_{\mathbf{f}}(S^i_{j\oplus}) = S^i_{j+1}\right) \middle| \bigwedge_{(i',j')\prec(i,j)} \left(\boldsymbol{\varphi}_{\mathbf{f}}(S^{i'}_{j'\oplus}) = S^{i'}_{j'+1}\right)\right] \\ = & \underbrace{\sum_{\text{p-}\nabla\text{-cf} \in \text{p-}\nabla\text{-CF}(\mathbf{M},\mathbf{Z})}}_{\text{Eq. (49),(50)}}\end{aligned} \tag{47}$$

$$\underbrace{\prod_{(i,j)=(1,1)}^{(q,k_q-1)} \mathbb{P}\left[\left(\boldsymbol{\varphi}_{\mathbf{f}}(S^i_{j\oplus}) = S^i_{j+1}\right) \middle| \bigwedge_{(i',j')\prec(i,j)} \left(\boldsymbol{\varphi}_{\mathbf{f}}(S^{i'}_{j'\oplus}) = S^{i'}_{j'+1}\right)\right]}_{\text{Eq. (37)}}, \tag{48}$$

where the second equality comes from the fact that constraints (32) exclude those ∇-c that have probability 0. Let us also make the division of p-∇-CF$(\mathbf{M},\mathbf{Z})$ explicit

$$\text{p-}\nabla\text{-CF}(\mathbf{M},\mathbf{Z}) = \bigcup_{\bar{u}_{\text{abs}}=1}^{\mu} \bigcup_{u_{\text{abs}}=0}^{\mu} \bigcup_{\bar{u}_{\text{squ}}=0}^{\zeta} \bigcup_{u_{\text{squ}}=0}^{\zeta} \bigcup_{P\in\mathcal{P}^*(\mathbf{M},\mathbf{Z},\bar{u}_{\text{abs}},u_{\text{abs}},\bar{u}_{\text{squ}},u_{\text{squ}})} \text{p-}\nabla\text{-CF}(\mathbf{M},\mathbf{Z},P). \tag{49}$$

Next we use Eq. (37) 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

$$|\text{p-}\nabla\text{-CF}(\mathbf{M},\mathbf{Z},P)| = \textsc{Calc}(P) \tag{50}$$

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

$$\mathbf{p}(|\mathcal{C}|^{-1}) = \sum_{\bar{u}_{\text{abs}},u_{\text{abs}},\bar{u}_{\text{squ}},u_{\text{squ}},P} \textsc{Calc}(P)\left(\frac{1}{2^r\cdot|\mathcal{C}|}\right)^{\bar{u}_{\text{abs}}+\bar{u}_{\text{squ}}} \tag{51}$$

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 $\boxed{\text{u}}$ in both outer and inner part) sates:

$$\prod_{i=1}^{q}\prod_{j=1}^{|\mathbf{M}^i|_r-1} (2^r - jq - i)\,(|\mathcal{C}| - jq - i)$$
$$\prod_{i=1}^{q}\prod_{j=|\mathbf{M}^i|_r}^{k_i-2} (|\mathcal{C}| - jq - i)\prod_{i=1}^{q}\left(2^{r|\mathbf{Z}^i|_r-\ell_i}|\mathcal{C}|\right)(2^r|\mathcal{C}|)^{-\kappa}. \tag{52}$$

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

$$\eta := q(m+z-2). \tag{53}$$

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.

In our proof we have focused on the case of $\boldsymbol{\varphi}_{\mathbf{f}}$ being a random transformation. In Section 6 we provide the details that should be considered to show that Theorem 8 holds also for random permutations.

6 Internal Permutations

In this section we prove the main result but for the internal function $\boldsymbol{\varphi}_{\mathbf{f}}$ being a random permutation. We use Zhandry's PRF/PRP switching lemma from [25]. In the full version of the paper [12], we also give a direct proof, resulting in a slightly worse bound.

Theorem 16. $\textsc{SpGen}_{\boldsymbol{\varphi}_{\mathbf{f}}}$ *for a random permutation* $\boldsymbol{\varphi}_{\mathbf{f}}$ *is quantumly indistinguishable from a random oracle. More concretely, for all quantum algorithms* A *making at most* q *quantum queries to* $\textsc{SpGen}$*, 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,*

$$\left| \underset{\boldsymbol{\varphi}_{\mathbf{f}} \overset{\$}{\leftarrow} \mathcal{T}(\mathcal{S})}{\mathbb{P}} \left[\mathrm{A}^{|\textsc{SpGen}_{\boldsymbol{\varphi}_{\mathbf{f}}}\rangle}(.) = 1 \right] - \underset{\mathbf{h} \leftarrow \mathfrak{R}}{\mathbb{P}} \left[\mathrm{A}^{|\mathbf{h}\rangle}(.) = 1 \right] \right| < \frac{\pi^2}{3} \eta^3 |\mathcal{C}|^{-1}, \tag{54}$$

where the set of permutations is denoted by $\mathcal{T}(\mathcal{S}) := \{\boldsymbol{\varphi}_{\mathbf{f}} : \mathcal{S} \to \mathcal{S} \mid \boldsymbol{\varphi}_{\mathbf{f}} \text{ is a bijection}\}$*. 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 [25] 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 $\textsc{SpGen}$ using a random permutation from $\textsc{SpGen}$ 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

$$\begin{aligned}
&\left| \underset{\boldsymbol{\varphi}_{\mathbf{f}} \xleftarrow{\$} \mathcal{T}(\mathcal{S})}{\mathbb{P}} \left[\mathrm{A}^{|\textsc{SpGen}_{\boldsymbol{\varphi}_{\mathbf{f}}}\rangle}(.) = 1 \right] - \underset{\mathbf{h} \leftarrow \mathfrak{R}}{\mathbb{P}} \left[\mathrm{A}^{|\mathbf{h}\rangle}(.) = 1 \right] \right| \\
&\qquad \leq \left| \underset{\boldsymbol{\varphi}_{\mathbf{f}} \xleftarrow{\$} \mathcal{T}(\mathcal{S})}{\mathbb{P}} \left[\mathrm{A}^{|\textsc{SpGen}_{\boldsymbol{\varphi}_{\mathbf{f}}}\rangle}(.) = 1 \right] - \underset{\boldsymbol{\varphi}_{\mathbf{f}} \xleftarrow{\$} \mathcal{S}^{\mathcal{S}}}{\mathbb{P}} \left[\mathrm{A}^{|\textsc{SpGen}_{\boldsymbol{\varphi}_{\mathbf{f}}}\rangle}(.) = 1 \right] \right| \\
&\qquad\qquad + \left| \underset{\boldsymbol{\varphi}_{\mathbf{f}} \xleftarrow{\$} \mathcal{S}^{\mathcal{S}}}{\mathbb{P}} \left[\mathrm{A}^{|\textsc{SpGen}_{\boldsymbol{\varphi}_{\mathbf{f}}}\rangle}(.) = 1 \right] - \underset{\mathbf{h} \leftarrow \mathfrak{R}}{\mathbb{P}} \left[\mathrm{A}^{|\mathbf{h}\rangle}(.) = 1 \right] \right| && (55) \\
&\qquad \leq \left| \underset{\boldsymbol{\varphi}_{\mathbf{f}} \xleftarrow{\$} \mathcal{T}(\mathcal{S})}{\mathbb{P}} \left[\mathrm{B}^{|\boldsymbol{\varphi}_{\mathbf{f}}\rangle}(.) = 1 \right] - \underset{\phi \xleftarrow{\$} \mathcal{S}^{\mathcal{S}}}{\mathbb{P}} \left[\mathrm{B}^{|\boldsymbol{\varphi}_{\mathbf{f}}\rangle}(.) = 1 \right] \right| + \frac{\pi^2}{6} \eta^3 |\mathcal{C}|^{-1} && (56) \\
&\qquad \leq \frac{\pi^2}{3} \eta^3 |\mathcal{C}|^{-1}. && (57)
\end{aligned}$$

□

7 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, 27] 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+ [23] in the instantiation using the Haraka hash function [15].

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).

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. Berlin, Heidelberg: Springer, 2008, pp. 181–197. 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*. Springer, 2011, pp. 41–69. 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] Jan Czajkowski, Leon Groot Bruinderink, Andreas Hülsing, Christian Schaffner, and Dominique Unruh. "Post-quantum Security of the Sponge Construction". In: *Post-Quantum Cryptography*. Springer International Publishing, 2018, pp. 185–204. DOI: `10.1007/978-3-319-79063-3_9`.

[12] Jan Czajkowski, Andreas Hülsing, and Christian Schaffner. *Quantum Indistinguishability of Random Sponges*. Cryptology ePrint Archive, Report 2019/069. 2019. URL: `https://eprint.iacr.org/2019/069`.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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

[19] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. 10th anniversary. Cambridge: Cambridge University Press, 2010.

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

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

[22] 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.

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

[24] Mark Zhandry. “How to Construct Quantum Random Functions”. In: *FOCS 2013*. IEEE Computer Society, 2012, pp. 679–687. DOI: `10.1109/FOCS.2012.37`.

[25] Mark Zhandry. “A note on the quantum collision and set equality problems”. In: *Quantum Information & Computation* 15.7&8 (2015), pp. 557–567.

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

[27] 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.