Security of Truncated Permutation Without Initial Value

Lorenzo Grassi and Bart Mennink

Radboud University, Nijmegen, The Netherlands
l.grassi@cs.ru.nl, b.mennink@cs.ru.nl

Abstract. Indifferentiability is a powerful notion in cryptography. If a construction is proven to be indifferentiable from an ideal object, it can under certain assumptions instantiate that ideal object in higher-level constructions. Indifferentiability is a particularly useful model for cryptographic hash functions, and myriad results are known proving that a hash function behaves like a random oracle under the assumption that the underlying primitive (typically a compression function, a block cipher, or a permutation) is random. Recently, advances have been made in proving indifferentiability of one-way functions with fixed input length. One such example is truncation of a permutation. If one evaluates a random permutation on an input value concatenated with a fixed initial value, and truncates the output, one obtains a construction that is indifferentiable from a random function up to a certain bound (Dodis et al., FSE 2009; Choi et al., ASIACRYPT 2019). Security of this construction, however, is in part determined by the length of the initial value; omission of this fixed value yields an insecure construction.
In this paper, we reconsider truncation of a permutation, and prove that the construction is indifferentiable from a random oracle, even if this fixed initial value is replaced by a randomized value. This randomized value may be the same for different evaluations of the construction, or freshly generated, up to the discretion of the adversary. The security level is the same as that of truncation with fixed initial value, up to collisions in the randomized value.
We show that our construction has immediate implications in the context of parallel variable-length digest generation. In detail, we describe Cascade-MGF, that operates on top of any cryptographic hash function and uses the hash function output as randomized initial value in truncation. We demonstrate that Cascade-MGF compares favorably over earlier parallel variable-length digest generation constructions, namely Counter-MGF and Chained-MGF, in almost all settings.

Keywords: Random permutation – Truncation – Indifferentiability – MGF – Digest generation

1 Introduction

A cryptographic hash function is a one-way function that maps data of arbitrary size to an output of a fixed size. Cryptographic hash functions are amongst the

most-studied and most-used cryptographic functions. They are used to provide integrity and authenticity in a large number of applications and protocols, including digital signatures, message authentication codes (MACs), and other forms of authentication.

The first hash functions appeared in the 70s, when Rabin introduced his iterative hash function design [40] and Merkle his ideas on tree hashing [36]. The iterative Merkle-Damgård construction, independently described by Damgård and Merkle [17, 35], later became the predominant approach in hash function design. Given a compression function $\mathcal{F}$ that maps $2n$ bits to n bits, the construction first pads and splits an arbitrarily sized input $M \in \{0,1\}^*$ injectively into n-bit blocks $M_0, M_1, \ldots, M_\mu$. The hash value is obtained by compressing these blocks one-by-one into an n-bit state:

$$h_{i+1} = \mathcal{F}(h_i, M_i) \quad \text{for } i = 0, \ldots, \mu\,, \tag{1}$$

where $h_0 = IV \in \{0,1\}^n$ is an initial value. Classical hash functions, including SHA-1, SHA-2, and MD5, are of this form.

In more recent years, the approach of permutation based hashing has gained popularity, mainly due to the rise of the sponge hash function construction [4,6], that is (among others) used as mode in the SHA-3 construction Keccak [8]. The sponge construction accommodates for both arbitrarily sized inputs and arbitrarily sized outputs. Let $\mathcal{P}$ be a permutation over $\{0,1\}^b$, and let $b = r + c$, where c denotes the capacity and r the rate. As before, an input message $M \in \{0,1\}^*$ is first injectively padded and split into r-bit blocks $M_0, M_1, \ldots, M_\mu$. Then, the message blocks are compressed one-by-one into a b-bit state:

$$h_{i+1} = \mathcal{P}(h_i \oplus (M_i \| 0^c)) \quad \text{for } i = 0, \ldots, \mu\,, \tag{2}$$

where $h_0 = IV \in \{0,1\}^b$ is an initial value. Let $h_{i+1} = \mathcal{P}(h_i)$ for $i \geq \mu + 1$. After the absorption of the last message block, the output is of the form

$$\mathrm{left}_r(h_{\mu+1}) \parallel \mathrm{left}_r(h_{\mu+2}) \parallel \mathrm{left}_r(h_{\mu+3}) \parallel \cdots\,,$$

where $\mathrm{left}_r(\cdot)$ denotes the r leftmost bits of its input.

All of these constructions have faced extensive security analysis. Whereas originally the focus was collision resistance, preimage resistance, and second preimage resistance, the current trend is to argue that a hash function construction is secure in the indifferentiability model, described and recalled in detail in Section 2.1. This model, introduced by Maurer et al. [29] and tailored to hash functions by Coron et al. [13], considers a security game where an adversary has access to either the hash function construction and an idealized primitive, or it has access to a random oracle and a simulator with the same interface as the hash function primitive. The goal of the simulator is to "mimic" the behavior of the idealized primitive so that any transcript an adversary has from communication with the random oracle and the simulator is hard to distinguish from a transcript that it may obtain from the actual construction and the idealized primitive. Although the plain Merkle-Damgård construction was *not* indifferentiable (see Coron et al. [13]), several variations of it have been proven to be

indifferentiable up to around $2^{n/2}$ queries [11, 13]. Likewise, Bertoni et al. [6] proved that the sponge construction (based on a random function or a permutation) is indifferentiable from a random oracle up to around $2^{c/2}$ queries.

The Merkle-Damgård indifferentiability result was proven under the assumption that the underlying function $\mathcal{F}$ is a random compression function. In practice, however, compression functions are built from invertible primitives such as block ciphers or permutations, e.g., the PGV compression functions [10,39]. Such compression functions are often easy to differentiate from random, which makes the aforementioned indifferentiable result futile. Instead, the research community had to resort to proving indifferentiability of Merkle-Damgård constructions based on a block cipher directly [13].

1.1 State of the Art on Compression Function Design

Having said that, there *do* exist compression functions based on block ciphers or permutations that are indifferentiable from a random function, up to a proper bound, and that can be used to instantiate $\mathcal{F}$ in the Merkle-Damgård construction. Two notable block cipher based examples are a double block length compression function of Mennink [30, 31] and the compression function used in BLAKE2 [2, 27]. Both constructions achieve indifferentiability by operating on an internal state that is larger than the block size of the compression function.

A notable permutation based example is the compression function used in the MD6 hash function [41]. Given $\mathcal{P}$ a permutation over $\{0,1\}^b$, the compression function consists of truncating (TRUNC) the output of the permutation:

$$\text{TRUNC}(I) = \text{left}_n\left(\mathcal{P}(IV \| I)\right) , \tag{3}$$

where $IV \in \{0,1\}^m$ is an initial value, $I \in \{0,1\}^{b-m}$ is the input, and where $\text{left}_n(\cdot)$ returns the n leftmost bits (where $n < b$). Dodis et al. [18] proved that this compression function is hard to distinguish from random. Choi et al. [12] recently derived an improved bound and proved that (3) is indifferentiable from a random function up to around $\min\left\{2^{\frac{2b-n}{3}}, \frac{2^{b-n}}{b-n}, 2^m\right\}$ queries.

Note that the TRUNC construction is not a compression function in the strict sense of the word: its input may be larger or smaller than its output, or they may be of the same size, depending on the parameter choice, and it should rather be named a one-way function. Another well-known permutation based one-way function design that is proven to be indifferentiable from a random function is the sum of independent permutations (SOP) construction, that operates based on two permutations $\mathcal{P}_1$ and $\mathcal{P}_2$ over $\{0,1\}^b$:

$$\text{SOP}(I) = \mathcal{P}_1(I) \oplus \mathcal{P}_2(I) , \tag{4}$$

where $I \in \{0,1\}^b$ is the input. The earliest analysis of a construction of this kind (in which the random permutations are publicly available to the adversary) is by Mandal et al. [28], who proved $2b/3$-bit security. The proof turned out to

have a subtle but non-negligible flaw which has been fixed by Mennink and Preneel [32]. Later, Lee [26] proved improved security for the general construction, and Bhattacharya and Nandi [9] improved all these known bounds and proved (full) b-bit indifferentiability of the sum of $k \geq 2$ independent permutations.

1.2 Improving Truncation

Common to all aforementioned compression function constructions is that they operate on an increased internal state and/or make multiple primitive calls to achieve indifferentiability. In addition, TRUNC has the property of additionally taking an initial value $IV \in \{0,1\}^m$ as input. This fixed initial value is, in fact, crucial for the indifferentiability proof: if omitted, the TRUNC construction (3) can be easily distinguished from random. (To wit, also the proven security bound becomes void for $m = 0$.)

Still, intuitively, the initial value is overkill. To see this, assume for the sake of example that we *drop* the initial value. In other words, we consider TRUNC of (3) with $m = 0$. If we set $n = c$, we obtain a compression function from b to c bits, and we can use it in the Merkle-Damgård construction (1) with state c and message block size $b - c$. The resulting construction is very similar to the sponge construction (2), the only difference is that message blocks are not added to the outer part but rather substituted, and this construction is known to be secure [6]. (This is known as the Grindahl construction [25].) Bottom line is that this initial value in TRUNC helps us in proving indifferentiability of the compression function and making it possible to instantiate the Merkle-Damgård construction with TRUNC. On the other hand, it is overkill in the sense that it is not strictly necessary for guaranteeing the security of the hashing scheme (equivalently, its omission does not make the resulting hashing scheme insecure).

1.3 Truncation Without Fixed IV

In this work, we take a closer look at the TRUNC construction, and particularly the role of the initial value IV. Concretely, we consider TRUNC of (3) where the fixed initial value IV is replaced by a random value. The adversary may choose to evaluate TRUNC multiple times for the same random value, it may choose to evaluate TRUNC for a different random value each query, and it may learn all random values used. We prove that this construction is still indifferentiable from a random oracle up to a comparable bound: the only difference occurs in the event of collisions in the random initial values.

More detailed, our proof incorporates an additional random oracle $\mathcal{H}$, and queries that the adversary makes to the TRUNC construction do not just consist of an input value $I \in \{0,1\}^{b-m}$, but also include a message M. The IV is subsequently replaced by $\mathcal{H}(M)$. Formally, for a hash function $\mathcal{H}$ with range $\{0,1\}^m$ and a permutation $\mathcal{P}$ over $\{0,1\}^b$, our construction RTRUNC is defined as

$$\mathrm{RTRUNC}(M, I) = \mathrm{left}_n\left(\mathcal{P}(\mathcal{H}(M)\|I)\right) . \tag{5}$$

Under the assumption that $\mathcal{H}$ is a random oracle and $\mathcal{P}$ is a random permutation, we prove that the RTRUNC construction is indifferentiable from a random oracle up to around

$$\min\left\{2^{\frac{2b-n}{3}}, 2^{\frac{m}{2}}, \frac{2^{b-n}}{b-n}\right\} \approx \min\left\{2^{\frac{2b-n}{3}}, 2^{\frac{m}{2}}, 2^{b-n}\right\} \tag{6}$$

queries. The proof is given in Section 4. It is inspired by the proof of Choi et al. for their isolated compression function, but deals with the complicating factor that comes with the presence of the hash function outputs.

1.4 Application: Parallel Digest Generation

Despite the fact that the applicability of earlier compression function indifferentiability results was negligible, mostly due to its expensive internal operation to make the indifferentiability proof work, our construction has direct practical applications. The main application, in fact, immediately comes from the proof approach: RTRUNC allows to extend *any* cryptographic hash function $\mathcal{H}$ into a parallel eXtendable Output Function (XOF) [38] that generates arbitrarily sized outputs. Note that, indeed, the Merkle-Damgård construction only outputs a fixed sized digest; the sponge construction does allow for arbitrarily sized outputs, but this output generation is inherently sequential. Parallel digest generation can lead to a significant speed-up in certain implementations, and would particularly be relevant in use with parallel tree hashing.

The quest for parallel digest generation is not new. Indeed, evaluating several permutations simultaneously in modern CPUs is faster than evaluating them in sequence. Hence, it is desirable to have schemes that can be efficiently parallelized. In the case of PRFs, this goal has been achieved by, e.g., the Farfalle construction proposed by Bertoni et al. [5] at ToSC 2017. For hashing, there already exist several constructions for achieving such a goal, which are commonly called "Mask Generation Functions" (MGFs).

Counter-MGF. To the best of our knowledge, the first construction – denoted as MGF1 ("Mask Generating Function 1") – was introduced by Kaliski and Staddon [24] in 1998 for use in public key cryptography. The majority of existing MGFs [1, 22, 23, 42] follow the counter-based design and have been standardized by ANSI, IEEE, and ISO/IEC. Focusing on the MGF1 construction, it is built on top of a hash function $\mathcal{H} : \{0,1\}^* \to \{0,1\}^m$. It takes as input an arbitrarily sized message M, glues a counter $\langle i \rangle_l$ of fixed size l to it, and outputs

$$\mathcal{H}(M \| \langle 0 \rangle_l) \parallel \mathcal{H}(M \| \langle 1 \rangle_l) \parallel \mathcal{H}(M \| \langle 2 \rangle_l) \parallel \cdots . \tag{7}$$

This construction is also known as Counter-MGF. Its indifferentiability from a (variable output length) random oracle follows from the indifferentiability of $\mathcal{H}$ from a (fixed output length) random oracle. See also Suzuki and Yasuda [43].

Chained-MGF. A comparable approach is Chained-MGF [37]. This mask generation function is built on top of a hash function $\mathcal{H} : \{0,1\}^* \to \{0,1\}^m$ and a one-way function $\mathcal{F} : \{0,1\}^{m+l} \to \{0,1\}^n$. It evaluates $\mathcal{H}$ on the message M, glues a counter $\langle i \rangle_l$ of fixed size l to this digest, and outputs

$$\mathcal{F}(\mathcal{H}(M) \| \langle 0 \rangle_l) \parallel \mathcal{F}(\mathcal{H}(M) \| \langle 1 \rangle_l) \parallel \mathcal{F}(\mathcal{H}(M) \| \langle 2 \rangle_l) \parallel \cdots . \tag{8}$$

This construction is proven to be indifferentiable from a random oracle up to the random oracle security of $\mathcal{H}$ and $\mathcal{F}$ by Suzuki and Yasuda [43]. Note, in particular, that security is capped by $2^{m/2}$ because security breaks in case one finds collisions in the output of $\mathcal{H}$.

Clearly, Chained-MGF has a disadvantage over Counter-MGF in that it uses two independent primitives. On the other hand, it *does* allow for more freedom in the choice of the finalizing one-way function $\mathcal{F}$, and the design is ignorant of the actual hash function in use. This is an advantage that will become clearer if we apply these modes to permutation-based hashing.

Cascade-MGF. The transition of our construction RTRUNC to a new MGF, which we dub Cascade-MGF, is immediate. For a hash function $\mathcal{H} : \{0,1\}^* \to \{0,1\}^m$ and a permutation $\mathcal{P} : \{0,1\}^b \to \{0,1\}^b$, it plainly evaluates RTRUNC on message M and counter $\langle i \rangle_l$ of fixed size $l = b - m$, and outputs

$$\begin{aligned} &\mathrm{RTRUNC}(M, \langle 0 \rangle_l) \parallel \cdots \parallel \mathrm{RTRUNC}(M, \langle i \rangle_l) \parallel \cdots \\ &= \mathrm{left}_n\left(\mathcal{P}(\mathcal{H}(M) \| \langle 0 \rangle_l)\right) \parallel \cdots \parallel \mathrm{left}_n\left(\mathcal{P}(\mathcal{H}(M) \| \langle i \rangle_l)\right) \parallel \cdots , \end{aligned} \tag{9}$$

where $\langle i \rangle_l \in \{0,1\}^l$ denotes the bit representation of $i \in \mathbb{Z}_{2^{b-m}}$. Note that Cascade-MGF of (9) is *exactly* equal to Chained-MGF of (8) instantiated with the construction (3) of Choi et al., *but with the IV removed*.

Comparison. In Section 5 we perform a generic comparison of Cascade-MGF with Counter-MGF and Chained-MGF with the focus on parallelizable permutation-based hashing and achieving k-bit security. It appears that Cascade-MGF compares favorably in most cases. An overview of this general comparison is given in Table 1. For the sake of exemplification, we now restrict our focus to $k = 128$-bit security with the use of a 384-bit permutation such as GIMLI [3] or Xoodoo [14]. In this case, the hash function $\mathcal{H}$ outputs digests of size $m = 256$ bits.

It appears that Cascade-MGF now achieves the exact same level of security as Chained-MGF instantiated with TRUNC (3). The only difference is that our construction does not use/need an initial value whereas Chained-MGF with TRUNC requires a $k = 128$-bit initial value. Due to this, our construction allows for a counter of size $l = b - m = 128$ bits whereas Chained-MGF with TRUNC allows for a counter of size $l = b - m - k = 0$ bits. As a concrete application, the squeezing phase in the recent tree-sponge construction proposed by Gunsing [20] can be instantiated via Cascade-MGF instead of Chained-MGF.

Chained-MGF instantiated with SOP (4) allows for the generation of more output blocks. Keeping the security parameters as in above choices, Chained-MGF

with SOP can generate 384-bit digests at a time as opposed to 256-bit digests in Cascade-MGF. The price to pay is that SOP makes two permutation evaluations per digest blocks instead of one, effectively making it less efficient.

The comparison of Cascade-MGF with Counter-MGF is less trivial. The reason is that Cascade-MGF is more versatile and is defined regardless of the hash function $\mathcal{H}$ in use, whereas Counter-MGF processes the counter $\langle i \rangle_l$ by the hash function. This means that a comparison between Cascade-MGF and Counter-MGF can only be made for a specific hash function choice. In Section 5.2, we instantiate both constructions with a minimal parallelizable permutation-based tree hash construction that is proven to be indifferentiable [15, 21], and argue that even in this case, Cascade-MGF has advantages. First of all, and more importantly, the cost in terms of the number function/permutation calls for generating an output of arbitrary length is independent of the size of the input message for Cascade-MGF, while it depends on it for Counter-MGF. In particular, we show that the cost in term of function/permutation calls of Counter-MGF is *at least double* with respect to the cost necessary for Counter-MGF, but such factor can even be much bigger depending on the size of the input message. Secondly, in Cascade-MGF one can take a smaller permutation for digest generation. On the downside, Counter-MGF would then be based on only one primitive whereas Cascade-MGF takes two. This can be remedied by instantiating the primitives using a single permutation with different round constants.

2 Preliminaries

Notation. For $m, n \in \mathbb{N}$, $\{0,1\}^n$ denotes the set of bit strings of length n. By $\{0,1\}^*$ we denote the set of arbitrarily sized strings, and by $\{0,1\}^\infty$ the set of infinitely long strings. We denote by $\mathrm{Hw}(x)$ the Hamming Weight of a binary string $x \in \{0,1\}^*$. We denote by $\mathrm{func}(m,n)$ the set of all functions from $\{0,1\}^m$ to $\{0,1\}^n$ and by $\mathrm{perm}(n)$ the set of permutations on $\{0,1\}^n$. Abusing notation, we denote by $\mathrm{func}(*,n)$ the set of all functions from $\{0,1\}^*$ to $\{0,1\}^n$. For a finite set $\mathfrak{X}$, $x \xleftarrow{\$} \mathfrak{X}$ denotes the uniform random sampling of an element x from $\mathfrak{X}$. The definition extends to $\mathrm{func}(*,n)$ by lazy sampling.

A random oracle $\mathcal{RO}$ gives access to a function that takes as input binary strings of arbitrary length and returns a random infinite string for each input, that is, $\mathcal{RO} : \{0,1\}^* \to \{0,1\}^\infty$. In a slightly more practical view, $\mathcal{RO}$ gets both a binary string and a length parameter $\ell \in \mathbb{N}$ as inputs, and it outputs a random string of length ℓ. If it is queried twice for the same message but for different length parameters $\ell, \ell' \in \mathbb{N}$, the shorter output is a substring of the longer one.

For $m, n \in \mathbb{N}$ with $m \geq n$, we denote by $\mathrm{left}_n : \{0,1\}^m \to \{0,1\}^n$ the function that outputs the leftmost n bits of the input. Likewise, $\mathrm{right}_n : \{0,1\}^m \to \{0,1\}^n$ outputs the rightmost n bits of the input. For a set $\mathfrak{X} \subseteq \{0,1\}^m$, we write $\mathrm{left}_n(\mathfrak{X}) = \{\mathrm{left}_n(x) \mid x \in \mathfrak{X}\}$, $\mathrm{right}_n(\mathfrak{X}) = \{\mathrm{right}_n(x) \mid x \in \mathfrak{X}\}$, and $\mathfrak{X} \| *^n = \{x \| y \in \{0,1\}^{m+n} \mid x \in \mathfrak{X} \wedge y \in \{0,1\}^n\}$.

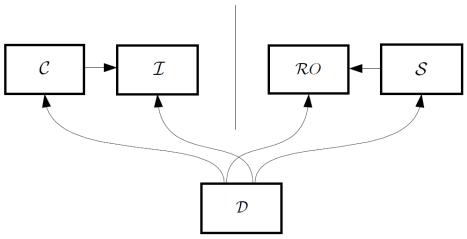


Fig. 1: The indifferentiability model.

2.1 Indifferentiability Model

Consider a hash function construction $\mathcal{C}$ built on top of an ideal component $\mathcal{I}$. To measure how good this hash function behaves like a random oracle $\mathcal{RO}$, we adopt the indifferentiability framework of Maurer et al. [29], and more precisely its version tailored to hash functions by Coron et al. [13]. In this framework, we consider a distinguisher $\mathcal{D}$. This distinguisher has access to either of two worlds: the real world $(\mathcal{C}[\mathcal{I}], \mathcal{I})$ and the simulated world $(\mathcal{RO}, \mathcal{S}[\mathcal{RO}])$, where $\mathcal{S}$ is a *simulator* that has the same interface as $\mathcal{I}$ and that has as goal to mimic its behavior in such a way that transcripts appearing in the ideal world are hard to distinguish from transcripts appearing in the real world. The goal of the distinguisher is to determine, for a given simulator $\mathcal{S}$, which world it is communicating with. If this is computationally hard, we say that $\mathcal{C}[\mathcal{I}]$ behaves like a random oracle, or simply that it is indifferentiable from a random oracle (up to a certain bound). Formally, we have the following definition.

Definition 1. *Let $\mathcal{C}$ by a cryptographic hash function with access to an ideal component $\mathcal{I}$. Let $\mathcal{RO}$ be a random oracle with the same interface as $\mathcal{C}$. We say that $\mathcal{C}$ is (Q, q, ε)-indifferentiable from $\mathcal{RO}$ if there exists a simulator $\mathcal{S}$ such that*

$$Adv_{\mathcal{C},\mathcal{S}}(\mathcal{D}) = \left|\Pr\left(\mathcal{D}^{\mathcal{C}[\mathcal{I}],\mathcal{I}} = 1\right) - \Pr\left(\mathcal{D}^{\mathcal{RO},\mathcal{S}[\mathcal{RO}]} = 1\right)\right| < \varepsilon\,,$$

for any distinguisher $\mathcal{D}$ making at most Q queries to the outer construction ($\mathcal{C}[\mathcal{I}]$ in the real world and $\mathcal{RO}$ in the simulated world) and at most q queries to the inner construction ($\mathcal{I}$ in the real world and $\mathcal{S}[\mathcal{RO}]$ in the simulated world).

The indifferentiability model is depicted in Figure 1.

In our work, we will consider hash functions $\mathcal{C}$ built on top of a set of components $\mathcal{I}$, namely a hash function $\mathcal{H}$ *and* a random permutation $\mathcal{P}$. In this case, $\mathcal{S}$ will also consist of two collaborating sub-simulators, and we split the inner complexity q into $q_{\mathcal{H}}$ and $q_{\mathcal{P}}$. Also, as we consider information-theoretic distinguishers and maximize over all of them, we will consider deterministic distinguishers only (without loss of generality).

2.2 χ^2 Method

At Crypto 2017, Dai et al. [16] introduced the *chi-squared method (χ^2 method)*, which can be exploited to obtain an upper bound on the statistical distance between two joint probability distributions.

Let $\mathcal{W}_0$ and $\mathcal{W}_1$ be two random systems over a sample space Ω. Let $\mathcal{D}$ be a deterministic distinguisher that makes ρ oracle queries to one of the two random systems. For each $j \in \{1, \ldots, \rho\}$, we denote by $Z_{\mathcal{W},j}$ the random variable over Ω that follows the distribution of the j-the answer obtained by $\mathcal{D}$ interacting with $\mathcal{W}$. Let

$$\boldsymbol{Z}_{\mathcal{W}}^{j} = (Z_{\mathcal{W},1}, Z_{\mathcal{W},2}, \ldots, Z_{\mathcal{W},j}) ,$$

and for each $\boldsymbol{z}_{j-1} = (z_1, z_2, \ldots, z_{j-1}) \in \Omega^{j-1}$, let

$$p_{\mathcal{W},j}^{\boldsymbol{z}_{j-1}}(z) = \Pr\left(Z_{\mathcal{W},j} = z \mid \boldsymbol{Z}_{\mathcal{W}}^{j-1} = \boldsymbol{z}_{j-1}\right) .$$

Assume that $\mathcal{W}_0$ and $\mathcal{W}_1$ are such that $p_{\mathcal{W}_0,j}^{\boldsymbol{z}_{j-1}}(z) > 0$ whenever $p_{\mathcal{W}_1,j}^{\boldsymbol{z}_{j-1}}(z) > 0$. Define, for any $j \in \{1, \ldots, \rho\}$ and any $\boldsymbol{z}_{j-1} = (z_1, z_2, \ldots, z_{j-1}) \in \Omega^{j-1}$,

$$\chi^2(\boldsymbol{z}_{j-1}) = \sum_{z \in \Omega \text{ such that } p_{\mathcal{W}_0,j}^{\boldsymbol{z}_{j-1}}(z) > 0} \frac{\left(p_{\mathcal{W}_0,j}^{\boldsymbol{z}_{j-1}}(z) - p_{\mathcal{W}_1,j}^{\boldsymbol{z}_{j-1}}(z)\right)^2}{p_{\mathcal{W}_0,j}^{\boldsymbol{z}_{j-1}}(z)} .$$

Dai et al. [16] proved that the distinguishing advantage between $\mathcal{W}_0$ and $\mathcal{W}_1$, denoted $|\boldsymbol{Z}_{\mathcal{W}_0} - \boldsymbol{Z}_{\mathcal{W}_1}|$, is upper bounded as follows:

$$|\boldsymbol{Z}_{\mathcal{W}_0} - \boldsymbol{Z}_{\mathcal{W}_1}| \leq \left(\frac{1}{2}\sum_{j=1}^{\rho} \mathrm{Ex}\left(\chi^2(\boldsymbol{z}_{j-1})\right)\right)^{\frac{1}{2}} .$$

3 The RTRUNC and Cascade-MGF Constructions

In this section, we describe the RTRUNC construction that we are going to analyze, as well as the Cascade-MGF hash function mode that can naturally be built on top of RTRUNC.

Let $b, m, n \in \mathbb{N}$ such that $m, n \leq b$, and let $l = b - m$. Let $\mathcal{H} \in \mathrm{func}(*, m)$ be a hash function and $\mathcal{P} \in \mathrm{perm}(b)$ a permutation. The function RTRUNC is defined as

$$\begin{aligned} \mathrm{RTRUNC} : \{0,1\}^* \times \{0,1\}^l &\to \{0,1\}^n , \\ (M, I) &\mapsto \mathrm{left}_n\left(\mathcal{P}(\mathcal{H}(M) \| I)\right) . \end{aligned} \tag{10}$$

As already informally explained in Section 1, this construction immediately yields an MGF, which we dub Cascade-MGF. This construction is built on the same primitives, namely a hash function $\mathcal{H} \in \mathrm{func}(*, m)$ and a permutation

$\mathcal{P} \in \mathrm{perm}(b)$, and it consists of concatenating multiple evaluations of RTRUNC for different inputs $I = \langle i \rangle_l$ for $i = 0, 1, 2, \ldots$:

$$\mathrm{RTRUNC}(M, \langle 0 \rangle_l) \parallel \mathrm{RTRUNC}(M, \langle 1 \rangle_l) \parallel \mathrm{RTRUNC}(M, \langle 2 \rangle_l) \parallel \cdots . \tag{11}$$

If we prove that the RTRUNC construction of (10) is indifferentiable from a fixed output length random oracle, then (11) is indifferentiable from a variable length random oracle. It thus suffices to prove the former, and this is the topic of next section.

Before proceeding with that proof, we admit that, even after concatenating outputs, (11) is not variable output length, as the counter can take at most 2^l values. Nevertheless, if l is large enough, for example if l is at least as much as the targeted security parameter, this is sufficient. Note that a similar limitation holds, e.g., for the sponge construction: in theory it can output an arbitrary amount of output blocks, but the security proof dictates that its security cannot be guaranteed once the permutation is evaluated more than $2^{c/2}$ times.

4 Indifferentiability of the RTRUNC Construction

In this section, we prove the indifferentiability of the RTRUNC construction.

Theorem 1. *Let $b, m, n \in \mathbb{N}$ such that $m, n \leq b$, and let $l = b - m$. Let $\mathcal{H} \xleftarrow{\$} \mathrm{func}(*, m)$ be a random hash function and $\mathcal{P} \xleftarrow{\$} \mathrm{perm}(b)$ a random permutation. Consider the* RTRUNC *construction of (10), which we denote by $\mathcal{C}$. Let $\mathcal{RO}$ be a random oracle with the same interface as $\mathcal{C}$. There exists a simulator $\mathcal{S}$, explicitly constructed in the proof, such that*

$$\begin{aligned} Adv_{\mathcal{C},\mathcal{S}}(\mathcal{D}) \leq & \frac{\binom{q_\mathcal{H}}{2} + 3 \cdot q_\mathcal{H} \cdot q_\mathcal{P}}{2^m} + \frac{Q \cdot q_\mathcal{P}}{2^{b-2}} + \frac{(3 \cdot \ln(Q) + 3n + 1) \cdot q_\mathcal{P}}{2^{b-n-1}} \\ & + \left(\frac{6 \cdot (Q + q_\mathcal{H} + q_\mathcal{P})^3}{2^{2b-n}} + \frac{3 \cdot (Q + q_\mathcal{H} + q_\mathcal{P})^2}{2^m} + \frac{5 \cdot (Q + q_\mathcal{H} + q_\mathcal{P})}{2^{b-n}} \right)^{\frac{1}{2}} \end{aligned} \tag{12}$$

for any distinguisher $\mathcal{D}$ making Q queries to the outer construction and $q_\mathcal{H}$ and $q_\mathcal{P}$ to the inner constructions, where $Q + q_\mathcal{H} + q_\mathcal{P} \leq 2^{m-1}$ and $1 + q_\mathcal{P} \leq 2^{b-n-1}$.

An interpretation of the security bound will be given in Section 5.

The first step of the proof will be to design a simulator $\mathcal{S}$. This will be done in Section 4.1. Note that, in fact, this simulator must simulate multiple functions: a hash function $\mathcal{S}_\mathcal{H}$ as well as the forward and inverse interfaces $\mathcal{S}_\mathcal{P}$ and $\mathcal{S}_\mathcal{P}^{-1}$. The next step is to bound the distance $\mathrm{Adv}_{\mathcal{C},\mathcal{S}}(\mathcal{D})$ of Definition 1 for the given simulator and for any computationally unbounded distinguisher that can make Q queries to the outer construction and $q_\mathcal{H}$ and $q_\mathcal{P}$ to the inner constructions. This is done in Section 4.2. This bounding itself relies on a triangle inequality with an intermediate world, by bounding the two distances from the real and from the simulated world to this intermediate world. These two bounds are derived in separate lemmas in Sections 4.3 and 4.4.

4.1 Simulator

The first step is to design a simulator $\mathcal{S}[\mathcal{RO}]$, which consists of three algorithms: $\mathcal{S}_{\mathcal{H}}$, $\mathcal{S}_{\mathcal{P}}$, and $\mathcal{S}_{\mathcal{P}}^{-1}$. These three algorithms are related, i.e., any algorithm might have access to the query history of another algorithm. In addition, they may query $\mathcal{RO}$. The goal is to design algorithms that are hard to distinguish from random functions $\mathcal{H}$, $\mathcal{P}$, and $\mathcal{P}^{-1}$, respectively, and that are consistent with the random oracle $\mathcal{RO}$.

Simulators. To store input-output tuples of $\mathcal{S}_{\mathcal{H}}, \mathcal{S}_{\mathcal{P}}$ and $\mathcal{S}_{\mathcal{P}}^{-1}$, we maintain the following initially empty sets $\mathfrak{C}_{\mathcal{H}}$ and $\mathfrak{C}_{\mathcal{P}}$:

$$\mathfrak{C}_{\mathcal{H}} = \{(M, x) \in \{0,1\}^* \times \{0,1\}^m \mid \mathcal{S}_{\mathcal{H}}(M) = x\}\,,$$
$$\mathfrak{C}_{\mathcal{P}} = \{(X, Y) \in \{0,1\}^b \times \{0,1\}^b \mid \mathcal{S}_{\mathcal{P}}(X) = Y \text{ and } \mathcal{S}_{\mathcal{P}}^{-1}(Y) = X\}\,.$$

We additionally define the domain and range values respectively of $\mathfrak{C}_{\mathcal{H}}$ and $\mathfrak{C}_{\mathcal{P}}$ as follows:

$$\mathfrak{D}_{\mathcal{H}} = \{M \in \{0,1\}^* \mid \exists x \in \{0,1\}^m \text{ such that } (M, x) \in \mathfrak{C}_{\mathcal{H}}\}\,,$$
$$\mathfrak{R}_{\mathcal{H}} = \{x \in \{0,1\}^m \mid \exists M \in \{0,1\}^* \text{ such that } (M, x) \in \mathfrak{C}_{\mathcal{H}}\}\,,$$
$$\mathfrak{D}_{\mathcal{P}} = \{X \in \{0,1\}^b \mid \exists Y \in \{0,1\}^b \text{ such that } (X, Y) \in \mathfrak{C}_{\mathcal{P}}\}\,,$$
$$\mathfrak{R}_{\mathcal{P}} = \{Y \in \{0,1\}^b \mid \exists X \in \{0,1\}^b \text{ such that } (X, Y) \in \mathfrak{C}_{\mathcal{P}}\}\,.$$

Moreover, for each $y \in \{0,1\}^n$, we define $\mathfrak{R}_{\mathcal{P}}^y$ as follows:

$$\mathfrak{R}_{\mathcal{P}}^y = \{y' \in \{0,1\}^{b-n} \mid y \| y' \in \mathfrak{R}_{\mathcal{P}}\}\,.$$

Likewise, to store the input-output tuples of $\mathcal{C}$, we maintain the following initially empty set $\mathfrak{C}_{\mathcal{C}}$:

$$\mathfrak{C}_{\mathcal{C}} = \{((M, I), y) \in \{0,1\}^* \times \{0,1\}^l \times \{0,1\}^n \mid \mathcal{C}(M, I) = y\}\,.$$

We additionally define the domain and range values of $\mathfrak{C}_{\mathcal{C}}$ as follows:

$$\mathfrak{D}_{\mathcal{C}} = \{(M, I) \in \{0,1\}^* \times \{0,1\}^l \mid \exists x \in \{0,1\}^n \text{ such that } ((M, I), y) \in \mathfrak{C}_{\mathcal{C}}\}\,,$$
$$\mathfrak{R}_{\mathcal{C}} = \{y \in \{0,1\}^n \mid \exists (M, I) \in \{0,1\}^* \times \{0,1\}^l \text{ such that } ((M, I), y) \in \mathfrak{C}_{\mathcal{C}}\}\,.$$

Note that the simulator has *no* access to $\mathfrak{C}_{\mathcal{C}}$; we will need it later to bound the indifferentiability advantage.

Based on this, simulator $\mathcal{S}_{\mathcal{H}}$ is now given in Algorithm 1, simulator $\mathcal{S}_{\mathcal{P}}$ in Algorithm 2, and simulator $\mathcal{S}_{\mathcal{P}}^{-1}$ in Algorithm 3.

Discussion. We briefly elaborate on some design choices of the simulator.

Regarding $\mathcal{S}_{\mathcal{H}}$, the output is chosen from the set $\{0,1\}^m \setminus (\mathfrak{R}_{\mathcal{H}} \cup \text{left}_m(\mathfrak{D}_{\mathcal{P}}))$. One of our goals is to avoid a collision at the output of $\mathcal{S}_{\mathcal{H}}$. The problem is not about collisions itself (note that a collision can also occur in the real world), but

Algorithm 1: Simulator $\mathcal{S}_{\mathcal{H}}$

Data: input $M \in \{0,1\}^*$
Result: output $x \in \{0,1\}^m$
if $M \in \mathfrak{D}_{\mathcal{H}}$ **then**
  return x such that $(M, x) \in \mathfrak{C}_{\mathcal{H}}$
$x \xleftarrow{\$} \{0,1\}^m \setminus (\mathfrak{R}_{\mathcal{H}} \cup \text{left}_m(\mathfrak{D}_{\mathcal{P}}))$
$\mathfrak{C}_{\mathcal{H}} \leftarrow \mathfrak{C}_{\mathcal{H}} \cup \{(M, x)\}$
return x

Algorithm 2: Simulator $\mathcal{S}_{\mathcal{P}}$

Data: input $X \in \{0,1\}^b$
Result: output $Y \in \{0,1\}^b$
if $X \in \mathfrak{D}_{\mathcal{P}}$ **then**
  return $Y \in \mathfrak{R}_{\mathcal{P}}$ such that $(X, Y) \in \mathfrak{C}_{\mathcal{P}}$
parse $X = x \| I$ where $x = \text{left}_m(X) \in \{0,1\}^m$
if $x \in \mathfrak{R}_{\mathcal{H}}$ **then**
  let $M \in \{0,1\}^*$ be such that $(M, x) \in \mathfrak{C}_{\mathcal{H}}$
  $y \leftarrow \mathcal{RO}(M, I)$
  $y' \xleftarrow{\$} \{0,1\}^{b-n} \setminus \mathfrak{R}^y_{\mathcal{P}}$
  $Y \leftarrow y \| y'$
else
  $Y \xleftarrow{\$} \{0,1\}^b \setminus \mathfrak{R}_{\mathcal{P}}$
$\mathfrak{C}_{\mathcal{P}} \leftarrow \mathfrak{C}_{\mathcal{P}} \cup \{(X, Y)\}$
return Y

Algorithm 3: Simulator $\mathcal{S}_{\mathcal{P}}^{-1}$

Data: input $Y \in \{0,1\}^b$
Result: output $X \in \{0,1\}^b$
if $Y \in \mathfrak{R}_{\mathcal{P}}$ **then**
  return $X \in \mathfrak{D}_{\mathcal{P}}$ such that $(X, Y) \in \mathfrak{C}_{\mathcal{P}}$
$X \xleftarrow{\$} \{0,1\}^b \setminus \left(\mathfrak{D}_{\mathcal{P}} \cup \mathfrak{R}_{\mathcal{H}} \| *^{b-m}\right)$
$\mathfrak{C}_{\mathcal{P}} \leftarrow \mathfrak{C}_{\mathcal{P}} \cup \{(X, Y)\}$
return X

rather about the fact that the two outputs of the overall construction $\mathcal{C}[\mathcal{H}, \mathcal{P}]$ would be equal, and this could be problematic when defining $\mathcal{S}_{\mathcal{P}}$.

In a similar way, we want to avoid that calls to $\mathcal{S}_{\mathcal{H}}$ create collisions between the outputs of $\mathcal{S}_{\mathcal{H}}$ and the leftmost m bits of the inputs to $\mathcal{S}_{\mathcal{P}}$. This type of collisions is also explicitly avoided in calls to $\mathcal{S}_{\mathcal{P}}^{-1}$. The problem of collisions between the outputs of $\mathcal{S}_{\mathcal{H}}$ and the inputs to $\mathcal{S}_{\mathcal{P}}$ is related to the consistency between the inner constructions and the outer one, in this case the random oracle $\mathcal{RO}$. For example, assume that $\mathcal{S}_{\mathcal{H}}(M)$ returns a value x that already belongs to $\text{left}_m(\mathfrak{D}_{\mathcal{P}})$, or in a similar way that $\mathcal{S}_{\mathcal{P}}^{-1}$ returns a value X such that

$\text{left}_m(X) \in \mathfrak{R}_{\mathcal{H}}$. Let $I \in \{0,1\}^l$ be such that $\text{right}_l(X) = I$. In this case, an attacker can check the consistency of these two simulator queries with $\mathcal{RO}$, by verifying if $\mathcal{RO}(M, I) = \text{left}_n(Y)$. This equality would hold with probability 1 in the real world.

Besides these two collisions, no other types of collisions are avoided, and the simulators $\mathcal{S}_{\mathcal{H}}$ and $\mathcal{S}_{\mathcal{P}}^{\pm}$ behave like a random hash function and permutation, respectively, with *one* difference: if the simulator $\mathcal{S}_{\mathcal{P}}$ is queried on an input X such that $\text{left}_m(X) \in \mathfrak{R}_{\mathcal{H}}$, the simulator *must* maintain oracle consistency, i.e., query its random oracle $\mathcal{RO}$ to generate a response to the query.

4.2 Proof of Theorem 1

Let $\mathcal{C}$ be the construction of (10) defined via a random hash function $\mathcal{H}$ and a random permutation $\mathcal{P}$. Let $\mathcal{RO}$ be a random oracle with the same interface as $\mathcal{C}$, and let $\mathcal{S}$ be the simulator of Section 4.1. Let $\mathcal{W}_{\text{S}} = (\mathcal{RO}, \mathcal{S}_{\mathcal{H}}[\mathcal{RO}], \mathcal{S}_{\mathcal{P}}^{\pm}[\mathcal{RO}])$ and $\mathcal{W}_{\text{R}} = (\mathcal{C}[\mathcal{H}, \mathcal{P}], \mathcal{H}, \mathcal{P})$ denote the simulated world and the real world, respectively. Let $\mathcal{D}$ be any distinguisher that makes at most Q construction queries (to $\mathcal{RO}$ or $\mathcal{C}[\mathcal{H}, \mathcal{P}]$), $q_{\mathcal{H}}$ queries to the first primitive oracle ($\mathcal{S}_{\mathcal{H}}[\mathcal{RO}]$ or $\mathcal{H}$) and $q_{\mathcal{P}}$ queries to the second primitive oracle ($\mathcal{S}_{\mathcal{P}}[\mathcal{RO}]$ or $\mathcal{P}$). Assume that $\mathcal{D}$ never makes redundant queries, i.e., query an oracle twice on the same input. From Definition 1, our goal is to bound

$$\text{Adv}_{\mathcal{C},\mathcal{S}}(\mathcal{D}) = \left|\Pr\left(\mathcal{D}^{\mathcal{W}_{\text{S}}} = 1\right) - \Pr\left(\mathcal{D}^{\mathcal{W}_{\text{R}}} = 1\right)\right| . \tag{13}$$

Additional World. First, we will consider the differences between the two worlds. Focusing on $\mathcal{S}_{\mathcal{H}}$ and $\mathcal{H}$, it is obvious that the former never results in collisions, but they might occur in the latter. This means that there exist communication transcripts that can occur in $\mathcal{W}_{\text{R}}$ but not in $\mathcal{W}_{\text{S}}$. At the same time, the random oracle $\mathcal{RO}$ in $\mathcal{W}_{\text{S}}$ outputs random strings, whereas in $\mathcal{W}_{\text{R}}$ the outputs of the function $\mathcal{C}[\mathcal{H}, \mathcal{P}]$ depend on the details of the function $\mathcal{H}$ and of the permutation $\mathcal{P}$. Hence, there exist communication transcripts that can occur in $\mathcal{W}_{\text{S}}$ but not in $\mathcal{W}_{\text{R}}$.

As our goal is to apply the chi-squared method, our first step is to introduce an intermediate world $\mathcal{W}_{\text{I}} = (\mathcal{C}[\mathcal{H}^\star, \mathcal{P}^\star], \mathcal{H}^\star, \mathcal{P}^\star)$, which has the same oracle interface as $\mathcal{W}_{\text{S}}$ and $\mathcal{W}_{\text{R}}$. The idea is that the world $\mathcal{W}_{\text{I}}$ behaves closely to $\mathcal{W}_{\text{R}}$, and that $\mathcal{W}_{\text{I}}$ is in the support of $\mathcal{W}_{\text{S}}$, and this world reminds of the intermediate world introduced by Choi et al. [12], though it is more involved as a hash function interface is added. These functions maintain initially empty sets to store input-output tuples $\mathfrak{C}_{\mathcal{H}}^\star, \mathfrak{C}_{\mathcal{P}}^\star$ and $\mathfrak{C}_{\mathcal{C}}^\star$, in a similar vein as in Section 4.1, with $\mathfrak{D}_{\mathcal{H}}^\star$, $\mathfrak{R}_{\mathcal{H}}^\star$, $\mathfrak{D}_{\mathcal{P}}^\star$, $\mathfrak{R}_{\mathcal{P}}^\star$, $\mathfrak{R}_{\mathcal{P}}^{\star\, y}$, $\mathfrak{D}_{\mathcal{C}}^\star$, and $\mathfrak{R}_{\mathcal{C}}^\star$ defined analogously as before. The algorithms $\mathcal{H}^\star$, $\mathcal{P}^\star$, and $\mathcal{P}^{\star\,-1}$ are now given in Algorithms 4, 5, 6, respectively.

In a nutshell, the world $\mathcal{W}_{\text{I}}$ operates as $\mathcal{W}_{\text{R}}$ but instantiates it with lazily-sampled primitives $\mathcal{H}^\star$ and $\mathcal{P}^\star$ that slightly deviate from $\mathcal{H}$ and $\mathcal{P}$. In particular, $\mathcal{H}^\star$ never samples any element in $\mathfrak{R}_{\mathcal{H}}^\star \cup \text{left}_m(\mathfrak{D}_{\mathcal{P}}^\star)$, and $\mathcal{P}^{\star\,-1}$ never samples any element of $\mathfrak{D}_{\mathcal{P}}^\star \cup (\mathfrak{R}_{\mathcal{H}}^\star \| *^{b-m})$. The world uses flags denoted by bad_1, bad_2, and bad_3 (all initialized as false), to mark events in which $\mathcal{W}_{\text{I}}$ differs from $\mathcal{W}_{\text{R}}$.

Algorithm 4: Procedure $\mathcal{H}^\star$ with appended bad events

Data: input $M \in \{0,1\}^*$
Result: output $x \in \{0,1\}^m$

$x \xleftarrow{\$} \{0,1\}^m$
if $x \in \mathfrak{R}^\star_{\mathcal{H}}$ ***or*** $x \in \mathrm{left}_m(\mathfrak{D}^\star_{\mathcal{P}})$ **then**
  if $x \in \mathfrak{R}^\star_{\mathcal{H}}$ **then**
    $\mathsf{bad}_1 \leftarrow$ true
  if $x \in \mathrm{left}_m(\mathfrak{D}^\star_{\mathcal{P}})$ **then**
    $\mathsf{bad}_2 \leftarrow$ true
  $x \xleftarrow{\$} \{0,1\}^m \setminus (\mathfrak{R}^\star_{\mathcal{H}} \cup \mathrm{left}_m(\mathfrak{D}^\star_{\mathcal{P}}))$
$\mathfrak{C}^\star_{\mathcal{H}} \leftarrow \mathfrak{C}^\star_{\mathcal{H}} \cup \{(M, x)\}$
return x

Algorithm 5: Procedure $\mathcal{P}^\star$ with appended bad events

Data: input $X \in \{0,1\}^b$
Result: output $Y \in \{0,1\}^b$

$Y \xleftarrow{\$} \{0,1\}^b \setminus \mathfrak{R}^\star_{\mathcal{P}}$
$\mathfrak{C}^\star_{\mathcal{P}} \leftarrow \mathfrak{C}^\star_{\mathcal{P}} \cup \{(X, Y)\}$
return Y

Algorithm 6: Procedure $\mathcal{P}^{\star\,-1}$ with appended bad events

Data: input $Y \in \{0,1\}^b$
Result: output $X \in \{0,1\}^b$

parse $Y = y \| y'$ where $y = \mathrm{left}_n(Y) \in \{0,1\}^n$
if $Y \notin \mathfrak{R}^\star_{\mathcal{P}}$ **then**
  $X \xleftarrow{\$} \{0,1\}^b \setminus \mathfrak{D}^\star_{\mathcal{P}}$
  if $\mathrm{left}_m(X) \in \mathfrak{R}^\star_{\mathcal{H}}$ **then**
    $\mathsf{bad}_2 \leftarrow$ true
    $X \xleftarrow{\$} \{0,1\}^b \setminus \left(\mathfrak{D}^\star_{\mathcal{P}} \cup \mathfrak{R}^\star_{\mathcal{H}} \| *^{b-m}\right)$
  $\mathfrak{C}^\star_{\mathcal{P}} \leftarrow \mathfrak{C}^\star_{\mathcal{P}} \cup \{(X, Y)\}$
else
  $\mathsf{bad}_3 \leftarrow$ true
  let $X' \in \{0,1\}^b$ be such that $(X', Y) \in \mathfrak{C}^\star_{\mathcal{P}}$
  $X \xleftarrow{\$} \{0,1\}^b \setminus \left(\mathfrak{D}^\star_{\mathcal{P}} \cup \mathfrak{R}^\star_{\mathcal{H}} \| *^{b-m}\right)$
  $y' \xleftarrow{\$} \{0,1\}^{b-n} \setminus \mathfrak{R}^{\star\,y}_{\mathcal{P}}$
  $Y' \leftarrow y \| y'$
  $\mathfrak{C}^\star_{\mathcal{P}} \leftarrow (\mathfrak{C}^\star_{\mathcal{P}} \setminus (\{X', Y\}) \cup \{(X, Y), (X', Y')\}$
return X

A significant change is in the oracle $\mathcal{P}^{\star\,-1}$, where it makes a distinction between whether or not $Y \notin \mathfrak{R}^\star_{\mathcal{P}}$. Recall that the adversary never evaluates repeated queries. This means that if $Y \in \mathfrak{R}^\star_{\mathcal{P}}$ holds, necessarily there had been

a query (M, I) to $\mathcal{C}(M, I)$ that returned $y = \text{left}_n(Y)$. In this case, the distinguisher did not obtain any information on the $b - n$ rightmost bits of Y, yet. The only thing it knows, is that there should be an evaluation

$$\mathcal{P}^\star(\hat{*}\|I) = y\|*$$

for unknown $\hat{*} \in \{0,1\}^n$ and $* \in \{0,1\}^{b-m}$. When this inverse query $P^{\star -1}(y\|*)$ is made later during the attack, the rightmost bits $\text{right}_{b-n}(y\|*)$ are replaced by a new element y', and $P^{\star -1}(y\|*)$ is also given a new element X outside $\mathfrak{D}^\star_\mathcal{P} \cup \mathfrak{R}^\star_\mathcal{H}\|*^{b-m}$.

Triangle Inequality. Using intermediate world $\mathcal{W}_\text{I}$, a triangle inequality yields for (13):

$$\begin{aligned}\text{Adv}_{\mathcal{C},\mathcal{S}}(\mathcal{D}) \leq \\ \left|\Pr\left(\mathcal{D}^{\mathcal{W}_\text{S}} = 1\right) - \Pr\left(\mathcal{D}^{\mathcal{W}_\text{I}} = 1\right)\right| + \left|\Pr\left(\mathcal{D}^{\mathcal{W}_\text{I}} = 1\right) - \Pr\left(\mathcal{D}^{\mathcal{W}_\text{R}} = 1\right)\right| .\end{aligned} \quad (14)$$

Bounds on the remaining two terms are derived separately. In Lemma 1, a bound on the distance between $\mathcal{W}_\text{I}$ and $\mathcal{W}_\text{R}$ is derived, and in Lemma 2, a bound on the distance between $\mathcal{W}_\text{S}$ and $\mathcal{W}_\text{I}$ is derived.

Lemma 1. *Let $\mathcal{W}_I = (\mathcal{C}[\mathcal{H}^\star, \mathcal{P}^\star], \mathcal{H}^\star, \mathcal{P}^\star)$ and $\mathcal{W}_R = (\mathcal{C}[\mathcal{H}, \mathcal{P}], \mathcal{H}, \mathcal{P})$ be respectively the intermediate and the real world, as defined before. Then,*

$$\begin{aligned}\left|\Pr\left(\mathcal{D}^{\mathcal{W}_I} = 1\right) - \Pr\left(\mathcal{D}^{\mathcal{W}_R} = 1\right)\right| \leq \frac{\binom{q_\mathcal{H}}{2} + 3 \cdot q_\mathcal{H} \cdot q_\mathcal{P}}{2^m} + \frac{Q \cdot q_\mathcal{P}}{2^{b-2}} \\ + \frac{(3 \cdot \ln(Q) + 3n + 1) \cdot q_\mathcal{P}}{2^{b-n-1}}\end{aligned}$$

for any distinguisher $\mathcal{D}$ making Q queries to the outer construction and $q_\mathcal{H}$ and $q_\mathcal{P}$ to the inner constructions, where $Q + q_\mathcal{H} + q_\mathcal{P} \leq 2^{b-1}$.

Lemma 2. *Let $\mathcal{W}_S = (\mathcal{RO}, \mathcal{S}_\mathcal{H}[\mathcal{RO}], \mathcal{S}^\pm_\mathcal{P}[\mathcal{RO}])$ and $\mathcal{W}_I = (\mathcal{C}[\mathcal{H}^\star, \mathcal{P}^\star], \mathcal{H}^\star, \mathcal{P}^\star)$ be respectively the simulated world and the intermediate world, as defined before.*

$$\begin{aligned}\left|\Pr\left(\mathcal{D}^{\mathcal{W}_S} = 1\right) - \Pr\left(\mathcal{D}^{\mathcal{W}_I} = 1\right)\right| \leq \left(\frac{6 \cdot (Q + q_\mathcal{H} + q_\mathcal{P})^3}{2^{2b-n}}\right. \\ \left. + \frac{3 \cdot (Q + q_\mathcal{H} + q_\mathcal{P})^2}{2^m} + \frac{5 \cdot (Q + q_\mathcal{H} + q_\mathcal{P})}{2^{b-n}}\right)^{\frac{1}{2}}\end{aligned}$$

for any distinguisher $\mathcal{D}$ making Q queries to the outer construction and $q_\mathcal{H}$ and $q_\mathcal{P}$ to the inner constructions, where $Q + q_\mathcal{H} + q_\mathcal{P} \leq 2^{m-1}$ and $1 + q_\mathcal{P} \leq 2^{b-n-1}$.

The proof of Lemma 1 is given in Section 4.3, and consists of bounding the probability that a bad event occurs in $\mathcal{W}_\text{I}$. The proof of Lemma 2 is given in Section 4.4, and is based on the chi-squared method. The proof of Theorem 1 is immediately completed by plugging the bounds of Lemmas 1 and 2 into (14).

Additional Notation. For world $\mathcal{W}_{\mathrm{I}}$, and specifically for evaluations of $\mathcal{P}^\star$ and $\mathcal{P}^{\star -1}$, we introduce the following additional notation. Consider any query $\mathcal{P}^{\star -1}(Y)$ with $y = \mathrm{left}_n(Y)$. At the point of making this query,

- V_y counts the number of elements X where $\mathcal{P}^\star(X)$ has been determined by a function query or a distinguisher query such that $\mathcal{P}^\star(X) = y \| y'$ for some $y' \in \{0,1\}^{b-n}$:

$$V_y = |\{X \in \{0,1\}^b \mid \exists y' \in \{0,1\}^{b-n} \text{ such that } \mathcal{P}^\star(X) = y \| y'\}| \,;$$

- S_y counts the number of elements X where $\mathcal{P}^\star(X)$ has been determined *only* by a distinguisher query such that $\mathcal{P}^\star(X) = y \| y'$ for some $y' \in \{0,1\}^{b-n}$;
- F_y counts the number of elements X where $\mathcal{P}^\star(X)$ has been partially determined only by a function query and $\mathcal{P}^\star(X) = y \| \star$ for some unknown $\star \in \{0,1\}^{b-n}$, in such a way that $F_y = V_y - S_y$.

Finally, let $V = \sum_{y \in \{0,1\}^n} V_y$. At any point in time, $V = |\mathfrak{C}^\star_{\mathcal{P}}|$.

4.3 Upper Bound on Distance Between $\mathcal{W}_{\mathrm{I}}$ and $\mathcal{W}_{\mathrm{R}}$ (Lemma 1)

The two worlds behave in the same way until one of the bad events is set to true in $\mathcal{W}_{\mathrm{I}}$. Hence, we can upper bound this term by computing the probability that one of the bad events is set to true:

$$\begin{aligned}\left|\Pr\left(\mathcal{D}^{\mathcal{W}_{\mathrm{I}}} = 1\right) - \Pr\left(\mathcal{D}^{\mathcal{W}_{\mathrm{R}}} = 1\right)\right| &\leq \Pr(\mathsf{bad}_1 \cup \mathsf{bad}_2 \cup \mathsf{bad}_3) \\ &\leq \Pr(\mathsf{bad}_1) + \Pr(\mathsf{bad}_2) + \Pr(\mathsf{bad}_3)\,.\end{aligned}$$

In the following, we compute $\Pr(\mathsf{bad}_i)$ for each $i \in \{1,2,3\}$. The proof of the lemma is then completed by a simple addition of the three terms.

Probability of bad_1. This event happens if $\mathcal{H}^\star$ returns a value x that already belongs to $\mathfrak{R}^\star_{\mathcal{H}}$, or equivalently, a value x for which there exists an earlier M' such that $(M', x) \in \mathfrak{C}^\star_{\mathcal{H}}$. As the distinguisher makes $q_{\mathcal{H}}$ queries to $\mathcal{H}^\star$, the probability that this occurs is bounded as follows:

$$\Pr(\mathsf{bad}_1) \leq \sum_{j=1}^{q_{\mathcal{H}}} \frac{j-1}{2^m} \leq \frac{\binom{q_{\mathcal{H}}}{2}}{2^m}\,.$$

Probability of bad_2. This event happens if $\mathcal{H}^\star$ returns a value x that already belongs to $\mathrm{left}_m(\mathfrak{D}^\star_{\mathcal{P}})$ or if $\mathcal{P}^{\star -1}$ returns a value X such that $\mathrm{left}_m(X) \in \mathfrak{R}^\star_{\mathcal{H}}$. In a query to $\mathcal{H}^\star$, bad_2 is set with probability at most $q_{\mathcal{P}}/2^m$. In a query to $\mathcal{P}^{\star -1}$, bad_2 is set with probability at most $(q_{\mathcal{H}} \cdot 2^{b-m})/(2^b - q_{\mathcal{P}}) \leq 2q_{\mathcal{H}}/2^m$ (provided that $q_{\mathcal{P}} \leq 2^{b-1}$). After $q_{\mathcal{H}}$ queries to $\mathcal{H}^\star$ and at most $q_{\mathcal{P}}$ queries to $\mathcal{P}^{\star -1}$, the probability of this event to occur is upper bounded by:

$$\Pr(\mathsf{bad}_2) \leq \sum_{j=1}^{q_{\mathcal{H}}} \frac{q_{\mathcal{P}}}{2^m} + \sum_{j=1}^{q_{\mathcal{P}}} \frac{2q_{\mathcal{H}}}{2^m} = \frac{3 \cdot q_{\mathcal{H}} \cdot q_{\mathcal{P}}}{2^m}\,.$$

Probability of bad_3. The event is very similar to bad event E_2 of Choi et al. [12], and we adopt their reasoning. This event happens if a query $Y = y\|y'$ to $\mathcal{P}^{\star -1}$ belongs to $\mathfrak{R}^{\star}_{\mathcal{C}}\|*^{b-m}$, that is if $y = \mathrm{left}_n(Y)$ is the result of a query to the outer construction. Consider any query. Note that this query fixes $y = \mathrm{left}_n(Y)$. The probability, conditioned on the previous queries, that this query with $\mathrm{left}_n(Y) = y$ sets bad_3 is at most $\frac{F_y \cdot 2^n}{2^b - q_{\mathcal{P}}}$, as for any guess there are F_y possible values y' that could set the bad event, and the adversary knows at most $q_{\mathcal{P}}$ earlier outcomes of $\mathcal{P}^{\star}$. Provided that $q_{\mathcal{P}} \leq 2^{b-1}$, and as the adversary can choose y, the conditional probability that the j-th query sets bad_3 is upper bounded by

$$\frac{\max_{y \in \{0,1\}^n} F_y}{2^{b-n-1}} .$$

Thus, summing over all queries we get

$$\Pr(\mathsf{bad}_3) \leq \sum_{j=1}^{q_{\mathcal{P}}} \mathrm{Ex}_j \left(\frac{\max_{y \in \{0,1\}^n} F_y}{2^{b-n-1}} \right) , \tag{15}$$

where $\mathrm{Ex}_j(\cdot)$ denotes the expectation taken over the interaction between $\mathcal{D}$ and $\mathcal{P}^{\star}$ up to the j-th simulator query. Choi et al. [12] derived the following bound:

$$\mathrm{Ex}_j \left(\max_{y \in \{0,1\}^n} F_y \right) \leq \frac{Q}{2^{n-1}} + 3 \cdot \ln(Q) + 3n + 1 , \tag{16}$$

provided that $Q + q_{\mathcal{H}} + q_{\mathcal{P}} \leq 2^{b-1}$. The proof of this bound is included in Supplementary Material [19, App. A] for reference. By combining (15) and (16), we obtain:

$$\Pr(\mathsf{bad}_3) \leq \sum_{j=1}^{q_{\mathcal{P}}} \left(\frac{Q}{2^{b-2}} + \frac{3 \cdot \ln(Q) + 3n + 1}{2^{b-n-1}} \right) = \frac{Q \cdot q_{\mathcal{P}}}{2^{b-2}} + \frac{(3 \cdot \ln(Q) + 3n + 1) \cdot q_{\mathcal{P}}}{2^{b-n-1}} .$$

4.4 Upper Bound on Distance Between $\mathcal{W}_{\mathrm{S}}$ and $\mathcal{W}_{\mathrm{I}}$ (Lemma 2)

We will use the chi-squared method (recalled in Section 2.2) to provide an upper bound of the distance between the simulated world and the intermediate world. The analysis is inspired by Choi et al. [12], but crucially differs on certain aspects. Most importantly, the elimination of the initial value IV and the subsequent changes to the intermediate world have created significant differences in the distributions between the two worlds $\mathcal{W}_{\mathrm{S}}$ and $\mathcal{W}_{\mathrm{I}}$, as we will explain below.

Note that, by design, the support of the intermediate world $\mathcal{W}_{\mathrm{I}}$ is contained in the support of the simulated world $\mathcal{W}_{\mathrm{S}}$. Let $\Omega = \{0,1\}^n \times \{0,1\}^m \times \{0,1\}^b$ be the set that contains all possible answers for oracle queries to the simulated world

$\mathcal{W}_{\mathrm{S}}$. For fixed $j \in \{1, \ldots, Q + q_{\mathcal{H}} + q_{\mathcal{P}}\}$ and $\boldsymbol{z}_{j-1} = (z_1, z_2, \ldots, z_{j-1}) \in \Omega^{j-1}$ such that $p^{j-1}_{\mathcal{W}_{\mathrm{S}}}(\boldsymbol{z}_{j-1}) > 0$, our goal is to compute a bound on

$$\chi^2(\boldsymbol{z}_{j-1}) = \sum_{z \in \Omega \text{ such that } p^{\boldsymbol{z}_{j-1}}_{\mathcal{W}_{\mathrm{S}},j}(z) > 0} \frac{\left(p^{\boldsymbol{z}_{j-1}}_{\mathcal{W}_{\mathrm{S}},j}(z) - p^{\boldsymbol{z}_{j-1}}_{\mathcal{W}_{\mathrm{I}},j}(z)\right)^2}{p^{\boldsymbol{z}_{j-1}}_{\mathcal{W}_{\mathrm{S}},j}(z)} \,. \tag{17}$$

Note that the distinguisher can make four types of queries:

- to the outer construction (either the function $\mathcal{RO}$ in world $\mathcal{W}_{\mathrm{S}}$ or $\mathcal{C}[\mathcal{H}^\star, \mathcal{P}^\star]$ in world $\mathcal{W}_{\mathrm{I}}$);
- to the first primitive (either the function $\mathcal{S}_{\mathcal{H}}[\mathcal{RO}]$ in world $\mathcal{W}_{\mathrm{S}}$ or $\mathcal{H}^\star$ in world $\mathcal{W}_{\mathrm{I}}$);
- to the forward interface of the second primitive (either the function $\mathcal{S}_{\mathcal{P}}[\mathcal{RO}]$ in world $\mathcal{W}_{\mathrm{S}}$ or $\mathcal{P}^\star$ in world $\mathcal{W}_{\mathrm{I}}$);
- to the forward interface of the second primitive (either the function $\mathcal{S}^{-1}_{\mathcal{P}}[\mathcal{RO}]$ in world $\mathcal{W}_{\mathrm{S}}$ or $\mathcal{P}^{\star -1}$ in world $\mathcal{W}_{\mathrm{I}}$).

To bound $\chi^2(\boldsymbol{z}_{j-1})$ of (17), we will make a case distinction depending on the type of oracle query, below. Afterwards, we will combine the computations and conclude the proof using the chi-squared technique.

Query to Outer Construction. Suppose that the j-th query is an outer construction query. For any $y \in \{0,1\}^n$, we have that

$$p^{\boldsymbol{z}_{j-1}}_{\mathcal{W}_{\mathrm{S}},j}(y) = \frac{1}{2^n} \,, \qquad \text{and} \qquad p^{\boldsymbol{z}_{j-1}}_{\mathcal{W}_{\mathrm{I}},j}(y) = \frac{2^{b-n} - |\mathfrak{R}^{\star\, y}_{\mathcal{P}}|}{2^b - |\mathfrak{R}^\star_{\mathcal{P}}|} \,.$$

For world $\mathcal{W}_{\mathrm{S}}$, this is obvious as the outer construction is the random oracle. For world $\mathcal{W}_{\mathrm{I}}$, note that different inputs for $\mathcal{C}$ are mapped into different inputs for $\mathcal{P}^\star$, as the middle state value is drawn $x \xleftarrow{\$} \{0,1\}^m \setminus (\mathfrak{R}^\star_{\mathcal{H}} \cup \mathrm{left}_m(\mathfrak{D}^\star_{\mathcal{P}}))$. The resulting output Y of $\mathcal{P}^\star$ is drawn from a set of $2^b - |\mathfrak{R}^\star_{\mathcal{P}}|$ elements and exactly $2^{b-n} - |\mathfrak{R}^{\star\, y}_{\mathcal{P}}|$ of them satisfy $\mathrm{left}_n(Y) = y$.

We thus obtain for (17) that, for outer construction queries,

$$\chi^2(\boldsymbol{z}_{j-1}) = \sum_{y \in \{0,1\}^n} \frac{\left(p^{\boldsymbol{z}_{j-1}}_{\mathcal{W}_{\mathrm{S}},j}(y) - p^{\boldsymbol{z}_{j-1}}_{\mathcal{W}_{\mathrm{I}},j}(y)\right)^2}{p^{\boldsymbol{z}_{j-1}}_{\mathcal{W}_{\mathrm{S}},j}(y)} = \sum_{y \in \{0,1\}^n} \frac{(2^n \cdot |\mathfrak{R}^{\star\, y}_{\mathcal{P}}| - |\mathfrak{R}^\star_{\mathcal{P}}|)^2}{2^n \cdot (2^b - |\mathfrak{R}^\star_{\mathcal{P}}|)^2} \,.$$

Using that $|\mathfrak{R}^\star_{\mathcal{P}}| \le Q + q_{\mathcal{P}} \le 2^{b-1}$ and subsequently using that $|\mathfrak{R}^\star_{\mathcal{P}}| = \sum_{y \in \{0,1\}^n} |\mathfrak{R}^{\star\, y}_{\mathcal{P}}|$, we can bound this term as follows:

$$\begin{aligned}
\chi^2(\boldsymbol{z}_{j-1}) &\le \frac{4}{2^{2b-n}} \cdot \sum_{y \in \{0,1\}^n} \left(|\mathfrak{R}^{\star\, y}_{\mathcal{P}}| - \frac{|\mathfrak{R}^\star_{\mathcal{P}}|}{2^n} \right)^2 \\
&\le \frac{4}{2^{2b-n}} \cdot \left(\sum_{y \in \{0,1\}^n} (|\mathfrak{R}^{\star\, y}_{\mathcal{P}}|)^2 + \sum_{y \in \{0,1\}^n} \frac{(|\mathfrak{R}^\star_{\mathcal{P}}|)^2}{2^{2n}} \right)
\end{aligned}$$

$$\leq \frac{4}{2^{2b-n}} \cdot \left(\left(\sum_{y \in \{0,1\}^n} |\mathfrak{R}_{\mathcal{P}}^{\star\, y}| \right)^2 + \frac{(|\mathfrak{R}_{\mathcal{P}}^{\star}|)^2}{2^n} \right) = \frac{4 \cdot (|\mathfrak{R}_{\mathcal{P}}^{\star}|)^2}{2^{2b-n}} \cdot \left(1 + \frac{1}{2^n}\right)$$

$$\leq \frac{6 \cdot (Q + q_{\mathcal{P}})^2}{2^{2b-n}}\,, \tag{18}$$

using that $1 + \frac{1}{2^n} \leq 3/2$.

Query to First Primitive. Suppose that the j-th query is a query to the first primitive. For any $x \in \{0,1\}^m$, we have that

$$p_{\mathcal{W}_{\mathrm{S}},j}^{\boldsymbol{z}^{j-1}}(x) = \begin{cases} \dfrac{1}{2^m - |\mathfrak{R}_{\mathcal{H}} \cup \mathrm{left}_m(\mathfrak{D}_{\mathcal{P}})|} & \text{if } x \in \{0,1\}^m \setminus (\mathfrak{R}_{\mathcal{H}} \cup \mathrm{left}_m(\mathfrak{D}_{\mathcal{P}}))\,, \\ 0 & \text{otherwise}\,, \end{cases}$$

$$p_{\mathcal{W}_{\mathrm{I}},j}^{\boldsymbol{z}^{j-1}}(x) = \begin{cases} \dfrac{1}{2^m - |\mathfrak{R}_{\mathcal{H}}^{\star} \cup \mathrm{left}_m(\mathfrak{D}_{\mathcal{P}}^{\star})|} & \text{if } x \in \{0,1\}^m \setminus (\mathfrak{R}_{\mathcal{H}}^{\star} \cup \mathrm{left}_m(\mathfrak{D}_{\mathcal{P}}^{\star}))\,, \\ 0 & \text{otherwise}\,. \end{cases}$$

Note that, despite what intuition suggests, these distributions are *not* the same. In world $\mathcal{W}_{\mathrm{I}}$, any construction query adds tuples to $\mathfrak{C}_{\mathcal{H}}^{\star}$ and $\mathfrak{C}_{\mathcal{P}}^{\star}$, whereas this is not the case for $\mathcal{W}_{\mathrm{S}}$. In the proof of Choi et al., this issue is resolved by restricting world $\mathcal{W}_{\mathrm{I}}$ in such a way that the differences are annihilated. In our case, with the omission of the initial value, this is not possible.[1]

Nevertheless, we *do* have that $(\mathfrak{R}_{\mathcal{H}} \cup \mathrm{left}_m(\mathfrak{D}_{\mathcal{P}})) \subseteq (\mathfrak{R}_{\mathcal{H}}^{\star} \cup \mathrm{left}_m(\mathfrak{D}_{\mathcal{P}}^{\star}))$. Thus, (17) satisfies, for queries to the first primitive,

$$\begin{aligned} \chi^2(\boldsymbol{z}_{j-1}) &= \sum_{x \in \{0,1\}^m} \frac{\left(p_{\mathcal{W}_{\mathrm{S}},j}^{\boldsymbol{z}^{j-1}}(x) - p_{\mathcal{W}_{\mathrm{I}},j}^{\boldsymbol{z}^{j-1}}(x)\right)^2}{p_{\mathcal{W}_{\mathrm{S}},j}^{\boldsymbol{z}^{j-1}}(x)} \\ &= \sum_{x \in \left(\mathfrak{R}_{\mathcal{H}}^{\star} \cup \mathrm{left}_m(\mathfrak{D}_{\mathcal{P}}^{\star})\right) \setminus (\mathfrak{R}_{\mathcal{H}} \cup \mathrm{left}_m(\mathfrak{D}_{\mathcal{P}}))} p_{\mathcal{W}_{\mathrm{S}},j}^{\boldsymbol{z}^{j-1}}(x) \\ &\quad + \sum_{x \in \{0,1\}^m \setminus \left(\mathfrak{R}_{\mathcal{H}}^{\star} \cup \mathrm{left}_m(\mathfrak{D}_{\mathcal{P}}^{\star})\right)} \frac{\left(p_{\mathcal{W}_{\mathrm{S}},j}^{\boldsymbol{z}^{j-1}}(x) - p_{\mathcal{W}_{\mathrm{I}},j}^{\boldsymbol{z}^{j-1}}(x)\right)^2}{p_{\mathcal{W}_{\mathrm{S}},j}^{\boldsymbol{z}^{j-1}}(x)}\,, \end{aligned}$$

where $p_{\mathcal{W}_{\mathrm{I}},j}^{\boldsymbol{z}^{j-1}}(x) = 0$ in the first sum. As $|\,(\mathfrak{R}_{\mathcal{H}}^{\star} \cup \mathrm{left}_m(\mathfrak{D}_{\mathcal{P}}^{\star})) \setminus (\mathfrak{R}_{\mathcal{H}} \cup \mathrm{left}_m(\mathfrak{D}_{\mathcal{P}}))\,| \leq Q$, we can bound this term as follows:

$$\chi^2(\boldsymbol{z}_{j-1}) \leq \frac{Q}{2^m - |\mathfrak{R}_{\mathcal{H}} \cup \mathrm{left}_m(\mathfrak{D}_{\mathcal{P}})|}$$

[1] Note that Choi et al. did not have a hash primitive, whereas we do, so the drawn parallel here is a bit odd. The comparison with Choi et al.'s proof and the induced difficulties become more apparent when we consider inverse queries, later on.

$$
\begin{aligned}
&+ 2^m \cdot \frac{\left(|\mathfrak{R}_{\mathcal{H}} \cup \mathrm{left}_m(\mathfrak{D}_{\mathcal{P}})| - |\mathfrak{R}^{\star}_{\mathcal{H}} \cup \mathrm{left}_m(\mathfrak{D}^{\star}_{\mathcal{P}})|\right)^2}{(2^m - |\mathfrak{R}_{\mathcal{H}} \cup \mathrm{left}_m(\mathfrak{D}_{\mathcal{P}})|) \cdot (2^m - |\mathfrak{R}^{\star}_{\mathcal{H}} \cup \mathrm{left}_m(\mathfrak{D}^{\star}_{\mathcal{P}})|)^2} \\
&\leq \frac{Q}{2^m - |\mathfrak{R}_{\mathcal{H}} \cup \mathrm{left}_m(\mathfrak{D}_{\mathcal{P}})|} + \frac{2^m \cdot Q^2}{(2^m - |\mathfrak{R}^{\star}_{\mathcal{H}} \cup \mathrm{left}_m(\mathfrak{D}^{\star}_{\mathcal{P}})|)^3} \\
&\leq \frac{2 \cdot Q}{2^m} + \frac{8 \cdot Q^2}{2^{2m}}\,, \qquad (19)
\end{aligned}
$$

where we used that $|\mathfrak{R}^{\star}_{\mathcal{H}} \cup \mathrm{left}_m(\mathfrak{D}^{\star}_{\mathcal{P}})| \leq Q + q_{\mathcal{H}} + q_{\mathcal{P}} \leq 2^{m-1}$.

Forward Query to Second Primitive. Suppose that the j-th query is a forward query $X = x\|I$ to the second primitive. We can distinguish three cases:

1. a query that does not complete a construction evaluation, namely a query X for which $x \notin \mathfrak{R}_{\mathcal{H}}$ (equivalently, for which $\nexists M \in \mathfrak{D}_{\mathcal{H}}$ such that $\mathcal{H}(M) = x$);
2. a forward query on an element related to a previous outer construction query, namely a query X for which there exists $M \in \mathfrak{D}_{\mathcal{H}}$ such that (i) $\mathcal{H}(M) = x$ and (ii) $(M, I) \in \mathfrak{D}_{\mathcal{C}}$;
3. a query that does complete a construction evaluation, namely a query X for which there exists $M \in \mathfrak{D}_{\mathcal{H}}$ such that (i) $\mathcal{H}(M) = x$ and (ii) $(M, I) \notin \mathfrak{D}_{\mathcal{C}}$.

We will analyze these three cases separately.

First Case. For any $Y \in \{0,1\}^b$, we have that

$$
p^{\boldsymbol{z}^{j-1}}_{\mathcal{W}_{\mathrm{S}},j}(Y) = \begin{cases} \dfrac{1}{2^b - |\mathfrak{R}_{\mathcal{P}}|} & \text{if } Y \in \{0,1\}^b \setminus \mathfrak{R}_{\mathcal{P}}\,, \\ 0 & \text{otherwise}\,, \end{cases}
$$

$$
p^{\boldsymbol{z}^{j-1}}_{\mathcal{W}_{\mathrm{I}},j}(Y) = \begin{cases} \dfrac{1}{2^b - |\mathfrak{R}^{\star}_{\mathcal{P}}|} & \text{if } Y \in \{0,1\}^b \setminus \mathfrak{R}^{\star}_{\mathcal{P}}\,, \\ 0 & \text{otherwise}\,. \end{cases}
$$

As for the case of queries to the first primitive, these distributions are not the same. Nevertheless, we *do* have that $\mathfrak{R}_{\mathcal{P}} \subseteq \mathfrak{R}^{\star}_{\mathcal{P}}$. Thus, (17) satisfies, for forward queries to the first primitive of type 1,

$$
\begin{aligned}
\chi^2(\boldsymbol{z}_{j-1}) &= \sum_{Y \in \{0,1\}^b} \frac{\left(p^{\boldsymbol{z}^{j-1}}_{\mathcal{W}_{\mathrm{S}},j}(Y) - p^{\boldsymbol{z}^{j-1}}_{\mathcal{W}_{\mathrm{I}},j}(Y)\right)^2}{p^{\boldsymbol{z}^{j-1}}_{\mathcal{W}_{\mathrm{S}},j}(Y)} \\
&= \sum_{Y \in \mathfrak{R}^{\star}_{\mathcal{P}} \setminus \mathfrak{R}_{\mathcal{P}}} p^{\boldsymbol{z}^{j-1}}_{\mathcal{W}_{\mathrm{S}},j}(Y) + \sum_{Y \in \{0,1\}^b \setminus \mathfrak{R}^{\star}_{\mathcal{P}}} \frac{\left(p^{\boldsymbol{z}^{j-1}}_{\mathcal{W}_{\mathrm{S}},j}(Y) - p^{\boldsymbol{z}^{j-1}}_{\mathcal{W}_{\mathrm{I}},j}(Y)\right)^2}{p^{\boldsymbol{z}^{j-1}}_{\mathcal{W}_{\mathrm{S}},j}(Y)}\,.
\end{aligned}
$$

where $p^{\boldsymbol{z}^{j-1}}_{\mathcal{W}_{\mathrm{I}},j}(Y) = 0$ in the first sum. As $|\mathfrak{R}^{\star}_{\mathcal{P}} \setminus \mathfrak{R}_{\mathcal{P}}| \leq Q$, we can bound this term as follows:

$$
\chi^2(\boldsymbol{z}_{j-1}) \leq \frac{Q}{2^b - |\mathfrak{R}_{\mathcal{P}}|} + 2^b \cdot \frac{\left(|\mathfrak{R}_{\mathcal{P}}| - |\mathfrak{R}^{\star}_{\mathcal{P}}|\right)^2}{(2^b - |\mathfrak{R}_{\mathcal{P}}|) \cdot (2^b - |\mathfrak{R}^{\star}_{\mathcal{P}}|)^2}
$$

$$\leq \frac{Q}{2^b - |\mathfrak{R}_{\mathcal{P}}|} + \frac{2^b \cdot Q^2}{(2^b - |\mathfrak{R}^{\star}_{\mathcal{P}}|)^3}$$
$$\leq \frac{2 \cdot Q}{2^b} + \frac{8 \cdot Q^2}{2^{2b}}\,, \tag{20}$$

where we used that $|\mathfrak{R}^{\star}_{\mathcal{P}}| \leq Q + q_{\mathcal{P}} \leq 2^{b-1}$.

Second Case. Let $M \in \mathfrak{D}_{\mathcal{H}}$ be the *unique* (by design of $\mathcal{S}_{\mathcal{H}}$ and $\mathcal{H}^{\star}$) value such that $(M, x) \in \mathfrak{C}_{\mathcal{H}}$. Let y be such that $((M, I), y) \in \mathfrak{C}_{\mathcal{C}}$. By design, in both worlds, the response Y will be of the form $y \| y'$ for some $y' \in \{0,1\}^{b-n}$. As a matter of fact, in both worlds, the value y' is randomly drawn in such a way that $y\|y'$ does not collide with a former primitive evaluation. In other words, for any $Y \in \{0,1\}^b$, we have that

$$p^{\boldsymbol{z}_{j-1}}_{\mathcal{W}_{\mathrm{S}},j}(Y) = \begin{cases} \dfrac{1}{2^{b-n} - |\mathfrak{R}^{y}_{\mathcal{P}}|} & \text{if } Y = y\|y' \text{ with } y' \in \{0,1\}^{b-n} \setminus \mathfrak{R}^{y}_{\mathcal{P}}\,, \\ 0 & \text{otherwise}\,, \end{cases}$$
$$p^{\boldsymbol{z}_{j-1}}_{\mathcal{W}_{\mathrm{I}},j}(Y) = \begin{cases} \dfrac{1}{2^{b-n} - |\mathfrak{R}^{\star\,y}_{\mathcal{P}}|} & \text{if } Y = y\|y' \text{ with } y' \in \{0,1\}^{b-n} \setminus \mathfrak{R}^{\star\,y}_{\mathcal{P}}\,, \\ 0 & \text{otherwise}\,. \end{cases}$$

The case of world $\mathcal{W}_{\mathrm{I}}$, however, needs some clarification. In principle, procedure $\mathcal{P}^{\star}$ draws $Y \xleftarrow{\$} \{0,1\}^b \setminus \mathfrak{R}^{\star}_{\mathcal{P}}$. However, as we condition on the query history, we are *given* the earlier tuple including the value y. Condition on this, Y is drawn uniformly randomly such that $\mathrm{left}_n(Y) = y$ and such that Y hits no other range value. This is equivalent to drawing $y' \xleftarrow{\$} \{0,1\}^{b-n} \setminus \mathfrak{R}^{\star\,y}_{\mathcal{P}}$, for given y. Note, also, that this value might have been given a different value internally in the shuffling of $\mathcal{P}^{\star-1}$, but also here, Y is generated identically.

As for the case of queries to the first primitive, these distributions are not the same. Nevertheless, we *do* have that $\mathfrak{R}^{y}_{\mathcal{P}} \subseteq \mathfrak{R}^{\star\,y}_{\mathcal{P}}$. Thus, (17) satisfies, for forward queries to the first primitive of type 2,

$$\chi^2(\boldsymbol{z}_{j-1}) = \sum_{Y \in \{0,1\}^b} \frac{\left(p^{\boldsymbol{z}_{j-1}}_{\mathcal{W}_{\mathrm{S}},j}(Y) - p^{\boldsymbol{z}_{j-1}}_{\mathcal{W}_{\mathrm{I}},j}(Y)\right)^2}{p^{\boldsymbol{z}_{j-1}}_{\mathcal{W}_{\mathrm{S}},j}(Y)}$$
$$= \sum_{\substack{Y = y\|y' \text{ with} \\ y' \in \mathfrak{R}^{\star\,y}_{\mathcal{P}} \setminus \mathfrak{R}^{y}_{\mathcal{P}}}} p^{\boldsymbol{z}_{j-1}}_{\mathcal{W}_{\mathrm{S}},j}(Y) + \sum_{\substack{Y = y\|y' \text{ with} \\ y' \in \{0,1\}^{b-n} \setminus \mathfrak{R}^{\star\,y}_{\mathcal{P}}}} \frac{\left(p^{\boldsymbol{z}_{j-1}}_{\mathcal{W}_{\mathrm{S}},j}(Y) - p^{\boldsymbol{z}_{j-1}}_{\mathcal{W}_{\mathrm{I}},j}(Y)\right)^2}{p^{\boldsymbol{z}_{j-1}}_{\mathcal{W}_{\mathrm{S}},j}(Y)}\,.$$

where $p^{\boldsymbol{z}_{j-1}}_{\mathcal{W}_{\mathrm{I}},j}(Y) = 0$ in the first sum. As $|\mathfrak{R}^{\star\,y}_{\mathcal{P}} \setminus \mathfrak{R}^{y}_{\mathcal{P}}| \leq 1$ (as in world $\mathcal{W}_{\mathrm{I}}$, $\mathcal{H}^{\star}$ does not output collisions), we can bound this term as follows:

$$\chi^2(\boldsymbol{z}_{j-1}) \leq \frac{1}{2^{b-n} - |\mathfrak{R}^{y}_{\mathcal{P}}|} + 2^{b-n} \cdot \frac{\left(|\mathfrak{R}^{y}_{\mathcal{P}}| - |\mathfrak{R}^{\star\,y}_{\mathcal{P}}|\right)^2}{(2^{b-n} - |\mathfrak{R}^{y}_{\mathcal{P}}|) \cdot (2^{b-n} - |\mathfrak{R}^{\star\,y}_{\mathcal{P}}|)^2}$$

$$\leq \frac{1}{2^{b-n} - |\mathfrak{R}_{\mathcal{P}}^{y}|} + \frac{2^{b-n}}{(2^{b-n} - |\mathfrak{R}_{\mathcal{P}}^{\star\, y}|)^3}$$
$$\leq \frac{2}{2^{b-n}} + \frac{8}{2^{2b-2n}}\,, \tag{21}$$

where we used that $|\mathfrak{R}_{\mathcal{P}}^{\star\, y}| \leq 1 + q_{\mathcal{P}} \leq 2^{b-n-1}$.

Third Case. Let $M \in \mathfrak{D}_{\mathcal{H}}$ be the *unique* (by design of $\mathcal{S}_{\mathcal{H}}$ and $\mathcal{H}^{\star}$) value such that $(M, x) \in \mathfrak{C}_{\mathcal{H}}$. In world $\mathcal{W}_{\mathrm{S}}$, the response Y is generated by calling $y \leftarrow \mathcal{RO}(M, I)$ and generating $y' \xleftarrow{\$} \{0,1\}^{b-n} \setminus \mathfrak{R}_{\mathcal{P}}^{y}$. Thus, for any $Y \in \{0,1\}^b$, we have that

$$p_{\mathcal{W}_{\mathrm{S}},j}^{\boldsymbol{z}_{j-1}}(Y) = \begin{cases} \frac{1}{2^n} \cdot \frac{1}{2^{b-n} - |\mathfrak{R}_{\mathcal{P}}^{y}|} & \text{if } Y = y\|y' \text{ with } y \in \{0,1\}^n,\ y' \in \{0,1\}^{b-n} \setminus \mathfrak{R}_{\mathcal{P}}^{y}\,, \\ 0 & \text{otherwise}\,. \end{cases}$$

For world $\mathcal{W}_{\mathrm{I}}$, the response Y is generated as $Y \xleftarrow{\$} \{0,1\}^b \setminus \mathfrak{R}_{\mathcal{P}}^{\star}$. Thus, for any $Y \in \{0,1\}^b$, we have that

$$p_{\mathcal{W}_{\mathrm{I}},j}^{\boldsymbol{z}_{j-1}}(Y) = \begin{cases} \frac{1}{2^b - |\mathfrak{R}_{\mathcal{P}}^{\star}|} & \text{if } Y \in \{0,1\}^b \setminus \mathfrak{R}_{\mathcal{P}}^{\star}\,, \\ 0 & \text{otherwise}\,. \end{cases}$$

As for the case of queries to the first primitive, these distributions are not the same. Nevertheless, we *do* have that $\mathfrak{R}_{\mathcal{P}}^{y} \subseteq \mathfrak{R}_{\mathcal{P}}^{\star\, y}$, and any value $Y \in \{0,1\}^b \setminus \mathfrak{R}_{\mathcal{P}}^{\star}$ can be written as $Y = y\|y'$ with $y \in \{0,1\}^n$, $y' \in \{0,1\}^{b-n} \setminus \mathfrak{R}_{\mathcal{P}}^{\star\, y}$. Thus, (17) satisfies, for forward queries to the first primitive of type 3,

$$\chi^2(\boldsymbol{z}_{j-1}) = \sum_{Y \in \{0,1\}^b} \frac{\left(p_{\mathcal{W}_{\mathrm{S}},j}^{\boldsymbol{z}_{j-1}}(Y) - p_{\mathcal{W}_{\mathrm{I}},j}^{\boldsymbol{z}_{j-1}}(Y)\right)^2}{p_{\mathcal{W}_{\mathrm{S}},j}^{\boldsymbol{z}_{j-1}}(Y)}$$
$$= \sum_{\substack{Y = y\|y' \text{ with} \\ y \in \{0,1\}^n \text{ and} \\ y' \in \mathfrak{R}_{\mathcal{P}}^{\star\, y} \setminus \mathfrak{R}_{\mathcal{P}}^{y}}} p_{\mathcal{W}_{\mathrm{S}},j}^{\boldsymbol{z}_{j-1}}(Y) + \sum_{\substack{Y = y\|y' \text{ with} \\ y \in \{0,1\}^n \text{ and} \\ y' \in \{0,1\}^{b-n} \setminus \mathfrak{R}_{\mathcal{P}}^{\star\, y}}} \frac{\left(p_{\mathcal{W}_{\mathrm{S}},j}^{\boldsymbol{z}_{j-1}}(Y) - p_{\mathcal{W}_{\mathrm{I}},j}^{\boldsymbol{z}_{j-1}}(Y)\right)^2}{p_{\mathcal{W}_{\mathrm{S}},j}^{\boldsymbol{z}_{j-1}}(Y)}\,.$$

where $p_{\mathcal{W}_{\mathrm{I}},j}^{\boldsymbol{z}_{j-1}}(Y) = 0$ in the first sum. As $|\mathfrak{R}_{\mathcal{P}}^{\star\, y} \setminus \mathfrak{R}_{\mathcal{P}}^{y}| \leq 1$ for any y (as in world $\mathcal{W}_{\mathrm{I}}$, $\mathcal{H}^{\star}$ does not output collisions), we can bound this term as follows:

$$\chi^2(\boldsymbol{z}_{j-1}) \leq \frac{2^n}{2^b - 2^n \cdot |\mathfrak{R}_{\mathcal{P}}^{y}|} + \sum_{y \in \{0,1\}^n} \frac{(2^n \cdot |\mathfrak{R}_{\mathcal{P}}^{y}| - |\mathfrak{R}_{\mathcal{P}}^{\star}|)^2}{(2^b - 2^n \cdot |\mathfrak{R}_{\mathcal{P}}^{y}|)(2^b - |\mathfrak{R}_{\mathcal{P}}^{\star}|)^2}\,.$$

Using that $|\mathfrak{R}_{\mathcal{P}}^{\star}| \leq Q + q_{\mathcal{P}} \leq 2^{b-1}$ and that $|\mathfrak{R}_{\mathcal{P}}^{y}| \leq |\mathfrak{R}_{\mathcal{P}}^{\star\, y}| \leq 1 + q_{\mathcal{P}} \leq 2^{b-n-1}$, we can bound this term as follows:

$$\chi^2(\boldsymbol{z}_{j-1}) \leq \frac{2}{2^{b-n}} + \frac{8}{2^{3b-2n}} \cdot \sum_{y \in \{0,1\}^n} \left(|\mathfrak{R}_{\mathcal{P}}^{y}| - \frac{|\mathfrak{R}_{\mathcal{P}}^{\star}|}{2^n}\right)^2$$

$$\leq \frac{2}{2^{b-n}} + \frac{8}{2^{3b-2n}} \cdot \left(\sum_{y \in \{0,1\}^n} (|\mathfrak{R}^y_{\mathcal{P}}|)^2 + \sum_{y \in \{0,1\}^n} \frac{(|\mathfrak{R}^\star_{\mathcal{P}}|)^2}{2^{2n}} \right)$$
$$\leq \frac{2}{2^{b-n}} + \frac{8}{2^{3b-2n}} \cdot \left(\left(\sum_{y \in \{0,1\}^n} |\mathfrak{R}^y_{\mathcal{P}}| \right)^2 + \frac{(|\mathfrak{R}^\star_{\mathcal{P}}|)^2}{2^n} \right)$$
$$\leq \frac{2}{2^{b-n}} + \frac{12 \cdot (Q + q_{\mathcal{P}})^2}{2^{3b-2n}}\,, \tag{22}$$

again using that $1 + \frac{1}{2^n} \leq 3/2$.

Inverse Query to Second Primitive. Suppose that the j-th query is an inverse query to the second primitive. For any $X \in \{0,1\}^b$, we have that

$$p^{\boldsymbol{z}_{j-1}}_{\mathcal{W}_{\mathrm{S}},j}(X) = \begin{cases} \dfrac{1}{2^b - |\mathfrak{D}_{\mathcal{P}} \cup \mathfrak{R}_{\mathcal{H}} \| *^{b-m}|} & \text{if } X \in \{0,1\}^b \setminus \left(\mathfrak{D}_{\mathcal{P}} \cup \mathfrak{R}_{\mathcal{H}} \| *^{b-m}\right), \\ 0 & \text{otherwise}\,, \end{cases}$$
$$p^{\boldsymbol{z}_{j-1}}_{\mathcal{W}_{\mathrm{I}},j}(X) = \begin{cases} \dfrac{1}{2^b - |\mathfrak{D}^\star_{\mathcal{P}} \cup \mathfrak{R}^\star_{\mathcal{H}} \| *^{b-m}|} & \text{if } X \in \{0,1\}^b \setminus \left(\mathfrak{D}^\star_{\mathcal{P}} \cup \mathfrak{R}^\star_{\mathcal{H}} \| *^{b-m}\right), \\ 0 & \text{otherwise}\,. \end{cases}$$

As for the case of queries to the first primitive, these distributions are not the same. Choi et al. had a comparable case, but they managed to construct $\mathcal{W}_{\mathrm{I}}$ in such a way that the values were drawn identically to world $\mathcal{W}_{\mathrm{S}}$. Due to our omission of the initial value IV, this is (again) not an option for us. Thus, we follow a comparable reasoning as for queries to the first primitive.

We *do* have that $\left(\mathfrak{D}_{\mathcal{P}} \cup \mathfrak{R}_{\mathcal{H}} \| *^{b-m}\right) \subseteq \left(\mathfrak{D}^\star_{\mathcal{P}} \cup \mathfrak{R}^\star_{\mathcal{H}} \| *^{b-m}\right)$. Thus, (17) satisfies, for inverse queries to the second primitive,

$$\begin{aligned} \chi^2(\boldsymbol{z}_{j-1}) &= \sum_{X \in \{0,1\}^b} \frac{\left(p^{\boldsymbol{z}_{j-1}}_{\mathcal{W}_{\mathrm{S}},j}(X) - p^{\boldsymbol{z}_{j-1}}_{\mathcal{W}_{\mathrm{I}},j}(X)\right)^2}{p^{\boldsymbol{z}_{j-1}}_{\mathcal{W}_{\mathrm{S}},j}(X)} \\ &= \sum_{X \in \left(\mathfrak{D}^\star_{\mathcal{P}} \cup \mathfrak{R}^\star_{\mathcal{H}} \| *^{b-m}\right) \setminus \left(\mathfrak{D}_{\mathcal{P}} \cup \mathfrak{R}_{\mathcal{H}} \| *^{b-m}\right)} p^{\boldsymbol{z}_{j-1}}_{\mathcal{W}_{\mathrm{S}},j}(X) \\ &\quad + \sum_{X \in \{0,1\}^b \setminus \left(\mathfrak{D}^\star_{\mathcal{P}} \cup \mathfrak{R}^\star_{\mathcal{H}} \| *^{b-m}\right)} \frac{\left(p^{\boldsymbol{z}_{j-1}}_{\mathcal{W}_{\mathrm{S}},j}(X) - p^{\boldsymbol{z}_{j-1}}_{\mathcal{W}_{\mathrm{I}},j}(X)\right)^2}{p^{\boldsymbol{z}_{j-1}}_{\mathcal{W}_{\mathrm{S}},j}(X)}\,. \end{aligned}$$

where $p^{\boldsymbol{z}_{j-1}}_{\mathcal{W}_{\mathrm{I}},j}(X) = 0$ in the first sum. As $|\left(\mathfrak{D}^\star_{\mathcal{P}} \cup \mathfrak{R}^\star_{\mathcal{H}} \| *^{b-m}\right) \setminus \left(\mathfrak{D}_{\mathcal{P}} \cup \mathfrak{R}_{\mathcal{H}} \| *^{b-m}\right)| \leq 2^{b-m} \cdot Q$, we can bound this term as follows:

$$\chi^2(\boldsymbol{z}_{j-1}) \leq \frac{2^{b-m} \cdot Q}{2^b - |\mathfrak{D}_{\mathcal{P}} \cup \mathfrak{R}_{\mathcal{H}} \| *^{b-m}|}$$

$$
\begin{aligned}
&+ 2^b \cdot \frac{\left(|\mathfrak{D}_{\mathcal{P}} \cup \mathfrak{R}_{\mathcal{H}}\| *^{b-m} | - |\mathfrak{D}^{\star}_{\mathcal{P}} \cup \mathfrak{R}^{\star}_{\mathcal{H}}\| *^{b-m} |\right)^2}{(2^b - |\mathfrak{D}_{\mathcal{P}} \cup \mathfrak{R}_{\mathcal{H}}\| *^{b-m} |) \cdot (2^b - |\mathfrak{D}^{\star}_{\mathcal{P}} \cup \mathfrak{R}^{\star}_{\mathcal{H}}\| *^{b-m} |)^2} \\
&\leq \frac{2^{b-m} \cdot Q}{2^b - |\mathfrak{D}_{\mathcal{P}} \cup \mathfrak{R}_{\mathcal{H}}\| *^{b-m} |} + \frac{2^{3b-2m} \cdot Q^2}{(2^b - |\mathfrak{D}^{\star}_{\mathcal{P}} \cup \mathfrak{R}^{\star}_{\mathcal{H}}\| *^{b-m} |)^3} \\
&\leq \frac{2 \cdot Q}{2^m} + \frac{8 \cdot Q^2}{2^{2m}}\,, \qquad (23)
\end{aligned}
$$

where we used that $|\mathfrak{D}^{\star}_{\mathcal{P}} \cup \mathfrak{R}^{\star}_{\mathcal{H}}\| *^{b-m} | \leq 2^{b-m} \cdot Q + 2^{b-m} \cdot q_{\mathcal{H}} + q_{\mathcal{P}} \leq 2^{b-1}$.

Conclusion. The j-th query is of either of the four types outlined in the beginning of this section (construction query, first primitive query, forward second primitive query (of any type), or inverse primitive query). We can thus obtain that the term of (17) satisfies

$$
\begin{aligned}
\chi^2(\boldsymbol{z}_{j-1}) &\leq \max\{(18), (19), (20), (21), (22), (23)\} \\
&\leq \frac{12 \cdot (Q + q_{\mathcal{P}})^2}{2^{2b-n}} + \frac{6 \cdot Q}{2^m} + \frac{10}{2^{b-n}}\,,
\end{aligned}
$$

where we summarize that the individual terms held conditioned on the fact that $Q + q_{\mathcal{H}} + q_{\mathcal{P}} \leq 2^{m-1}$ and $1 + q_{\mathcal{P}} \leq 2^{b-n-1}$. Using the chi-squared method, we obtain the following bound on the distance between $\mathcal{W}_{\mathrm{S}}$ and $\mathcal{W}_{\mathrm{I}}$:

$$
\begin{aligned}
\left|\Pr\left(\mathcal{D}^{\mathcal{W}_{\mathrm{S}}} = 1\right) - \Pr\left(\mathcal{D}^{\mathcal{W}_{\mathrm{I}}} = 1\right)\right| &\leq \left(\frac{1}{2} \sum_{j=1}^{Q+q_{\mathcal{H}}+q_{\mathcal{P}}} \mathrm{Ex}\left(\chi^2(\boldsymbol{z}_{j-1})\right)\right)^{\frac{1}{2}} \\
&\leq \left(\frac{6 \cdot (Q + q_{\mathcal{H}} + q_{\mathcal{P}})^3}{2^{2b-n}} + \frac{3 \cdot (Q + q_{\mathcal{H}} + q_{\mathcal{P}})^2}{2^m} + \frac{5 \cdot (Q + q_{\mathcal{H}} + q_{\mathcal{P}})}{2^{b-n}}\right)^{\frac{1}{2}}.
\end{aligned}
$$

5 Application and Comparison

We will compare the Cascade-MGF construction of (11) with the existing MGF constructions mentioned in Section 1, Counter-MGF and Chained-MGF. In this comparison, we aim for a security level k. Note that both Cascade-MGF and Chained-MGF can be compared regardless of the actual hash function $\mathcal{H}$ in use, and we will compare these two schemes in Section 5.1. The efficiency of Counter-MGF depends on the actual hash function $\mathcal{H}$ in use, and we will mount a comparison of our scheme with Counter-MGF in Section 5.2.

5.1 Cascade-MGF Versus Chained-MGF

The two schemes can be compared regardless of the hash function $\mathcal{H}$ in use, the only thing we require of $\mathcal{H}$ is that it must output digests of size $m = 2k$ bits, for both schemes. Likewise, in order to achieve k-bit security, we set $n = 2k$ for both schemes. (Strictly seen, we have a $\log_2(k)$ loss in the output, in the sense that

Table 1: Security of existing methods for variable-length digest generation, tailored to the use of a b-bit permutation with k-bit security. The size of the digest blocks discards logarithmic factors for simplicity – see Section 5.1 for details.

Construction	(Underlying) primitive(s)	Size of digest block	# primitive call(s) per digest block	Max # of digest blocks	Reference
Chained-MGF with TRUNC (3)	$\mathcal{H}, \mathcal{P}$	$b-k$	1	$\min\{2^k, 2^{b-3k}\}$	(8), [43]
Chained-MGF with SOP (4)	$\mathcal{H}, \mathcal{P}_1, \mathcal{P}_2$	b	2	$\min\{2^k, 2^{b-2k}\}$	(8), [43]
Cascade-MGF	$\mathcal{H}, \mathcal{P}$	$b-k$	1	$\min\{2^k, 2^{b-2k}\}$	(9)

digest blocks are of size around $2k - \log_2(k)$ bits. This makes the comparison much harder to grasp, and in addition, the same loss occurs for both schemes.)

For our construction Cascade-MGF, we can conclude from Theorem 1 that we require the width b of the permutation $\mathcal{P}$ to satisfy $b \geq \max\left\{\frac{3k+n}{2}, k+n\right\} = 3k$. As the permutation $\mathcal{P}$ takes as input the hash result $\mathcal{H}(M) \in \{0,1\}^{2k}$ and the counter $\langle i \rangle_l$, we can conclude that in Cascade-MGF we can take $l = b - 2k$.

For fair comparison, we will consider Chained-MGF to be instantiated with a permutation-based $\mathcal{F}$ as well. We will consider it to be instantiated with either TRUNC of (3) or SOP of (4). The comparison is summarized in Table 1.

Comparison to Chained-MGF with TRUNC (3). Also in this case, we require $b \geq 3k$ for the same reason as for Cascade-MGF. In order to achieve k bits of security, it is required that the initial value IV is of size k bits. As the b-bit hash function must absorb the hash output $\mathcal{H}(M) \in \{0,1\}^{2k}$, the initial value $IV \in \{0,1\}^k$, *and* the counter $\langle i \rangle_l$, we can conclude that the construction can only take $l = b - 3k$ bits of counter. (Note that also for this construction, we have omitted the $\log_2(k)$ loss in the output, in the sense that digest blocks are of size around $2k - \log_2(k)$ bits.)

We can observe that Cascade-MGF is exactly as secure and as efficient as Chained-MGF with TRUNC, with the sole difference that our construction allows for the generation of more output blocks: $\min\{2^k, 2^{b-2k}\}$ as opposed to $\min\{2^k, 2^{b-3k}\}$. The difference is particularly significant if one instantiates $\mathcal{P}$ with a small permutation. For example, focusing on $k = 128$-bit security, one can instantiate $\mathcal{P}$ using a 384-bit permutation such as GIMLI [3] or Xoodoo [14]. In this case, Cascade-MGF can be used to generate up to 2^{128} digest blocks, whereas Chained-MGF with TRUNC can only be used to generate 1 digest block (as the counter size is 0).

As a concrete application, Gunsing [20] recently proposed the tree sponge, a generalization of the sequential sponge construction with parallel absorbing and squeezing. Referring to [20, Sect. 5.3], the squeezing phase is set up via a Chained-MGF instantiated with TRUNC. Our results imply that such construction would benefit by replacing Chained-MGF with Cascade-MGF.

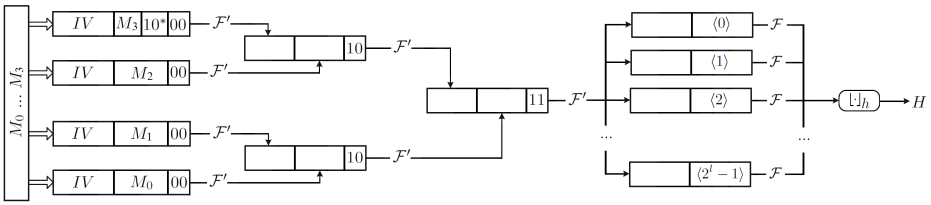


Fig. 2: The Cascade-MGF construction instantiated via a tree hash construction.

Comparison to Chained-MGF with SOP (4). The sum of permutation construction achieves optimal b-bit security, and by using this function as finalization in Chained-MGF, one can output b-bit digest blocks in parallel. This is more than the $(b-k)$-bit (or in fact, $b-k-\log_2(k)$-bit) digest blocks in Cascade-MGF, but the downside is that two permutation calls are made for one output block. Concretely, Cascade-MGF improves over Chained-MGF with SOP if $b/2 \leq b-k$, or equivalently if $2k \leq b$. By construction, this is always the case, recalling that the hash function output is of size $2k$ bits.

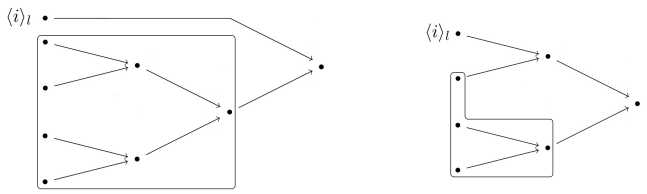
5.2 Cascade-MGF Versus Counter-MGF

A comparison between the Counter-MGF and Cascade-MGF constructions cannot be mounted without being specific about the hash function $\mathcal{H} : \{0,1\}^* \to \{0,1\}^m$ in use, where $m = 2k$ for k-bit security. Since the focus of this work is a parallel digest generation, it makes sense to consider parallelizable hashing as well, and more specifically, tree hashing.

The idea of tree hashing [33,34] is to partition the message into blocks, which are subsequently placed at the leaves of a tree. Each non-leaf node of the tree is then a compression function evaluation of its child notes. The indifferentiability of permutation-based tree hashing was analyzed by Dodis et al. [18] and Bertoni et al. [7], and more recently by Daemen et al. [15] and Gunsing et al. [21].

Let $\mathcal{P}' : \{0,1\}^{b'} \to \{0,1\}^{b'}$ be a permutation. Define $\mathcal{F}' : \{0,1\}^{b'} \to \{0,1\}^m$ as $\mathcal{F}'(X) = \text{left}_m(\mathcal{P}'(X))$. The most minimalistic permutation-based tree hashing construction that is proven to be indifferentiable initiates the leaves with an m-bit initial value IV, with message bits (possibly injectively padded), and frame bits 00. Non-leaf nodes consist of chaining bits coming from evaluations of $\mathcal{F}'$ of the children and frame bits 10. The final evaluation takes frame bits 11. The construction is depicted in Figure 2, where it is already used to instantiate $\mathcal{H}$ in Cascade-MGF.

Daemen et al. [15] and Gunsing et al. [21] proved that if $\mathcal{P}'$ is a random permutation, this tree hash function construction behaves like a random oracle against distinguishers with a complexity of around $\min\{2^{m/2}, 2^{(b'-m)/2}\}$. It follows that k-bit security is achieved for $m \geq 2k$ and $b' \geq 4k$. Strictly seen, putting $m = 2k$, we require $b' = 4k + 2$ for the frame bits $\{00, 10, 11\}$ to be taken into account.



(a) (b)

Fig. 3: The Counter-MGF construction instantiated via a tree hash function. The highlighted sub-trees are independent of the counter $\langle i \rangle_l$. We emphasize that the initial points represent the result of the compression between the initial value `IV` and the corresponding message block/$\langle i \rangle$.

Algorithm 7: Generation of t output blocks via Counter-MGF instantiated with a tree-hash function

Data: input message $M \in \{0,1\}^*$
Result: output $h_0, h_1, \ldots, h_{t-1} \in \{0,1\}^m$ for $t \geq 1$
parse $M \| 10^* = M_0 \| M_1 \| \ldots \| M_{z-1}$ for $z \geq 1$ and $M_0, M_1, \ldots, M_{z-1} \in \{0,1\}^m$
let $\zeta_0, \zeta_1, \ldots, \zeta_{\lfloor \log_2(z) \rfloor} \in \{0,1\}$ be such that $z = \sum_{i=0}^{\lfloor \log_2(z) \rfloor} \zeta_i \cdot 2^i$
let $\mathfrak{S}_0, \mathfrak{S}_1, \ldots, \mathfrak{S}_{\lfloor \log_2(z) \rfloor}$ be empty sets
$l \leftarrow 0$
`// Initially parallelizable phase`
for i ***from*** *0* ***to*** $\lfloor \log_2(z) \rfloor$ **do**
    if $\zeta_i = 1$ **then**
        `// Computing the hash of the subtrees`
        for j ***from*** *0* ***to*** $2^{\zeta_i} - 1$ **do**
            $\mathfrak{S}_i \leftarrow \mathfrak{S}_i \cup \{M_l\}$
            $l \leftarrow l + 1$
        $h'_i \leftarrow$ tree-hash$(\mathfrak{S}_i)$
`// Non-parallelizable phase`
for i ***from*** *0* ***to*** $t - 1$ **do**
    $h_i \leftarrow$ tree-hash$(\langle i \rangle, IV)$
    for j ***from*** *0* ***to*** $\lfloor \log_2(z) \rfloor$ **do**
        if $\zeta_j = 1$ **then**
            $h_i \leftarrow$ tree-hash(h'_j, h_i)
return $h_0, h_1, \ldots, h_{t-1}$

For the permutation $\mathcal{P}$ in Cascade-MGF, the same restrictions as before apply, and most importantly, $b \geq 3k$. This means that one can perform digest generation with a permutation that is approximately 25% smaller than the one used for hashing, without any security sacrifice. If necessary, we expect that Cascade-MGF can also run on a single permutation, or strictly seen two permutations that are similar, possibly instantiated with different round constants.

Comparison to Counter-MGF. By the generic design of the Counter-MGF construction (7), the same truncated permutation is used both in the absorbing and squeezing part. Hence, compared to Cascade-MGF, it is not possible to work with a smaller permutation in the squeezing part.

Secondly, and more importantly, the cost for generating an output of size $n \cdot t$ for $t \geq 1$ could be much higher for Counter-MGF than for Cascade-MGF, depending on the size of the input message M and on the value of $m = 2k$. Consider, for example, the simplified depiction of tree hashing in Figure 3:

- In case (a), the number of message blocks happens to be a power of 2. In this case, the digest blocks are generated in parallel, since the message is fully absorbed independently of the counter $\langle i \rangle$.
- In case (b), only part of the message can be absorbed independently of the counter $\langle i \rangle$, and the digest blocks are only partially generated in parallel (the extreme case showed in the picture occurs when the number of blocks that composed M is a power of 2 minus 1).

As a result, while the cost in terms of the number of function/permutation calls is the same for Counter-MGF and for Cascade-MGF in the first case (a), the cost is much smaller for Cascade-MGF than for Counter-MGF in the second case (b). In more detail, the cost for Counter-MGF is at least *twice* that of Cascade-MGF. Indeed, given an input message composed of z blocks, the cost for generating t output blocks via Cascade-MGF instantiated via a tree-hash function consists of $2 \cdot z - 1$ $\mathcal{F}'$-calls for the compression part, and t $\mathcal{P}$-calls for the expansion part, for a total of

$$t + 2 \cdot z - 1 \in \mathcal{O}(t)$$

function/permutation calls. In the case of Counter-MGF instantiated via a tree-hash function, the cost is

$$(1 + \mathrm{Hw}(z)) \cdot t + 2 \cdot z - \mathrm{Hw}(z) \in \mathcal{O}\left((1 + \mathrm{Hw}(z)) \cdot t\right)$$

function/permutation calls, based on the algorithm given in Algorithm 7. The difference between the two cases is approximately a factor $(1 + \mathrm{Hw}(z)) \geq 2$, and the maximum is attained when the number of blocks z is a power of 2 minus 1.

Acknowledgments. Lorenzo Grassi is supported by the European Research Council under the ERC advanced grant agreement under grant ERC-2017-ADG Nr. 788980 ESCADA. Bart Mennink is supported by the Netherlands Organisation for Scientific Research (NWO) under grant VI.Vidi.203.099.

References

1. ANSI: ANSI X9.44: Public-Key Cryptography for the Financial Services Industry Key Establishment Using Integer Factorization Cryptography (2002)
2. Aumasson, J., Neves, S., Wilcox-O'Hearn, Z., Winnerlein, C.: BLAKE2: Simpler, Smaller, Fast as MD5. In: Jr., M.J.J., Locasto, M.E., Mohassel, P., Safavi-Naini, R. (eds.) ACNS 2013. LNCS, vol. 7954, pp. 119–135. Springer (2013)

3. Bernstein, D.J., Kölbl, S., Lucks, S., Massolino, P.M.C., Mendel, F., Nawaz, K., Schneider, T., Schwabe, P., Standaert, F., Todo, Y., Viguier, B.: Gimli : A Cross-Platform Permutation. In: CHES 2017. LNCS, vol. 10529, pp. 299–320. Springer (2017)
4. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Sponge functions (2007), in: Ecrypt Hash Workshop 2007
5. Bertoni, G., Daemen, J., Hoffert, S., Peeters, M., Van Assche, G., Van Keer, R.: Farfalle: parallel permutation-based cryptography. IACR Trans. Symmetric Cryptol. 2017(4), 1–38 (2017)
6. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the Indifferentiability of the Sponge Construction. In: EUROCRYPT 2008. LNCS, vol. 4965, pp. 181–197. Springer (2008)
7. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Sufficient conditions for sound tree and sequential hashing modes. Int. J. Inf. Sec. 13(4), 335–353 (2014)
8. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: The KECCAK reference (2011)
9. Bhattacharya, S., Nandi, M.: Full Indifferentiable Security of the Xor of Two or More Random Permutations Using the χ^2 Method. In: EUROCRYPT 2018. LNCS, vol. 10820, pp. 387–412. Springer (2018)
10. Black, J., Rogaway, P., Shrimpton, T., Stam, M.: An Analysis of the Blockcipher-Based Hash Functions from PGV. Journal of Cryptology 23(4), 519–545 (2010)
11. Chang, D., Lee, S., Nandi, M., Yung, M.: Indifferentiable Security Analysis of Popular Hash Functions with Prefix-Free Padding. In: ASIACRYPT 2006. LNCS, vol. 4284, pp. 283–298. Springer (2006)
12. Choi, W., Lee, B., Lee, J.: Indifferentiability of Truncated Random Permutations. In: ASIACRYPT 2019. LNCS, vol. 11921, pp. 175–195. Springer (2019)
13. Coron, J., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård Revisited: How to Construct a Hash Function. In: CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer (2005)
14. Daemen, J., Hoffert, S., Van Assche, G., Van Keer, R.: The design of Xoodoo and Xoofff. IACR Trans. Symmetric Cryptol. 2018(4), 1–38 (2018)
15. Daemen, J., Mennink, B., Van Assche, G.: Sound Hashing Modes of Arbitrary Functions, Permutations, and Block Ciphers. IACR Trans. Symmetric Cryptol. 2018(4), 197–228 (2018)
16. Dai, W., Hoang, V.T., Tessaro, S.: Information-Theoretic Indistinguishability via the Chi-Squared Method. In: CRYPTO 2017. LNCS, vol. 10403, pp. 497–523. Springer (2017)
17. Damgård, I.: A Design Principle for Hash Functions. In: CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer (1989)
18. Dodis, Y., Reyzin, L., Rivest, R.L., Shen, E.: Indifferentiability of Permutation-Based Compression Functions and Tree-Based Modes of Operation, with Applications to MD6. In: FSE 2009. LNCS, vol. 5665, pp. 104–121. Springer (2009)
19. Grassi, L., Mennink, B.: Security of Truncated Permutation Without Initial Value. Cryptology ePrint Archive, Paper 2022/508 (2022)
20. Gunsing, A.: Block-Cipher-Based Tree Hashing. In: CRYPTO 2022. LNCS, Springer (2022), to appear
21. Gunsing, A., Daemen, J., Mennink, B.: Errata to Sound Hashing Modes of Arbitrary Functions, Permutations, and Block Ciphers. IACR Trans. Symmetric Cryptol. 2020(3), 362–366 (2020)
22. IEEE Computer Society: IEEE 1363.1 Standard Specifications For Public-Key Cryptography (2000)

23. ISO/IEC: ISO/IEC 18033-2 Information technology – Security techniques – Encryption algorithms – Part 2: Asymmetric ciphers (2006)
24. Kaliski, B., Staddon, J.: PKCS #1: RSA cryptography specifications version 2.0. RFC 2437, 1–39 (1998)
25. Knudsen, L.R., Rechberger, C., Thomsen, S.S.: The Grindahl Hash Functions. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 39–57. Springer (2007)
26. Lee, J.: Indifferentiability of the Sum of Random Permutations Toward Optimal Security. IEEE Trans. Inf. Theory 63(6), 4050–4054 (2017)
27. Luykx, A., Mennink, B., Neves, S.: Security Analysis of BLAKE2's Modes of Operation. IACR Trans. Symmetric Cryptol. 2016(1), 158–176 (2016)
28. Mandal, A., Patarin, J., Nachef, V.: Indifferentiability beyond the Birthday Bound for the Xor of Two Public Random Permutations. In: INDOCRYPT 2010. LNCS, vol. 6498, pp. 69–81. Springer (2010)
29. Maurer, U.M., Renner, R., Holenstein, C.: Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In: TCC 2004. LNCS, vol. 2951, pp. 21–39. Springer (2004)
30. Mennink, B.: Optimal Collision Security in Double Block Length Hashing with Single Length Key. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 526–543. Springer (2012)
31. Mennink, B.: Indifferentiability of Double Length Compression Functions. In: Stam, M. (ed.) IMACC 2013. LNCS, vol. 8308, pp. 232–251. Springer (2013)
32. Mennink, B., Preneel, B.: On the XOR of Multiple Random Permutations. In: ACNS 2015. LNCS, vol. 9092, pp. 619–634. Springer (2015)
33. Merkle, R.C.: Protocols for Public Key Cryptosystems. In: Proceedings of the 1980 IEEE Symposium on Security and Privacy. pp. 122–134. IEEE Computer Society (1980)
34. Merkle, R.C.: A Digital Signature Based on a Conventional Encryption Function. In: CRYPTO 1987. LNCS, vol. 293, pp. 369–378. Springer (1987)
35. Merkle, R.C.: A Certified Digital Signature. In: CRYPTO 1989. LNCS, vol. 435, pp. 218–238. Springer (1989)
36. Merkle, R.C.: Secrecy, authentication and public key systems (1979), PhD thesis, UMI Research Press
37. NIST: NIST SP800-108: Recommendation for Key Derivation Using Pseudorandom Functions (2009)
38. NIST: NIST FIPS 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions (2015)
39. Preneel, B., Govaerts, R., Vandewalle, J.: Hash Functions Based on Block Ciphers: A Synthetic Approach. In: CRYPTO 1993. LNCS, vol. 773, pp. 368–378. Springer (1993)
40. Rabin, M.O.: Digitalized signatures. Foundations of Secure Computation pp. 155 – 166 (1978)
41. Rivest, R., Agre, B., Bailey, D.V., Crutchfield, C., Dodis, Y., Fleming, K.E., Khan, A., Krishnamurthy, J., Lin, Y., Reyzin, L., Shen, E., Sukha, J., Sutherland, D., Tromer, E., Yin, Y.L.: The MD6 hash function – A proposal to NIST for SHA-3 (2008), submission to NIST's SHA-3 competition
42. RSA Security: PKCS#1 v2.1: RSA Cryptography Standard (2002)
43. Suzuki, K., Yasuda, K.: On the security of the cryptographic mask generation functions standardized by ANSI, IEEE, ISO/IEC, and NIST (2012), NTT Technical Review