Indifferentiability of the Sponge Construction with a Restricted Number of Message Blocks

Charlotte Lefevre

Digital Security Group, Radboud University, Nijmegen, The Netherlands
charlotte.lefevre@ru.nl

Abstract. The sponge construction is a popular method for hashing. Quickly after its introduction, the sponge was proven to be tightly indifferentiable from a random oracle up to $\approx 2^{c/2}$ queries, where c is the capacity. However, this bound is not tight when the number of message blocks absorbed is restricted to $\ell < \lceil \frac{c}{2(b-c)} \rceil + 1$ (but still an arbitrary number of blocks can be squeezed). In this work, we show that this restriction leads to indifferentiability from a random oracle up to $\approx \min\left\{2^{b/2}, \max\left\{2^{c/2}, 2^{b-\ell\times(b-c)}\right\}\right\}$ queries, where $b > c$ is the permutation size. Depending on the parameters chosen, this result allows to have enhanced security or to absorb at a larger rate for applications that require a fixed-length input hash function.

Keywords: sponge, lightweight cryptography, indifferentiability

1 Introduction

The sponge construction is a permutation-based mode used among others for hashing. Introduced by Bertoni et al. [BDPV07], it gained quickly in popularity after the SHA-3 competition [NIS12] won by `Keccak` [BDPA11], which relies on the sponge construction. Informally, the sponge is based on a keyless permutation of size b bits. The global state of the sponge, also of size b bits, is split as the sum $r + c$, where r is called the rate, and c the capacity. The data to hash is first injectively padded, absorbed by blocks of r bits, and a digest is extracted, also by blocks of r bits. Between each data absorption and extraction, the permutation is applied on the entire state.

In the ideal permutation model, the sponge has been proven to be secure up to $\approx 2^{c/2}$ queries [BDPV08]. The bound was proven in the indifferentiability framework of Maurer et al. [MRH04] and Coron et al. [CDMP05]. Informally, this result means that the sponge construction based on an ideal permutation behaves like a random oracle in single stage games [RSS11], so that the mode has no underlying structural weaknesses. As a matter of fact, the indifferentiability bound is tight. Indeed, after $\approx 2^{c/2}$ absorb calls, one can find collisions between the states of the sponge over c bits, and by using a *subsequent* absorb call, the partial collision can be transformed into a full-state collision. This fact was already known by the designers of the sponge [BDPV07], and we explain the attack in detail in Section 3.4. One important property of this attack is that in order to succeed with high probability, the adversary needs to be able to absorb $\lceil \frac{c}{2r} \rceil + 1$ message blocks.

Contribution. In this work, we consider the sponge construction where at most ℓ message blocks are allowed to be absorbed but arbitrary-length digests can be extracted. To have a slightly more general result, we allow that the message blocks are absorbed with a rate r_a different from the squeezing rate r_s, similarly to the PHOTON construction [GPP11]. We show in Theorem 1 that the restriction on the number of message blocks leads to indifferentiability from a random oracle up to $\approx \min\left\{2^{c_s}, 2^{b/2}, \max\left\{2^{c_a/2}, 2^{b-\ell\times r_a}\right\}\right\}$ queries (up to constant and logarithmic factors), where $c_a := b - r_a, c_s := b - r_s$. A comparison of our bound with already existing works is made at the beginning of Section 4. Moreover, our proof can be easily adapted to provide security in the public indifferentiability framework [YMO09, DRS09, MPS12] up to $\approx \min\left\{2^{b/2}, \max\left\{2^{c_a/2}, 2^{b-\ell\times r_a}\right\}\right\}$ queries.

Related Work. As a matter of fact, when one message block is absorbed and the digest is extracted in one squeeze call, the sponge boils down to a truncated permutation. Choi et al. [CLL19] proved that a truncated random permutation is indifferentiable from a $\mathcal{RO}$ up to $\approx \min\left\{2^{\frac{b+c_s}{3}}, 2^{c_a}, 2^{c_s}\right\}$ queries. This result has been generalized to the case where the fixed prefix is randomized [GM22]. The developers of the PHOTON construction [GPP11] proposed to squeeze the digest at a rate $r' > r$ (note that r' corresponds to r_s in our case). Moreover, Naito and Ohta [NO14] proposed to absorb the first message block with a rate $r'' > r$, and proved indifferentiability of the underlying construction. In detail, if $c' := b - r', c'' := b - r''$, then as long as $c' \geq c/2 + \log(c)$ and $c'' \geq c/2$, the security bound that they proved does not degrade significantly compared to the plain sponge.[1] In particular, the squeezing rate and the first message block size can be significantly larger than the rate used in the subsequent absorb calls.

Application. Hash functions are usually described as functions that map binary strings of arbitrary length into digests of fixed size. However, a restricted input-length hash function is sufficient for several applications. For instance, some proof of knowledge protocols require only hashing of finite field elements. The Fiat-Shamir Heuristic [FS86] allows to remove interactivity of a public-coin proof by making use of a hash function for which the input has a length fixed in advance. Moreover, a restricted input-length hash function could be sufficient for password hashing applications. Now, given the bound of Theorem 1 and a desired security level k, one can wonder whether there are better parameter choices, i.e., can we maximize the rates r_s and r_a and the size of input strings? First, the choice $c_s = k$ maximizes the squeezing rate. Now, if arbitrary-size strings should be processed, then one has to choose $c_a = 2k$ to maximize r_a. Otherwise, we can aim at $b - \ell r_a \geq k$. From $c_s = k$, this implies $r_s \geq \ell r_a$. Thus to maximize efficiency and the domain size, the best choice is $r_a = r_s$ and $\ell = 1$, so that $c_a = c_s = k$.

Outline. The remainder of this paper is as follows. Section 2 introduces the notation and preliminary context, Section 3 presents in detail the sponge construction and the specificities of the security model to our setting, Section 4 is dedicated to the indifferentiability result and the proof, and finally Section 5 discusses the tightness of the bound.

2 Preliminaries

2.1 Notation

For a finite set S, $x \xleftarrow{\$} S$ means that the element x is sampled uniformly at random from S. $\mathbf{1}_{x \in S}$ denotes the indicator function of S, i.e.,

$$\mathbf{1}_{x \in S} = \begin{cases} 1 \text{ if } x \in S\,, \\ 0 \text{ otherwise}\,. \end{cases}$$

$x := y$ means that x is defined to be equal to y. For $i, j \in \mathbb{N}$ such that $i \leq j$, $[\![i, j]\!]$ denotes the set $\{i, \ldots, j\}$. For $a \in \mathbb{N}, \{0,1\}^a$ denotes the set of binary strings of size a, and $\{0,1\}^*$ denotes $\bigcup_{a \in \mathbb{N}} \{0,1\}^a$. We use the symbol ϵ to denote the empty string. In the following, let $X, Y \in \{0,1\}^*$. If L is an ordered list, for $1 \leq i < j \leq |L|$, $L[i:j]$ denotes the sub-list of L containing the ith to the jth element (latter excluded). We also use this notation for binary strings, i.e., $X[i:j]$ denotes the bits of X from position i to j. Moreover, let $b, r, c \in \mathbb{N}$ such that $b = r + c$. If $|X| = |Y| = b$, then $\text{outer}_r(X) := X[0:r]$, $\text{inner}_c(x) := X[b-c:b]$. Finally, we write $X \overset{c}{=} Y$ whenever $\text{inner}_c(X) = \text{inner}_c(Y)$.

[1] More precisely, in this case the adversarial advantage is upper bounded by the square root of the adversarial advantage of the plain sponge construction.

2.2 A Useful Lemma

In the proof we need to use the following result based on Choi et al.'s work [CLL19, page 187], that we state here for completeness.

Lemma 1. *Let $q, R \in \mathbb{N}$. Consider the experiment of throwing uniformly at random q balls in R bins. For $u \in [\![1, R]\!]$, denote by S_u the size of the bin number u. Then one has*

$$\mathsf{E}\left(\max_{u\in[\![1,R]\!]} S_u\right) \leq \frac{2q}{R} + 3\ln(R) + 3\ln(q) + 3\,.$$

Proof. Let $u \in [\![1, R]\!]$. This is clear that S_u follows a binomial distribution with q experiments and success probability $p := \frac{1}{R}$. Let $\mu := pq$ be the expectation of S_u. Then, from the Chernoff bound, we know that for any $\delta > 1$,

$$\mathbf{Pr}\left(S_u \geq (1+\delta)\mu\right) \leq e^{-\frac{\delta\mu}{3}}\,,$$

thus for any $j > 2\mu$,

$$\mathbf{Pr}\left(S_u \geq j\right) \leq e^{\frac{\mu-j}{3}}\,.$$

Whenever $j \geq \mu + 3\ln\left(\frac{2q}{p}\right)$, the quantity $e^{\frac{\mu-j}{3}}$ is upper bounded by $\frac{p}{2q}$. Therefore, for any $j \geq 2\mu + 3\ln\left(\frac{2q}{p}\right)$,

$$\mathbf{Pr}\left(S_u \geq j\right) \leq \frac{p}{2q}\,. \tag{1}$$

Now,

$$\begin{aligned}
\mathsf{E}\left(\max_{u\in[\![1,R]\!]} S_u\right) &= \sum_{j=1}^{q} \mathbf{Pr}\left(\max_{u\in[\![1,R]\!]} S_u \geq j\right) \\
&= \sum_{j=1}^{2\mu+3\ln\left(\frac{2q}{p}\right)} \mathbf{Pr}\left(\max_{u\in[\![1,R]\!]} S_u \geq j\right) + \sum_{j=2\mu+3\ln\left(\frac{2q}{p}\right)}^{q} \mathbf{Pr}\left(\bigcup_{u\in[\![1,R]\!]} S_u \geq j\right) \\
&\leq 2\mu + 3\ln\left(\frac{2q}{p}\right) + \sum_{j=2\mu+3\ln\left(\frac{p}{2q}\right)}^{q} \sum_{u=1}^{R} \mathbf{Pr}\left(S_u \geq j\right) \\
&\leq 2\mu + 3\ln\left(\frac{2q}{p}\right) + q \times R \times \frac{p}{2q}\,,
\end{aligned}$$

where the last inequality uses (1). By replacing p by $\frac{1}{R}$ and μ by $\frac{q}{R}$, we obtain

$$\begin{aligned}
\mathsf{E}\left(\max_{u\in[\![1,R]\!]} S_u\right) &\leq \frac{2q}{R} + 3\ln(2) + 3\ln(R) + 3\ln(q) + \frac{1}{2} \\
&\leq \frac{2q}{R} + 3\ln(R) + 3\ln(q) + 3\,,
\end{aligned}$$

which concludes the proof.

2.3 Security Model

We will use the indifferentiability framework, introduced by Maurer et al. [MRH04], and refined in the context of hash functions by Coron et al. [CDMP05]. We describe here the general indifferentiability framework, and will highlight the particularities of the security model in our setting in Section 3.2. Let $r_s \in \mathbb{N}$. Consider a construction $\mathcal{H}$ based on an ideal primitive $\mathcal{P}$,

which gives an extendable output function denoted by $\mathcal{H}^{\mathcal{P}}$. We consider random oracles [BR93] with an infinite domain and codomain, denoted by $\mathcal{RO}$. For $M \in \{0,1\}^*$, and $k, k' \in \mathbb{N}$, $\mathcal{RO}(M)[k : k']$ denotes the bits from position k to k' (latter excluded) of the stream $\mathcal{RO}(M)$. Let $\mathcal{S}$ be an algorithm called simulator which has the same input domain and outputs strings in the same range as $\mathcal{P}$, and that is allowed to query $\mathcal{RO}$. Consider a distinguisher $\mathcal{D}$ that has access to either $(\mathcal{H}^{\mathcal{P}}, \mathcal{P})$ or $(\mathcal{RO}, \mathcal{S})$. W.l.o.g., we assume that $\mathcal{D}$ never makes queries for which it already knows the answer. The primitive query history of the distinguisher is an ordered list denoted by $\mathcal{Q}_{\mathcal{P}}$, where for $X, Y \in \{0,1\}^b$, $\mathrm{d} \in \{\mathrm{fwd}, \mathrm{inv}\}, i \in \mathbb{N}$, $\mathcal{Q}_{\mathcal{P}}[i] = (X, Y, \mathrm{d})$ means that the query i was in direction d, and $\mathcal{D}$ received as answer X or Y depending on the direction of the query. Similarly, the construction query history is denoted by $\mathcal{Q}_C$, it is described in our particular setting in Section 3.2. The indifferentiability advantage of $\mathcal{D}$ from a random oracle, denoted by $\mathbf{Adv}^{\mathrm{iff}}_{\mathcal{H}}(\mathcal{D})$, is defined as follows:

$$\mathbf{Adv}^{\mathrm{iff}}_{\mathcal{H}}(\mathcal{D}) = \left| \mathbf{Pr}\left(\mathcal{D}^{\mathcal{H}^{\mathcal{P}},\mathcal{P}} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{\mathcal{RO},\mathcal{S}} = 1\right) \right| .$$

Moreover, $\mathbf{Adv}^{\mathrm{iff}}_{\mathcal{H}}(q)$ denotes the supremum of the set $\mathbf{Adv}^{\mathrm{iff}}_{\mathcal{H}}(\mathcal{D})$, over all distinguishers allowed to make at most q queries. Note that the way queries are made can be refined. In the case of the sponge, one construction query has a practical cost which depends on the length of the padded message and the number of bits of the stream to extract. In Section 3.2 we specify a metrics to count construction queries.

Public indifferentiability. In the indifferentiability setting, the simulator is *not* allowed to access the construction queries made by the adversary. This yields a security model which is too strong for some applications where queries to the hash function can be disclosed to all parties without affecting the security. Among others, this is the case for some signature schemes, proof protocols or encryption protocols, e.g. [BR96, BR93, FS86]. This motivated the introduction of the public indifferentiability framework [YMO09, DRS09, MPS12]. Public differentiability differs from indifferentiability precisely in the fact that the simulator has access to all construction queries made by the adversary. $\mathbf{Adv}^{\mathrm{pubiff}}_{\mathcal{H}}(\mathcal{D})$ and $\mathbf{Adv}^{\mathrm{pubiff}}_{\mathcal{H}}(q)$ are defined similarly to the indifferentiability setting. Public indifferentiability yields a weaker security model. For instance the (plain) Merkle-Damgård construction [Mer89, Dam89] is not indifferentiable, but it is publicly indifferentiable [DRS09].

3 Sponge Construction

3.1 The Sponge Construction

In this section we describe the sponge construction where the absorbing and squeezing rates are different. Let $b, c_a, r_a, c_s, c_a \in \mathbb{N}$ such that $b = r_a + c_a = r_s + c_s$. Let $\mathcal{P}$ be a permutation, and $\mathrm{IV} \in \{0,1\}^b$. Let *pad* be an injecting padding that splits the message to hash $M \in \{0,1\}^*$ into k blocks of length r_a and such that the last block is not zero. The inverse function of *pad* is denoted by *unpad*. We consider that $unpad(M)$ returns the symbol $\perp$ whenever M does not correspond to a valid padding. Algorithm 1 defines the sponge construction based on $\mathcal{P}$, and Fig. 1 illustrates it.

3.2 Security Model of the Sponge with Restricted Message Blocks

In the following, let $\mathcal{H}$ be the sponge construction.

Construction queries and their cost. In the real world, a construction query takes two parameters: a message $M \in \{0,1\}^*$, and an integer $k \in \mathbb{N}$. $\mathcal{H}^{\mathcal{P}}(M, k)$ corresponds to the $(k+1)^{\mathrm{th}}$ first squeezed blocks after having absorbed M. Thus in order to match the real world, we consider that the random oracle queried with inputs $M \in \{0,1\}^*$ and $k \in \mathbb{N}$ gives $\mathcal{RO}(M)[0 : r_s \times (k+1)]$. Moreover, a construction query in the real world has a practical cost which depends on the

Algorithm 1: The sponge construction. IV is any fixed b-bit string, $M \in \{0,1\}^*$ is the message to process, $n \in \mathbb{N}$ the number of bits to extract

```
Function Sponge^P (M, n):
    /* Initialization */
    S ← IV; // State of the sponge
    Z ← ∅; // Output string
    m_1 || ··· || m_k ← pad(M);
    /* Absorption */
    for i = 1, ..., k do
        S ← P(S ⊕ (m_i || 0^{c_a})) ;
    end
    /* Squeezing */
    for i = 1, ..., ⌈n / r_s⌉ do
        Z ← Z || outer_{r_s}(S);
        S ← P(S);
    end
    return Z[0 : n]
end
```

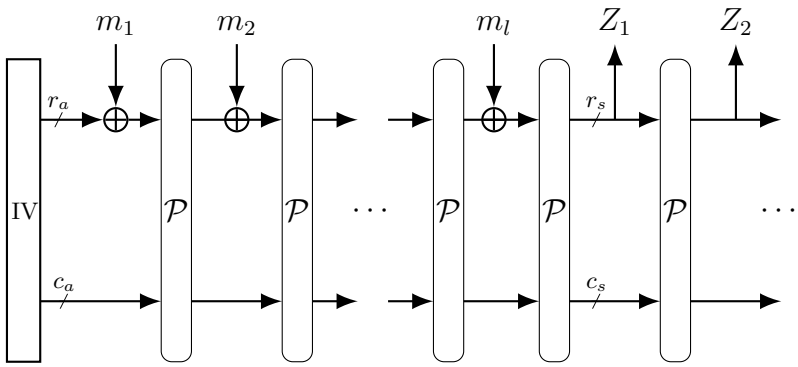


Fig. 1: The sponge construction.

number of underlying permutation evaluations required to compute the digest. Following the approach of Bertoni et al. [BDPV08], we measure the cost of a construction query based on the number of permutation evaluations required in the real world to produce the output. More precisely, if $|pad(M)| = l \times r_a$, then a query with input M and k has a cost of $l + k$. We define the construction query history $\mathcal{Q}_C$ as an ordered list, which contains elements of the form (M, k, Z). In the real world, Z corresponds to the $(k+1)^{\text{th}}$ squeezed block after having absorbed M. In the ideal world, Z equals $\mathcal{RO}(M)[r_s \times k : r_s \times (k+1)]$.

Indifferentiability of the sponge when restricting the number of message blocks. When the number of message blocks is restricted to ℓ, we impose that the distinguisher can make construction queries only with messages M such that $|pad(M)| \leq \ell \times r_a$. More formally, denote by $\texttt{RDist}[q_{\mathcal{P}}, q_C, \ell]$ the set of distinguishers that satisfy the constraint above making at most $q_{\mathcal{P}}$ primitive queries and construction queries with a total cost of at most q_C. The restricted indifferentiability (resp., public indifferentiability) advantage of the sponge from a random oracle is defined as follows:

$$\mathbf{Adv}_{\mathcal{H}}^{\text{R-iff}}(q_{\mathcal{P}}, q_C, \ell) = \sup_{\mathcal{D} \in \texttt{RDist}[q_{\mathcal{P}}, q_C, \ell]} \mathbf{Adv}_{\mathcal{H}}^{\text{iff}}(\mathcal{D}) \ ,$$

$$\mathbf{Adv}_{\mathcal{H}}^{\text{R-pubiff}}(q_{\mathcal{P}}, q_C, \ell) = \sup_{\mathcal{D} \in \texttt{RDist}[q_{\mathcal{P}}, q_C, \ell]} \mathbf{Adv}_{\mathcal{H}}^{\text{pubiff}}(\mathcal{D}) \ .$$

3.3 Graph Construction of the Sponge with Restricted Message Blocks

We adapt the graph representation from [BDPV08] to our setting. This graph is derivable from a primitive query history $\mathcal{Q}$. The nodes are all elements in $\{0,1\}^b$. For $X, Y \in \{0,1\}^b, m \in \{0,1\}^{r_a}$, we write $X \xrightarrow{m} Y$ whenever there exists $\mathrm{d} \in \{\mathrm{fwd}, \mathrm{inv}\}$ such that $(X \oplus (m \| 0^{c_a}), Y, \mathrm{d}) \in \mathcal{Q}$; and $X \to Y$ whenever $X \xrightarrow{0^{r_a}} Y$. The set `ValidPaths` gathers the nodes involved in paths, i.e.,

$$\texttt{ValidPaths} := \Big\{ \mathrm{IV} \xrightarrow{m_1} N_{A,1} \xrightarrow{m_2} \cdots \xrightarrow{m_l} N_{S,1} \to N_{S,2} \to \cdots \to N_{S,k} \mid m_l \neq 0^{r_a} \wedge k \in \mathbb{N} \setminus \{0\} \wedge l \leq \ell \Big\}.$$

The set `Rooted` includes all the nodes which appear in valid paths. The set `rRooted` is a compact way of characterizing the paths. Informally, if the simulator receives a forward query with X, and $(X, M, k) \in \texttt{rRooted}$, and $\mathit{unpad}(M) \neq \perp$, then in order to produce a consistent answer with respect to the random oracle, the answer Y must satisfy $\mathrm{outer}_{r_s}(Y) = \mathcal{RO}(\mathit{unpad}(M))[r_s \times k : r_s \times (k+1)]$. Finally, we define the set `AbsorbPath` as follows:

$$\texttt{AbsorbPath} = \{\mathrm{IV}\} \cup \Big\{ Y \mid \exists l < \ell, m_1, \ldots, m_l \in \{0,1\}^{r_a}, N_{A,1}, \ldots, N_{A,l-1} \in \{0,1\}^b \text{ such that } \mathrm{IV} \xrightarrow{m_1} N_{A,1} \xrightarrow{m_2} \cdots \xrightarrow{m_l} Y \Big\}.$$

In other words this set gathers the nodes where absorption of a message block is still possible. Note that the obtained path is not necessarily a valid path, since the message blocks $m_1, \ldots, m_l$ can all be equal to 0^{r_a}. One important remark is that, as long as no inner collisions occur with nodes in `AbsorbPath`,

$$|\texttt{AbsorbPath}| \leq \min \Big\{ q+1, 2 \times 2^{(\ell-1) \times r_a} \Big\}, \tag{2}$$

where $q := |\mathcal{Q}|$. Looking ahead, `AbsorbPath` contains the only nodes where inner collisions are dangerous.

We will see later that in some situations, the simulator has a query history different from the primitive query history of the distinguisher. Therefore, when this is not clear from the context, for $\texttt{Set} \in \{\texttt{ValidPaths}, \texttt{Rooted}, \texttt{rRooted}, \texttt{AbsorbPath}\}$ we use the notation $\texttt{Set}(\mathcal{Q})$ to clarify which query history is used to build `Set`.

3.4 Differentiability Attack on the Sponge Construction

In this section we describe a differentiability attack on the (plain) sponge [BDPV07]. Let $k = \frac{c_a}{2r_a}$, and for simplicity, assume that k is an integer. Consider the following distinguisher $\mathcal{D}$:

1. $\mathcal{D}$ first makes all possible k first absorb calls with primitive queries. It obtains $2^{k \times r_a}$ different states $(Y_i)_{i \in [\![1, 2^{k \times r_a}]\!]}$ such that
$$\mathrm{IV} \xrightarrow{m_1} N_{m_1} \xrightarrow{m_2} N_{m_2} \to \cdots \xrightarrow{m_k} Y_i\,;$$
2. If the simulator emulates well a permutation,[2] then with high probability there exist $i \neq j$ such that $Y_i \overset{c_a}{=} Y_j$ and neither $\mathrm{outer}_{r_a}(Y_i)$ nor $\mathrm{outer}_{r_a}(Y_j)$ is equal to 0^{r_a}. Let M_i (resp., M_j) be the concatenation of all message blocks used to reach Y_i (resp., Y_j);
3. The distinguisher makes the construction queries associated to $\mathit{unpad}(M_i \| \mathrm{outer}_{r_a}(Y_i))$ and $\mathit{unpad}(M_j \| \mathrm{outer}_{r_a}(Y_j))$. If both outputs coincide, then the distinguisher returns "real", otherwise "ideal".

[2] If this is not the case, then we can use another distinguisher which exploits this difference of behavior.

In the real world, two sequences of message blocks that lead to the same state have the same digest, while in the ideal world, the outputs are independent. $\mathcal{D}$ succeeds with high probability, and makes $2^{k \times r_a} = 2^{c_a/2}$ queries. However, this attack requires at least $k+1$ absorb calls. Thus, whenever $k+1 > \ell$, this attack cannot be applied to the sponge when restricting the number of message blocks to ℓ.

4 Indifferentiability of the Sponge when Restricting the Message Blocks

In this section, we prove indifferentiability of the sponge when restricting the number of message blocks, as stated in Theorem 1.

Theorem 1. *Let $\mathcal{H}$ be the sponge construction, and b, c_a, r_a, c_s, r_s as described in Section 3.1. Then one has*

$$\mathbf{Adv}_{\mathcal{H}}^{\mathrm{R-iff}}(q_C, q_{\mathcal{P}}, \ell) \leq \frac{q_{\mathcal{P}}\left(12\ln(q) + 12r_s + 12\right)}{2^{c_s}} + \frac{3q^2}{2^b} + \min\left\{\frac{10q(q+1)}{2^{c_a}}, \frac{20q}{2^{b-\ell\times r_a}}\right\},$$

where $q := q_{\mathcal{P}} + q_C$.

Interpretation of the bound. In the following, we ignore logarithmic factors. When the maximum number of message blocks absorbed is not a limiting factor (i.e., when $c_a/2 \leq r_a(\ell-1)$), we obtain an upper bound of the form

$$\mathcal{O}\left(\frac{q}{2^{c_s}} + \frac{q^2}{2^{c_a}}\right),$$

which matches PHOTON indifferentiability bound when $c_s \geq c_a/2 + \log(c_a)$ [NO14]. On the other hand, when $c_a/2 > r_a(\ell-1)$, we obtain an upper bound of the form

$$\mathcal{O}\left(\frac{q}{2^{c_s}} + \frac{q^2}{2^b} + \frac{q}{2^{b-\ell\times r_a}}\right).$$

Moreover, when only one message block is absorbed (i.e., $\ell = 1$), our bound matches again the one from Naito and Ohta [NO14] by setting $r'' = r_a$ and $r = 0$ (recall that r'' is the rate of the first message block, and r is the rate when absorbing subsequent message blocks). To our knowledge, the bound provides new results whenever $c_a/2 > r_a \times (\ell-1)$ and when $\ell \neq 1$.

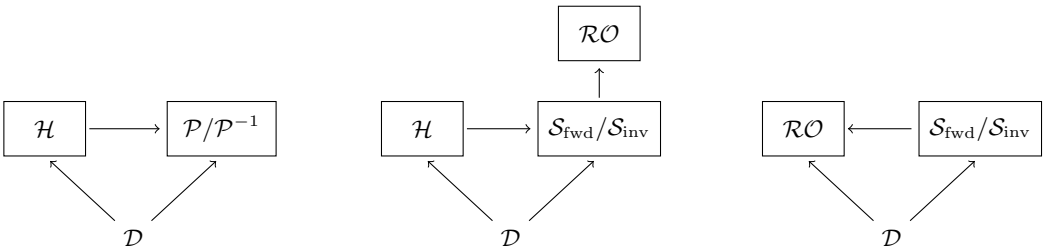
Outline. The remainder of this section is organized as follows. Section 4.1 introduces the simulator used for the proof, Section 4.2 introduces an intermediate world, Section 4.3 lists the bad events, Section 4.4 splits and computes the probabilities, and Section 4.5 adapts the result to the setting of public indifferentiability.

4.1 Simulator Definition

The simulator $\mathcal{S}$ is described algorithmically in Algorithm 2. On a high-level view, $\mathcal{S}$ keeps track of the graph construction from Section 3.1 to ensure that its responses are consistent with the random oracle. To do that, it logs the query history in an ordered list called $\mathcal{Q}_{\mathrm{Sim}}$. For $i \in \mathbb{N}$, $\mathcal{Q}_{\mathrm{Sim}}[i]$ contains a tuple (X, Y, d) which corresponds to the ith query. For any fresh query, $\mathcal{S}$ behaves as a random function. We could have a simulator which outputs permutation-consistent responses, but this would not improve significantly the bound. More precisely, the term $\frac{q^2}{2^b}$ in the bound of Theorem 1 does not only come from the PRP/PRF switching lemma, but appears at other places. For example, at the end of Section 5 we present an attack with a cost of $\approx 2^{b/2}$ queries which would succeed even when using a permutation-consistent variant of $\mathcal{S}$.

Algorithm 2: Simulator definition. `Init` () is run once at the beginning of the game, $\mathcal{S}_{\text{fwd}}$ (resp., $\mathcal{S}_{\text{inv}}$) is the algorithm run by $\mathcal{S}$ on forward (resp., inverse) queries. The inputs X and Y are b-bit strings.

```
Function Init ():
    Q_Sim ← ∅;
    /* ϵ does not correspond to a valid padding, but this initialization allows
    to take into account paths of form IV --0^{k r_a}--> Y */;
    rRooted ← {(IV, ϵ, 1)};
end
Function S_fwd (X):
    if ∃Y ∈ {0,1}^b, d ∈ {fwd, inv} such that (X, Y, d) ∈ Q_Sim then
        return Y;
    end
    /* If ∃ IV --M||0^{(k-1) r_a}--> X ⊕ (m||0^{c_a}) where X ⊕ (m||0^{c_a}) ∈ AbsorbPath */;
    if ∃M ∈ {0,1}*, m ∈ {0,1}^{r_a} \ {0^{r_a}}, N ∈ {0,1}^b, k ∈ ℕ such that (N, M, k) ∈ rRooted
      and X = N ⊕ (m||0^{c_a}) and k + |M|/r_a < ℓ then
        M' ← M||0^{(k-1) r_a}||m ;
        y_r ←$ {0,1}^{r_s};
        if unpad(M') ≠ ⊥ then
            y_r ← RO (unpad(M')) [0 : r_s];
        end
        y_c ←$ {0,1}^{c_s};
        Y ← y_r||y_c;
        Q_Sim ← Q_Sim ∪ {(X, Y, fwd)};
        rRooted ← rRooted ∪ {(Y, M', 1)};
        return Y;
    end
    else if ∃M ∈ {0,1}*, k ∈ ℕ such that (X, M, k) ∈ rRooted then
        y_r ←$ {0,1}^{r_s};
        if unpad(M) ≠ ⊥ then
            y_r ← RO (unpad(M)) [r_s × k : r_s × (k + 1)];
        end
        y_c ←$ {0,1}^{c_s};
        Y ← y_r||y_c;
        Q_Sim ← Q_Sim ∪ {(X, Y, fwd)};
        rRooted ← rRooted ∪ {(Y, M, k + 1)};
        return Y;
    end
    Y ←$ {0,1}^b;
    Q_Sim ← Q_Sim ∪ {(X, Y, fwd)};
    return Y;
end
Function S_inv (Y):
    if ∃X ∈ {0,1}^b, d ∈ {fwd, inv} such that (X, Y, d) ∈ Q_Sim then
        return X;
    end
    X ←$ {0,1}^b;
    Q_Sim ← Q_Sim ∪ {(X, Y, inv)};
    return X;
end
```



(a) Real world W_1. (b) Intermediate world W_2. (c) Ideal world W_3.

Fig. 2: Definition of the different worlds. $\mathcal{P}$ returns responses with lazy sampling.

4.2 World Decomposition

Denote by W_1 the real world, i.e., implemented by $(\mathcal{H}^{\mathcal{P}}, \mathcal{P})$ as shown in Fig. 2a, and let W_3 be the ideal world giving access to $(\mathcal{RO}, \mathcal{S})$ (see Fig. 2c). We introduce an intermediate world W_2, depicted in Fig. 2b. This world gives access to $(\mathcal{H}^{\mathcal{S}}, \mathcal{S})$, where the simulator is based on a random oracle hidden from the adversary. This decomposition, done among others in the proof of PHOTON indifferentiability [NO14], allows to separate the quality of randomness of the simulator from the consistency of the answers.

W_1 versus W_2. As detailed more formally in Section 4.4.2, the difference between W_1 and W_2 lies in the difference of randomness. More precisely, the ideal world implements a perfectly random permutation, while the simulator behaves like a two-sided random function. This is a well known result, and is upper bounded by the PRP/PRF switching lemma.

W_2 versus W_3. The simulator from world W_2 has a priori more knowledge than the one from W_3, since it indirectly knows the construction queries made by the adversary. In Section 4.3, we define a bad event **GUESS** which prevents the adversary from exploiting this difference of behavior. Moreover, we have to guarantee that the simulator behaves consistently according to the random oracle, i.e., whenever one can compute $\mathcal{H}^{\mathcal{P}}(M, k)$ from the query history $\mathcal{Q}_{\mathcal{P}}$, then it coincides with $\mathcal{RO}(M)[0 : r_s \times k]$. We will define additional bad events to capture inconsistent answers. These bad events, including **GUESS**, are given in Section 4.3.

4.3 Bad Events

In this section, we formally define bad events over the query history of the simulator. For the sake of the proof, we define an enhanced simulator query history $\text{Ext}(\mathcal{Q}_{\text{Sim}})$ that contains additionally the origin $\text{O} \in \{\text{C}, \text{D}\}$ of the query, i.e., elements in $\text{Ext}(\mathcal{Q}_{\text{Sim}})$ are of the form $(X, Y, \text{d}, \text{O})$, where "C" indicates that the query was made by the construction and "D" by the adversary. Note that the simulator has no access to this list, it is introduced only to rigorously analyze the security. This enhanced query history is only useful for W_2. We also define $\text{Fresh}(\mathcal{Q}_{\text{Sim}})$, which is the query history without duplicated queries. More precisely, the ith element of $\text{Fresh}(\mathcal{Q}_{\text{Sim}})$ is a tuple $(X_i, Y_i, d_i) \in \mathcal{Q}_{\text{Sim}}$ such that there is no $j < i$ such that $\text{Fresh}(\mathcal{Q}_{\text{Sim}})[j] = (X_i, Y_i, d)$ for $d \in \{\text{fwd}, \text{inv}\}$. Let σ be the total number of queries made to the simulator, so that $\sigma = q_C + q_{\mathcal{P}}$ in W_2, $\sigma = q_{\mathcal{P}}$ in W_3. Moreover, let $\bar{\sigma}$ be the total number of *fresh* queries to the simulator, so that $\bar{\sigma} \leq \sigma$. For simplicity of notation, for $i \in [\![1, \bar{\sigma}]\!]$, (X_i, Y_i) denotes the first two elements in $\text{Fresh}(\mathcal{Q}_{\text{Sim}})[i]$. We define the following bad events:

$$
\begin{aligned}
\textbf{GUESS_SQU} : & \exists i \in [\![1,\sigma]\!] \text{ such that } \mathrm{Ext}(\mathcal{Q}_{\mathrm{Sim}})[i] = (X, Y, \mathrm{fwd}, \mathrm{D}) \text{ and} \\
& X \notin \mathtt{Rooted}(\mathcal{Q}_{\mathcal{P}}[0:i-1]) \text{ and } \exists j < i \text{ such that} \\
& \mathrm{Ext}(\mathcal{Q}_{\mathrm{Sim}})[j] = (X', Y', \mathrm{fwd}, \mathrm{C}) \text{ and } X = X' \\
\text{or } & \exists i \in [\![1,\sigma]\!] \text{ such that } \mathrm{Ext}(\mathcal{Q}_{\mathrm{Sim}})[i] = (X, Y, \mathrm{inv}, \mathrm{D}) \text{ and} \\
& \exists j < i \text{ such that } \mathrm{Ext}(\mathcal{Q}_{\mathrm{Sim}})[j] = (X', Y', \mathrm{fwd}, \mathrm{C}) \text{ and } Y = Y' , \\
\textbf{GUESS_ABS} : & \exists i \in [\![1,\sigma]\!] \text{ such that } \mathrm{Ext}(\mathcal{Q}_{\mathrm{Sim}})[i] = (X, Y, \mathrm{fwd}, \mathrm{D}) \text{ and} \\
& X \notin \mathtt{Rooted}(\mathcal{Q}_{\mathcal{P}}[0:i-1]) \text{ and } \exists j < i \text{ such that} \\
& \mathrm{Ext}(\mathcal{Q}_{\mathrm{Sim}})[j] = (X', Y', \mathrm{fwd}, \mathrm{C}), Y' \in \mathtt{AbsorbPath}(\mathcal{Q}_{\mathrm{Sim}}), \text{ and } X \overset{c_a}{=} Y' , \\
\textbf{GUESS} : & \textbf{GUESS_SQU} \vee \textbf{GUESS_ABS} ,
\end{aligned}
$$

$$
\begin{aligned}
\textbf{COL} : & \exists i \in [\![1,\bar{\sigma}]\!] \text{ such that } \mathrm{Fresh}(\mathcal{Q}_{\mathrm{Sim}})[i] = (X_i, Y_i, \mathrm{fwd}) \text{ and} \\
& \exists j < i \text{ such that } Y_i = Y_j \\
\text{or } & \exists i \in [\![1,\bar{\sigma}]\!] \text{ such that } \mathrm{Fresh}(\mathcal{Q}_{\mathrm{Sim}})[i] = (X_i, Y_i, \mathrm{inv}) \text{ and} \\
& \exists j < i \text{ such that } X_i = X_j , \\
\textbf{CONNECT} : & \exists i \in [\![1,\bar{\sigma}]\!] \text{ such that } \mathrm{Fresh}(\mathcal{Q}_{\mathrm{Sim}})[i] = (X_i, Y_i, \mathrm{fwd}) \text{ and} \\
& \exists j < i \text{ such that } Y_i = X_j \\
\text{or } & \exists i \in [\![1,\bar{\sigma}]\!] \text{ such that } \mathrm{Fresh}(\mathcal{Q}_{\mathrm{Sim}})[i] = (X_i, Y_i, \mathrm{inv}) \text{ and} \\
& \exists j < i \text{ such that } X_i = Y_j ,
\end{aligned}
$$

$$
\begin{aligned}
\textbf{INNER}^1 : & \exists i \in [\![1,\bar{\sigma}]\!], m \in \{0,1\}^{r_a} \text{ such that } \mathrm{Fresh}(\mathcal{Q}_{\mathrm{Sim}})[i] = (X_i, Y_i, \mathrm{fwd}) \text{ and} \\
& X_i \oplus (m \| 0^{c_a}), Y_i \in \mathtt{AbsorbPath}(\mathrm{Fresh}(\mathcal{Q}_{\mathrm{Sim}})) \text{ and} \\
& \exists j < i \text{ such that either } Y_i \overset{c_a}{=} X_j \text{ or } Y_i \overset{c_a}{=} Y_j , \\
\textbf{INNER}^2 : & \exists i \in [\![1,\bar{\sigma}]\!] \text{ such that } \mathrm{Fresh}(\mathcal{Q}_{\mathrm{Sim}})[i] = (X_i, Y_i, \mathrm{fwd}) \text{ and } \exists Y \in \{0,1\}^b \\
& \text{such that } Y \in \mathtt{AbsorbPath}(\mathrm{Fresh}(\mathcal{Q}_{\mathrm{Sim}})[0:i-1]) \text{ and } Y_i \overset{c_a}{=} Y \\
\text{or } & \exists i \in [\![1,\bar{\sigma}]\!] \text{ such that } \mathrm{Fresh}(\mathcal{Q}_{\mathrm{Sim}})[i] = (X_i, Y_i, \mathrm{inv}) \text{ and } \exists Y \in \{0,1\}^b \\
& \text{such that } Y \in \mathtt{AbsorbPath}(\mathrm{Fresh}(\mathcal{Q}_{\mathrm{Sim}})[0:i-1]) \text{ and } X_i \overset{c_a}{=} Y , \\
\textbf{INNER} : & \textbf{INNER}^1 \vee \textbf{INNER}^2 .
\end{aligned}
$$

Moreover, let

$$\textbf{BAD} = \textbf{GUESS} \vee \textbf{COL} \vee \textbf{CONNECT} \vee \textbf{INNER} .$$

Interpretation. **GUESS** is an event that can be set only in W_2. It reflects the fact that the distinguisher is able to enter in the middle of a path without having made all the queries before that path. To do that, the adversary has two different possibilities. The first one, corresponding to **GUESS_SQU**, consists of making use of the outer parts disclosed by the construction queries. In other words, for a fixed $u \in \{0,1\}^{r_s}$, if there exist x construction queries which output is u, then a primitive query with input $u \| w$ has a probability of $\approx \dfrac{x}{2^{c_s}}$ to set **GUESS_SQU**. The second strategy corresponds to **GUESS_ABS**. It targets the nodes in `AbsorbPath`, and only concerns forward queries. In this setting, besides having access to r_s bits, the adversary is allowed to overwrite up to r_a bits, as illustrated in Fig. 3. As long as **GUESS** is not set, the adversary is not aware of paths where the simulator in W_2 has more knowledge than the one from W_3. The remaining bad events **COL**, **CONNECT**, and **INNER** concern the consistency of the answers with respect to the random oracle, i.e., when a path connects to a node or two paths collide without the simulator being able to ensure consistency. **COL** flags full-state

collisions, and **CONNECT** is set when the order of the queries is not respected. The event **INNER** flags both potential full-state collisions and inconsistent queries that involve nodes in `AbsorbPath`. Note that the event of a query hitting the IV is captured by the event $\mathbf{INNER}^2$. Finally, **COL** is also useful to bound the distance between W_1 and W_2.

4.4 Probability Splitting and Computation

Remember that $q := q_{\mathcal{P}} + q_C$. For $i \in [\![1, \sigma]\!]$ and any event $\mathbf{evt}$, $\mathbf{evt}_i$ denotes that $\mathbf{evt}$ is triggered after the first i queries. Moreover, for $W_x \in \{W_1, W_2, W_3\}$, denote by $\mathbf{Pr}_{W_x}(\mathbf{evt})$ the probability that $\mathbf{evt}$ is set in world W_x.

4.4.1 Outline. Let $\mathcal{D}$ be any distinguisher, we have

$$\begin{aligned}\mathbf{Adv}^{\mathrm{iff}}_{\mathcal{H}}(\mathcal{D}) &= \left|\mathbf{Pr}\left(\mathcal{D}^{W_1} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{W_3} = 1\right)\right| \\ &\leq \left|\mathbf{Pr}\left(\mathcal{D}^{W_1} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{W_2} = 1\right)\right| + \qquad (3)\\ &\quad \left|\mathbf{Pr}\left(\mathcal{D}^{W_2} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{W_3} = 1\right)\right| . \qquad (4)\end{aligned}$$

Looking ahead, we will show in Section 4.4.3 that

$$\left|\mathbf{Pr}\left(\mathcal{D}^{W_2} = 1 \mid \neg\mathbf{BAD}\right) - \mathbf{Pr}\left(\mathcal{D}^{W_3} = 1 \mid \neg\mathbf{BAD}\right)\right| = 0 . \qquad (5)$$

Using this fact, denoting $P := \mathbf{Pr}\left(\mathcal{D}^{W_2} = 1 \mid \neg\mathbf{BAD}\right) = \mathbf{Pr}\left(\mathcal{D}^{W_3} = 1 \mid \neg\mathbf{BAD}\right)$, the distance (4) can be decomposed as follows:

$$\begin{aligned}(4) &= \left|\mathbf{Pr}\left(\mathcal{D}^{W_2} = 1 \wedge \mathbf{BAD}\right) - \mathbf{Pr}\left(\mathcal{D}^{W_3} = 1 \wedge \mathbf{BAD}\right) + \right.\\ &\quad \left.\mathbf{Pr}\left(\mathcal{D}^{W_2} = 1 \wedge \neg\mathbf{BAD}\right) - \mathbf{Pr}\left(\mathcal{D}^{W_3} = 1 \wedge \neg\mathbf{BAD}\right)\right| \\ &= \left|\mathbf{Pr}\left(\mathcal{D}^{W_2} = 1 \wedge \mathbf{BAD}\right) - \mathbf{Pr}\left(\mathcal{D}^{W_3} = 1 \wedge \mathbf{BAD}\right) + \right.\\ &\quad \left.P\left(1 - \mathbf{Pr}\left(\mathcal{D}^{W_2} \text{ sets } \mathbf{BAD}\right)\right) - P\left(1 - \mathbf{Pr}\left(\mathcal{D}^{W_3} \text{ sets } \mathbf{BAD}\right)\right)\right| \\ &= \left|\mathbf{Pr}\left(\mathcal{D}^{W_2} \text{ sets } \mathbf{BAD}\right)\left(\mathbf{Pr}\left(\mathcal{D}^{W_2} = 1 \mid \mathbf{BAD}\right) - P\right) - \right.\\ &\quad \left.\mathbf{Pr}\left(\mathcal{D}^{W_3} \text{ sets } \mathbf{BAD}\right)\left(\mathbf{Pr}\left(\mathcal{D}^{W_3} = 1 \mid \mathbf{BAD}\right) - P\right)\right| \\ &\leq 2 \times \max\left\{\mathbf{Pr}\left(\mathcal{D}^{W_2} \text{ sets } \mathbf{BAD}\right), \mathbf{Pr}\left(\mathcal{D}^{W_3} \text{ sets } \mathbf{BAD}\right)\right\} . \qquad (6)\end{aligned}$$

Therefore, the advantage of the adversary can be upper bounded as the sum of (3) and (6). The remainder of the proof is organized as follows. Section 4.4.2 is dedicated to upper bounding the distance in (3), then Section 4.4.3 shows the claim made in (5), i.e., the distance between W_2 and W_3 conditioned on $\neg\mathbf{BAD}$ is zero, Section 4.4.4 upper bounds (6), and we conclude in Section 4.4.5.

4.4.2 W_1 Versus W_2. Because the padding is prefix-free, the $\mathcal{RO}$ calls in lines 13 and 24 of Algorithm 2 are never accessed twice for the same entry. Therefore, the simulator behaves exactly as a random permutation as long as **COL** is not set in W_2. Therefore, by the fundamental lemma of game playing [BR06],

$$\left|\mathbf{Pr}\left(\mathcal{D}^{W_1} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{W_2} = 1\right)\right| \leq \mathbf{Pr}\left(\mathcal{D}^{W_2} \text{ sets } \mathbf{COL}\right) .$$

Moreover, for any $i \in [\![1, q]\!]$, one has

$$\mathbf{Pr}_{W_2}\left(\mathbf{COL}_i \mid \neg\mathbf{COL}_{i-1}\right) \leq \frac{i-1}{2^b} .$$

Therefore

$$(3) \leq \mathbf{Pr}\left(\mathcal{D}^{W_2} \text{ sets } \mathbf{COL}\right) \leq \frac{q^2}{2^{b+1}} . \qquad (7)$$

4.4.3 W_2 Versus W_3 as Long as no Bad.

The only difference between W_2 and W_3 lies in the consistency of the responses: the construction oracle in W_2 calls the simulator, while in W_3, the answers come from a random oracle. As explained in Section 4.3, $\neg$**GUESS** prevents the distinguisher from being aware of paths where the simulator from W_2 has more knowledge than the one in W_3. Therefore, with $\neg$**GUESS**, we can assume that both simulators provide the same level of consistency, and w.l.o.g., we focus on the simulator from W_3. In this world, the construction queries come from a random oracle and $\mathcal{Q}_{\text{Sim}}$ coincides with the primitive query history of the distinguisher $\mathcal{Q}_{\mathcal{P}}$. Let $i \in [\![1, q_{\mathcal{P}}]\!]$, assume that the query number i gives $\mathcal{Q}_{\mathcal{P}}[i] := (X_i, Y_i, \mathrm{d}_i)$. Assume by contradiction that this query breaks the consistency, while **BAD** is not set. We split the cases depending on the direction of the query.

Forward query. In order to break the consistency, the query must be involved in a valid path. There are two possibilities, depending on whether $X_i \in \texttt{AbsorbPath}$ or not:

(i) There exists a valid path

$$\text{IV} \xrightarrow{m_1} N_{A,1} \xrightarrow{m_2} \cdots \xrightarrow{m_l} N_{S,1} \to N_{S,2} \to \cdots \to N_{S,k} \to X_i \to Y_i\,,$$

with $m_l \neq 0^{r_a}$. In other words, X_i is involved in a path during the squeezing phase. Note that we can have $k = 0$. Let $k' := k + 1$.

(ii) There exists a valid path

$$\text{IV} \xrightarrow{m_1} N_{A,1} \xrightarrow{m_2} \cdots \xrightarrow{m_{l-2}} N_{A,l-2} \xrightarrow{m_{l-1}} X_i \oplus (m_l \| 0^{c_a}) \xrightarrow{m_l} Y_i$$

where $m_l \neq 0^{r_a}$. Concretely, it means that $X_i \in \texttt{AbsorbPath}$. In that case, let $k' := 0$.

Let $M := m_1 \| \cdots \| m_l$. In both cases, $(X_i, M, k') \in \texttt{rRooted}(\mathcal{Q}_{\mathcal{P}})$. Because of $\neg$**BAD**, the existence of such a path is unique. Indeed, if X_i is involved in two different paths, then it implies that either a full-state collision or a collision with an element in $\texttt{AbsorbPath}$ occurred, which are prevented by respectively $\neg$**COL** and $\neg$**INNER**. Thus line 21 of the algorithm of $\mathcal{S}_{\text{fwd}}$ (Algorithm 2) is satisfied for this particular M and k', and by construction of the simulator the answer Y is consistent with respect to $\mathcal{RO}$, i.e., $\text{outer}_{r_s}(Y) = \mathcal{RO}\,(\mathit{unpad}(M))\,[r_s \times k' : r_s \times (k'+1)]$. Moreover, thanks to $\neg$**CONNECT** $\wedge$ $\neg$**INNER**, there are no edges starting from Y_i resulting to a valid path. Therefore, this query cannot break the consistency.

Inverse query. An inverse query breaking the consistency implies that X_i connects either to the IV, or to an already existing path such that the newly created path is valid. These two cases are prevented by $\neg$**COL** $\wedge$ $\neg$**INNER** $\wedge$ $\neg$**CONNECT**.

Conclusion. As long as **BAD** is not set, all queries to $\mathcal{H}^{\mathcal{S}}$ and to $\mathcal{RO}$ give the same output, and both simulators exhibit the same behavior. Therefore,

$$\left|\mathbf{Pr}\left(\mathcal{D}^{W_3} = 1 \mid \neg\mathbf{BAD}\right) - \mathbf{Pr}\left(\mathcal{D}^{W_2} = 1 \mid \neg\mathbf{BAD}\right)\right| = 0\,. \tag{8}$$

4.4.4 Probability of BAD.

Basic reasoning. In W_3, the adversary can learn information about the randomness used by the simulator from the construction queries *before* making the underlying primitive query. This can make the probability computation quite hard to evaluate. Fortunately, in W_3, the bad events are **COL**, **CONNECT**, and **INNER**, and they only concern the (fresh) query history of the simulator. We can thus consider a more powerful adversary $\mathcal{D}'$, which replaces its construction queries by primitive queries. The probability that $\mathcal{D}'$ sets **BAD** is no smaller in W_2 than in W_3. Moreover, the only meaningful metrics for **COL**, **CONNECT**, and **INNER** is the total number of queries made to the simulator. Thus the query cost of $\mathcal{D}'$ in W_2 does not change

compared to the one of $\mathcal{D}$, since it makes 0 construction queries and at most $q_{\mathcal{P}} + q_C$ primitive queries. Therefore,

$$\mathbf{Pr}_{W_3}(\mathcal{D} \text{ sets } \mathbf{BAD}) \leq \mathbf{Pr}_{W_3}(\mathcal{D}' \text{ sets } \mathbf{BAD}) \leq \mathbf{Pr}_{W_2}(\mathcal{D}' \text{ sets } \mathbf{BAD}) \,. \tag{9}$$

Thus from now, we can focus on the probability computation in W_2. By basic probability, we have

$$\begin{aligned}\mathbf{Pr}_{W_2}(\mathbf{BAD}) &\leq \sum_{i=1}^{q} \mathbf{Pr}_{W_2}(\mathbf{BAD}_i \mid \neg\mathbf{BAD}_{i-1}) \\ &\leq \sum_{i=1}^{q} \Big(\mathbf{Pr}_{W_2}(\mathbf{GUESS}_i \mid \neg\mathbf{BAD}_{i-1}) + \mathbf{Pr}_{W_2}(\mathbf{COL}_i \mid \neg\mathbf{BAD}_{i-1}) \\ &\quad + \mathbf{Pr}_{W_2}(\mathbf{CONNECT}_i \mid \neg\mathbf{BAD}_{i-1}) + \mathbf{Pr}_{W_2}(\mathbf{INNER}_i \mid \neg\mathbf{BAD}_{i-1})\Big) \,.\end{aligned} \tag{10}$$

We evaluate each term individually.

GUESS_SQU$_i$. Setting this event is similar to a guessing game: in order to win, the adversary must be able to guess a node N in a valid path. Thanks to the construction oracle, $\mathcal{D}$ it has access to the r_s upper bits of q_C different nodes. Let $u \in \{0,1\}^{r_s}$, we define, similarly to Choi et al. [CLL19], the random variable F_u as follows:

$$F_u := |\{(x, y, \text{fwd}, C) \in \text{Ext}(\mathcal{Q}_{\text{Sim}}) \mid \text{outer}_{r_s}(y) = u\}| \,,$$

i.e., F_u is the number of construction queries which outer part hit u. The distribution of the random variables $(F_u)_{u\in\{0,1\}^{r_s}}$ is the same as the bin-and-balls experiment described in Lemma 1. Now, given a query $v\|w$ with $v \in \{0,1\}^{r_s}$ and $w \in \{0,1\}^{c_s}$, the probability that **GUESS_SQU**$_i$ is set, conditioned on the query history of the $i-1$ previous queries $\mathcal{Q}_{i-1}$, is upper bounded by

$$\frac{\max_{u\in\{0,1\}^{r_s}} F_u}{2^{c_s}} \,.$$

Therefore, by summing over all possible $\mathcal{Q}_{i-1}$ we obtain

$$\mathbf{Pr}_{W_2}(\mathbf{GUESS_SQU}_i \mid \neg\mathbf{BAD}_{i-1}) \leq \frac{\mathsf{E}\left(\max_{u\in\{0,1\}^{r_s}} F_u\right)}{2^{c_s}} \,.$$

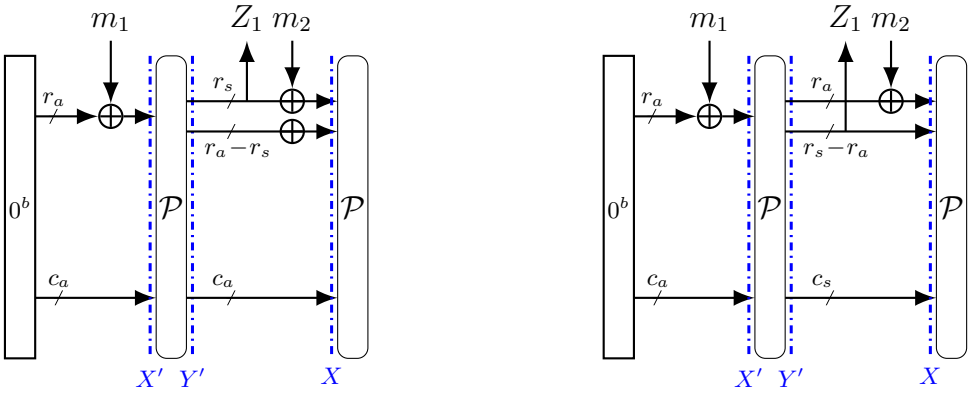
We can use Lemma 1, and obtain

$$\mathsf{E}\left(\max_{a\in\{0,1\}^{r_s}} F_u\right) \leq \frac{2q_C}{2^{r_s}} + 3r_s + 3\ln(q_C) + 3 \,,$$

so that

$$\mathbf{Pr}_{W_2}(\mathbf{GUESS_SQU}_i \mid \neg\mathbf{BAD}_{i-1}) \leq \frac{2q_C}{2^b} + \frac{3r_s + 3\ln(q_C) + 3}{2^{c_s}} \,. \tag{11}$$

GUESS_ABS$_i$. For the event **GUESS_ABS**$_i$, the situation is more complex. Indeed, assume first that $c_a = c_s$. Then in order to win, the adversary has to guess c_a bits. The r_a upper bits do not matter, since the adversary can overwrite them (see Fig. 3b, where $r_s - r_a = 0$). Therefore, in this setting the adversary is more powerful than in the case of the event **GUESS_SQU**$_i$. However, the number of nodes that the adversary can guess here is upper bounded by $|\texttt{AbsorbPath}|$, which is much smaller than q_C in some settings. Now, Fig. 3 illustrates the two different possibilities depending on the values of r_s and r_a. First, if $r_s < r_a$ (cf., Fig. 3a), the adversary has



(a) When $r_s < r_a$.

(b) When $r_s \geq r_a$.

Fig. 3: Illustration of the event **GUESS_ABS**. X' and Y' are nodes from a prior construction query, and the distinguisher makes a forward query with input X.

access to and can control r_s bits. It can add any bits to $r_a - r_s$ other bits, but has no access to them. Therefore, in this setting the success probability is upper bounded by

$$\frac{|\texttt{AbsorbPath}|}{2^{c_s}} \leq \frac{|\texttt{AbsorbPath}|}{2^{c_a}} .$$

On the other hand, if $r_s \geq r_a$ (cf., Fig. 3b), The adversary has access and can control r_a bits, has access to $r_s - r_a$ bits, and has to guess the c_s remaining ones. The situation therefore again boils down to a bin-and-balls game. This time, $|\texttt{AbsorbPath}|$ different balls are thrown in $R := 2^{r_s - r_a}$ bins. For $u \in [\![1, R]\!]$, let S_u be the number of balls in the bucket u. The probability to set $\mathbf{GUESS_ABS}_i$ conditioned on the query history is upper bounded by

$$\frac{\max_{u \in [\![1,R]\!]} S_u}{2^{c_s}} .$$

We can use again Lemma 1, and obtain

$$\mathsf{E}\left(\max_{a \in \{0,1\}^{r_s - r_a}} F_u\right) \leq \frac{2\,|\texttt{AbsorbPath}|}{2^{r_s - r_a}} + 3r_s + 3\ln(|\texttt{AbsorbPath}|) + 3 ,$$

so that for any choice of r_a and r_s,

$$\mathbf{Pr}_{W_2}\left(\mathbf{GUESS_ABS}_i \mid \neg\mathbf{BAD}_{i-1}\right) \leq \frac{2\,|\texttt{AbsorbPath}|}{2^{c_a}} + \frac{3r_s + 3\ln(|\texttt{AbsorbPath}|) + 3}{2^{c_s}} . \quad (12)$$

$\mathbf{COL}_i$ *and* $\mathbf{CONNECT}_i$. As long as $\mathbf{BAD}_{i-1}$ is not set, the remaining events can be set with a non-zero probability only if the query is fresh. Here, the adversary has no access to the randomness source used by the simulator. Thus, the probabilities of the bad events $\mathbf{COL}_i$ and $\mathbf{CONNECT}_i$ can be upper bounded as follows:

$$\mathbf{Pr}_{W_2}\left(\mathbf{COL}_i \mid \neg\mathbf{BAD}_{i-1}\right) \leq \frac{i-1}{2^b} ,$$
$$\mathbf{Pr}_{W_2}\left(\mathbf{CONNECT}_i \mid \neg\mathbf{BAD}_{i-1}\right) \leq \frac{i-1}{2^b} . \quad (13)$$

$\mathbf{INNER}_i$. For any query with a node in $\texttt{AbsorbPath}$, the probability that it sets $\mathbf{INNER}_i^1$ is upper bounded by $\frac{2q}{2^{c_a}}$. Therefore,

$$\mathbf{Pr}\left(\mathbf{INNER}_i^1 \mid \neg\mathbf{BAD}_{i-1}\right) \leq \mathbf{1}_{Y_i \in \texttt{AbsorbPath}} \times \frac{2q}{2^{c_a}} .$$

Moreover, the probability to set $\mathbf{INNER}_i^2$ is upper bounded as follows:

$$\mathbf{Pr}\left(\mathbf{INNER}_i^2 \mid \neg\mathbf{BAD}_{i-1}\right) \leq \frac{|\texttt{AbsorbPath}|}{2^{c_a}}.$$

Therefore,

$$\mathbf{Pr}\left(\mathbf{INNER}_i \mid \neg\mathbf{BAD}_{i-1}\right) \leq \mathbf{1}_{Y_i \in \texttt{AbsorbPath}} \times \frac{2q}{2^{c_a}} + \frac{|\texttt{AbsorbPath}|}{2^{c_a}}. \tag{14}$$

Wrap-up. Remark that the event **GUESS** can be set only with a primitive query from the distinguisher, so that the adversary has at most $q_{\mathcal{P}}$ attempts. Now, plugging (11) to (14) into (10) gives

$$\begin{aligned}
\mathbf{Pr}_{W_2}(\mathbf{BAD}) &\leq \sum_{i=1}^{q_{\mathcal{P}}} \left(\frac{2q_C}{2^b} + \frac{6r_s + 3\ln(q_C) + 3\ln(|\texttt{AbsorbPath}|) + 6}{2^{c_s}} + \frac{2\,|\texttt{AbsorbPath}|}{2^{c_a}} \right) \\
&\quad + \sum_{i=1}^{q} \left(\frac{2(i-1)}{2^b} + \mathbf{1}_{Y_i \in \texttt{AbsorbPath}} \times \frac{2q}{2^{c_a}} + \frac{|\texttt{AbsorbPath}|}{2^{c_a}} \right) \\
&\leq \frac{q_{\mathcal{P}}\,(6\ln(q) + 6r_s + 6)}{2^{c_s}} + \frac{5q\,|\texttt{AbsorbPath}|}{2^{c_a}} + \frac{q^2}{2^b} \\
&\leq \frac{q_{\mathcal{P}}\,(6\ln(q) + 6r_s + 6)}{2^{c_s}} + \frac{q^2}{2^b} + \min\left\{ \frac{5q(q+1)}{2^{c_a}}, \frac{10q}{2^{b-\ell\times r_a}} \right\},
\end{aligned} \tag{15}$$

where the penultimate inequality uses $2q_C q_{\mathcal{P}} \leq q^2$, and the last inequality uses (2). Note that we implicitly used $\neg\mathbf{BAD}_{i-1}$ at each step to upper bound $|\texttt{AbsorbPath}|$.

Finally, using (9), we can infer that the bound in (15) also applies in W_3, so that

$$(6) \leq \frac{q_{\mathcal{P}}\,(12\ln(q) + 12r_s + 12)}{2^{c_s}} + \frac{2q^2}{2^b} + \min\left\{ \frac{10q(q+1)}{2^{c_a}}, \frac{20q}{2^{b-\ell\times r_a}} \right\}. \tag{16}$$

4.4.5 Conclusion. Remember that in Section 4.4.1, we established that the distinguisher advantage is upper bounded by the sum of (3) and (6). These terms are themselves upper bounded in respectively (7) and (16), which concludes the proof. □

4.5 Public Indifferentiability

Public indifferentiability was introduced in [YMO09, DRS09] and refined in [MPS12]. As detailed in Section 2.3, public indifferentiability is a weaker security notion, which has nevertheless concrete applications. In the setting of public indifferentiability, the simulator is aware of all construction queries made by the distinguisher. We define a simulator $\mathcal{S}'$ which, when being informed of the construction query with input (M, k), where $(M, k, Y) \in \texttt{rRooted}$, makes the corresponding forward query to $\mathcal{S}$. In this setting, **GUESS** is no more a bad event, yet the remainder part of Section 4.4.3 remains unchanged. We can copy the proof of Section 4.4, but without using the bad event **GUESS**. This gives a bound

$$\mathbf{Adv}_{\mathcal{H}}^{\mathrm{R-pubiff}}(q_C, q_{\mathcal{P}}, \ell) \leq \frac{3q^2}{2^b} + \min\left\{ \frac{6q(q+1)}{2^{c_a}}, \frac{12q}{2^{b-\ell\times r_a}} \right\}.$$

In particular, the number of squeezed bits per construction query can be arbitrary.

5 Tightness of the Result

Our result guarantees indifferentiability up to $\min\left\{2^{c_s}, 2^{b/2}, \max\left\{2^{c_a/2}, 2^{b-\ell\times r_a}\right\}\right\}$ queries (up to logarithmic and constant factors). In the following, we show different attacks that apply to our simulator. The best strategy depends on the parameters used, as clarified in the following.

Attack in $\approx 2^{c_s}$ queries. Consider the following attack:

1. Take $M \xleftarrow{\$} \{0,1\}^{r_a} \setminus \{0^{r_a}\}$, make a construction query with the message M, obtain $u \in \{0,1\}^{r_s}$;
2. For all $v \in \{0,1\}^{c_s}$, query $\mathcal{P}^{-1}(u\|v)$;
3. If no response hits $\mathrm{IV} \oplus (M\|0^{c_a})$, then return "ideal", otherwise "real".

In the real world, the value $\mathrm{IV} \oplus (M\|0^{c_a})$ is hit with probability 1, while in the ideal world, it is with probability $\frac{1}{2^{r_s}}$. This attack costs $\approx 2^{c_s}$ queries.

Attack in $\approx 2^{b/2}$ queries. Because the simulator acts as a random function, a simple attack looking for collisions with primitive queries would give a high advantage after $2^{b/2}$ queries. Yet we want to show that, even when the simulator is permutation-consistent, going beyond the birthday barrier seems hard. Consider the following attack:

1. Make $2^{b/2}$ different forward primitive queries (not starting from the IV) to obtain $X_i \to Y_i$ for $i \in [\![1, 2^{b/2}]\!]$;
2. Choose a message $M \xleftarrow{\$} \{0,1\}^{r_a} \setminus \{0^{r_a}\}$, and expand the path starting from the IV with M by making $2^{b/2}$ forward primitive queries, so that we obtain the following valid path:

$$\mathrm{IV} \xrightarrow{M} N_1 \to \cdots \to N_{2^{b/2}}\,;$$

3. If **CONNECT** is set with the node N_k hitting X_i, then make the construction query associated to $\mathrm{IV} \xrightarrow{M} N_1 \to \cdots \to N_{k-1} \to X_i \to Y_i$. If the answer is consistent, return "real", otherwise "ideal".

Attack in $\approx \max\left\{2^{c_a/2}, 2^{b-\ell\times r_a}\right\}$ queries. When $c_a/2 \geq b - \ell r_a$, then it means that the number of absorbed message blocks is not limiting, so that the attack described in Section 3.4 applies. Otherwise, when $c_a/2 < b - \ell r_a$, consider the following attack:

1. Make all possible first $\ell - 1$ absorb calls, so that the list `AbsorbPath` is complete. This costs $\approx 2^{r_a \times (\ell-1)}$ queries;
2. Choose a message $M \xleftarrow{\$} \{0,1\}^{r_a} \setminus \{0^{r_a}\}$, and expand the path starting from the IV with M by making $2^{b-\ell\times r_a}$ forward primitive queries, so that we obtain the following valid path:

$$\mathrm{IV} \xrightarrow{M} N_1 \to \cdots \to N_{2^{b-\ell\times r_a}}\,;$$

3. With high probability there exists an inner collision on c_a bits between a node $Z \in$ `AbsorbPath` $\setminus \{N_1, \ldots, N_{\ell-1}\}$ and an N_i. If such a collision is found, use the last absorb call on the node Z to transform this partial collision into a full-state collision.

As explained in Section 3.4, such a collision would break the consistency with high probability in the ideal world. This attack costs $\max\left\{2^{r_a\times(\ell-1)}, 2^{b-\ell\times r_a}\right\} = 2^{b-\ell\times r_a}$ queries.

Applicability of the attacks to any simulator. In the case where $\min\{2^{r_a}, 2^{r_s}\}$ queries are unreachable in practice, then the first attack works for any simulator. Indeed, when "real" is returned in the ideal world, then it means that the simulator has found a preimage of a random oracle, which is expected to cost $\min\{2^{r_a}, 2^{r_s}\}$ queries. Having a simulator which defeats the second attack seems hard. Indeed, this attack relies on the event **CONNECT**. Any good simulator should set this event with probability $\approx \frac{q^2}{2^b}$ (otherwise one can distinguish easily), yet it is not clear on how to deal with this event without breaking the consistency of the answers. Finally, the second attack resembles the classical differentiability attack from Section 3.4.

Acknowledgements

The author would like to thank the anonymous reviewers for their valuable comments, and Bart Mennink for his insightful comments as well as the fruitful discussions on this work. The author is supported by the Netherlands Organisation for Scientific Research (NWO) under grant OCENW.KLEIN.435.

References

BDPA11. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "The Keccak reference", SHA-3 competition (round 3), 2011. https://keccak.team/files/Keccak-reference-3.0.pdf.

BDPV07. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sponge functions. Ecrypt Hash Workshop 2007, May 2007.

BDPV08. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the Indifferentiability of the Sponge Construction. In Nigel P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, volume 4965 of *Lecture Notes in Computer Science*, pages 181–197. Springer, 2008.

BR93. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993*, pages 62–73. ACM, 1993.

BR96. Mihir Bellare and Phillip Rogaway. The exact security of digital signatures - how to sign with RSA and rabin. In Ueli M. Maurer, editor, *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416. Springer, 1996.

BR06. Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, 2006.

CDMP05. Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-Damgård Revisited: How to Construct a Hash Function. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*, pages 430–448. Springer, 2005.

CLL19. Wonseok Choi, ByeongHak Lee, and Jooyoung Lee. Indifferentiability of truncated random permutations. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part I*, volume 11921 of *Lecture Notes in Computer Science*, pages 175–195. Springer, 2019.

Dam89. Ivan Damgård. A design principle for hash functions. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427. Springer, 1989.

DRS09. Yevgeniy Dodis, Thomas Ristenpart, and Thomas Shrimpton. Salvaging merkle-damgård for practical applications. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *Lecture Notes in Computer Science*, pages 371–388. Springer, 2009.

FS86. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

GM22. Lorenzo Grassi and Bart Mennink. Security of truncated permutation without initial value. *IACR Cryptol. ePrint Arch.*, page 508, 2022. to appear in ASIACRYPT2022.

GPP11. Jian Guo, Thomas Peyrin, and Axel Poschmann. The PHOTON family of lightweight hash functions. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 222–239. Springer, 2011.

Mer89. Ralph C. Merkle. One way hash functions and DES. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446. Springer, 1989.

MPS12. Avradip Mandal, Jacques Patarin, and Yannick Seurin. On the public indifferentiability and correlation intractability of the 6-round feistel construction. In Ronald Cramer, editor, *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, volume 7194 of *Lecture Notes in Computer Science*, pages 285–302. Springer, 2012.

MRH04. Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In Moni Naor, editor, *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39. Springer, 2004.

NIS12. NIST. SHA-3 Competition, 2007-2012. https://csrc.nist.gov/projects/hash-functions/sha-3-project.

NO14. Yusuke Naito and Kazuo Ohta. Improved indifferentiable security analysis of PHOTON. In Michel Abdalla and Roberto De Prisco, editors, *Security and Cryptography for Networks - 9th International Conference, SCN 2014, Amalfi, Italy, September 3-5, 2014. Proceedings*, volume 8642 of *Lecture Notes in Computer Science*, pages 340–357. Springer, 2014.

RSS11. Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. Careful with composition: Limitations of the indifferentiability framework. In Kenneth G. Paterson, editor, *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, volume 6632 of *Lecture Notes in Computer Science*, pages 487–506. Springer, 2011.

YMO09. Kazuki Yoneyama, Satoshi Miyagawa, and Kazuo Ohta. Leaky random oracle. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 92-A(8):1795–1807, 2009.