On the security of the keyed sponge construction

Guido Bertoni[1], Joan Daemen[1], Michaël Peeters[2], and Gilles Van Assche[1]

[1] STMicroelectronics
[2] NXP Semiconductors

Abstract. Previously we have proven that the advantage in differentiating the sponge construction from a random oracle is upper bounded by $N^2 2^{-(c+1)}$ with N the number of calls to the underlying transformation or permutation and c the capacity. In this paper we prove that the advantage in distinguishing a keyed sponge from a random oracle is upper bounded by $\max((M^2/2 + 2MN)2^{-c}, N2^{-|K|})$, with M the data complexity, N the time complexity and $|K|$ the length of the key. This bound is smaller than the indifferentiability bound and allows decreasing the capacity for a given required security level or achieves a higher security level for a given capacity. This new bound has positive implications for all applications in which a sponge function is used in conjunction with a key.

Keywords: sponge construction, generic attacks, keyed sponge, key recovery, pre-image resistance

1 Introduction

Sponge functions are versatile cryptographic primitives that can be used for hashing, message authentication code (MAC) computation, stream encryption, single-pass authenticated encryption, pseudo-random sequence generation and other applications. We refer to [6,7,5] for description of such modes of use.

A sponge function consists of the application of the sponge construction to a fixed-length permutation or transformation f. The sponge construction is sound in the sense that attacks that do no exploit specific properties of f, i.e., generic attacks, are very unlikely to have success. More exactly, in [3] we have proven that the advantage in differentiating the sponge construction calling a random permutation or transformation f from a random oracle is upper bounded by $N^2 2^{-(c+1)}$ with N the number of calls to the underlying transformation or permutation and c the capacity. This implies that with respect to generic attacks of complexity N, the sponge construction offers the same level of security as a random oracle, as long as $N^2 2^{-(c+1)}$ is negligible. Once N approaches $2^{c/2}$ this is no longer the case and the indifferentiability bound does not give any guarantees. As pointed out in [2,3], this is due to the existence of *inner collisions* in a sponge function.

In [2] we already noticed that the expected workload of determining the inner state of a sponge function is probably much higher than $2^{c/2}$. However, we were not able prove a lower bound for this workload. In modes of use where a sponge function is used in conjunction with a key, the difficulty of state recovery is crucial and the amount of sponge function processing with a given key is usually too small for inner collisions to actually occur. Hence, while the indifferentiability bound suggests that the capacity must be twice the size of the key, our investigations in [2] suggest that a capacity slightly larger than the key may suffice. In other words, the security level of a sponge function in a keyed mode is probably much higher than $2^{c/2}$. Note that this includes the traditional criterion of pre-image resistance, where the pre-image unknown to the adversary can be considered as a key.

In [7] we have proven upper bounds to the success probability of state recovery. In this paper we generalize this to an upper bound for the advantage of distinguishing a keyed sponge from a random oracle. Indistinguishability is a very strong notion of security as it implies the inability of generating collisions, retrieving the state or guessing the key. Hence, this upper bound allows keyed sponges to

claim a higher security level for a given capacity or to take a smaller capacity for a required security level in any application. Note in a sponge function the sum of bitrate and capacity is equal to the width of the underlying permutation (or transformation) and hence decreasing capacity results in an immediate increase of bitrate and hence efficiency.

In this paper we concentrate on the case that f is a permutation because of its practical relevance. The same bound can be proven for the case that f is a transformation and we leave it as an exercise for the reader.

The remainder of this paper is organized as follows. First we remind the reader of the sponge construction and define the keyed sponge in Section 2. This is followed by a quick introduction of indistinguishability in Section 3. In Section 4 we prove in a number of steps the new bound. Finally, in Section 5 we discuss the implications of this bound.

2 The sponge construction

The sponge construction [2] builds a function $\text{SPONGE}[f, \text{pad}, r]$ with variable-length input and arbitrary output length using a fixed-length permutation (or transformation) f, a padding function "pad" and a parameter *bitrate* r. A sponge function, that is, a function implementing the sponge construction, provides a particular way to generalize hash functions and has the same interface as a random oracle.

2.1 Definition

The permutation f operates on a fixed number of bits, the *width* b. The sponge construction has a state of b bits. First, all the bits of the state are initialized to zero. The input message is padded with pad up to a multiple r and cut into blocks of r bits. For the padding function we use the following notation: the padding of a message M to a sequence of x-bit blocks is denoted by $M||\text{pad}[x](|M|)$.

After padding it proceeds in two phases: the *absorbing phase* followed by the *squeezing phase*:

Absorbing phase The r-bit input message blocks are XORed into the first r bits of the state, interleaved with applications of the function f. When all message blocks are processed, the sponge construction switches to the squeezing phase.

Squeezing phase The first r bits of the state are returned as output blocks, interleaved with applications of the function f. The number of iterations is determined by the requested number of bits.

Finally the output is truncated to the requested length. The sponge construction is illustrated in Figure 1 and Algorithm 1 provides a formal definition.

The value $c = b - r$ is called the *capacity*. The first r bits of a state s are called its *outer part* $\widetilde{s}$ and the last c bits its *inner part* $\widehat{s}$. The inner par of the state is never directly affected by the input blocks and never output during the squeezing phase. The capacity c actually determines the attainable security level of the construction [3].

2.2 Graph representation of sponge operations

In [2] we used a graph representation to prove bounds on success probability of generating collisions. We adopt this graph representation for describing the knowledge of the adversary.

We consider the transformation f as a directed graph whose vertex set (called *nodes*) is $\mathbb{Z}_2^b = \mathbb{Z}_2^r \times \mathbb{Z}_2^c$ and whose edges are $(s, f(s))$. It has both 2^b nodes and edges. From the node graph, we derive the (directed) supernode graph, with vertex set (called *supernodes*) equal to $\mathbb{Z}_2^c$. In this

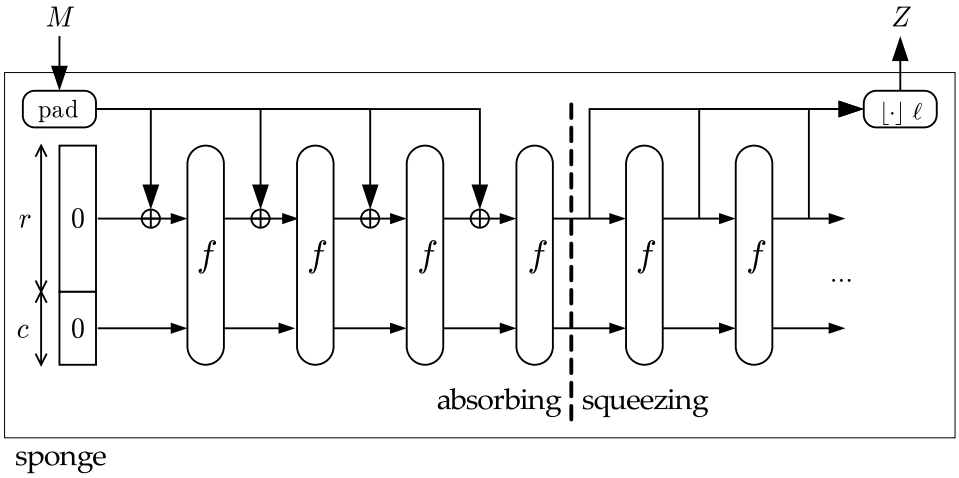

Fig. 1. The sponge construction

Algorithm 1 The sponge construction SPONGE$[f, \text{pad}, r]$

Require: $r < b$

Interface: $Z = \text{sponge}(M, \ell)$ with $M \in \mathbb{Z}_2^*$, integer $\ell > 0$ and $Z \in \mathbb{Z}_2^\ell$
$P = M||\text{pad}[r](|M|)$
Let $P = P_0||P_1||\dots||P_w$ with $|P_i| = r$
$s = 0^b$
for $i = 0$ to w **do**
  $s = s \oplus (P_i||0^{b-r})$
  $s = f(s)$
end for
$Z = \lfloor s \rfloor_r$
while $|Z| < \ell$ **do**
  $s = f(s)$
  $Z = Z||\lfloor s \rfloor_r$
end while
return $\lfloor Z \rfloor_\ell$

graph, an edge $(\widehat{s}, \widehat{t})$ is in the edge set if and only if $\exists \widetilde{s}, \widetilde{t}$ such that (s, t) with $s = (\widetilde{s}, \widehat{s})$ and $t = (\widetilde{t}, \widehat{t})$ is an edge in the node graph. The set of supernodes is a partition of the nodes where a supernode contains the 2^r nodes with the same inner part.

The sponge construction operates on a chaining variable s and its operation can be seen as a walk through the node graph of the chaining variable. We denote the chaining variable before absorbing an input block p_i by s_i. Its initial value is $s_0 = 0^b = (0^r, 0^c)$. Then for each block p_i, it performs a two-step transition. First, it moves to the node s' within the same supernode with $\widetilde{s'} = \widetilde{s_i} \oplus p_i$, and then it follows the edge starting from s', arriving in s_{i+1}. After absorbing all blocks of p it arrives in node $s_{|p|}$ with $|p|$ denoting the number of blocks in p. Then it gives the outer part of $s_{|p|}$ as output z_0. For each additional block z_i squeezed it follows the edge from $s_{|p|+i-1}$ arriving in $s_{|p|+i}$ and gives out the outer part of the latter as z_i. Note that this can be considered a special case of the above two-step transition if we extend p with blocks $p_{|p|+i} = 0^r$ for all $i \geq 0$.

Clearly, the chaining variable s_i is completely determined by the first i blocks of p. We call this a *path* to s_i. Or more exactly:

Definition 1 ([2]). *First, the empty string is a path to the node $0^b = (0^r, 0^c)$. Then, if p is a path to node s and there is an edge $((\widetilde{s} \oplus a, \widehat{s}), t)$ in the node graph, $p' = p||a$ is a path to node t.*

Note that although a path completely determines a node, there may be many paths to a node. A pair of different paths to the same supernode is called an *inner collision*.

It follows from the above that the j-th output block z_j of $z = \text{SPONGE}[f, \text{pad}, r](p)$ is the outer part of the node with path $p||0^{rj}$. And so, given a path p (different from 0^{rj}) to a node s, one can find its outer part by a call to the sponge construction. We have $\widetilde{s} = z_j$ with $z = \text{SPONGE}[f, \text{pad}, r](p')$ and p' and j given by $p = p'||0^{rj}$ such that p' is a valid sponge input, i.e., $|p'| > 0$ and $p'_{|p'|-1} \neq 0^r$. For a path of form 0^{rj} there is no such p' and hence the sponge construction cannot be queried to obtain $\widetilde{s}$.

2.3 The keyed sponge

We define a keyed sponge as a sponge function in which every input presented to the sponge is first pre-pended with a secret key K:

$$\text{KEYEDSPONGE}[f, \text{pad}, r, K](M, \ell) = \text{SPONGE}[f, \text{pad}, r](K||M, \ell).$$

We write $Z = \text{KEYEDSPONGE}(M, \ell)$ and assume the parameters are clear from the context.

3 Indistinguishability from a random oracle

Indistinguishability deals with the ability of telling apart two systems, typically a concrete construction and an ideal system. An adversary can query the two systems, one at the left and one at the right, and a priori does not know which system is which. Based on the responses to these queries it shall then decide whether the concrete construction is at the left or at the right. If this is hard, the concrete construction can replace the ideal system in application without any significant loss of security [9].

In this paper we prove upper bounds for the success probability of distinguishing a keyed sponge from a random oracle. We use the definition of random oracle from [1]. A random oracle, denoted $\mathcal{RO}$, takes as input binary strings of any length and returns for each input a random infinite string, i.e., it is a map from $\mathbb{Z}_2^*$ to $\mathbb{Z}_2^\infty$, chosen by selecting each bit of $\mathcal{RO}(x)$ uniformly and independently, for every x.

The success probability depends on the guessing rule used by the adversary. Let R_{KS} be the set of observations for which she guesses that the system at the left is the keyed sponge. The probability of success reads

$$\Pr(\text{success}) = \frac{1}{2} + \frac{1}{2} \sum_{r \in R_{\mathrm{KS}}} \Pr(r \text{ observed}|\textsc{KeyedSponge}) - \Pr(r \text{ observed}|\mathcal{RO}). \tag{1}$$

To maximize the success probability, the adversary has to choose

$$R_{\mathrm{KS}} = \{r \ : \ \Pr(r \text{ observed}|\textsc{KeyedSponge}) \geq \Pr(r \text{ observed}|\mathcal{RO})\}.$$

The quantity $\sum_{r \in R_{\mathrm{KS}}} \Pr(r \text{ observed}|\textsc{KeyedSponge}) - \Pr(r \text{ observed}|\mathcal{RO})$ is called the *distinguishing advantage*.

4 Upper bounds for distinguishing a keyed sponge from a random oracle

In this section we prove an upper bound for the success probability of distinguishing a keyed sponge from a random oracle below the indifferentiability bound. We start by defining the setting, the way we model the knowledge of the adversary and the cost of queries. Then we prove that in the absence of inner collisions and inner clashes (see Section 4.4), the advantage of telling apart the output of a sponge function and that of a random oracle is zero. Hence the event r observed in Section 3 is an inner collision or an inner clash. We then provide upper bounds for the marginal success probabilities of individual queries in the absence of inner collisions or inner clashes. Finally, we combine these to prove an upper bound on the advantage of an attack of given data and time complexity.

4.1 The setting

In the context of our proof the adversary shall distinguish between a keyed sponge and a random oracle using their responses to queries. We assume that the adversary has no a priori knowledge about f and that she can learn about f by making queries to it. Hence, the adversary shall distinguish between two systems that each have two components, as illustrated in Figure 2:

- The system at the left is the combination of the function f and the keyed sponge function $x = \textsc{KeyedSponge}(M, \ell)$. The adversary can make queries to both components separately, where the latter in turn calls the former to construct its responses. These are the two different interfaces to the system at the left.
- The system at the right consists of a random oracle $\mathcal{RO}$ and the function f, working independently.

Clearly $\mathcal{RO}$ is independent of f, so the goal of this setting is to show that it is hard to see the dependence of a keyed sponge on its underlying permutation f for an adversary who does not know the key.

We give a proof of indistinguishability for the case that f is a random permutation. A random permutation operating on a certain domain is a permutation selected randomly and uniformly from all permutations operating on that domain.

The keyed sponge function $\textsc{KeyedSponge}(x, \ell)$ provides one interface denoted by $\mathcal{H}$, taking a binary string $x \in \mathbb{Z}_2^*$ and an integer ℓ and returning a binary string $y \in \mathbb{Z}_2^\ell$, the sponge output truncated to ℓ bits. For its execution it can make calls to f and has access to a key K. The permutation f has an interface f that takes as input an element s of $\mathbb{Z}_2^b$ and returns $t = f(s)$ and

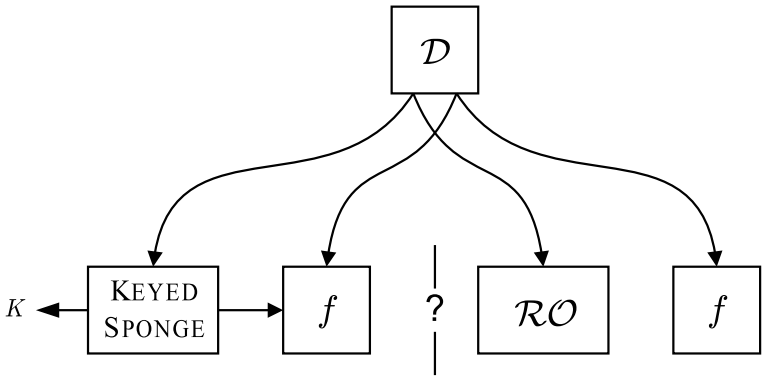

Fig. 2. The distinguishing setup

an interface f^{-1} that takes as input an element s of $\mathbb{Z}_2^b$ and returns $t = f^{-1}(s)$. The union of the two interfaces f and f^{-1} is denoted by $f^{\pm 1}$. Note that the sponge construction in Algorithm 1 only uses the interface f.

At the right is the system with $\mathcal{RO}$ and f. It offers the same interface as the left system, i.e., $\mathcal{RO}$ provides the interface $\mathcal{H}$ and returns an output truncated to the requested length.

4.2 The knowledge of the adversary

The adversary can keep track of what she learns from the responses received to the queries she sent. We represent this knowledge in two graphs: one representing her knowledge on the keyed sponge or random oracle: the so-called *$\mathcal{H}$-knowledge graph*, and the other representing her knowledge on f: the *f-knowledge graph*.

The f-knowledge graph is a subgraph of the one discussed in Section 2.2. It has a number of edges corresponding to the past queries to $f^{\pm 1}$ and for each edge the adversary knows the value of the node where it starts and the node where it arrives. We denote the set of nodes (and the corresponding set of values) with an outgoing edge by $\mathcal{F}_o$ and the set of nodes with an incoming edge by $\mathcal{F}_i$. Additionally, we denote the set of supernodes (and the corresponding set of inner values) with an incoming edge by $\widehat{\mathcal{F}_i}$. We say a node s is in $\widehat{\mathcal{F}_i}$ if the supernode $\widehat{s}$ is in $\widehat{\mathcal{F}_i}$. Figure 3 gives an example of an f-knowledge graph. In this example every supernode has 4 nodes. Nodes in black are in $\mathcal{F}_o$, nodes in gray are in $\mathcal{F}_i$ and nodes indicated by gray squares are in both. Supernodes in $\widehat{\mathcal{F}_i}$ are indicated with a black oval, those not in $\widehat{\mathcal{F}_i}$ with a gray oval.

The $\mathcal{H}$-knowledge graph can be seen as a *blinded* subgraph of the one discussed in Section 2.2. Here blinded means that the adversary does not know the inner part of its (super-)nodes except the root and that it may contain edges starting from and arriving in nodes with unknown outer parts. More particularly, the adversary only knows the outer parts of the nodes that are traversed by the sponge function in the squeezing phase. The lack of knowledge on inner parts prohibits the adversary to detect inner collisions directly and hence to determine the topology of the graph correctly. Moreover, one may wonder whether such a blinded graph is sufficient for representing the adversary's knowledge in all phases of a distinguishing exercise. We address these problems in the following way. If $\mathcal{H}$ is a keyed sponge, we consider the adversary to be successful if one of the following two events occurs in the $\mathcal{H}$-knowledge graph:

– Detection of an inner collision, as a random oracle has no inner collision;

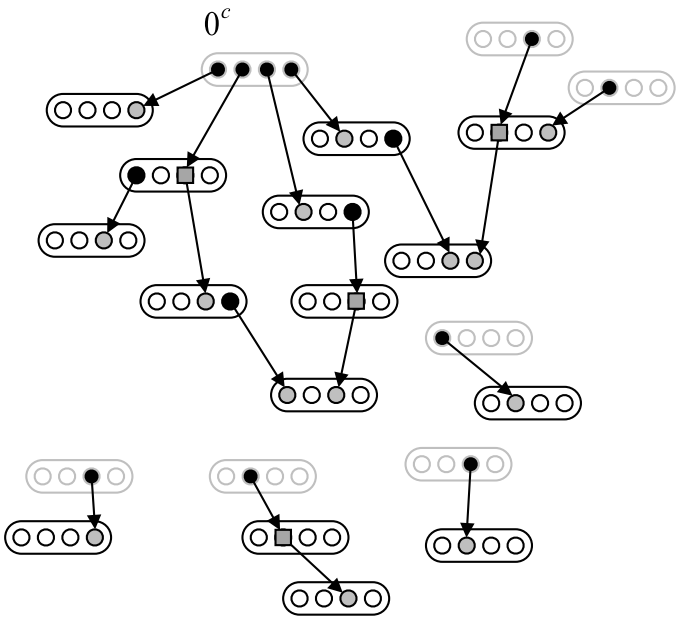

Fig. 3. Example of f-knowledge graph

– Knowledge of the inner part of a node with an incoming edge and known outer part, as this allows the adversary to distinguish it from a random oracle quickly with some queries to f.

So during the distinguishing experiment the $\mathcal{H}$-knowledge graph does not exhibit any inner collisions. It follows that its supernodes form a tree and the topology is therefore uniquely determined. Moreover, at all times the inner parts of the nodes with incoming edges and known outer parts are unknown.

We denote the set of nodes in the $\mathcal{H}$-knowledge graph with an incoming edge and known outer part by $\mathcal{R}_i$ and the set of nodes with an outgoing edge by $\mathcal{R}_o$. Additionally, we denote the set of supernodes containing a node in $\mathcal{R}_i$ by $\widehat{\mathcal{R}_i}$. We say a node s is in $\widehat{\mathcal{R}_i}$ if the supernode $\widehat{s}$ is in $\widehat{\mathcal{R}_i}$. Figure 4 gives an example of a $\mathcal{H}$-knowledge graph. Nodes in black are in $\mathcal{R}_o$, nodes in gray are in $\mathcal{R}_i$ and nodes that are in both are indicated by a gray square. Supernodes in $\widehat{\mathcal{R}_i}$ are indicated with a black oval, those not in $\widehat{\mathcal{R}_i}$ with a gray oval.

4.3 The cost of queries

The bound we provide in this paper is on the success probability of a distinguishing attack as a function of the cost function that models the effort from the adversary. We define here a cost function similar to the one used in our indifferentiability bounds [3]. However, unlike in the indifferentiability bounds, where the cost could be expressed in a single variable, here the cost consists of two variables. These two variables model different aspects of an attack:

Data complexity This models queries sent to the keyed instance. In a setup attacking a concrete cryptosystem, this models the amount of access to the keyed instance. In practice one can limit the adversary's access to the instance. This is sometimes also referred to as the online complexity.

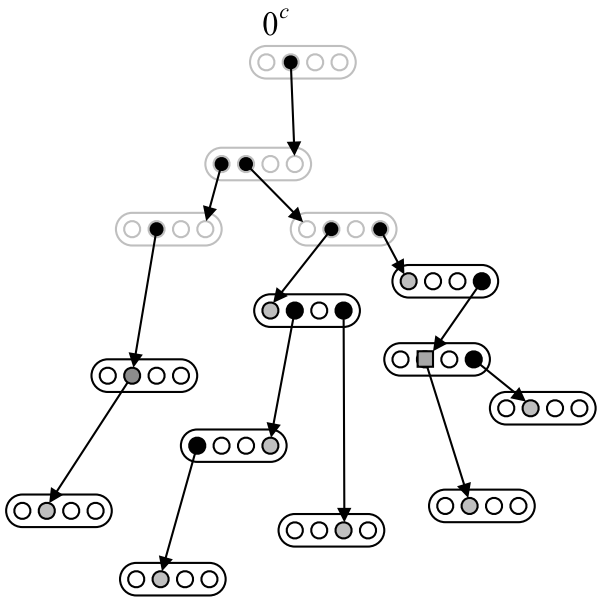

Fig. 4. Example of $\mathcal{H}$-knowledge graph

Time complexity This models computations requiring no access to the keyed instance. In a concrete setup the only limitation of the time complexity is the computing power and time available to the adversary. This is sometimes also referred to as the offline complexity.

The unit of our cost function is the number of calls to $f^{\pm 1}$:

- The time complexity N is the number of queries to $f^{\pm 1}$, where only fresh queries are counted. A query is not fresh if its response is already known due to previous queries to $f^{\pm 1}$.
- The data complexity M is the number of fresh calls to f due to queries to $\mathcal{H}$ as if $\mathcal{H} =$ KeyedSponge. Here a call to $f(s)$ is not fresh if it has already been made due to a prior query to $\mathcal{H}$.

Clearly, the number of edges in the f-knowledge graph is equal to N. Per definition, after a number of queries to $\mathcal{H}$ with a total cost of M, the $\mathcal{H}$-knowledge graph has M edges.

The convention of only counting fresh queries/calls only benefits to the adversary and is thus on the safe side regarding security.

4.4 Conditions for uniform and independent output of the keyed sponge

We introduce now the concept of an event that can occur in the distinguishing experiment, an *inner clash*.

Definition 2. *An inner clash occurs if $\mathcal{H}$ is the keyed sponge and if due to a query the intersection of $\widehat{\mathcal{F}_i}$ and $\widehat{\mathcal{R}_i}$ becomes non-empty.*

We now prove the following lemma.

Lemma 1. *In the absence of inner collisions in the $\mathcal{H}$-graph and of inner clashes, the response to a query to $\mathcal{H}$ is uniformly and independently distributed.*

Proof. If $\mathcal{H}$ is the random oracle the lemma is true. Otherwise, the query results in a number of new edges in the $\mathcal{H}$-knowledge graph and its response is the concatenation of outer parts of nodes traversed in the squeezing phase. The following reasoning applies for each of the edges that arrives in such a node. Let s be the node where such an edge starts and $t = f(s)$ where it arrives. We compute the probability that the outer part of t has a particular value a, on the condition that the new node t does not introduce an inner collision nor an inner clash. This probability must be taken over the set F of functions f that are compatible with the knowledge on f from all prior queries to $f^{\pm 1}$ and the sponge function. We will prove that this probability is uniform: equal to 2^{-r} independently of the value a.

Due to the requirements related to inner collisions and inner clashes, the set of possible values for t is $\mathbb{Z}_2^b \setminus (\mathbb{Z}_2^r \times (\widehat{\mathcal{R}_i} \cup \widehat{\mathcal{F}_i})) = \mathbb{Z}_2^r \times (\mathbb{Z}_2^c \setminus (\widehat{\mathcal{R}_i} \cup \widehat{\mathcal{F}_i}))$. Clearly, this set contains an equal number of elements for each given outer part and hence the outer part of t has a uniform distribution. □

Without the requirement that t shall not result in an inner collision or inner clash, its possible values are given by the set $\mathbb{Z}_2^b \setminus (\mathcal{R}_i \cup \mathcal{F}_i)$. This follows immediately from the fact that s has not been queried before and that f is a permutation. Note that the outer part of t is in general not distributed uniformly. Hence, it is the requirement on the absence of inner collisions and inner clashes that gives the sponge its uniformity in this setting.

From Lemma 1 it follows that the set of observations R_{KS} in Equation (1) coincide with the occurrence of an inner collision or an inner clash. As a random oracle has no finite inner state, these cannot occur for a random oracle and we have:

$$\Pr(\text{success}) = \frac{1}{2} + \frac{1}{2}\Pr(\text{inner collision or inner clash}|\textsc{KeyedSponge}).$$

4.5 Success probabilities of queries

From Lemma 1 it follows immediately that an adversary can only have non-zero advantage in distinguishing a keyed sponge function from a random oracle if she succeeds in generating an inner collision in the sponge function or an inner clash between queries to $f^{\pm 1}$ and queries to $\mathcal{H}$. As a matter of fact, we consider the adversary to have succeeded as soon as an inner clash occurs or an inner collision occurs in the $\mathcal{H}$ graph, and we ignore the cost of detecting it. This generosity towards the adversary puts us on the safe side.

Assume the adversary has performed i fresh queries to $f^{\pm 1}$ and a number of queries to $\mathcal{H}$ with a total cost of j. We now provide bounds on the probability of success of the next query as a function of i and j, for the three types of query, assuming that no inner collision or inner clash has occurred yet (indicated by NIC).

Queries to $\mathcal{H}$ In the $\mathcal{H}$-knowledge graph, a query results in the addition of a number of edges and nodes. Rather than computing an upper bound for the success probability for a full query, we compute it for the success probability for each edge added in the $\mathcal{H}$-knowledge graph, i.e., per increment of the cost by 1. More specifically, we compute an upper bound on the probability that an inner collision or an inner clash occurs when adding an edge to the $\mathcal{H}$-knowledge graph.

Consider the addition of an edge from s to a new node t. The probability of an inner collision is the number of nodes in the $\mathcal{H}$-knowledge graph that have no incoming edge divided by the total number of nodes minus those that already have an incoming edge in the $\mathcal{H}$-knowledge or f-knowledge graph:

$$\Pr(\text{inner collision}|\text{NIC}) \leq \frac{(2^r - 1)j + 2^r}{2^{r+c} - (i + j)}. \tag{2}$$

For the probability of success of an inner clash we have the following lemma.

Lemma 2. *The probability of an inner clash for an edge added to the $\mathcal{H}$-knowledge graph is upper bounded by:*

$$\Pr(\textit{inner clash}|\text{NIC}) \leq \frac{2i}{2^c - (i+j)}.$$

Proof. The probability of an inner clash is the probability that the new node t is in $\widehat{\mathcal{F}_i}$. Here we consider two cases: either the node s from which the edge starts is in $\mathcal{F}_o$ or it isn't. So the probability of an inner clash is:

$$\begin{aligned}\Pr(t \in \widehat{\mathcal{F}_i}|\text{NIC}) &= \Pr(t \in \widehat{\mathcal{F}_i}|s \in \mathcal{F}_o)\Pr(s \in \mathcal{F}_o|\text{NIC}) + \Pr(t \in \widehat{\mathcal{F}_i}|s \notin \mathcal{F}_o)\left(1 - \Pr(s \in \mathcal{F}_o|\text{NIC})\right) \\ &\leq \Pr(t \in \widehat{\mathcal{F}_i}|s \in \mathcal{F}_o)\Pr(s \in \mathcal{F}_o|\text{NIC}) + \Pr(t \in \widehat{\mathcal{F}_i}|s \notin \mathcal{F}_o).\end{aligned}$$

Clearly $\Pr(t \in \widehat{\mathcal{F}_i}|s \in \mathcal{F}_o) = 1$. For computing $\Pr(s \in \mathcal{F}_o|\text{NIC})$ we consider the case that the adversary can choose the outer part of s. She can optimize her success probability taking for $\widetilde{s}$ the value that occurs most often in nodes in $\mathcal{F}_o$. If we denote the number of nodes in $\mathcal{F}_o$ with outer part equal to $\widetilde{s}$ by $m_f(\widetilde{s})$ and the number of nodes in $\mathcal{R}_o$ with outer part equal to $\widetilde{s}$ by $m_H(\widetilde{s})$, this results in

$$\Pr(s \in \mathcal{F}_o|\text{NIC}) = \frac{m_f(\widetilde{s})}{2^c - m_H(\widetilde{s})},$$

this is the number of nodes in $\mathcal{F}_o$ with outer part equal to $\widetilde{s}$ divided by the total number of nodes with outer part equal to $\widetilde{s}$ that are not in $\mathcal{R}_o$ right before the query. The nodes with outer part $\widetilde{s}$ in $\mathcal{R}_o$ are excluded because they already have an outgoing edge. Clearly, this expression is correct even if the adversary cannot choose or even does not know $\widetilde{s}$.

Finally we have:

$$\Pr(t \in \widehat{\mathcal{F}_i}|s \notin \mathcal{F}_o) \leq \frac{(2^r - 1)i}{2^{r+c} - i},$$

the maximum possible number of nodes in $\widehat{\mathcal{F}_i}$ where an edge starting from s can arrive divided by the total number of nodes minus those in $\mathcal{F}_o$. This results in the following upper bound for the probability of an inner clash:

$$\begin{aligned}\Pr(t \in \widehat{\mathcal{F}_i}|\text{NIC}) &\leq \frac{m_f(\widetilde{s})}{2^c - m_H(\widetilde{s})} + \frac{(2^r - 1)i}{2^{r+c} - i} \\ &\leq \frac{m_f(\widetilde{s})}{2^c - m_H(\widetilde{s})} + \frac{i}{2^c - i} \\ &\leq \frac{m_f(\widetilde{s})}{2^c - (i + m_H(\widetilde{s}))} + \frac{i}{2^c - (i + m_H(\widetilde{s}))}.\end{aligned}$$

Using the fact that $m_f(\widetilde{s}) \leq i$ and $m_H(\widetilde{s}) \leq j$ this simplifies to:

$$\Pr(\text{inner clash}|\text{NIC}) \leq \frac{2i}{2^c - (i+j)}.$$

□

Queries to $f^{\pm 1}$ A query to $f^{\pm 1}$ cannot result in an inner collision. In the following lemma we provide an upper bound to the probability of an inner clash of queries to f and f^{-1}.

Lemma 3. *The probability of an inner clash due to a query to $f^{\pm 1}$ is upper bounded by:*

$$\Pr(\textit{inner clash}|\text{NIC}) \leq \frac{2j}{2^c - (i+j)}.$$

Proof. The probability of an inner clash of a query $t = f(s)$ is the probability that t is in $\widehat{\mathcal{R}_i}$. Here we consider again two cases: either s is in $\mathcal{R}_o$ or it isn't. So the probability of an inner clash is:

$$\begin{aligned}\Pr(t \in \widehat{\mathcal{R}_i}|\text{NIC}) &= \Pr(t \in \widehat{\mathcal{R}_i}|s \in \mathcal{R}_o)\Pr(s \in \mathcal{R}_o|\text{NIC}) + \Pr(t \in \widehat{\mathcal{R}_i}|s \notin \mathcal{R}_o)\left(1 - \Pr(s \in \mathcal{R}_o|\text{NIC})\right) \\ &\leq \Pr(s \in \mathcal{R}_o|\text{NIC}) + \Pr(t \in \widehat{\mathcal{R}_i}|s \notin \mathcal{R}_o).\end{aligned}$$

Similarly to the case of queries to $\mathcal{H}$, we have

$$\Pr(s \in \mathcal{R}_o|\text{NIC}) = \frac{m_H(\widetilde{s})}{2^c - m_f(\widetilde{s})},$$

and

$$\Pr(t \in \widehat{\mathcal{R}_i}|s \notin \mathcal{R}_o) \leq \frac{(2^r - 1)j}{2^{r+c} - j}.$$

This results in the following upper bound for the probability of an inner clash:

$$\begin{aligned}\Pr(t \in \widehat{\mathcal{F}_i}|\text{NIC}) &\leq \frac{m_H(\widetilde{s})}{2^c - m_f(\widetilde{s})} + \frac{(2^r - 1)j}{2^{r+c} - 2^r j} \\ &\leq \frac{m_H(\widetilde{s})}{2^c - m_f(\widetilde{s})} + \frac{i}{2^c - j} \\ &\leq \frac{m_H(\widetilde{s})}{2^c - (j + m_f(\widetilde{s}))} + \frac{i}{2^c - (j + m_f(\widetilde{s}))}.\end{aligned}$$

Using the fact that $m_f(\widetilde{s}) \leq i$ and $m_H(\widetilde{s}) \leq j$ this simplifies to:

$$\Pr(\text{inner clash}|\text{NIC}) \leq \frac{2j}{2^c - (i + j)}.$$

We now deal with queries to f^{-1}. Let us denote the number of nodes in $\mathcal{F}_i$ with outer part equal to $\widetilde{t}$ by $m_{if}(\widetilde{t})$ and the number of nodes in $\mathcal{R}_i$ with outer part equal to $\widetilde{t}$ by $m_{iH}(\widetilde{t})$. The probability that a query $s = f^{-1}(t)$ results in an inner clash is fully determined by t and given by

$$\Pr(t \in \widehat{\mathcal{R}_i}|\text{NIC}) = \frac{m_{iH}(\widetilde{t})}{2^c - m_{if}(\widetilde{t})},$$

the number of nodes in $\mathcal{R}_i$ with outer part equal to $\widetilde{t}$ divided by the total number of nodes with outer part equal to $\widetilde{t}$ not in $\mathcal{F}_i$ before the query. As $m_{iH}(\widetilde{t}) \leq j$ and $m_{if}(\widetilde{t}) \leq i$, we have:

$$\Pr(\text{inner clash}|\text{NIC}) \leq \frac{j}{2^c - i} \leq \frac{2j}{2^c - (i + j)}.$$

□

4.6 Total success probability

We can now use Equation (2) and Lemmas 2, 3 to obtain an upper bound on the success probability of distinguishing for given values of N and M.

The probability of an inner collision is given by the following lemma.

Lemma 4. *The probability of an inner collision after a sequence of queries to $\mathcal{H}$ with cost M and queries to $f^{\pm 1}$ with cost N is upper bounded by:*

$$\Pr(\textit{inner collision}) \leq 1 - \exp\left(\frac{-M^2}{2^{c+1}}\right)$$

Proof. Let i_j be the number of $f^{\pm 1}$ queries performed before query j to $\mathcal{H}$. The probability that an inner collision occurs in the $\mathcal{H}$-knowledge graph is given by:

$$\begin{aligned}\Pr(\text{inner collision}) &= 1 - \prod_{j=0}^{M-1} \left(1 - \frac{(2^r-1)j + 2^r}{2^{r+c} - (i_j + j)}\right) \\ &= 1 - \prod_{j=0}^{M-1} \left(1 - \frac{\frac{j+1}{2^c} - \frac{j}{2^{r+c}}}{1 - \frac{i_j + j}{2^{r+c}}}\right)\end{aligned}$$

Using the approximation $\log(1-\epsilon) \approx -\epsilon$ for $\epsilon \ll 1$ and neglecting the last term in the denominator by exploiting $i_j + j \ll 2^{r+c}$ yields

$$\Pr(\text{inner collision}) = 1 - \exp\left(-\sum_{j=0}^{M-1} \frac{j+1}{2^c} - \frac{j}{2^{r+c}}\right) = 1 - \exp\left(-\frac{M(M+1)}{2^{c+1}} + \frac{M(M-1)}{2^{r+c+1}}.\right)$$

This is the same expression as the success probability for generating inner collisions in a random P-sponge obtained in [2] and the advantage of differentiating a random P-sponge from a random oracle in [3]. For $M \gg 1$ and $2^r \gg 1$, this simplifies to

$$\Pr(\text{inner collision}) \leq 1 - \exp\left(\frac{-M^2}{2^{c+1}}\right).$$

□

An upper bound to the probability of an inner clash is given by the following lemma.

Lemma 5. *The probability of an inner clash in a sequence of queries to $\mathcal{H}$ with cost M and queries to $f^{\pm 1}$ with cost N for an adversary with no knowledge on the inner parts of the nodes in the $\mathcal{H}$-knowledge graph is upper bounded by:*

$$\Pr(\textit{inner clash}) \leq 1 - \exp\left(\frac{-2MN}{2^c}\right).$$

Proof. For generating inner clashes the success probability of the k-th query depends on whether it is a query to $\mathcal{H}$ or to $f^{\pm 1}$. Let $\delta_k = 1$ if the k-th query is one to $f^{\pm 1}$ and $\delta_k = 0$ otherwise. Let now i_k be the number of f-queries performed before the k-th query and j_k the number of $\mathcal{H}$-queries performed before the k-th query. We have $i_k = \sum_{\ell<k} \delta_\ell$ and $j_k = \sum_{\ell<k}(1-\delta_\ell)$. We can combine Lemma 2 and Lemma 3 into:

$$\Pr(\text{inner clash in } k\text{-th query}) \leq \frac{2(1-\delta_k)i_k + 2\delta_k j_k}{2^c - (k-1)}.$$

The total inner clash probability is now upper bounded by:

$$\Pr(\text{inner clash}) \leq 1 - \prod_{k=1}^{M+N} \left(1 - \frac{2(1-\delta_k)i_k + 2\delta_k j_k}{2^c - (k-1)}\right).$$

Using the approximation $\log(1-\epsilon) \approx -\epsilon$ for $\epsilon \ll 1$ and assuming $k \ll 2^c$ yields

$$\Pr(\text{inner clash}) \leq 1 - \exp\left(-\frac{2}{2^c} \sum_{k=1}^{M+N} (1-\delta_k)i_k + \delta_k j_k\right). \tag{3}$$

Working out the sum in the righthand member of equation (3) yields

$$\begin{aligned}\sum_k (1-\delta_k)i_k + \delta_k j_k &= \sum_k \sum_{\ell<k} (1-\delta_k)\delta_\ell + \sum_k \sum_{\ell<k} \delta_k(1-\delta_\ell) \\ &= \sum_k \sum_{\ell<k} (1-\delta_k)\delta_\ell + \sum_k \sum_{k<\ell} \delta_\ell(1-\delta_k) \\ &= \sum_k \sum_{\ell\neq k} (1-\delta_k)\delta_\ell \\ &= \left(\sum_k (1-\delta_k)\right)\left(\sum_\ell \delta_\ell\right) - \sum_k (1-\delta_k)\delta_k \\ &= MN - 0 = MN.\end{aligned}$$

Filling this in in equation (3) yields the result. □

The success probability in Lemma 5 is under the condition that the adversary has no knowledge on the inner parts of nodes in $\mathcal{R}_i$. However, the inner part of a node in $\mathcal{R}_i$ can be found by determining the path to it, i.e., by guessing the key K. When found, it is trivial to generate an inner clash. To verify the correctness of a key guess, the adversary can choose a node in $\mathcal{R}_i$, make key guesses and use f-queries to emulate the keyed sponge. In general in a keyed sponge the key is fully absorbed before the squeezing phase begins and hence any node in $\mathcal{R}_i$ has in its path the full key K. This implies that the full key must be guessed to have success. Every key guess takes at least a single call to f, hence the success probability of key guessing in an attack with data complexity N is at most the probability of guessing a correct key value from a given distribution in the first N attempts. If the length of the key is not fixed, the scheme must ensure input decodability: it must be possible to parse the input to the sponge function to identify the key K and the message M. We refer to [5] for methods for ensuring input decodability. If the key has a fixed length and a uniform distribution, the probability of success is simply $\frac{N}{2^{|K|}}$.

Combining this with Lemma 4 and Lemma 5 results in the main theorem of this paper.

Theorem 1. *The advantage of distinguishing* KeyedSponge$[f, K]$ *from a random oracle, with f a random permutation and K uniformly distributed, is upper bounded by:*

$$\max\left(1 - \exp\left(-\frac{\frac{M^2}{2} + 2MN}{2^c}\right), \frac{N}{2^{|K|}}\right) ,$$

where M is the data complexity and N the time complexity.

5 Discussion and conclusions

Theorem 1 has implications for all use cases where a keyed sponge function is used. In fact it imposes an upper bound to the success probability of attacks that are generic in f. We discuss some interesting use cases in the following subsections.

Note that in [7] we have proven upper bounds for the success probability of state retrieval attacks for keyed sponges for active and passive adversaries. Those bounds are tighter but are less generic than that of Theorem 1.

5.1 Safety margin with respect to a flat sponge claim

The bound provided in Theorem 1 only holds when the keyed sponge makes use of a random permutation. When a concrete permutation is taken, no such bounds can be given. See for example [8] and also [9] for discussions on this subject. However, it guarantees that using the sponge construction excludes generic attacks with a success probability higher than the sum of our bound and the success probability the attack would have for a random oracle.

In the *hermetic sponge strategy* [4] one adopts the sponge construction and builds an underlying permutation f that should not have any structural distinguishers. The capacity of the construction determines the claimed security level: one makes a *flat sponge claim* [4] with capacity matching the one of the construction. The flat sponge claim is a simplification in the sense that it considers only the worst-case success probability, determined by the sponge indifferentiability result of [3], which depends solely on the capacity of the random sponge. The bound in this paper shows that when being used in conjunction with a key, the success probability of generic attacks is much smaller than the flat sponge claim states. Hence, this can be seen as an extra safety margin.

5.2 Increasing the bitrate in a keyed sponge

In [6,7,5] we specify modes of use of the sponge construction where a key is pre-pended to the input. This includes MAC computation, stream encryption, authenticated encryption and reseedable pseudorandom sequence generators.

In many applications we can assume the implementation imposes an upper limit for M with a given key K, e.g. $M < 2^a$ with a the *usage exponent* with a value typically between 20 and 40. This may be due to physical constraints or imposed by the protocol. Assuming such a limit with $a \ll c/2$, the probability of inner collisions becomes negligible and the expected data complexity for distinguishing without key guessing is 2^{c-a-1}. In symmetric primitives that have a key, a typical security requirement is that there should be no attacks faster than exhaustive key search. For a keyed sponge this requirement translates to a lower bound for the capacity:

$$c \geq |K| + a + 1. \tag{4}$$

This is especially relevant in applications with constrained resources. Consider for instance an application requiring a keyed sponge with a required security level of 80-bits against all attacks, hence with an 80-bit key. According to the flat sponge claim this would require a capacity of at least 160 bits. If for efficiency reasons we impose that the bitrate is at least 20 % of the width of the permutation f this results in a bitrate $r = 40$ and a width $b = r + c = 200$. If we assume a usage exponent of at most 39, Equation (4) requires a capacity of only 120 bits. With the same efficiency requirement now a permutation f with width 150 bits is sufficient, and using a 200-bit permutation would double the bitrate to 80 bits.

5.3 Pre-image resistance

Usually pre-image resistance is defined as the infeasibility of finding an input x with a given output $y = \mathcal{H}(x)$. This can be modeled as a keyed sponge use case with $K = x$. Here the data complexity M is the number of r-blocks required to code x and y minus 1 and is a small integer ϵ, typically 1. The advantage of distinguishing the keyed sponge from a random oracle without exhaustive pre-image search then becomes:

$$\frac{\epsilon^2/2 + 2\epsilon N}{2^c}.$$

For a given y, the generic pre-image attack on a random oracle has expected time complexity $N = 2^{|y|}$. It follows that a sponge with capacity c offers the same level of pre-image resistance against generic attacks as a random oracle for outputs y with length $|y|$ up to $c - \log_2(\epsilon) - 1$. Note that the indifferentiability bound of [3] only guarantees the same for lengths $|y|$ up to $c/2$.

References

1. M. Bellare and P. Rogaway, *Random oracles are practical: A paradigm for designing efficient protocols*, ACM Conference on Computer and Communications Security 1993 (ACM, ed.), 1993, pp. 62–73.
2. G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, *Sponge functions*, Ecrypt Hash Workshop 2007, May 2007, also available as public comment to NIST from `http://www.csrc.nist.gov/pki/HashWorkshop/Public_Comments/2007_May.html`.
3. ———, *On the indifferentiability of the sponge construction*, Advances in Cryptology – Eurocrypt 2008 (N. P. Smart, ed.), Lecture Notes in Computer Science, vol. 4965, Springer, 2008, `http://sponge.noekeon.org/`, pp. 181–197.
4. ———, *Cryptographic sponges*, 2009, `http://sponge.noekeon.org/`.
5. ———, *Duplexing the sponge: single-pass authenticated encryption and other applications*, Second SHA-3 candidate conference, August 2010.
6. ———, Keccak *sponge function family main document*, NIST SHA-3 Submission (updated), June 2010, `http://keccak.noekeon.org/`.
7. ———, *Sponge-based pseudo-random number generators*, CHES (S. Mangard and F.-X. Standaert, eds.), Lecture Notes in Computer Science, vol. 6225, Springer, August 2010, pp. 33–47.
8. R. Canetti, O. Goldreich, and S. Halevi, *The random oracle methodology, revisited*, Proceedings of the 30th Annual ACM Symposium on the Theory of Computing, ACM Press, 1998, pp. 209–218.
9. U. Maurer, R. Renner, and C. Holenstein, *Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology*, Theory of Cryptography - TCC 2004 (M. Naor, ed.), Lecture Notes in Computer Science, no. 2951, Springer-Verlag, 2004, pp. 21–39.