

Subject: OFFICIAL COMMENT: Shabal

From: Søren Steffen Thomsen <S.Thomsen@mat.dtu.dk>

Date: Wed, 8 Apr 2009 14:09:52 +0200

To: "hash-function@nist.gov" <hash-function@nist.gov>

CC: "hash-forum@nist.gov" <hash-forum@nist.gov>

Dear all,

we (Lars Knudsen, Krystian Matusiewicz, and myself) made some observations on the keyed permutation used in Shabal, including fixed points and key-collisions in constant time. The observations do not seem to contradict any security claims made by the designers of Shabal.

Details in this note: <http://www.mat.dtu.dk/people/S.Thomsen/shabal/shabal.pdf>.

Best regards,
Søren.

--

Søren Steffen Thomsen
Postdoctoral researcher
DTU Mathematics

Technical University of Denmark
Department of Mathematics
Matematiktorvet 303S
Building 303S
2800 Kgs. Lyngby
Direct +45 4525 3010
Mobile +45 2290 5443
S.Thomsen@mat.dtu.dk
www.mat.dtu.dk/

Subject: OFFICIAL COMMENT: Shabal

From: Jean-Philippe Aumasson <jeanphilippe.aumasson@gmail.com>

Date: Tue, 14 Apr 2009 11:39:04 -0400

To: Multiple recipients of list <hash-forum@nist.gov>

We (Atefeh Mashatan, Willi Meier, and myself) made some observations on Shabal, including

- * a related-key distinguisher for its permutation with any number of rounds

- * non-trivial pseudo-collisions for a variant of Shabal that makes $24N$ iterations in the final loop of the permutation instead of 36, for any integer $N > 1$

More details can be found in

<http://www.131002.net/data/papers/AMM09.pdf>

From: hash-forum@nist.gov on behalf of Anne Canteaut [Anne.Canteaut@inria.fr]
Sent: Wednesday, May 06, 2009 2:11 AM
To: Multiple recipients of list
Subject: OFFICIAL comment: Shabal

Follow Up Flag: Follow up
Flag Status: Red

Attachments: distinguishers.pdf



distinguishers.pdf
(324 KB)

We (the Shabal team) have some new results on the security of Shabal's mode of operation, which point out that the round keyed permutation of Shabal does not need to be an ideal cipher to achieve the SHA-3 security requirements. The attached document actually provides with a new indistinguishability proof for Shabal's mode of operation where the keyed permutation is not assumed to be an ideal cipher anymore, but complies with a distinguishing property.

Most interestingly, we prove that the recent related-key distinguishers for Shabal's keyed permutation due to Knudsen et al. and to Aumasson et al. do not weaken the security of Shabal.

The Shabal team.

Indifferentiability with Distinguishers: Why Shabal Does Not Require Ideal Ciphers

Emmanuel Bresson, Anne Canteaut, Benoît Chevallier-Mames, Christophe Clavier,
Thomas Fuhr, Aline Gouget, Thomas Icart, Jean-François Misarsky,
María Naya-Plasencia, Pascal Paillier, Thomas Pornin, Jean-René Reinhard,
Céline Thuillet, Marion Videau

May 6, 2009

Abstract

Shabal is based on a new provably secure mode of operation. Some related-key distinguishers for the underlying keyed permutation have been exhibited recently by Aumasson *et al.* and Knudsen *et al.*, but with no visible impact on the security of Shabal. This paper then aims at extensively studying such distinguishers for the keyed permutation used in Shabal, and at clarifying the impact that they exert on the security of the full hash function. Most interestingly, a new security proof for Shabal's mode of operation is provided where the keyed permutation is not assumed to be an ideal cipher anymore, but observes a distinguishing property *i.e.*, an explicit relation verified by all its inputs and outputs. As a consequence of this extended proof, all known distinguishers for the keyed permutation are proven not to weaken the security of Shabal. In our study, we provide the foundation of a generalization of the indifferentiability framework to *biased* random primitives, this part being of independent interest.

1 Introduction

Shabal is one of the fastest unbroken candidate to the NIST hash competition. It is based on a new mode of operation, which is in some sense intermediate between the classical Merkle-Damgård construction and the sponge construction, and which is provably secure. In this mode of operation, depicted on Figure 1, the internal state is split into three parts A , B and C of respective sizes ℓ_a , ℓ_m and ℓ_m . At each message round, a new message block M of size ℓ_m is processed and the new internal state is obtained by

$$\begin{aligned}(A, B) &\leftarrow \mathcal{P}_{M,C}(A \oplus W, B \boxplus M) \\ (A, B, C) &\leftarrow (A, C \boxminus M, B)\end{aligned}$$

where W is a 64-bit counter, and \mathcal{P} is a keyed permutation over the set of $(\ell_a + \ell_m)$ -bit elements. Notation \boxplus (resp. \boxminus) corresponds to the wordwise addition (resp. subtraction) modulo 2^{32} .

Once the whole padded message has been processed, three final additional rounds are performed without incrementing the counter, and the hash value corresponds to the (truncated) C -part of the internal state. The parameter set used in the function submitted to NIST is $\ell_a = 384$ bits and $\ell_m = 512$ bits.

Shabal's mode of operation belongs to the class of *supercharged mode of operation* introduced by Stam [6]. The underlying design idea was to adapt the provably secure mode of operation of the sponge construction in order to use a permutation over a smaller set, which can be faster. This mode of operation has been proven secure in the ideal cipher model in [3, Chapter 5] in the following sense: it is indifferentiable from a random oracle up to $2^{\frac{\ell_a + \ell_m}{2}}$ evaluations of \mathcal{P} or \mathcal{P}^{-1} .

Therefore, this notion is quite different from the notion of “non-pseudorandomness of the round permutation”, which would mean that for a given value of the key, the permutation \mathcal{P} is not pseudorandom. The “non-pseudorandomness of the compression function” is not the relevant notion either: the compression function in Shabal’s mode of operation is not pseudorandom since it is proven collision-free (see [3, Th 6, Page 122]). Thus, the appropriate formulation of the consequence of related-key distinguishers is that *the keyed permutation can be distinguished from an ideal cipher*.

2.2 New distinguishers for \mathcal{P} and \mathcal{P}^{-1}

We first recall the definition of the keyed permutation \mathcal{P} . In Shabal, the choice of parameters p and r are $p = 3$ and $r = 12$.

```

Input:  $M, A, B, C$ 
Output:  $A, B$ 
for  $i$  from 0 to 15 do
     $B[i] \leftarrow B[i] \lll 17$ 
end for
for  $j$  from 0 to  $p - 1$  do
    for  $i$  from 0 to 15 do
         $A[i + 16j \bmod r] \leftarrow \mathcal{U}(A[i + 16j \bmod r] \oplus C[8 - i \bmod 16] \oplus \mathcal{V}(A[i - 1 + 16j \bmod r] \lll 15))$ 
         $A[i + 16j \bmod r] \leftarrow A[i + 16j \bmod r] \oplus M[i]$ 
         $A[i + 16j \bmod r] \leftarrow A[i + 16j \bmod r] \oplus B[i + 13 \bmod 16] \oplus (B[i + 9 \bmod 16] \wedge \overline{B[i + 6 \bmod 16]})$ 
         $B[i] \leftarrow (B[i] \lll 1) \oplus A[i + 16j \bmod r]$ 
    end for
end for
// final update
for  $j$  from 0 to 35 do
     $A[j \bmod r] \leftarrow A[j \bmod r] + C[j + 3 \bmod 16]$ 
end for

```

Efficient distinguishers for \mathcal{P}^{-1} have been presented in Shabal’s submission document [3]. A much more expensive distinguisher based on a cube tester was then presented by Aumasson in [1], — it is worth noticing that the description given in [1] is erroneous since this distinguisher also requires the knowledge of C (otherwise the final update on A cannot be inverted). The related-key distinguisher exhibited in [4, 2] exploits the fact that, for some differences $\Delta_1, \Delta_2 \in \{0, 1\}^{\ell_m}$, the images of any fixed input (A, B) for both pairs of parameters (M, C) and $(M \oplus \Delta_1, C \oplus \Delta_2)$ are equal. The interesting point here is that this property does not depend on the number of loops p .

The main property used in most of these related-key distinguishers, which has been discussed in Shabal’s submission document, originates from the structure of \mathcal{P}^{-1} . In turn, the words of the B -part of the output of \mathcal{P}^{-1} do not depend on all the words of parameter M . Using the same tools as in the distinguishers presented in Shabal’s documentation and an exhaustive search, we have found the best possible related-key distinguishers for $p = 3$. These distinguishers are all derived from the following basic dependence relations.

The technique used for finding the basic relations relies on the fact that two different types of equations can be used for computing the output B of \mathcal{P}^{-1} :

$$A[i + 12] = \mathcal{U}(A[i] \oplus \mathcal{V}(A[i + 11] \lll 15) \oplus C[8 - i] \oplus \overline{B[i + 6]} B[i + 9] \oplus B[i + 13] \oplus M[i]) \quad (1)$$

$$(B[i] \lll 1) = B[i + 16] \oplus \overline{A[i + 12]}. \quad (2)$$

With those relations, we have been able to compute each word of B without having to know the whole message, M . Those relations are summarized in Table 1, where 1 (resp. 0) at the

intersection of row $B[i]$ and column $M[j]$ means that $B[i]$ depends (resp. does not depend) on $M[j]$. The next example shows how the dependencies for $B[15]$ are determined.

	$M[0]$	$M[1]$	$M[2]$	$M[3]$	$M[4]$	$M[5]$	$M[6]$	$M[7]$	$M[8]$	$M[9]$	$M[10]$	$M[11]$	$M[12]$	$M[13]$	$M[14]$	$M[15]$
$B[15]$	0	0	0	0	0	0	1	1	0	0	0	1	0	1	0	0
$B[14]$	0	0	0	0	0	1	1	0	0	0	1	0	1	0	0	1
$B[13]$	0	0	0	0	1	1	0	0	0	1	0	1	0	0	1	1
$B[12]$	0	0	0	1	1	0	0	0	1	0	1	0	0	1	1	1
$B[11]$	0	0	1	1	0	0	0	1	0	1	0	0	1	1	1	1
$B[10]$	0	1	1	0	0	0	1	0	1	0	0	1	1	1	1	1
$B[9]$	1	1	0	0	0	1	0	1	0	0	1	1	1	1	1	0
$B[8]$	1	0	0	0	1	0	1	0	0	1	1	1	1	1	0	1
$B[7]$	0	0	0	1	0	1	0	0	1	1	1	1	1	0	1	1
$B[6]$	0	0	1	0	1	0	0	1	1	1	1	1	0	1	1	1
$B[5]$	0	1	0	1	0	0	1	1	1	1	1	0	1	1	1	1
$B[4]$	1	0	1	0	0	1	1	1	1	1	0	1	1	1	1	0
$B[3]$	0	1	0	0	1	1	1	1	1	0	1	1	1	1	0	1
$B[2]$	1	0	0	1	1	1	1	1	0	1	1	1	1	0	1	1
$B[1]$	0	0	1	1	1	1	1	0	1	1	1	1	0	1	1	1
$B[0]$	0	1	1	1	1	1	0	1	1	1	1	0	1	1	1	1

Table 1: Dependence relations between the words of the B -part of the output of \mathcal{P}^{-1} and the words of M .

Example: a relation for $B[15]$. Here, we show how $B[15]$ can be computed from the inputs of \mathcal{P}^{-1} , A' and B' , and given C and only four words of M :

- From C and A' , we can invert the final update of A and compute the 12 words $A[48], \dots, A[59]$.
- Using Equation (2) for i from 36 to 47, we compute $B[36], \dots, B[47]$.
- Now we use Equation (1).
 From (1) for $i = 38$ we obtain that $A[38]$ depends on $A[49], A[50], B[44], B[47], B[51], C[2], M[6]$.
 From (1) for $i = 39$ we obtain that $A[39]$ depends on $A[50], A[51], B[45], B[48], B[52], C[1], M[7]$.
 From (1) for $i = 43$ we obtain that $A[43]$ depends on $A[54], A[55], B[49], B[52], B[56], C[13], M[11]$.
 From (1) for $i = 45$ we obtain that $A[45]$ depends on $A[56], A[57], B[51], B[54], B[58], C[11], M[13]$.
- Now using (2) for $i = 33$, we obtain $B[33]$ from $B[49]$ and $A[45]$.
 Using (2) for $i = 31$, we obtain $B[31]$ from $B[47]$ and $A[43]$.
 Using (2) for $i = 27$, we obtain $B[27]$ from $B[43]$ and $A[39]$.
- We apply (1) for $i = 27$, in order to compute $A[27]$ which depends on $A[38], A[39], B[33], B[36], B[40], C[13], M[11]$.
- Using (2) for $i = 15$, we finally obtain $B[15]$ from $B[31]$ and $A[27]$.

Thus, $B[15]$ depends on $M[6], M[7], M[11], M[13]$ only.

A related-key distinguisher for \mathcal{P}^{-1} . With the previously shown algorithm, we can build a distinguisher on \mathcal{P}^{-1} in a trivial way which works with a single query, since an adversary is able to compute some words of the output of \mathcal{P}^{-1} without calling \mathcal{P} .

1. The challenger chooses (A', B') and (M, C) .
2. The adversary knows C and the four words $M[6], M[7], M[11], M[13]$ only. He makes the query $\mathcal{P}_{M,C}^{-1}(A', B')$.
3. From (A', B') and four known words of M , he computes $B[15]$ and checks whether this corresponds to the value of $B[15]$ obtained in the response.

A distinguisher for \mathcal{P} . We also have a distinguisher for \mathcal{P} with makes two queries in the following model:

1. The challenger chooses randomly (A, B) and two sets of parameters $(M, C), (M', C')$ with $M[i] = M'[i]$ for $i \in \{6, 7, 11, 13\}$.
2. The adversary knows C, C' and $M[6], M[7], M[11], M[13]$; the other part of M and M' is unknown. He makes the queries $\mathcal{P}_{M,C}(A, B)$ and $\mathcal{P}_{M',C'}(A, B)$.
3. From the responses, he computes $B[15]$ and checks whether he gets the same value.

By using the previous independence relationships, it appears that from any value (A', B', C) , it is possible to choose some M in such a way that certain words of B are equal to a target value. Then, we can show that the highest number of words of B before the message insertion which can be fixed to a target value is equal to 7. However, extending this property to the whole compression function is much more difficult, because of the final update of A . Moreover, these distinguishers have no impact on the security of *Shabal*, as shown later on. We also comment that in the context of (second)-preimage attacks, when computing forwards, the value of C used in the permutation is fixed. But when computing backwards the value of C will be $B' + M$. If we want to use the previous property to fix say, 7 words of B , then we must know C before being able to determine M . As a result, and because C depends on M , we cannot apply the observed property.

2.3 Distinguishers on the compression function \mathcal{R}

When considering not only \mathcal{P} , but the whole function $\mathcal{R}^{\mathcal{P}}: (A, B, C, M) \mapsto (A', B', C')$ defined by

$$\begin{aligned} (A', C') &= \mathcal{P}_{M,C}(A, B \boxplus M) \\ B' &= C \boxplus M, \end{aligned}$$

no distinguisher has been presented so far. Now, when computing backwards, it is impossible to determine some word of B from the knowledge of (A', B', C') and of some words of M only. The reason is that \mathcal{P}^{-1} is parameterized by $C = B' \boxplus M$ which depends on M , and this C is used in the final update of A (*i.e.*, in the first operation in \mathcal{P}^{-1}). Even if computing each $B[i]$ involves all words of M , it may involve fewer information words of M only. For instance, $B[15]$ is completely determined by $M[1, 2, 3, 5, 6, 7, 9, 10, 11, 13, 14, 15]$ and by $M[4] \boxplus M[8] \boxplus M[12]$ and $M[0] \boxplus M[4] \boxplus M[12]$. We have performed an exhaustive search, based on this final update, in order to determine the number of information bits of M which are involved for computing each word of B . Using the previously described technique, we obtain that only the three words $B[15], B[14]$ or $B[11]$ do not depend of all information words of M . It is worth noticing that the independencies of these three words are different, implying that computing the triple $(B[11], B[14], B[15])$ requires the knowledge of all information words of M .

This type of distinguishers affects the resistance of the hash function to (second) preimage attacks. As explained in [3, Ch. 12], the attacker is able, by computing backwards, to choose several message blocks which lead to a given target value for three words of B . Therefore, a (second)-preimage attack on *Shabal* with complexity $2^{\frac{\ell_a + 2\ell_m - 3 \times 32}{2}}$, instead of $2^{\frac{\ell_a + 2\ell_m}{2}}$, might be mounted. However, this complexity is still much higher than for the generic attack for the largest size of the message digest ℓ_h which is 512 bits. Since no better distinguisher of this type has been found by an exhaustive search, this is the best attack which could be performed on *Shabal* using this type of distinguishers.

3 Indifferentiability with distinguishers: why *Shabal* does not require ideal ciphers

This rest of this document considers the following question. Assume that we are given a hash function $\mathcal{C}^{\mathcal{P}}$ which is made of an operating mode \mathcal{C} making calls to an internal primitive \mathcal{P} .

Assume further that \mathcal{C} is proven to be indistinguishable from a random oracle, which in turn means that $\mathcal{C}^{\mathcal{P}}$ behaves ideally assuming that \mathcal{P} behaves ideally. Now when specifying an instantiation of the hash construction, one has to select a functional embodiment for \mathcal{P} and provide a full-fledge description of the primitive \mathcal{P} . Assume now that the specified primitive does not behave ideally (or at least not in the sense required by the indistinguishability proof for \mathcal{C}). This can be reformulated by saying that there exists some non-trivial statistical relation \mathcal{R} which connects inputs and outputs of \mathcal{P} . It seems at first sight that the indistinguishability result on \mathcal{C} does not constitute a security argument anymore since the basic requirement of the proof (\mathcal{P} behaves ideally) is obviously not obeyed. The question we ask is whether $\mathcal{C}^{\mathcal{P}}$ can still be considered as a good hash function.

At a higher level, the question relates to a more general paradigm: can we prove that a construction $\mathcal{C}^{\mathcal{P}}$ behaves ideally even though \mathcal{P} does not? More generally, we may ask ourselves to which extent $\mathcal{C}^{\mathcal{P}}$ differs from an ideal hash function when \mathcal{P} differs from an ideal primitive. Obviously, it is desirable that a construction \mathcal{C} remain close to ideal even when \mathcal{P} is far from ideal. One may think of this notion as a form of robustness: even if a weakness is discovered on the full-fledge primitive \mathcal{P} in the future, the hash function $\mathcal{C}^{\mathcal{P}}$ would remain almost equally ideal. Thus our motivation is driven by practical considerations; robust indistinguishable constructions answer the quest for more durable hash constructions.

We provide a proof that the hash construction *Shabal* is robust in the above sense. We give a proper definition of robustness for *Shabal* and fully describe an extended proof methodology in the indistinguishability framework which captures this notion. Our results show that *Shabal* behaves ideally even when powerful distinguishers are known on the inner keyed permutation \mathcal{P} . We provide a precise and quantitative security bound as a function of the statistical biases introduced by the distinguisher on \mathcal{P} .

3.1 Capturing distinguishers into indistinguishability proofs

The indistinguishability framework. We focus on the indistinguishability proof of *Shabal*'s mode of operation. Recall that the concept of indistinguishability [5] specifies a security game played between an oracle system Q and a distinguisher \mathcal{D} . Q may contain several components, typically a cryptographic construction $\mathcal{C}^{\mathcal{P}}$ which calls some inner primitive \mathcal{P} . Construction \mathcal{C} is said to be indistinguishable up to a certain security bound if the system $Q = (\mathcal{C}^{\mathcal{P}}, \mathcal{P})$ can be replaced by a second oracle system $Q' = (\mathcal{H}, \mathcal{S}^{\mathcal{H}})$ with identical interface in such a way that \mathcal{D} cannot tell the difference (see Figure 2). Here \mathcal{H} is a random oracle and \mathcal{S} is a simulator which must behave like \mathcal{P} . In the case of *Shabal*'s mode of operation, \mathcal{S} corresponds to a simulator of both \mathcal{P} and \mathcal{P}^{-1} .

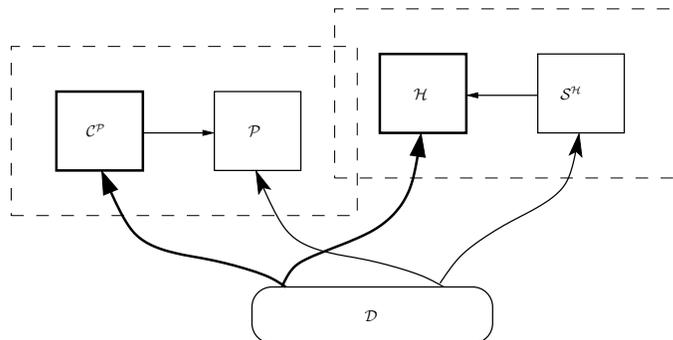


Figure 2: The cryptographic construction $\mathcal{C}^{\mathcal{P}}$ has oracle access to \mathcal{P} . The simulator $\mathcal{S}^{\mathcal{H}}$ has oracle access to the random oracle \mathcal{H} . The distinguisher interacts either with $Q = (\mathcal{C}^{\mathcal{P}}, \mathcal{P})$ or with $Q' = (\mathcal{H}, \mathcal{S}^{\mathcal{H}})$ and has to tell them apart.

In its interaction with the system Q or Q' , the distinguisher makes *left calls* to either $\mathcal{C}^{\mathcal{P}}$ or \mathcal{H} and *right calls* to either \mathcal{P} or $\mathcal{S}^{\mathcal{H}}$. We will call N the *total* number of right calls *i.e.*, the number

of calls received by \mathcal{P} when \mathcal{D} interacts with Q – regardless of their origin which may be either $\mathcal{C}^{\mathcal{P}}$ or \mathcal{D} . We define the advantage of distinguisher \mathcal{D} as

$$\text{Adv}(\mathcal{D}) = |\Pr [\mathcal{D}^{\mathcal{Q}} = 1 \mid \mathcal{Q} = (\mathcal{C}^{\mathcal{P}}, \mathcal{P})] - \Pr [\mathcal{D}^{\mathcal{Q}} = 1 \mid \mathcal{Q} = (\mathcal{H}, \mathcal{S}^{\mathcal{H}})]|$$

where probabilities are taken over the random coins of all parties. Obviously $\text{Adv}(\mathcal{D})$ is a function of N . The indistinguishability proof therefore consists in constructing an appropriate simulator of \mathcal{P} and \mathcal{P}^{-1} and in estimating the advantage of the corresponding distinguisher.

Capturing distinguishers in the indistinguishability framework. In our extended indistinguishability framework, the distinguishing algorithm \mathcal{D} is also submitted to a random experiment where \mathcal{D} interacts with the oracle system Q . However, the ideal primitive \mathcal{P} is not considered as being ideal anymore: instead of being a keyed permutation uniformly selected from the space PERM of all keyed permutations, \mathcal{P} is now randomly selected from some subspace $\text{PERM}[\mathcal{R}] \subseteq \text{PERM}$. The subspace $\text{PERM}[\mathcal{R}]$ is defined as the collection of all keyed permutations $(k, x) \mapsto \mathcal{P}(k, x) = y$ such that

$$\mathcal{R}(k, x, y) = 1$$

holds for any tuple (k, x, y) , for some explicitly given relation \mathcal{R} . It is understood that \mathcal{R} is some formula that relates inputs, parameters and outputs of \mathcal{P} and that \mathcal{R} has nothing to do with an oracle: it is an explicit predicate that \mathcal{D} knows and which can be "hardwired" in the code of \mathcal{D} . We comment that \mathcal{R} provides a test to decide whether a given function $\mathcal{P} \leftarrow \text{PERM}$ belongs or not to $\text{PERM}[\mathcal{R}]$; one simply evaluates \mathcal{P} or \mathcal{P}^{-1} once on random values to get a tuple (k, x, y) and tests whether the predicate holds for these values. Hence we view \mathcal{R} as a distinguishing algorithm for $\text{PERM}[\mathcal{R}]$ and we alternately refer to \mathcal{R} as a distinguisher on \mathcal{P} . Reminding that this distinguisher has nothing to do with the distinguisher \mathcal{D} : \mathcal{D} tells apart the two oracle systems Q and Q' , whereas \mathcal{R} captures a statistical constraint (*i.e.*, a bias) on the input-output behavior of \mathcal{P} .

Let us now assume that the relation \mathcal{R} is fixed and given. We consider the above security game where instead of defining a "perfect" ideal cipher for \mathcal{P} , \mathcal{P} is replaced with a "biased" ideal cipher in the sense that \mathcal{P} is drawn uniformly at random from $\text{PERM}[\mathcal{R}]$ (instead of PERM) during the game. This defines a new security game where the advantage of \mathcal{D} becomes

$$\text{Adv}(\mathcal{D}, \mathcal{R}) = |\Pr [\mathcal{D}^{\mathcal{Q}} = 1 \mid \mathcal{Q} = (\mathcal{C}^{\mathcal{P}}, \mathcal{P})] - \Pr [\mathcal{D}^{\mathcal{Q}} = 1 \mid \mathcal{Q} = (\mathcal{H}, \mathcal{S}^{\mathcal{H}})]|$$

where probabilities are taken over the random coins of all parties (this is again some function of N but which now depends on \mathcal{R} as well). We say that \mathcal{C} is indistinguishable with respect to \mathcal{R} when \mathcal{D} 's advantage remains negligibly small.

It is important to note that distinguishers arising from a known input-output relation \mathcal{R} are a special class of distinguishers on \mathcal{P} . In the general case, a distinguisher on \mathcal{P} is a probabilistic algorithm which adaptively interacts with an oracle instantiated with either \mathcal{P} or an ideal version of \mathcal{P} and eventually outputs a guess on the instantiation. To do so, the distinguisher attempts to detect that a relation holds (with some extra probability) over a series of input-output data, parts of which have been chosen adaptively. Here, we consider a class of particularly strong distinguishers on \mathcal{P} which are based on a direct and explicit relation which *always* holds on *any* input-output tuple. Although we feel that indistinguishability can be extended to take general distinguishers on \mathcal{P} into account, we are mostly interested in strong distinguishers since we actually know examples of such relations \mathcal{R} preserved by the specific primitive \mathcal{P} of Shabal. We therefore focus on this only case, noting that our extension of indistinguishability is also simpler to define.

Defining the bias of \mathcal{R} . We need a metric to tell "how severe" is the constraint imposed on \mathcal{P} by the given relation \mathcal{R} , as opposed to an arbitrary, unconstrained choice of \mathcal{P} in the space PERM . There may be several ways to define such a metric; we adopt two that are strong enough for our purposes. In Shabal, the keyed permutation \mathcal{P} takes an input $(A, B) \in \{0, 1\}^{\ell_a} \times \{0, 1\}^{\ell_m}$ and parameter $(M, C) \in \{0, 1\}^{\ell_m} \times \{0, 1\}^{\ell_m}$ and outputs a pair $(A', B') \in \{0, 1\}^{\ell_a} \times \{0, 1\}^{\ell_m}$. Let

$\mathcal{R}(M, A, B, C, A', B')$ be an input-output relation for \mathcal{P} and consider that M, A, B, C, B' are fixed. Let us consider the set

$$\text{PERM}[\mathcal{R}, M, A, B, C, B'] \subseteq \text{PERM}[\mathcal{R}]$$

of all keyed permutations \mathcal{P} such that $\mathcal{P}_{M,C}(A, B) = (A', B')$ for some $A' \in \{0, 1\}^{\ell_a}$. We will say that $\text{ForSamp}_{\mathcal{R}}$ is a (forward) sampling algorithm for \mathcal{R} when on any input tuple (M, A, B, C, B') , $\text{ForSamp}_{\mathcal{R}}(M, A, B, C, B')$ selects a keyed permutation $\mathcal{P} \leftarrow \text{PERM}[\mathcal{R}, M, A, B, C, B']$ uniformly at random and outputs A' . We will assume wlog that given a relation \mathcal{R} , one can construct such a sampling algorithm $\text{ForSamp}_{\mathcal{R}}$ and that $\text{ForSamp}_{\mathcal{R}}$ can be implemented efficiently. We view $\text{ForSamp}_{\mathcal{R}}$ as an algorithmic representation of \mathcal{R} and the related-key distinguishers discussed in the previous sections can be reformulated by making their sampling algorithm explicit.

Definition 1 (Forward bias of \mathcal{R}). *\mathcal{R} is said to have forward bias $\tau \in (0, \ell_a)$ if for any choice of M, A, B, C and A' , it holds that*

$$\Pr [B' \leftarrow \{0, 1\}^{\ell_m} : \text{ForSamp}_{\mathcal{R}}(M, A, B, C, B') = A'] \leq 2^{-(\ell_a - \tau)}$$

where the probability is taken over the uniform choice of $B' \leftarrow \{0, 1\}^{\ell_m}$ and the internal coins of the algorithm $\text{ForSamp}_{\mathcal{R}}$ in the random selection of $\mathcal{P} \leftarrow \text{PERM}[\mathcal{R}, M, A, B, C, B']$.

As soon to be discussed, we also need to define a *backward bias* for \mathcal{R} . However in this case, the bias is simpler to define. We consider a second sampling algorithm $\text{BacSamp}_{\mathcal{R}}$ which, taking as input a tuple (M, A', B', C) , randomly selects a keyed permutation $\mathcal{P} \leftarrow \text{PERM}[\mathcal{R}]$ (with uniform distribution) and outputs the pair $(A, B) = \mathcal{P}_{M,C}^{-1}(A', B')$. Again, algorithm $\text{BacSamp}_{\mathcal{R}}$ is a reformulation of the Boolean relation \mathcal{R} and we assume that it can always be made explicit. The backward bias is defined as the statistical bias of $\text{BacSamp}_{\mathcal{R}}$:

Definition 2 (Backward bias of \mathcal{R}). *\mathcal{R} is said to have backward bias $\lambda \in (0, \ell_a + \ell_m)$ if for any choice of M, A', B', C and A, B it holds that*

$$\Pr [\text{BacSamp}_{\mathcal{R}}(M, A', B', C) = (A, B)] \leq 2^{-(\ell_a + \ell_m - \lambda)}$$

where the probability is taken over the internal random choice of $\mathcal{P} \leftarrow \text{PERM}[\mathcal{R}]$.

We leave it as an open problem to analytically link the two biases τ and λ in the general case. However, we comment that a careful study of a given relation \mathcal{R} (e.g., the related-key distinguishers discussed above) should provide at least numerical values for τ and λ . We then give a quantitative security bound that tells how far from a random oracle the mode of operation of *Shabal* behaves as a function of the two biases introduced by the relation \mathcal{R} . The following sections expose the original security proof [3] of *Shabal* and extend the proof to encompass distinguishers on \mathcal{P} . This can be seen as a generalization of the original proof to distinguishing relations with non-zero bias τ and λ . We end up with a simple extension of the indistinguishability bound that includes non-zero forward and backward bias and which we state as a Theorem.

3.2 The original security proof in the plain ideal cipher model

A general game-based indistinguishability proof technique is presented in [3]. In order to analyze the mode of operation in a more fluent fashion within the proof, we view the current message block as a part of the current internal state. The set of all possible internal states, which we denote by \mathcal{X} contains all the $(3\ell_m + \ell_a)$ -bit strings. In this presentation, we simplify *Shabal*'s mode of operation by removing the influence of the counter and the three final rounds (all these features are taken into account in the original proof). It is easily seen that the proof can be extended to take these into account and that they have little influence on the security bound. Thus, we assume that the i -th message round executes two subroutines: message insertion $\text{Insert}[M_i]$ defined as

$$\text{Insert}[M_i](M_{i-1}, A, B, C) = (M_i, A, C \boxplus M_{i-1} \boxplus M_i, B)$$

and \mathcal{P} . The initial internal state is the initialization vector $x_0 = (0, A_0, B_0, C_0)$. We will sketch the proof and rely on [3] for a detailed exposition of the methodology. The simulator of $(\mathcal{P}, \mathcal{P}^{-1})$ is obtained by dynamically constructing a graph $\mathcal{G} = (X, Y, Z) \subseteq \mathcal{X} \times \mathcal{X} \times \mathcal{X}^2$ where $X \cup Y$ is the set of nodes and Z the set of edges. Y is the set of queries to \mathcal{P} received by \mathcal{S} and X the set of responses returned by \mathcal{S} (completed with the M -part and the C -part of their preimage to yield a proper internal state $\in \mathcal{X}$). X also contains the input queries to \mathcal{P}^{-1} in which case their outputs are appended to Y . An edge between y and x in the graph is denoted by $y \xrightarrow{\mathcal{P}} x$. A path from the initial state x_0 to $x \in \mathcal{X}$ in the graph is a non-empty list of ℓ_m -bit message blocks $\mu = \langle M_1, \dots, M_k \rangle$ such that there exist k edges in the graph of the form $y_i \xrightarrow{\mathcal{P}} x_i$, $1 \leq i < k$, and $y_{k-1} \xrightarrow{\mathcal{P}} x$ satisfying

$$\text{Insert}[M_{i+1}](x_i) = y_{i+1}, \quad 0 \leq \forall i < k.$$

A path to $y \in Y$ is defined in a similar fashion, the path of y being defined as the path of the only $x \in X$ such that $y \xrightarrow{\mathcal{P}} x$. We use the simulator \mathcal{S} defined in Figure 3. The goal of \mathcal{S} consists in keeping generating associations $y \mapsto \mathcal{P}(y)$ and $x \mapsto \mathcal{P}^{-1}(x)$ for inputs $x, y \in \mathcal{X}$ chosen by \mathcal{D} which are consistent with the values output by \mathcal{H} . Again we refer to [3] for details on the overall proof technique.

The following result holds:

Theorem 1. *Assume \mathcal{P} is an ideal cipher and let \mathcal{H} be a random oracle. Then, the simulator \mathcal{S} defined as per Figure 3 is such that for any distinguisher \mathcal{D} totalling at most N right calls to \mathcal{P} and \mathcal{P}^{-1} ,*

$$\text{Adv}(\mathcal{D}) \leq \Pr[\text{Abort}_1] + \Pr[\text{Abort}_2] + \Pr[\text{Abort}_4] + \Pr[\text{Abort}_5] + N \cdot 2^{-(\ell_a + \ell_m)}.$$

Moreover, all four probabilities $\Pr[\text{Abort}_i]$ are upper-bounded by $\frac{N(N-1)}{2} 2^{-(\ell_a + \ell_m)}$.

We now discuss the above result in more detail. The original proof, taken away the final rounds and the counter W , shows that

$$\Pr[\text{Abort}_1] \leq \frac{N(N-1)}{2} \cdot p_1, \quad \Pr[\text{Abort}_2] \leq \frac{N(N-1)}{2} \cdot p_2, \quad (3)$$

$$\Pr[\text{Abort}_4] \leq \frac{N(N-1)}{2} \cdot p_4, \quad \Pr[\text{Abort}_5] \leq \frac{N(N-1)}{2} \cdot p_5, \quad (4)$$

where p_1, p_2, p_4, p_5 are probability bounds defined as follows.

Probability bound p_1 . Let us fix $\tilde{x} = (\tilde{M}, \tilde{A}, \tilde{B}, \tilde{C}) \in \mathcal{X}$ as well as $y = (M, A, B, C) \in \mathcal{X}$, and let us consider the distribution

$$D(y) = \{(M, A', B', C) \mid (A', B') \leftarrow \{0, 1\}^{\ell_a} \times \{0, 1\}^{\ell_m}\}.$$

Then, taking probabilities over the uniformly random selection $x \leftarrow D(y)$ we define

$$\begin{aligned} p_1(\tilde{x}, y) &= \Pr[\exists \mathbf{m}, \tilde{\mathbf{m}} \in \{0, 1\}^{\ell_m} : \text{Insert}[\mathbf{m}](x) = \text{Insert}[\tilde{\mathbf{m}}](\tilde{x})] \\ &= \Pr \left[\begin{array}{l} \exists \mathbf{m}, \tilde{\mathbf{m}} \in \{0, 1\}^{\ell_m} : \\ \mathbf{m} = \tilde{\mathbf{m}} \\ A' = \tilde{A} \\ B' = \tilde{B} \\ C \boxplus M \boxplus \mathbf{m} = \tilde{C} \boxplus \tilde{M} \boxplus \tilde{\mathbf{m}} \end{array} \right] \\ &= \Pr[(A', B') = (\tilde{A}, \tilde{B})] \cdot \delta[C \boxplus M = \tilde{C} \boxplus \tilde{M}] \end{aligned}$$

We then upper bound $p_1(\tilde{x}, y)$ by

$$p_1 = \max_{\tilde{A}, \tilde{B}} \Pr[(A', B') = (\tilde{A}, \tilde{B})]$$

and it is obvious that $p_1 = 2^{-(\ell_a + \ell_m)}$. We remind that $\delta[E]$ returns 1 when event E is realized and 0 otherwise.

<p>Initialization of \mathcal{S}</p> <p>No input, no output</p> <ol style="list-style-type: none"> 1. set $X = Y = Z = \emptyset$
--

<p>Simulation of \mathcal{P}</p> <p>Input: $y = (M, A, B, C) \in \mathcal{X}$</p> <p>Output: (A', B')</p> <ol style="list-style-type: none"> 1. add node y to Y 2. if there exists an edge $y \xrightarrow{\mathcal{P}} x \in Z$ <ol style="list-style-type: none"> (a) return (A', B') where $x = (M, A', B', C)$ 3. if y has a path μ in graph \mathcal{G} <ol style="list-style-type: none"> (a) compute $\mathcal{M} = \text{unpad}(\mu)$ (b) call \mathcal{H} to get $h = \mathcal{H}(\mathcal{M})$ (c) set $B' = h$ (d) randomly select $A' \leftarrow \{0, 1\}^{\ell_a}$ 4. else <ol style="list-style-type: none"> (a) randomly select $B' \leftarrow \{0, 1\}^{\ell_m}$ (b) randomly select $A' \leftarrow \{0, 1\}^{\ell_a}$ 5. add node $x = (M, A', B', C)$ to X and edge $y \xrightarrow{\mathcal{P}} x$ to Z 6. if for some $\tilde{x} \in X$ one has $\text{Insert}[\tilde{M}](\tilde{x}) = \text{Insert}[M](x)$ for some M and \tilde{M} (event Abort₁), then abort. 7. if x admits a path in \mathcal{G} and $\exists \tilde{y} \in Y$ such that $\text{Insert}[M](x) = \tilde{y}$ for some M (event Abort₂), then abort. 8. return (A', B')

<p>Simulation of \mathcal{P}^{-1}</p> <p>Input: $x = (M, A, B, C) \in \mathcal{X}$</p> <p>Output: (A', B')</p> <ol style="list-style-type: none"> 1. add node x to X 2. if there exists an edge $y \xrightarrow{\mathcal{P}} x \in Z$ <ol style="list-style-type: none"> (a) return (A', B') where $y = (M, A', B', C)$ 3. randomly select $A' \leftarrow \{0, 1\}^{\ell_a}$ and $B' \leftarrow \{0, 1\}^{\ell_m}$ 4. add node $y = (M, A', B', C)$ to X and edge $y \xrightarrow{\mathcal{P}} x$ to Z 5. if $\exists \tilde{x} \in X$, $\tilde{x} \neq x$ such that $y \xrightarrow{\mathcal{P}} \tilde{x} \in Z$ (event Abort₄), then abort 6. if $\exists \tilde{x} \in X$ such that $\text{Insert}[M](\tilde{x}) = y$ for some M (event Abort₅), then abort 7. return (A', B')
--

Figure 3: Original simulator \mathcal{S} for \mathcal{P} and \mathcal{P}^{-1} in the plain ideal cipher model.

Probability bound p_2 . Let us now fix $\tilde{y} = (\tilde{M}, \tilde{A}, \tilde{B}, \tilde{C}) \in \mathcal{X}$ as well as $y = (M, A, B, C) \in \mathcal{X}$. Then, taking probabilities over the uniformly random selection $x \leftarrow D(y)$ we define

$$\begin{aligned}
p_2(\tilde{y}, y) &= \Pr [\exists \mathbf{m} \in \{0, 1\}^{\ell_m} : \text{Insert}[\mathbf{m}](x) = \tilde{y}] \\
&= \Pr \left[\exists \mathbf{m} \in \{0, 1\}^{\ell_m} : \begin{array}{l} \mathbf{m} = \tilde{M} \\ A' = \tilde{A} \\ C \boxplus M \boxplus \mathbf{m} = \tilde{B} \\ B' = \tilde{C} \end{array} \right] \\
&= \Pr [(A', B') = (\tilde{A}, \tilde{C})] \cdot \delta [C \boxplus M \boxplus \tilde{M} = \tilde{B}]
\end{aligned}$$

An upper bound for $p_2(\tilde{y}, M, C)$ is then

$$p_2 = \max_{\tilde{A}, \tilde{C}} \Pr \left[(A', B') = (\tilde{A}, \tilde{C}) \right]$$

and again it is obvious that $p_2 = 2^{-(\ell_a + \ell_m)}$.

Probability bound p_4 . Let us fix $\tilde{x} = (\tilde{M}, \tilde{A}, \tilde{B}, \tilde{C}) \in \mathcal{X}$ and $y = (M, A, B, C) \in \mathcal{X}$. Taking probabilities over the random selection $x \leftarrow D(y)$ we set

$$\begin{aligned} p_4(\tilde{y}, y) &= \Pr [x = \tilde{x}] \\ &= \Pr \left[\begin{array}{l} M = \tilde{M} \\ A' = \tilde{A} \\ B' = \tilde{B} \\ C = \tilde{C} \end{array} \right] \\ &= \Pr \left[(A', B') = (\tilde{A}, \tilde{B}) \right] \cdot \delta \left[M = \tilde{M} \wedge C = \tilde{C} \right] \end{aligned}$$

and an upper bound for $p_4(\tilde{y}, y)$ is then

$$p_4 = \max_{\tilde{A}, \tilde{B}} \Pr \left[(A', B') = (\tilde{A}, \tilde{B}) \right] = 2^{-(\ell_a + \ell_m)}.$$

Probability bound p_5 . Again we fix $\tilde{x} = (\tilde{M}, \tilde{A}, \tilde{B}, \tilde{C}) \in \mathcal{X}$ and $x = (M, A, B, C) \in \mathcal{X}$, and consider the following probability taken over the random selection $y \leftarrow D(x)$:

$$\begin{aligned} p_5(\tilde{y}, x) &= \Pr \left[\exists \mathbf{m} \in \{0, 1\}^{\ell_m} : \text{Insert}[\mathbf{m}](\tilde{x}) = y \right] \\ &= \Pr \left[\begin{array}{l} \mathbf{m} = M \\ \tilde{A} = A' \\ \tilde{C} \boxplus \tilde{M} \boxplus \mathbf{m} = B' \\ \tilde{B} = C \end{array} \right] \\ &= \Pr \left[(A', B') = (\tilde{A}, \tilde{C} \boxplus \tilde{M} \boxplus M) \right] \cdot \delta \left[\tilde{B} = C \right] \end{aligned}$$

Then we bound $p_5(\tilde{y}, x)$ again by

$$p_5 = \max_{\tilde{A}, \tilde{C}, \tilde{M}, M} \Pr \left[(A', B') = (\tilde{A}, \tilde{C} \boxplus \tilde{M} \boxplus M) \right] = 2^{-(\ell_a + \ell_m)}.$$

Combining these results, we get an indifferentiability bound of

$$\text{Adv}(\mathcal{D}) \leq N(2N - 1) \cdot 2^{-(\ell_a + \ell_m)}$$

which shows that the mode of operations of Shabal (in this simplified version) behaves like a random oracle up to roughly $2^{\frac{(\ell_a + \ell_m)}{2}} = 2^{448}$ calls to its primitive \mathcal{P} , since $\ell_m = 512$ and $\ell_a = 384$.

3.3 Extending the proof to biased permutations \mathcal{P}

We now assume that \mathcal{P} is not an ideal cipher anymore *i.e.*, randomly selected from PERM but that there exists some biased relation \mathcal{R} for \mathcal{P} . In particular, there exists a set of known Boolean relations which hold for all pairs $((M, A, B, C), (A', B'))$ with $(A', B') = \mathcal{P}_{M,C}(A, B)$ and which can be computed without any call to \mathcal{P} . In the following, $\text{Im}(\mathcal{R})$ denotes the set of all tuples (M, A, B, C, A', B') satisfying the predicate \mathcal{R} . As discussed above, we assume that we are given efficient subroutines comprising

- a sampling algorithm $\text{ForSamp}_{\mathcal{R}}$ which samples all possible outputs A' such that $(M, A, B, C, A', B') \in \text{Im}(\mathcal{R})$ over a random choice of $\mathcal{P} \leftarrow \text{PERM}[\mathcal{R}, M, A, B, C, B']$, for any inputs (M, A, B, C, B') ;
- a sampling algorithm $\text{BacSamp}_{\mathcal{R}}$ which samples all possible inputs (A, B) such that $(M, A, B, C, A', B') \in \text{Im}(\mathcal{R})$ over a random choice of $\mathcal{P} \leftarrow \text{PERM}[\mathcal{R}]$ for any inputs (M, A', B', C) .

We then consider the second simulator \mathcal{S} for \mathcal{P} and \mathcal{P}^{-1} as depicted on Fig. 3. The new simulator is similar to the one given in the plain ideal cipher model, excepted that we apply the three following modifications:

- Line 3(d) in the simulation of \mathcal{P} , A' is sampled by $\text{ForSamp}_{\mathcal{R}}$ instead of taking a uniformly random ℓ_a -bit string;
- Line 4(b) in the simulation of \mathcal{P} , A' is also sampled by $\text{ForSamp}_{\mathcal{R}}$;
- Line 3 in the simulation of \mathcal{P}^{-1} , (A', B') are sampled by $\text{BacSamp}_{\mathcal{R}}$.

These modifications lead to the new simulator defined on Figure 4.

Using the sampling algorithms to get A' or (A', B') modifies the probability that an adversary succeeds in making our simulator abort during the game. Even though the above Theorem is still applicable, the probabilities of events Abort_i for $i \in [1, 5]$ must be analyzed with greater care here. It is easily seen that the inequalities $\Pr[\text{Abort}_i] \leq \frac{N(N-1)}{2} \cdot p_i$ still hold; however the bounds p_i must be recomputed. We now reevaluate these one by one.

New probability bound p_1 . Remind that we fixed $\tilde{x} = (\tilde{M}, \tilde{A}, \tilde{B}, \tilde{C}) \in \mathcal{X}$ as well as $y = (M, A, B, C) \in \mathcal{X}$. We now consider the distribution of outputs x of $\mathcal{P}(y)$ as generated by the new simulator

$$D(y) = \{x = (M, A', B', C) \mid B' \leftarrow \{0, 1\}^{\ell_m}, A' \leftarrow \text{ForSamp}_{\mathcal{R}}(M, A, B, C, B')\}.$$

Then, taking probabilities over the uniformly random selection $x \leftarrow D(y)$ we connect Abort_1 to

$$\begin{aligned} p_1(\tilde{x}, y) &= \Pr \left[\exists \mathbf{m}, \tilde{\mathbf{m}} \in \{0, 1\}^{\ell_m} : \text{Insert}[\mathbf{m}](x) = \text{Insert}[\tilde{\mathbf{m}}](\tilde{x}) \right] \\ &= \Pr \left[\begin{array}{l} \exists \mathbf{m}, \tilde{\mathbf{m}} \in \{0, 1\}^{\ell_m} : \\ \mathbf{m} = \tilde{\mathbf{m}} \\ A' = \tilde{A} \\ B' = \tilde{B} \\ C \boxplus M \boxplus \mathbf{m} = \tilde{C} \boxplus \tilde{M} \boxplus \tilde{\mathbf{m}} \end{array} \right] \\ &= \Pr [A' = \tilde{A}] \cdot \Pr [B' = \tilde{B}] \cdot \delta [C \boxplus M = \tilde{C} \boxplus \tilde{M}] \end{aligned}$$

We then upper bound $p_1(\tilde{x}, y)$ by

$$p_1 = \max_{\tilde{A}, \tilde{B}} \Pr [A' = \tilde{A}] \cdot \Pr [B' = \tilde{B}] \leq 2^{-(\ell_a - \tau)} \cdot 2^{-\ell_m}.$$

New probability bound p_2 . We now fix $\tilde{y} = (\tilde{M}, \tilde{A}, \tilde{B}, \tilde{C}) \in \mathcal{X}$ as well as $y = (M, A, B, C) \in \mathcal{X}$. Taking probabilities over the selection $x \leftarrow D(y)$, expressing Abort_2 boils down to evaluating

$$\begin{aligned} p_2(\tilde{y}, y) &= \Pr \left[\exists \mathbf{m} \in \{0, 1\}^{\ell_m} : \text{Insert}[\mathbf{m}](x) = \tilde{y} \right] \\ &= \Pr \left[\begin{array}{l} \exists \mathbf{m} \in \{0, 1\}^{\ell_m} : \\ \mathbf{m} = \tilde{M} \\ A' = \tilde{A} \\ C \boxplus M \boxplus \mathbf{m} = \tilde{B} \\ B' = \tilde{C} \end{array} \right] \\ &= \Pr [A' = \tilde{A}] \cdot \Pr [B' = \tilde{C}] \cdot \delta [C \boxplus M \boxplus \tilde{M} = \tilde{B}] \end{aligned}$$

An upper bound for $p_2(\tilde{y}, y)$ is then

$$p_2 = \max_{\tilde{A}, \tilde{C}} \Pr [A' = \tilde{A}] \cdot \Pr [B' = \tilde{C}] \leq 2^{-(\ell_a - \tau)} \cdot 2^{-\ell_m}.$$

Initialization of \mathcal{S}	
No input, no output	
1. set $X = Y = Z = \emptyset$	

Simulation of \mathcal{P}	
Input: $y = (M, A, B, C) \in \mathcal{X}$	
Output: (A', B')	
1. add node y to Y	
2. if there exists an edge $y \xrightarrow{\mathcal{P}} x \in Z$	
(a) return (A', B') where $x = (M, A', B', C)$	
3. if y has a path μ in graph \mathcal{G}	
(a) compute $\mathcal{M} = \text{unpad}(\mu)$	
(b) call \mathcal{H} to get $h = \mathcal{H}(\mathcal{M})$	
(c) set $B' = h$	
(d) run $\text{ForSamp}_{\mathcal{R}}(M, A, B, C, B')$ to get A'	
4. else	
(a) randomly select $B' \leftarrow \{0, 1\}^{\ell_m}$	
(b) run $\text{ForSamp}_{\mathcal{R}}(M, A, B, C, B')$ to get A'	
5. add node $x = (M, A', B', C)$ to X and edge $y \xrightarrow{\mathcal{P}} x$ to Z	
6. if for some $\tilde{x} \in X$ one has $\text{Insert}[\tilde{M}](\tilde{x}) = \text{Insert}[M](x)$ for some M and \tilde{M} (event Abort_1), then abort.	
7. if x admits a path in \mathcal{G} and $\exists \tilde{y} \in Y$ such that $\text{Insert}[M](x) = \tilde{y}$ for some M (event Abort_2), then abort.	
8. return (A', B')	

Simulation of \mathcal{P}^{-1}	
Input: $x = (M, A, B, C) \in \mathcal{X}$	
Output: (A', B')	
1. add node x to X	
2. if there exists an edge $y \xrightarrow{\mathcal{P}} x \in Z$	
(a) return (A', B') where $y = (M, A', B', C)$	
3. run $\text{BacSamp}_{\mathcal{R}}(M, A, B, C)$ to get (A', B')	
4. add node $y = (M, A', B', C)$ to X and edge $y \xrightarrow{\mathcal{P}} x$ to Z	
5. if $\exists \tilde{x} \in X$, $\tilde{x} \neq x$ such that $y \xrightarrow{\mathcal{P}} \tilde{x} \in Z$ (event Abort_4), then abort	
6. if $\exists \tilde{x} \in X$ such that $\text{Insert}[M](\tilde{x}) = y$ for some M (event Abort_5), then abort	
7. return (A', B')	

Figure 4: New simulator \mathcal{S} for \mathcal{P} and \mathcal{P}^{-1} in the "biased" ideal cipher model. The simulator makes calls to an internal subroutine $\text{ForSamp}_{\mathcal{R}}$ that samples the relation \mathcal{R} with a certain bias $\tau \geq 0$. When $\tau = 0$, the sampling algorithm is reduced to a uniform selection over ℓ_a -bit strings and we recover the original simulator.

New probability bound p_4 . We now fix $\tilde{x} = (\tilde{M}, \tilde{A}, \tilde{B}, \tilde{C}) \in \mathcal{X}$ and $y = (M, A, B, C) \in \mathcal{X}$. Taking probabilities over $x \leftarrow D(y)$ we get

$$\begin{aligned} p_4(\tilde{y}, y) &= \Pr[x = \tilde{x}] \\ &= \Pr \left[\begin{array}{l} M = \tilde{M} \\ A' = \tilde{A} \\ B' = \tilde{B} \\ C = \tilde{C} \end{array} \right] \\ &= \Pr[A' = \tilde{A}] \cdot \Pr[B' = \tilde{B}] \cdot \delta[M = \tilde{M} \wedge C = \tilde{C}] \end{aligned}$$

which gives an upper bound for $p_4(\tilde{y}, y)$ as

$$p_4 = \max_{\tilde{A}, \tilde{B}} \Pr[A' = \tilde{A}] \cdot \Pr[B' = \tilde{B}] \leq 2^{-(\ell_a - \tau)} \cdot 2^{-\ell_m}.$$

New probability bound p_5 . Here we fix $\tilde{x} = (\tilde{M}, \tilde{A}, \tilde{B}, \tilde{C}) \in \mathcal{X}$ and $x = (M, A, B, C) \in \mathcal{X}$ and consider the following probability taken over the random selection $y \leftarrow D(x)$:

$$\begin{aligned} p_5(\tilde{y}, x) &= \Pr[\exists \mathbf{m} \in \{0, 1\}^{\ell_m} : \text{Insert}[\mathbf{m}](\tilde{x}) = y] \\ &= \Pr \left[\begin{array}{l} \exists \mathbf{m} \in \{0, 1\}^{\ell_m} : \\ \tilde{C} \boxplus \tilde{M} \boxplus \mathbf{m} = B' \\ \tilde{B} = C \end{array} \right] \\ &= \Pr[(A', B') = (\tilde{A}, \tilde{C} \boxplus \tilde{M} \boxplus M)] \cdot \delta[\tilde{B} = C]. \end{aligned}$$

Then we bound $p_5(\tilde{y}, x)$ by

$$p_5 = \max_{\tilde{A}, \tilde{C}, \tilde{M}, M} \Pr[(A', B') = (\tilde{A}, \tilde{C} \boxplus \tilde{M} \boxplus M)] \leq 2^{-(\ell_a + \ell_m - \lambda)}.$$

Wrapping it up. Putting it altogether, we finally get the following security bound.

Theorem 2. *Assume that the keyed permutation \mathcal{P} is taken uniformly at random in the space $\text{PERM}[\mathcal{R}]$ of all keyed permutations which observe a certain Boolean relation \mathcal{R} holding with probability one on their input, parameter and output. Assume further that \mathcal{R} has a forward bias $\tau \in (0, \ell_a)$ and backward bias $\lambda \in (0, \ell_a + \ell_m)$. Then the (simplified) mode of operation of *Shabal* is indistinguishable from a random oracle. More precisely,*

$$\text{Adv}(\mathcal{D}, \mathcal{R}) \leq \frac{3N(N-1)}{2} 2^{-(\ell_a + \ell_m - \tau)} + \frac{N(N-1)}{2} 2^{-(\ell_a + \ell_m - \lambda)} + N 2^{-(\ell_a + \ell_m)}.$$

We see that one of the two terms $N^2 2^{-(\ell_a + \ell_m - \tau)}$ and $N^2 2^{-(\ell_a + \ell_m - \lambda)}$ must dominate the adversary's advantage when $\tau, \lambda > 0$. If one assumes to always have $\lambda < \tau$, the security bound (in bits) decreases linearly as τ ranges from 0 to ℓ_a . Ultimately when $\tau = \ell_a$, we reach the limit of $N = O(2^{\frac{\ell_m}{2}})$ adversarial observations and the mode of operation of *Shabal* has no security margin anymore (but remains ideal). Conversely if $\lambda > \tau$ then the term $N^2 2^{-(\ell_a + \ell_m - \lambda)}$ dominates and the mode of operation does not behave ideally anymore if $\ell_a + \ell_m - \lambda \leq \ell_h$ (recall that *Shabal* truncates the B -part of the last internal state to its ℓ_h leftmost bits). We therefore confirm that distinguishing relations can indeed be used to break *Shabal*, but at the condition that

$$\lambda > \ell_a + \ell_m - \ell_h = 384$$

in the most favorable case ($\ell_h = 512$). We leave it as an open challenge to come up with a related-key distinguisher for $p = 3$ that features a backward bias at least equal to 384.

References

- [1] J.-P. Aumasson. On the pseudorandomness of Shabal’s keyed permutation. Public comment on the NIST Hash competition, 2009.
- [2] J.-P. Aumasson, A. Mashatan, and W. Meier. More on the pseudorandomness of Shabal’s permutation. Public comment on the NIST Hash competition, 2009.
- [3] E. Bresson, A. Canteaut, B. Chevallier-Mames, C. Clavier, T. Fuhr, A. Gouget, T. Icart, J.-F. Misarsky, M. Naya-Plasencia, P. Paillier, T. Pornin, J.-R. Reinhard, C. Thuillet, and M. Videau. Shabal, a submission to NIST’s cryptographic hash algorithm competition. Submission to the NIST Hash competition, 2008.
- [4] L. Knudsen, K. Matusiewicz, and S.S. Thomsen. Observations on the Shabal keyed permutation. Public comment on the NIST Hash competition, 2009.
- [5] U. Maurer, R. Renner, and C. Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In *Theory of cryptography – TCC 2004*, volume 2951 of *LNCS*, pages 21–39. Springer, 2004.
- [6] M. Stam. Blockcipher based hashing revisited. In *Fast Software Encryption - FSE 2009*, Lecture Notes in Computer Science. Springer, 2009. To appear.